Heuristic Analysis

Stefan Heidekrüger

July 3, 2017

This document contains some analysis on the position evaluation heuristics that have been tried and ultimately selected for the *Knight-Isolation* game playing agent. We will briefly describe the methodology as well as the findings of our analysis.

Methodololgy

The goal of an evaluation heuristic v is to approximate the *value* v^* of the game (from the point of view of the agent player.) As such, it's always the goal to win the game and avoid losing it, so it makes sense to set $v(\text{winning position}) = \infty$ and $v(\text{losing position}) = -\infty$. (These values are well-defined, as our alpha-beta-tuning algorithm solely relies on *ordinal* comparisons of values; if we instead employed Monte-Carlo tree search or other similar algorithms, we would need to use finite values in order to *weigh* moves somehow —and also worry about interval scaling.)

More importantly, let's consider position that are neither clearly winning or losing. We decided that the heuristic should be primarily decided based on the *number of possible moves for the playing agent* n and the *number of possible moves for the opposing player* m. Other measures, such as centrality, were briefly considered but were quickly discarded when they didn't show much promise in the first experiments.

As it's obvious that m should affect the heuristic negatively (and n positively), we considered polynomial functions of the form

$$v(n, m) = an^b - cm^d$$

with $0 \leq a, c \lessapprox 10$, $0 < b, d \lessapprox 3$ (a value of 0 for b or d would be irrelevant due to translation invariance).

Due to time-constraints, we did not perform an exhaustive parameter-search in this space, but rather manually tried different combinations for a very superficial search by hand. In each case, decisions were made on a handful of cases of round-robin tournaments as provided in `tournameny.py` as no resources for more robust methods were available. It should be noted that the results contained "a lot" of variance and any analysis result should therefore be taken with a grain of salt and only be seen as a rough draft rather than thoroughly tested insights on valuable heuristics. The given implementation `AB-Improved` served as the main baseline for comparison

Results and Final Heuristics

From the first few experiments, it quickly became clear that the following should hold:

- $a > 0, c > 0$
- n should be weighed stronger than m, roughly speaking $cm^d = \mathcal{O}(an^d)$

After some experimentation, we finally considered the following heuristics (along with their win-rates over 5 consecutive tournaments):

- v_0: baseline `AB-Improved` (61.4, 57.1, 61.4, 67.1, 61.4, avg: 61.7)
- $v_1 = 2n^2 - m\sqrt{m}$ (68.6, 57.1, 67.1, 67.1, 64.3, avg: 64.84)
- $v_2 = n\sqrt{n} - 2m$ (60.0, 54.3, 65.7, 72.9, 54.3, avg: 61.44)
- $v_3 = 2n - m$[1] (64.3, 65.7, 55.7, 57.1, 60.0, avg: 60.56)

We can see that from this list, v_1 not only achieves the highest average score —it also (weakly) dominates the baseline player across these tournaments. As such, we chose

$$v_1 = 2n^2 - m^{1.5}$$

as our final heuristic.

[1]With additional check: If only possible move can be blocked in opponents next move, then $v_3 = -100$