A description of ***heuristicAri***
The value that is returned in the heuristic is a combination of six pairs of values.

1. The number of pawns for each player
nrOfWhite
nrOfBlack

2. The number of pawns that are protecting one or more pawns
for each player
nrOfWhiteProtectors
nrOfBlackProtectors

3. The number of pawns that threaten some enemy pawn
for each player
nrOfWhiteThreateners
nrOfBlackThreateners

4. Distance of the most advanced pawn to the opposite side
for each player
distMostAdvWhite
distMostAdvBlack

5. The number of pawns that are blocked for each player.
That is pawns that can not move.
nrOfBlockedWhite
nrOfBlockedBlack

6. The number of columns on the board that have only white or only black pawns
nrOfOnlyWhiteColumns
nrOfOnlyBlackColumns

If the agent is black the heuristic returns this value:
50 + (nrOfBlack - nrOfWhite
+ nrOfBlackProtectors - nrOfWhiteProtectors
+ nrOfBlackThreateners - nrOfWhiteThreateners
- distMostAdvBlack + distMostAdvWhite
- nrOfBlockedBlack + nrOfBlockedWhite
+ nrOfOnlyBlackColumns - nrOfOnlyWhiteColumns)

If the agent is white, the opposite value is returned.
50 - (nrOfBlack - nrOfWhite
+ nrOfBlackProtectors - nrOfWhiteProtectors
+ nrOfBlackThreateners - nrOfWhiteThreateners
- distMostAdvBlack + distMostAdvWhite
- nrOfBlockedBlack + nrOfBlockedWhite
+ nrOfOnlyBlackColumns - nrOfOnlyWhiteColumns)

A description of ***heuristicHoddi***
The value that is returned in the heuristic is a combination of six pairs of values.

The number of pawns who are being threatened by enemy pawns.
blackThreatened
whiteThreatened

The number of pawns remaining on the table.
nrOfBlacks
nrOfWhites

The number of pawns who are being defended by other pawns.
blackDefended
whiteDefended

The number of pawns who are protecting other pawns.
blackProtectors
whiteProtectors

The distance of the most advanced pawn to the opposite side.
distMostAdvancedBlack
distMostAdvancedWhite

Number of empty squares infront of each pawn.
blackMobility
whiteMobility

valMostAdv = 1
valAmount = 1
valThreat = 1
valDefend = 1
valProtect = 1
valForward = 1
valMobility = -1

If the agent is black the heuristic return this value:
```
50 + valMostAdv*(distMostAdvancedWhite - distMostAdvancedBlack)
  + valAmount*(-nrOfWhites + nrOfBlacks)
  + valThreat*(whiteThreatened - blackThreatened)
  + valDefend*(blackDefended - whiteDefended)
  + valProtect*(blackProtectors - whiteProtectors)
  + valForward*(blackForward – whiteForward)
  + valMobility*(whiteMobility - blackMobility);
```

If the agent is white it returns the negated value:
```
50 - (valMostAdv*(distMostAdvancedWhite - distMostAdvancedBlack)
  + valAmount*(-nrOfWhites + nrOfBlacks)
  + valThreat*(whiteThreatened - blackThreatened)
  + valDefend*(blackDefended - whiteDefended)
  + valProtect*(blackProtectors - whiteProtectors)
  + valForward*(blackForward – whiteForward)
  + valMobility*(whiteMobility - blackMobility));
```

A description of ***heuristicBinni***
The value that is returned in the heuristic is a combination of four pairs of values.
This was supposed to try and strive to hold as many in a row to combat Ari's heuristic.
Changes got lost in a merge conflict but still it worked well in some cases as black.

