SEGClobber - A Linear Clobber Solver

Taylor Folkersen, Zahra Bashir, Fatemeh Tavakoli, Martin Müller

University of Alberta

Linear Clobber

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.
- Black/White stones, empty spaces: ■■□_□

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.
- Black/White stones, empty spaces: ■■□_□
- Black can move to: ■_■_□

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.
- Black/White stones, empty spaces: ■■□_□
- Black can move to: ■_■_□
- White can move to: ■□__□

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.
- Black/White stones, empty spaces: ■■□_□
- Black can move to: ■_■_□
- White can move to: ■□__□
- Must clobber an opponent stone! Illegal White move: ■■□□_

Linear Clobber

- Clobber invented in 2001 by Michael Albert et al.
- Black/White stones, empty spaces: ■■□_□
- Black can move to: ■_■_□
- White can move to: ■□__□
- Must clobber an opponent stone! Illegal White move: ■■□□_
- Breaks into subgames frequently: ■□■□_■■□ = ■□■□ + ■■□

Conjectures

Conjectures

- Notation: (■□)^3 = ■□■□■□

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3
    - Verified for n <= 19

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3
    - Verified for n <= 19
    - We verify for n <= 50

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3
    - Verified for n <= 19
    - We verify for n <= 50
  - (■■□)^n = ⌊(n + 1)/2⌋·↑

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3
    - Verified for n <= 19
    - We verify for n <= 50
  - (■■□)^n = ⌊(n + 1)/2⌋·↑
    - Verified for n <= 17
    - We verify for n <= 28

Conjectures

- Notation: (■□)^3 = ■□■□■□
- Albert et al. made 2 conjectures about linear Clobber:
  - (■□)^n is a first player win for n != 3
    - Verified for n <= 19
    - We verify for n <= 50
  - (■■□)^n = $\lfloor (n + 1)/2 \rfloor \cdot \uparrow$
    - Verified for n <= 17
    - We verify for n <= 28
- Chen et al. recently proposed a proof of (■□)^n conjecture

Transposition Table

Transposition Table

- 64 bit Zobrist hashes

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity
    - Replacement policy

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity
  - Replacement policy
- Contents of 1 entry:

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity
  - Replacement policy
- Contents of 1 entry:
  - Win/loss

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity
  - Replacement policy
- Contents of 1 entry:
  - Win/loss
  - Heuristic value, “best” move

Transposition Table

- 64 bit Zobrist hashes
- 4 way set associativity
  - Replacement policy
- Contents of 1 entry:
  - Win/loss
  - Heuristic value, “best” move
  - Age, depth (for replacement policy)

Database (DB)

Database (DB)

- Represents all games up to length 16

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)

Database (DB)

- Represents all games up to length 16
    - All 3^16 = 43,046,721 games
    - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
    - Outcome class (L, R, P, N)
        - > 0, < 0, $= 0$, $\not\gtrless 0$

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\not\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player
    - Among equal moves, only the “simplest” option is kept

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless\!\!\!/\ 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player
    - Among equal moves, only the “simplest” option is kept
  - Complexity score

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player
    - Among equal moves, only the “simplest” option is kept
  - Complexity score
  - Simplest (nondominated) move

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player
    - Among equal moves, only the “simplest” option is kept
  - Complexity score
  - Simplest (nondominated) move
  - Link to SEG (Simplest Equal Game)

Database (DB)

- Represents all games up to length 16
  - All 3^16 = 43,046,721 games
  - Uses only 866,924 entries (~2% of all boards)
- Entry contents:
  - Outcome class (L, R, P, N)
    - > 0, < 0, $= 0$, $\gtrless 0$
  - Lower/upper bounds along 2 scales: $[-31\uparrow, 31\uparrow]$ and $[-31\uparrow\star, 31\uparrow\star]$
  - Nondominated moves for each player
    - Among equal moves, only the “simplest” option is kept
  - Complexity score
  - Simplest (nondominated) move
  - Link to SEG (Simplest Equal Game)
  - The board itself

Normal Form

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
   - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
    - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
    - (4, 3, 3, 2)

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
   - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
   - (4, 3, 3, 2)
   - Relaxed normal form

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
    - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
    - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
    - (4, 3, 3, 2)
    - Relaxed normal form
4. Flip subgames

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
   - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
   - (4, 3, 3, 2)
   - Relaxed normal form
4. Flip subgames
   - ■ □ ■ ■ _ ■ □ ■ _ □ ■ ■ _ □ ■

Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
   - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
   - (4, 3, 3, 2)
   - Relaxed normal form
4. Flip subgames
   - ■ □ ■ ■ _ ■ □ ■ _ □ ■ ■ _ □ ■
5. Sort equal length subgames in decreasing order

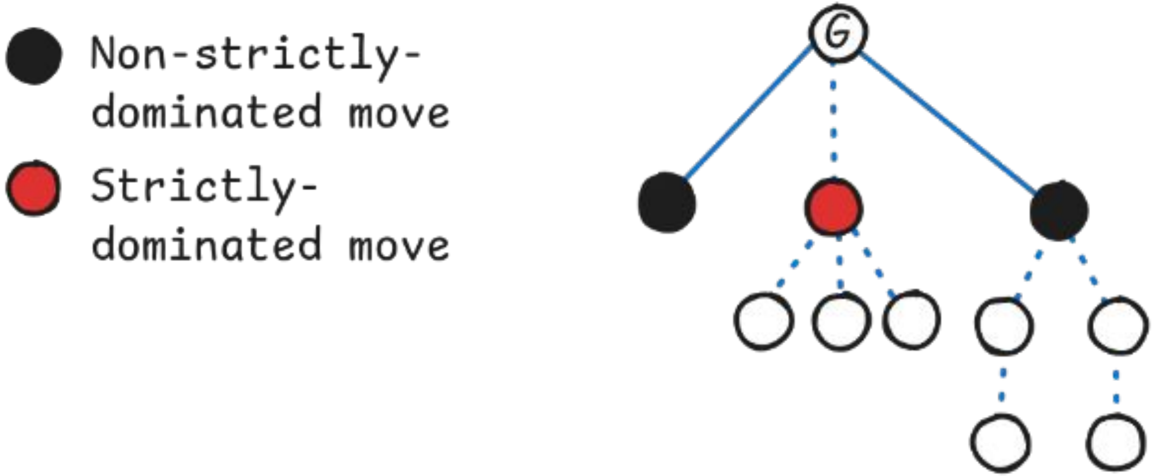
Normal Form

- ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ □ _ _ _ ■ □ _
- Shape: (3, 4, 3, 1, 2)

1. Remove subgames of length 1
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ _ _ _ _ _ ■ □ _
2. Omit redundant empty squares
   - ■ □ ■ _ ■ ■ □ ■ _ □ ■ ■ _ ■ □
3. Sort subgames so that the shape is in decreasing order
   - ■ ■ □ ■ _ ■ □ ■ _ □ ■ ■ _ ■ □
   - (4, 3, 3, 2)
   - Relaxed normal form
4. Flip subgames
   - ■ □ ■ ■ _ ■ □ ■ _ □ ■ ■ _ □ ■
5. Sort equal length subgames in decreasing order
   - ■ □ ■ ■ _ □ ■ ■ _ ■ □ ■ _ □ ■

