Quarto!

Wouter M. Koolen

Cakes Talk
Thursday 29th September, 2011

Goals of this talk

- Become a departmental celebrity.
- Serve ~~Dutch stroopwafels~~ Belgian cookies.
- Popularise *Quarto!*
- Legitimise hobby project.
- Fun and empowering toolbox:
    - Combinatorial game theory
    - Academic programming
- Nice example of brain vs computational power:
    - Thought-assisted combinatorial search
    - Combinatorial-search-assisted thought
- Fascinating symmetries

Goals of this talk

- Become a departmental celebrity. Success!
- Serve ~~Dutch stroopwafels~~ Belgian cookies. Success!
- Popularise *Quarto!*
- Legitimise hobby project.
- Fun and empowering toolbox:
  - Combinatorial game theory
  - Academic programming
- Nice example of brain vs computational power:
  - Thought-assisted combinatorial search
  - Combinatorial-search-assisted thought
- Fascinating symmetries

Outline

1. Quarto crash course

2. The value of Quarto

3. Playing Optimally

Rules: the pieces and Quarto

- The **pieces** are the 16 realisations of four binary properties:

$$\underbrace{\{\text{dark}, \text{light}\}}_{\text{colour}} \times \underbrace{\{\text{tall}, \text{short}\}}_{\text{height}} \times \underbrace{\{\text{round}, \text{square}\}}_{\text{shape}} \times \underbrace{\{\text{hollow}, \text{solid}\}}_{\text{consistency}}$$

- Four pieces form **Quarto** if they agree on a property.

$$Q\{p, q, r, s\} \qquad \text{iff} \qquad p_i = q_i = r_i = s_i \quad \text{for some property } i$$

Rules: the pieces and Quarto

- The **pieces** are the 16 realisations of four binary properties:

$$\underbrace{\{\text{dark}, \text{light}\}}_{\text{colour}} \times \underbrace{\{\text{tall}, \text{short}\}}_{\text{height}} \times \underbrace{\{\text{round}, \text{square}\}}_{\text{shape}} \times \underbrace{\{\text{hollow}, \text{solid}\}}_{\text{consistency}}$$

- Four pieces form **Quarto** if they agree on a property.

$$Q\{p, q, r, s\} \qquad \text{iff} \qquad p_i = q_i = r_i = s_i \quad \text{for some property } i$$

Rules: the pieces and Quarto

- The **pieces** are the 16 realisations of four binary properties:

$$\underbrace{\{\text{dark}, \text{light}\}}_{\text{colour}} \times \underbrace{\{\text{tall}, \text{short}\}}_{\text{height}} \times \underbrace{\{\text{round}, \text{square}\}}_{\text{shape}} \times \underbrace{\{\text{hollow}, \text{solid}\}}_{\text{consistency}}$$

- Four pieces form **Quarto** if they agree on a property.

$$Q\{p, q, r, s\} \qquad \text{iff} \qquad p_i = q_i = r_i = s_i \quad \text{for some property } i$$

Rules: the pieces and Quarto

- The **pieces** are the 16 realisations of four binary properties:

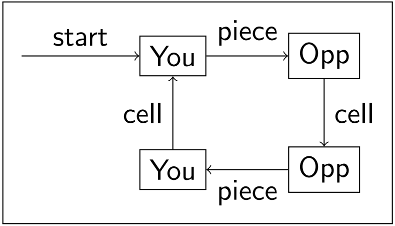
$$\underbrace{\{\text{dark}, \text{light}\}}_{\text{colour}} \times \underbrace{\{\text{tall}, \text{short}\}}_{\text{height}} \times \underbrace{\{\text{round}, \text{square}\}}_{\text{shape}} \times \underbrace{\{\text{hollow}, \text{solid}\}}_{\text{consistency}}$$

- Four pieces form **Quarto** if they agree on a property.

$$Q\{p, q, r, s\} \qquad \text{iff} \qquad p_i = q_i = r_i = s_i \quad \text{for some property } i$$

Rules: board, turns and winning

- The **board** has 4×4 cells. Initially empty. Pieces are put aside.
- The game proceeds in **rounds**. Each round has two **plies**:
  - One player gives an unused piece to the other player.
  - The other player places that piece on an empty cell.



- **Win** by forming Quarto in a row, column or (co)diagonal.
- **Draw** when all pieces placed without Quarto.

