

main-mathieu

April 12, 2024

0.1 Import libraries

```
[ ]: from stockfish import Stockfish
      from stockfish import StockfishException
      import pandas as pd
      import numpy as np
      import re
      import os
      import pickle
      from concurrent.futures import ThreadPoolExecutor, as_completed
      from timeit import default_timer as timer
      import random
      import string
      import matplotlib.pyplot as plt
      import seaborn as sns
      import chess
      import chess.svg
      from IPython.display import SVG, display, HTML, Image
```

0.2 Import dataset

```
[ ]: with open('valid_positions.txt') as f:
      pos = f.readlines()
      pos1 = []
      for i in range(len(pos)):
          pos1.append(re.sub('\n', '', pos[i]))
      dataset = pos1
```

We are using a reduced version of the original dataset, with some invalid positions excluded

0.3 Additional functions

```
[ ]: STOCKFISH15_LOCATION = '/home/mathieuacher/Downloads/
      ↵stockfish_15_linux_x64_avx2/stockfish_15_x64_avx2'

      #evaluation function
      def evaluation(pos, elo, depth):
          stockfish = Stockfish(STOCKFISH15_LOCATION)
```

```

stockfish.set Elo_rating(elo)
stockfish.set_depth(depth)
stockfish.set_fen_position(pos)
ev = stockfish.get_evaluation()
pos = stockfish.get_top_moves(1)
return ev, pos

def evaluationondouble(pos1, pos2, elo, depth):
    stockfish = Stockfish(STOCKFISH15_LOCATION)
    stockfish.set Elo_rating(elo)
    stockfish.set_depth(depth)
    stockfish.set_fen_position(pos1)
    ev1 = stockfish.get_evaluation()
    stockfish.set_fen_position(pos2)
    ev2 = stockfish.get_evaluation()
    return ev1, ev2

def show_pos(fen_list, size, space=10):
    boards = [chess.Board(fen) for fen in fen_list]
    svg_boards = [chess.svg.board(board=board, size=size) for board in boards]
    html_code = "<div style='display: inline-block; margin-right: {}px; '>{}</div>".format(space, "</div><div style='display: inline-block; margin-right:{}px; '>".format(space)).join(svg_boards)
    display(HTML(html_code))

```

```

[ ]: #delta
def dif1(val1, val2):
    if abs(val1) + abs(val2) == 0:
        return 0
    else:
        return abs(val1 - val2) / (abs(val1) + abs(val2))

#epsilon
def dif2(val1, val2):
    return float(abs(val1 - val2))

```

```

[ ]: def checkcheckW(pos):
    def make_matrix(board): # type(board) == chess.Board()
        pgn = board
        foo = [] # Final board
        pieces = pgn.split(" ", 1)[0]
        rows = pieces.split("/")
        for row in rows:
            foo2 = [] # This is the row I make
            for thing in row:
                if thing.isdigit():
                    for i in range(0, int(thing)):

```

```

        foo2.append('.')
    else:
        foo2.append(thing)
    foo.append(foo2)
    return foo

whites = ['Q', 'K', 'R', 'B', 'N', 'P']
blacks = ['q', 'k', 'r', 'b', 'n', 'p']
a = make_matrix(pos)
check = False
pos = []
for i in range(8):
    for j in range(8):
        if a[i][j] == 'K':
            pos.append(i)
            pos.append(j)
for i in list(reversed(range(pos[1]))):
    if a[pos[0]][i] in whites or a[pos[0]][i] in list(blacks[j] for j in
[1, 3, 4, 5]):
        break
    elif a[pos[0]][i] == 'q' or a[pos[0]][i] == 'r':
        check = True
        break
    for i in list(range(pos[1] + 1, 8)):
        if a[pos[0]][i] in whites or a[pos[0]][i] in list(blacks[j] for j in
[1, 3, 4, 5]):
            break
        elif a[pos[0]][i] == 'q' or a[pos[0]][i] == 'r':
            check = True
            break
    for i in list(reversed(range(pos[0]))):
        if a[i][pos[1]] in whites or a[i][pos[1]] in list(blacks[j] for j in
[1, 3, 4, 5]):
            break
        elif a[i][pos[1]] == 'q' or a[i][pos[1]] == 'r':
            check = True
            break
    for i in list(range(pos[0] + 1, 8)):
        if a[i][pos[1]] in whites or a[i][pos[1]] in list(blacks[j] for j in
[1, 3, 4, 5]):
            break
        elif a[i][pos[1]] == 'q' or a[i][pos[1]] == 'r':
            check = True
            break
r = 8 - np.maximum(abs(pos[0]), abs(pos[1]))
r1 = np.minimum(abs(pos[0]), abs(pos[1]))
r2 = np.minimum(r, r1)

```

```

    for i in list(range(1, r)):
        if a[pos[0] + i][pos[1] + i] in whites or a[pos[0] + i][pos[1] + i] in u
        ↵list(blacks[j] for j in [1, 2, 4, 5]):
            break
        elif a[pos[0] + i][pos[1] + i] == 'b' or a[pos[0] + i][pos[1] + i] == u
        ↵'q':
            check = True
            break
        for i in list(range(1, r2)):
            if a[pos[0] - i][pos[1] + i] in whites or a[pos[0] - i][pos[1] + i] in u
            ↵list(blacks[j] for j in [1, 2, 4, 5]):
                break
            elif a[pos[0] - i][pos[1] + i] == 'b' or a[pos[0] - i][pos[1] + i] == u
            ↵'q':
                check = True
                break
            for i in list(range(1, r2)):
                if a[pos[0] - i][pos[1] - i] in whites or a[pos[0] - i][pos[1] - i] in u
                ↵list(blacks[j] for j in [1, 2, 4, 5]):
                    break
                elif a[pos[0] - i][pos[1] - i] == 'b' or a[pos[0] - i][pos[1] - i] == u
                ↵'q':
                    check = True
                    break
                for i in list(range(1, r)):
                    if a[pos[0] + i][pos[1] - i] in whites or a[pos[0] + i][pos[1] - i] in u
                    ↵list(blacks[j] for j in [1, 2, 4, 5]):
                        break
                    elif a[pos[0] + i][pos[1] - i] == 'b' or a[pos[0] + i][pos[1] - i] == u
                    ↵'q':
                        check = True
                        break
                    if pos[0] + 2 < 8 and pos[1] - 1 > 0 and a[pos[0] + 2][pos[1] - 1] == 'n':
                        check = True
                    if pos[0] + 2 < 8 and pos[1] + 1 < 8 and a[pos[0] + 2][pos[1] + 1] == 'n':
                        check = True
                    if pos[0] - 2 > 0 and pos[1] - 1 > 0 and a[pos[0] - 2][pos[1] - 1] == 'n':
                        check = True
                    if pos[0] - 2 > 0 and pos[1] + 1 < 8 and a[pos[0] - 2][pos[1] + 1] == 'n':
                        check = True
                    if pos[0] + 1 < 8 and pos[1] - 2 > 0 and a[pos[0] + 1][pos[1] - 2] == 'n':
                        check = True
                    if pos[0] + 1 < 8 and pos[1] + 2 < 8 and a[pos[0] + 1][pos[1] + 2] == 'n':
                        check = True
                    if pos[0] - 1 > 0 and pos[1] - 2 > 0 and a[pos[0] - 1][pos[1] - 2] == 'n':
                        check = True

```

```

if pos[0] - 1 > 0 and pos[1] + 2 < 8 and a[pos[0] - 1][pos[1] + 2] == 'n':
    check = True
if pos[0] - 1 > 0 and pos[1] - 1 > 0 and a[pos[0] - 1][pos[1] - 1] == 'p':
    check = True
if pos[0] - 1 > 0 and pos[1] + 1 < 8 and a[pos[0] - 1][pos[1] + 1] == 'p':
    check = True
return check

def checkcheckB(pos):
def make_matrix(board): # type(board) == chess.Board()
    pgn = board
    foo = [] # Final board
    pieces = pgn.split(" ", 1)[0]
    rows = pieces.split("/")
    for row in rows:
        foo2 = [] # This is the row I make
        for thing in row:
            if thing.isdigit():
                for i in range(0, int(thing)):
                    foo2.append('.')
            else:
                foo2.append(thing)
        foo.append(foo2)
    return foo

whites = ['Q', 'K', 'R', 'B', 'N', 'P']
blacks = ['q', 'k', 'r', 'b', 'n', 'p']
a = make_matrix(pos)
check = False
pos = []
for i in range(8):
    for j in range(8):
        if a[i][j] == 'k':
            pos.append(i)
            pos.append(j)
for i in list(reversed(range(pos[1]))):
    if a[pos[0]][i] in blacks or a[pos[0]][i] in list(whites[j] for j in ↵[1, 3, 4, 5]):
        break
    elif a[pos[0]][i] == 'Q' or a[pos[0]][i] == 'R':
        check = True
        break
for i in list(range(pos[1] + 1, 8)):
    if a[pos[0]][i] in blacks or a[pos[0]][i] in list(whites[j] for j in ↵[1, 3, 4, 5]):
        break

```

```

    elif a[pos[0]][i] == 'Q' or a[pos[0]][i] == 'R':
        check = True
        break
    for i in list(reversed(range(pos[0]))):
        if a[i][pos[1]] in blacks or a[i][pos[1]] in list(whites[j] for j in
        ↪[1, 3, 4, 5]):
            break
        elif a[i][pos[1]] == 'Q' or a[i][pos[1]] == 'R':
            check = True
            break
    for i in list(range(pos[0] + 1, 8)):
        if a[i][pos[1]] in blacks or a[i][pos[1]] in list(whites[j] for j in
        ↪[1, 3, 4, 5]):
            break
        elif a[i][pos[1]] == 'Q' or a[i][pos[1]] == 'R':
            check = True
            break
    r = 8 - np.maximum(abs(pos[0]), abs(pos[1]))
    r1 = np.minimum(abs(pos[0]), abs(pos[1]))
    r2 = np.minimum(r, r1)
    for i in list(range(1, r)):
        if a[pos[0] + i][pos[1] + i] in blacks or a[pos[0] + i][pos[1] + i] in
        ↪list(whites[j] for j in [1, 2, 4, 5]):
            break
        elif a[pos[0] + i][pos[1] + i] == 'B' or a[pos[0] + i][pos[1] + i] ==
        ↪'Q':
            check = True
            break
    for i in list(range(1, r2)):
        if a[pos[0] - i][pos[1] + i] in blacks or a[pos[0] - i][pos[1] + i] in
        ↪list(whites[j] for j in [1, 2, 4, 5]):
            break
        elif a[pos[0] - i][pos[1] + i] == 'B' or a[pos[0] - i][pos[1] + i] ==
        ↪'Q':
            check = True
            break
    for i in list(range(1, r2)):
        if a[pos[0] - i][pos[1] - i] in blacks or a[pos[0] - i][pos[1] - i] in
        ↪list(whites[j] for j in [1, 2, 4, 5]):
            break
        elif a[pos[0] - i][pos[1] - i] == 'B' or a[pos[0] - i][pos[1] - i] ==
        ↪'Q':
            check = True
            break
    for i in list(range(1, r)):

```

```

        if a[pos[0] + i][pos[1] - i] in blacks or a[pos[0] + i][pos[1] - i] in u
    ↵list(whites[j] for j in [1, 2, 4, 5]):
        break
    elif a[pos[0] + i][pos[1] - i] == 'B' or a[pos[0] + i][pos[1] - i] == u
    ↵'Q':
        check = True
        break
    if pos[0] + 2 < 8 and pos[1] - 1 > 0 and a[pos[0] + 2][pos[1] - 1] == 'N':
        check = True
    if pos[0] + 2 < 8 and pos[1] + 1 < 8 and a[pos[0] + 2][pos[1] + 1] == 'N':
        check = True
    if pos[0] - 2 > 0 and pos[1] - 1 > 0 and a[pos[0] - 2][pos[1] - 1] == 'N':
        check = True
    if pos[0] - 2 > 0 and pos[1] + 1 < 8 and a[pos[0] - 2][pos[1] + 1] == 'N':
        check = True
    if pos[0] + 1 < 8 and pos[1] - 2 > 0 and a[pos[0] + 1][pos[1] - 2] == 'N':
        check = True
    if pos[0] + 1 < 8 and pos[1] + 2 < 8 and a[pos[0] + 1][pos[1] + 2] == 'N':
        check = True
    if pos[0] - 1 > 0 and pos[1] - 2 > 0 and a[pos[0] - 1][pos[1] - 2] == 'N':
        check = True
    if pos[0] - 1 > 0 and pos[1] + 2 < 8 and a[pos[0] - 1][pos[1] + 2] == 'N':
        check = True
    if pos[0] - 1 > 0 and pos[1] - 1 > 0 and a[pos[0] - 1][pos[1] - 1] == 'P':
        check = True
    if pos[0] - 1 > 0 and pos[1] + 1 < 8 and a[pos[0] - 1][pos[1] + 1] == 'P':
        check = True
    return check

def bothcheck(pos):
    both = False
    if checkcheckB(pos) == True and checkcheckW(pos) == True:
        both = True
    return both

```

0.4 Functions to alterate positions

```
[ ]: def sim_axis(pos):
    pos_transformed = []
    col = pos[-1:]
    aux1 = pos[:-2]
    aux2 = aux1.split("/")
    aux3 = []
    for j in range(8):
        aux3.append(aux2[j][::-1])
    aux4 = '/'.join(aux3)
```

```

pos_transformed.append(aux4)
if col == 'w':
    pos_transformed = pos_transformed[0] + ' w'
if col == 'b':
    pos_transformed = pos_transformed[0] + ' b'
return pos_transformed

def sim_diag(pos):
    def Convert(string):
        list1 = []
        list1[:0] = string
        return list1

    pos_transformed = []
    colour = pos[-1:]
    aux = pos.replace('8', '11111111').replace('7', '1111111').replace('6',
    ↵'111111').replace('5', '11111').replace('4',
    ↵'1111').replace(
    ↵'3', '111').replace('2', '11')
    aux2 = aux.split('/')
    for i in range(8):
        aux2[i] = Convert(aux2[i])
        aux3 = [] * 8

    for j in reversed(range(8)):
        aux4 = []
        for i in reversed(range(8)):
            aux4.append(aux2[i][j])
        aux3.append(aux4)
    for i in reversed(range(8)):
        aux3[i] = "".join(aux3[i])
    pos_transformed = "/".join(aux3)
    if colour == 'b':
        pos_transformed = pos_transformed + ' b'
    else:
        pos_transformed = pos_transformed + ' w'
        pos_transformed = pos_transformed.replace('11111111', '8').
    ↵replace('1111111', '7').replace('111111', '6').replace(
        '11111', '5').replace('1111', '4').replace('111', '3').replace('11',
    ↵'2')

    return pos_transformed

def sim_mirror(pos):

```

```

pos_transformed = []
colour = pos[-1:]
aux1 = pos[:-2]
aux2 = aux1.swapcase()
aux3 = aux2.split("/")
aux4 = []
for j in reversed(aux3):
    aux4.append(j)
    aux5 = '/'.join(aux4)
pos_transformed.append(aux5)
if colour == 'b':
    pos_transformed = pos_transformed[0] + ' w'
else:
    pos_transformed = pos_transformed[0] + ' b'
return pos_transformed

def replace(pos):
    if pos.count('B') > 0 & pos.count('R') > 0:
        n = random.random()
        if n > 0.5:
            aux = pos.replace('R', 'Q', 1)
        else:
            aux = pos.replace('B', 'Q', 1)
    elif pos.count('B') > 0 & pos.count('R') == 0:
        aux = pos.replace('B', 'Q', 1)
    elif pos.count('R') > 0 & pos.count('B') == 0:
        aux = pos.replace('R', 'Q', 1)
    else:
        aux = pos
    return aux

```

0.5 MRs

```

[ ]: #for sim_axis and sim_diag
def MR_equi(o1, o2, delta, epsilon):
    final = False
    if o1.get('type') != o2.get('type'):
        final = False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') != o2.get('value'):
        final = False
    elif o1.get('type') == o2.get('type') == 'cp' and dif1(o1.get('value'), o2.get('value')) > delta and dif2(
            o1.get('value'), o2.get('value')) >= epsilon:
        final = False
    else:

```

```

        final = True
    return final

#for sim_mirror
def MR_mirror(o1, o2, delta, epsilon):
    final = False
    if o1.get('type') != o2.get('type'):
        final = False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') != (-1 * o2.get('value')):
        #elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') * (-1 * o2.get('value')) < 0:
            final = False
    elif o1.get('type') == o2.get('type') == 'cp' and dif1(o1.get('value'), (-1 * o2.get('value'))) > delta and dif2(o1.get('value'), (-1 * o2.get('value'))) >= epsilon:
        final = False
    else:
        final = True
    return final

#for replace
def MR_better_original(o1,o2,delta,epsilon):
    final = bool()
    if o1.get('type') != o2.get('type') and o2.get('type') != 'mate':
        final = False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') < o2.get('value') and o1.get('value') > 0:
        final = False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') > o2.get('value') and o1.get('value') < 0:
        final = False
    elif o1.get('type') == o2.get('type') == 'cp' and o1.get('value') > o2.get('value') and dif1(o1.get('value'),o2.get('value')) > delta and dif2(o1.get('value'),o2.get('value')) >= epsilon:
        final = False
    else:
        final = True
    return final

#when using the evaluation after the best move
def MR_first(o1, o2, delta, epsilon):
    final = False
    if o1[0].get('Mate') != o2[0].get('Mate'):
        final = False

```

```

    elif o1[0].get('Mate') == o2[0].get('Mate') == None and dif1(o1[0].
        ↪get('Centipawn'),
                                         o2[0].
        ↪get('Centipawn')) > delta and dif2(
            o1[0].get('Centipawn'), o2[0].get('Centipawn')) >= epsilon:
                final = False
            else:
                final = True
        return final

```

0.5.1 Problem with the original MR_better : example

```
[ ]: pos = '1B1n1r2/8/8/1n5b/3r4/8/5k2/2K5'
pos2 = replace(pos)
ev1 = evaluation(pos,5000,10)[0]
ev2 = evaluation (pos2, 5000, 10)[0]

print(ev1, '\n', ev2)

{'type': 'mate', 'value': -10}
{'type': 'cp', 'value': -824}
```

Before replace -> mate in 10 for Black After replace -> advantage of 824 centipawns for Black

The evaluation got better for White, which should meet the MR requirements but:

```
[ ]: print(MR_better_original(ev1,ev2,1,100))
```

False

It seems that some conditions are incorrect in the original MR_better function, so we modified it as follows:

```
[ ]: def MR_better(o1, o2, delta, epsilon):
    final = False
    # knowing that we're upgrading white pieces (always)
    # if we switch from diff in centipawns to mate for Black --> False
    if o1.get('type') == 'cp' and o2.get('type') == 'mate' and o2.get('value') <
        0:
        final = False
    # if we switch from mate to diff in centipawns for White --> False
    elif o1.get('type') == 'mate' and o2.get('type') == 'cp' and o1.
        ↪get('value') > 0:
        final = False
    # ex : from #7 to #9 --> False
    elif o1.get('type') == o2.get('type') == 'mate' and o2.get('value') > o1.
        ↪get('value') > 0:
        final = False
    # ex : from #-9 to #-7 --> False
```

```

    elif o1.get('type') == o2.get('type') == 'mate' and 0 > o2.get('value') >_u
    ↵o1.get('value'):
        final = False
    # ex : from #5 to #-5 --> False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value') > 0 >_u
    ↵o2.get('value'):
        final = False
    # if centipawns diff for both evaluations, will depend on thresholds
    elif o1.get('type') == o2.get('type') == 'cp' and o1.get('value') > o2.
    ↵get('value') and dif1(o1.get('value')),

        ↵           o2.get('value')) > delta and dif2(
            o1.get('value'), o2.get('value')) >= epsilon:
                final = False
        else:
            final = True
    return final

```

0.6 Main functions

0.6.1 Function for mass evaluations

```
[ ]: def main(dataset, elo, depth, type):
    global threadsfinis
    global threadstotaux
    pos1 = []
    pos2 = []
    results1 = []
    results2 = []
    start = timer()
    index = []
    if type == 0:
        for i in range(len(dataset)):
            if bothcheck(dataset[i]) == False and_
    ↵bothcheck(sim_mirror(dataset[i])) == False:
                if i%5==0:
                    print(i)
                try:
                    ev1 = evaluation(dataset[i], elo, depth)[0]
                    ev2 = evaluation(sim_mirror(dataset[i]), elo, depth)[0]
                    pos1.append(dataset[i]) ## adding original position to_
    ↵dataframe
                    pos2.append(sim_mirror(dataset[i])) ## adding transformed_
    ↵position to dataframe
                    results1.append(ev1) ## adding evaluation of original_
    ↵position to dataframe

```

```

        results2.append(ev2) ## adding evaluation of transformed
        ↵position to dataframe
            index.append(i) ## adding index to dataframe
        except StockfishException:
            pass
    elif type == 1:
        for i in range(len(dataset)):
            if bothcheck(dataset[i]) == False and
        ↵bothcheck(sim_axis(dataset[i])) == False:
                if i%5==0:
                    print(i)
            try:
                ev1 = evaluation(dataset[i], elo, depth)[0]
                ev2 = evaluation(sim_axis(dataset[i]), elo, depth)[0]
                pos1.append(dataset[i]) ## adding original position to
        ↵dataframe
                    pos2.append(sim_axis(dataset[i])) ## adding transformed
        ↵position to dataframe
                        results1.append(ev1) ## adding evaluation of original
        ↵position to dataframe
                        results2.append(ev2) ## adding evaluation of transformed
        ↵position to dataframe
                            index.append(i) ## adding index to dataframe
            except StockfishException:
                pass
    elif type == 2:
        for i in range(len(dataset)):
            #ignoring positions with pawns
            if bothcheck(dataset[i]) == False and
        ↵bothcheck(sim_diag(dataset[i])) == False and 'p' not in dataset[
            i] and 'P' not in dataset[i]:
                if i%5==0:
                    print(i)
            try:
                ev1 = evaluation(dataset[i], elo, depth)[0]
                ev2 = evaluation(sim_diag(dataset[i]), elo, depth)[0]
                pos1.append(dataset[i]) ## adding original position to
        ↵dataframe
                    pos2.append(sim_diag(dataset[i])) ## adding transformed
        ↵position to dataframe
                        results1.append(ev1) ## adding evaluation of original
        ↵position to dataframe
                        results2.append(ev2) ## adding evaluation of transformed
        ↵position to dataframe
                            index.append(i) ## adding index to dataframe
            except StockfishException:

```

```

        pass

elif type == 3:
    for i in range(len(dataset)):
        #ignoring positions without white rooks or bishops
        if bothcheck(dataset[i]) == False and
bothcheck(replace(dataset[i])) == False and (
            'B' in dataset[i] or 'R' in dataset[i]):
            if i%5==0:
                print(i)
            try:
                ev1 = evaluation(dataset[i], elo, depth)[0]
                ev2 = evaluation(replace(dataset[i]), elo, depth)[0]
                pos1.append(dataset[i]) ## adding original position to
dataframe
                pos2.append(replace(dataset[i])) ## adding transformed
position to dataframe
                results1.append(ev1) ## adding evaluation of original
position to dataframe
                results2.append(ev2) ## adding evaluation of transformed
position to dataframe
                index.append(i) ## adding index to dataframe
            except StockfishException:
                pass

elif type == 4:
    for i in range(len(dataset)):
        if bothcheck(dataset[i]) == False and
bothcheck(sim_axis(dataset[i])) == False:
            if i%5==0:
                print(i)
            try:
                ev1 = evaluation(dataset[i], elo, depth)[1]
                ev2 = evaluation(sim_axis(dataset[i]), elo, depth)[1]
                # ajout d'un len à la fin parce que souci si on essaie de
faire jouer un coup sur une position où ça n'est pas possible (mat, pat)
                if len(ev1) != 0 and len(ev2) != 0:
                    # MR.
append(MR_first(evaluation(dataset[i],elo,depth)[1],evaluation(sim_axis(dataset[i]),
elo, depth)[1], delta, epsilon)) ## adding MR to dataframe
                pos1.append(dataset[i]) ## adding original position to
dataframe
                pos2.append(sim_axis(dataset[i])) ## adding
transformed position to dataframe
                results1.append(ev1) ## adding evaluation of original
position to dataframe

```

```

        results2.append(ev2) ## adding evaluation of transformed position to dataframe
    except StockfishException:
        pass

d = {'index': index, 'pos1': pos1, 'evaluation pos1': results1, 'pos2': pos2, 'evaluation pos2': results2}
df = pd.DataFrame(data=d) ## creating dataframe

# print(timer()-start)
return df

```

0.6.2 Function for mass MR checks

```

[ ]: def MR(df, type, delta, epsilon):
    ev1s = df["evaluation pos1"]
    ev2s = df["evaluation pos2"]
    MRs = []
    if type == 0:
        for k in range(len(ev1s)):
            MRs.append(MR_mirror(ev1s[k], ev2s[k], delta, epsilon))
    if type == 1 or type == 2:
        for k in range(len(ev1s)):
            MRs.append(MR_equi(ev1s[k], ev2s[k], delta, epsilon))
    if type == 3:
        for k in range(len(ev1s)):
            MRs.append(MR_better(ev1s[k], ev2s[k], delta, epsilon))
    if type == 6:
        for k in range(len(ev1s)):
            MRs.append(MR_better_original(ev1s[k], ev2s[k], delta, epsilon))
    if type == 4:
        for k in range(len(ev1s)):
            MRs.append(MR_first(ev1s[k], ev2s[k], delta, epsilon))
    falses = trues = 0
    for k in range(len(MRs)):
        if MRs[k]:
            trues += 1
        else:
            falses += 1
    #print("MR valides : ", trues, "; MR invalides : ", falses, "; Ratio : ", trues / (trues + falses) * 100, "%")
    return trues / (trues + falses) * 100

```

0.6.3 Function for grids

```
[ ]: deltas = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5]
      epsilons = [0, 5, 10, 15, 20, 25, 30, 40, 50, 60, 75, 100]

def tests(evas, typent, d, add='', size1 = 100):
    if typent==5:
        type = 1
    else:
        type = typent
    results_grid = []

    for delta in deltas:
        delta_results = []

        for epsilon in epsilons:
            result = 100 - MR(evas, type, delta, epsilon)

            delta_results.append(result)

        results_grid.append(delta_results)

    df_results = pd.DataFrame(results_grid, index=deltas, columns=epsilons)

    plt.figure(figsize=(10*size1/100, 6*size1/100))
    sns.heatmap(df_results, cmap='RdYlGn_r', annot=True, fmt=".2f", cbar_kws={'label': 'Valeurs'}, annot_kws={"size": 10*size1/100})
    typent = ["sim_mirror", "sim_axis", "sim_diag", "replace", "best_move", "SF15 vs SF16", "replace_mr_og"][typent]
    label = 'MR violation rate, ' + typent + ', depth=' + str(d) + add

    plt.title(label)
    plt.xlabel('Epsilons')
    plt.ylabel('Deltas')
    plt.plot()
```

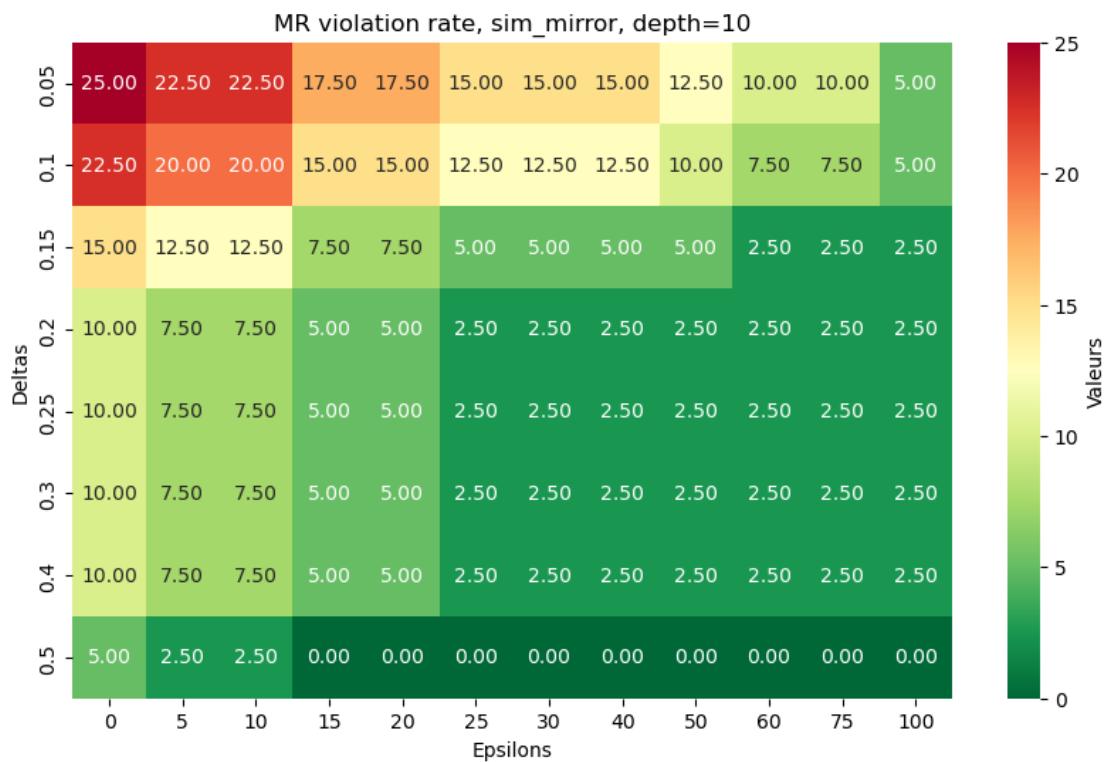
0.7 Application of main functions

An example with the first 40 positions of our dataset.

```
[ ]: evas = main(dataset[0:40], 4000, 10,0)
      #last parameter --> type;
      #0 : mirror,
      #1 : sim_axis,
      #2 : sim_diag,
      #3 : better,
      #4 : first
```

```
tests(evas,0,10)
```

```
0  
5  
10  
15  
20  
25  
30  
35
```



