

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: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$
 - Verified for $n \leq 19$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$
 - Verified for $n \leq 19$
 - We verify for $n \leq 50$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$
 - Verified for $n \leq 19$
 - We verify for $n \leq 50$
 - $(\blacksquare\blacksquare\square)^n = \lfloor (n+1)/2 \rfloor \cdot \uparrow$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$
 - Verified for $n \leq 19$
 - We verify for $n \leq 50$
 - $(\blacksquare\blacksquare\square)^n = \lfloor (n+1)/2 \rfloor \cdot \uparrow$
 - Verified for $n \leq 17$
 - We verify for $n \leq 28$

Conjectures

- Notation: $(\blacksquare\square)^3 = \blacksquare\square\blacksquare\square\blacksquare\square$
- Albert et al. made 2 conjectures about linear Clobber:
 - $(\blacksquare\square)^n$ is a first player win for $n \neq 3$
 - Verified for $n \leq 19$
 - We verify for $n \leq 50$
 - $(\blacksquare\blacksquare\square)^n = \lfloor (n + 1)/2 \rfloor \cdot \uparrow$
 - Verified for $n \leq 17$
 - We verify for $n \leq 28$
- Chen et al. recently proposed a proof of $(\blacksquare\square)^n$ 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 , $= 0$, ~~$\neq 0$~~

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 , $= 0$, $\neq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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)
 - $> 0, < 0, = 0, \not\geq 0$
 - 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

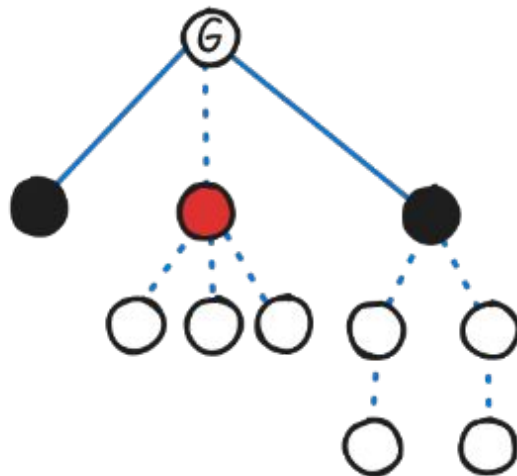
Normal Form

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

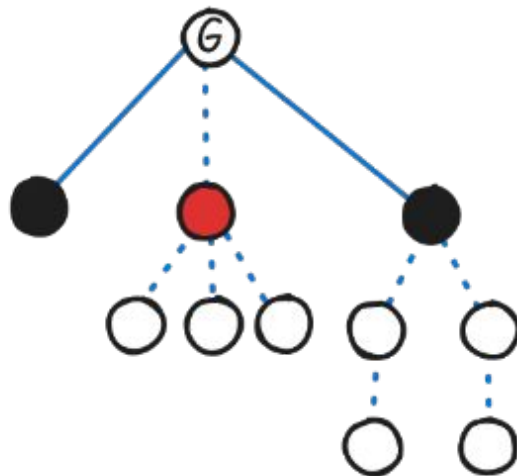
- Non-strictly-dominated move
- Strictly-dominated move



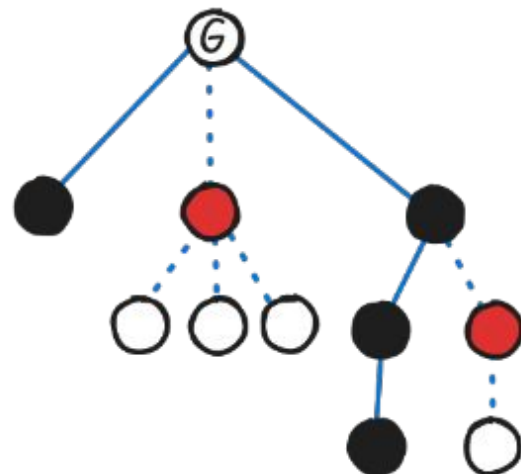
$$\text{CS3(G)} = 2$$

Complexity Score

- Non-strictly-dominated move
- Strictly-dominated move



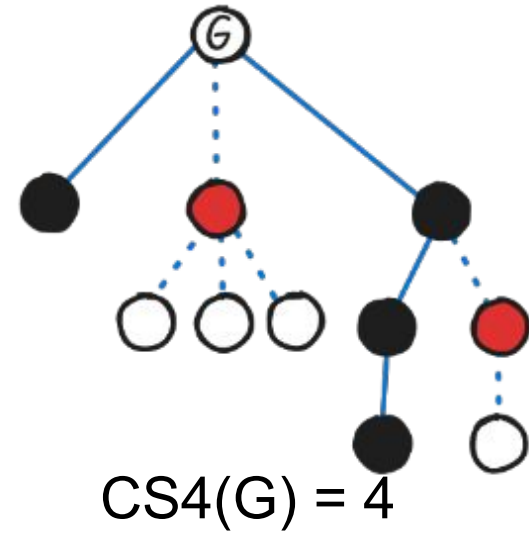
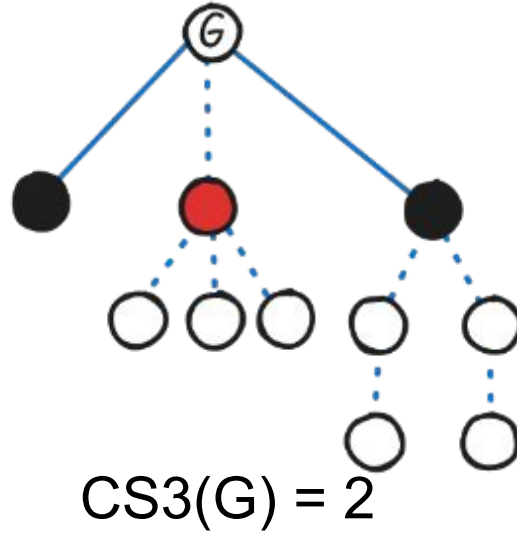
$$CS3(G) = 2$$



$$CS4(G) = 4$$

Complexity Score

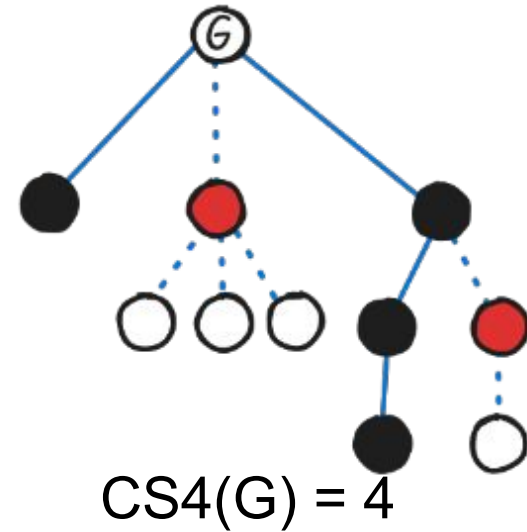
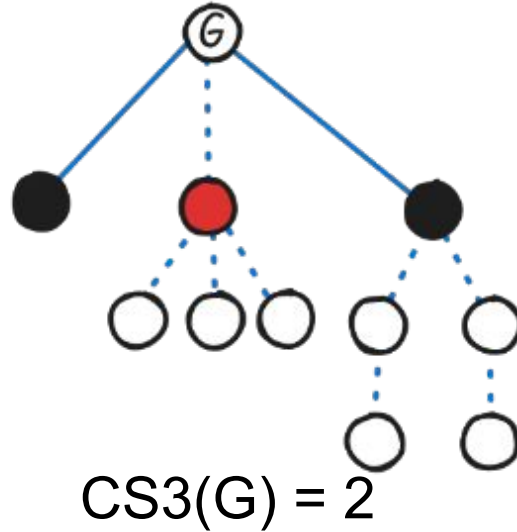
- Non-strictly-dominated move
- Strictly-dominated move



- $G1$ may replace $G2$ during search if:

Complexity Score

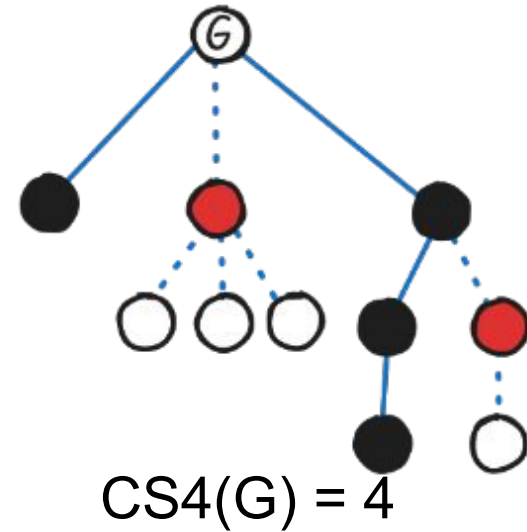
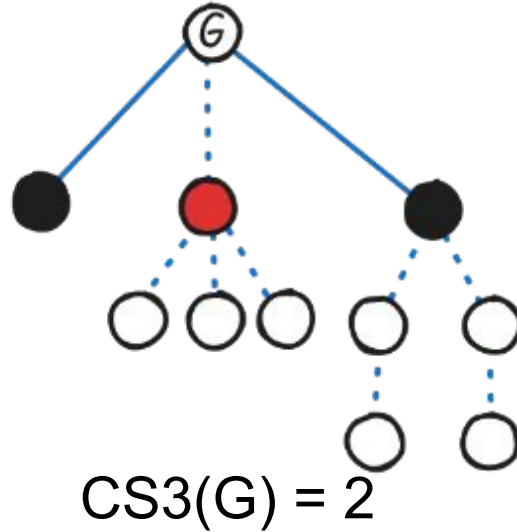
- Non-strictly-dominated move
- Strictly-dominated move



