UNIVERSITÀ DELLA CALABRIA

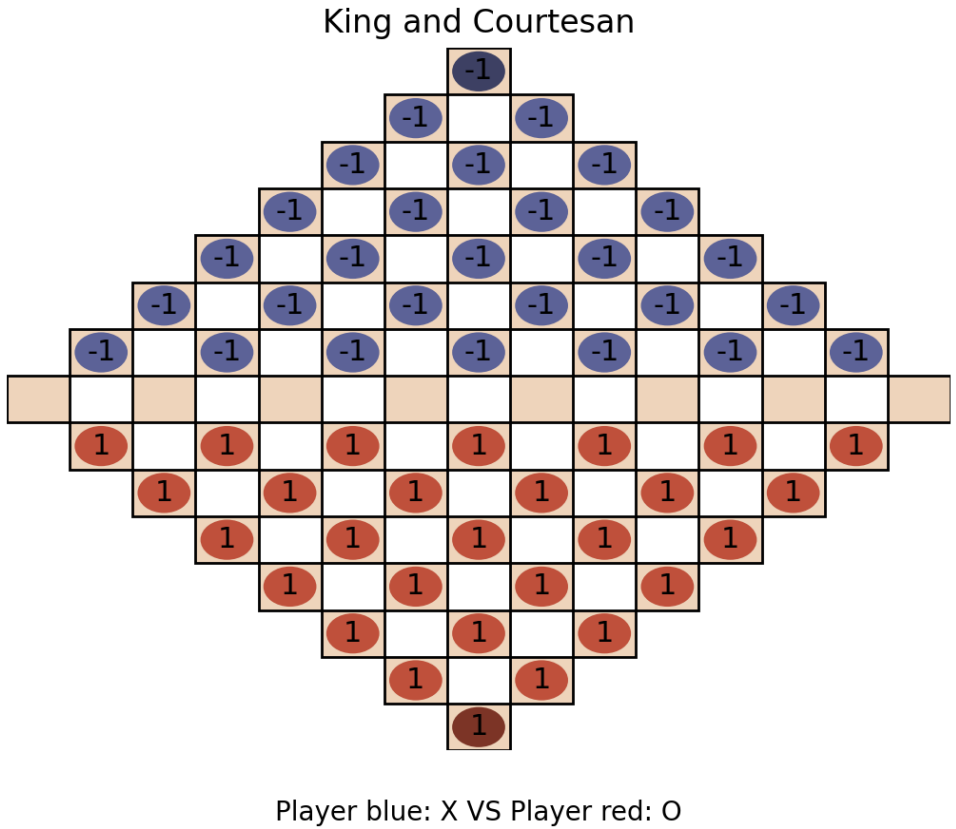
ANTONIO PAONESSA, 252318
GIULIA PERRI, 242645
DESIRÈ CHIAPPETTA, 257192

KING AND COURTESANS COMPETITION

1. Costruzione del modello:

Assegnazione di valori positivi alle nostre pedine e negativi ai nostri avversari.

```
if player=='X':
        myGoal=(14,7)
        opponentGoal=(0,7)
        symbols={'o':-1,'Q':-1,'x':1,'K':1,'#':0,' ':0}
else:
        myGoal=(0,7)
        opponentGoal=(14,7)
        symbols={'o':1,'Q':1,'x':-1,'K':-1,'#':0,' ':0}
```