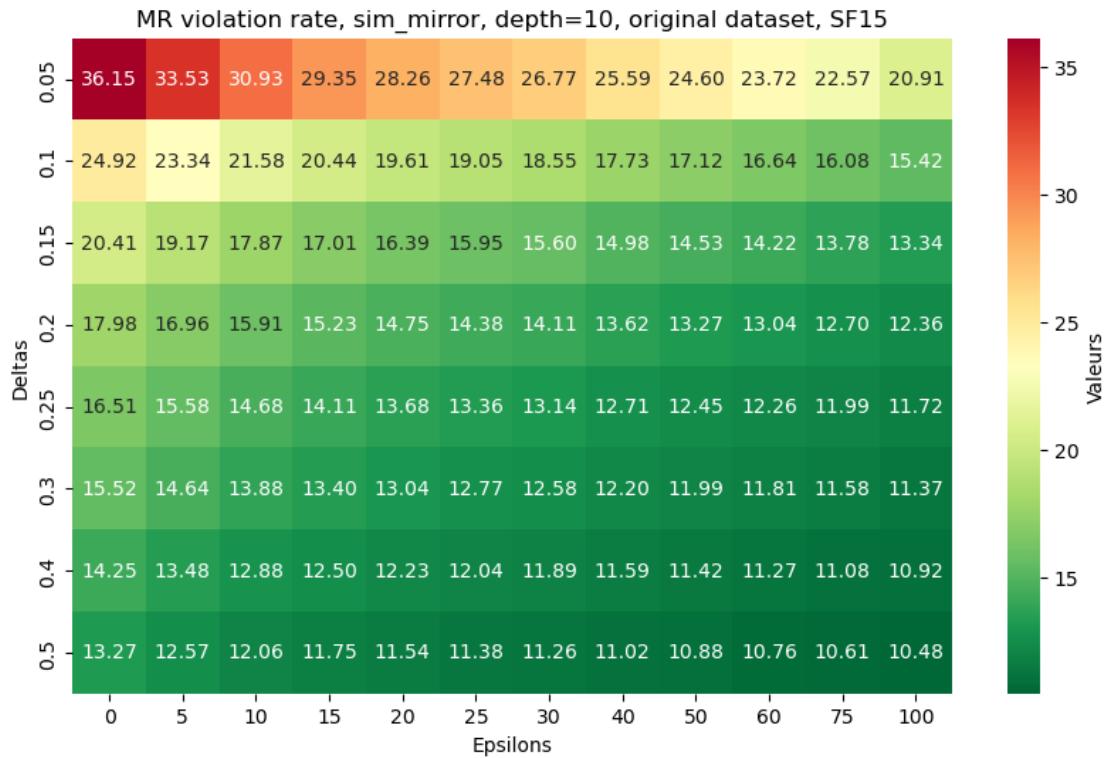
0.8 Observations

0.8.1 sim_mirror

Evaluations are stored in pickle files. ##### depth = 10, entire dataset (Link V16)

Showing at first the grid we got ourselves.

```
[ ]: path = os.path.join('sim_mirror_d=10','evaluations50000_sim_mirror_d_10.pkl')  
with open(path, 'rb') as file:  
    evaluations50000_sim_mirror_d_10 = pickle.load(file)  
  
tests(evaluations50000_sim_mirror_d_10,0,10, ', original dataset, SF15')
```



Here is the grid from the article.

```
[ ]: from IPython.display import display, Image
display(Image(filename='originalmirror10_am.png',width=900,height=900))
```

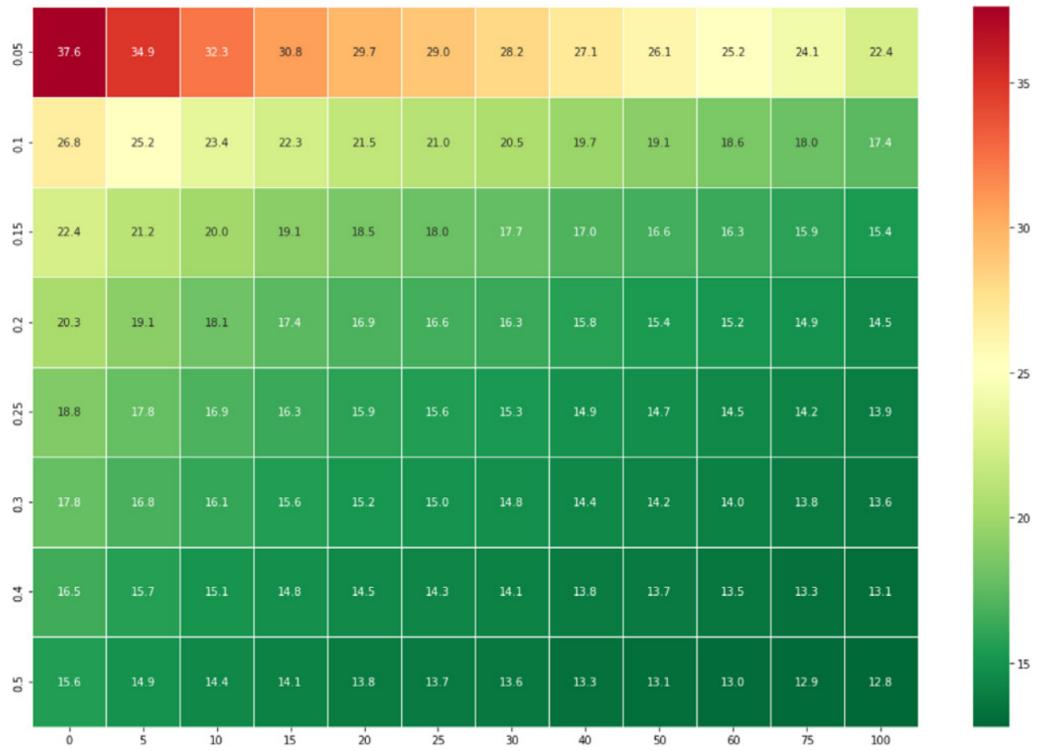


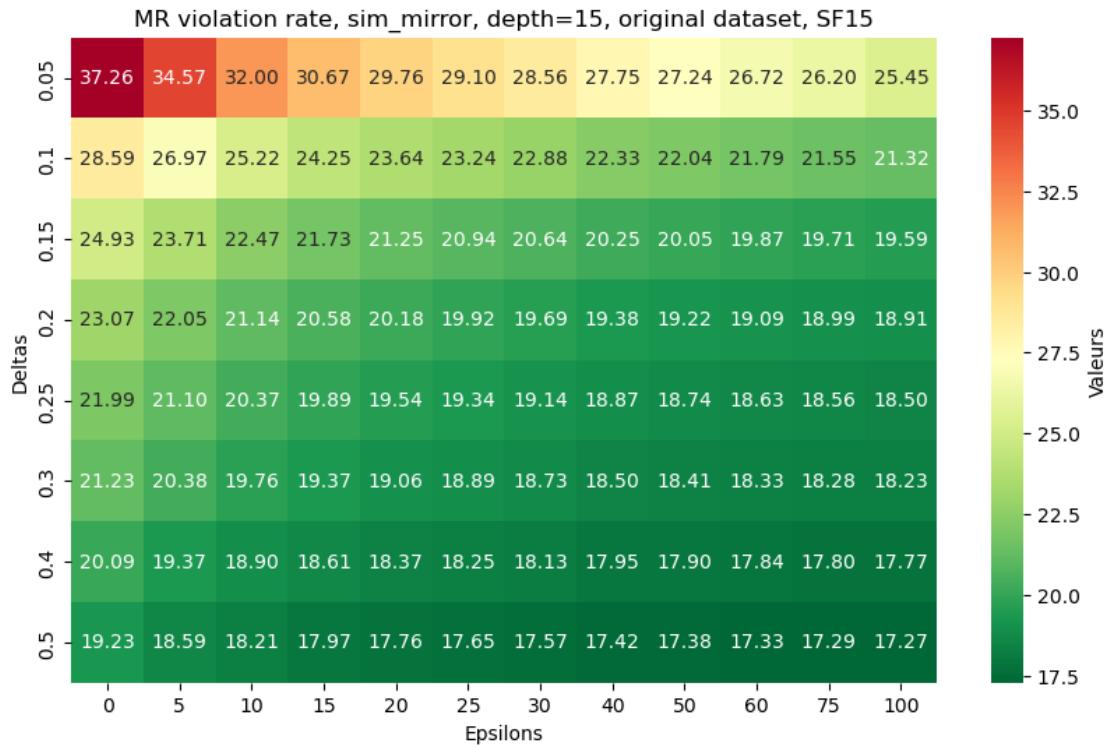
Fig. 8. Percentage of problems detected by $\text{MR}_{\text{mirror}}$. x-axis represents ϵ value and y-axis represents δ value.

The results are really similar, the minor differences we observe could be to system differences. Same comment for the other mutations, replace() aside.

depth = 15, entire dataset

```
[ ]: path = os.path.join('sim_mirror_d=15','evaluations50000_sim_mirror_d_15.pkl')
      with open(path, 'rb') as file:
          evaluations50000_sim_mirror_d_15 = pickle.load(file)

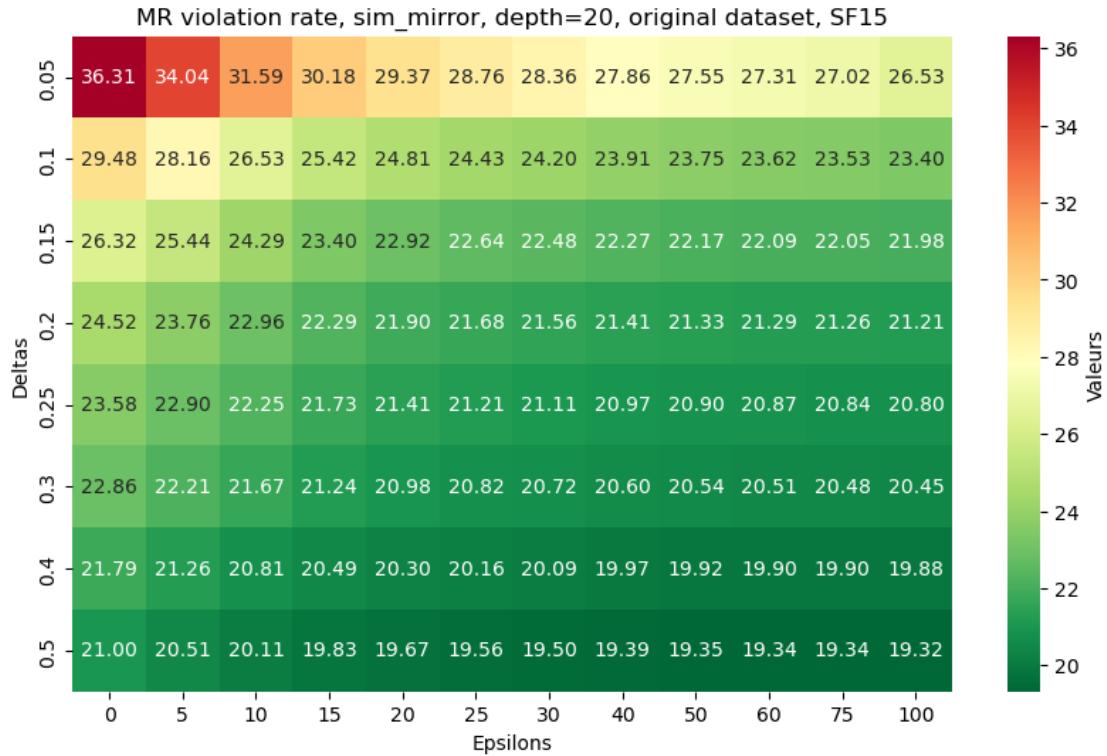
      tests(evaluations50000_sim_mirror_d_15,0,15, ' , original dataset, SF15')
```



depth = 20, entire dataset (Link V16)

```
[ ]: path = os.path.join('sim_mirror_d=20', 'evaluations50000_sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_mirror_d_20 = pickle.load(file)

tests(evaluations50000_sim_mirror_d_20, 0, 20, ', original dataset, SF15')
```



We can observe that despite increasing the depth, the results are not getting better, we could even say that it is worsening. We also observed that with each position mutations. We decided to check positions that had important deviations on depth 10 (that we found by setting important thresholds) when evaluating their equivalent counterpart. After checking many positions, we found that big differences are often observed when the advante for one side is big, in centipawns or when it is not far from a mate. Examples :

```
[ ]: pos1 = '1B4R1/7B/8/2K3n1/8/7k/R1r5/1B6 w'
pos2 = '1N5k/3Kb2q/1NP3p1/1R4B1/3R4/1B6/8/n3Q3 w'
show_pos([pos1,pos2],300)

print("pos1 : ",evaluationdouble(pos1,sim_mirror(pos1),50000,10))
print("pos2 : ",evaluationdouble(pos2,sim_axis(pos2),50000,10))
```

<IPython.core.display.HTML object>

```
pos1 :  ({'type': 'mate', 'value': 9}, {'type': 'cp', 'value': -2101})
pos2 :  ({'type': 'cp', 'value': 7305}, {'type': 'cp', 'value': 2850})
```

It seems that Stockfish starts to return high values when the positions are not far from a mate. If we increase the depth, the second evaluation also indicate a mate:

```
[ ]: print("pos1 : ",evaluationdouble(pos1,sim_mirror(pos1),50000,15))
print("pos2 : ",evaluationdouble(pos2,sim_axis(pos2),50000,15))
```

```
pos1 :  ({'type': 'mate', 'value': 6}, {'type': 'mate', 'value': -5})
pos2 :  ({'type': 'mate', 'value': 8}, {'type': 'mate', 'value': 7})
```

Reaching a certain depth, Stockfish returns the same values for both original and mutated positions.

```
[ ]: print("pos1 : ",evaluationdouble(pos1,sim_mirror(pos1),50000,20))
print("pos2 : ",evaluationdouble(pos2,sim_axis(pos2),50000,20))
```

```
pos1 :  ({'type': 'mate', 'value': 5}, {'type': 'mate', 'value': -5})
pos2 :  ({'type': 'mate', 'value': 7}, {'type': 'mate', 'value': 7})
```

We have to notice that MRs will always detect a problem if the evaluations' types are different OR if they are a mate for both positions but their values are not exactly the same. It is therefore strange to observe that increasing the depth does not improve the rate of respected MRs, but it could explain why that much problem are detected. We then checked the biggest deviations with depth 15.

```
[ ]: pos3 = '2N5/3k4/8/3B4/1K6/5n1r/8/4r3 b'
show_pos([pos3],300)
ev10 = evaluationdouble(pos3,sim_diag(pos3),50000,10)
print("pos3, d=10 : ", ev10, " - ", MR_equi(ev10[0],ev10[1],0.25,20))
ev15 = evaluationdouble(pos3,sim_diag(pos3),50000,15)
print("pos3, d=15 : ", ev15, " - ", MR_equi(ev15[0],ev15[1],0.25,20))
ev20 = evaluationdouble(pos3,sim_diag(pos3),50000,20)
print("pos3, d=20 : ", ev20, " - ", MR_equi(ev20[0],ev20[1],0.25,20))
ev25 = evaluationdouble(pos3,sim_diag(pos3),50000,25)
print("pos3, d=25 : ", ev25, " - ", MR_equi(ev25[0],ev25[1],0.25,20))
ev30 = evaluationdouble(pos3,sim_diag(pos3),50000,30)
print("pos3, d=30 : ", ev30, " - ", MR_equi(ev30[0],ev30[1],0.25,20))
```

```
<IPython.core.display.HTML object>

pos3, d=10 :  ({'type': 'cp', 'value': -1088}, {'type': 'cp', 'value': -1114})
- True
pos3, d=15 :  ({'type': 'cp', 'value': -5972}, {'type': 'cp', 'value': -1543})
- False
pos3, d=20 :  ({'type': 'mate', 'value': -11}, {'type': 'mate', 'value': -18})
- False
pos3, d=25 :  ({'type': 'mate', 'value': -9}, {'type': 'mate', 'value': -12}) - False
pos3, d=30 :  ({'type': 'mate', 'value': -9}, {'type': 'mate', 'value': -9}) - True
```

As we can see, positions with a big advantage for a side can have no problems when we use a low depth, then show big deviations when Stockfish starts to find a mate thanks to an increase in depth, and then converge when increasing the depth even more. That could explain why increasing the depth doesn't improve our results, since the large majority of the positions in our dataset are random positions, many of them have a strange and unlikely arrangement of pieces, often leading to an overwhelming advantage for a side. In the first time, we consequently decided to split positions where a mate is detected and positions where the difference is in centipawns.

Additional function

```
[ ]: def split_mate_cp(evaluations):
    pos1s_mate = []
    pos2s_mate = []
    ev1s_mate = []
    ev2s_mate = []
    indexy_mate = []
    pos1s_cp = []
    pos2s_cp = []
    ev1s_cp = []
    ev2s_cp = []
    indexy_cp = []
    index_cp = 0
    index_mate = 0
    for index, eva in evaluations.iterrows():
        if type(eva['evaluation pos1'])==list:
            if eva['evaluation pos1'][0].get('Mate')!=None or eva['evaluation pos2'][0].get('type')!=None:
                indexy_mate.append(index_mate)
                index_mate+=1
                pos1s_mate.append(eva['pos1'])
                pos2s_mate.append(eva['pos2'])
                ev1s_mate.append(eva['evaluation pos1'])
                ev2s_mate.append(eva['evaluation pos2'])
            else:
                indexy_cp.append(index_mate)
                index_cp += 1
                pos1s_cp.append(eva['pos1'])
                pos2s_cp.append(eva['pos2'])
                ev1s_cp.append(eva['evaluation pos1'])
                ev2s_cp.append(eva['evaluation pos2'])
        elif eva['evaluation pos1'].get('type')=='mate' or eva['evaluation pos2'].get('type')=='mate':
            indexy_mate.append(index_mate)
            index_mate+=1
            pos1s_mate.append(eva['pos1'])
            pos2s_mate.append(eva['pos2'])
            ev1s_mate.append(eva['evaluation pos1'])
            ev2s_mate.append(eva['evaluation pos2'])
        else:
            indexy_cp.append(index_mate)
            index_cp += 1
            pos1s_cp.append(eva['pos1'])
            pos2s_cp.append(eva['pos2'])
            ev1s_cp.append(eva['evaluation pos1'])
            ev2s_cp.append(eva['evaluation pos2'])
```

```

dmate = {'index': indexy_mate, 'pos1': pos1s_mate, 'evaluation pos1': ev1s_mate,
         'pos2': pos2s_mate, 'evaluation pos2': ev2s_mate}
dcp = {'index': indexy_cp, 'pos1': pos1s_cp, 'evaluation pos1': ev1s_cp,
        'pos2': pos2s_cp, 'evaluation pos2': ev2s_cp}
mate_evas = pd.DataFrame(data=dmate)
cp_evas = pd.DataFrame(data=dcp)
return cp_evas, mate_evas

```

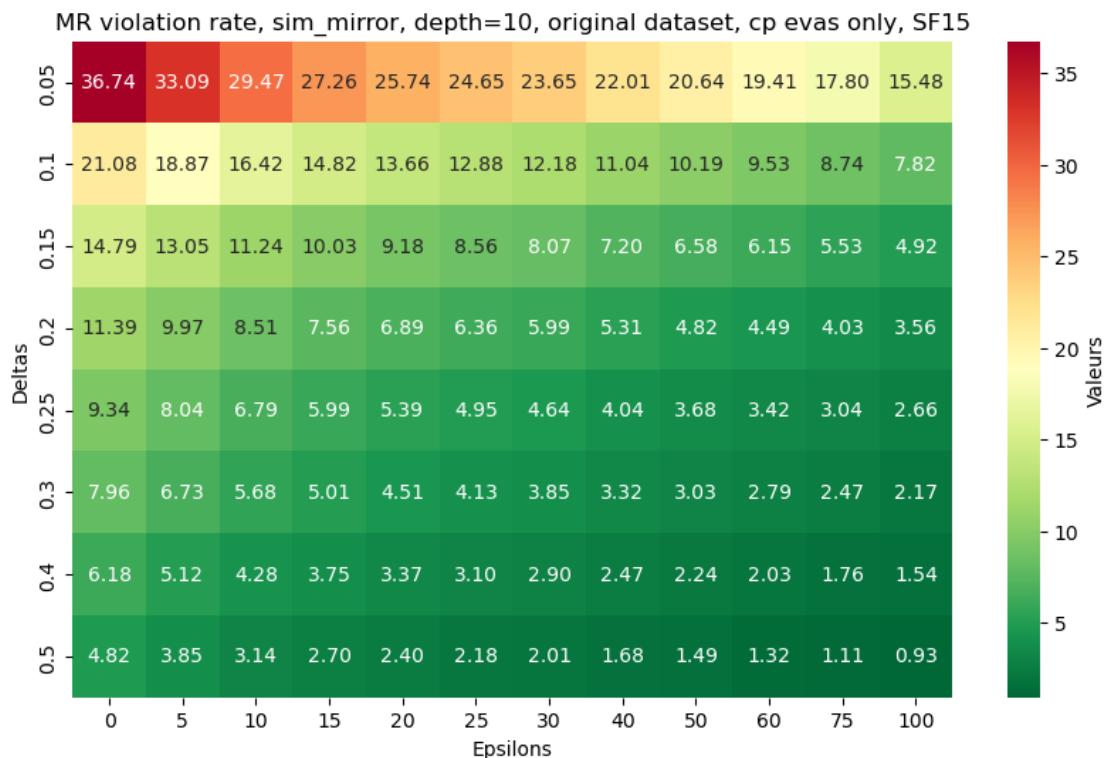
depth = 10, splitting (Link V16) We check the proportion of mate, and print the grids corresponding to the evaluations with a centipawns difference, then the ones with a mate, even though mate positions won't be affected by thresholds.

```
[ ]: print("Mates proportion : ", len(split_mate_cp(evaluations50000_sim_mirror_d_10)[1]) / (len(split_mate_cp(evaluations50000_sim_mirror_d_10)[0]) + len(split_mate_cp(evaluations50000_sim_mirror_d_10)[1]))
```

Mates proportion : 0.283332682622106

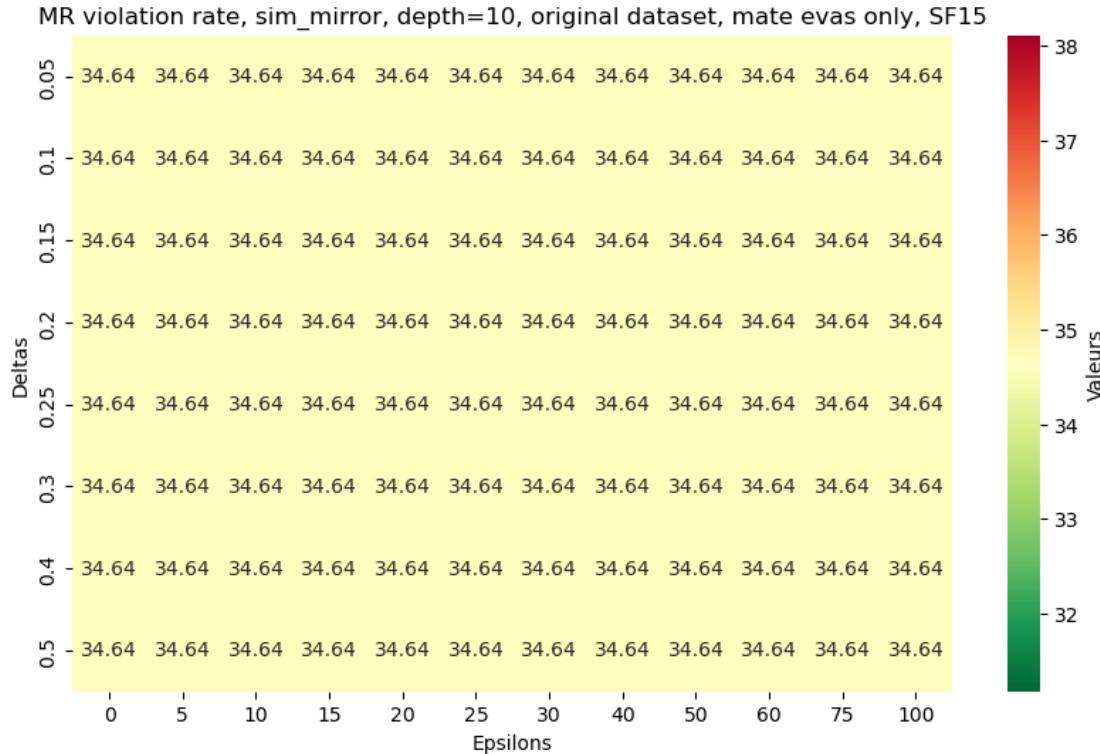
Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_10)[0], 0, 10, 'original dataset, cp evas only, SF15')
```



Evaluations with a mate

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_10)[1], 0, 10, ', original',
          ↪dataset, mate evas only, SF15')
```



As we said, mate positions are not affected by thresholds. Since 30.43% of the mate positions have problems detected by MRs, regardless of the thresholds every grid square sees its value affected by a 30.43% of failure rate with a weight of 26.6%. If we check only the positions with a difference in centipawns, we can see that many squares of the grid have now a way lower value than before.

Note : as the MRs were designed by the original article, MRs are violated if the type is mate and value isn't exactly the same after mutation. Actually on these positions, there is no case where both evaluations show a mate but indicate values with an opposite sign.

Now we'll try to increase the depth.

depth = 15, splitting

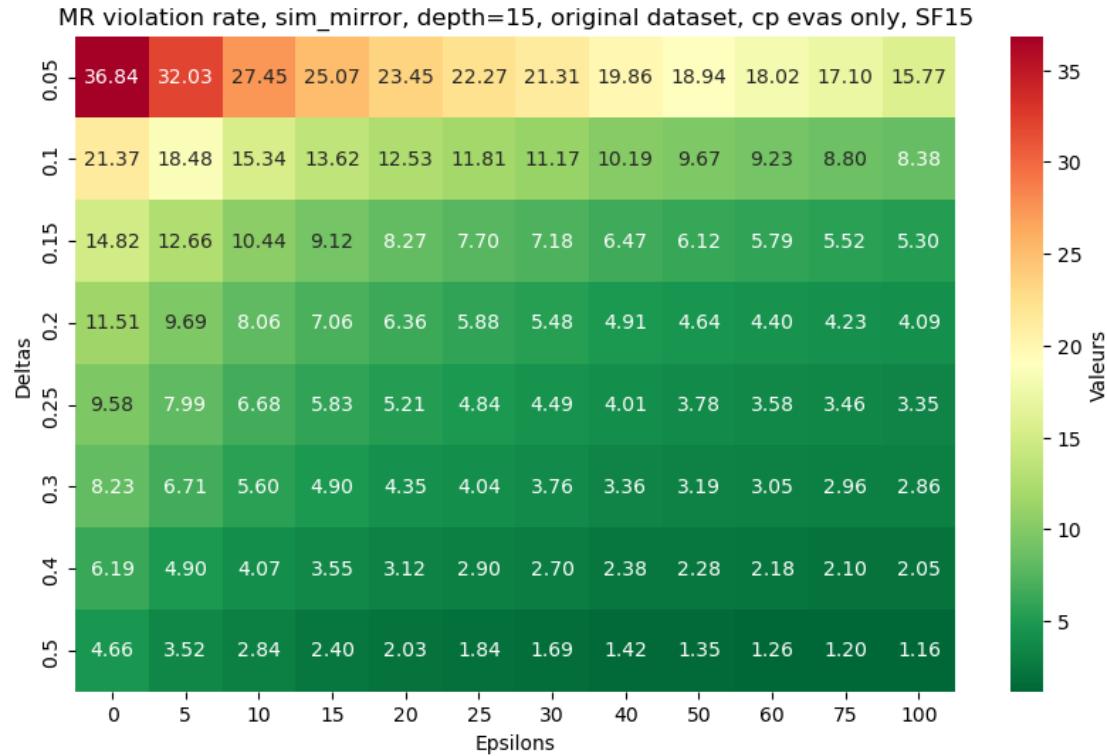
```
[ ]: print("Mates proportion : ", ↪
          ↪len(split_mate_cp(evaluations50000_sim_mirror_d_15)[1]) /
          ↪(len(split_mate_cp(evaluations50000_sim_mirror_d_15)[0])+len(split_mate_cp(evaluations50000
```

Mates proportion : 0.4397500976181179

The mates proportion became higher, which may be the reason why increasing the depth was not improving our results.