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

Complexity Score

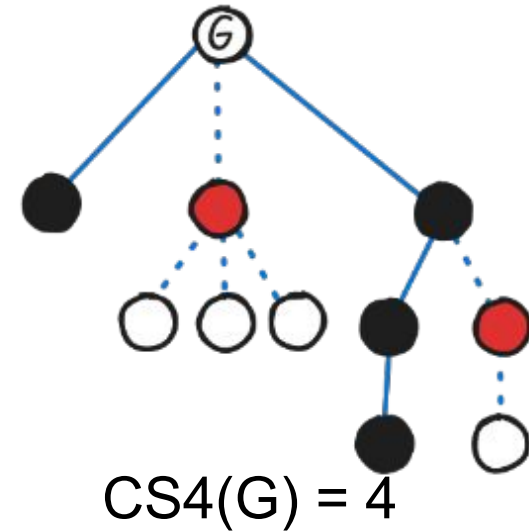
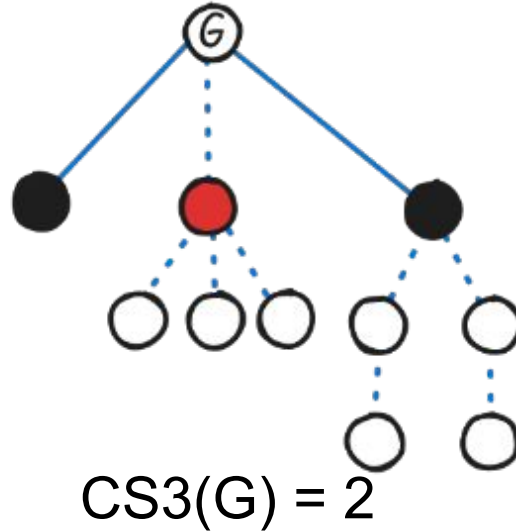
- Non-strictly-dominated move
- Strictly-dominated move



- $G1$ 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

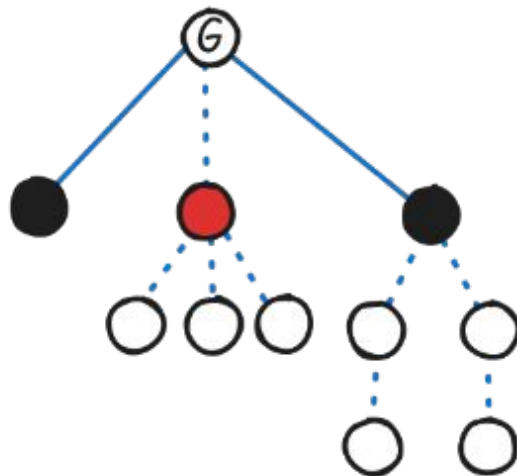
- Non-strictly-dominated move
- Strictly-dominated move



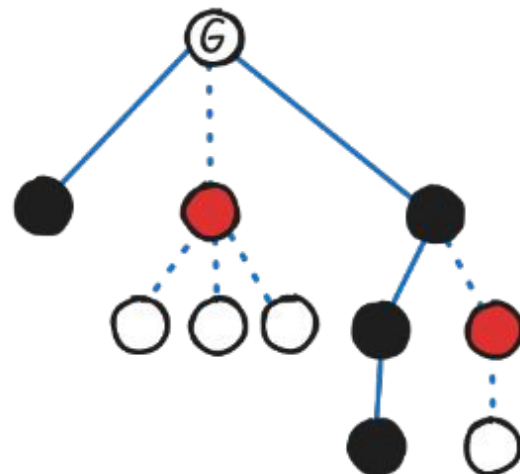
- $G1$ may replace $G2$ during search if:
 - $G1$ is **simpler** than $G2$: $CS(G1) < CS(G2)$
 - **AND** $G1$ comes **before** $G2$ in the database
 - To prevent cycles

Complexity Score

- Non-strictly-dominated move
- Strictly-dominated move



$$CS3(G) = 2$$

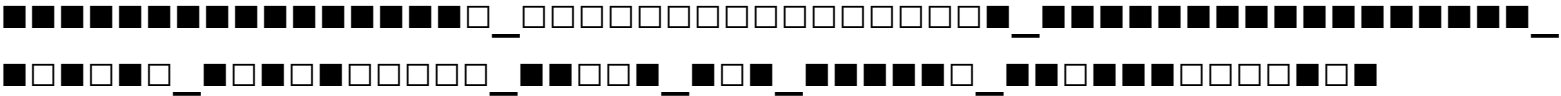


$$CS4(G) = 4$$

- 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

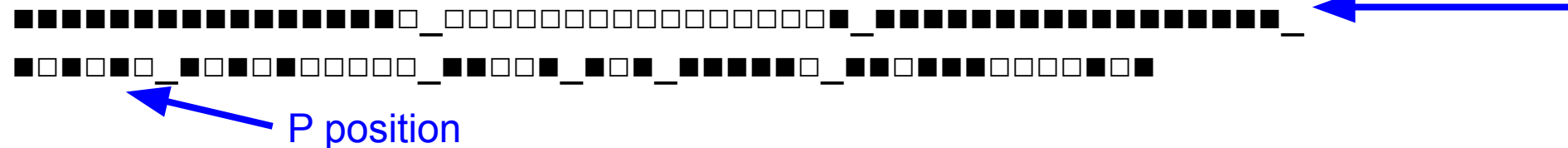
1. Set aside large subgames (those not in DB)



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

Sum Simplification

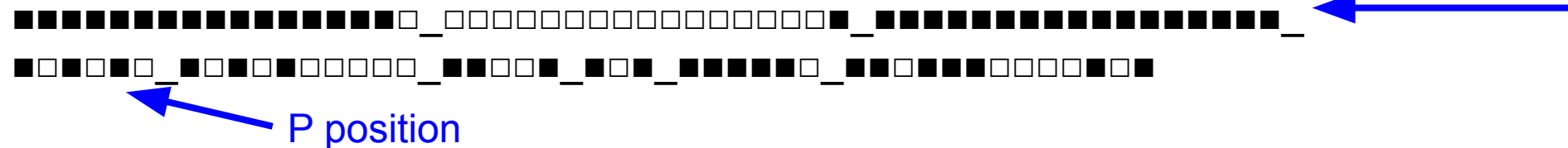
1. Set aside large subgames (those not in DB)



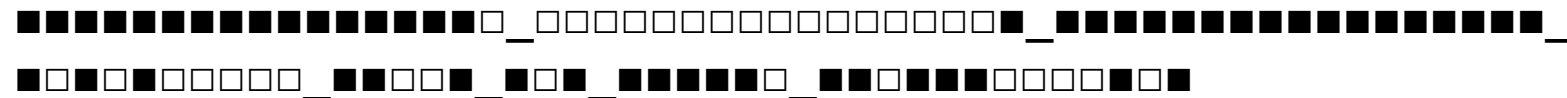
2. Using DB, delete single P positions (= 0)

Sum Simplification

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

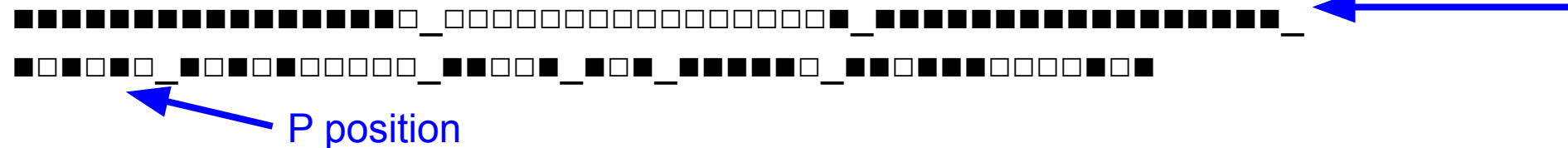


2. Using DB, delete single P positions (= 0)

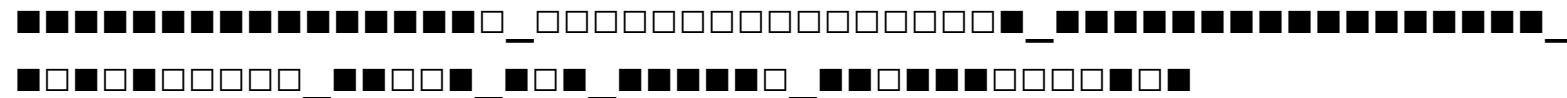


Sum Simplification

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



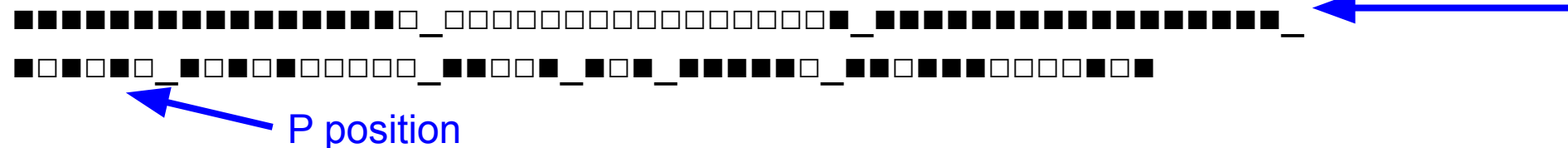
2. Using DB, delete single P positions (= 0)