Questions

- What is the *value of the game*? (i.e. when both players play optimally, does the starting player win, lose or draw?)
- How to play the optimal strategy?

Outline

Naive approach

$$\max_{p_1} \min_{c_1} \min_{p_2} \max_{c_2} \max_{p_3} \ldots \min_{p_{16}} \max_{c_{16}} V(p_1 c_1 \cdots p_{16} c_{16})$$

where

$$V(p_1 c_1 \cdots p_{16} c_{16}) = \begin{cases} -\infty & \text{You disobeyed the rules} \\ -1 & \text{You lose} \\ 0 & \text{Game is a draw} \\ +1 & \text{You win} \\ +\infty & \text{Opp disobeyed the rules} \end{cases}$$

Only $16^{32} \approx 3.4 \cdot 10^{38}$ operations.

Naive approach

$$\max_{p_1} \min_{c_1} \min_{p_2} \max_{c_2} \max_{p_3} \ldots \min_{p_{16}} \max_{c_{16}} V(p_1 c_1 \cdots p_{16} c_{16})$$

where

$$V(p_1 c_1 \cdots p_{16} c_{16}) = \begin{cases} -\infty & \text{You disobeyed the rules} \\ -1 & \text{You lose} \\ 0 & \text{Game is a draw} \\ +1 & \text{You win} \\ +\infty & \text{Opp disobeyed the rules} \end{cases}$$

Only $16^{32} \approx 3.4 \cdot 10^{38}$ operations.
Only $(16!)^2 \approx 4.4 \cdot 10^{26}$ when enforcing the rules.

Naive approach

$$\max_{p_1} \min_{c_1} \min_{p_2} \max_{c_2} \max_{p_3} \ldots \min_{p_{16}} \max_{c_{16}} V(p_1 c_1 \cdots p_{16} c_{16})$$

where

$$V(p_1 c_1 \cdots p_{16} c_{16}) = \begin{cases} -\infty & \text{You disobeyed the rules} \\ -1 & \text{You lose} \\ 0 & \text{Game is a draw} \\ +1 & \text{You win} \\ +\infty & \text{Opp disobeyed the rules} \end{cases}$$

Only $16^{32} \approx 3.4 \cdot 10^{38}$ operations.
Only $(16!)^2 \approx 4.4 \cdot 10^{26}$ when enforcing the rules.
Way too many.

Naive approach

$$\max_{p_1}\min_{c_1}\min_{p_2}\max_{c_2}\max_{p_3}\ldots\min_{p_{16}}\max_{c_{16}} V(p_1 c_1 \cdots p_{16} c_{16})$$

where

$$V(p_1 c_1 \cdots p_{16} c_{16}) = \begin{cases} -\infty & \text{You disobeyed the rules} \\ -1 & \text{You lose} \\ 0 & \text{Game is a draw} \\ +1 & \text{You win} \\ +\infty & \text{Opp disobeyed the rules} \end{cases}$$

Only $16^{32} \approx 3.4 \cdot 10^{38}$ operations.
Only $(16!)^2 \approx 4.4 \cdot 10^{26}$ when enforcing the rules.
Way too many. Q: Any ideas?

Exploiting positionality

In Quarto, the moves from and payoffs in any state depend only on the current position, and not on how the players got there.

function $\textsc{Val}(b)$
    if $\textsc{isQ}(b)$ **return** WIN
    if $\textsc{isFull}(b)$ **return** DRAW
    if we stored that b has value v then **return** v
    if b has given piece p **then**
        $v \leftarrow \max_{c \in cells(b)} \textsc{Val}(b[p@c])$
    else
        $v \leftarrow \max_{p \in pieces(b)} -\textsc{Val}(b \oplus p)$
    end if
    store that b has value v
    return v
end function

We now need $9.9 \cdot 10^{16}$ operations. Still no cigar.

Exploiting symmetries

Some positions are *equivalent*. It suffices to evaluate only one member of each equivalence class.

- Piece symmetries
- Board symmetries

Piece symmetries

Definition (Piece Symmetry)

A *piece symmetry* is a mapping of the 16 pieces to the 16 pieces that preserves Quarto's.

Piece symmetries

Definition (Piece Symmetry)

A *piece symmetry* is a mapping of the 16 pieces to the 16 pieces that preserves Quarto's.

Q: Find piece symmetries

Piece symmetries

Definition (Piece Symmetry)

A *piece symmetry* is a mapping of the 16 pieces to the 16 pieces that preserves Quarto's.