Complexity Score

Complexity Score

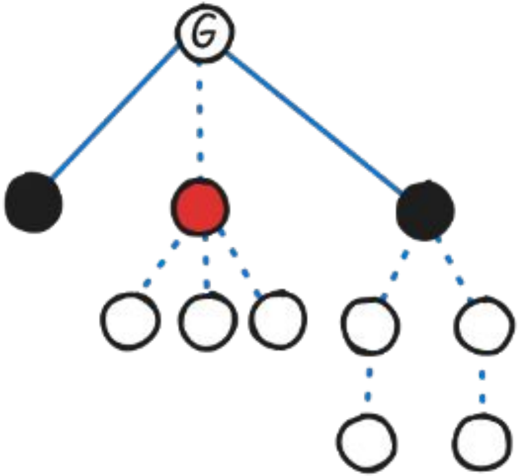


CS3(G) = 2

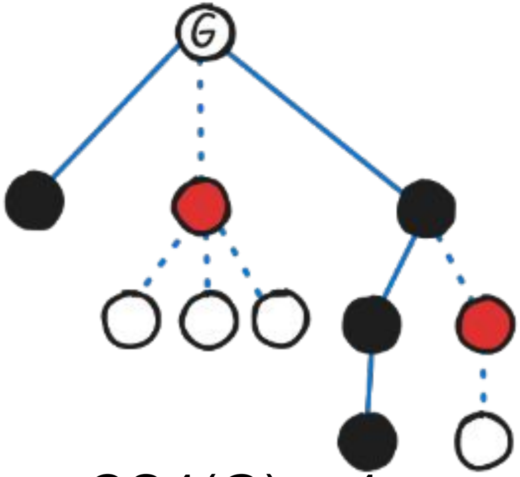
Complexity Score

Non-strictly-dominated move

Strictly-dominated move

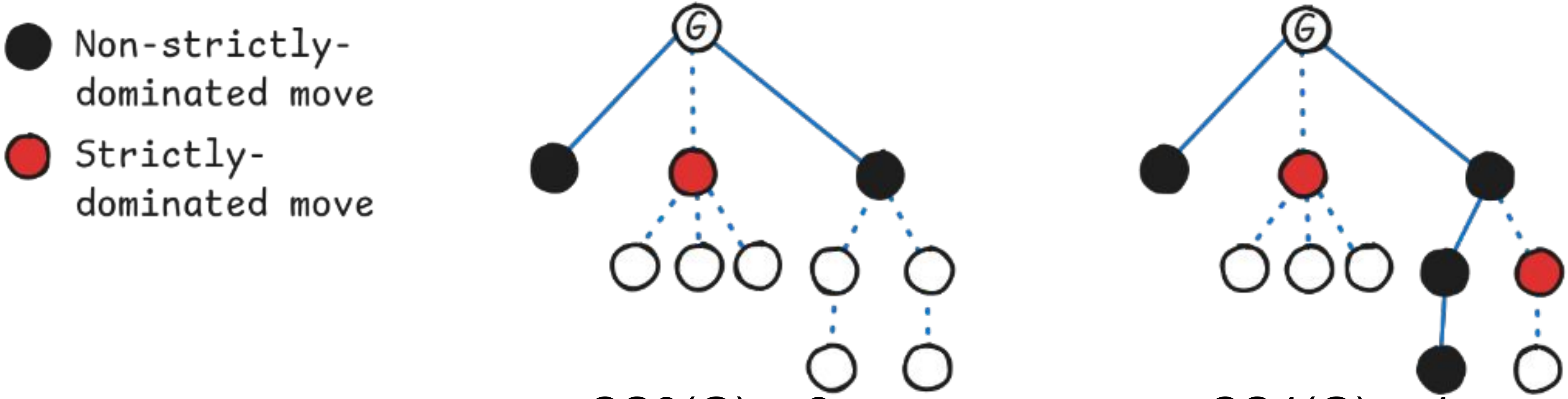


CS3(G) = 2



CS4(G) = 4

Complexity Score

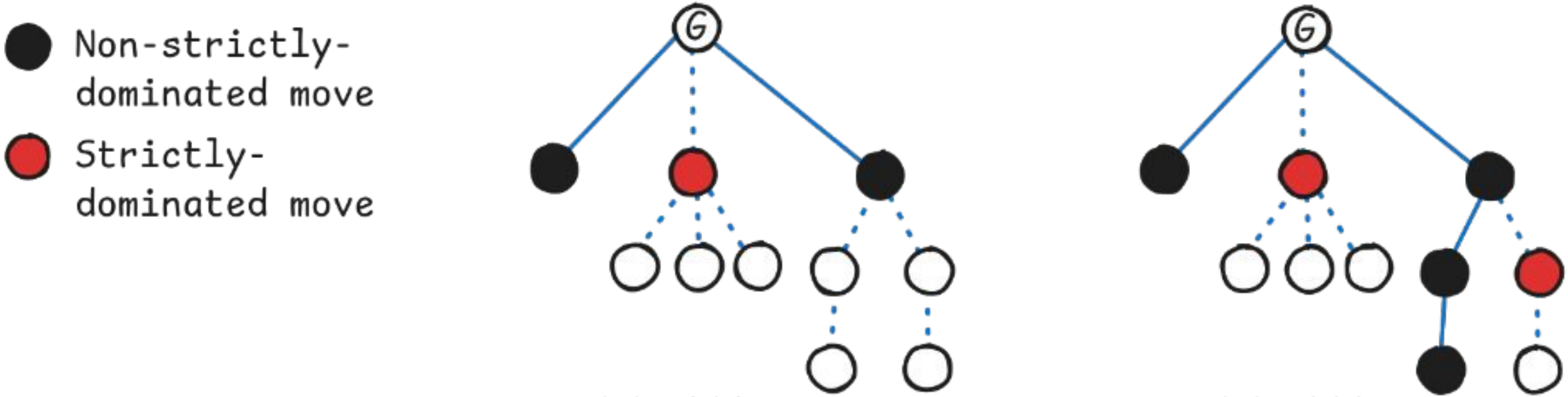


CS3(G) = 2

CS4(G) = 4

- G1 may replace G2 during search if:

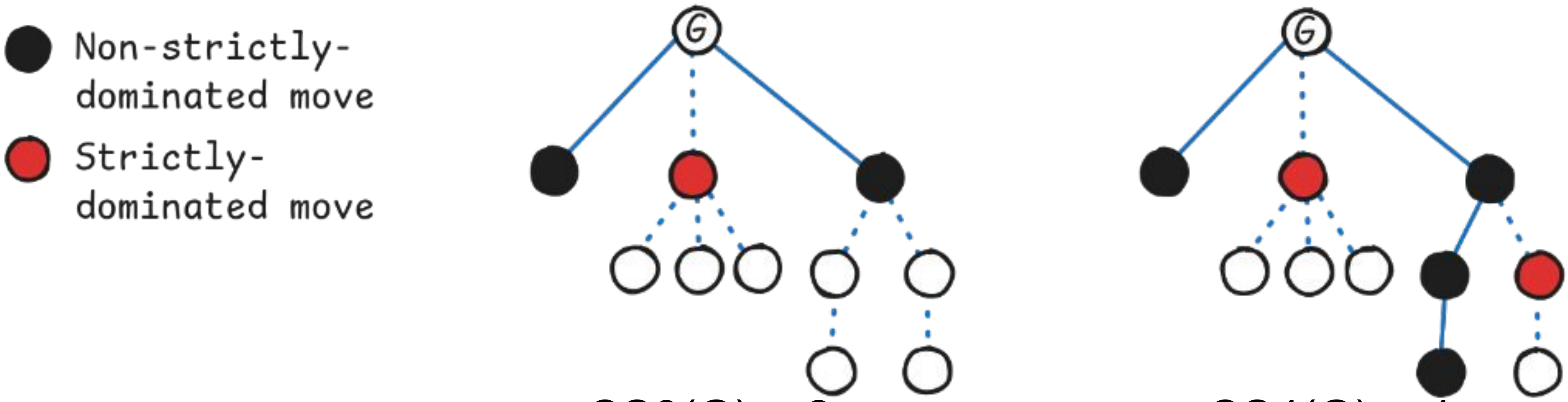
Complexity Score



CS3(G) = 2 CS4(G) = 4

- G1 may replace G2 during search if:
    - G1 is **simpler** than G2: CS(G1) < CS(G2)

Complexity Score

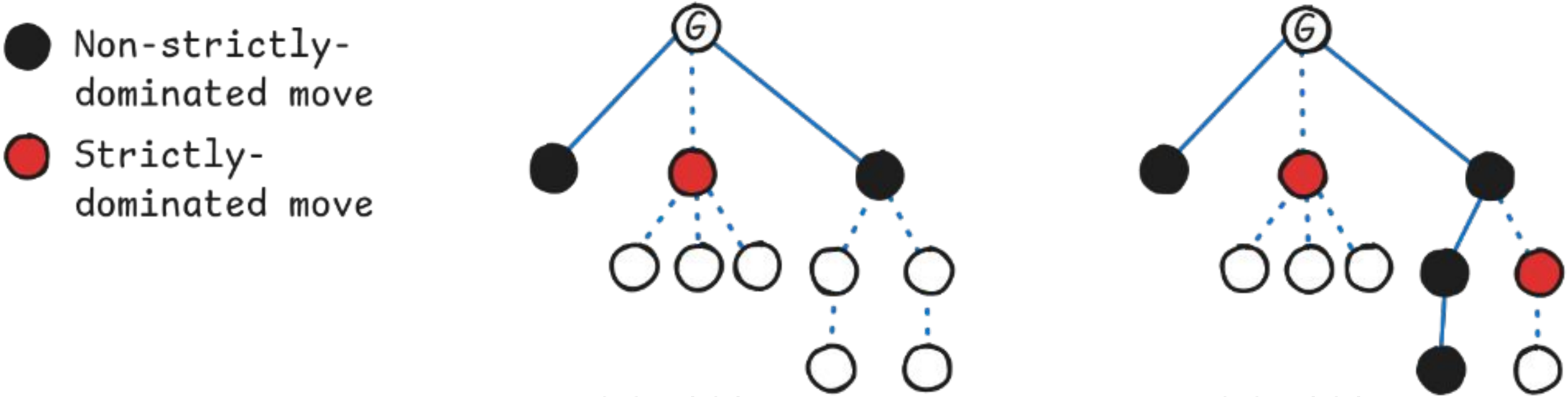


CS3(G) = 2

CS4(G) = 4

- G1 may replace G2 during search if:
  - G1 is **simpler** than G2: CS(G1) < CS(G2)
  - **AND** G1 comes **before** G2 in the database

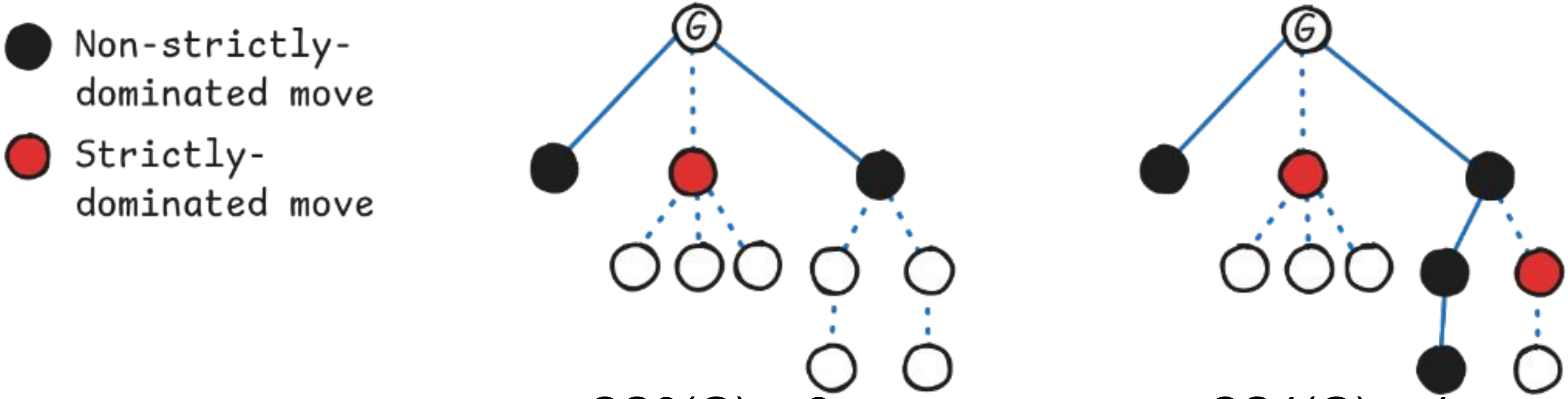
Complexity Score



CS3(G) = 2 CS4(G) = 4

- G1 may replace G2 during search if:
  - G1 is **simpler** than G2: CS(G1) < CS(G2)
  - **AND** G1 comes **before** G2 in the database
    - To prevent cycles