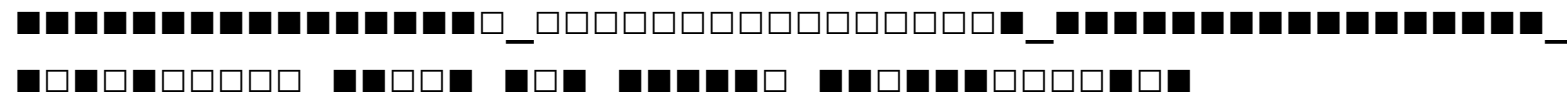
3. Sort remaining subgames in order of decreasing length

Sum Simplification

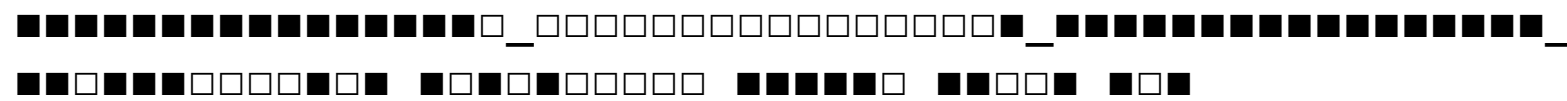
1. Set aside large subgames (those not in DB)



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

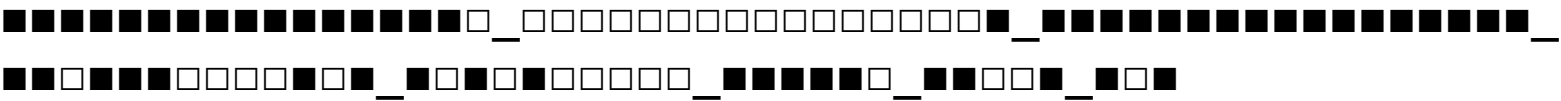


3. Sort remaining subgames in order of decreasing length



Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length



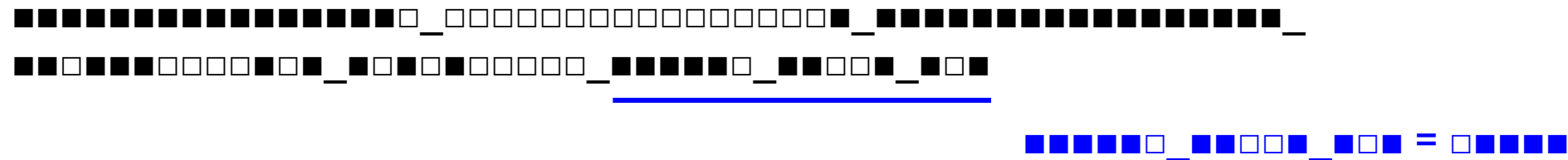
Sum Simplification (Continued)

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)

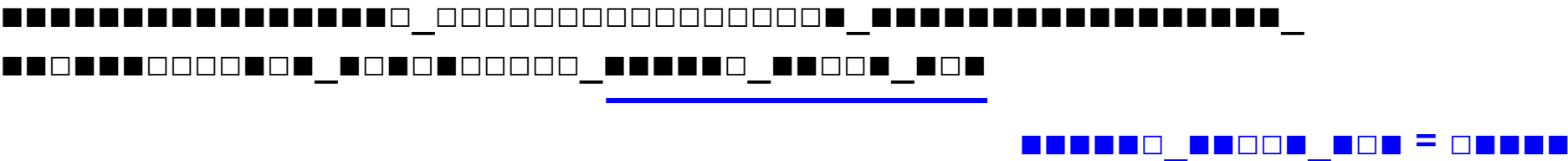
3. Sort remaining subgames in order of decreasing length

[illegible]

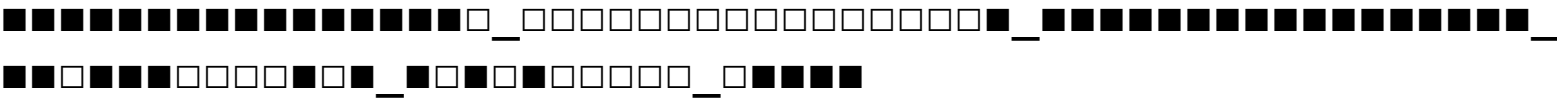
4. Replace sums of 3+ (using sliding window of length 16)

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

[illegible]

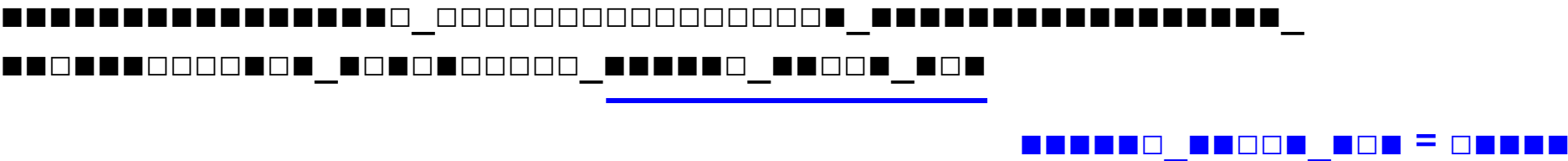
4. Replace sums of 3+ (using sliding window of length 16)

The diagram illustrates the process of finding a common denominator for two fractions. It shows the fractions $\frac{1}{2}$ and $\frac{1}{3}$ being converted to $\frac{2}{4}$ and $\frac{1}{3}$. A blue line connects the denominators 4 and 3, leading to the least common denominator 12. The final row shows the equivalent fractions $\frac{3}{6}$ and $\frac{4}{6}$, which are added to get $\frac{7}{6}$.

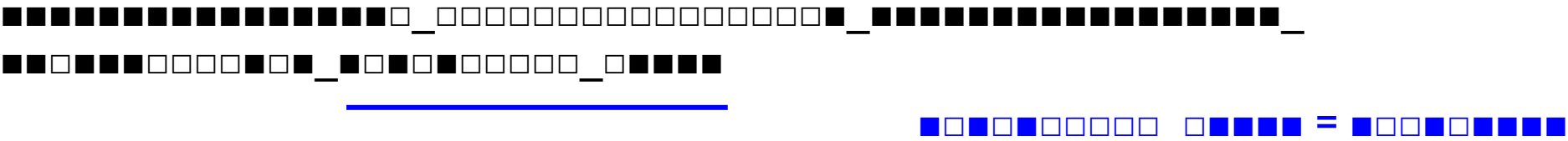
5. Replace pairs

Sum Simplification (Continued)

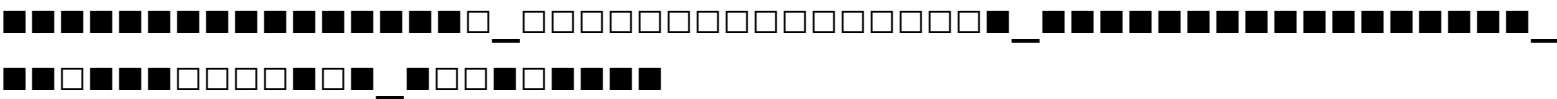
3. Sort remaining subgames in order of decreasing length



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)

Diagram illustrating the process of finding a common denominator for two fractions:

$$\frac{1}{2} + \frac{1}{3} = \frac{3}{6} + \frac{2}{6} = \frac{5}{6}$$

5. Replace pairs

6. Replace singles

Sum Simplification (Continued)

3. Sort remaining subgames in order of decreasing length

1	0	1	1	1							
1	0	1	0	1							
					=						
						1	0	0	1	0	0

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

A visual representation of the distributive property. On the left, there are 5 groups, each containing 2 white squares and 3 blue squares. This is followed by an equals sign. On the right, there are 10 white squares and 15 blue squares.

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)

A horizontal number line with arrows at both ends, labeled from 0 to 100 in increments of 10. A blue line segment is drawn above the number line, starting at 10 and ending at 20, with a blue arrow pointing from 10 to 20. Below the number line, the text "10 units" is written.