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_15)[0], 0, 15, ', original',
          ↪dataset, cp evas only, SF15')
```



Evaluations with a mate

```
[ ]: print('Mates failure rate :',
          ↪', 100-MR(split_mate_cp(evaluations50000_sim_mirror_d_15)[1], 0, 1, 1), '%')
```

Mates failure rate : 37.79968034096963 %

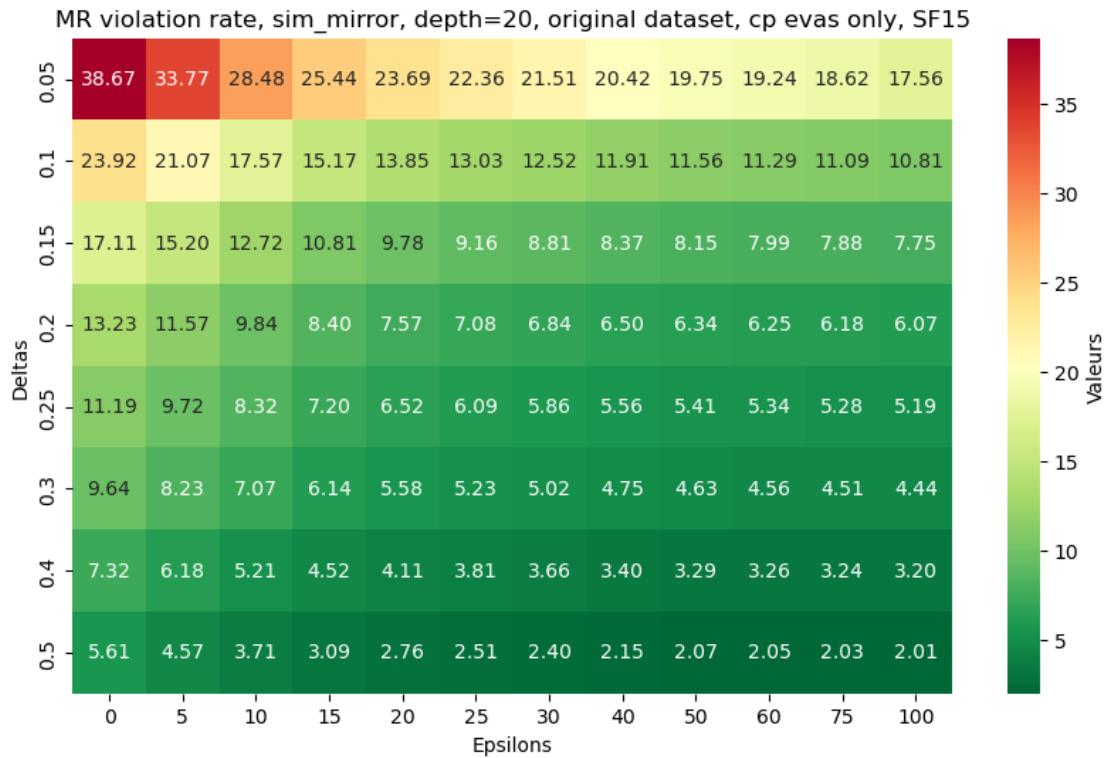
depth = 20, splitting (Link V16)

```
[ ]: print("Mates proportion : ", ↪
          ↪len(split_mate_cp(evaluations50000_sim_mirror_d_20)[1])/ ↪
          ↪(len(split_mate_cp(evaluations50000_sim_mirror_d_20)[0])+len(split_mate_cp(evaluations50000
```

Mates proportion : 0.5365958612056488

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_20)[0], 0, 20, ', original',
          ↪dataset, cp evas only, SF15')
```



Evaluations with a mate

```
[ ]: print('Mates failure rate :')
      ↵',100-MR(split_mate_cp(evaluations50000_sim_mirror_d_20)[1],0,1,1),'%')
```

Mates failure rate : 34.27863235401327 %

Increasing the depth did not improve our results, now we will make the grid for only the positions that were taken from real games. These positions are the 1234 last ones in our dataset.

depth = 10, real positions (Link V16)

```
[ ]: #function to get real positions
def real_positions(evas):
    real = []
    real2 = []
    indexy = []
    ev1s = []
    ev2s = []
    index = 0
    real_pos = dataset[24768:]
    #indexent2 = 0
    for indexent, eva in evas.iterrows():
        for pos in real_pos:
            if eva['pos1'] == pos and eva['pos1'] not in real:
```

```

    #print(indexent - indexent2, index)
    #indexent2 = indexent
    indexy.append(index)
    index+=1
    real.append(eva['pos1'])
    real2.append(eva['pos2'])
    ev1s.append(eva['evaluation pos1'])
    ev2s.append(eva['evaluation pos2'])
d = {'index': indexy, 'pos1': real, 'evaluation pos1': ev1s,
      'pos2': real2, 'evaluation pos2': ev2s}
df = pd.DataFrame(data=d)
return df

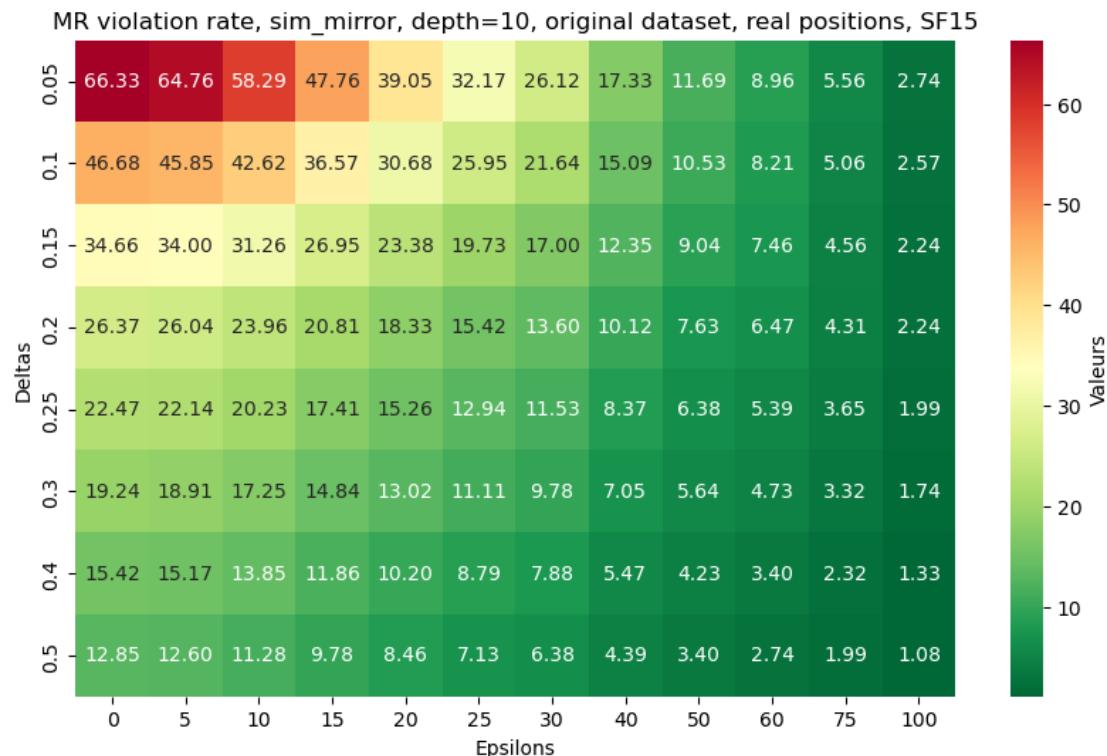
```

[]:

```

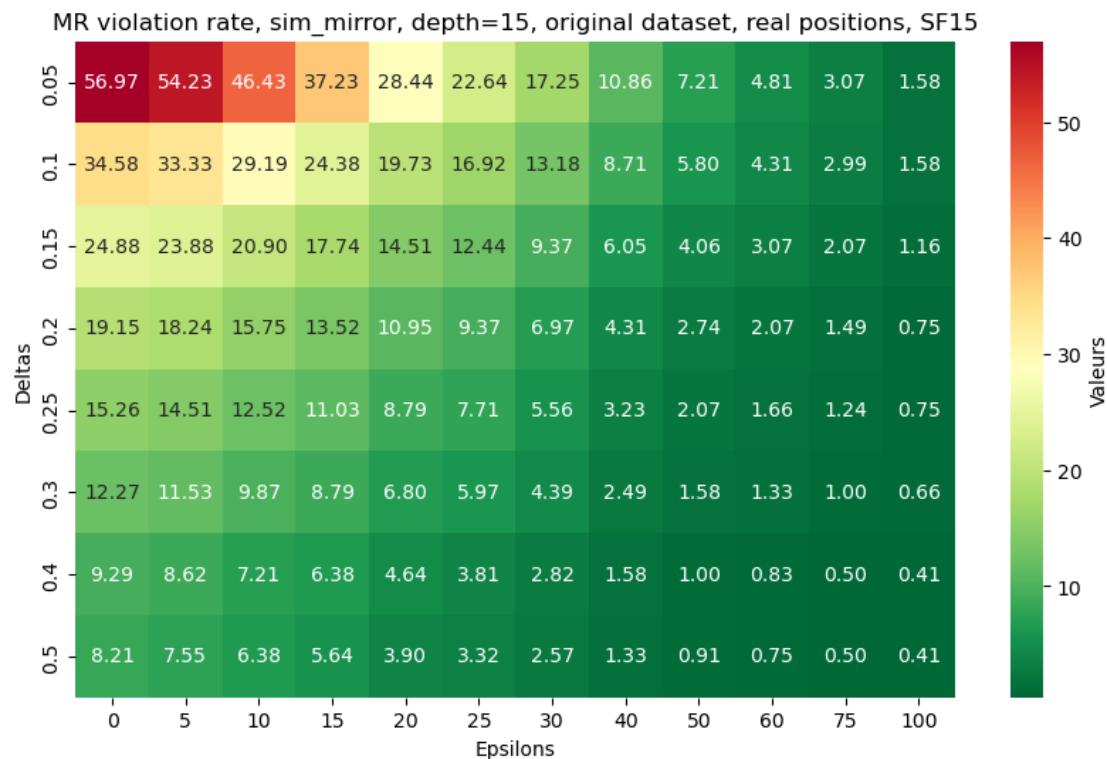
evaluations50000_sim_mirror_d_10_real = real_positions(evaluations50000_sim_mirror_d_10[24000:])
evaluations50000_sim_mirror_d_10_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_mirror_d_10_real, 0, 10, ', original dataset, real positions, SF15')

```



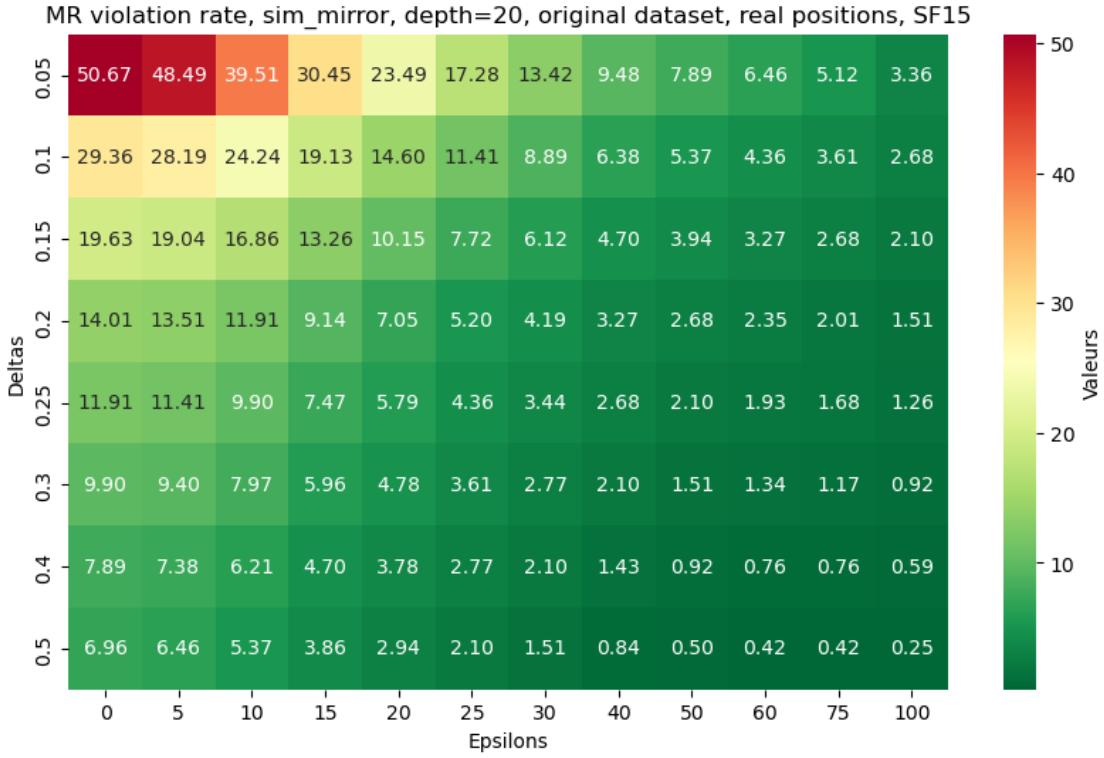
depth = 15, real positions

```
[ ]: evaluations50000_sim_mirror_d_15_real = real_positions(evaluations50000_sim_mirror_d_15[24000:])
evaluations50000_sim_mirror_d_15_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_mirror_d_15_real, 0, 15, ', original dataset, real_positions, SF15')
```



depth = 20, real positions (Link V16)

```
[ ]: evaluations50000_sim_mirror_d_20_real = real_positions(evaluations50000_sim_mirror_d_20[24000:])
evaluations50000_sim_mirror_d_20_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_mirror_d_20_real, 0, 20, ', original dataset, real_positions, SF15')
```



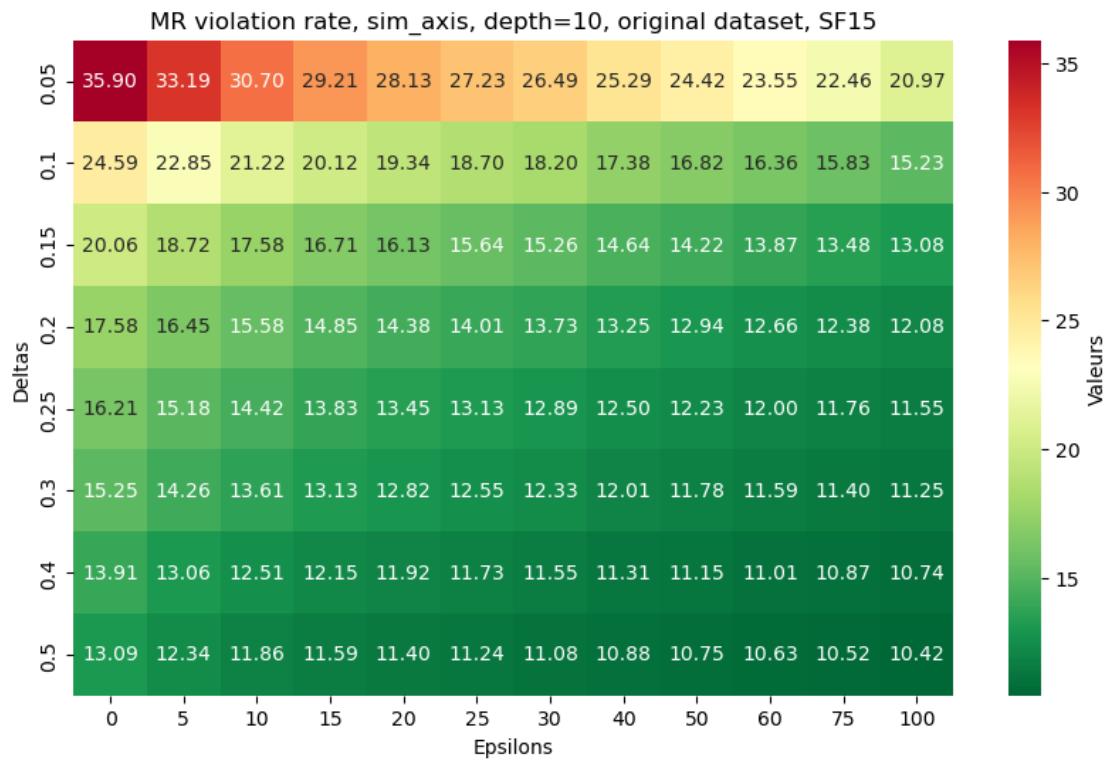
On plausible positions, we can see that beside for the tightest thresholds, the number of problems detected by the MRs are way lower, and increasing the depth is now improving our results. We can explain the increase in detected problems for tight thresholds with the fact that on real positions, centipawns values tend to be really lower than before. If for instance, we have a position with an evaluation with a centipawns value of 5 (meaning an advantage of 0.05 for White), and its mirrored version gets an evaluation with a centipawns value of -7 (meaning an advantage of 0.07 for Black), the delta value will be $\text{abs}((5-7)/(5+7)) = 0.17$, which is superior to 0.05. Since the centipawns difference is not equal to 0, the MRs will detect a problem, even though the difference is meaningless.

0.8.2 sim_axis

depth = 10, entire dataset (Link V16) Our grid in the first time.

```
[ ]: path = os.path.join('sim_axis_d=10','evaluations50000_sim_axis_d_10.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_axis_d_10 = pickle.load(file)

tests(evaluations50000_sim_axis_d_10,1,10, ' , original dataset, SF15')
```



```
[ ]: from IPython.display import display, Image
display(Image(filename='originalaxis10_am.png',width=900,height=900))
```

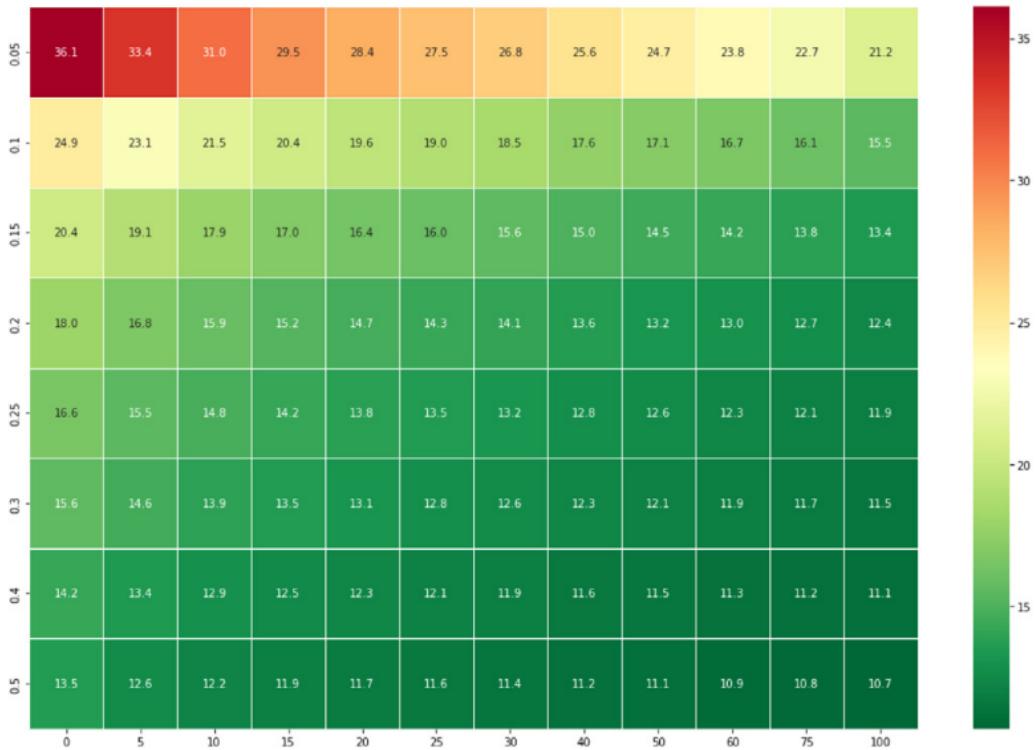


Fig. 6. Percentage of problems detected by MR_{equi} using the `sim_axis(p)` transformation. x-axis represents ϵ value and y-axis represents δ value.

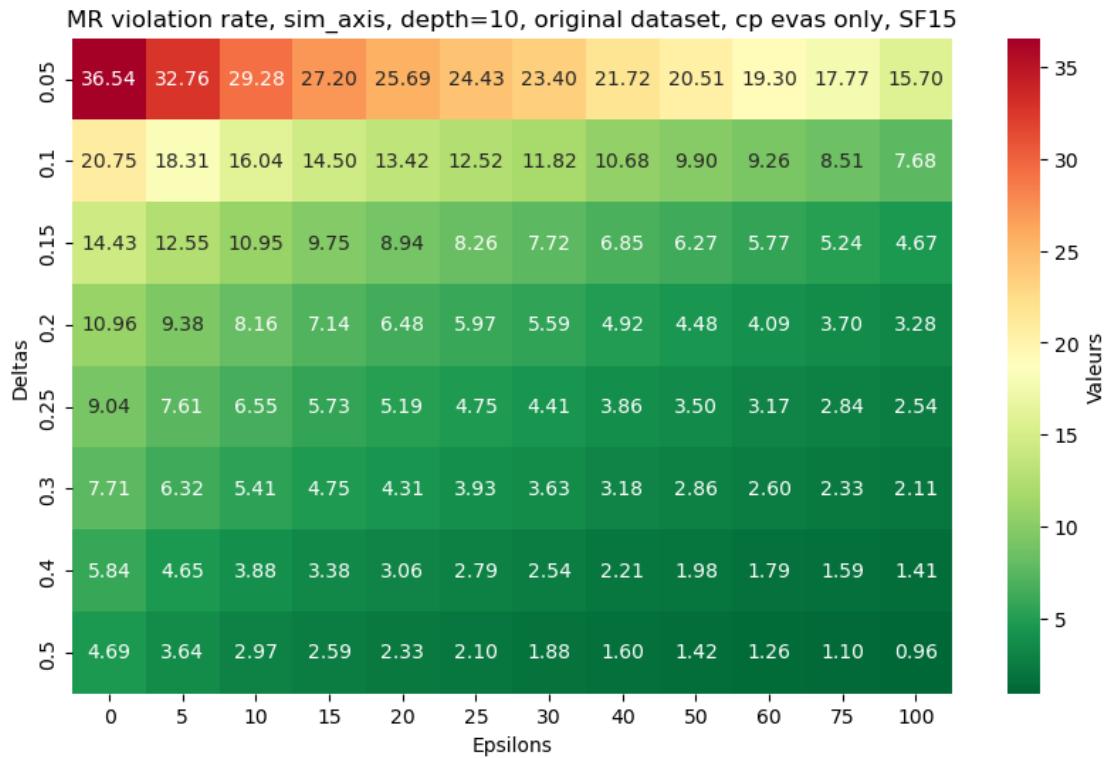
depth = 10, splitting (Link V16)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_sim_axis_d_10)[1])/  
    ↪(len(split_mate_cp(evaluations50000_sim_axis_d_10)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.28385690373311534

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_axis_d_10)[0], 1, 10, 'original'  
    ↪dataset, cp evas only, SF15')
```



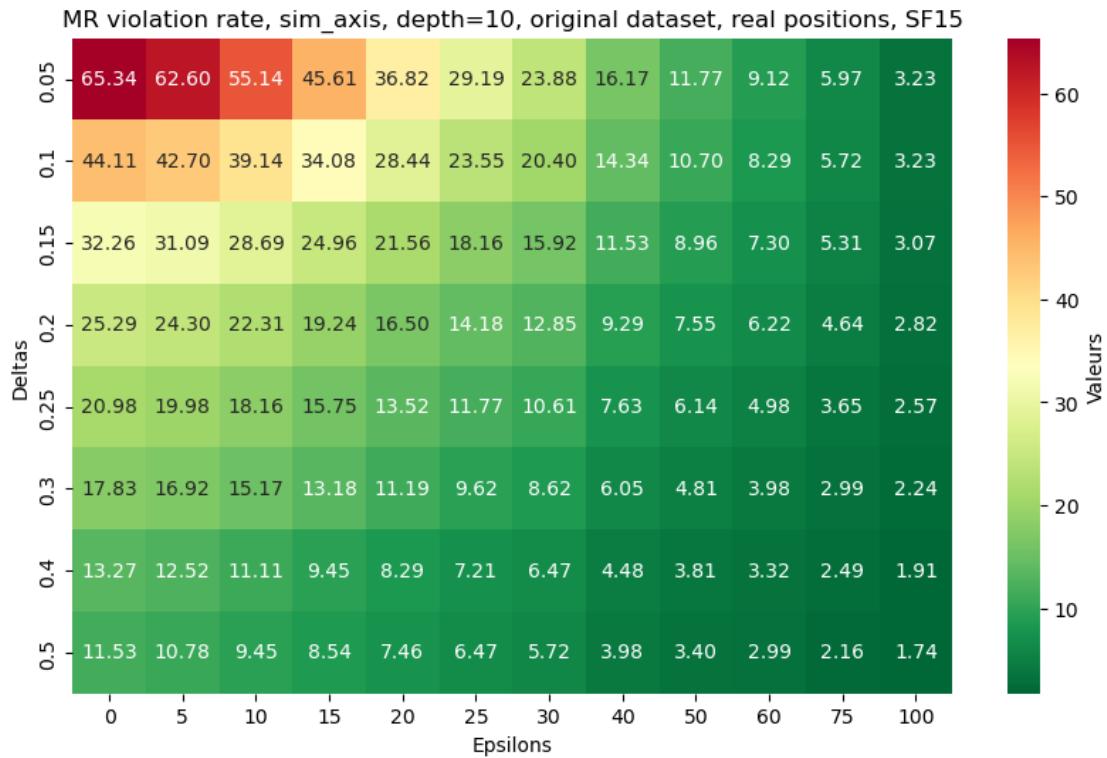
Evaluations with a mate

```
[ ]: print('Mates failure rate :')
      ↵', 100-MR(split_mate_cp(evaluations50000_sim_axis_d_10)[1],1,1,1), '%')
```

Mates failure rate : 34.28414701042239 %

depth = 10, real positions

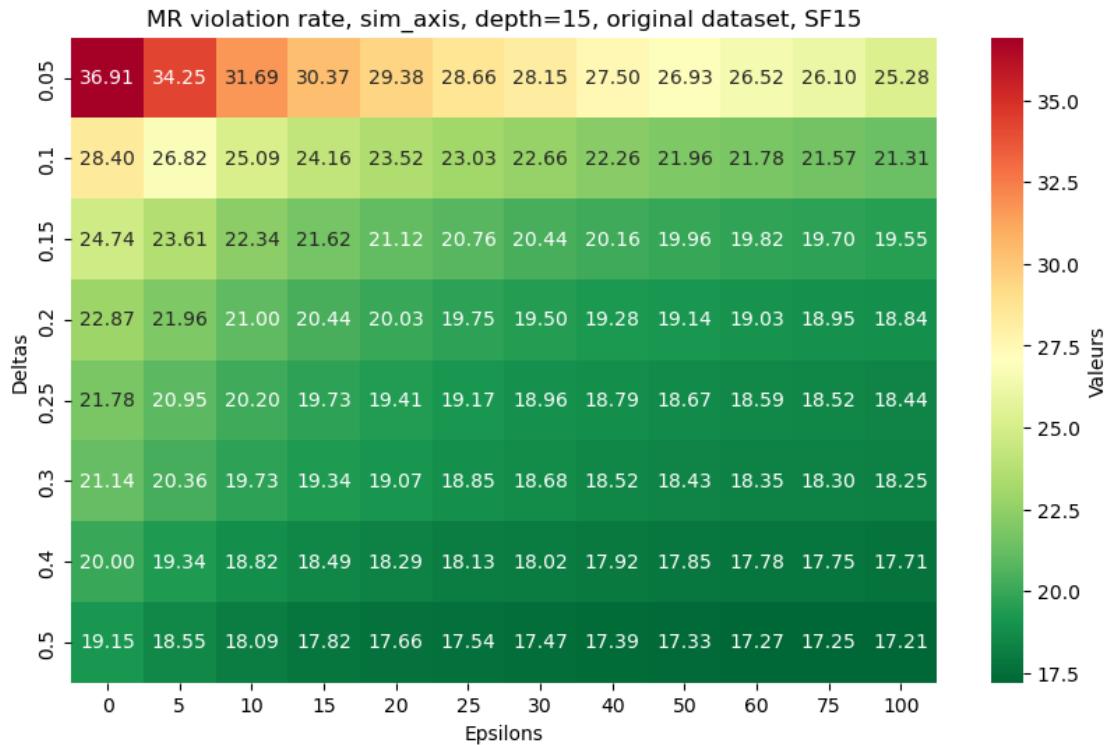
```
[ ]: evaluations50000_sim_axis_d_10_real =
      ↵real_positions(evaluations50000_sim_axis_d_10[24000:])
evaluations50000_sim_axis_d_10_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_axis_d_10_real, 1, 10, ', original dataset, real
      ↵positions, SF15')
```



depth = 15, entire dataset