Complexity Score



CS3(G) = 2

CS4(G) = 4

- G1 may replace G2 during search if:
  - G1 is **simpler** than G2: CS(G1) < CS(G2)
  - **AND** G1 comes **before** G2 in the database
    - To prevent cycles
  - A database entry contains the **simplest** such game, if it exists

Sum Simplification

Sum Simplification

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

Sum Simplification

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

Top row not in DB

Sum Simplification

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

Top row not in DB

2. Using DB, delete single P positions (= 0)

Sum Simplification

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

P position

Top row not in DB

2. Using DB, delete single P positions (= 0)

Sum Simplification

Top row not in DB

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

P position

2. Using DB, delete single P positions (= 0)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

Sum Simplification

Top row not in DB

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■■□□□□■□■

P position

2. Using DB, delete single P positions (= 0)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■■□□□□■□■

3. Sort remaining subgames in order of decreasing length

Sum Simplification

Top row not in DB

1. Set aside large subgames (those not in DB)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■□■□■□_■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

P position

2. Using DB, delete single P positions (= 0)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■□■□■□□□□□_■■□□■_■□■_■■■■■□_■■□■■■□□□□■□■

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

4. Replace sums of 3+ (using sliding window of length 16)

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

5. Replace pairs

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□□■□■■■■

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

■■□■■■□□□□■□■_■□□■□■■■■

6. Replace singles

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□□■□■■■■

6. Replace singles

■■□■■■□□□□■□■ = □□□■□■□

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□□■□■■■■

6. Replace singles

■■□■■■□□□□■□■ = □□□■□■□

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
□□□■□■□_■□□■□■■■■

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_■■■■■□_■■□□■_■□■

CS4: 5409 -> 8
■■■■■□_■■□□■_■□■ = □■■■■

4. Replace sums of 3+ (using sliding window of length 16)

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□■□■□□□□□_□■■■■

CS4: 52434 -> 80
■□■□■□□□□□_□■■■■ = ■□□■□■■■■

5. Replace pairs

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
■■□■■■□□□□■□■_■□□■□■■■■

CS4: 32747 -> 16
■■□■■■□□□□■□■ = □□□■□■□

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■
□□□■□■□_■□□■□■■■■

Sum Simplification (Continued)

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

Sum Simplification (Continued)

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

7. Now also consider previously set aside games.

Sum Simplification (Continued)

Resume considering top row

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

7. Now also consider previously set aside games.

Sum Simplification (Continued)

Resume
considering
top row

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

Sum Simplification (Continued)

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

Resume considering top row

No moves

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

Sum Simplification (Continued)

Resume considering top row

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

g + (-g) = 0

No moves

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

Sum Simplification (Continued)

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

g + (-g) = 0

Resume considering top row

No moves

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

□□□■□■□_■□□■□■■■■

Sum Simplification (Continued)

Resume considering top row

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

g + (-g) = 0

No moves

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

□□□■□■□_■□□■□■■■■

9. Convert to normal form

Sum Simplification (Continued)

Resume
considering
top row

6. Replace singles

■■■■■■■■■■■■■■■■□_□□□□□□□□□□□□□□□□■_■■■■■■■■■■■■■■■■■

□□□■□■□_■□□■□■■■■

g + (-g) = 0

No moves

7. Now also consider previously set aside games.

8. Delete subgames with no moves, and pairs of visually identifiable inverses

□□□■□■□_■□□■□■■■■

9. Convert to normal form

■□□■□■■■■_□□□■□■□

Search Algorithm

Search Algorithm

1. Simplify board

Search Algorithm

1. Simplify board
2. Check database, then transposition table

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
   - For subgame Gi with certain known outcome classes, try to solve G - Gi instead

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
    - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
   - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves
   - Move ordering informed by:

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
    - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves
    - Move ordering informed by:
        - Outcome classes

Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
    - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves
    - Move ordering informed by:
        - Outcome classes
        - DB “simplest moves”

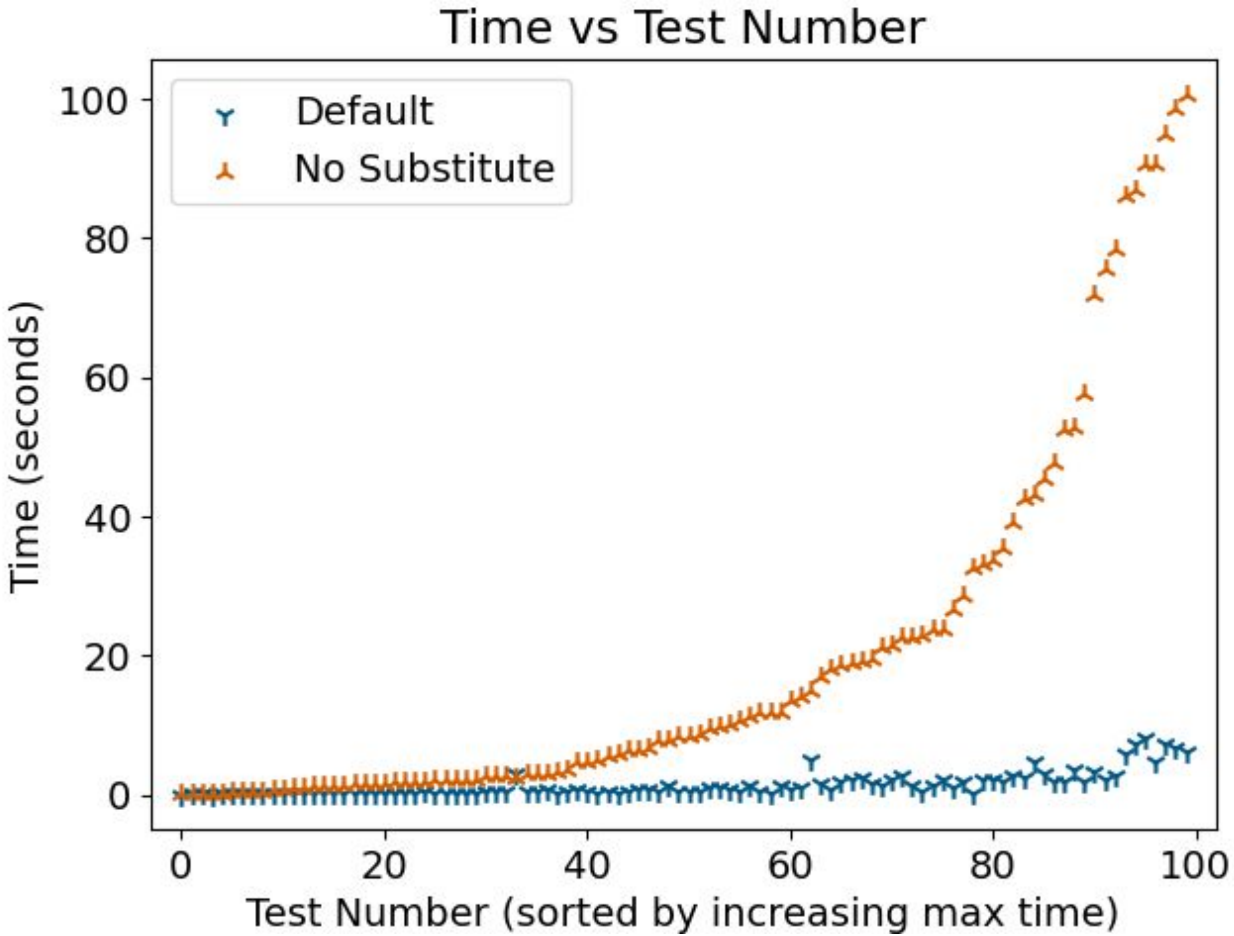
Search Algorithm

1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
    - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves
    - Move ordering informed by:
        - Outcome classes
        - DB “simplest moves”
        - Heuristic “best moves”

