Imperfect minimax search algorithm

1: procedure IMPERFECTMINIMAXSEARCH\((G, p, d)\)
2: if \(G\) is finished or \(d = 0\) then
3: return Evaluate\((G, p)\)
4: end if
5: \(s_{\text{max}} \leftarrow -\infty\)
6: for \(m \in \text{supp}\left(P_G(:|p)\right)\) such that \(m\) is valid in \(G\) do
7: \(q \leftarrow P_G(m|p)\)
8: \(G' \leftarrow G\) updated with move by \(m\) played by \(p\)
9: \(p' \leftarrow\) next player after \(p\) performs \(m\) in game \(G\)
10: \(s_{\text{max}} \leftarrow \max\{s_{\text{max}}, q \cdot \text{ImperfectMinimaxSearch}\((G', p', d-1)\)\}
11: end for
12: return \(s_{\text{max}}\)
13: end procedure

Results

Using the above algorithm and a deepened search after each turn, we received the following results for a team of AI versus a team of the simplest non-trivial strategy: greedy—i.e. dump the domino with the highest value.

<table>
<thead>
<tr>
<th>Wins</th>
<th>Losses</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>70.1%</td>
<td>28.1%</td>
<td>1.8%</td>
</tr>
</tbody>
</table>

We iteratively deepened the search every \(N/4\) plays in an exponential fashion (as the number of possible moves exponentially decreased) and left it as a tunable parameter. The depth was of the form

\[ D = \alpha 2^\beta |\bar{x}| \]

where we allowed \(\alpha = 6, \beta = 1/2\).