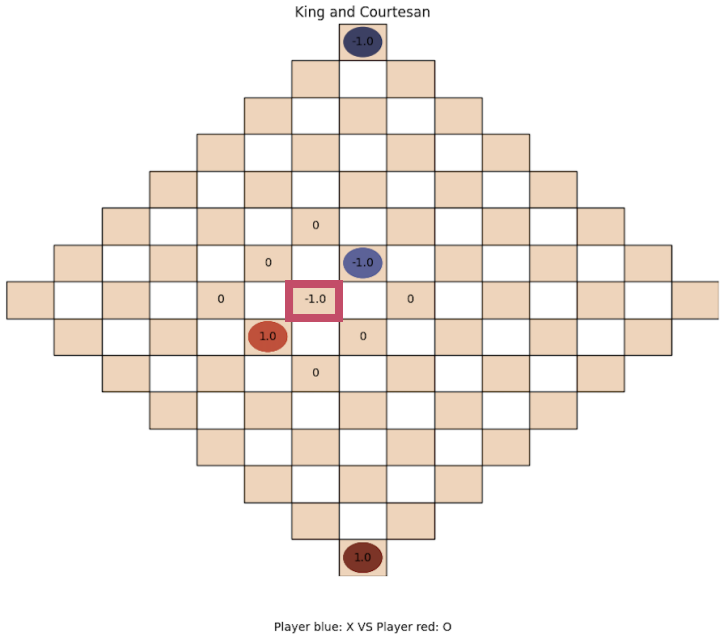


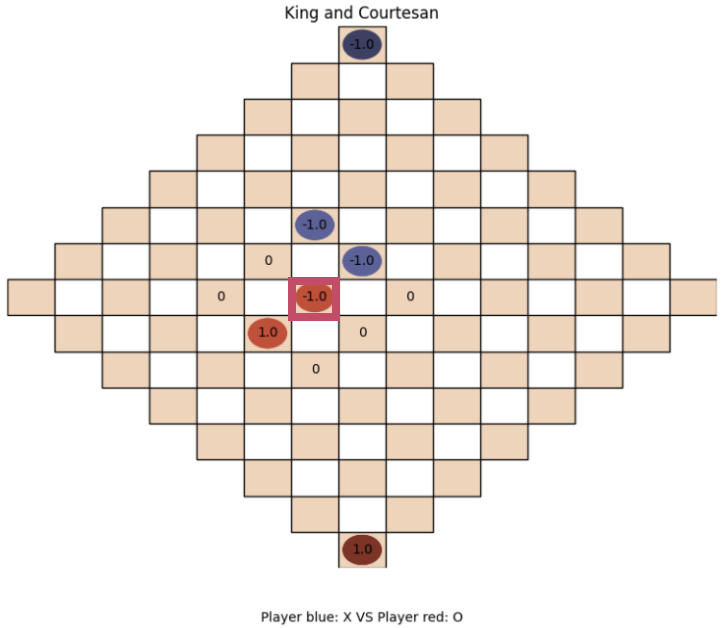
2. Board density:

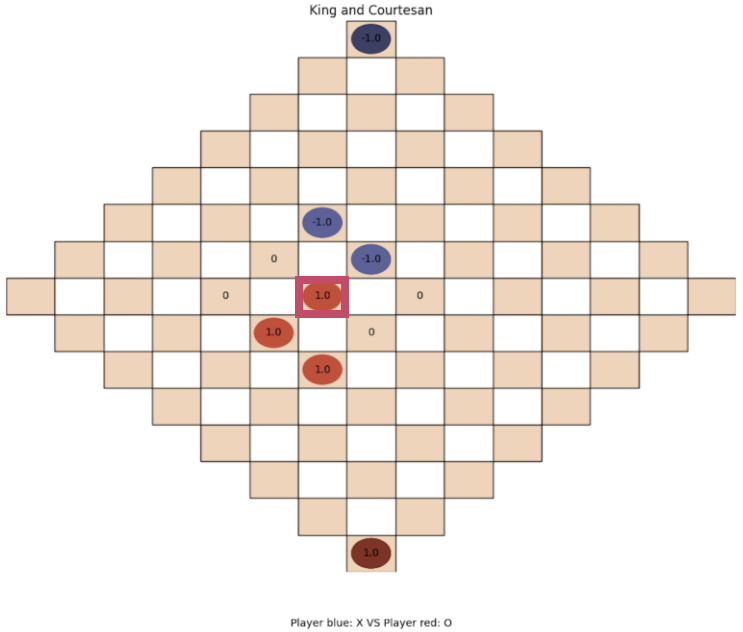
- Globale:

  Esamina il numero di pedine sulla scacchiera e confronta la nostra quantità con quella dell'avversario. L'obiettivo è massimizzare il numero delle nostre pedine e minimizzare quelle dell'avversario.