```
[ ]: path = os.path.join('sim_axis_d=15', 'evaluations50000_sim_axis_d_15.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_axis_d_15 = pickle.load(file)

tests(evaluations50000_sim_axis_d_15, 1, 15, ', original dataset, SF15')
```



depth = 15, splitting

```
[ ]: print("Mates proportion : ", ↴
    ↴len(split_mate_cp(evaluations50000_sim_axis_d_15)[1])/
    ↴(len(split_mate_cp(evaluations50000_sim_axis_d_15)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.44193560945225213

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_axis_d_15)[0], 1, 15, ', original', ↴
    ↴dataset, cp evas only, SF15')
```

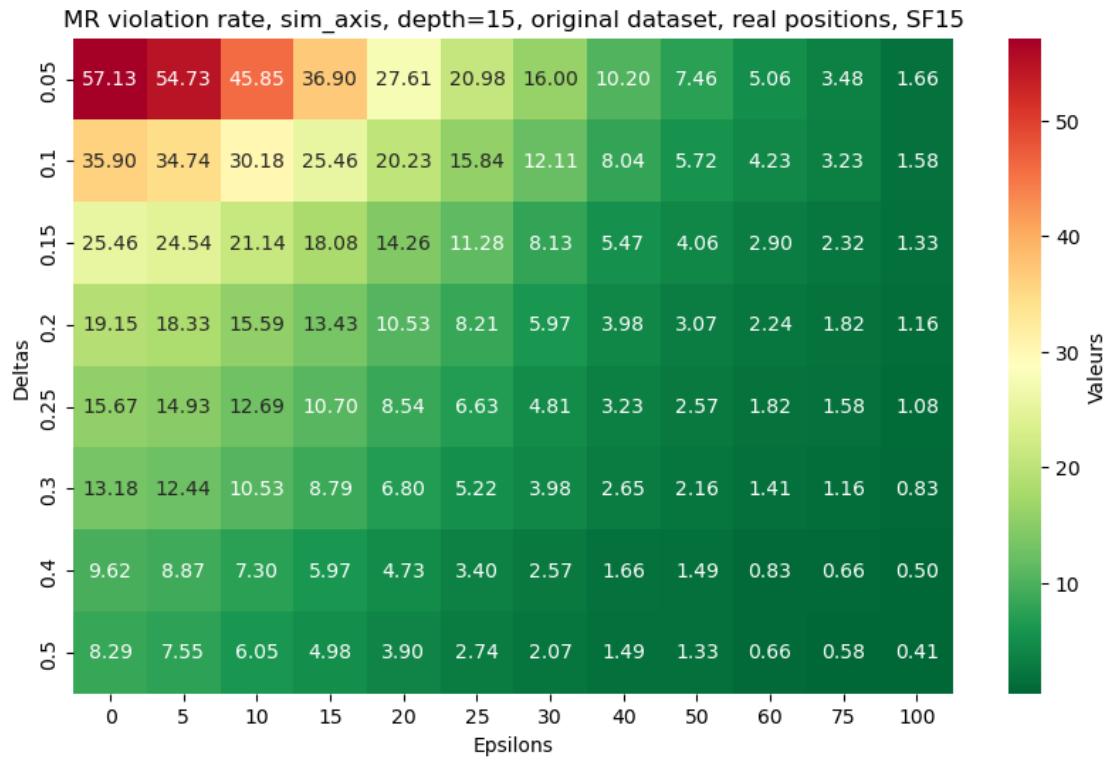


```
[ ]: print('Mates failure rate :')
      ↪', 100-MR(split_mate_cp(evaluations50000_sim_axis_d_15)[1], 1, 1, 1), '%')
```

Mates failure rate : 37.66737138830162 %

depth = 15, real positions

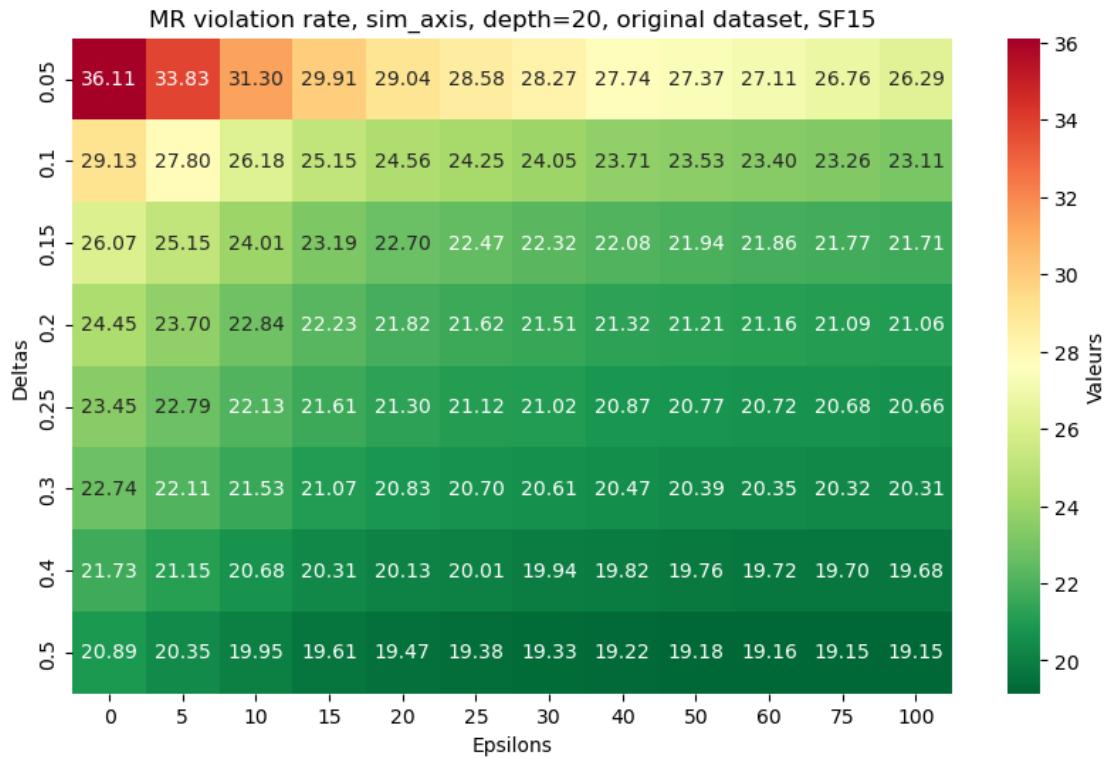
```
[ ]: evaluations50000_sim_axis_d_15_real =
      ↪real_positions(evaluations50000_sim_axis_d_15[24000:])
evaluations50000_sim_axis_d_15_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_axis_d_15_real, 1, 15, ', original dataset, real
      ↪positions, SF15')
```



depth = 20, entire dataset (Link V16)

```
[ ]: path = os.path.join('sim_axis_d=20','evaluations50000_sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_axis_d_20 = pickle.load(file)

tests(evaluations50000_sim_axis_d_20, 1, 20, 'original dataset, SF15')
```



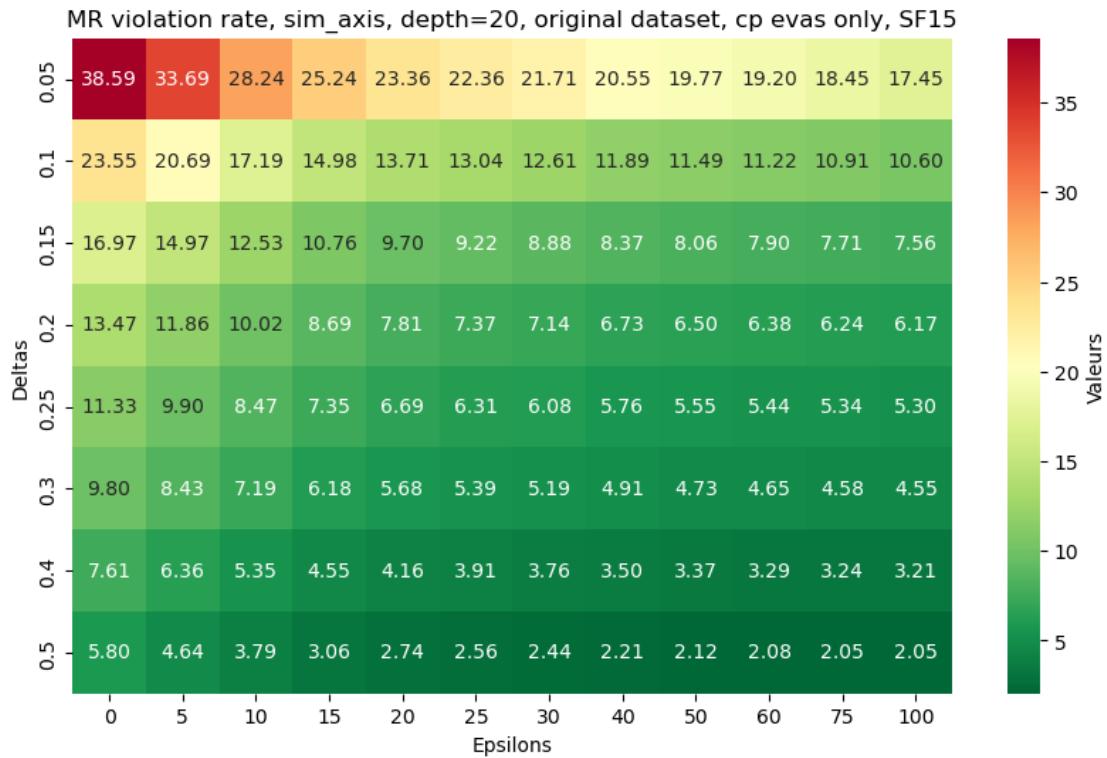
depth = 20, splitting (Link V16)

```
[ ]: print("Mates proportion : ",  
      ↪len(split_mate_cp(evaluations50000_sim_axis_d_20)[1])/  
      ↪(len(split_mate_cp(evaluations50000_sim_axis_d_20)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.5357728017476788

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_axis_d_20)[0],1,20, ' , original  
      ↪dataset, cp evas only, SF15')
```

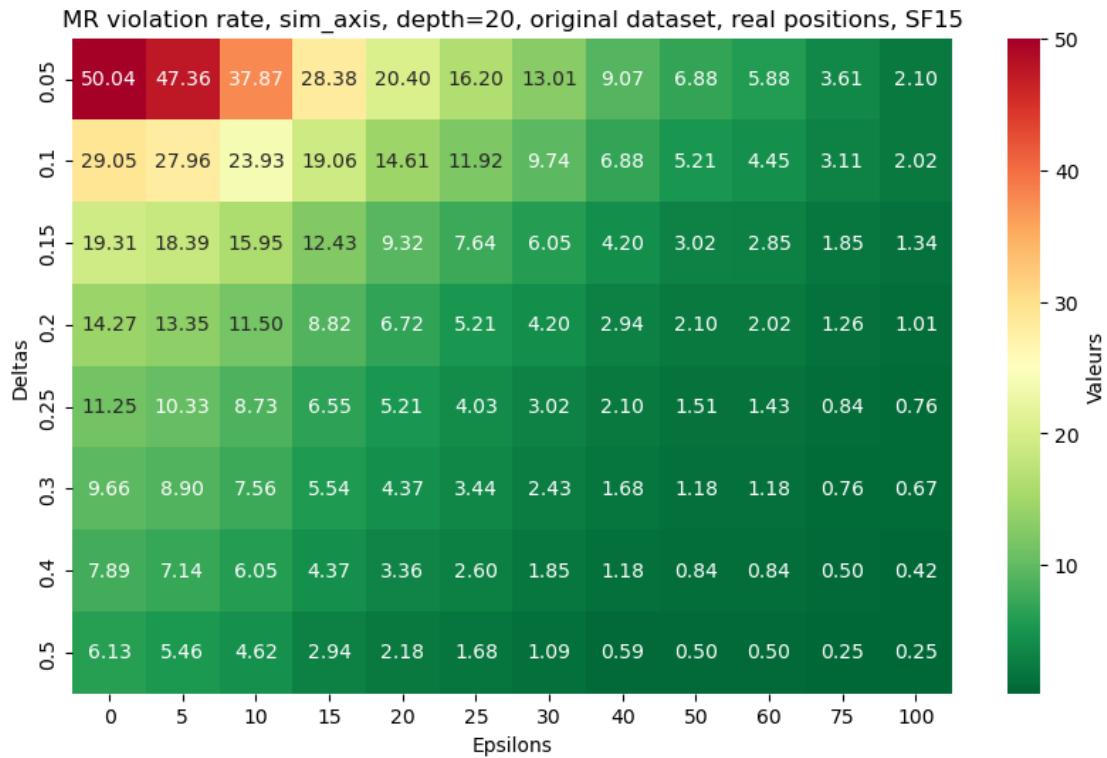


```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_sim_axis_d_20)[1], 1, 1, 1), '%')
```

Mates failure rate : 33.95951652832386 %

depth = 20, real positions (Link V16)

```
[ ]: evaluations50000_sim_axis_d_20_real =
    ↪real_positions(evaluations50000_sim_axis_d_20[24000:])
evaluations50000_sim_axis_d_20_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_axis_d_20_real, 1, 20, ', original dataset, real'
    ↪positions, SF15')
```

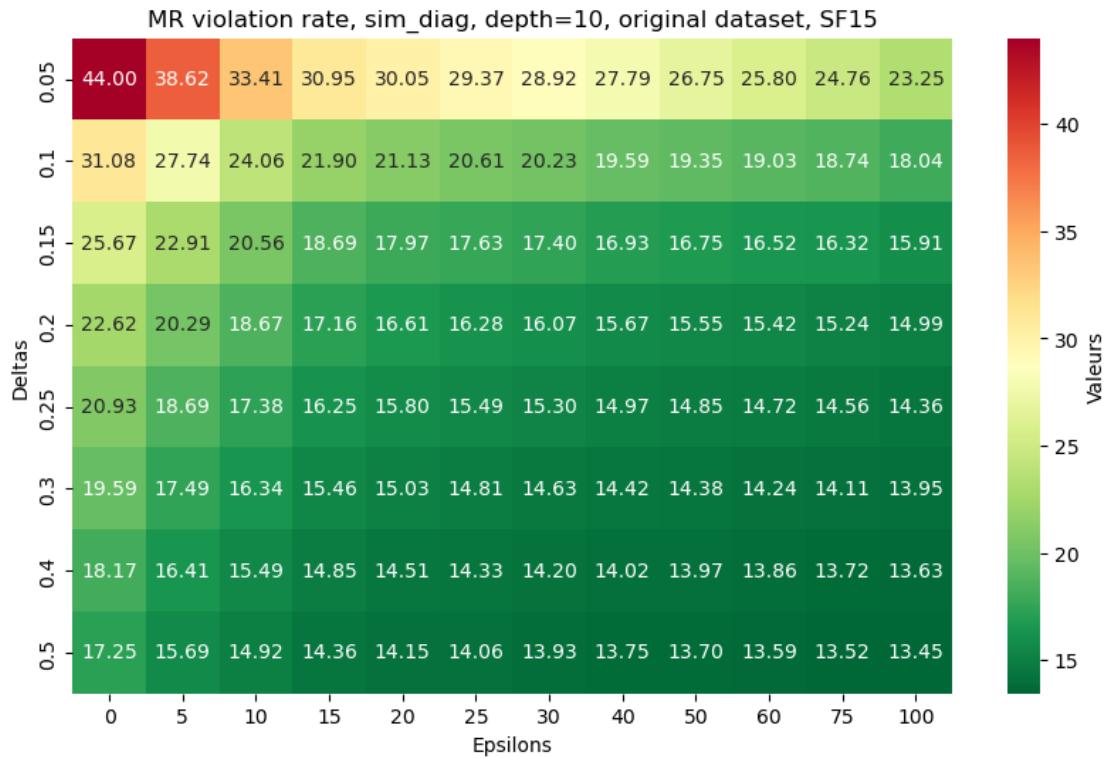


0.8.3 sim_diag

depth = 10, entire dataset Our grid.

```
[ ]: path = os.path.join('sim_diag_d=10', 'evaluations50000_sim_diag_d_10.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_diag_d_10 = pickle.load(file)

tests(evaluations50000_sim_diag_d_10, 2, 10, ', original dataset, SF15')
```



The grid from the article.

```
[ ]: from IPython.display import display, Image
display(Image(filename='originaldiag10_am.png',width=900,height=900))
```

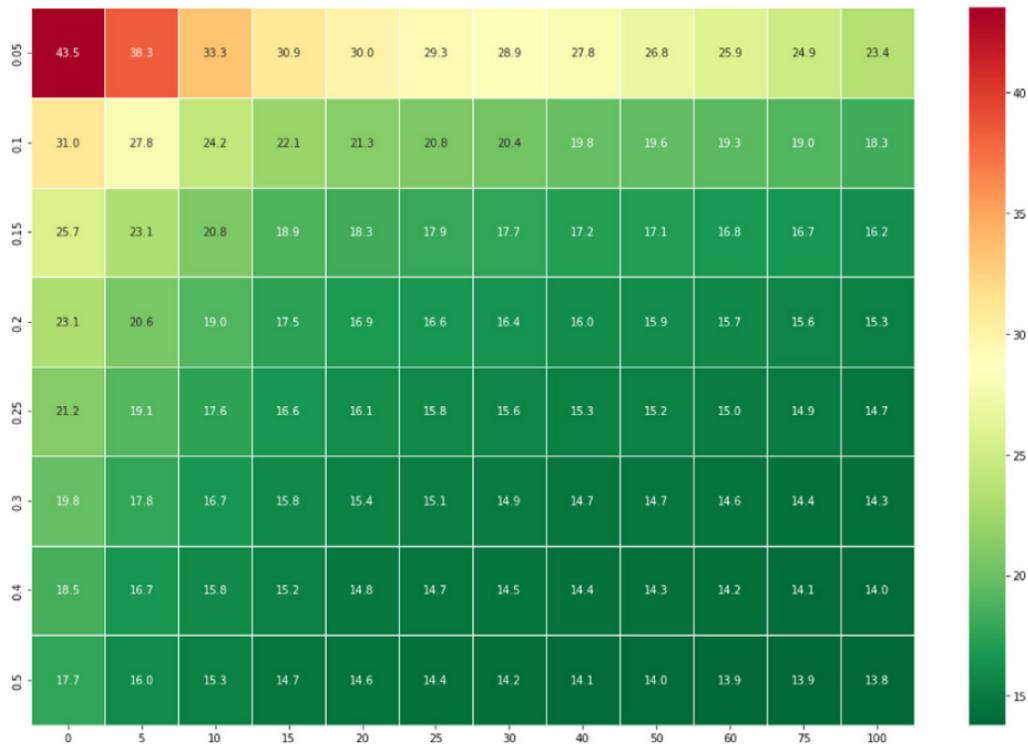


Fig. 7. Percentage of problems detected by MR_{equi} using the $\text{sim_diag}(p)$ transformation. x-axis represents ϵ value and y-axis represents δ value.

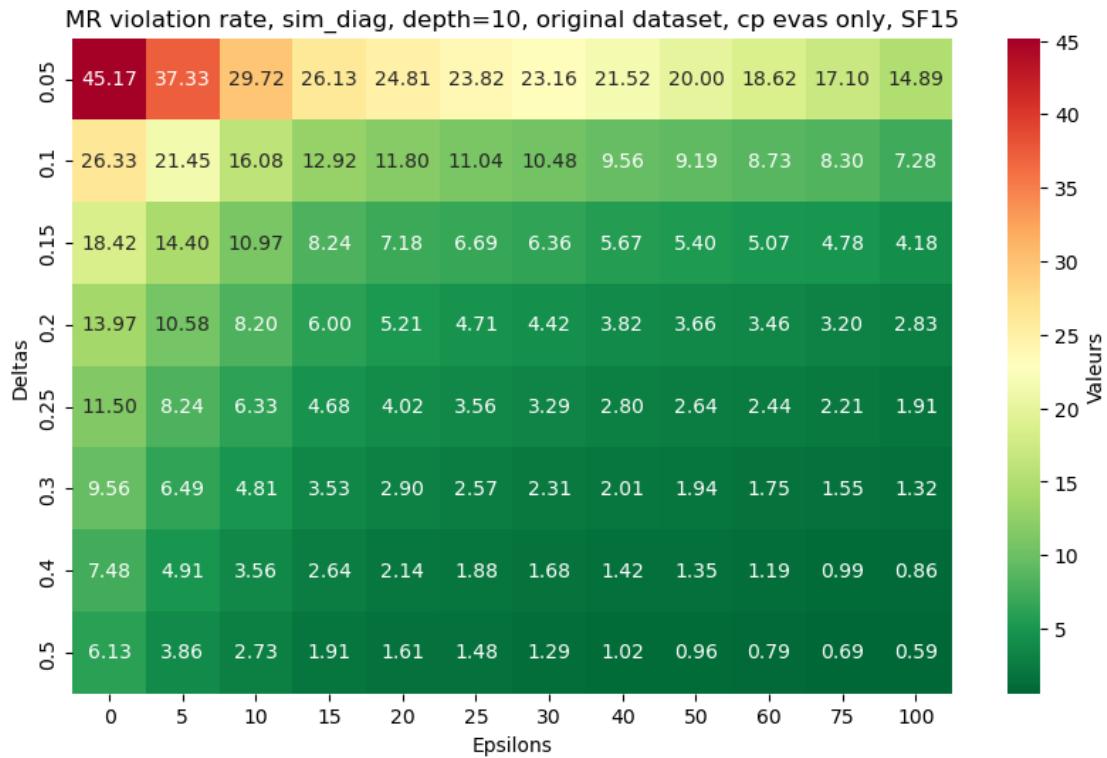
depth = 10, splitting

```
[ ]: print("Mates proportion : ", ↴
    ↴len(split_mate_cp(evaluations50000_sim_diag_d_10)[1])/
    ↴(len(split_mate_cp(evaluations50000_sim_diag_d_10)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.3148984198645598

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_diag_d_10)[0], 2, 10, ', original
    ↴dataset, cp evas only, SF15')
```



```
[ ]: print('Mates failure rate :')
[ ]:     ↪', 100-MR(split_mate_cp(evaluations50000_sim_diag_d_10)[1], 2, 1, 1), '%')
```

Mates failure rate : 41.43369175627241 %

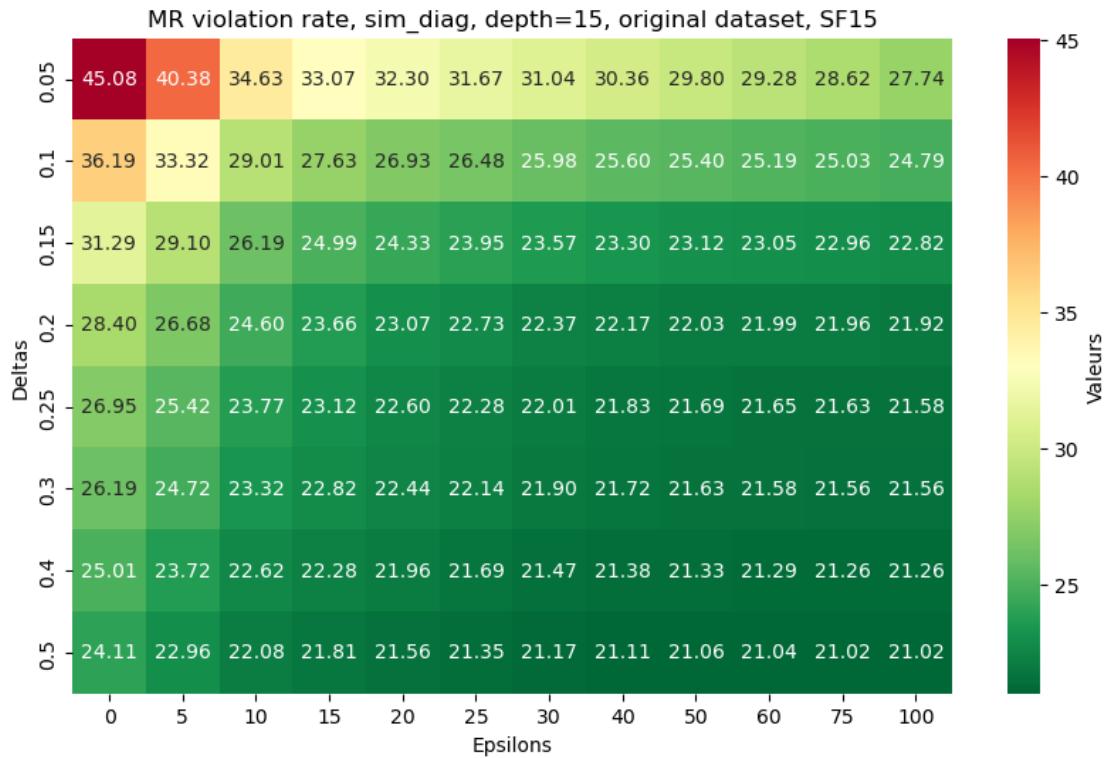
depth = 10, real positions Since this mutation only works on positions without any pawn, its interest on real positions is deeply lowered. Furthermore, our dataset doesn't contain real positions with no pawn.