Search Algorithm

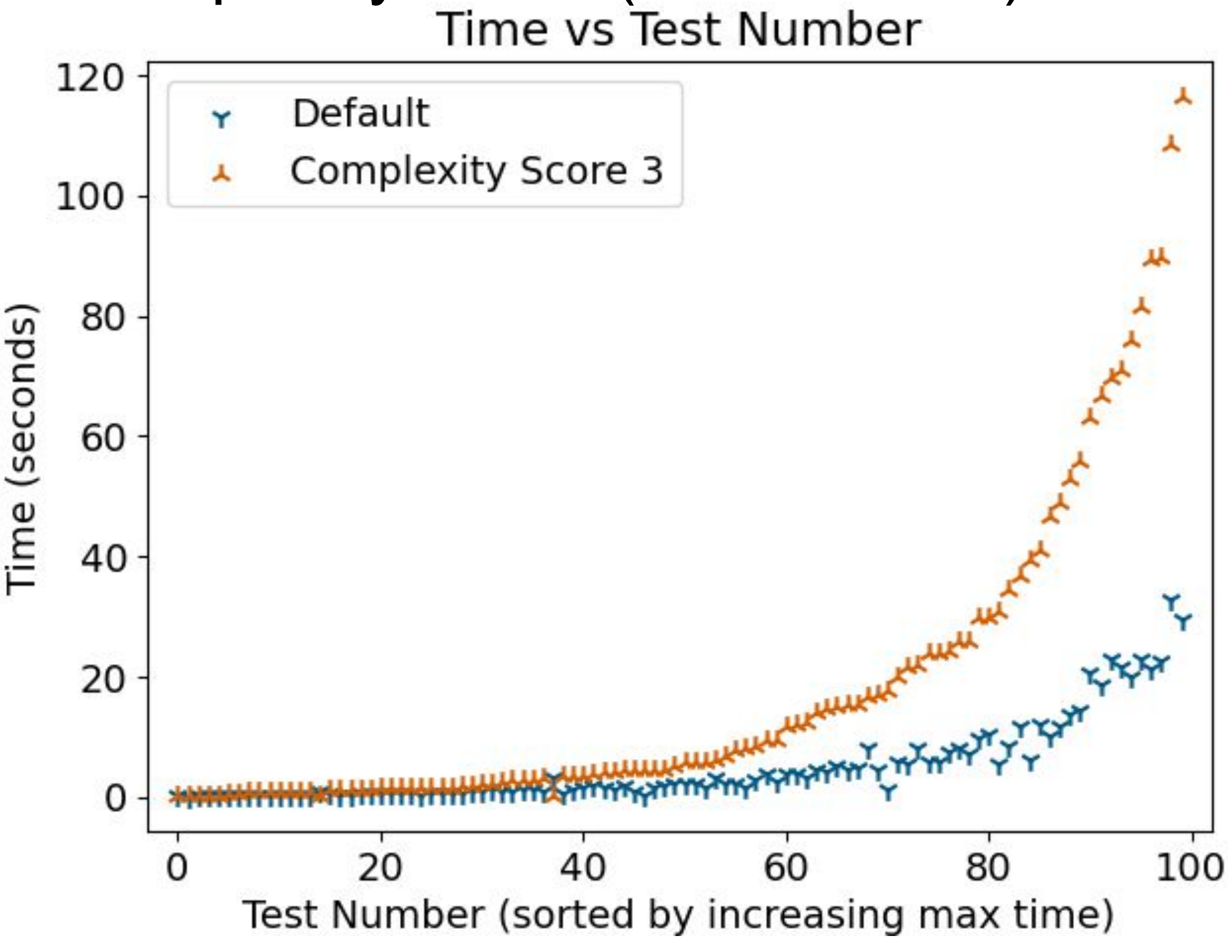
1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
    - For subgame Gi with certain known outcome classes, try to solve G - Gi instead
5. Generate and play moves
    - Move ordering informed by:
        - Outcome classes
        - DB “simplest moves”
        - Heuristic “best moves”
        - Middle of subgames

Ablation: Substitution



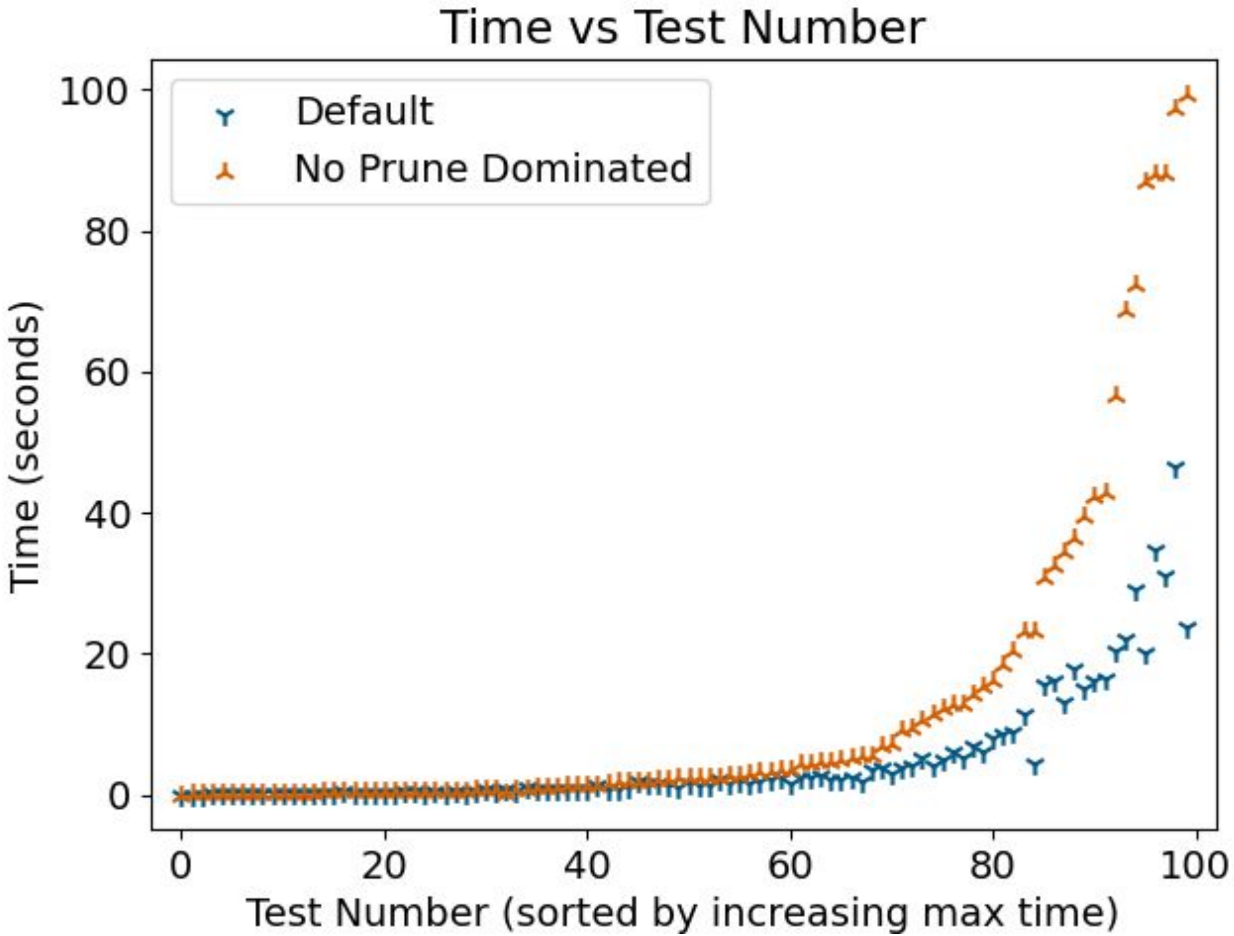
length=58

Ablation: Complexity Score (CS4 vs CS3)



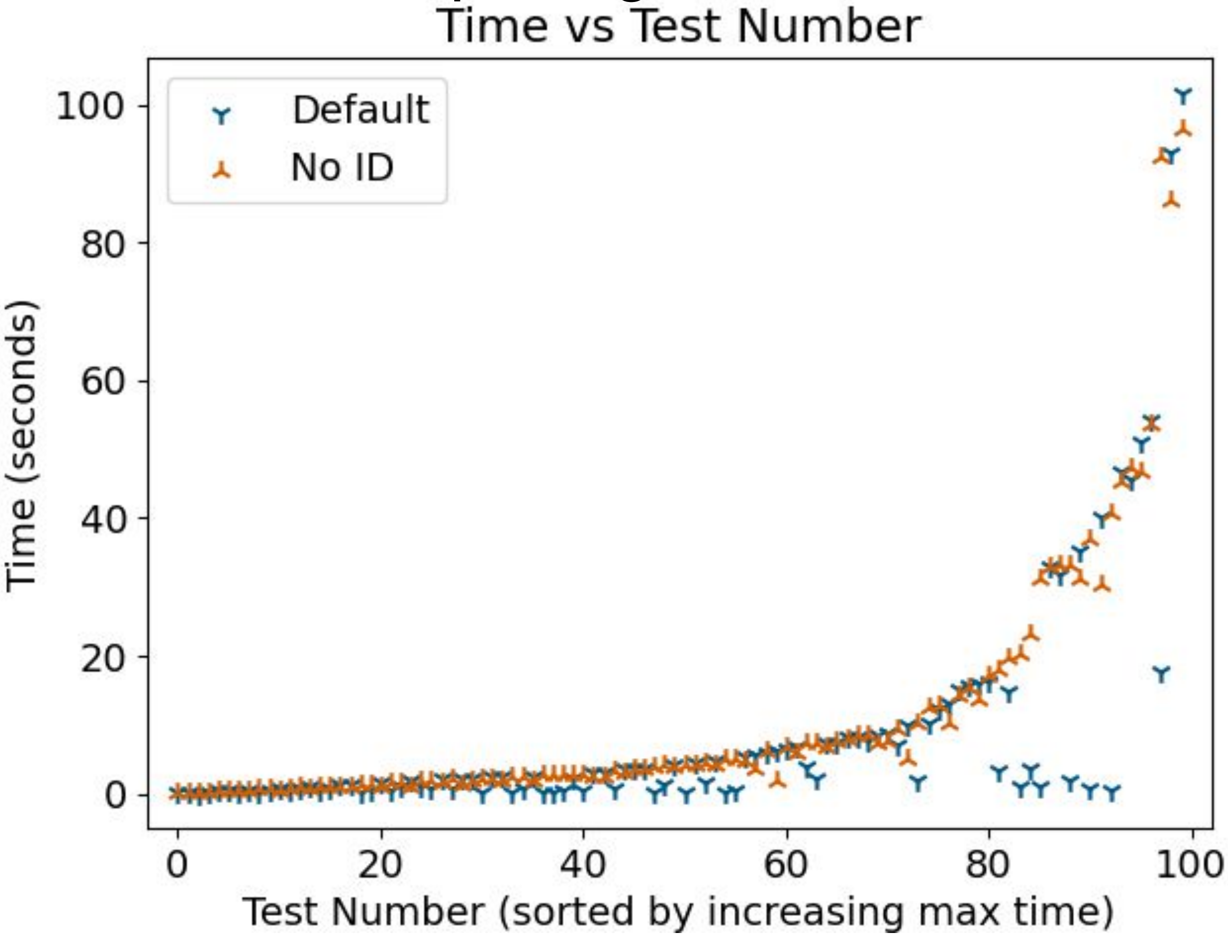
length=[60,66]