- Locale:



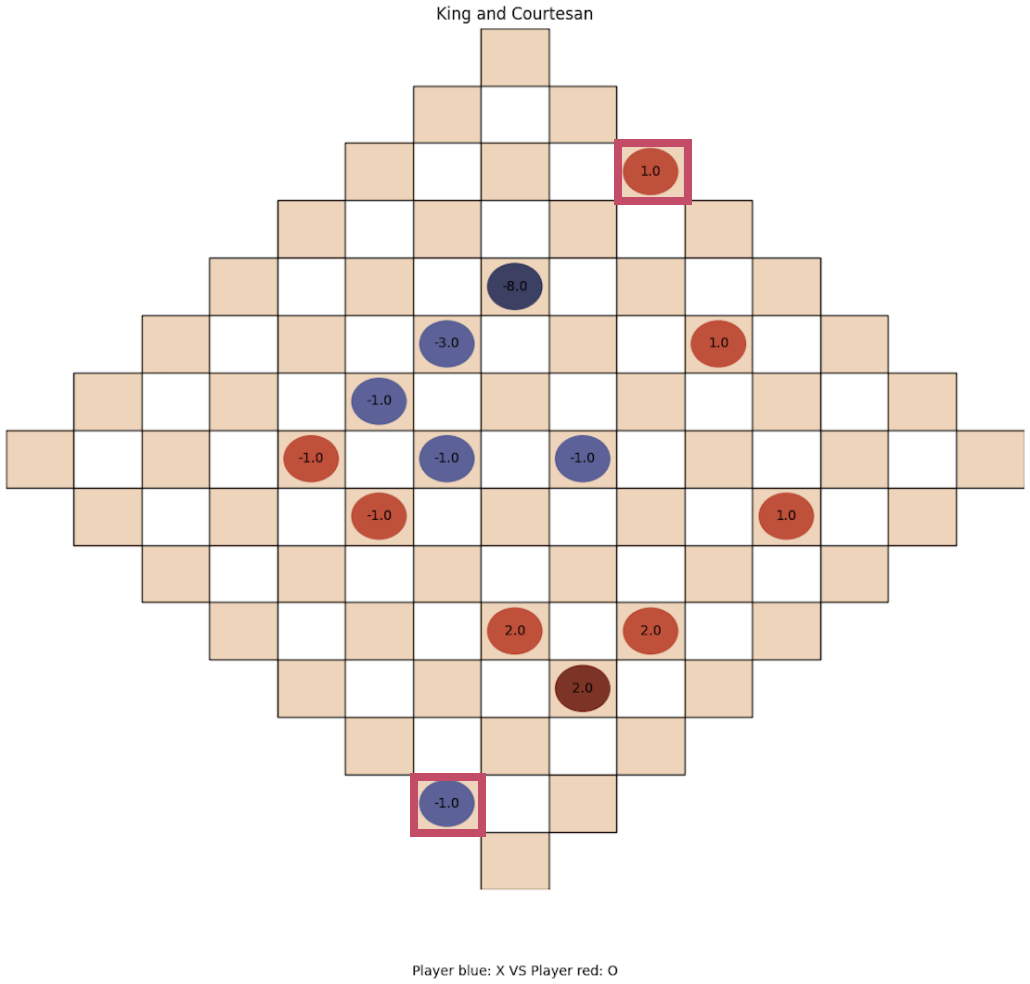




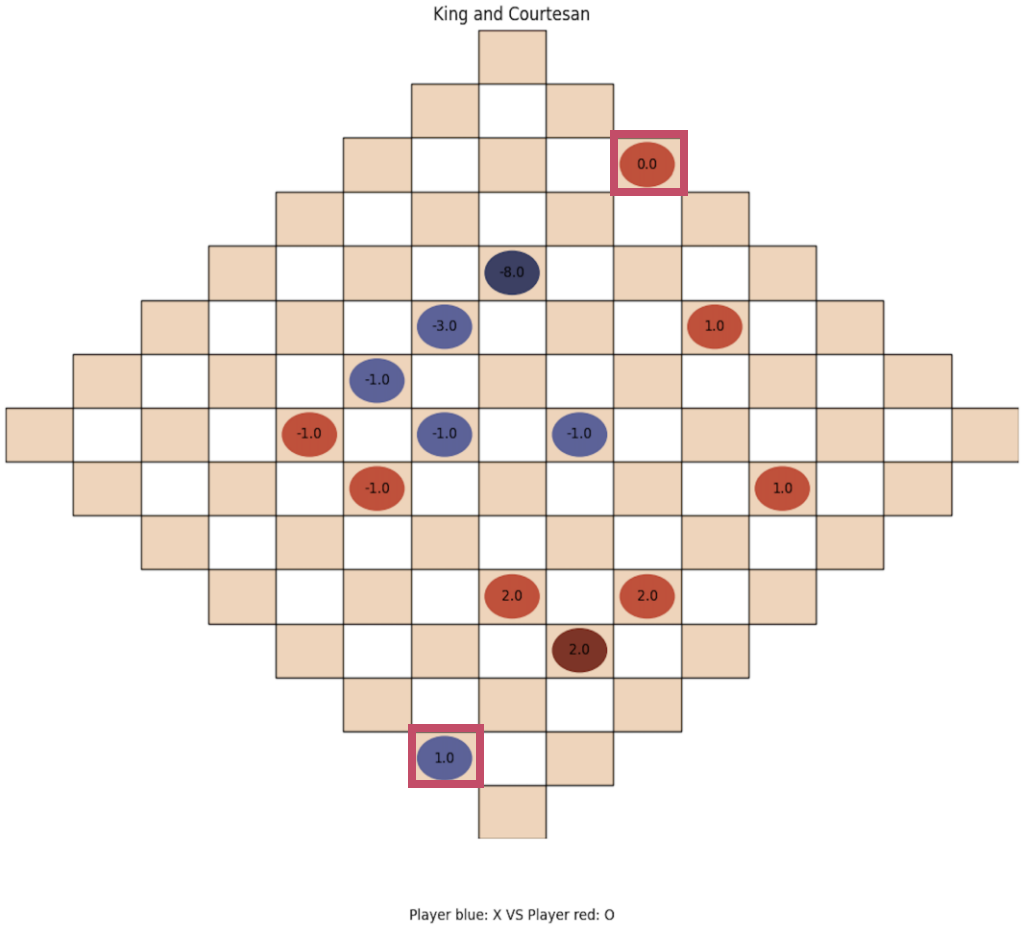
3. Board evaluation:

Guida il player nella difesa e nell'attacco, assegnando un punteggio ad una cella in base al valore delle celle adiacenti ad essa.

Local density:

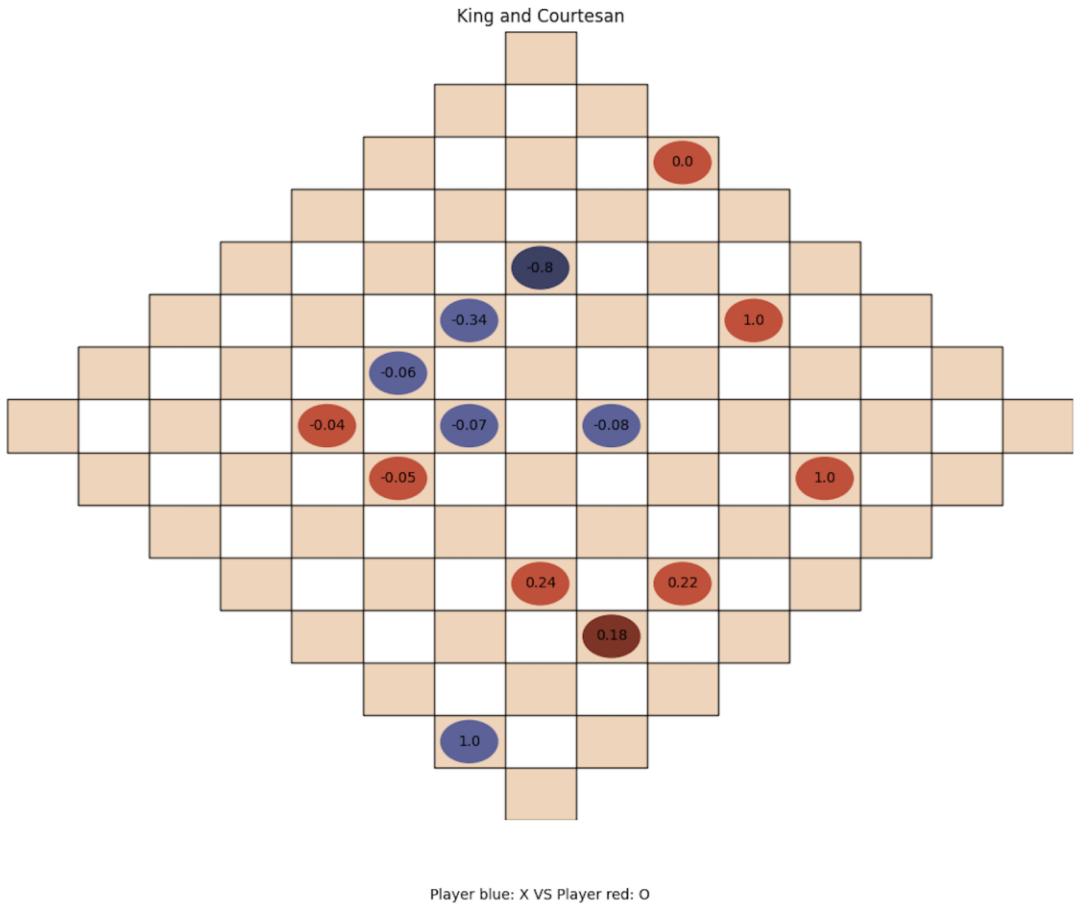


Con possible_move:



4. Distance:

La distanza che abbiamo utilizzato è quella euclidea.
L'uso della distanza nella valutazione della scacchiera ricade nei seguenti casi:

