Tablut Challenge

Giuliani Luca
Lombardi Alessandro
Mazzieri Diego

Norsemen

University of Bologna.
Master Degree in Artificial Intelligence

16/12/2019

Table of Contents

Who we are

Meet us on Netflix.

Workflow

The Tool

Gradle

The Language

Kotlin

The Algorithm

AI
MA

The Algorithm

Implementation

Minimax with Alpha-Beta cuts

Game states as nodes of the tree, actions as arcs

Search Strategy

Iterative Deepening because:

- Don't have to specify a depth
- Can take advantage of all available time
- Gets better as the game proceeds

Game Complexity

- **Branching factor** = at least ∼**40** available moves
- **Depth** = ∼**50** turns to reach a terminal state (playing reasonably)

Need of some kind of **heuristic function**...

The Algorithm

Heuristics

Pawns Difference

- **Proportion** of white pawns w.r.t. the number of black ones
- More **aggressive strategy** (both for white and black)
- Useful during the **first phase of the match**

King Surrounded

- **Closest neighbors** over the row and the column occupied by the king
- **Different weights** assigned to different cells depending on the **distance** – closer unit have more weight – and on the neighbor:

  Negative black pawns or citadels when they are next to the king
  Null citadels and throne when they are not next to the king
  Positive white pawns and throne when they are next to the king

- Minimum value between horizontal and vertical weights chosen

The Algorithm

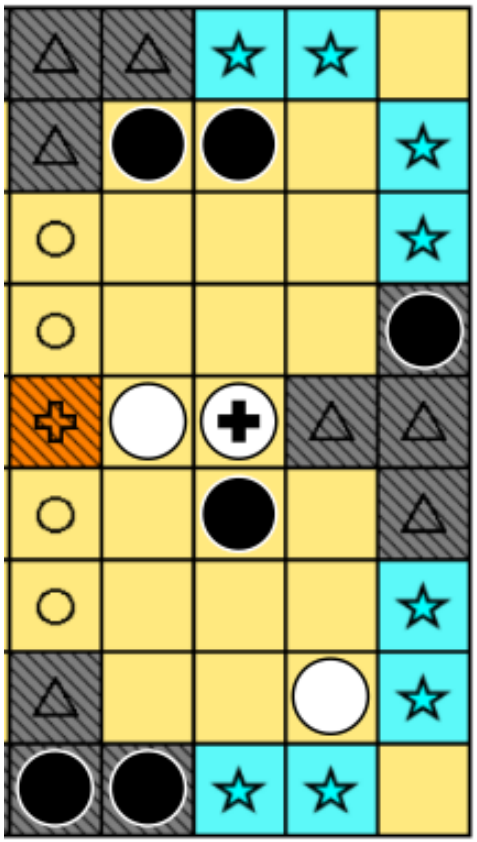
Heuristics

King Distance

- Offline assignment w.r.t. the king's cell $\Rightarrow$ **heat map**, relaxed state
- Leads to prefer cells that are nearer to the **escape cells**
- Naive measure, still:
  - helps to greedily search for **shortest and secure paths**
  - avoids the white player to have a too defensive approach

King Strategic Position

- Offline assignment w.r.t. the king's position $\Rightarrow$ **heat map**, relaxed state
- Leads to prefer cells that can reach an higher number of **escape cells**
- Naive measure, still:
  - helps to move the king **outside of the castle**
  - helps to move the king towards areas without obstacles



Examples

- The **Pawn Difference** heuristic is $\frac{2 * (3 - 1) - 6}{2 * (3 - 1) + 6} = -0.20$
- The **King Surrounded** heuristic is $\frac{min(-7 - 5, 7 - 7)}{7 * 2} = -0.86$
- The **King Distance** heuristic is $1 - 2 * \frac{4}{7} = -0.14$
- The **King Strategic Position** heuristic is $2 * \frac{2}{4} - 1 = 0$