Ablation: Prune Dominated Moves



length=[60,66]

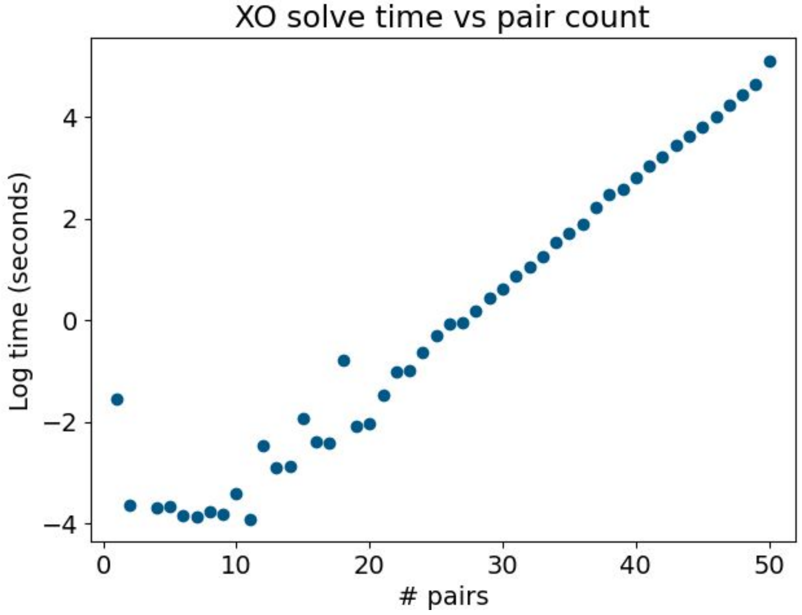
Ablation: Iterative Deepening



length=[60,66]

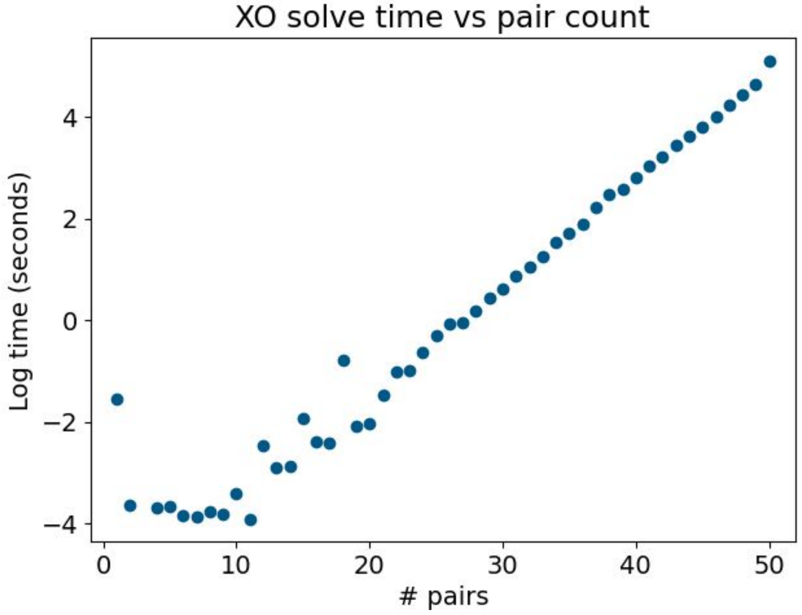
Verifying (■□)^n Conjecture

Verifying (■□)^n Conjecture



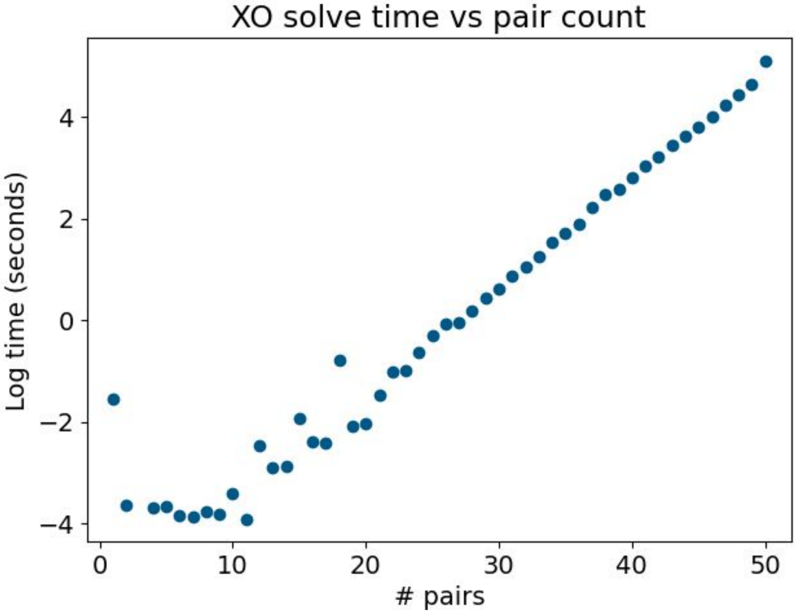
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs



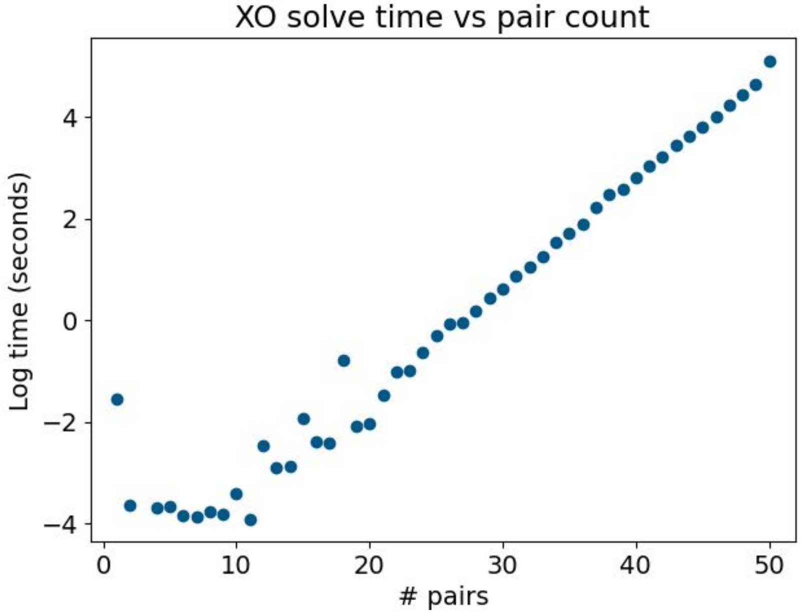
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries



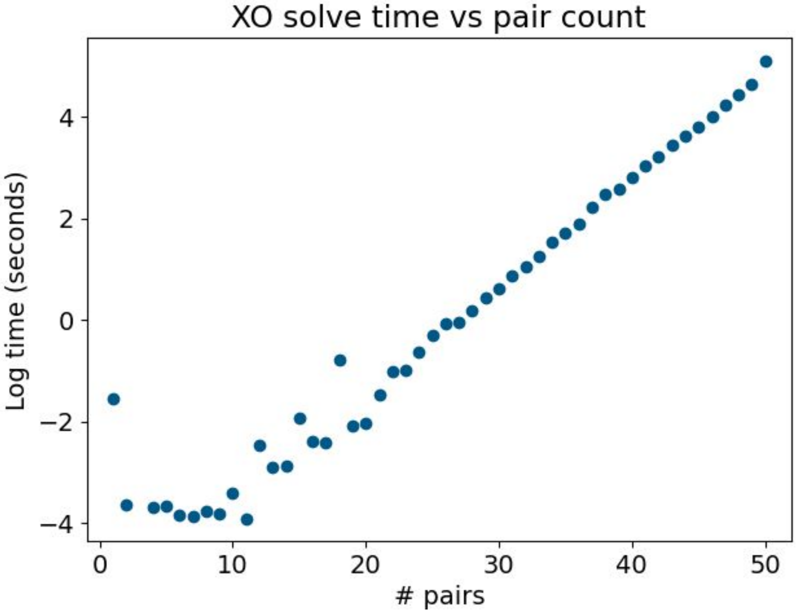
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right