Q: Find piece symmetries

Fact

There are $4!\,2^4 = 384$ *piece symmetries.*

- *the* 4 *properties can be reordered arbitrarily*
- *the* 2 *values of each property can be flipped*

Board s

Definition (Board Symmetry)

A *board symmetry* is a mapping of the 16 board cells to the 16 board cells that preserves Quarto's.

A board symmetry must map rows/columns to rows/columns and (co)diagonals to (co)diagonals.

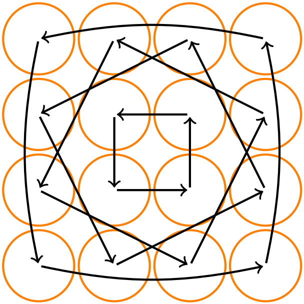
Definition (Board Symmetry)

A *board symmetry* is a mapping of the 16 board cells to the 16 board cells that preserves Quarto's.

A board symmetry must map rows/columns to rows/columns and (co)diagonals to (co)diagonals.

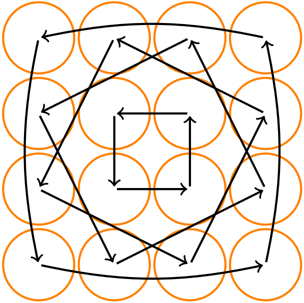
Q: Find board symmetries

Finding board symmetries



counter clockwise rotation

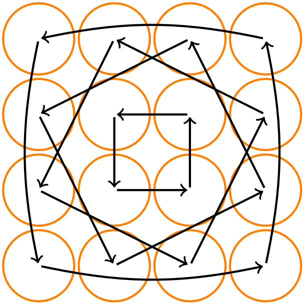
Finding board symmetries



counter clockwise rotation

Q: What about clockwise rotation?

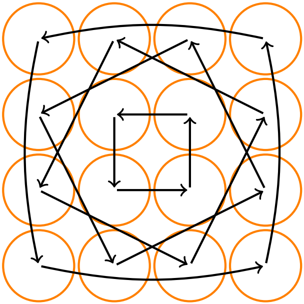
Finding board symmetries



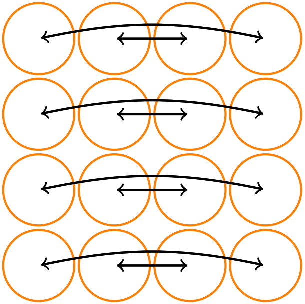
counter clockwise rotation

Q: What about clockwise rotation? A: Rotate ccw thrice

Finding board symmetries



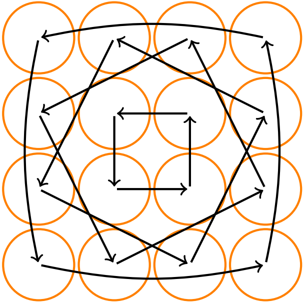
counter clockwise rotation



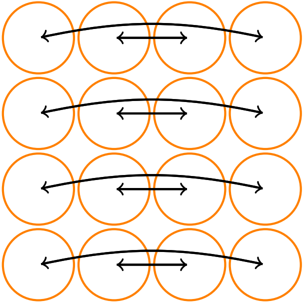
mirror over vertical axis

Q: What about clockwise rotation? A: Rotate ccw thrice

Finding board symmetries



counter clockwise rotation

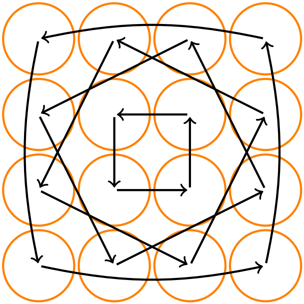


mirror over vertical axis

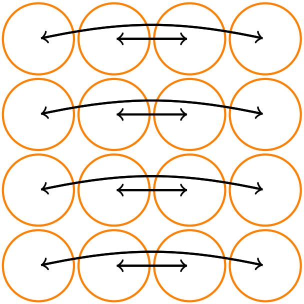
Q: What about clockwise rotation? A: Rotate ccw thrice

Q: Mirror over diagonal?