The Algorithm

Heuristics

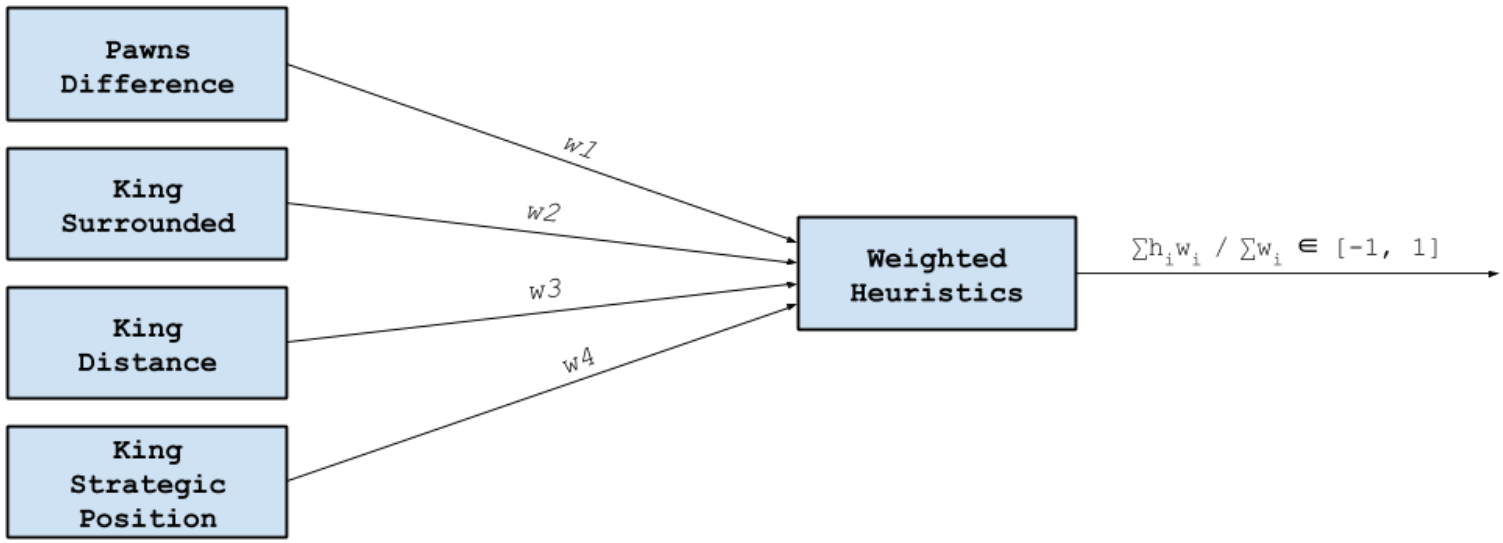
Common Features

Normalization each heuristic falls within the closed interval [-1, 1]

Symmetry the distribution is symmetric (no player is favored)

Weighted Heuristic

- A single strategy is not effective ⇒ **weighted combination** of them



Parameter Tuning

Two Possible Approaches

Manual Tuning

- simpler
- less effective
- requires domain knowledge

Genetic Algorithm

- harder
- more effective
- requires computational power

the computer we had

the computer we needed

We tried our best, yet we failed at running the genetic algorithm

Parameter Tuning

Our guesses

Black vs White

- as the single heuristics are balanced, one could use the same heuristic both for the black and the white player, considering the opposite value for the black one
- still, the two players have different strategies, hence the weights given to the single heuristics can be different

$\Rightarrow$ we came up with two different weighted heuristics

Norsemen White

- $W_{KingDistance} = 1.0$
- $W_{KingStrategicPosition} = 3.0$
- $W_{KingSurrounded} = 2.5$
- $W_{PawnsDifference} = 4.5$

Norsemen Black

- $W_{KingDistance} = 1.0$
- $W_{KingStrategicPosition} = 2.0$
- $W_{KingSurrounded} = 3.0$
- $W_{PawnsDifference} = 5.0$