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)

Frequency	Count
Daily	15
Several times a week	10
Once a week	5
Less than once a week	10

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)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

Sum Simplification (Continued)

6. Replace singles

Resume considering top row

7. Now also consider previously set aside games.

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

Resume
considering
top row



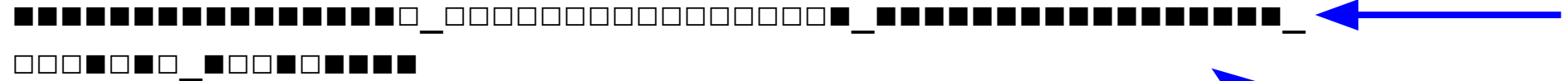
6. Replace singles

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

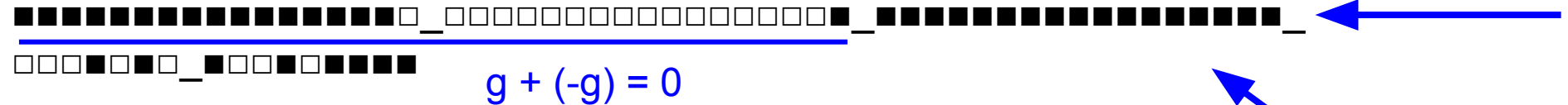


7. Now also consider previously set aside games.

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

Sum Simplification (Continued)

6. Replace singles

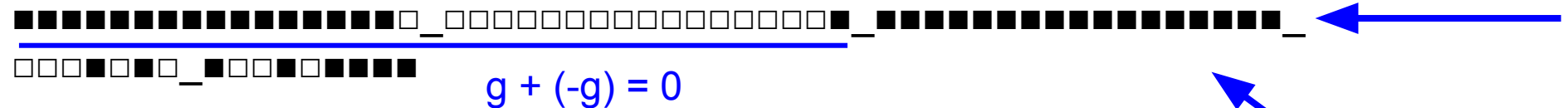


7. Now also consider previously set aside games.

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

Sum Simplification (Continued)

6. Replace singles



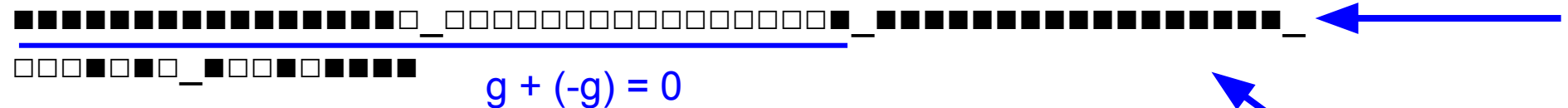
7. Now also consider previously set aside games.

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



Sum Simplification (Continued)

6. Replace singles



7. Now also consider previously set aside games.

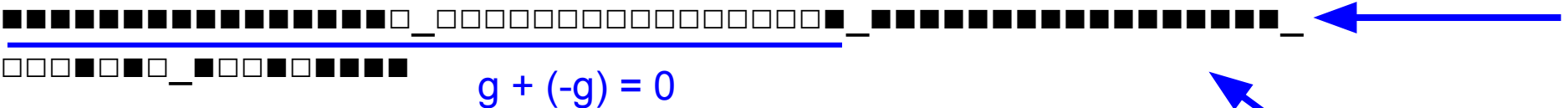
8. Delete subgames with no moves, and pairs of visually identifiable inverses



9. Convert to normal form

Sum Simplification (Continued)

6. Replace singles



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 G_i with certain known outcome classes, try to solve $G - G_i$ 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 G_i with certain known outcome classes, try to solve $G - G_i$ 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 G_i with certain known outcome classes, try to solve $G - G_i$ 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 G_i with certain known outcome classes, try to solve $G - G_i$ 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 G_i with certain known outcome classes, try to solve $G - G_i$ instead
5. Generate and play moves
 - Move ordering informed by:
 - Outcome classes
 - DB “simplest moves”

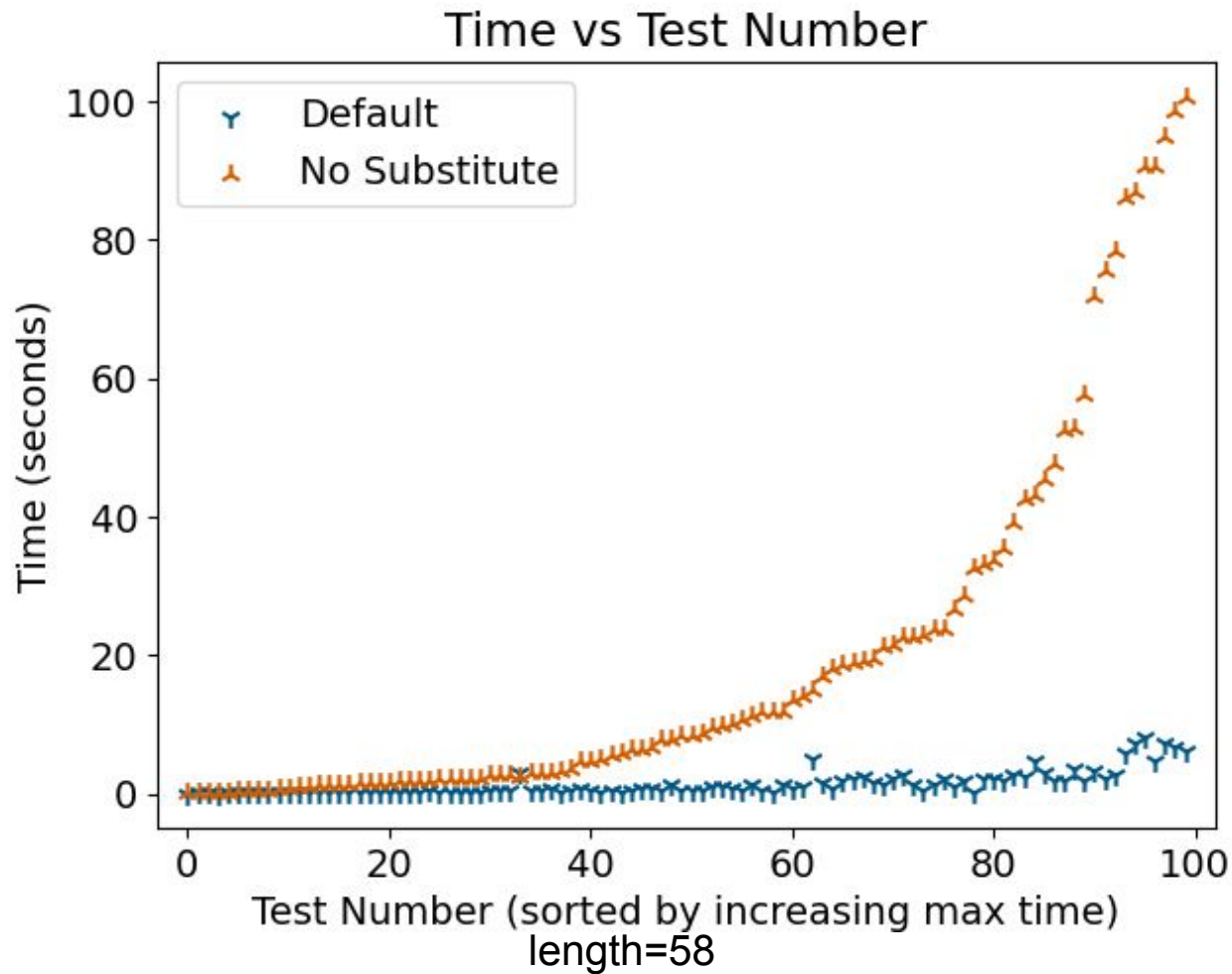
Search Algorithm

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

Search Algorithm

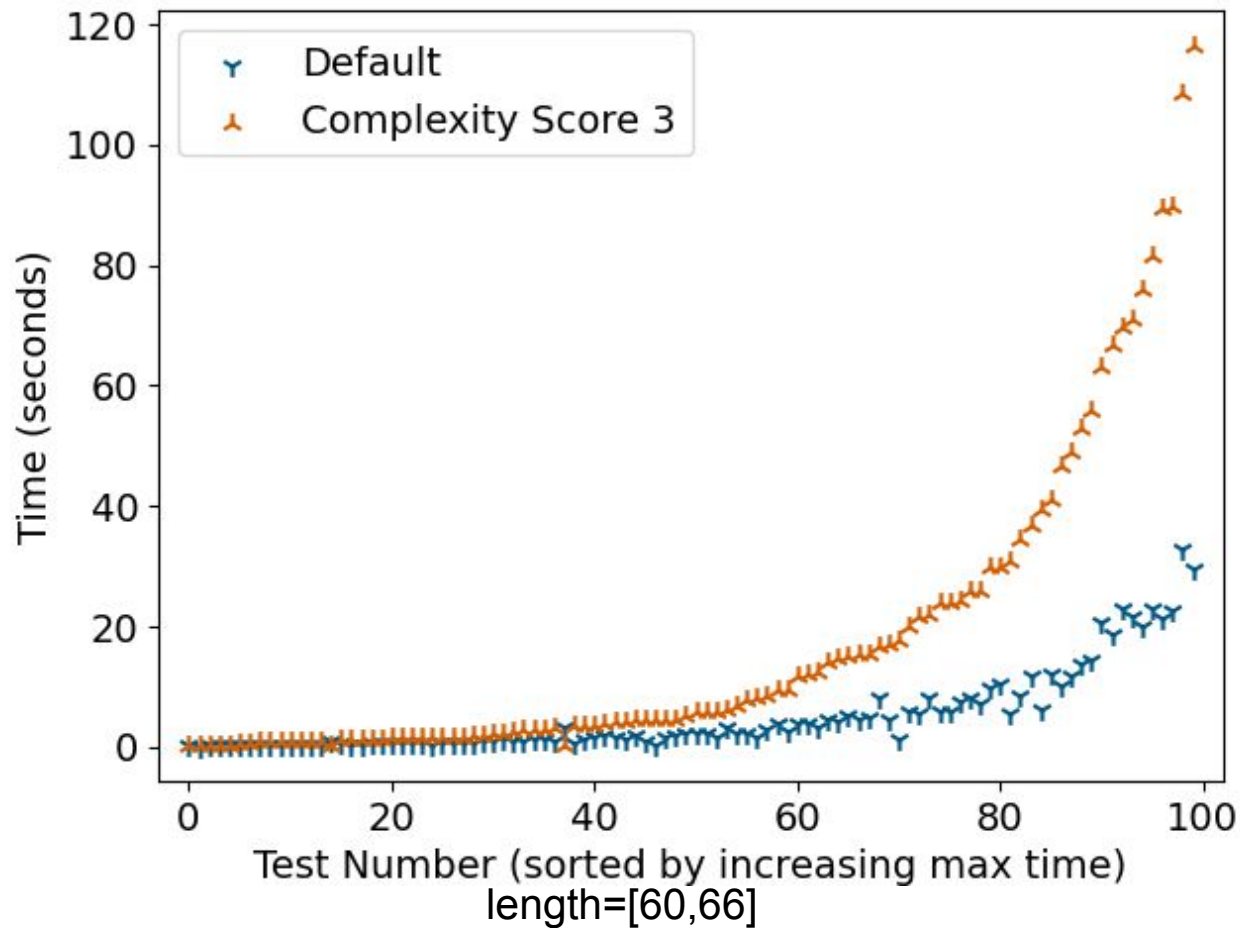
1. Simplify board
2. Check database, then transposition table
3. Try outcome class and bounds to solve
4. Speculative subgame removal
 - For subgame G_i with certain known outcome classes, try to solve $G - G_i$ instead
5. Generate and play moves
 - Move ordering informed by:
 - Outcome classes
 - DB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