Finding board symmetries



counter clockwise rotation

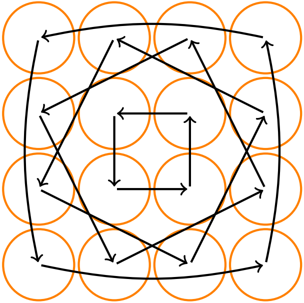


mirror over vertical axis

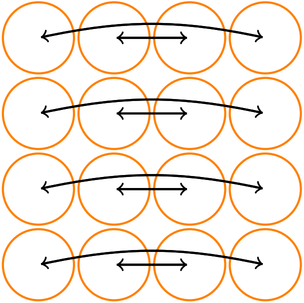
Q: What about clockwise rotation? A: Rotate ccw thrice

Q: Mirror over diagonal? A: rotate cw, then mirror

Finding board symmetries



counter clockwise rotation



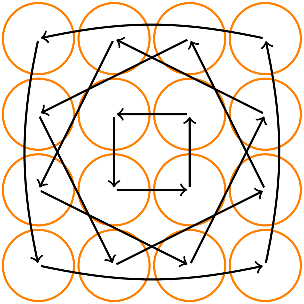
mirror over vertical axis

Q: What about clockwise rotation? A: Rotate ccw thrice

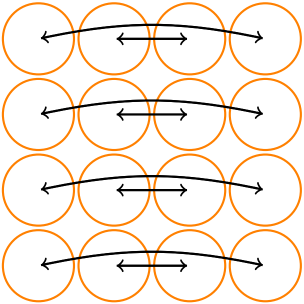
Q: Mirror over diagonal? A: rotate cw, then mirror

Q: Are there other board symmetries?

Finding board symmetries



counter clockwise rotation



mirror over vertical axis

Q: What about clockwise rotation? A: Rotate ccw thrice

Q: Mirror over diagonal? A: rotate cw, then mirror

Q: Are there other board symmetries?

Q: How to even approach such a question?

Exhaustive enumeration

```
procedure ENUM_SYM(M)
    if M violates group structure then return
    if |M| = 16 then
        print M
    else
        choose a free source cell i
        for each free target cell j do
            ENUM_SYM(M[i → j])
        end for
    end if
end procedure
```

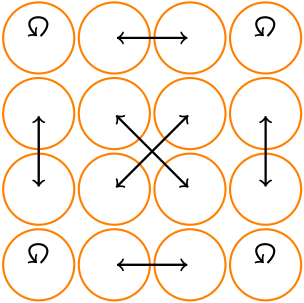
Exhaustive enumeration

```
procedure ENUM_SYM(M)
    if M violates group structure then return
    if |M| = 16 then
        print M
    else
        choose a free source cell i
        for each free target cell j do
            ENUM_SYM(M[i → j])
        end for
    end if
end procedure
```

Fact

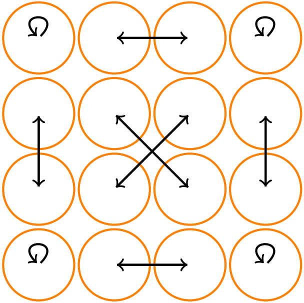
There are 32 board symmetries.

Finding board symmetries (ctd)

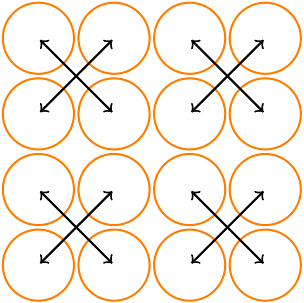


mid flip

Finding board symmetries (ctd)

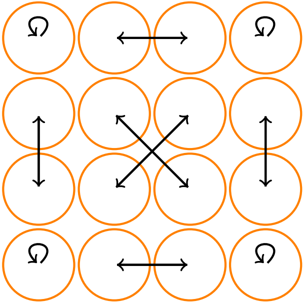


mid flip

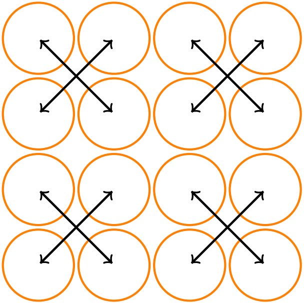


inside out

Finding board symmetries (ctd)



mid flip

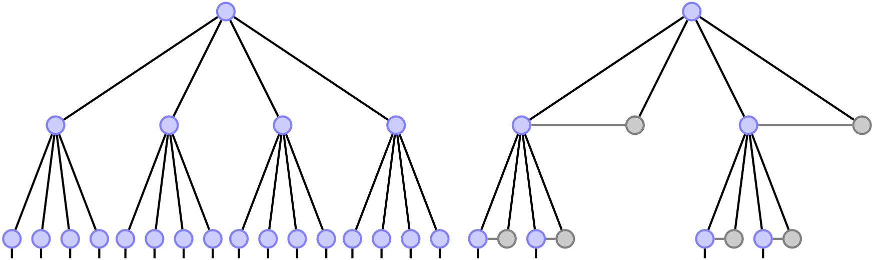


inside out

[Goo04] found 16 (inside out), and [Bro05] found 16(mid flip).