```
[ ]: len(real_positions(evaluations50000_sim_diag_d_10))
```

```
[ ]: 0
```

depth = 15, entire dataset

```
[ ]: path = os.path.join('sim_diag_d=15', 'evaluations50000_sim_diag_d_15.pkl')
[ ]: with open(path, 'rb') as file:
[ ]:     evaluations50000_sim_diag_d_15 = pickle.load(file)
[ ]: tests(evaluations50000_sim_diag_d_15, 2, 15, 'original dataset, SF15')
```



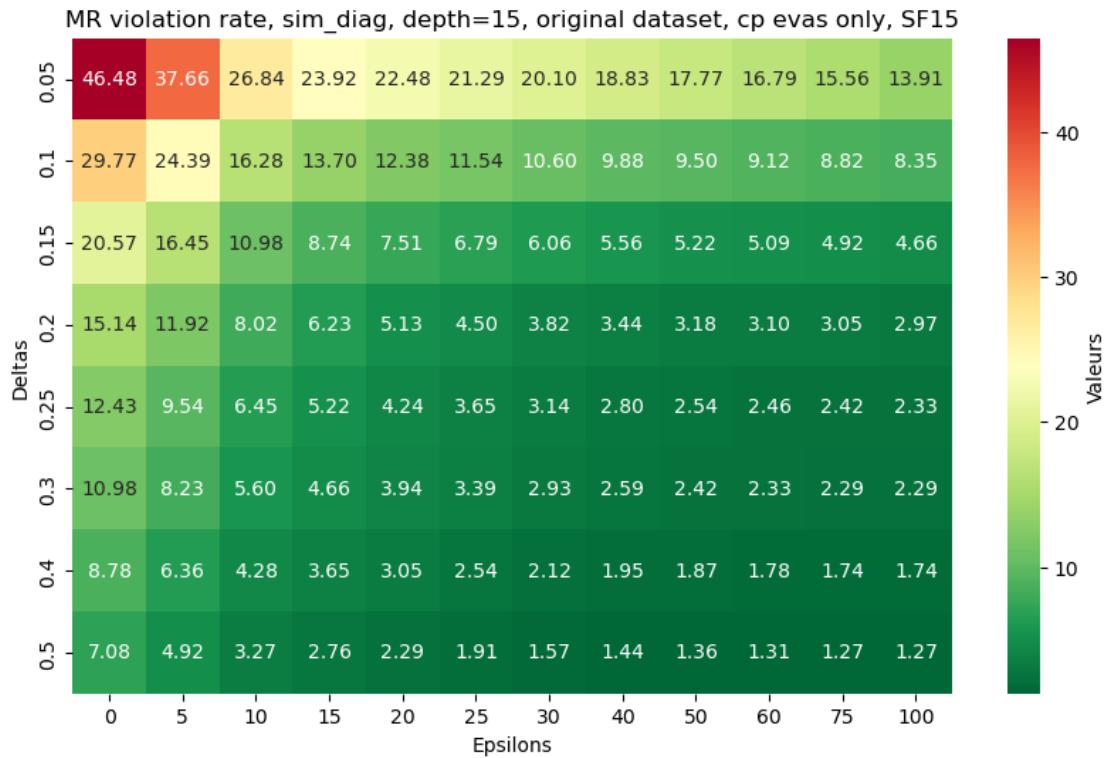
depth = 15, splitting

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_sim_diag_d_15)[1])/  
    ↪(len(split_mate_cp(evaluations50000_sim_diag_d_15)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.46772009029345374

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_diag_d_15)[0],2,15, ' , original  
    ↪dataset, cp evas only, SF15')
```



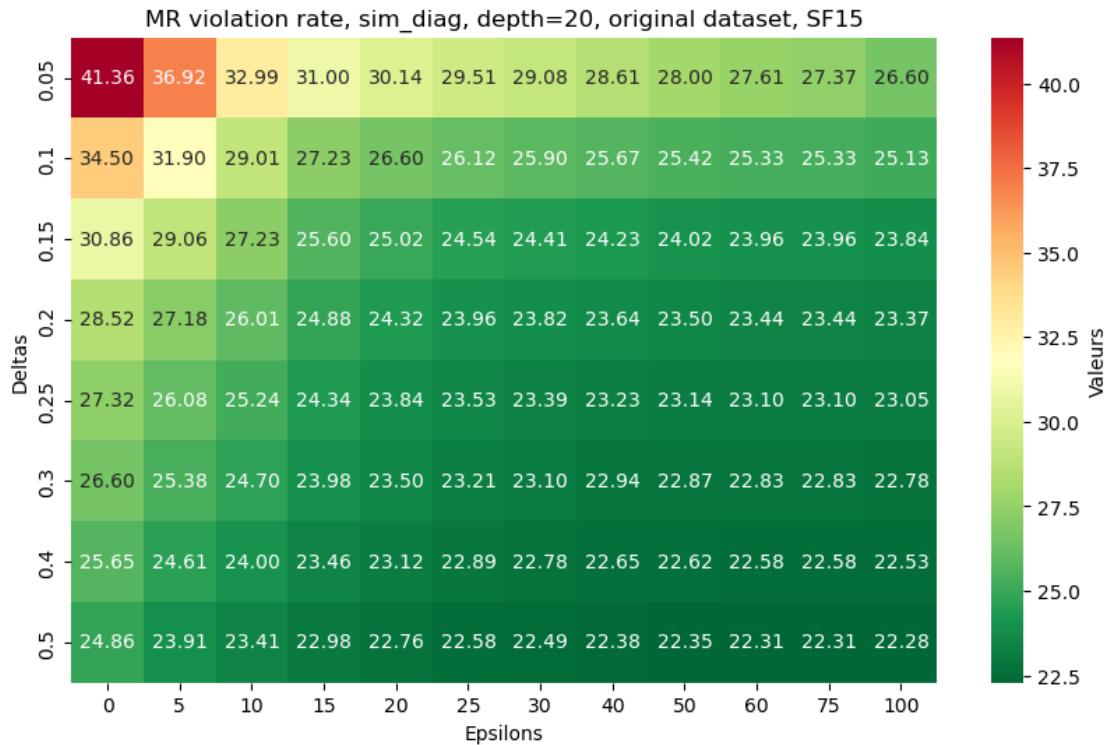
```
[ ]: print('Mates failure rate :')
    ↵',100-MR(split_mate_cp(evaluations50000_sim_diag_d_15)[1],2,1,1),'%')
```

Mates failure rate : 43.48455598455598 %

depth = 20, entire dataset (Link V16)

```
[ ]: path = os.path.join('sim_diag_d=20','evaluations50000_sim_diag_d_20.pkl')
with open(path, 'rb') as file:
    evaluations50000_sim_diag_d_20 = pickle.load(file)

tests(evaluations50000_sim_diag_d_20,2,20, ', original dataset, SF15')
```



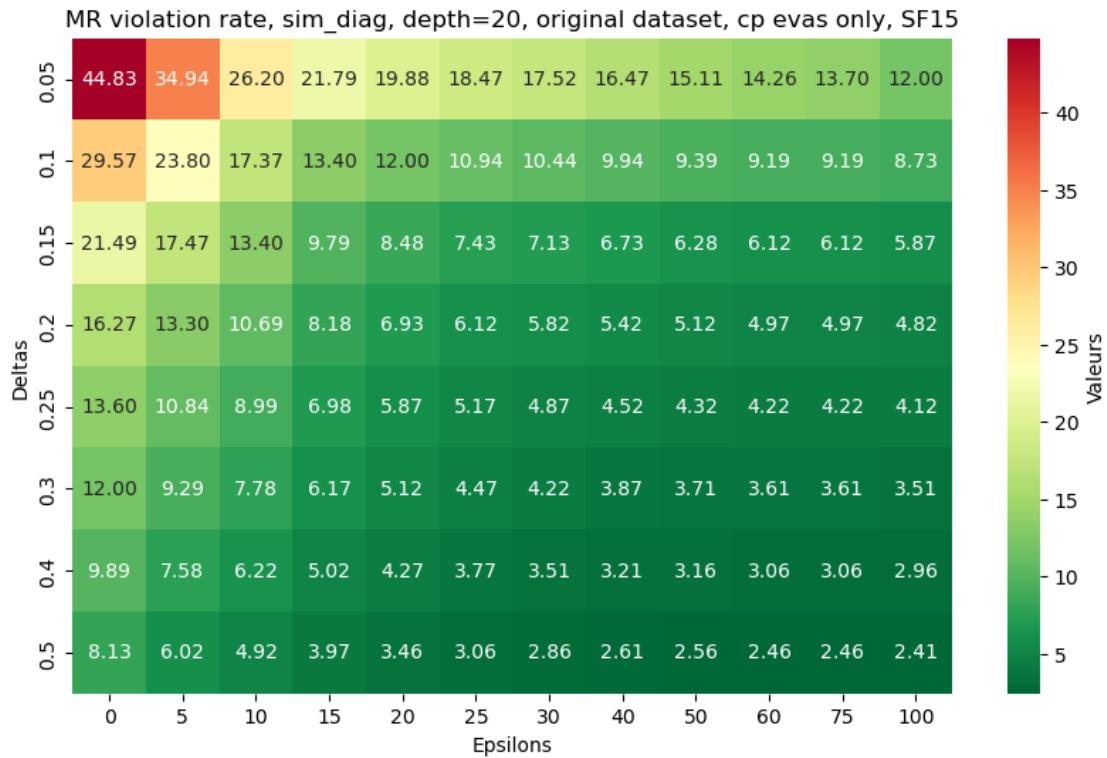
depth = 20, splitting (Link V16)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_sim_diag_d_20)[1])/  
    ↪(len(split_mate_cp(evaluations50000_sim_diag_d_20)[0])+len(split_mate_cp(evaluations50000_s
```

Mates proportion : 0.5502370738315647

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_diag_d_20)[0],2,20, 'original'  
    ↪dataset, cp evas only, SF15')
```



```
[ ]: print('Mates failure rate :')
    ↪',100-MR(split_mate_cp(evaluations50000_sim_diag_d_20)[1],2,1,1),'%')
```

Mates failure rate : 38.53098071399261 %

0.8.4 replace

depth = 10, entire dataset Our grid (note, the MR was slightly edited).

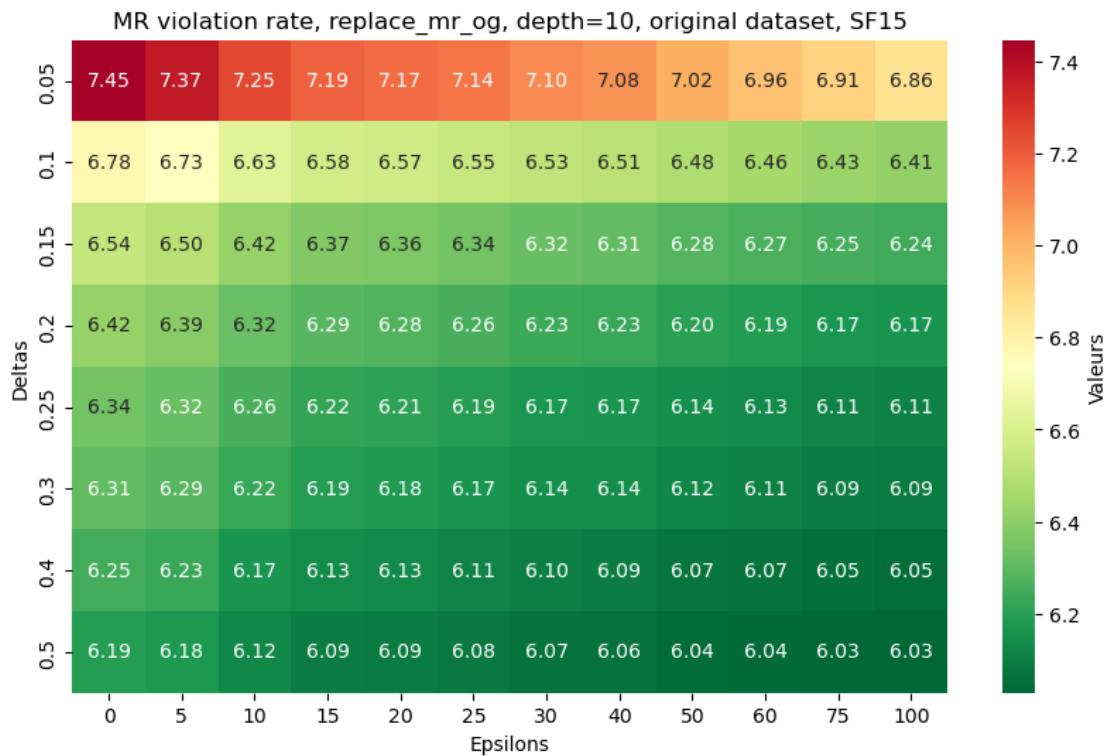
```
[ ]: path = os.path.join('replace_d=10','evaluations50000_replace_d_10.pkl')
with open(path, 'rb') as file:
    evaluations50000_replace_d_10 = pickle.load(file)

tests(evaluations50000_replace_d_10,3,10, ', original dataset, SF15')
```



Using the original MR_better, not edited:

```
[ ]: tests(evaluations50000_replace_d_10,6,10, ', original dataset, SF15')
```



The grid from the article.

```
[ ]: from IPython.display import display, Image
display(Image(filename='originalreplace10_am.png',width=900,height=900))
```

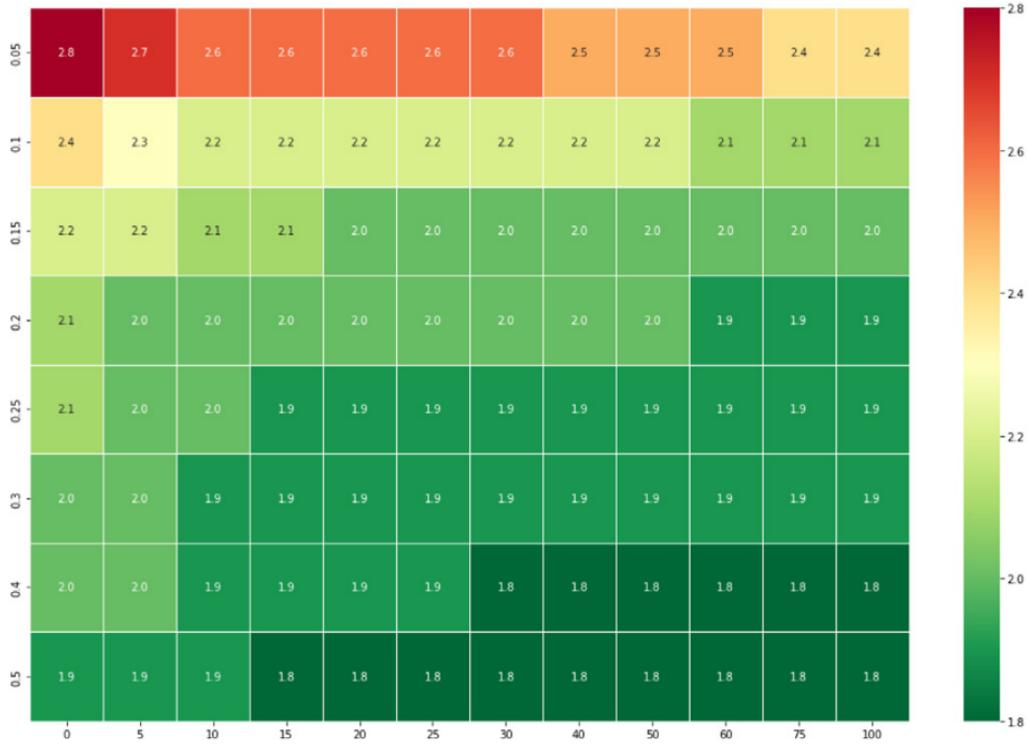


Fig. 9. Percentage of problems detected by $\text{MR}_{\text{better}}$. x -axis represents ϵ value and y -axis represents δ value.

The grid from the article is closer to the grid with the corrected $\text{MR}_{\text{better}}$ than the one with the $\text{MR}_{\text{better}}$ that was in their GitHub. Two hypothesis: they actually used a correct $\text{MR}_{\text{better}}$ and put somehow another one on GitHub, or they did not filter the positions which had no white bishops nor white rooks.

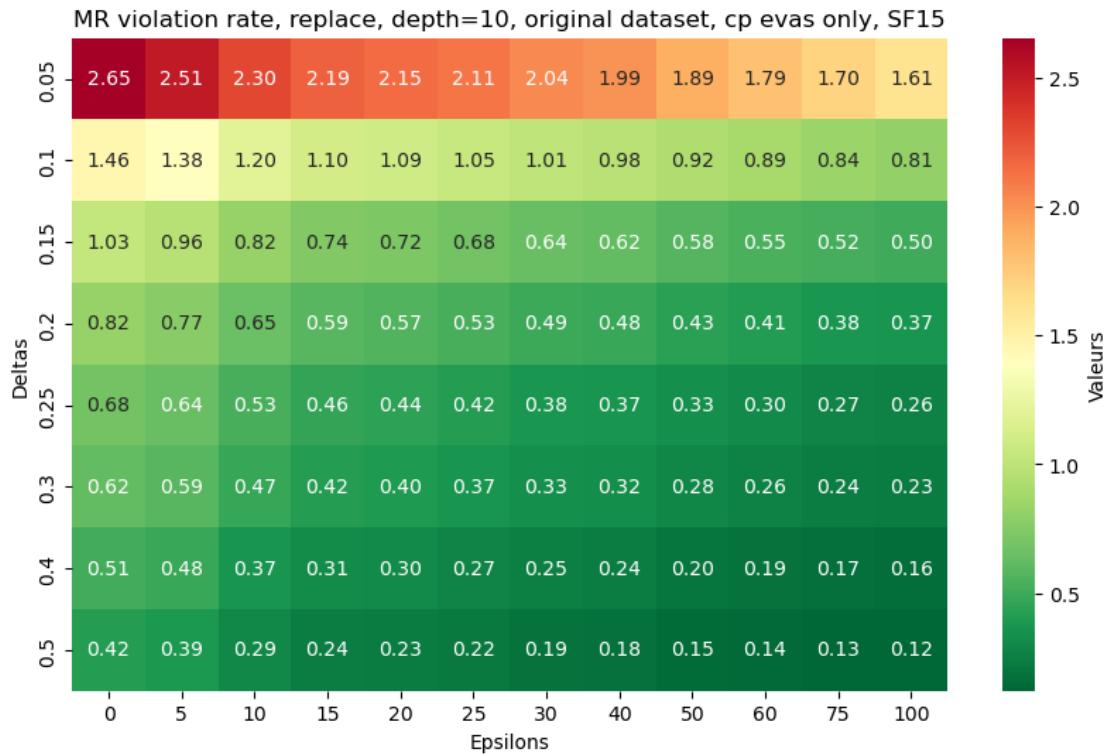
depth = 10, splitting

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_replace_d_10)[1])/  
    ↪(len(split_mate_cp(evaluations50000_replace_d_10)[0])+len(split_mate_cp(evaluations50000_re
```

Mates proportion : 0.43926964805504104

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_replace_d_10)[0], 3, 10, ', original'  
    ↪dataset, cp evas only, SF15')
```



```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_replace_d_10)[1], 3, 1, 1), '%')
```

Mates failure rate : 2.2530120481927725 %

depth = 10, real positions

```
[ ]: evaluations50000_replace_d_10_real =
    ↪real_positions(evaluations50000_replace_d_10[17000:])
evaluations50000_replace_d_10_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_replace_d_10_real, 3, 10, ', original dataset, real
    ↪positions, SF15')
```



depth = 15, entire dataset

```
[ ]: path = os.path.join('replace_d=15', 'evaluations50000_replace_d_15.pkl')
with open(path, 'rb') as file:
    evaluations50000_replace_d_15 = pickle.load(file)

tests(evaluations50000_replace_d_15, 3, 15, 'original dataset, SF15')
```

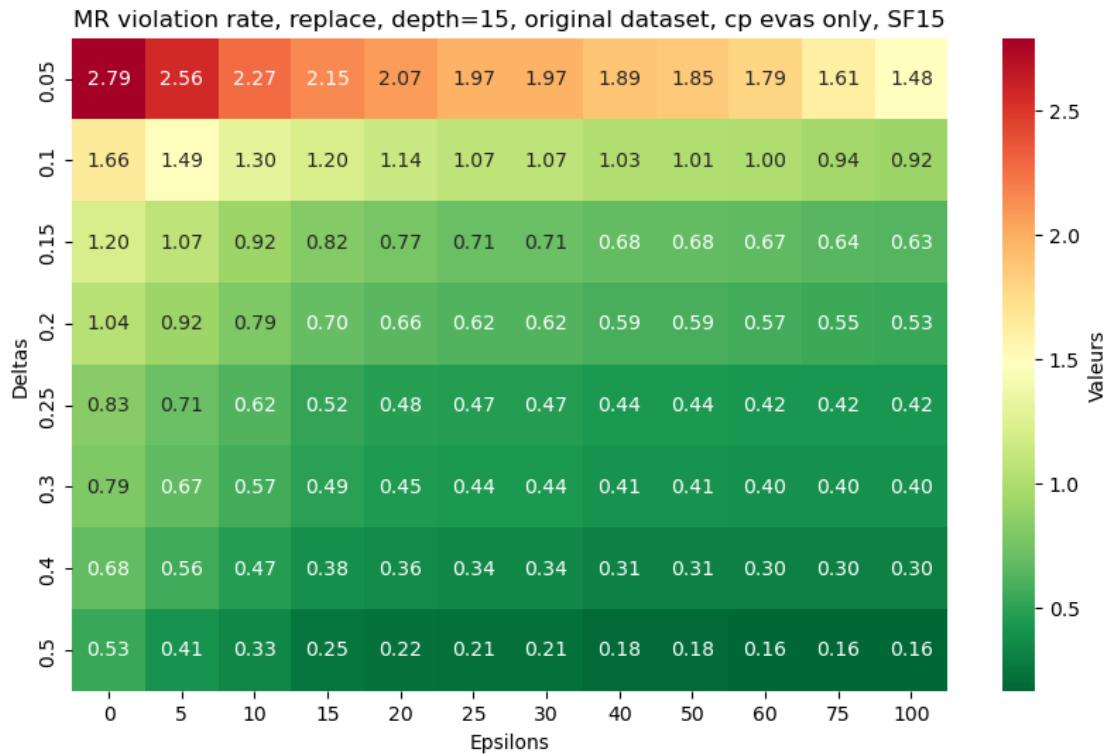


depth = 15, splitting

```
[ ]: print("Mates proportion : ", len(split_mate_cp(evaluations50000_replace_d_15)[1]) / (len(split_mate_cp(evaluations50000_replace_d_15)[0]) + len(split_mate_cp(evaluations50000_replace_d_15)[1])))
Mates proportion :  0.6127066553624417
```

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_replace_d_15)[0], 3, 15, 'original'
          , dataset, cp evas only, SF15')
```

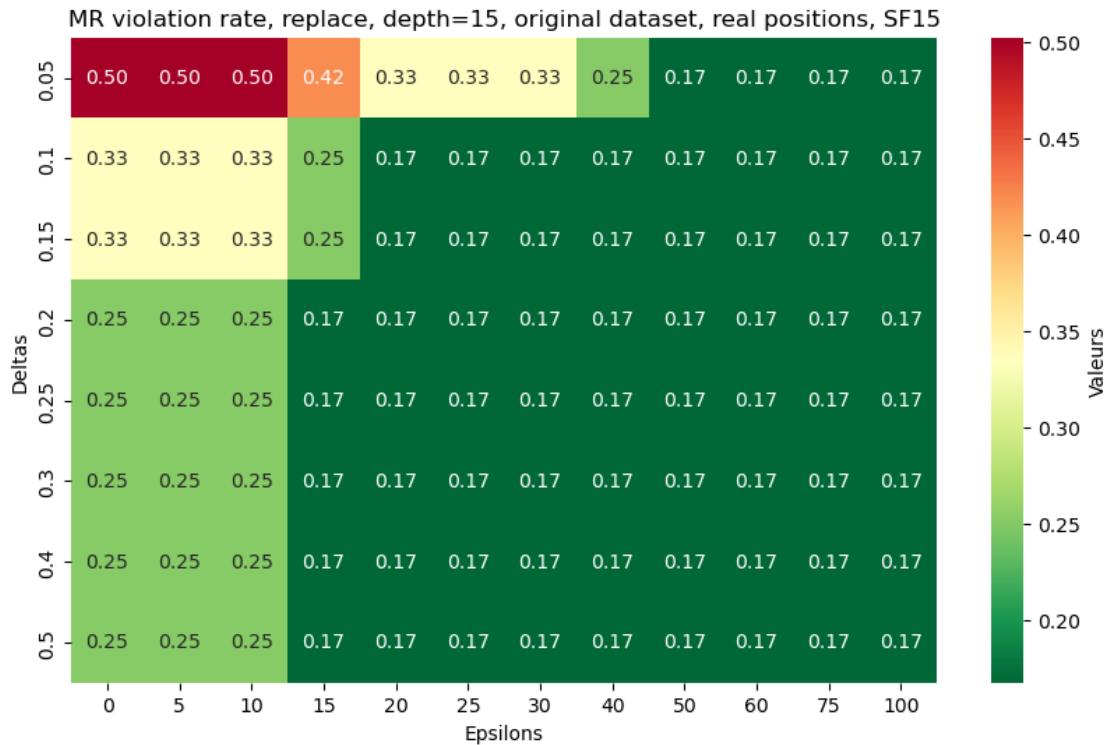


```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_replace_d_15)[1], 3, 1, 1), '%')
```

Mates failure rate : 2.084234195278043 %

depth = 15, real positions

```
[ ]: evaluations50000_replace_d_15_real =
    ↪real_positions(evaluations50000_replace_d_15[17000:])
evaluations50000_replace_d_15_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_replace_d_15_real, 3, 15, ', original dataset, real
    ↪positions, SF15')
```



depth = 20, entire dataset (Link V16)

```
[ ]: path = os.path.join('replace_d=20', 'evaluations50000_replace_d_20.pkl')
with open(path, 'rb') as file:
    evaluations50000_replace_d_20 = pickle.load(file)

tests(evaluations50000_replace_d_20, 3, 20, 'original dataset, SF15')
```



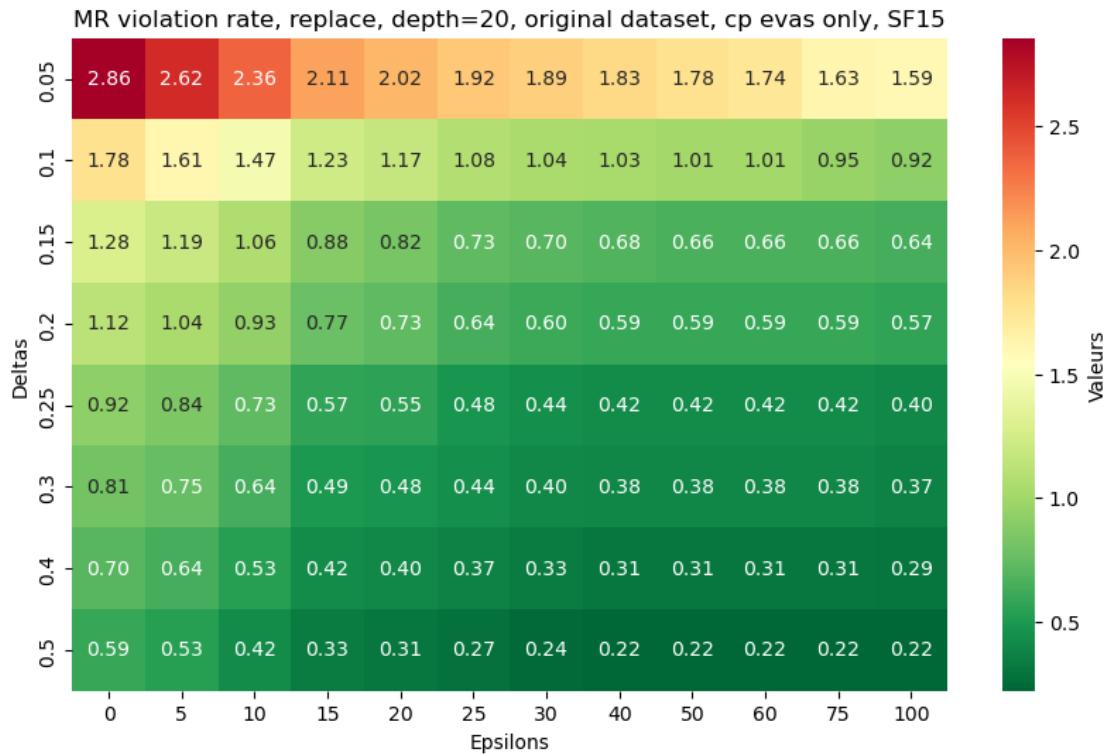
depth = 20, splitting (Link V16)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_replace_d_20)[1])/  
    ↪(len(split_mate_cp(evaluations50000_replace_d_20)[0])+len(split_mate_cp(evaluations50000_repl...))
```

Mates proportion : 0.7085891208028613

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_replace_d_20)[0],3,20, 'original'  
    ↪dataset, cp evas only, SF15')
```

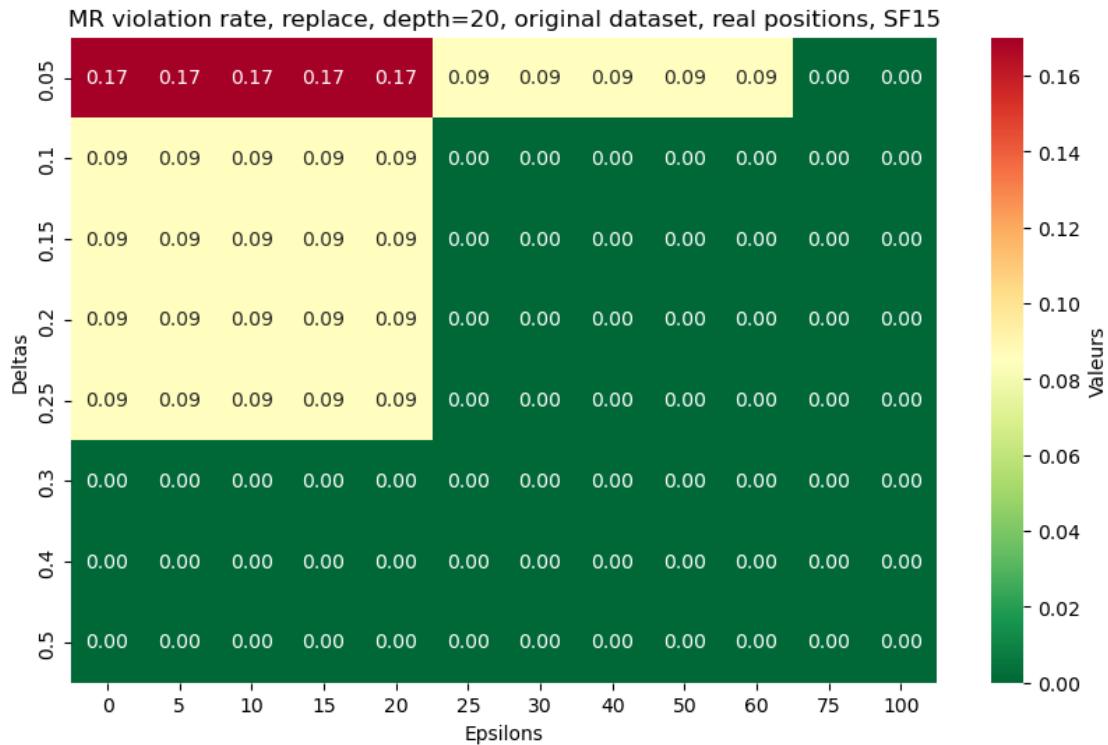


```
[ ]: print('Mates failure rate :')
    ↪',100-MR(split_mate_cp(evaluations50000_replace_d_20)[1],3,1,1),'%')
```

Mates failure rate : 1.7327105620009036 %

depth = 20, real positions (Link V16)

```
[ ]: evaluations50000_replace_d_20_real =
    ↪real_positions(evaluations50000_replace_d_20[17000:])
evaluations50000_replace_d_20_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_replace_d_20_real,3,20, 'original dataset, real'
    ↪positions, SF15')
```



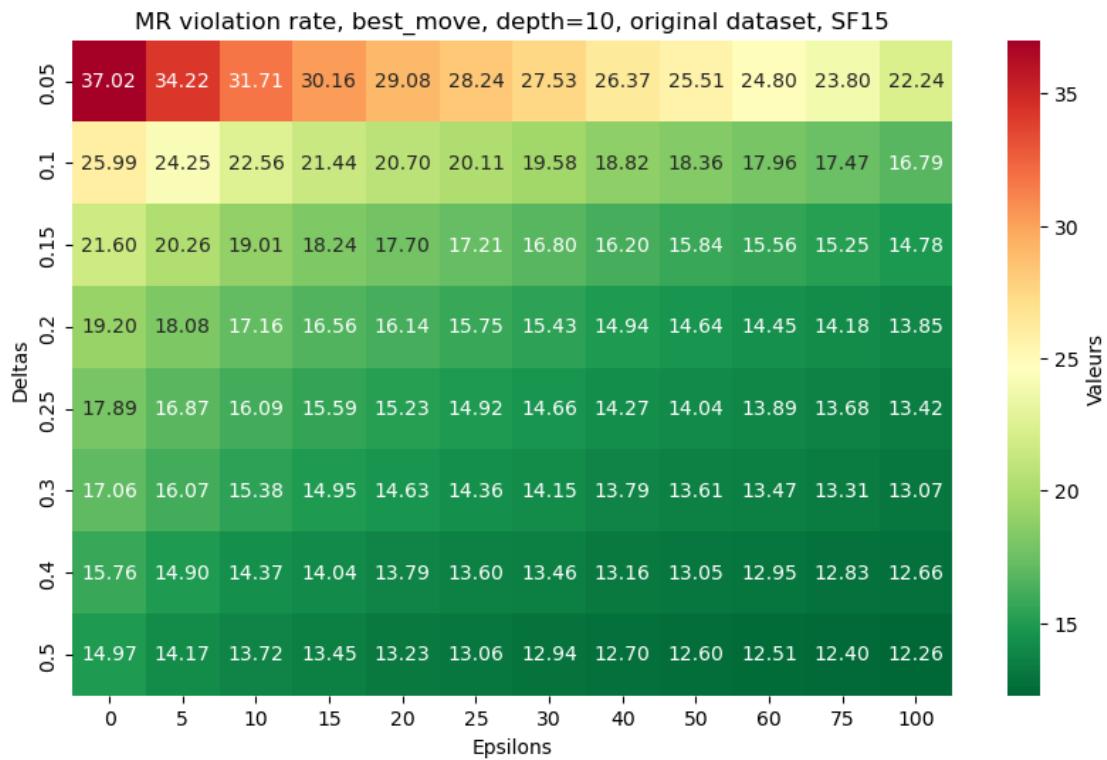
0.8.5 best_move

In this part, we are using sim_axis then taking the evaluation after the best move (according to Stockfish) is played.

depth = 10, entire dataset Our grid:

```
[ ]: path = os.path.join('best_move_d=10','evaluations50000_best_move_d_10.pkl')
with open(path, 'rb') as file:
    evaluations50000_best_move_d_10 = pickle.load(file)

tests(evaluations50000_best_move_d_10,4,10, ' , original dataset, SF15')
```



The grid from their article.