1. The number of pawns for each player
nrOfWhite
nrOfBlack

2. The number of pawns that threaten some enemy pawn
for each player
nrOfWhiteThreateners
nrOfBlackThreateners

3. The number of pawns that are defended by some friendly pawn
for each player
blackDefended
whiteDefended

4. Maximun number of pawns in a row for each player
maxWhitesInRow
maxBlacksInRow

If the agent is black the heuristic returns this value:

```
 int valAmount = 1;
 int valThreat = 1;
 int valD;efend = 1
 int valInRow = 1;

      if (role == Role.Black) {
         return 50 + valAmount*(-nrOfWhites + nrOfBlacks)
                  + valThreat*(whiteThreatened - blackThreatened)
                  + valDefend*(blackDefended - whiteDefended)
                           + valInRow*(maxBlacksInRow - maxWhitesInRow);

      }

      else {
         return 50 - ((-nrOfWhites + nrOfBlacks)
                + (whiteThreatened - blackThreatened)
                + (blackDefended - whiteDefended)
                + (maxWhitesInRow - maxBlacksInRow));
      }
```

A description of ***heuristicTryggvi***

Heuristic sem skilar heuristic fyrir fimm pör af gildum vmeð nokkrar vigtir fyrir hvert par.

1. Value of the most advanced pawn for each player, weight = 1
distMostAdvancedWhite
distMostAdvancedBlack

2. The number of pawns for each player, weight = 2
nrOfWhites
nrOfBlacks

3. Number of threatened pawns for each player, weight = 1
whiteThreatened
blackThreatened

4. Number of protects for each player, weight = 3
whiteProtectors
blackProtectors

5. Number of clear forward paths for each pawn for each player, weight = 1
whiteForward
blackForward

Heuristic calculation:
Black agent:
50 + valMostAdv*(distMostAdvancedWhite - distMostAdvancedBlack)
+ valNrPawns*(-nrOfWhites + nrOfBlacks)
+ valThreat*(whiteThreatened - blackThreatened)
+ valProtect*(blackProtectors - whiteProtectors)
+ valForward*(blackForward - whiteForward)

White agent:
50 - (valMostAdv*(distMostAdvancedWhite - distMostAdvancedBlack)
+ valNrPawns*(-nrOfWhites + nrOfBlacks)
+ valThreat*(whiteThreatened - blackThreatened)
+ valProtect*(blackProtectors - whiteProtectors)
+ valForward*(blackForward - whiteForward))

Results:

9x9 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	heuristicAri	heuristicAri	heuristicAri
heuristicHoddi(W)	TIE	0	TIE	heuristicHoddi
heuristicBinni(W)	heuristicAri	heuristicHoddi	0	heuristicTryggvi
heuristicTryggvi(W)	heuristicAri	TIE	heuristicBinni	0

8x8 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	heuristicAri	heuristicAri	heuristicAri
heuristicHoddi(W)	heuristicHoddi	0	heuristicHoddi	heuristicTryggvi
heuristicBinni(W)	heuristicAri	heuristicHoddi	0	heuristicTryggvi
heuristicTryggvi(W)	heuristicAri	heuristicTryggvi	heuristicTryggvi	0

7x7 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	TIE	heuristicBinni	heuristicAri
heuristicHoddi(W)	heuristicAri	0	heuristicBinni	heuristicHoddi
heuristicBinni(W)	heuristicAri	heuristicHoddi	0	heuristicTryggvi
heuristicTryggvi(W)	heuristicAri	heuristicHoddi	TIE	0

6x6 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	heuristicAri	heuristicAri	heuristicTryggvi
heuristicHoddi(W)	heuristicHoddi	0	heuristicHoddi	heuristicTryggvi
heuristicBinni(W)	heuristicAri	heuristicHoddi	0	heuristicTryggvi
heuristicTryggvi(W)	heuristicAri	heuristicHoddi	heuristicTryggvi	0

5x5 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	heuristicHoddi	heuristicAri	heuristicAri
heuristicHoddi(W)	TIE	0	TIE	TIE
heuristicBinni(W)	TIE	heuristicHoddi	0	heuristicBinni
heuristicTryggvi(W)	TIE	TIE	TIE	0

3x5 (playclock 5)	heuristicAri(B)	heuristicHoddi(B)	heuristicBinni(B)	heuristicTryggvi(B)
heuristicAri(W)	0	TIE	TIE	TIE
heuristicHoddi(W)	TIE	0	TIE	TIE
heuristicBinni(W)	TIE	TIE	0	TIE
heuristicTryggvi(W)	TIE	TIE	TIE	0

Conclusions:

Our results show that heuristicAri has the most success of the other three heuristics, considering mainly the large board sizes where a good heuristic will shine much more than on the smaller boards. It is our opinion that obviously there are some variables that work better than others, given the time the agents are allowed to think.