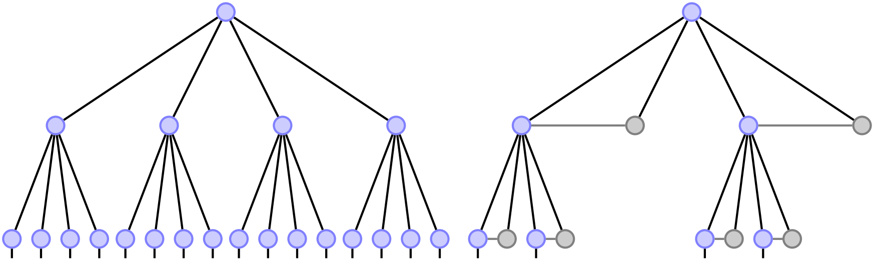
Symmetry benefits

Identifying d positions may *divide* the degree by d. Exponential gain.



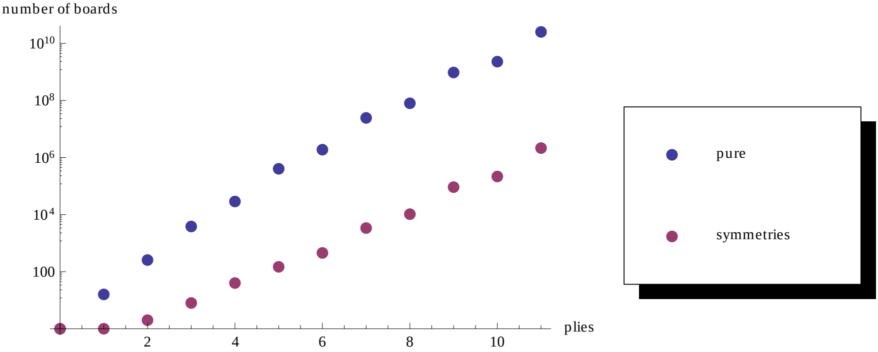
Symmetry benefits

Identifying d positions may *divide* the degree by d. Exponential gain.



Q: Will it always? If not, what is the least-favourable reduction?

Symmetry benefits ctd.



Good, but we need to explore 32 plies!

How to exploit symmetries

CANONISE picks a canonical representative of each equivalence class.

function VAL(b)
    if ISQ(b) **return** WIN
    if ISFULL(b) **return** DRAW
    $b \leftarrow$ CANONISE(b)
    if we stored that b has value v then **return** v
    if b has given piece p **then**
        $v \leftarrow \max_{c \in cells(b)} \text{VAL}(b[p@c])$
    else
        $v \leftarrow \max_{p \in pieces(b)} -\text{VAL}(b \oplus p)$
    end if
    store that b has value v
    return v
end function

The final trick

Alpha-beta pruning. Rule of thumb: explores only the square root of the original number of positions.

The final trick

Alpha-beta pruning. Rule of thumb: explores only the square root of the original number of positions.

Fact

The value of Quarto is draw.

Software finds out in 147 minutes on this laptop.

Outline

Implementing the optimal strategy

- So far, we computed the value of the empty board.
- But to play, we need to evaluate any board.
- We can evaluate positions at ≥ 10 plies from scratch in < 5 seconds.
- There are only 106 156 distinct positions at < 10 plies.
- Compute, once and for all, the value of *all of them*.
- Took about 2 weeks on this laptop.
- Now we can evaluate any position fast.
- To play, choose the child that evaluates to the value of the parent.

Kevin S. Brown.
414298141056 quarto draws suffice!
http://www.mathpages.com/home/kmath352.htm, June 2005.

Luc Goossens.
Quarto.
http://web.archive.org/web/20041012023358/http://ssel.vub.ac.be/Members/LucGoossens/quarto/quartotext.htm, October 2004.