Application of UCT Search to the Connection Games of Hex, Y, *Star, and Renkula!

Tapani Raiko and Jaakko Peltonen
Helsinki University of Technology
Traditional min-max search

```
min
1 2 3 2 4 2 2 3
```
Traditional min-max search
Traditional min-max search
Traditional min-max search

- Requires a fast evaluation function
- Typically equally deep for each branch
- Alpha-beta pruning etc. allow for deeper search
Connection Games

- Connection games are abstract board games where connectivity of game pieces is crucial.

- In all of the games considered here:
  - The board is initially empty.
  - Two players alternately place a piece of their own color to an empty point.
  - When the board is full, the exactly one of the players has met a winning criterion.
Game of Hex

- The goal for black is to connect the top and the bottom edges.
- White tries to connect the left and right edges.
Game of Y

- Both players try to connect all three edges with a single unbroken chain.
Game of Renkula!

- First published here
- Pieces are placed two at a time to exact opposites of the sphere
- Connecting any such pair with an unbroken chain gives a win
What can we infer from the rules?

- **Note 1:** A winning chain will always form a loop around the sphere.
- **Note 2:** If one of the players has formed a winning chain, the other player could no longer form a winning chain even if the game continued.
- **Note 3:** When the sphere is filled with stones, one of the players must have made a winning chain.
- **Note 4:** With perfect play, red can always win.
Different board sizes

<table>
<thead>
<tr>
<th>Board 1</th>
<th>Board 2</th>
<th>Board 3</th>
<th>Board 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>42 polygons:</td>
<td>92 polygons:</td>
<td>162 polygons:</td>
<td>362 polygons:</td>
</tr>
<tr>
<td>12 pentagons</td>
<td>12 pentagons</td>
<td>12 pentagons</td>
<td>12 pentagons</td>
</tr>
<tr>
<td>30 hexagons</td>
<td>80 hexagons</td>
<td>150 hexagons</td>
<td>350 hexagons</td>
</tr>
</tbody>
</table>
A tree search like before, but

- Evaluations of the game state are not needed
- Instead, the game is played randomly to the end, giving a random evaluation of a state
- The tree is grown one node at a time (like in best first search)
Tree grows by one node per play-out
Which node?

- In state $s$ within the tree, the node $a$ with the highest upper confidence bound $u(s,a)$ on the expected reward is chosen

$$u(s, a) = r(s, a) + c \sqrt{\frac{\log n(s)}{n(s, a)}},$$

- $r(s,a)$ is the current estimate of the reward
- $n(s,a)$ is the count of how many times the action $a$ has been chosen in state $s$ out of $n(s)$ times the state has been visited
- $c$ is a constant for which we used the value 1
Properties of UCT

- Play-out analysis avoids the estimation of a game state

- In connection games, the estimation is difficult (compare to piece count in chess)

- Using upper confidence gives a balance between exploration and exploitation: actions with good reward are chosen more often, but actions that are not explored much become interesting as the confidence is low
Heuristics for Connection Games

- Playing the game to the end in these games is equivalent to filling out the rest of the board with random colored pieces - this is faster.

- For the latest leaf node it does not make any difference which of the fill-out moves is counted as the first one - we can update all of them at once!

- As the fill-out phase is fast, it can be useful to do more than one fill-out at once.
Bamboo connection heuristic

- Bamboo connections are a simple shape that reappears very often in these games.
- Connection can be kept intact and it is often wise to do so.
- We recognize the shape and fill them with one stone of each color - this makes the program play stronger.
Try them out!

- Implementation for Renkula! is available at www.nbl.fi/~nbl924/renkula/
- Implementations of Hex, Y, and *Star are at www.cis.hut.fi/praiko/connectiongames/