Enumerative Combinatoric Algorithms

Combinatorial 2 Player Games

Connect-4 aka 4 Gewinnt
Examples: Tik Tak Toe (11)

Enumerate all **different** valid game positions of Tic Tac Toe after $k$ half-moves, considering symmetries.

$\circ$ starts, $\times$ follows $\rightarrow \left\lfloor \frac{k+1}{2} \right\rfloor \circ$, and $\left\lfloor \frac{k}{2} \right\rfloor \times$ tokens are placed after $k$ half-moves.
Examples: Tik Tak Toe (11)

Storing a board:

2 bit per square:

\[ 2 \times 9 = 18 \text{ bit, thus } 2^{18} = 262144 \text{ possible boards.} \]

3 possibilities per square:

\[ 3^9 = 19683 \text{ possible boards with } \lceil \log_2 3^9 \rceil = 15 \text{ bit}. \]
Examples: Tik Tak Toe (11)

<table>
<thead>
<tr>
<th>$n$ half-moves</th>
<th>game-tree</th>
<th>different boards</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>504</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3024</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15120</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>60480</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>181440</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>362880</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>362880</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>362880</td>
<td></td>
</tr>
</tbody>
</table>

branching factor: 9,8,7,...,2,1
Examples: Tik Tak Toe (11)

<table>
<thead>
<tr>
<th>$n$ half-moves</th>
<th>game-tree</th>
<th>different boards</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>504</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>3024</td>
<td>108</td>
</tr>
<tr>
<td>5</td>
<td>15120</td>
<td>174</td>
</tr>
<tr>
<td>6</td>
<td>60480</td>
<td>228</td>
</tr>
<tr>
<td>7</td>
<td>181440</td>
<td>174</td>
</tr>
<tr>
<td>8</td>
<td>362880</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>362880</td>
<td>23</td>
</tr>
<tr>
<td>sum</td>
<td>362880</td>
<td>850</td>
</tr>
</tbody>
</table>

- $362880 = \text{game-tree complexity}$
- $262144 = 2^{18}$
- $19683 = 3^9$
- $850$ different boards $= \text{state space complexity}$
Basics on 2 Player Games

- We consider 2 player perfect information games. The players are called \textit{first player (Alice)} and \textit{second player (Bob)}. 
- No hidden information, no randomness or chance, both players have all information.
- Players play in turns (not simultaneously).
- There is a finite number of game states (but a game might last forever which is considered a draw).
- Note that the game might be asymmetric, i.e., Alice and Bob have different tasks (e.g. Fox and Geese).
Basics for Computer Playing

- A game state needs to be stored memory efficient and complete (coding and decoding of states including player information etc.).
- Move generator (successors of a game state).
- Identify final states: win, lose, and draw states.
- Backwards move generator (predecessors of a game state). Not always possible.

Game states are important!
Different game states?

Two game states are equivalent, if they allow the same moves (w.r.t. the state), resulting in the same successor states (w.r.t. the state). Typically reflection, rotation, inversion, color-change, ... can be applied.

We only need to store the move information for one of the equivalent states (canonical state, fingerprint), as for all other states it follows by the equivalence operations.
Nine Men’s Morris (aka Mühle) (12)

Enumerate all different valid game positions for Nine Men’s Morris (number of non-equivalent states)...

How many non-equivalent states exist after 2 white tokens and 1 black token have been placed (white player starts)?

(perfect play always results in a draw)
Nine Men’s Morris (aka Mühle) (12)

How many non-equivalent states exist after 2 white tokens and 1 black token have been placed (white player starts)?

Pólya-Redfield Enumeration Theorem: 16 Operations:

$R_0$: ID: $r_0 = \binom{24}{2} \times 22 = 6072$

$R_1$ Rotation $90^\circ$ ($R_3$ Rotation $270^\circ$): $r_1 = r_3 = 0$

$R_2$ Rotation $180^\circ$: $r_2 = 0$

$R_4 \ldots R_7$ Reflections: $r_4 = \ldots = r_7 = 6 \times (9 + \binom{5}{2}) = 114$

$R_8$: In-Out Inversion: $r_8 = 8 \times (8 + \binom{7}{2}) = 232$

$R_9 \ldots R_{15}$: In-Out-Inversion plus $R_1 \ldots R_7$

$r_9 = r_{10} = r_{11} = 0$

$r_{12} = \ldots = r_{15} = 2 \times 11 = 22$

Number of orbits\(=\frac{6072 + 4 \times 114 + 232 + 4 \times 22}{16} = \frac{6848}{16} = 428\)
Levels of Game Solutions

- **Ultra-weakly solved**: We know which player can win, but not how (no strategy! Example: Chomp)
- **Weakly solved**: A strategy is known (from a start situation following the strategy)
- **Strongly solved**: A strategy is known from any valid state.
- **Ultra-strongly solved**: For any valid game state and any possible move it is known whether it is a win, draw or lose and in how many half-moves this happens.