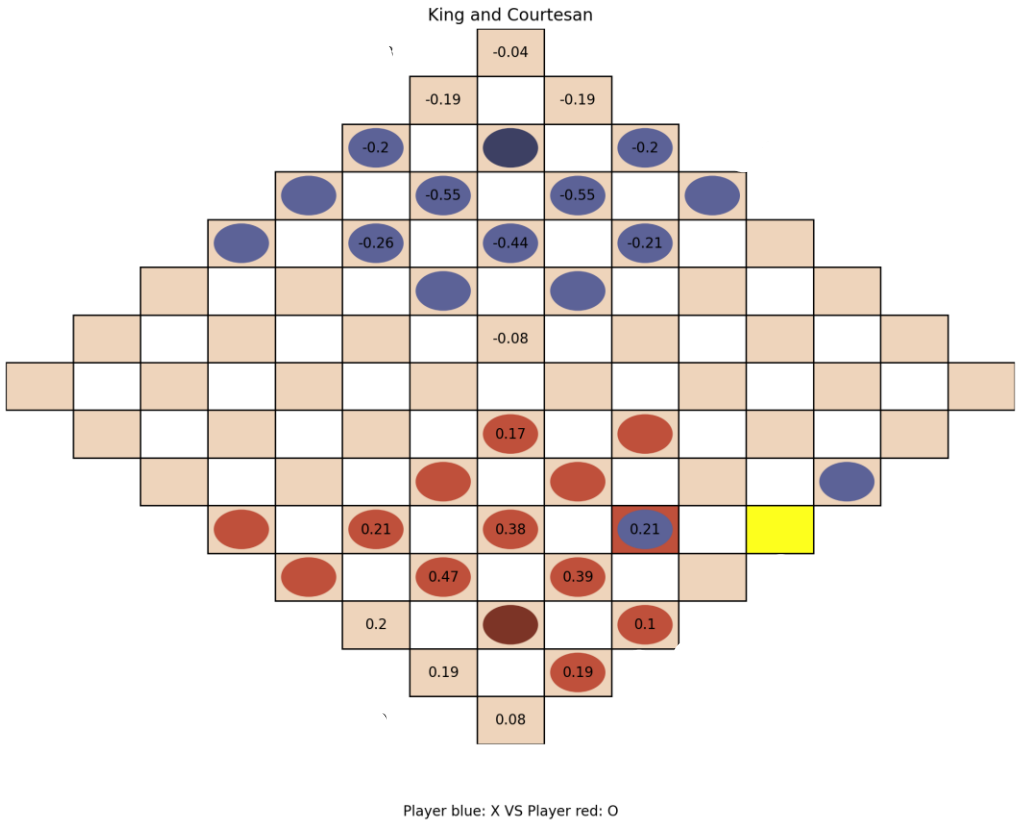



```python
if x == opponentKing:
    # distanceOppKingtoGoal == distance(x,opponentGoal)
    if density==0:
        return -1/distanceOppKingtoGoal if zeros==8 else +5/distanceOppKingtoGoal
    return density/distanceOppKingtoGoal

if x == myKing:
    # distanceMyKingtoGoal == distance(x,myGoal)
    if density==0:
        return +1/distanceMyKingtoGoal if zeros==8 else -8/distanceMyKingtoGoal
    return density/distanceMyKingtoGoal

if density==0 and zeros==8:
    if symbols[board[x]] == symbols[board[myKing]]:
        return symbols[board[x]] if direction(x,opponentKing)>0 else 0
    else:
        return symbols[board[x]] if direction(x,myKing)>0 else symbols[board[myKing]]

distances=distance(x,myKing)*distance(x,opponentKing)
return density/distances if density!=0 else symbols[board[x]]/distances
```

5. Valutazione dello stato:

```
res=f(w1*opponent_king_value+w2*my_king_value +w3*defensive+w4*offensive +w5*globalDensity)
```

w1	w2	w3	w4	w5
50	75	45	35	80



M Y _ K I N G _ V A L U E

```
my_king_value=eval(myKing)
```

O P P O N E N T _ K I N G _ V A L U E

```
opponent_king_value=eval(opponentKing)
```

O F F E N S I V E

```
for square in in_prossimità(opponentKing):
    offensive+=eval(square)
```