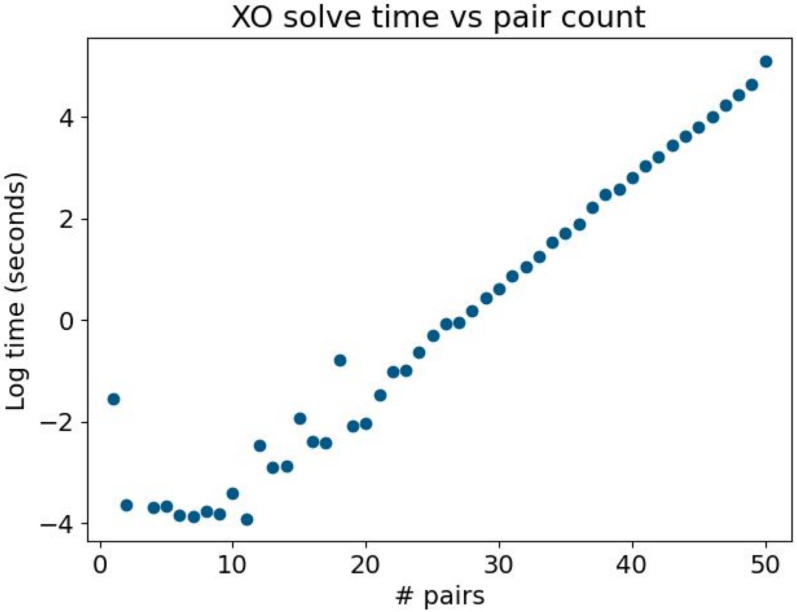
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time



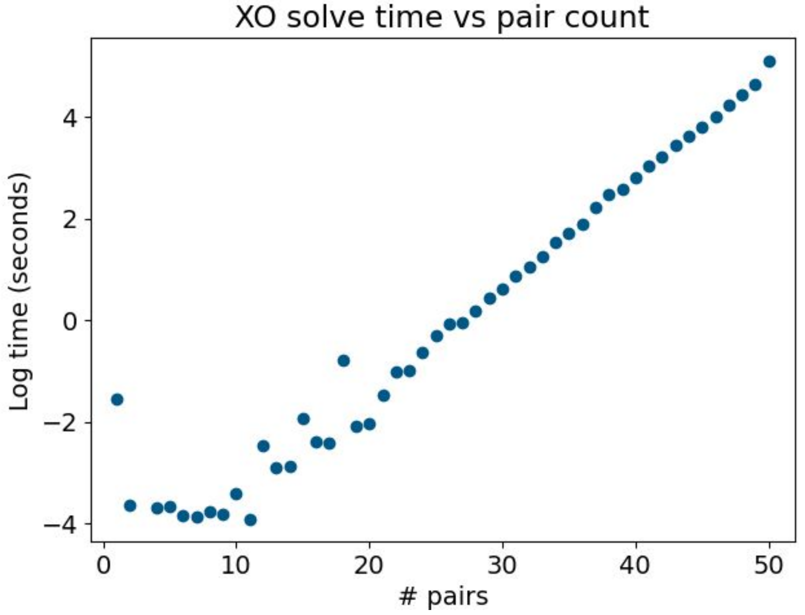
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black



Verifying (■□)^n Conjecture

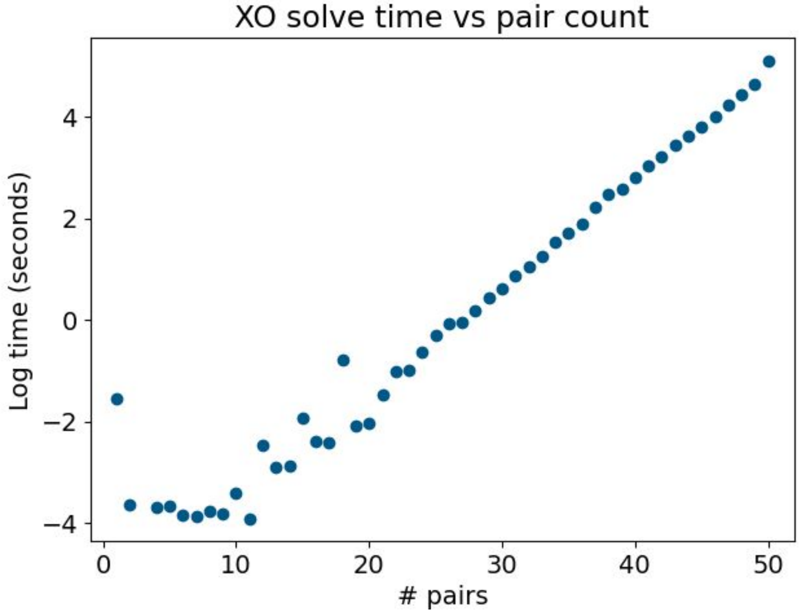
- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black



(■□)^19 = previous record

Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black

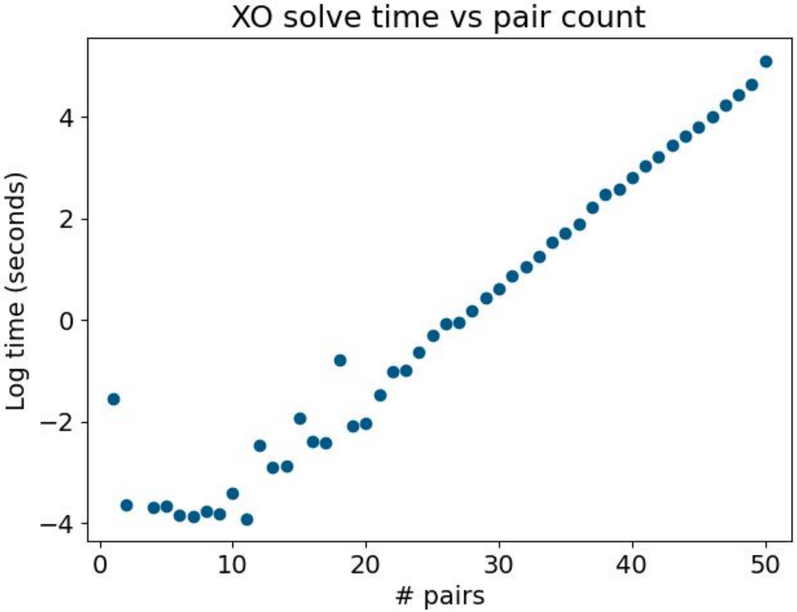


(■□)^19 = previous record

■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□

Verifying (■□)^n Conjecture

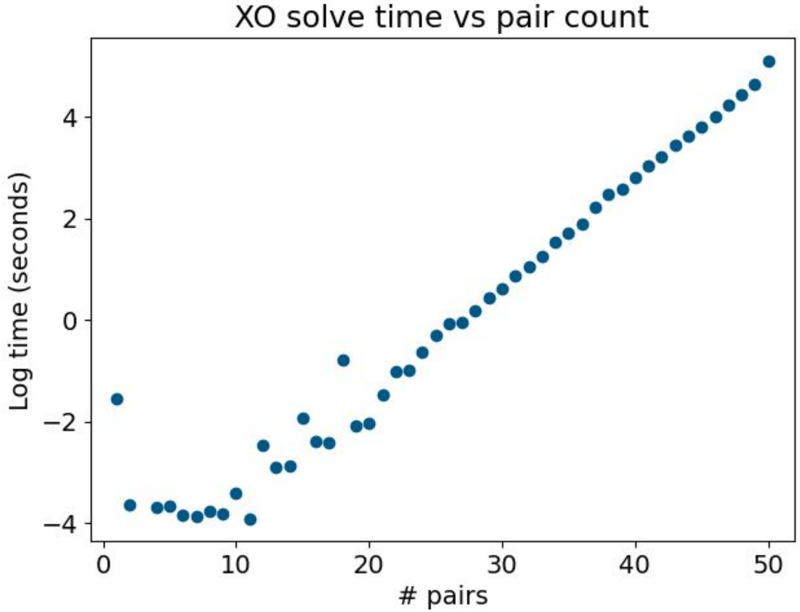
- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black



(■□)^19 = previous record
■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□
0.084 seconds on clean start!

Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black



(■□)^19 = previous record

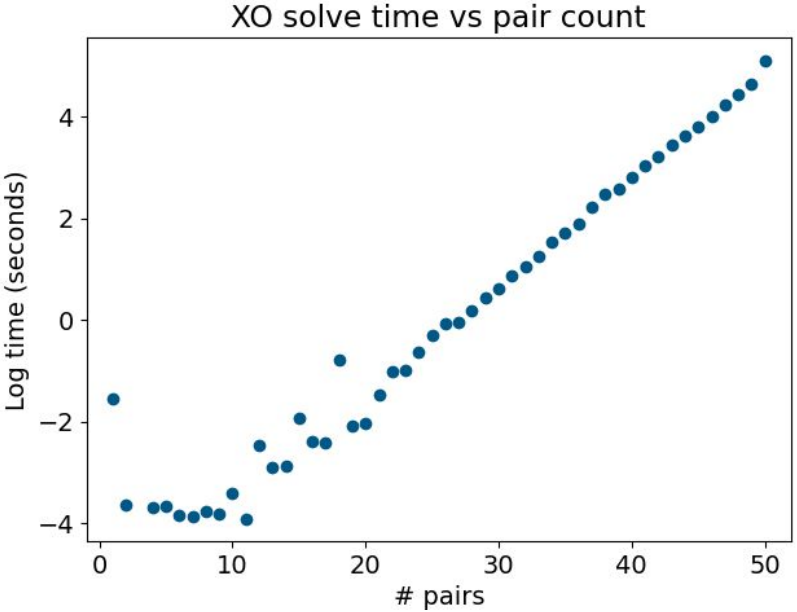
■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□

0.084 seconds on clean start!