```
[ ]: from IPython.display import display, Image
display(Image(filename='originalfirst10_am.png',width=900,height=900))
```

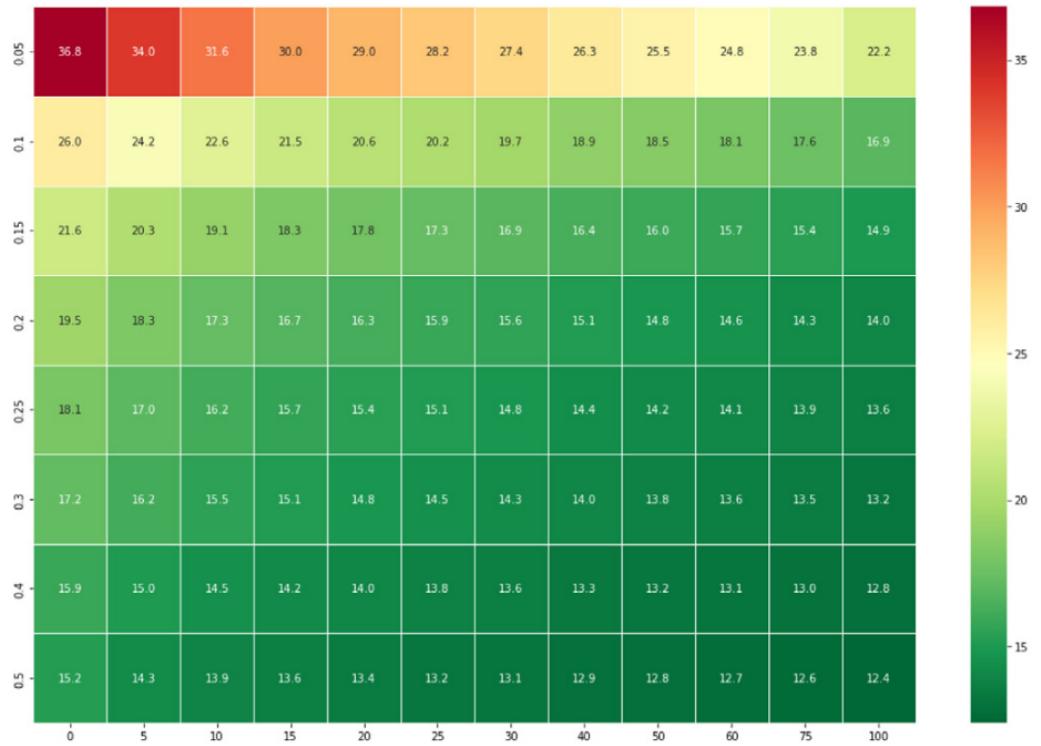


Fig. 10. Percentage of problems detected by MR_{first} using the $\text{sim_axis}(p)$ transformation. x -axis represents ϵ value and y -axis represents δ value.

depth = 10, splitting

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_best_move_d_10)[1])/  
    ↪(len(split_mate_cp(evaluations50000_best_move_d_10)[0])+len(split_mate_cp(evaluations50000_
```

Mates proportion : 0.28685196136408436

```
[ ]: tests(split_mate_cp(evaluations50000_best_move_d_10)[0],4,10, ', original  
    ↪dataset, cp evas only, SF15')
```



```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_best_move_d_10)[1], 4, 1, 1), '%')
```

Mates failure rate : 34.33205057724025 %

depth = 10, real positions

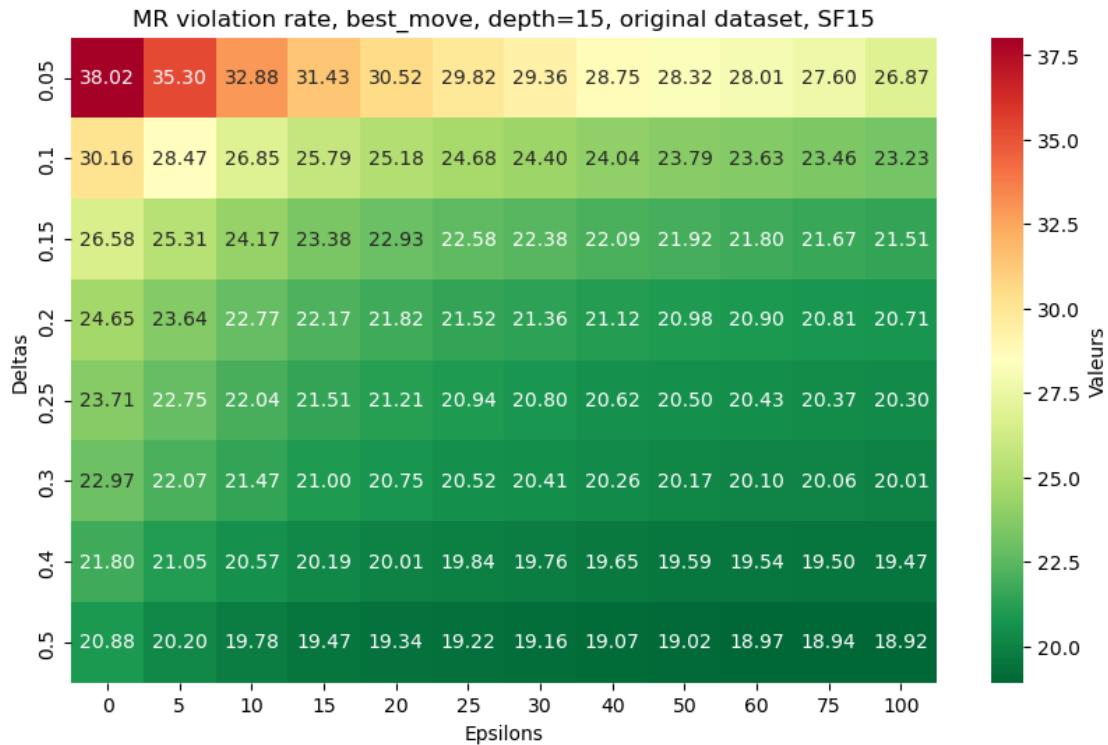
```
[ ]: evaluations50000_best_move_d_10_real =
    ↪real_positions(evaluations50000_best_move_d_10[24000:])
evaluations50000_best_move_d_10_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_best_move_d_10_real, 4, 10, ', original dataset, real
    ↪positions, SF15')
```



depth = 15, entire dataset

```
[ ]: path = os.path.join('best_move_d=15','evaluations50000_best_move_d_15.pkl')
with open(path, 'rb') as file:
    evaluations50000_best_move_d_15 = pickle.load(file)

tests(evaluations50000_best_move_d_15,4,15, ', original dataset, SF15')
```



depth = 15, splitting

```
[ ]: print("Mates proportion : ", ↴
    ↴len(split_mate_cp(evaluations50000_best_move_d_15)[1])/(
    ↴(len(split_mate_cp(evaluations50000_best_move_d_15)[0])+len(split_mate_cp(evaluations50000_...)
```

Mates proportion : 0.43591057840160863

```
[ ]: tests(split_mate_cp(evaluations50000_best_move_d_15)[0],4,15, 'original', ↴
    ↴dataset, cp evas only, SF15')
```

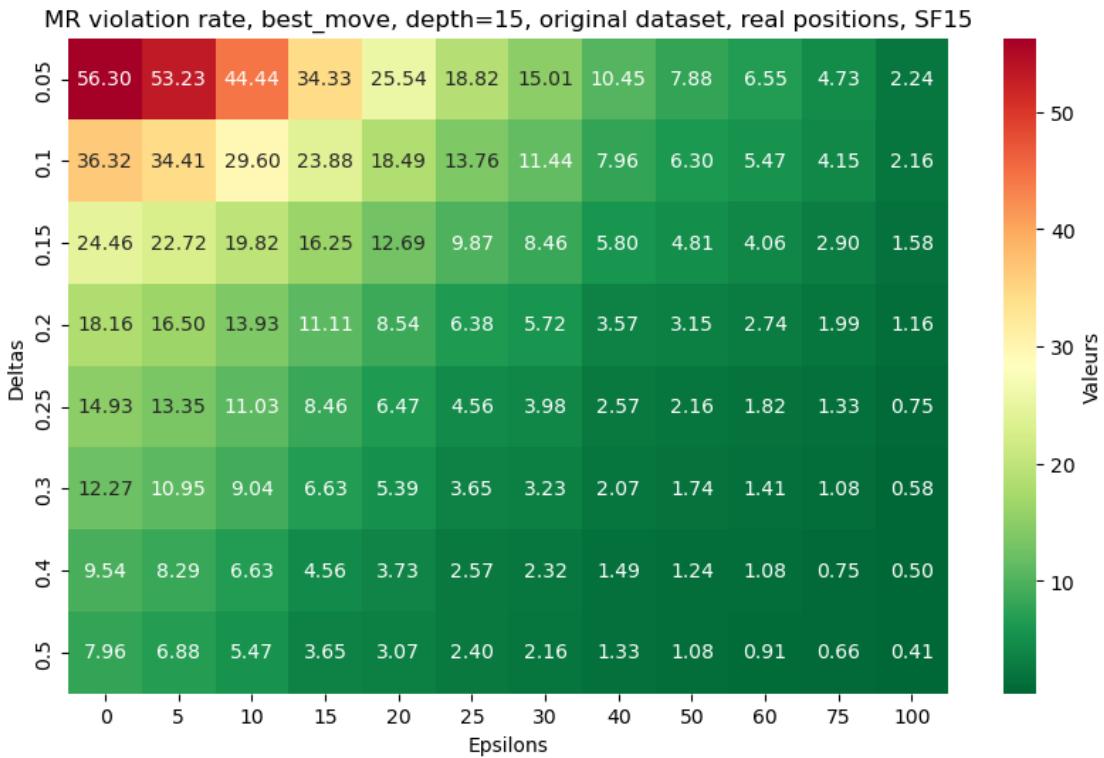


```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_best_move_d_15)[1], 4, 1, 1), '%')
```

Mates failure rate : 37.30101302460203 %

depth = 15, real positions

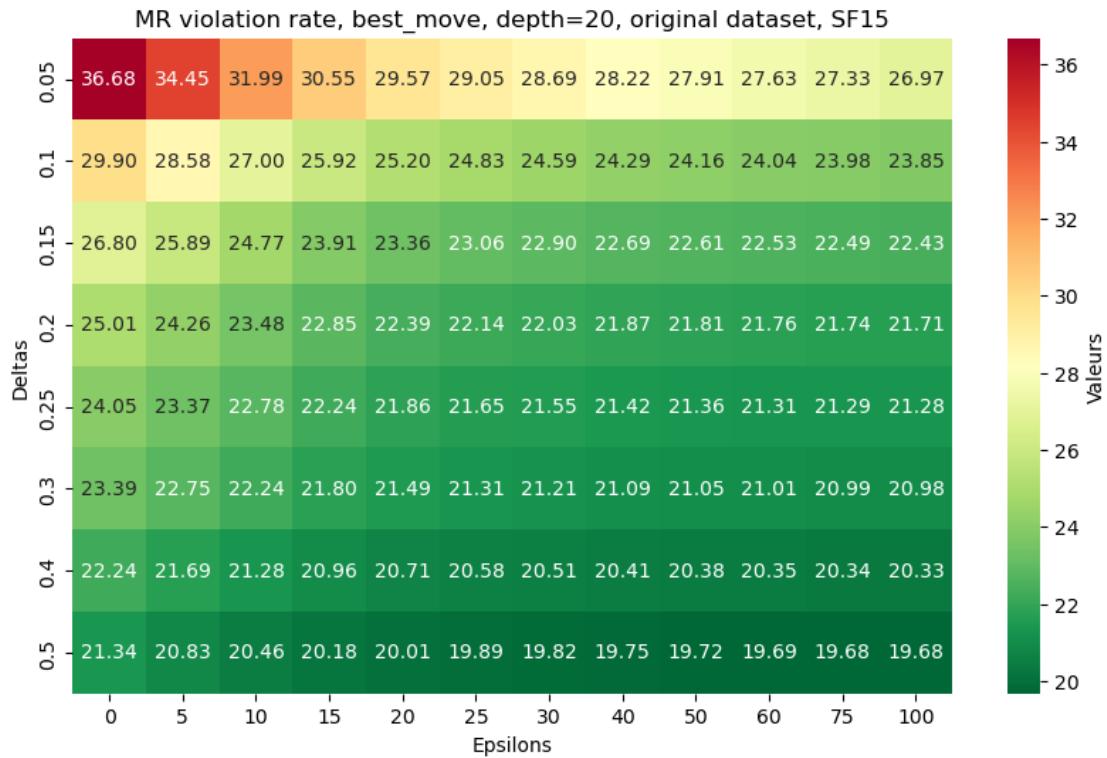
```
[ ]: evaluations50000_best_move_d_15_real =
    ↪real_positions(evaluations50000_best_move_d_15[24000:])
evaluations50000_best_move_d_15_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_best_move_d_15_real, 4, 15, ', original dataset, real
    ↪positions, SF15')
```



depth = 20, entire dataset (Link V16)

```
[ ]: path = os.path.join('best_move_d=20','evaluations50000_best_move_d_20.pkl')
with open(path, 'rb') as file:
    evaluations50000_best_move_d_20 = pickle.load(file)

tests(evaluations50000_best_move_d_20,4,20, ', original dataset, SF15')
```

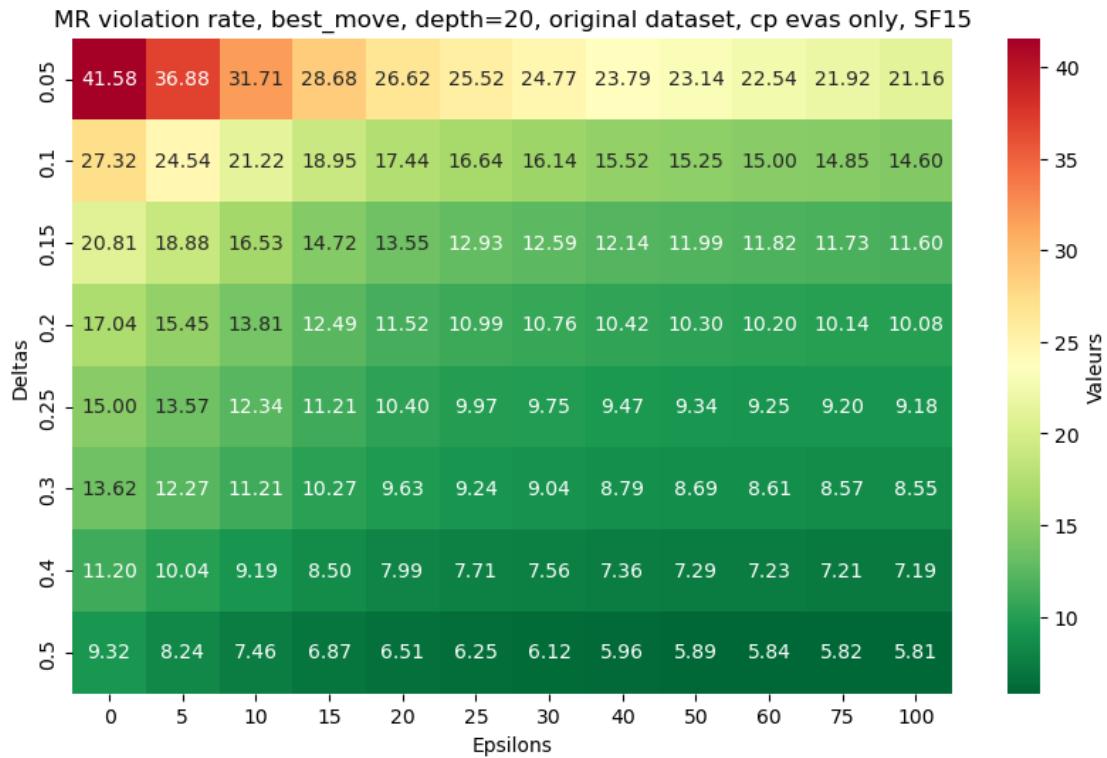


depth = 20, splitting (Link V16)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_best_move_d_20)[1])/  
    ↪(len(split_mate_cp(evaluations50000_best_move_d_20)[0])+len(split_mate_cp(evaluations50000_...))
```

Mates proportion : 0.5246750167924453

```
[ ]: tests(split_mate_cp(evaluations50000_best_move_d_20)[0],4,20, ' original  
    ↪dataset, cp evas only, SF15')
```

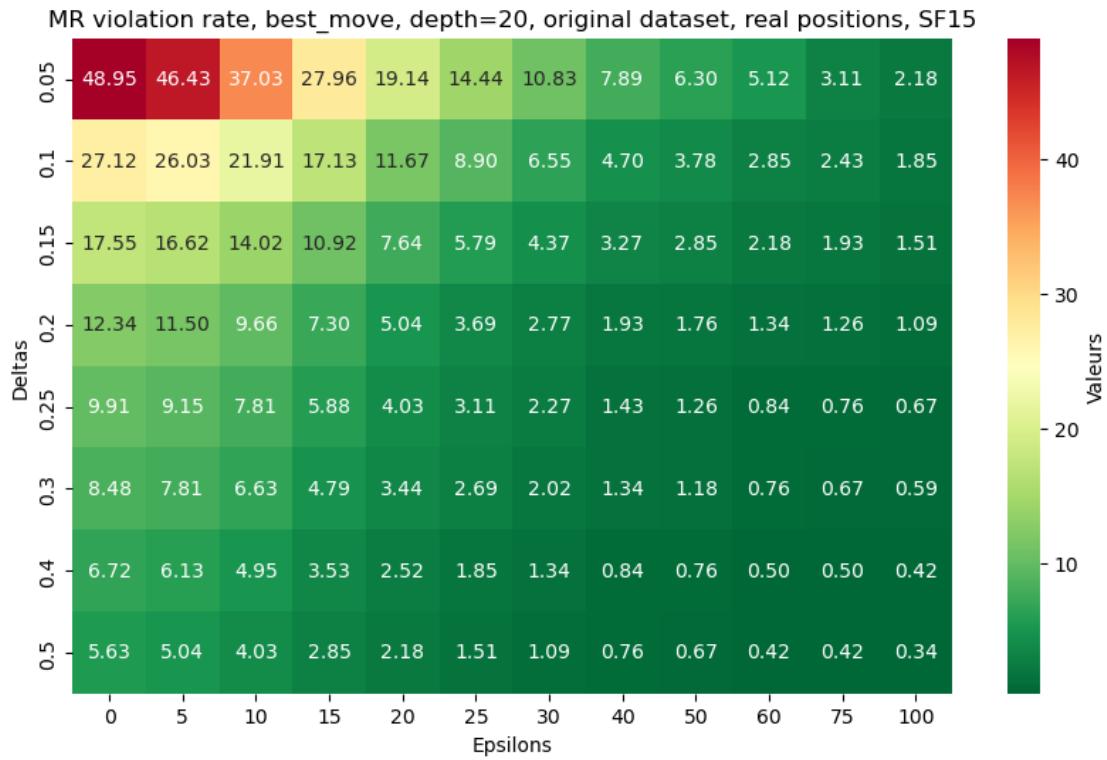


```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_best_move_d_20)[1], 4, 1, 1), '%')
```

Mates failure rate : 32.23887340914226 %

depth = 20, real positions (Link V16)

```
[ ]: evaluations50000_best_move_d_20_real =
    ↪real_positions(evaluations50000_best_move_d_20[22000:])
evaluations50000_best_move_d_20_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_best_move_d_20_real, 4, 20, ', original dataset, real
    ↪positions, SF15!')
```



0.8.6 Analysis on biggest gaps

Additional function Here is a function to plot evaluations by depth, until the MR is verified and the minimum depth (min_d) is reached. Will stop at max_d if can't verify MR before.

```
[ ]: def plotevas(data1,data2,val,type):
    print('MR verified at depth = ', val)
    ind1 = 0
    ind2 = 0
    cp1 = []
    cp2 = []
    mate1 = []
    mate2 = []
    values1cp = []
    values1mate = []
    values2cp = []
    values2mate = []
    if type == 4:
        for data in data1:
            ind1+=1
            if data.get('Mate') == None:
                cp1.append(ind1)
                values1cp.append(data.get('Centipawn'))
```

```

        else:
            mate1.append(ind1)
            values1mate.append(data.get('Mate'))

    for data in data2:
        ind2+=1
        if data.get('Mate') == None:
            cp2.append(ind2)
            values2cp.append(data.get('Centipawn'))
        else:
            mate2.append(ind2)
            values2mate.append(data.get('Mate'))

    else:
        for data in data1:
            ind1+=1
            if data.get('type') == 'cp':
                cp1.append(ind1)
                values1cp.append(data.get('value'))
            else:
                values1mate.append(data.get('value'))
                mate1.append(ind1)

    for data in data2:
        ind2+=1
        if type == 0:
            if data.get('type') == 'cp':
                cp2.append(ind2)
                values2cp.append(data.get('value') * (-1))
            else:
                mate2.append(ind2)
                values2mate.append(data.get('value') * (-1))

        else:
            if data.get('type') == 'cp':
                cp2.append(ind2)
                values2cp.append(data.get('value'))
            else:
                mate2.append(ind2)
                values2mate.append(data.get('value'))


plt.plot(cp1, values1cp, '--', color='#17d1d3')
plt.plot(cp2, values2cp, '--', color='#405ee3')

plt.grid(axis='y', color='#D3D3D3', linestyle='--', linewidth=0.5)

plt.ylabel('Centipawn advantage', color='blue')

```

```

plt.tick_params('y', colors='blue')

plt.twinx()

plt.plot(mate1, values1mate, '-.', color='#f7766d')
plt.plot(mate2, values2mate, '--', color='#df2c1f')

plt.ylabel('Mate in', color='red')
plt.tick_params('y', colors='red')

plt.show()

```

We are using delta = 0.25 and epsilon = 25 for these tests.

Stockfish returning opposite evaluations

`sim_mirror, sim_axis` No evidences of really large gaps.

REMINDER : with `sim_mirror`, since the mutation swap colors, the evaluation of the original position has to match with (-1)*the mutated evaluation.

