Report
Project tuc-half-checker
Artificial Intelligence
Aggelos Aggelidakis

The procedure followed for the work is as follows:

• Implementation of the minimax search algorithm

• Implement an evaluation function and a method of pruning

• Improve the search applying the algorithm if α-β pruning

Minimax implementation

I used the below pseudo code which I found at the book of the course to implement the minimax algorithm:

```
function MINIMAX-DECISION(state) returns an action
return argmax_[a in ACTIONS(s)] MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v = -infinity
for each a in ACTIONS(state) do
v = MAX(v, MIN-VALUE(RESULT(s, a)))
return v

function MIN-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v = infinity
for each a in ACTIONS(state) do
v = MIN(v, MAX-VALUE(RESULT(s, a)))
return v
```

Value function implementation

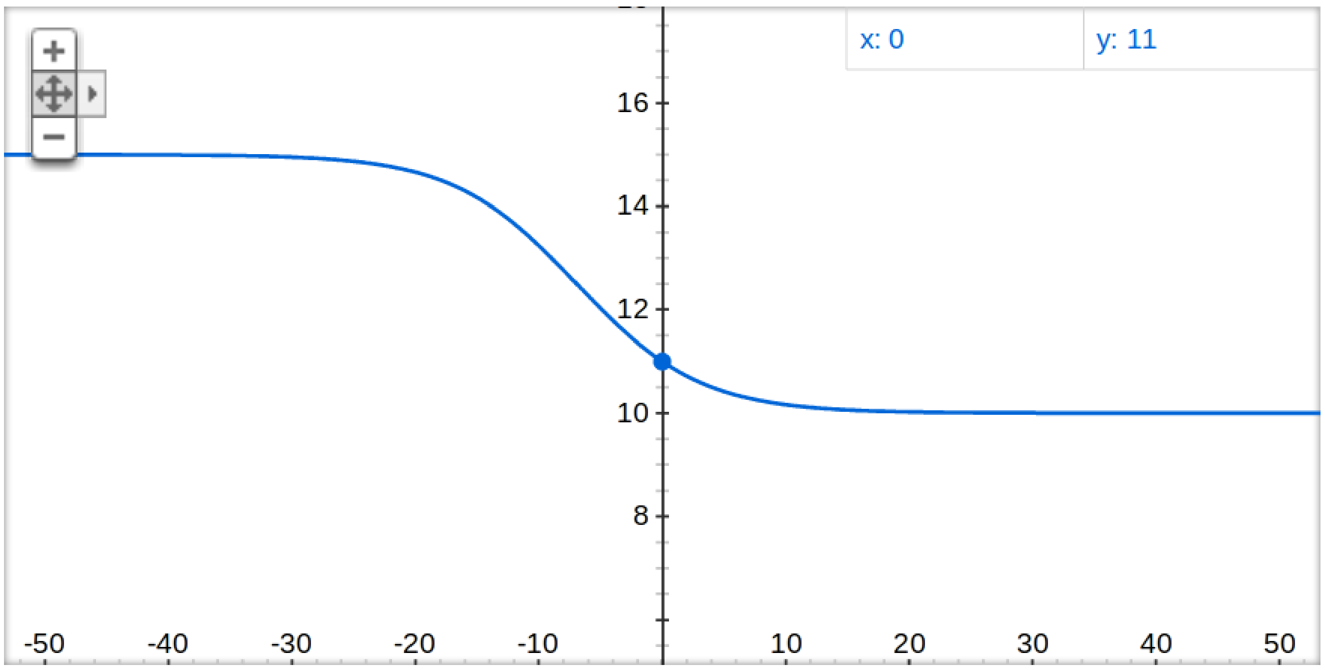
For the evaluation function I gave 1000 as a reward to any pawn that was still on
the chessboard and had eaten some opponents pawn. A reward of 1200 for those located in
the first and last column of the board and 1400 for pawns that were able to reach
at the opposite finish line. If we assume that reward 1000, is a default reward, I gave default reward in
a pawn which was on the board, you will definitely have to give a better reward to the pawns that are
located in the side columns, due to the fact that a pawn in those places could not
eaten by an opponent's pawn, but not so much better that a reward of a pawn which reached the
opposite side. The value value function is:

V(state) = | my (1400 * kings + 1200 * side_pieces + 1000 * simple_pieces) | – | opponent's (1400 *

kings + 1200 * side_pieces + 1000 * simple_pieces) |

α-β Pruning

The method of pruning was based on that function: 10/(2+8*exp(x/5)))+10, x is the value of the
evaluation function :



The function is a result of experiments, in any case I suggest a more improved function, than that.

Minimax α-β pruning

I used the below pseudo code which I found at the book of the course to implement the minimax
algorithm:

```
function ALPHA-BETA-SEARCH(state) returns an action
v = MAX-VALUE(state, -infinity, +infinity)
return the action in ACTIONS(state) with value v

function MAX-VALUE(state, alpha, beta) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v = -infinity
for each a in ACTIONS(state) do
v = MAX(v, MIN-VALUE(RESULT(s, a), alpha, beta))
if v >= beta then return v
alpha = MAX(alpha, v)
return v

function MIN-VALUE(state, alpha, beta) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
v = infinity
for each a in ACTIONS(state) do
v = MIN(v, MAX-VALUE(RESULT(s,a), alpha, beta))
if v <= alpha then return v
beta = MIN(beta, v)
return v
```