Ablation: Substitution

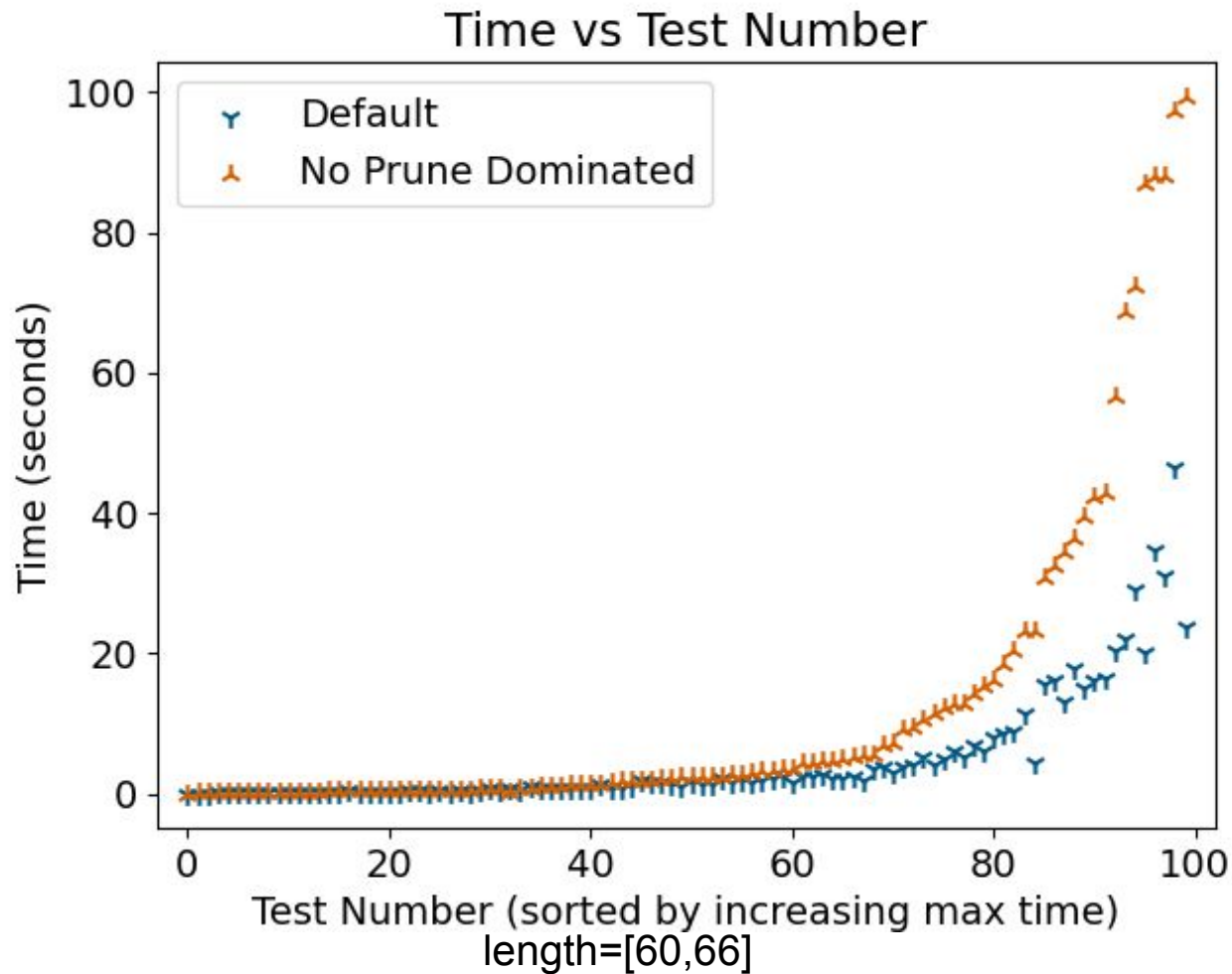


Ablation: Complexity Score (CS4 vs CS3)