We aim for ultra-strongly solved!
Connect-4

http://connect4.ist.tugraz.at:8080

Connect Four

On the move: Player B

Menu
- Restart game
- Toggle move infos
- Recommend move
- Undo last move
- Redo last move
- Save game...
- Load game...
- Delete game...

Options
- AI for Player A
- AI Level A: Perfect
- AI for Player B
- AI Level B: Perfect

History
1. Player A in col 4 (row 1) (Win in 40)

History Redo

Game has started at 2015-05-14 20:55:11
Game-Tree vs. State-Space Complexity

- **Game-Tree Complexity** Number of nodes the complete decision tree for a whole game has.

- **State-Space Complexity** Number of states which can be reached from the start state by valid moves.
Game-Tree vs. State-Space Complexity

- **Game-Tree Complexity** Number of nodes the complete decision tree for a whole game has

- **State-Space Complexity** Number of states which can be reached from the start state by valid moves

For most games a state might be reachable via many different sequences of valid moves. C cycles might even result in unbounded possibilities.
Game-Tree vs. State-Space Complexity

- **Game-Tree Complexity** Number of nodes the complete decision tree for a whole game has.

- **State-Space Complexity** Number of states which can be reached from the start state by valid moves.

For most games a state might be reachable via many different sequences of valid moves. C cycles might even result in unbounded possibilities.
Game-Tree vs. State-Space Complexity

- **Game-Tree Complexity** Number of nodes the complete decision tree for a whole game has

- **State-Space Complexity** Number of states which can be reached from the start state by valid moves

<table>
<thead>
<tr>
<th>game</th>
<th>state-space complexity</th>
<th>game-tree complexity</th>
<th>branching factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tic Tac Toe</td>
<td>$10^3$</td>
<td>$10^5$</td>
<td>5</td>
</tr>
<tr>
<td>Nine Men’s Morris</td>
<td>$10^{10}$</td>
<td>$10^{50}$</td>
<td>10-30</td>
</tr>
<tr>
<td>Pyraos</td>
<td>$10^{11}$</td>
<td>$10^{33}$</td>
<td>9</td>
</tr>
<tr>
<td>Awari</td>
<td>$10^{12}$</td>
<td>$10^{32}$</td>
<td>5-6</td>
</tr>
<tr>
<td>Connect-4</td>
<td>$10^{14}$</td>
<td>$10^{21}$</td>
<td>5-7</td>
</tr>
<tr>
<td>Abalone</td>
<td>$10^{25}$</td>
<td>$10^{180}$</td>
<td>65-70</td>
</tr>
<tr>
<td>Reversi</td>
<td>$10^{28}$</td>
<td>$10^{58}$</td>
<td>5-15</td>
</tr>
<tr>
<td>Chess</td>
<td>$10^{50}$</td>
<td>$10^{123}$</td>
<td>35</td>
</tr>
<tr>
<td>Go</td>
<td>$10^{171}$</td>
<td>$10^{360}$</td>
<td>300-400</td>
</tr>
</tbody>
</table>
Enumerate all states

We store all (non-equivalent) states in a set $S$, i.e., our approach is based on the state space complexity.

Initialize $S$ with the starting state.

$\forall$ non-processed states $s \in S$ DO

/* process newly added states */

$\forall$ successors $t$ of $s$ DO

compute canonical state $t'$ of $t$

IF $t' \notin S$ THEN add $t'$ to $S$

/* $S$ contains all states which are reachable from the start state via valid moves */
'Code' of a State

For every game state we compute a code (integer number ≥ 0) which contains all information when playing starts/continues from this state.