```
[ ]: p8mirror = 'r1b2rk1/p3n1pp/1pn1p3/q1ppPp2/3P4/P1PB1NQ1/2PB1PPP/R3K2R w'

show_pos([p8mirror],300)
print(evaluationdouble(p8mirror,sim_mirror(p8mirror),50000,20))
print(evaluationdouble(p8mirror,sim_axis(p8mirror),50000,20))
```

```
<IPython.core.display.HTML object>
({'type': 'cp', 'value': -32}, {'type': 'cp', 'value': -63})
({'type': 'cp', 'value': -32}, {'type': 'cp', 'value': 40})
```

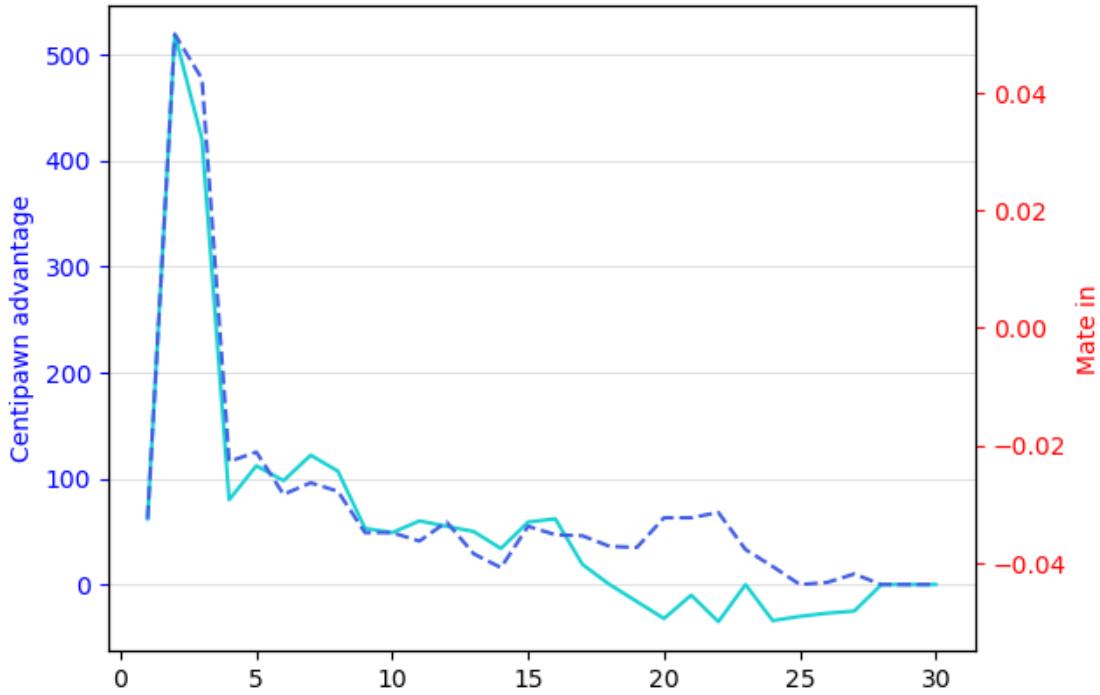
```
[ ]: chemin = os.path.join('plotivas', 'p8mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot8mirror = pickle.load(fichier)

plotivas(plot8mirror[0],plot8mirror[1],plot8mirror[2], 0)

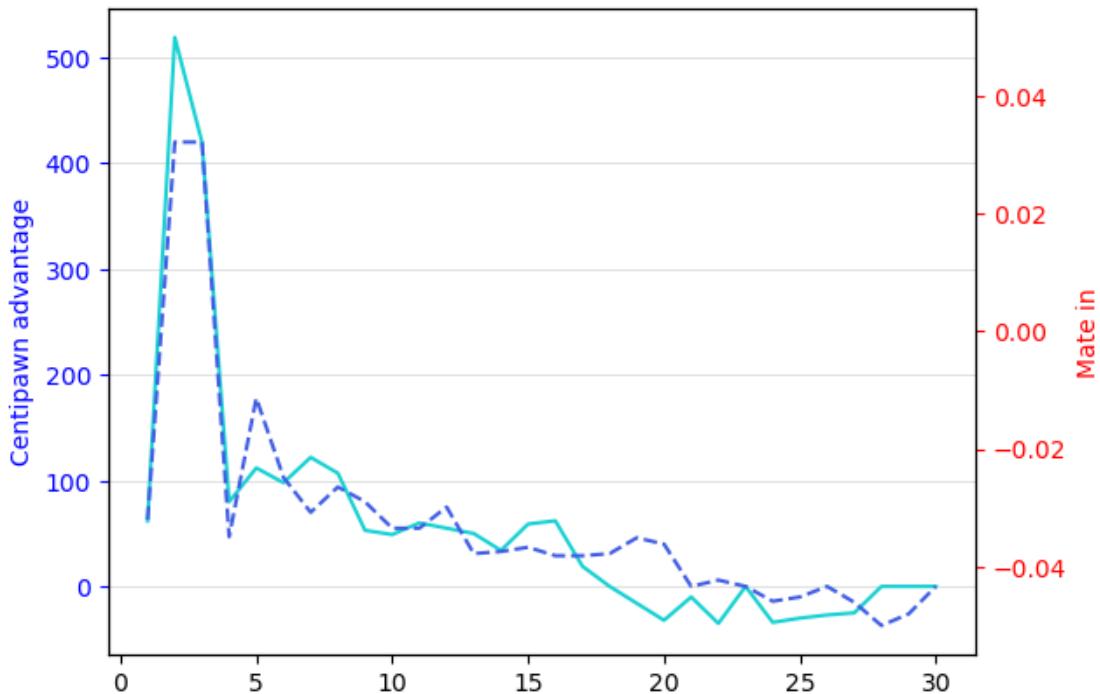
chemin = os.path.join('plotivas', 'p8axis.pkl')
with open(chemin, 'rb') as fichier:
    plot8axis = pickle.load(fichier)

plotivas(plot8axis[0],plot8axis[1],plot8axis[2], 1)
```

MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 28, 29, 30]



MR verified at depth = [2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 17, 21, 23, 24, 25, 27, 30]



```

best_move
[ ]: p1best = 'r2q1rk1/4b3ppp/3pb3/2n1pP2/1p2P1PP/1P3Q2/1BP1N1B1/2KR3R w'
      show_pos([p1best],300)
      print(evaluation(p1best,50000,20)[1],evaluation(sim_axis(p1best),50000,20)[1])

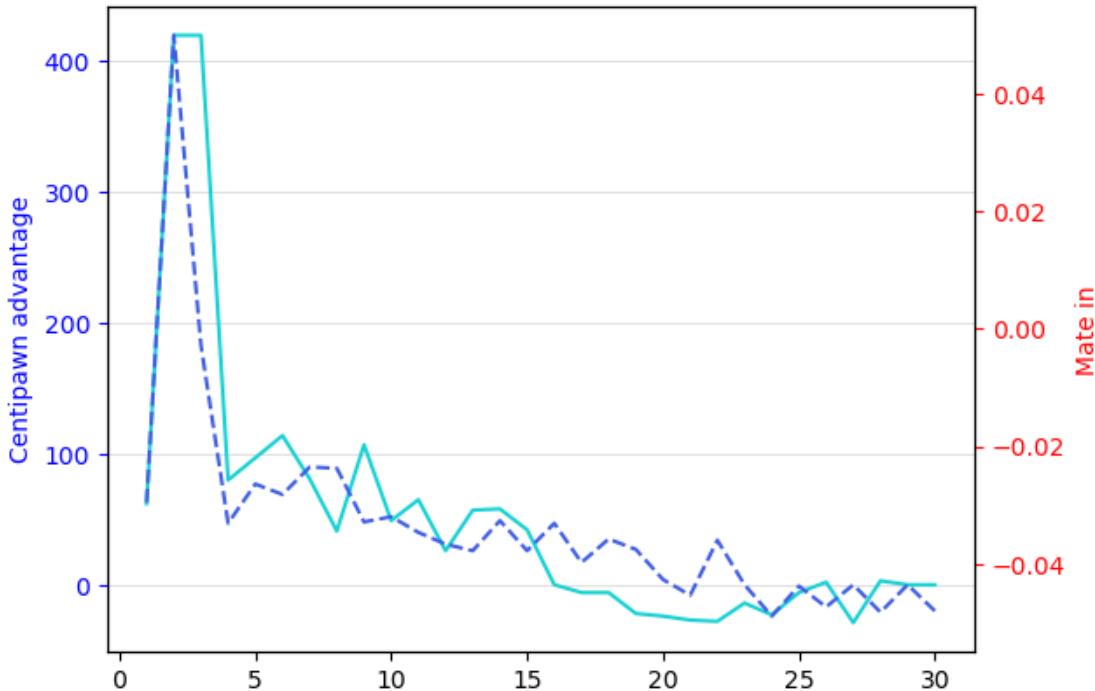
<IPython.core.display.HTML object>
[{'Move': 'f5e6', 'Centipawn': 74, 'Mate': None}, {'Move': 'c5d6', 'Centipawn': -12, 'Mate': None}]

[ ]: chemin = os.path.join('plotevas', 'p1best.pkl')
      with open(chemin, 'rb') as fichier:
          plot1best = pickle.load(fichier)

      plotevas(plot1best[0],plot1best[1],plot1best[2], 4)

```

MR verified at depth = [2, 5, 6, 7, 10, 11, 12, 14, 15, 17, 21, 23, 24, 25, 26, 28, 29, 30]



Stockfish having troubles with draw positions at depth = 20

sim_mirror

```
[ ]: p1mirror = '8/8/7k/8/4K3/4r3/2B4P/8 w'
      p2mirror = '7b/8/kq6/8/8/1N2R3/K2P4/8 w'

      show_pos([p1mirror,p2mirror],300)
      print(evaluationondouble(p1mirror,sim_mirror(p1mirror),50000,20))
      print(evaluationondouble(p2mirror,sim_mirror(p2mirror),50000,20))

<IPython.core.display.HTML object>

({'type': 'cp', 'value': 0}, {'type': 'cp', 'value': -402})
({'type': 'cp', 'value': -478}, {'type': 'cp', 'value': 0})

It was in fact verified with a lower depth, but it is verified again with an higher depth. For the first one, it also gets verified at depth 23 with false values.

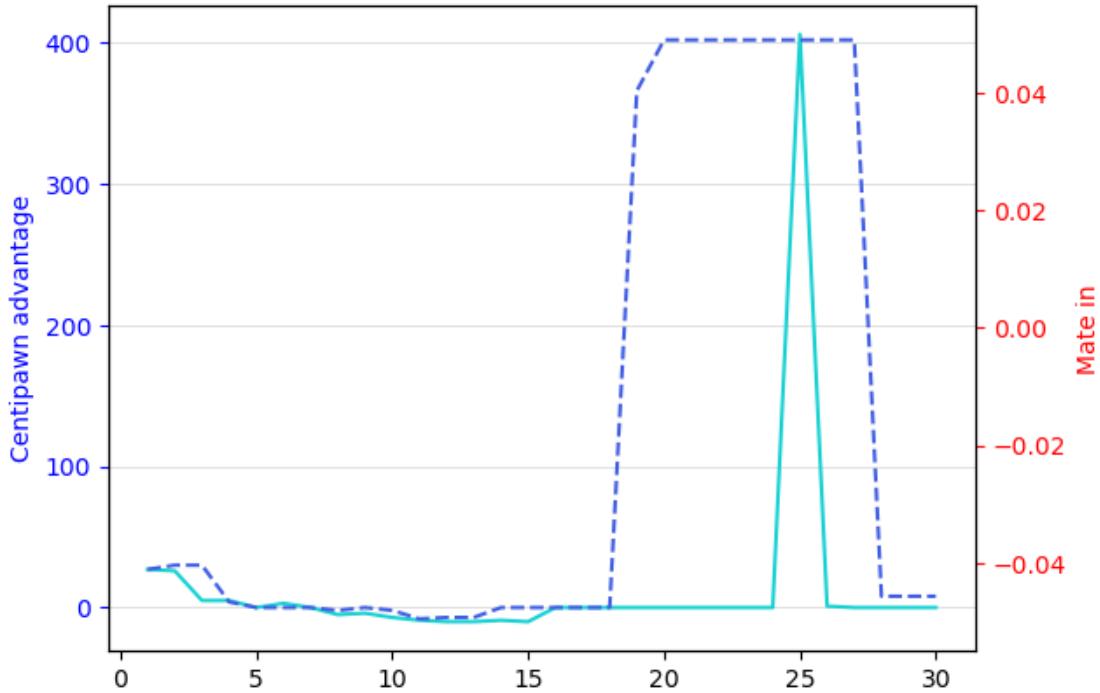
[ ]: chemin = os.path.join('plotevas', 'p1mirror.pkl')
      with open(chemin, 'rb') as fichier:
          plot1mirror = pickle.load(fichier)

          plotevas(plot1mirror[0],plot1mirror[1],plot1mirror[2], 0)

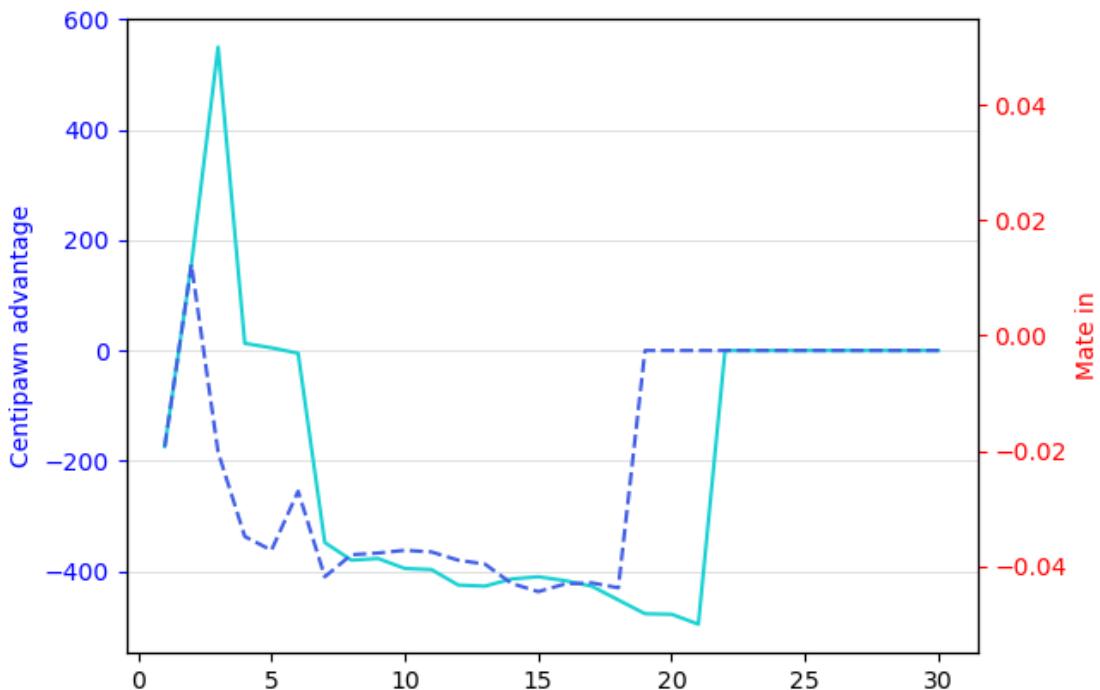
      chemin = os.path.join('plotevas', 'p2mirror.pkl')
      with open(chemin, 'rb') as fichier:
          plot2mirror = pickle.load(fichier)

          plotevas(plot2mirror[0],plot2mirror[1],plot2mirror[2], 0)

MR verified at depth = [2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 25, 28, 29, 30]
```



MR verified at depth = [2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 23, 24, 25, 26, 27, 28, 29, 30]



```

sim_axis, sim_diag
[ ]: print(evaluationdouble(p1mirror,sim_axis(p1mirror),50000,20))
print(evaluationdouble(p2mirror,sim_axis(p2mirror),50000,20))

print(evaluationdouble(p1mirror,sim_diag(p1mirror),50000,20))
print(evaluationdouble(p2mirror,sim_diag(p2mirror),50000,20))

print(evaluation(p1mirror,50000,20)[1],evaluation(sim_axis(p1mirror),50000,20)[1])
print(evaluation(p2mirror,50000,20)[1],evaluation(sim_axis(p2mirror),50000,20)[1])

({'type': 'cp', 'value': 0}, {'type': 'cp', 'value': 0})
({'type': 'cp', 'value': -478}, {'type': 'cp', 'value': -477})
({'type': 'cp', 'value': 0}, {'type': 'cp', 'value': 0})
({'type': 'cp', 'value': -478}, {'type': 'cp', 'value': -698})
[{'Move': 'e4e3', 'Centipawn': 0, 'Mate': None}][{'Move': 'd4d3', 'Centipawn': 0, 'Mate': None}]
[{'Move': 'a2b1', 'Centipawn': -501, 'Mate': None}][{'Move': 'h2g1',
'Centipawn': -498, 'Mate': None}]

```

Stockfish doesn't fail on the two previous position in sim_axis. Couldn't find other examples.

Stockfish showing gaps in evaluation on non-draw positions at depth = 20

```

sim_mirror
[ ]: p3mirror = '8/K7/3N4/2PR2r1/n2k4/3p3N/4Pn2/6B1 b'
p4mirror = '8/2p1k3/3p3p/2PP1pp1/1P1K1P2/6P1/8/8 w'
show_pos([p3mirror,p4mirror],300)

print(evaluationdouble(p3mirror,sim_mirror(p3mirror),50000,20))
print(evaluationdouble(p4mirror,sim_mirror(p4mirror),50000,20))

```

```

<IPython.core.display.HTML object>
({'type': 'cp', 'value': -222}, {'type': 'cp', 'value': 2})
({'type': 'cp', 'value': -1}, {'type': 'cp', 'value': -277})

[ ]: chemin = os.path.join('plotevas', 'p3mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot3mirror = pickle.load(fichier)

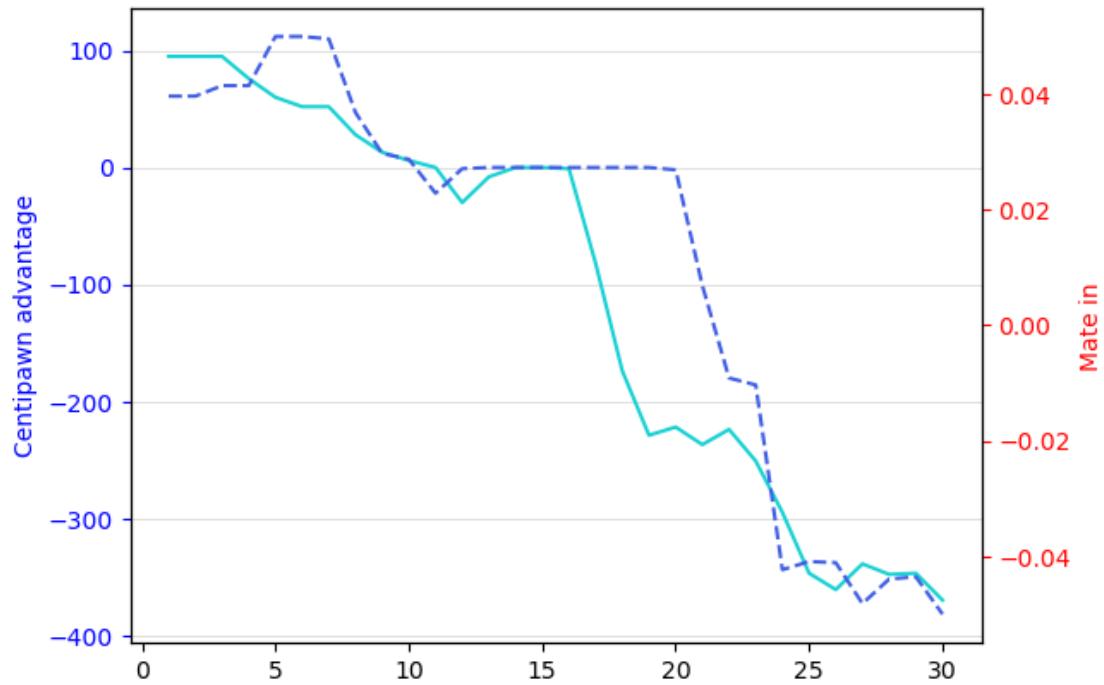
plotevas(plot3mirror[0],plot3mirror[1],plot3mirror[2], 0)

chemin = os.path.join('plotevas', 'p4mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot4mirror = pickle.load(fichier)

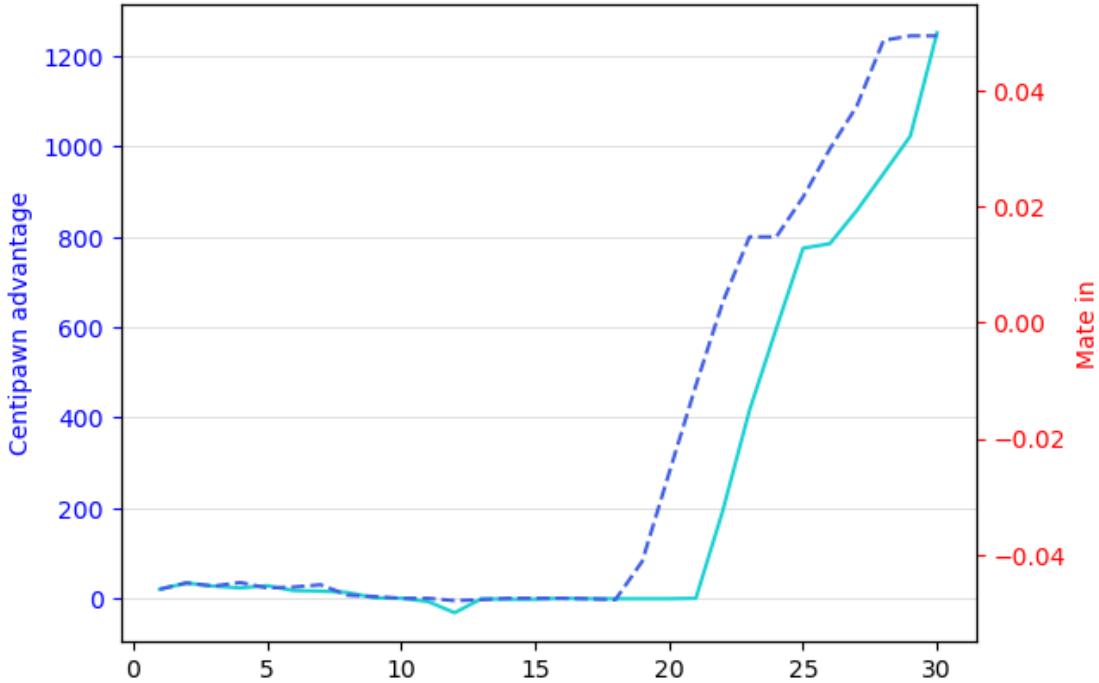
plotevas(plot4mirror[0],plot4mirror[1],plot3mirror[2], 0)

```

MR verified at depth = [2, 3, 4, 8, 9, 10, 11, 13, 14, 15, 16, 22, 23, 24, 25, 26, 27, 28, 29, 30]



MR verified at depth = [2, 3, 4, 8, 9, 10, 11, 13, 14, 15, 16, 22, 23, 24, 25, 26, 27, 28, 29, 30]



Stockfish shows a similar behaviour with the second position when using sim_axis.

sim_axis

```
[ ]: p3axis = '1K6/3R2kr/8/8/8/BQ5r/2b1r3/5n2 b'
      p2axis = 'rn1qr1k1/1p2bppp/p3p3/3pP3/P2P1B2/2RB1Q1P/1P3PP1/R5K1 w'
      show_pos([p3axis,p2axis],300)

      print(evaluationdouble(p3axis,sim_axis(p3axis),50000,20))
      print(evaluationdouble(p2axis,sim_axis(p2axis),50000,20))
```

<IPython.core.display.HTML object>

```
({'type': 'cp', 'value': 286}, {'type': 'cp', 'value': 48})
({'type': 'cp', 'value': 129}, {'type': 'cp', 'value': 344})
```

```
[ ]: chemin = os.path.join('plotevas', 'p3axis.pkl')
```

```
with open(chemin, 'rb') as fichier:
    plot3axis = pickle.load(fichier)
```

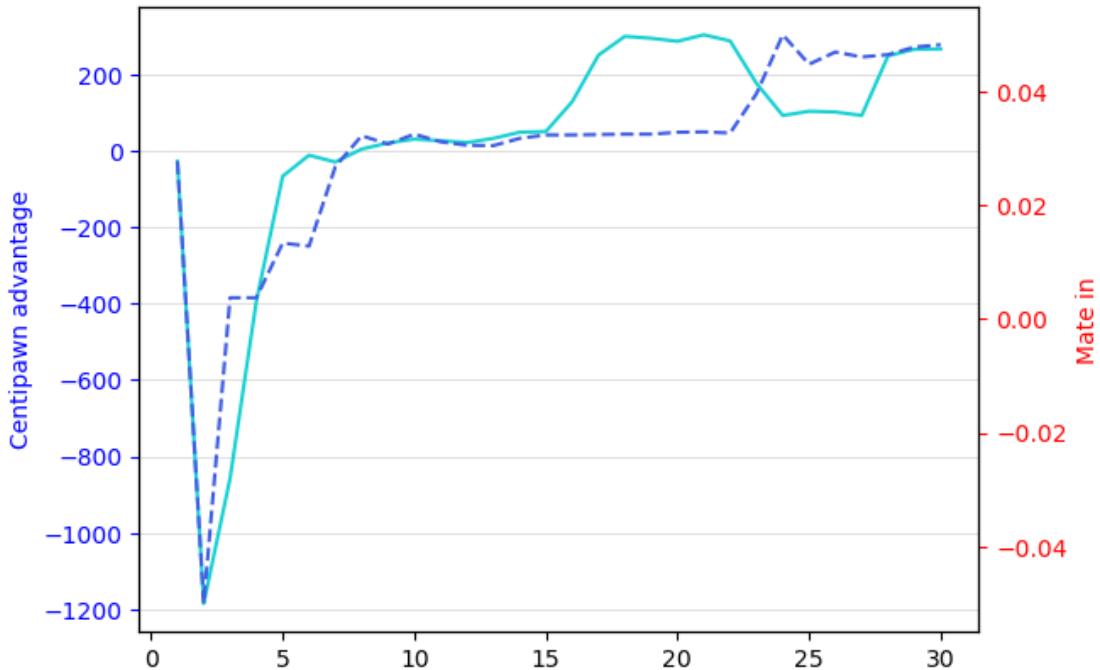
```
plotevas(plot3axis[0],plot3axis[1],plot3axis[2], 1)
```

```
chemin = os.path.join('plotevas', 'p2axis.pkl')
```

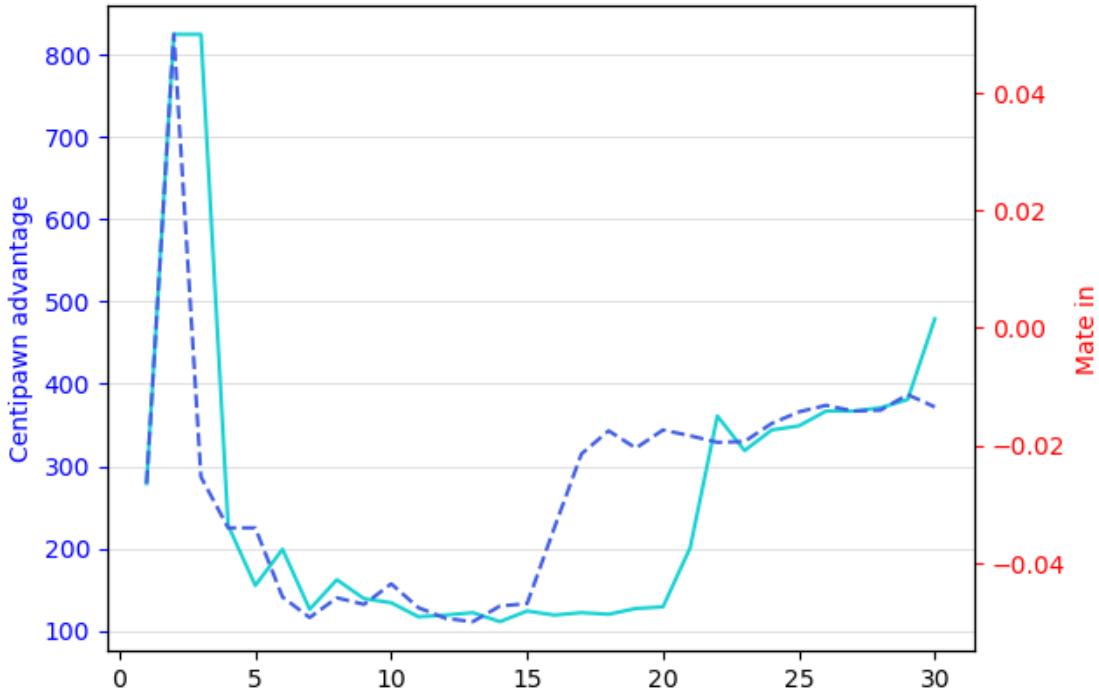
```
with open(chemin, 'rb') as fichier:
    plot2axis = pickle.load(fichier)
```

```
plot2evas(plot2axis[0],plot2axis[1],plot2axis[2], 1)
```

MR verified at depth = [2, 4, 7, 9, 10, 11, 12, 13, 14, 15, 23, 28, 29, 30]



MR verified at depth = [2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 22, 23, 24, 25, 26, 27, 28, 29, 30]



```
sim_diag
```

```
[ ]: p3diag = 'N7/8/1n6/3B4/4r3/4k3/2K5/6r1 w '
      p2diag = '8/8/8/n3N3/2Q1r3/5k2/1K6/8 b'
      show_pos([p3diag,p2diag],300)

      print(evaluationdouble(p3diag,sim_diag(p3diag),50000,20))
      print(evaluationdouble(p2diag,sim_diag(p2diag),50000,20))
```

```
<IPython.core.display.HTML object>
```

```
({'type': 'cp', 'value': -89}, {'type': 'cp', 'value': -246}
({'type': 'cp', 'value': 360}, {'type': 'cp', 'value': 145})
```

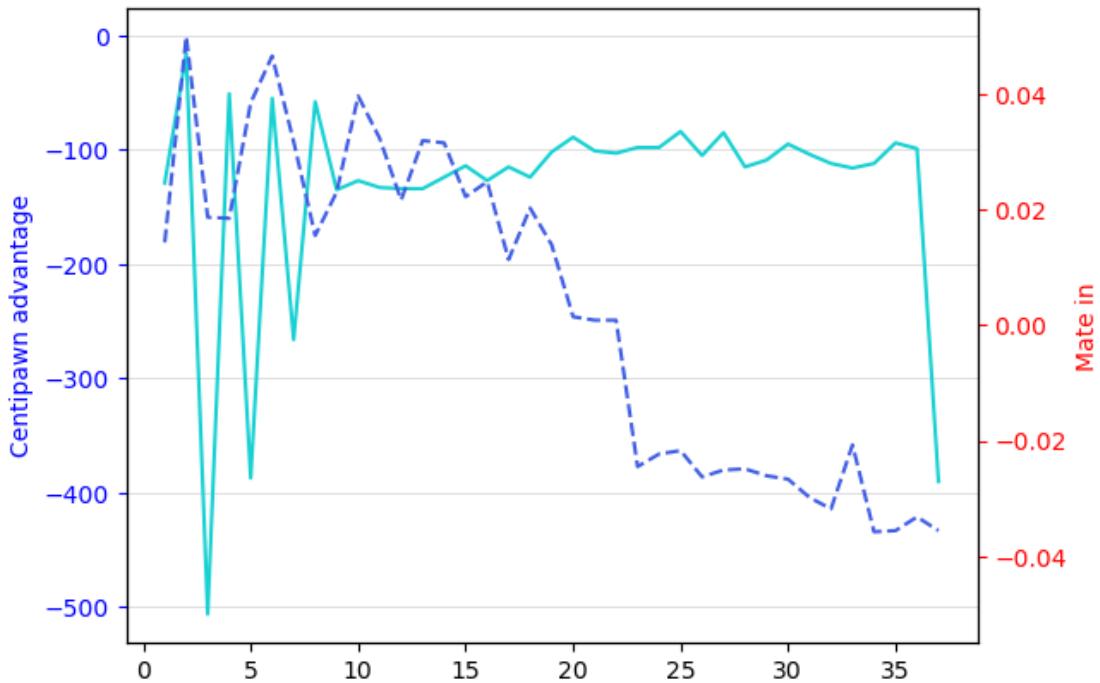
```
[ ]: chemin = os.path.join('plotevas', 'p3diag.pkl')
      with open(chemin, 'rb') as fichier:
          plot3diag = pickle.load(fichier)
```