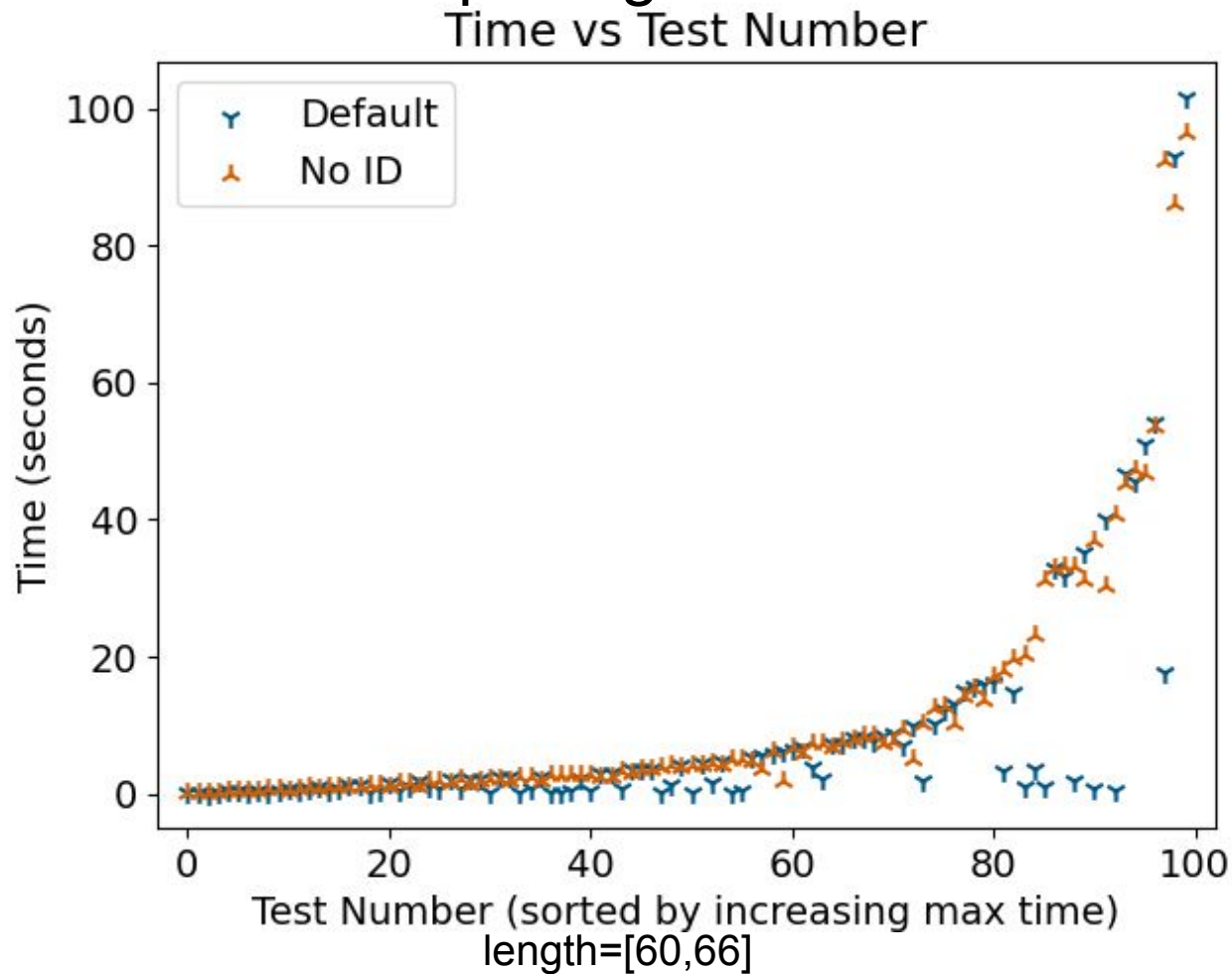
Time vs Test Number



Ablation: Prune Dominated Moves

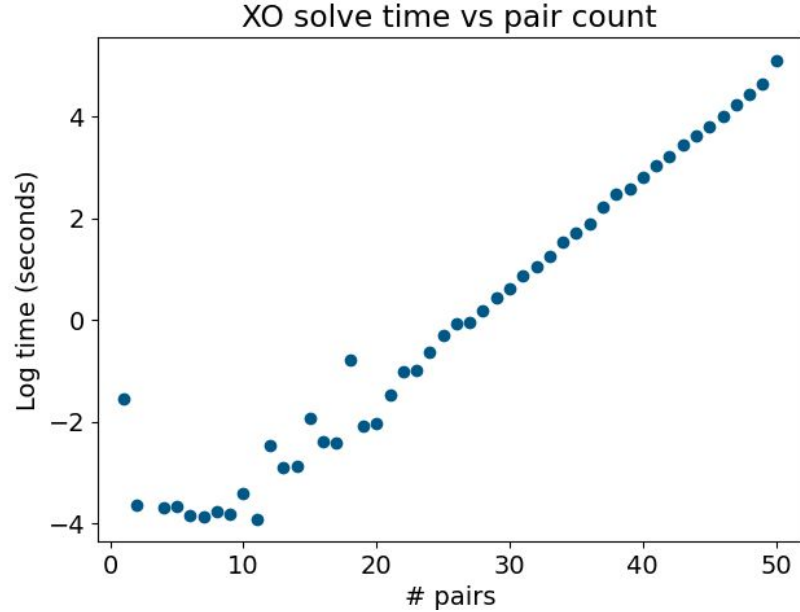


Ablation: Iterative Deepening



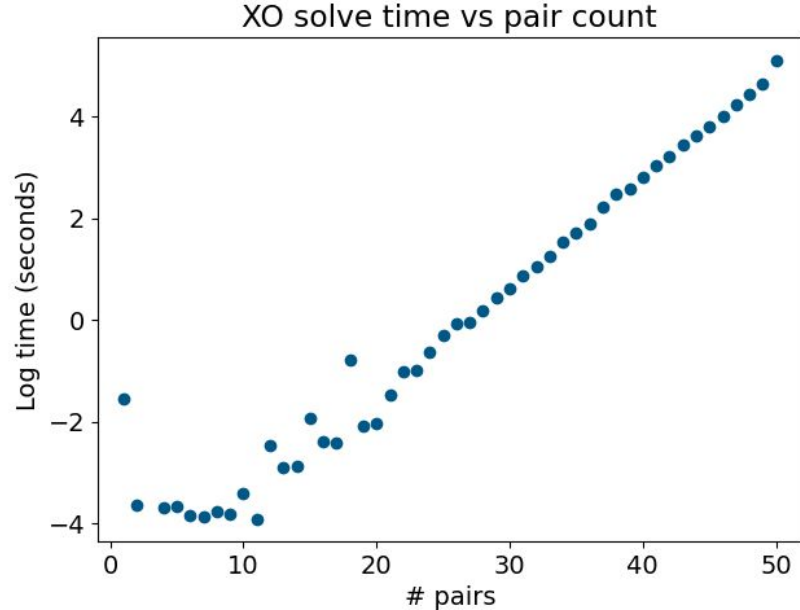
Verifying $(\blacksquare\Box)^n$ Conjecture

Verifying $(\blacksquare\blacksquare)^n$ Conjecture



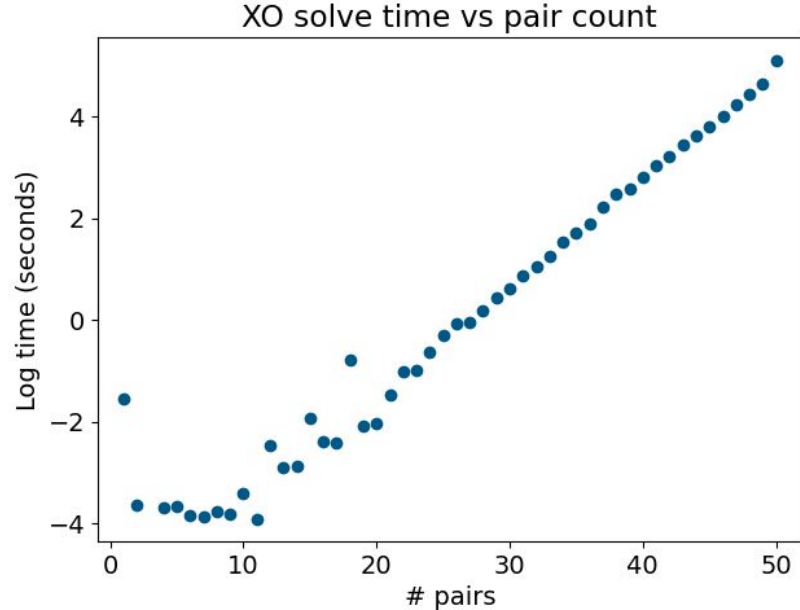
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs



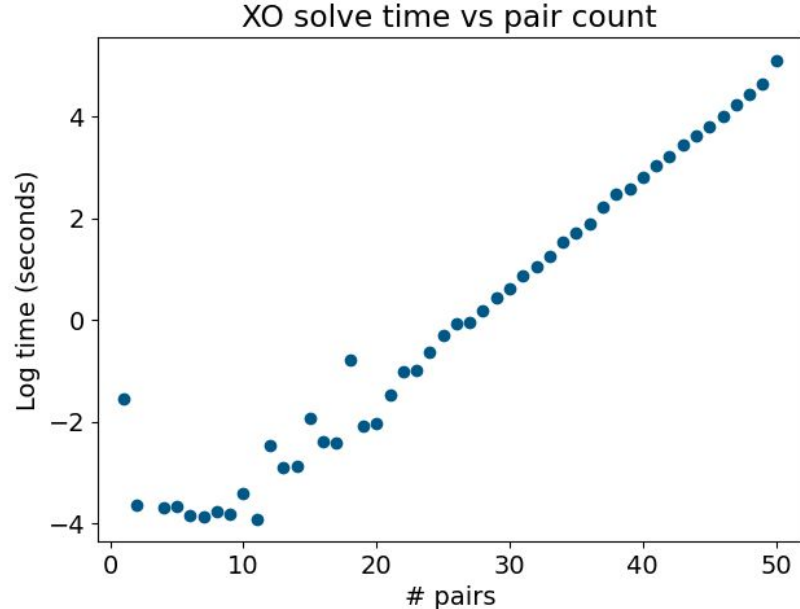
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries



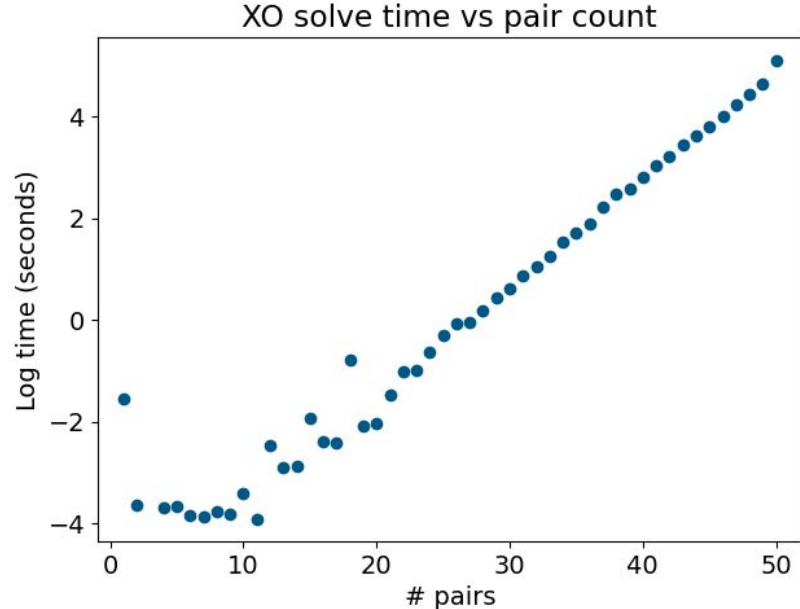
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- Root node plays hardcoded move: 13 -> right



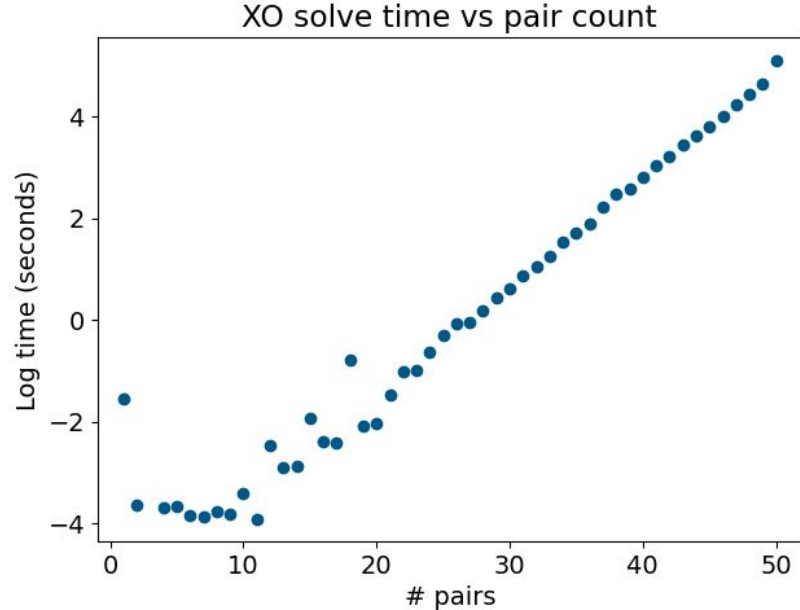
Verifying (■□)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- Root node plays hardcoded move: 13 -> right
- From n to $n+1$ -> $\sim 1.6x$ time