D E F E N S I V E

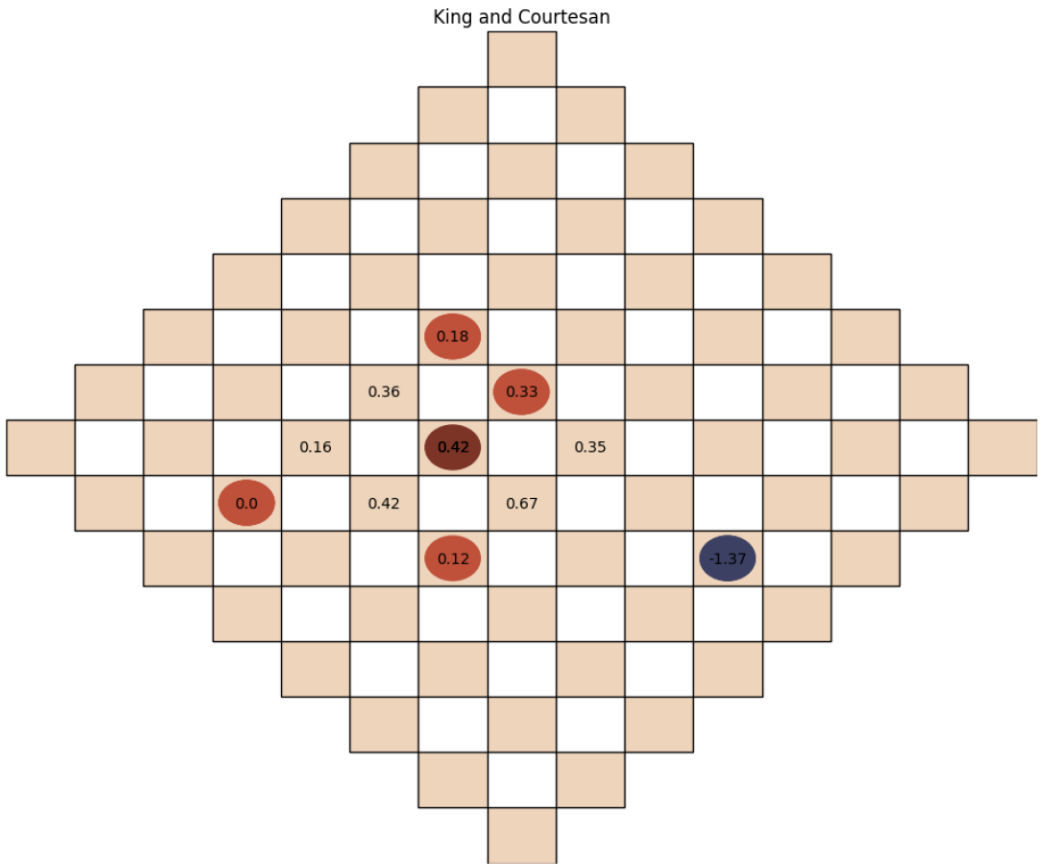
```
for square in in_prossimità(myKing):
    defensive+=eval(square)
```

6. Scelta del cut off

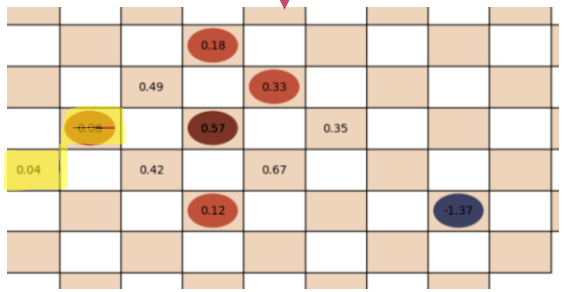
```
def playerStrategy(game,state):
    global turno
    turno+=1
    cutOff=2
    if turno>90: cutOff=3
    if turno>150: cutOff=4
    value,move,depth = h_alphabeta_search(game,state,cutoff_depth(cutOff))
    return move
```

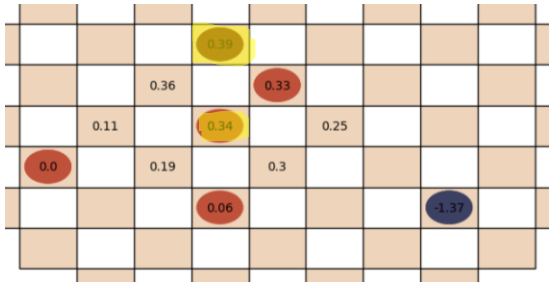
7. Valutazione a posteriori

Durante la competizione, abbiamo perso solo in una configurazione:




```
def compareDistance(myKing,myGoal,opponentKing,opponentGoal):
    if distance(myKing,myGoal) < distance(opponentKing,opponentGoal):
        return +1/distance(myKing,myGoal)
    if distance(myKing,myGoal) > distance(opponentKing,opponentGoal):
        return -1/distance(opponentKing,opponentGoal)
    return 0
```





-0.17149858514250882

h=0.031900694394514 9

-0.17149858514250882

h=0.038325324196052

+0.19611613513818404

h=0.042408720917190 5

GRAZIE PER L'ATTENZIONE

https://github.com/apaonessaa/Competizione-AI.git

Antonio Paonessa, 252318
Giulia Perri, 242645
Desirè Chiappetta, 257192