(■□)^50 = new record

Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black

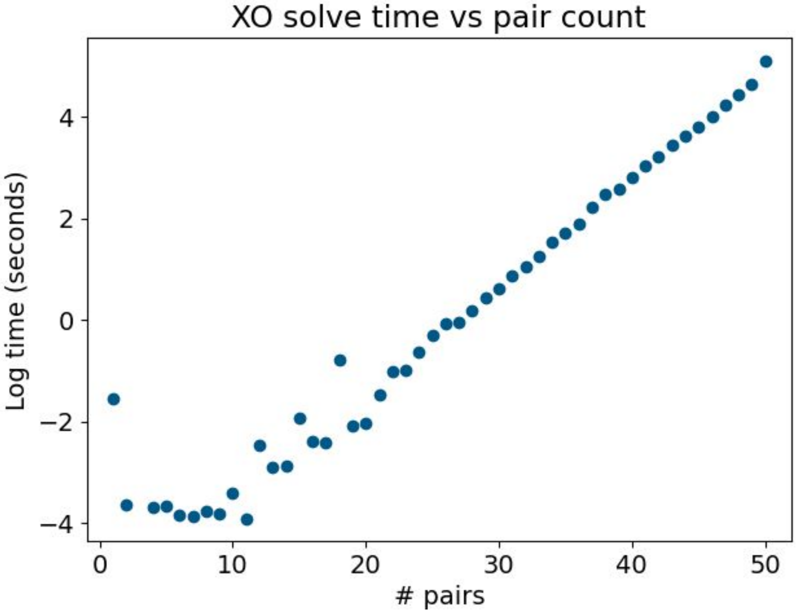


(■□)^19 = previous record
■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□
0.084 seconds on clean start!

(■□)^50 = new record
■□■
□

Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- Root node plays hardcoded move: 13 -> right
- From n to n+1 -> ~1.6x time
- Solve only for Black

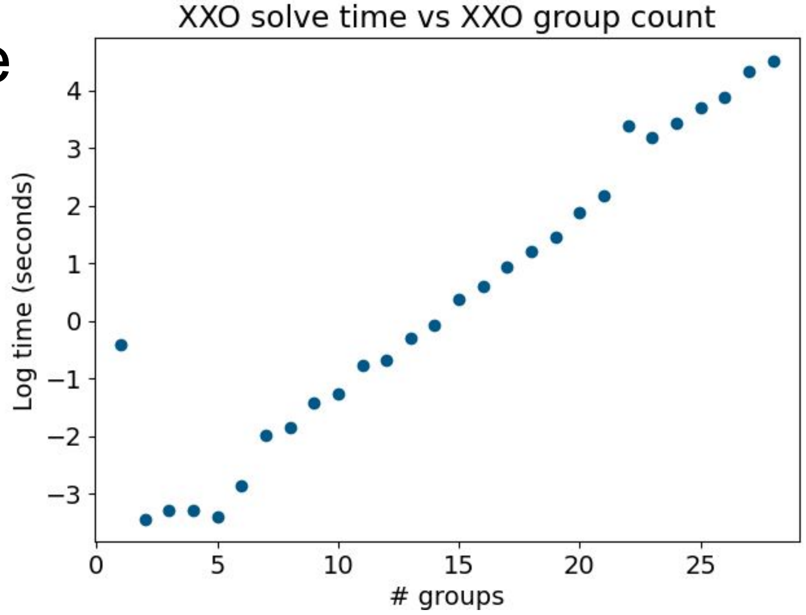


(■□)^19 = previous record
■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□
0.084 seconds on clean start!

(■□)^50 = new record
■□■
□
~35 hours

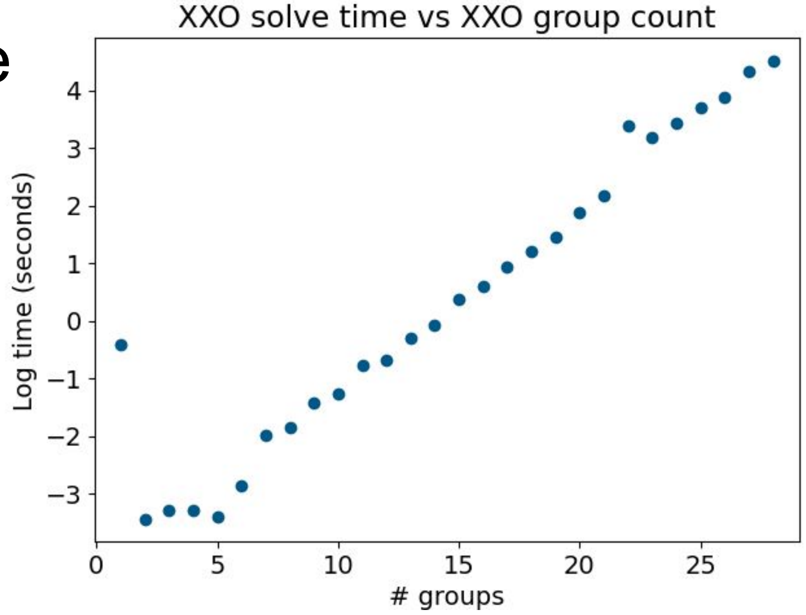
Verifying (■■□)^n Conjecture

Verifying (■■□)^n Conjecture



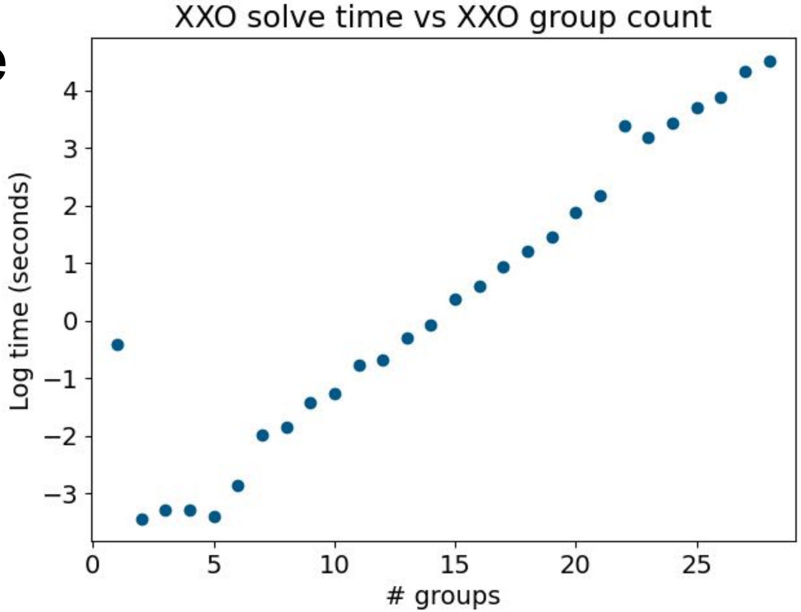
Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries



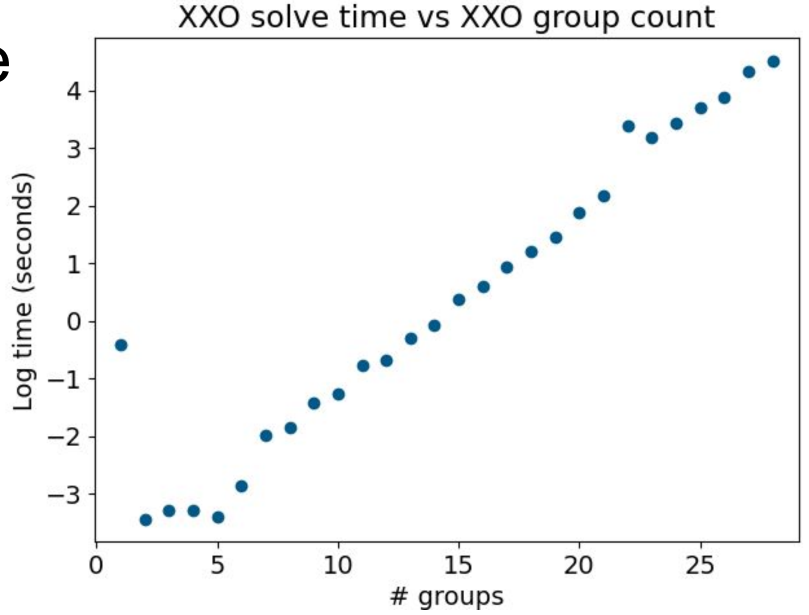
Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time



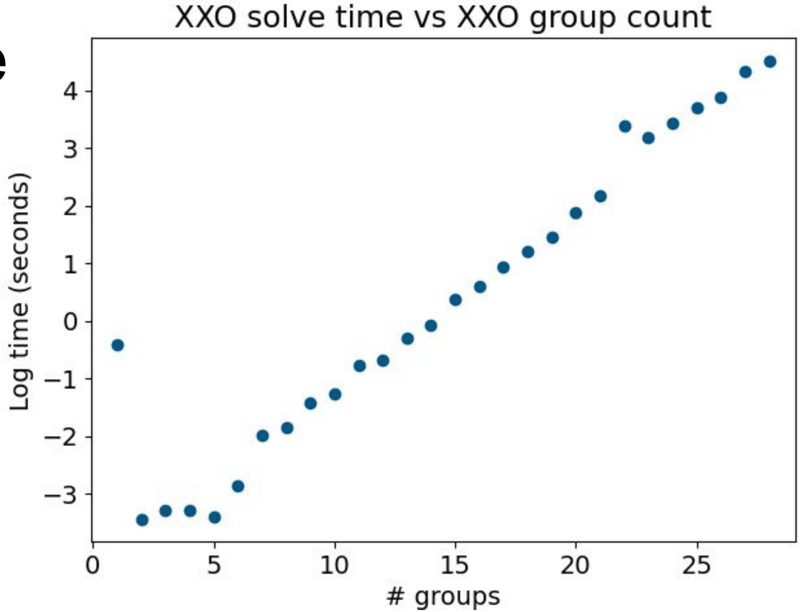
Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