Verifying (■□)^n Conjecture

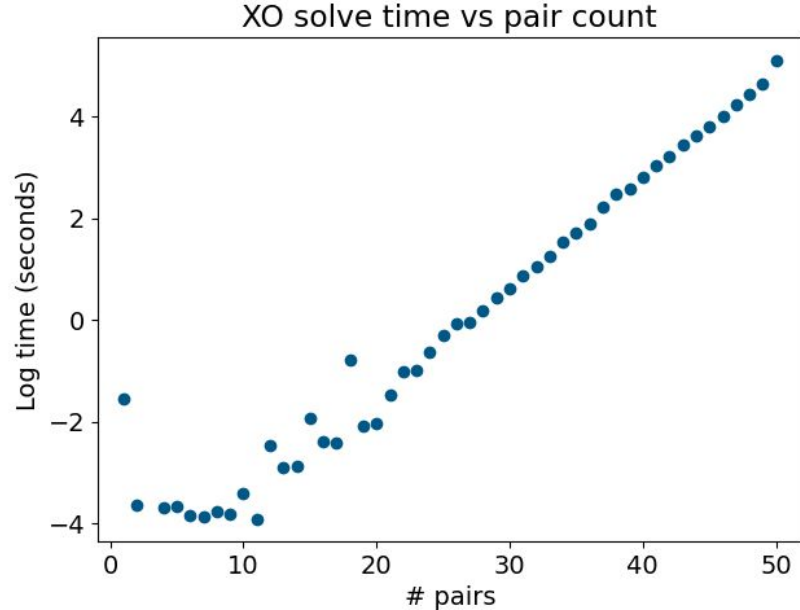
- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- Root node plays hardcoded move: 13 -> right
- From n to $n+1$ -> $\sim 1.6x$ time
- Solve only for Black



Verifying (■□)^n Conjecture

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

(■□)^19 = previous record



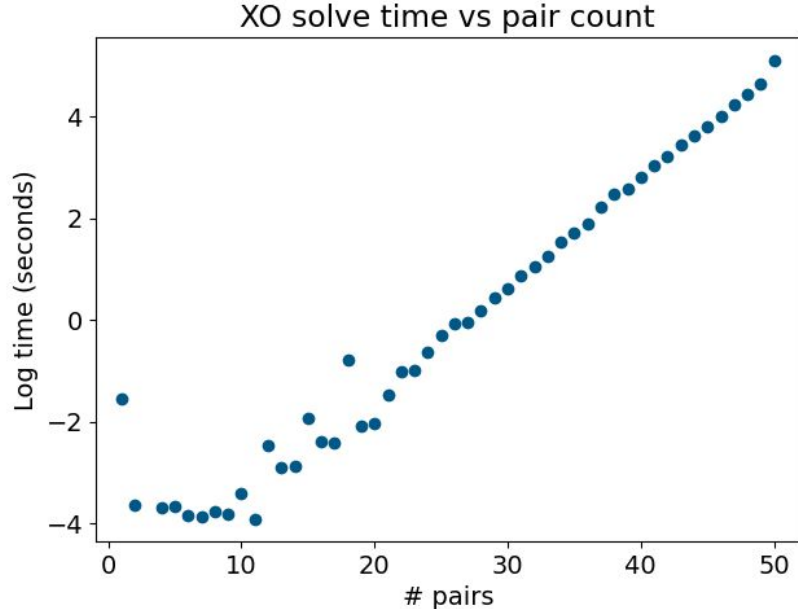
Verifying $(\blacksquare\Box)^n$ Conjecture

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

$(\blacksquare\square)^{19} = \text{previous record}$

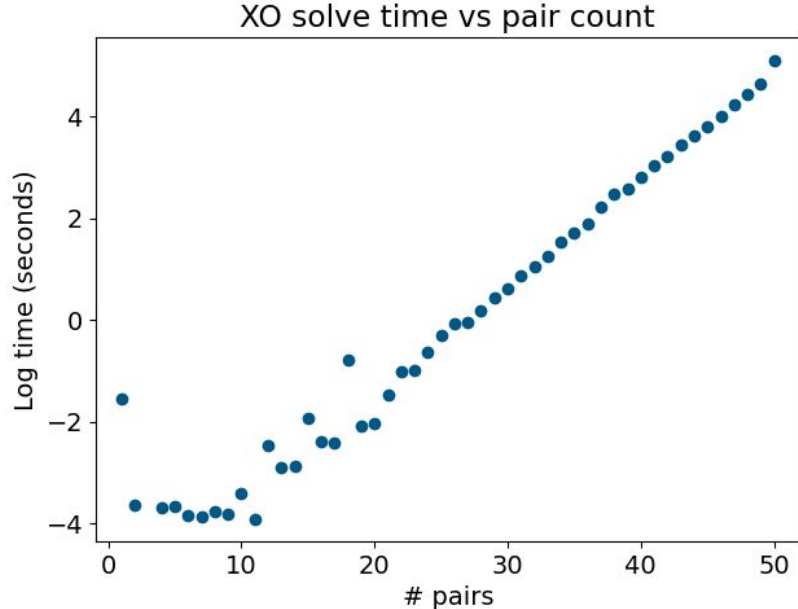
[illegible]

0.084 seconds on clean start!



Verifying $(\blacksquare\Box)^n$ Conjecture

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



$(\blacksquare\square)^{19} = \text{previous record}$

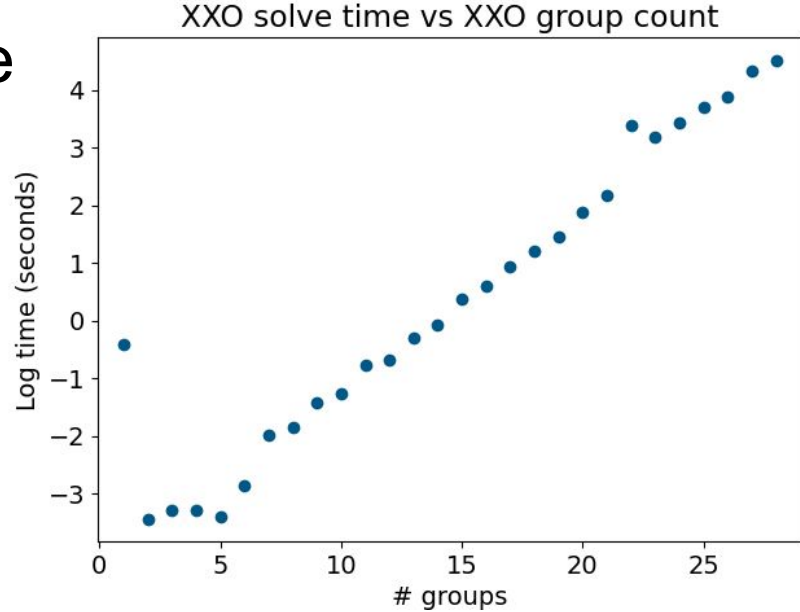
0 **1** **2** **3** **4** **5** **6** **7** **8** **9** **A** **B** **C** **D** **E** **F** **G** **H** **I** **J** **K** **L** **M** **N** **O** **P** **Q** **R** **S** **T** **U** **V** **W** **X** **Y** **Z**

0.084 seconds on clean start!

$$(\blacksquare\square)^{50} = \text{new record}$$

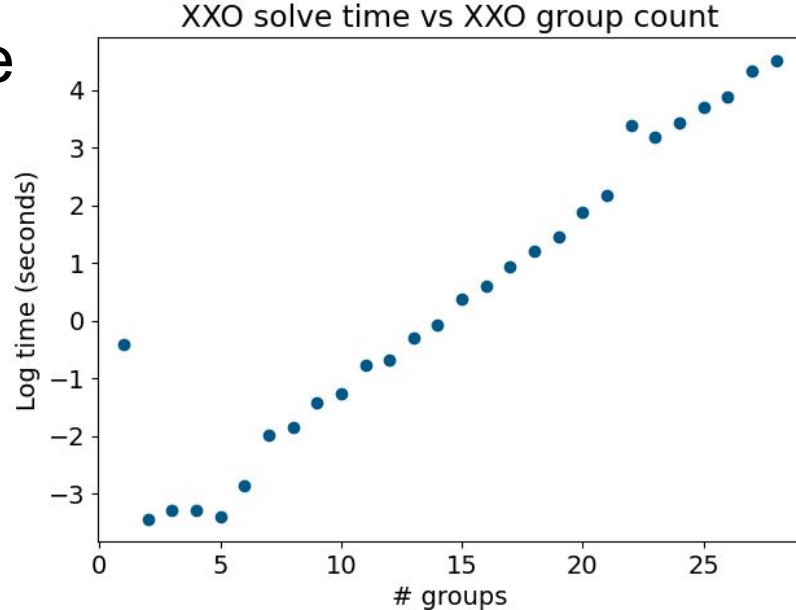
Verifying $(\blacksquare\blacksquare\square)^n$ Conjecture