- **WIN**: code odd: number of half-moves in which a win can be forced (if player plays perfect).
- **LOSE**: code even: number of half-moves in which the game is at most lost (if opponent plays perfect).
- **DRAW**: special code, e.g. -1; no number of half-moves possible.
How to compute 'Codes'

Init all states without valid moves with code := 0 or draw
/* terminal lose states */

IF successor state with even code existe THEN
code := (smallest even code of a successor state) + 1
/* WIN in that number of moves */
ELSE IF successor state with draw code exists THEN
code := draw
/* DRAW */
ELSE
code := (largest (odd) code of a successor state) + 1
/* LOSE in that number of moves */
Pseudocode to compute 'Codes'

Init all states without valid moves

/* terminal states without successors */

Init all remaining states with 'undefined'

FOR $k := 1$ TO max-depth /* $k = \# \text{ of half-moves} */

   $\forall$ states $s \in S$ with still undefined code DO

   IF $k$ is odd THEN

       IF $s$ has a successor with code $k - 1$ THEN

           code of $s$ is $k$ /* WIN state */

       ELSE /* $k$ is even */

           IF all successors of $s$ have odd codes THEN

               code of $s$ is $k$ /* LOSE state */

       ELSE /* $k$ is even */

   Set all 'undefined' states to draw.
How to use the 'Code'

How to play for a current state $s$:

- Compute all possible successors of $s$ and their codes.

- IF a successor with even code exist, make the move which leads to the successor with the smallest even code $k$. Message: ”I will win in $k$ half-moves.”.

- ELSE IF a successor with code draw exists, make the draw move. Message: ”You might make a draw.”.

- ELSE make the move to the successor with the highest (odd) code $k$. Message: ”You might win in $k$ half-moves.”.
Saving a state of Connect-4

With how many (well, few) byte can you store a game state of Connect-4? You are allowed to use the information on how many half-moves (0 to 42) have already been made.
Computer Playing: Connect-4

- All game states need to be stored memory efficient and complete: DONE!

- Move generator (successors of a game state): Just add a token in a non-full column. At most 7 successors exist.

- Identify final states: For lose (i.e., previous player win) just check up to 11 4-tuples (including new token). No win and 42 tokens placed: terminal draw state.

- For efficiency: backwards move generator (predecessors of a game state): ???
Connect-4: no backward moves

Not a valid position! For general sized boards it is NP-complete to decide if a position is valid.
## Connect-4: Number of States

<table>
<thead>
<tr>
<th>half-moves</th>
<th>different boards</th>
<th>half-moves</th>
<th>different boards</th>
<th>half-moves</th>
<th>different boards</th>
<th>half-moves</th>
<th>different boards</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8</td>
<td>91295</td>
<td>16</td>
<td>177841160</td>
<td>8</td>
<td>91295</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>269531</td>
<td>17</td>
<td>363798195</td>
<td>9</td>
<td>269531</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>10</td>
<td>809464</td>
<td>18</td>
<td>767435580</td>
<td>10</td>
<td>809464</td>
</tr>
<tr>
<td>3</td>
<td>121</td>
<td>11</td>
<td>2148087</td>
<td>19</td>
<td>1448894267</td>
<td>11</td>
<td>2148087</td>
</tr>
<tr>
<td>4</td>
<td>568</td>
<td>12</td>
<td>5832236</td>
<td>20</td>
<td>2818993420</td>
<td>12</td>
<td>5832236</td>
</tr>
<tr>
<td>5</td>
<td>2144</td>
<td>13</td>
<td>14105207</td>
<td>21</td>
<td>4907390200</td>
<td>13</td>
<td>14105207</td>
</tr>
<tr>
<td>6</td>
<td>8231</td>
<td>14</td>
<td>35045629</td>
<td>22</td>
<td>8788132016</td>
<td>14</td>
<td>35045629</td>
</tr>
<tr>
<td>7</td>
<td>27109</td>
<td>15</td>
<td>77785047</td>
<td>23</td>
<td>14066554884</td>
<td>15</td>
<td>77785047</td>
</tr>
</tbody>
</table>

| sum        | 33475164421      |

33475164421 states with 6 byte each: 200 GB + 34 GB
Connect-4: Current Data Base

Storing all possible positions?

- 2 bit per square: $2^{2 \times 6 \times 7} = 2^{84} \approx 2 \times 10^{25}$ states:
  1 byte each, 17592186000000 TB.

- 3 possibilities per square: $3^{6 \times 7} = 3^{42} \approx 10^{20}$ states:
  1 byte each, 99515990 TB.

- 6 byte per state: $2^{6 \times 8} = 281474976710656 \approx 10^{14}$
  states: 1 byte each, 281 TB.

Storing all possible states up to 23 half-moves: 234GB. Maximal remaining search depth: $42 - 23 = 19$, with $\approx 5$
possible moves in average.
Summary

- Aim for ultra-strongly solved games.
- Use hybrid approach (space-time tradeoff):
  - Enumerate/code states for a limited number of half-moves
  - Evaluate remaining end-games via the game-tree
- Provide all possible kind of information to fully analyse games: connect-4 is a first player win (play column 4) in 41 half-moves, ...
- Future: Find compact representation of data base via a small set of rules with list of exceptional states.