Verifying (■■□)^n Conjecture

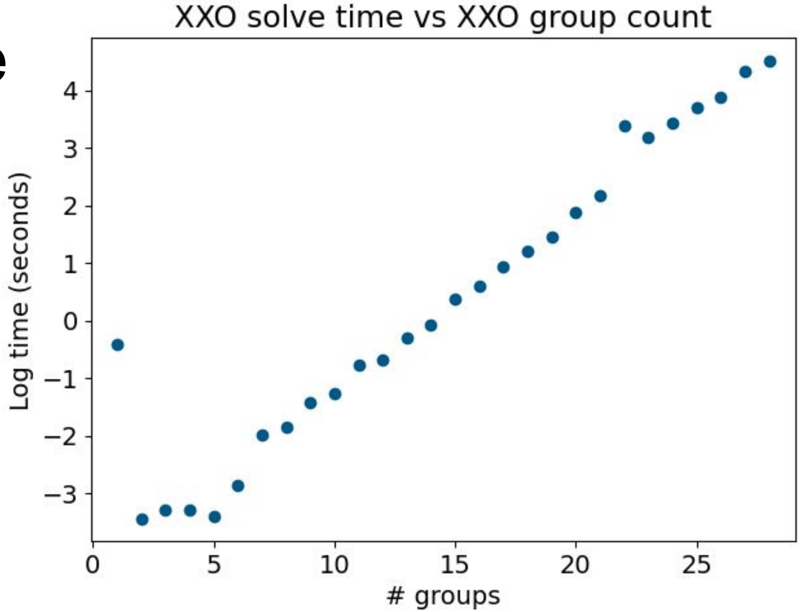
- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



(■■□)^17 + 9(□□■) = previous record

Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White

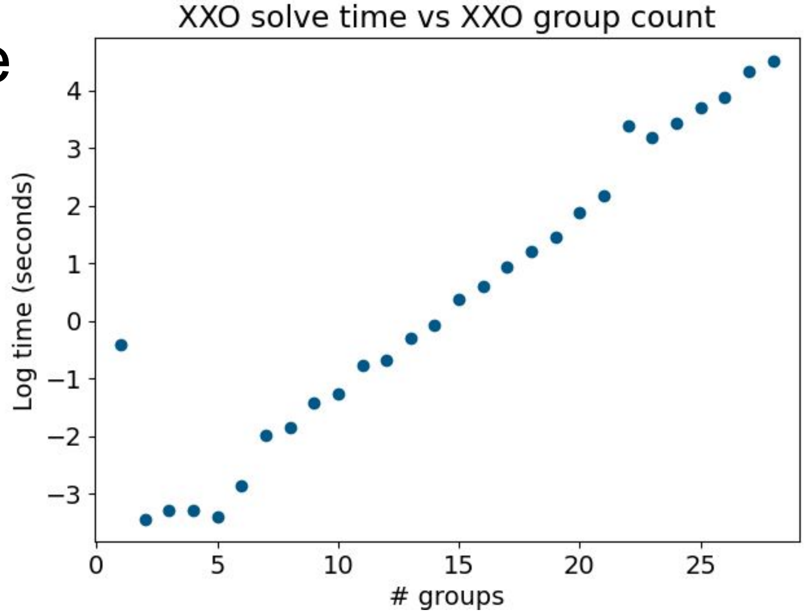


(■■□)^17 + 9(□□■) = previous record

■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■

Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



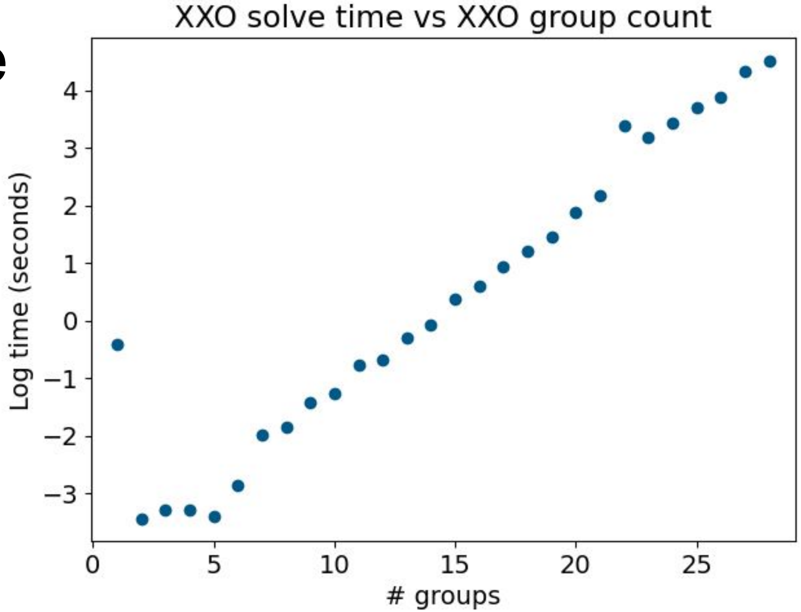
(■■□)^17 + 9(□□■) = previous record

■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■

12.72 seconds on clean start!

Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



(■■□)^17 + 9(□□■) = previous record

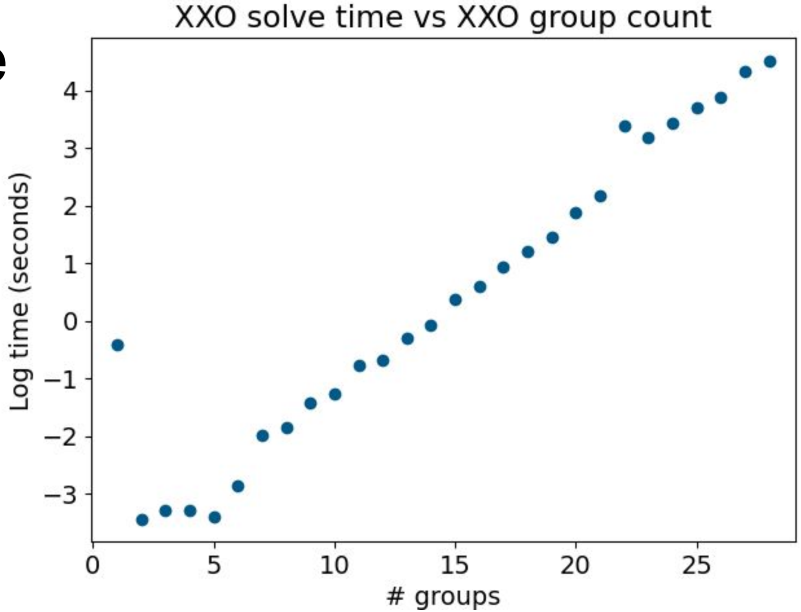
■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■

12.72 seconds on clean start!

(■■□)^28 + 14(□□■) = new record

Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



(■■□)^17 + 9(□□■) = previous record

■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■

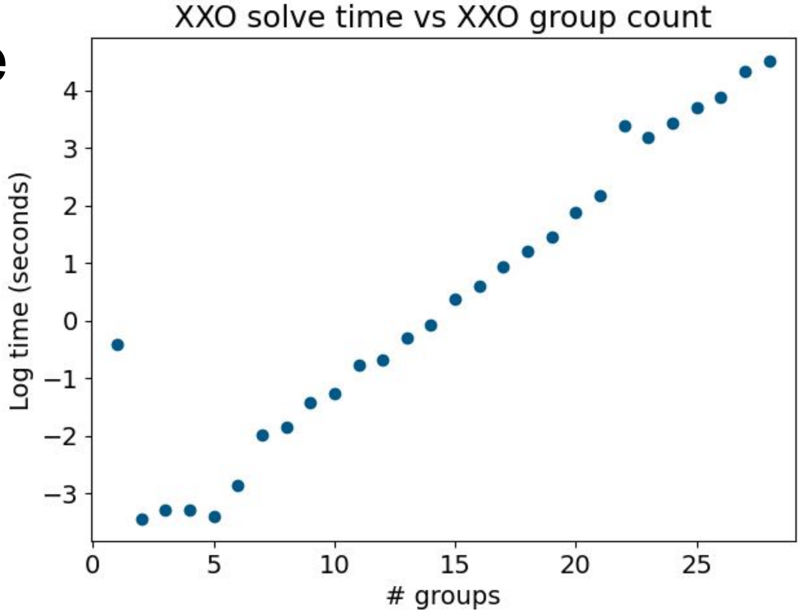
12.72 seconds on clean start!

(■■□)^28 + 14(□□■) = new record

■■□_□□■_□□■_□□■_□□
■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■

Verifying (■■□)^n Conjecture

- Transposition table not cleared between runs
- 4*2^29 table entries
- From n to n+1 -> ~2.12x time
- Solve for both Black and White



(■■□)^17 + 9(□□■) = previous record
■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□■■□_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■
12.72 seconds on clean start!

(■■□)^28 + 14(□□■) = new record
■■□_□□■_□□■_□□■_□□
■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■_□□■
~9 hours

SEGClobber Download

- https://github.com/tfolkersen/SEGClobber
- MIT license
- Thanks for listening!