Verifying (■■■)^n Conjecture



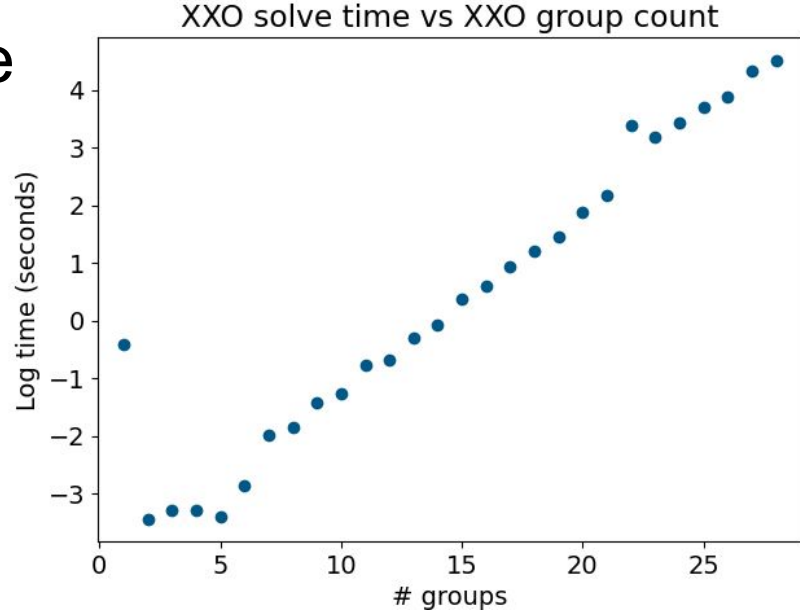
Verifying (■■■)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries



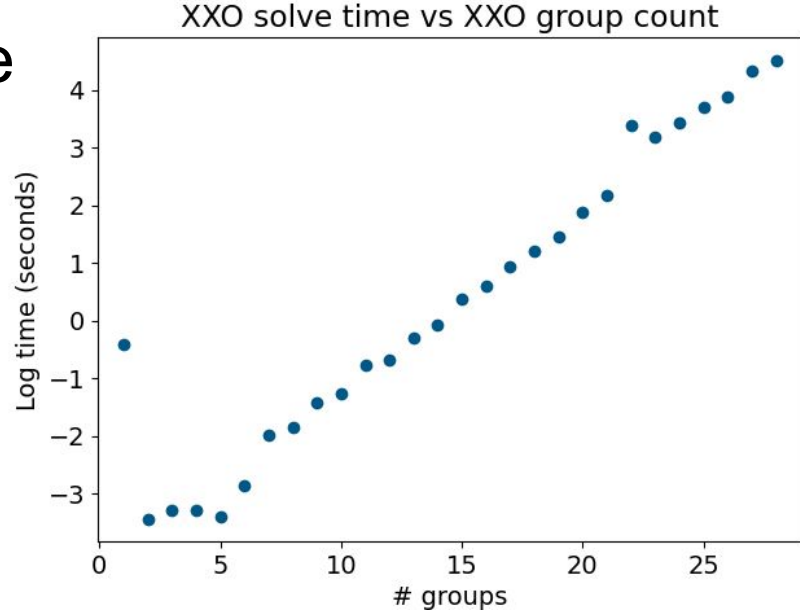
Verifying (■■■□)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- From n to $n+1 \rightarrow \sim 2.12x$ time



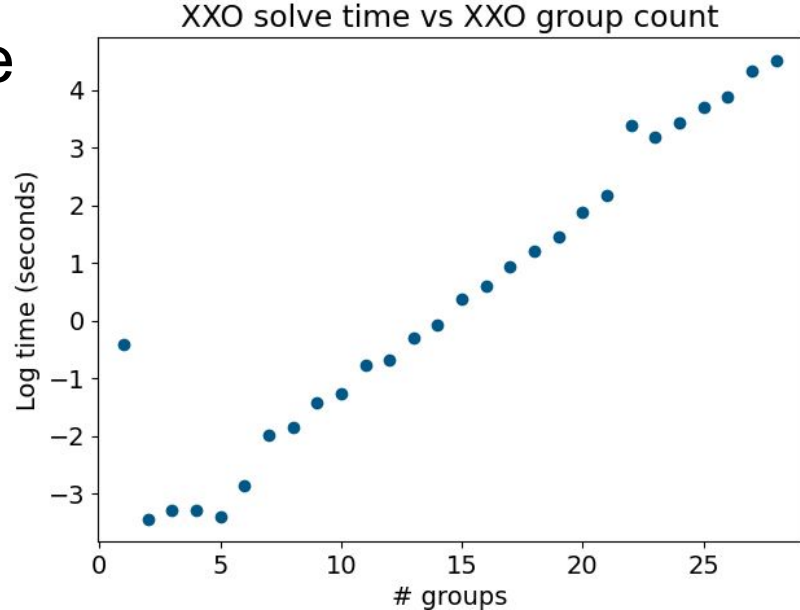
Verifying (■■■)^n Conjecture

- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- From n to $n+1 \rightarrow \sim 2.12x$ time
- Solve for both Black and White



Verifying (■■■)^n Conjecture

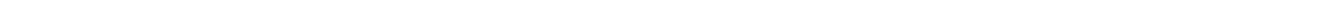
- Transposition table not cleared between runs
- $4 \cdot 2^{29}$ table entries
- From n to $n+1 \rightarrow \sim 2.12x$ time
- Solve for both Black and White



$(■■■)^{17} + 9(□□■) = \text{previous record}$

XXO solve time vs XXO group count

-
- A scatter plot showing the relationship between the number of groups (x-axis) and the log time in seconds (y-axis). The x-axis ranges from 0 to 28, and the y-axis ranges from -4 to 4. The data points show a general upward trend, with some fluctuations, indicating that log time increases as the number of groups increases.
- | # groups | Log time (seconds) |
|----------|--------------------|
| 1 | -0.5 |
| 2 | -3.5 |
| 3 | -3.3 |
| 4 | -3.3 |
| 5 | -3.4 |
| 6 | -2.9 |
| 7 | -2.1 |
| 8 | -1.9 |
| 9 | -1.4 |
| 10 | -1.3 |
| 11 | -0.8 |
| 12 | -0.7 |
| 13 | -0.3 |
| 14 | -0.1 |
| 15 | 0.4 |
| 16 | 0.6 |
| 17 | 0.9 |
| 18 | 1.2 |
| 19 | 1.5 |
| 20 | 1.9 |
| 21 | 2.2 |
| 22 | 3.4 |
| 23 | 3.2 |
| 24 | 3.5 |
| 25 | 3.7 |
| 26 | 3.9 |
| 27 | 4.3 |
| 28 | 4.5 |



12.72 seconds on clean start!

XXO solve time vs XXO group count

-
- A scatter plot showing the relationship between the number of groups (x-axis) and the log time in seconds (y-axis). The x-axis ranges from 0 to 30, and the y-axis ranges from -4 to 4. The data points show a general upward trend, with a slight dip around 23 groups.
- | # groups | Log time (seconds) |
|----------|--------------------|
| 1 | -0.5 |
| 2 | -3.5 |
| 3 | -3.3 |
| 4 | -3.3 |
| 5 | -3.4 |
| 6 | -2.9 |
| 7 | -2.1 |
| 8 | -1.9 |
| 9 | -1.4 |
| 10 | -1.3 |
| 11 | -0.8 |
| 12 | -0.7 |
| 13 | -0.3 |
| 14 | -0.1 |
| 15 | 0.4 |
| 16 | 0.6 |
| 17 | 0.9 |
| 18 | 1.2 |
| 19 | 1.5 |
| 20 | 1.9 |
| 21 | 2.2 |
| 22 | 3.4 |
| 23 | 3.2 |
| 24 | 3.5 |
| 25 | 3.7 |
| 26 | 3.9 |
| 27 | 4.3 |
| 28 | 4.5 |

Item	Male (%)	Female (%)
1	80	20
2	70	30
3	60	40
4	50	50
5	40	60
6	30	70
7	20	80
8	10	90
9	10	90
10	10	90
11	10	90
12	10	90
13	10	90
14	10	90
15	10	90
16	10	90
17	10	90
18	10	90
19	10	90
20	10	90
21	10	90
22	10	90
23	10	90
24	10	90
25	10	90
26	10	90
27	10	90
28	10	90
29	10	90
30	10	90
31	10	90
32	10	90
33	10	90
34	10	90
35	10	90
36	10	90
37	10	90
38	10	90
39	10	90
40	10	90
41	10	90
42	10	90
43	10	90
44	10	90
45	10	90
46	10	90
47	10	90
48	10	90
49	10	90
50	10	90
51	10	90
52	10	90
53	10	90
54	10	90
55	10	90
56	10	90
57	10	90
58	10	90
59	10	90
60	10	90
61	10	90
62	10	90
63	10	90
64	10	90
65	10	90
66	10	90
67	10	90
68	10	90
69	10	90
70	10	90
71	10	90
72	10	90
73	10	90
74	10	90
75	10	90
76	10	90
77	10	90
78	10	90
79	10	90
80	10	90
81	10	90
82	10	90
83	10	90
84	10	90
85	10	90
86	10	90
87	10	90
88	10	90
89	10	90
90	10	90
91	10	90
92	10	90
93	10	90
94	10	90
95	10	90
96	10	90
97	10	90
98	10	90
99	10	90
100	10	90

$$(\blacksquare\blacksquare\square)^{28} + 14(\square\square\blacksquare) = \text{new record}$$
[illegible]

SEGClobber Download

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