```
plotevas(plot3diag[0],plot3diag[1],plot3diag[2], 2)
```

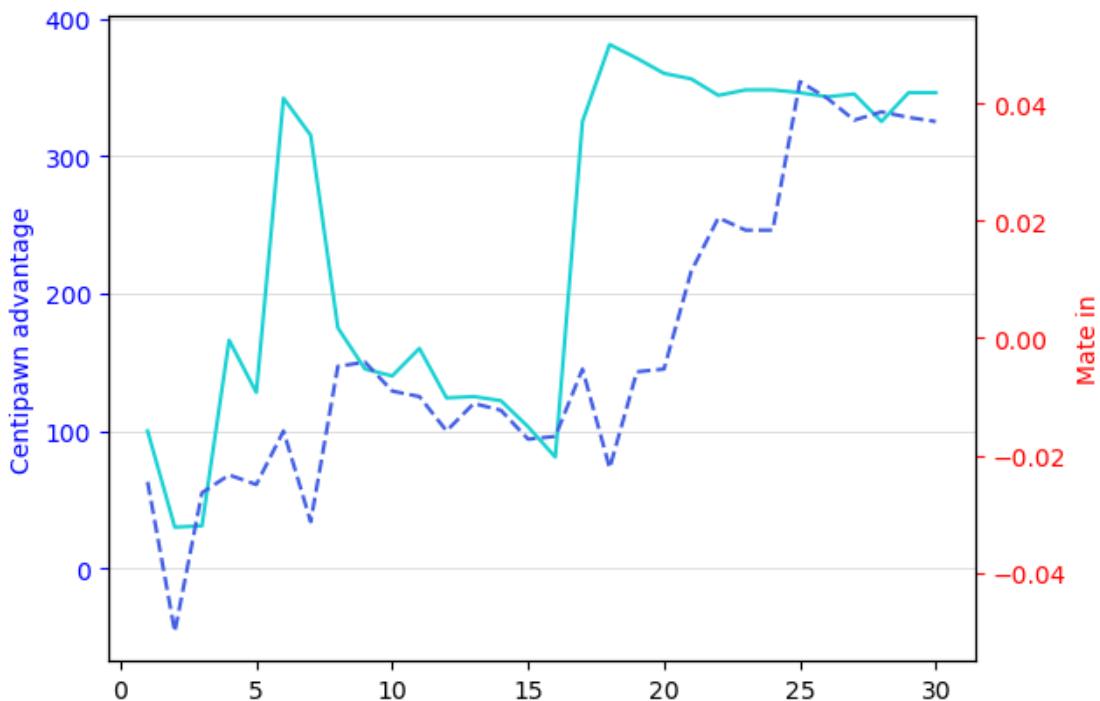
```
chemin = os.path.join('plotevas', 'p2diag.pkl')
      with open(chemin, 'rb') as fichier:
          plot2diag = pickle.load(fichier)
```

```
plotevas(plot2diag[0],plot2diag[1],plot2diag[2], 2)
```

MR verified at depth = [2, 9, 11, 12, 13, 14, 15, 16, 18, 37]



MR verified at depth = [3, 8, 9, 10, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]



```

best_move
[ ]: p2best = '8/1BK5/8/1pk5/8/8/5P2/8 b'
      p3best = '4K3/1P6/4Rqk1/6B1/4N2N/2q3r1/b1np4/8 b'
      show_pos([p2best,p3best],300)

      print(evaluation(p2best,50000,20)[1],evaluation(sim_axis(p2best),50000,20)[1])
      print(evaluation(p3best,50000,20)[1],evaluation(sim_axis(p3best),50000,20)[1])

<IPython.core.display.HTML object>

[['Move': 'b5b4', 'Centipawn': 0, 'Mate': None}, {'Move': 'g5g4', 'Centipawn': 731, 'Mate': None}], [{"Move": "g6h5", "Centipawn": -300, "Mate": None}], [{"Move": "b6a5", "Centipawn": -43, "Mate": None}]

[ ]: chemin = os.path.join('plotevas', 'p2best.pkl')
      with open(chemin, 'rb') as fichier:
          plot2best = pickle.load(fichier)

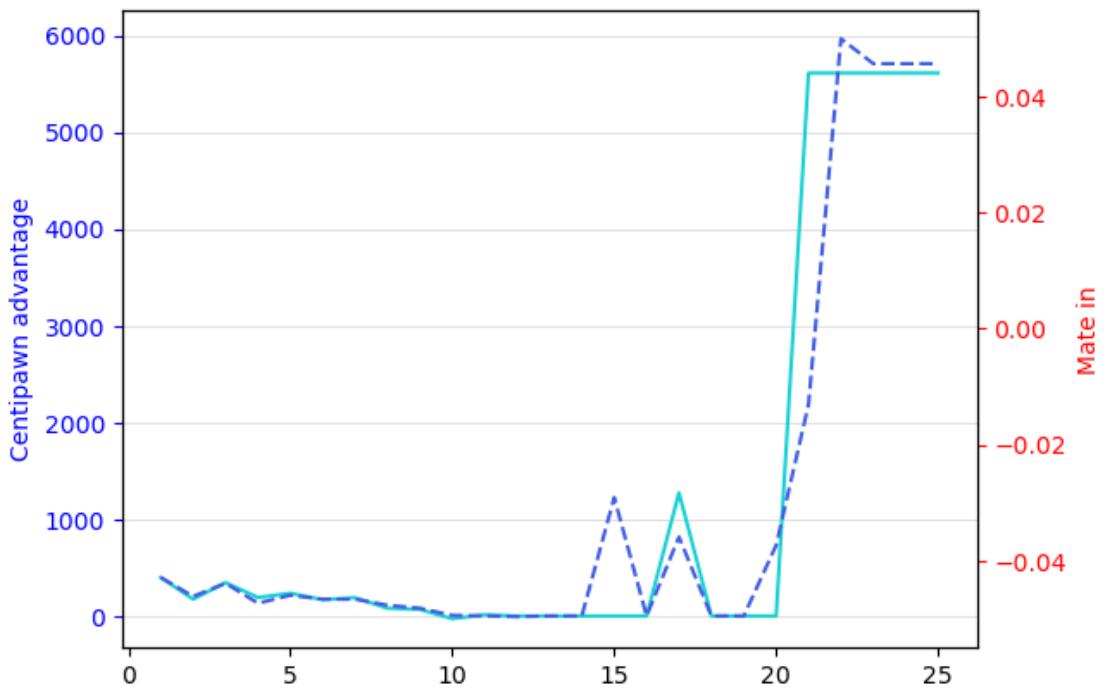
      plotevas(plot2best[0],plot2best[1],plot2best[2], 4)

      chemin = os.path.join('plotevas', 'p3best.pkl')
      with open(chemin, 'rb') as fichier:
          plot3best = pickle.load(fichier)

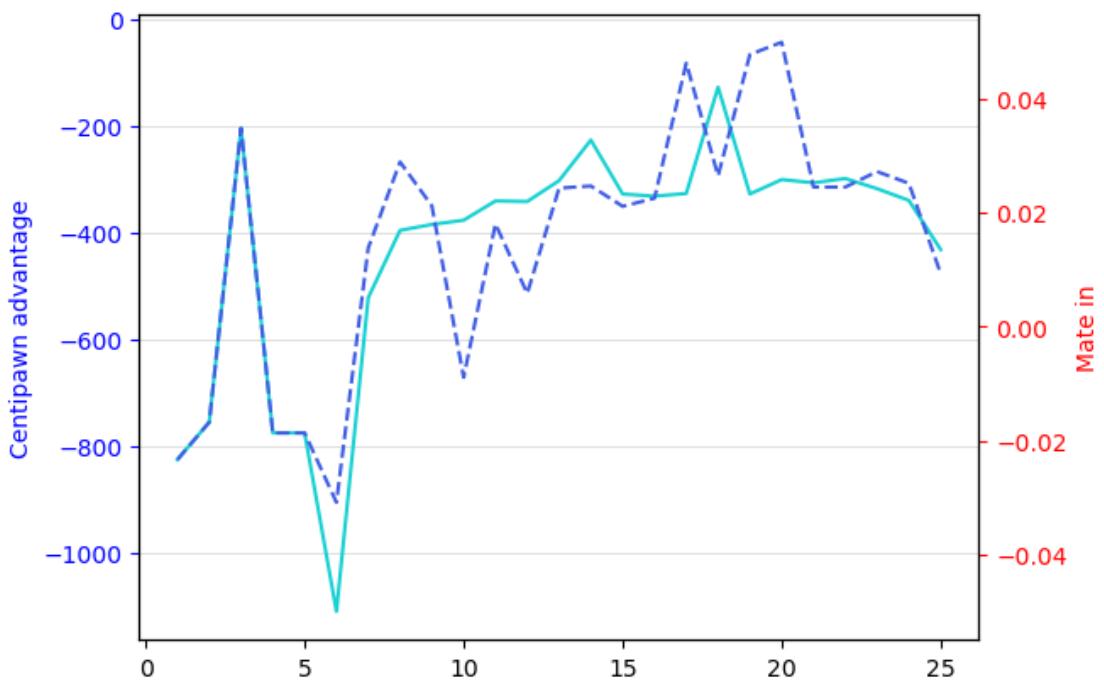
      plotevas(plot3best[0],plot3best[1],plot3best[2], 4)

```

MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 22, 23, 24, 25]



MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24, 25]



Stockfish fails to get the same mate value at depth = 20 on positions

sim_mirror

```
[ ]: p5mirror = '3b4/6k1/8/8/4N3/8/q7/2K1Rn2 w'
show_pos([p5mirror],300)
print(evaluationdouble(p5mirror,sim_mirror(p5mirror),50000,20))
```

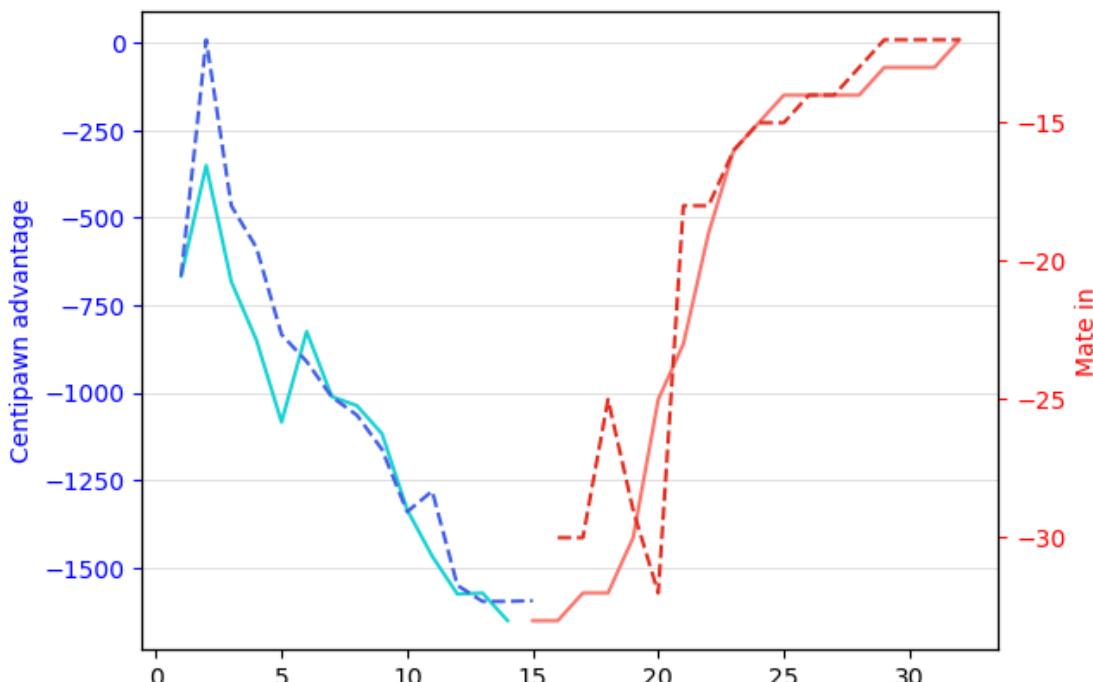
<IPython.core.display.HTML object>

({'type': 'mate', 'value': -25}, {'type': 'mate', 'value': 32})

```
[ ]: chemin = os.path.join('plotevas', 'p5mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot5mirror = pickle.load(fichier)
```

```
plotevas(plot5mirror[0],plot5mirror[1],plot5mirror[2], 0)
```

MR verified at depth = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 23, 24, 26, 27, 32]



sim_axis

```
[ ]: p4axis = '3N4/8/B3k3/1RK2B2/8/8/4P2r/6B1 b'
show_pos([p4axis],300)
print(evaluationdouble(p4axis,sim_axis(p4axis),50000,20))
```

<IPython.core.display.HTML object>

```

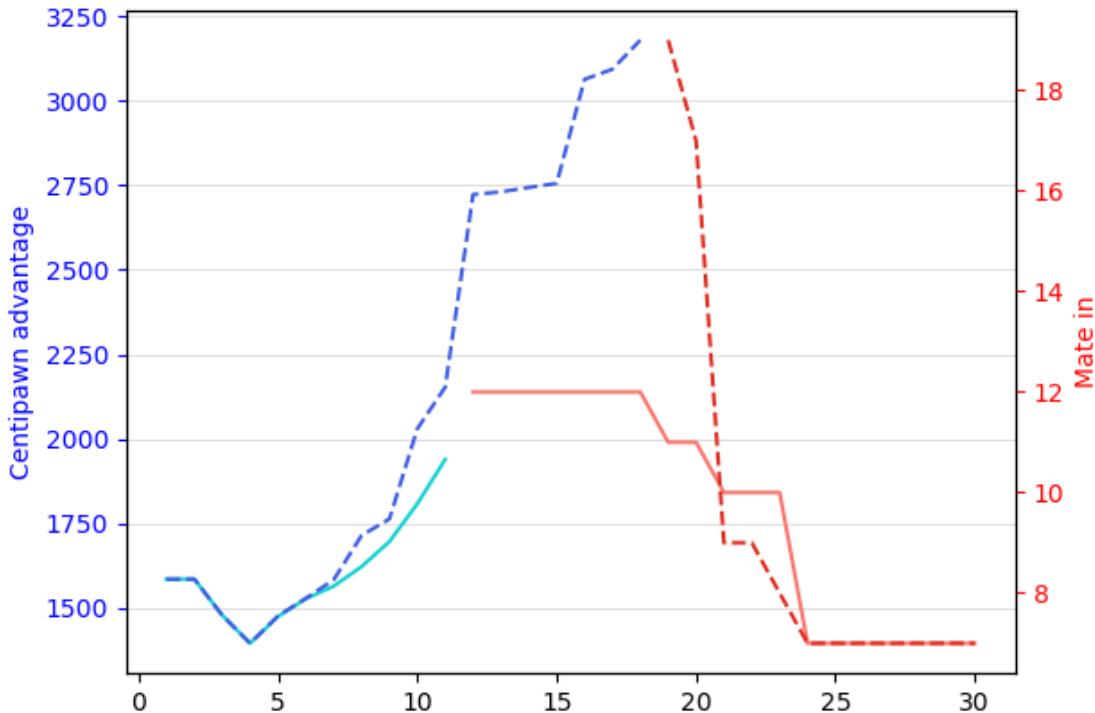
({'type': 'mate', 'value': 11}, {'type': 'mate', 'value': 17})

[ ]: chemin = os.path.join('plotevas', 'p4axis.pkl')
with open(chemin, 'rb') as fichier:
    plot4axis = pickle.load(fichier)

plotevas(plot4axis[0],plot4axis[1],plot4axis[2], 1)

```

MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 24, 25, 26, 27, 28, 29, 30]



```

sim_diag
[ ]: p4diag = '8/1nK5/6k1/8/6n1/8/8/4B1Qn b'
show_pos([p4diag],300)
print(evaluationondouble(p4diag,sim_diag(p4diag),50000,20))

<IPython.core.display.HTML object>

```

({'type': 'cp', 'value': 15265}, {'type': 'mate', 'value': 40})

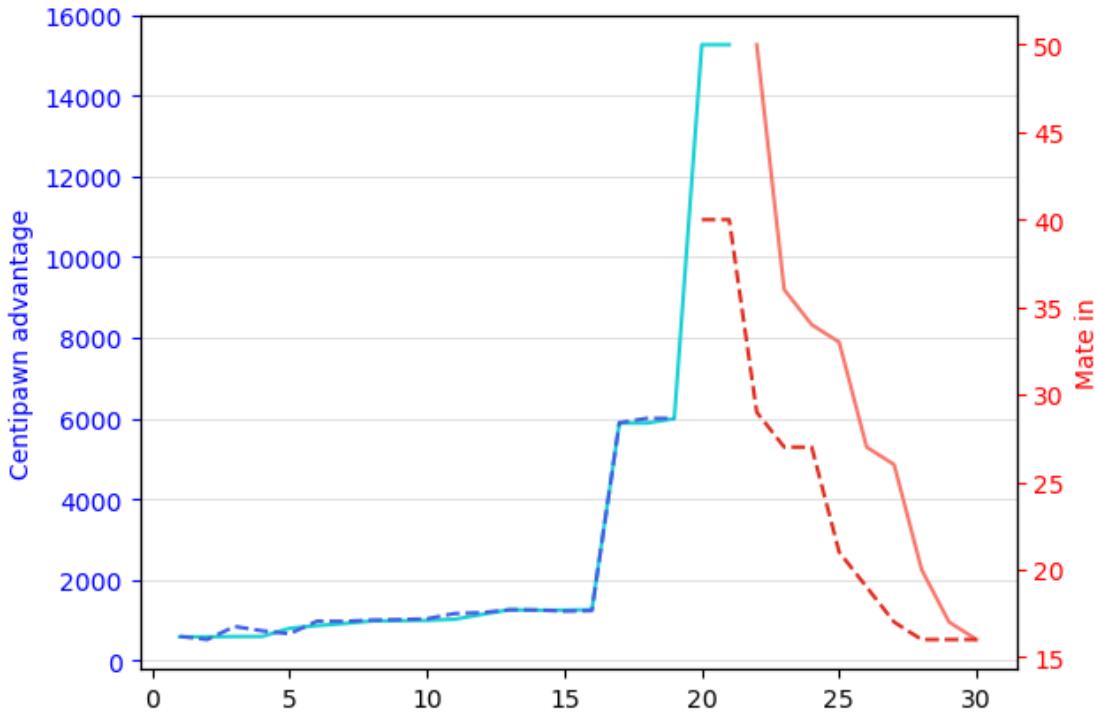
```

[ ]: chemin = os.path.join('plotevas', 'p4diag.pkl')
with open(chemin, 'rb') as fichier:
    plot4diag = pickle.load(fichier)

plotevas(plot4diag[0],plot4diag[1],plot4diag[2], 2)

```

```
MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,  
18, 19, 30]
```



best_move

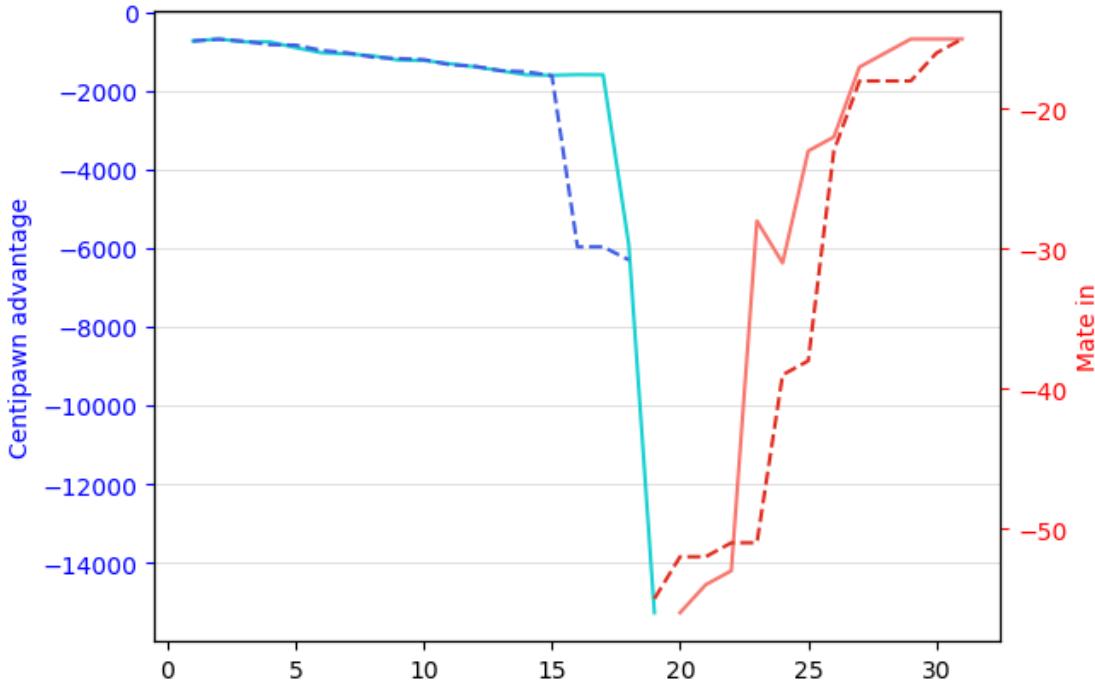
```
[ ]: p4best = '1n6/8/8/8/1K6/5pP1/3P1n2/2k5 w'  
show_pos([p4best],300)  
  
print(evaluation(p4best,50000,20)[1],evaluation(sim_axis(p4best),50000,20)[1])
```

```
<IPython.core.display.HTML object>
```

```
[{'Move': 'd2d4', 'Centipawn': None, 'Mate': -56}, {'Move': 'e2e4',  
'Centipawn': None, 'Mate': -52}]
```

```
[ ]: chemin = os.path.join('plot4best', 'p4best.pkl')  
with open(chemin, 'rb') as fichier:  
    plot4best = pickle.load(fichier)  
  
plot4best[0],plot4best[1],plot4best[2], 4)
```

```
MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 31]
```



Stockfish behaviour with near-mate positions

```
sim_mirror
[ ]: p6mirror = '8/1BK5/8/1pk5/8/8/5P2/8 b'
p7mirror = '3K4/8/7N/6n1/8/3k4/3p4/8 w'
show_pos([p6mirror,p7mirror],300)
print(evaluationdouble(p6mirror,sim_mirror(p6mirror),50000,20))
print(evaluationdouble(p7mirror,sim_mirror(p7mirror),50000,20))
```

<IPython.core.display.HTML object>

```
({'type': 'cp', 'value': 0}, {'type': 'cp', 'value': -1293})
({'type': 'cp', 'value': -15265}, {'type': 'cp', 'value': 5964})
```

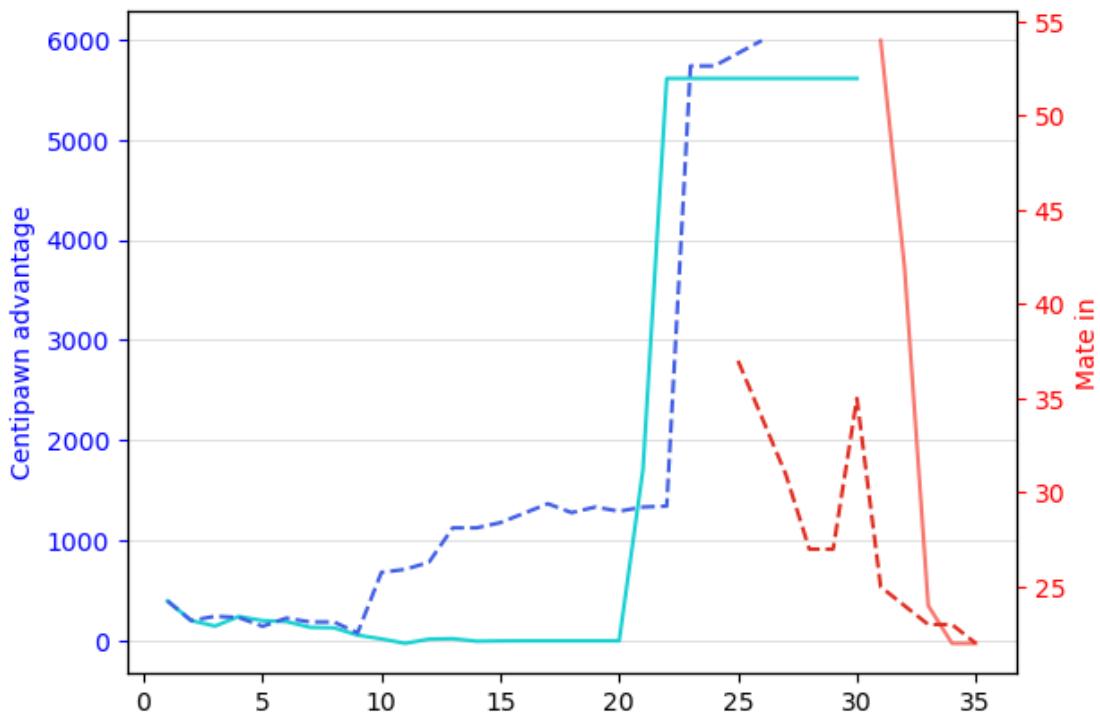
```
chemin = os.path.join('plotevas', 'p6mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot6mirror = pickle.load(fichier)

plotevas(plot6mirror[0],plot6mirror[1],plot6mirror[2], 0)

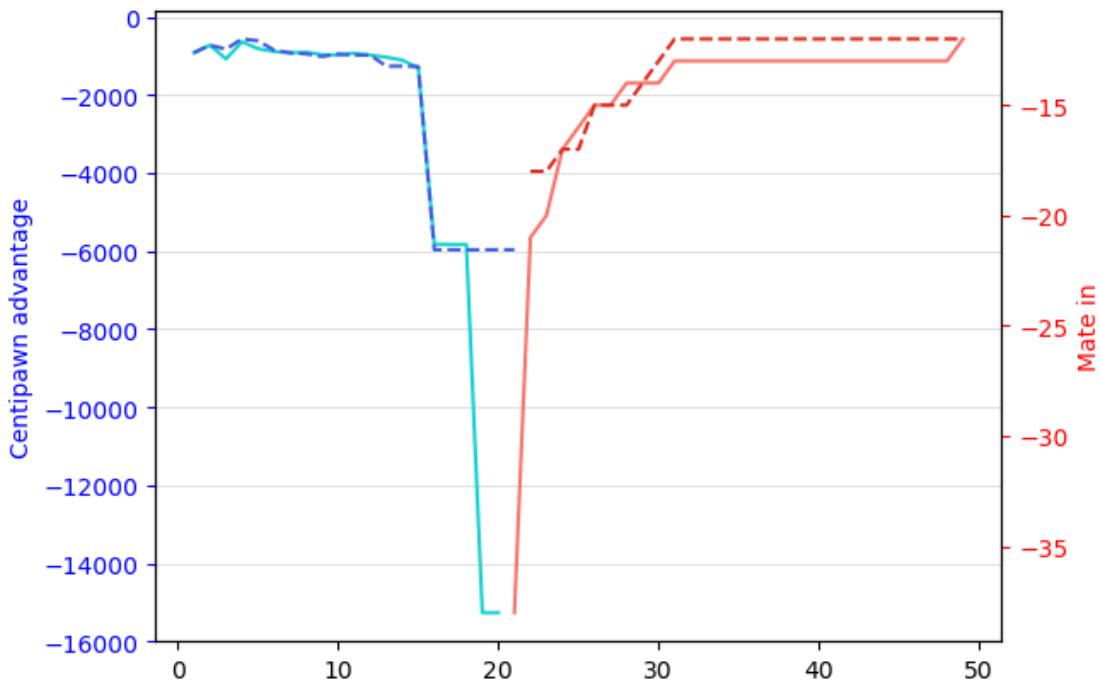
chemin = os.path.join('plotevas', 'p7mirror.pkl')
with open(chemin, 'rb') as fichier:
    plot7mirror = pickle.load(fichier)

plotevas(plot7mirror[0],plot7mirror[1],plot7mirror[2], 0)
```

MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 21, 23, 24, 26, 35]



MR verified at depth = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 24, 26, 27, 29, 49]



```

sim_axis
[ ]: p5axis = '3RR3/6k1/K7/8/8/7r/8/8 b'
      show_pos([p5axis],300)
      print(evaluationdouble(p5axis,sim_axis(p5axis),50000,20))

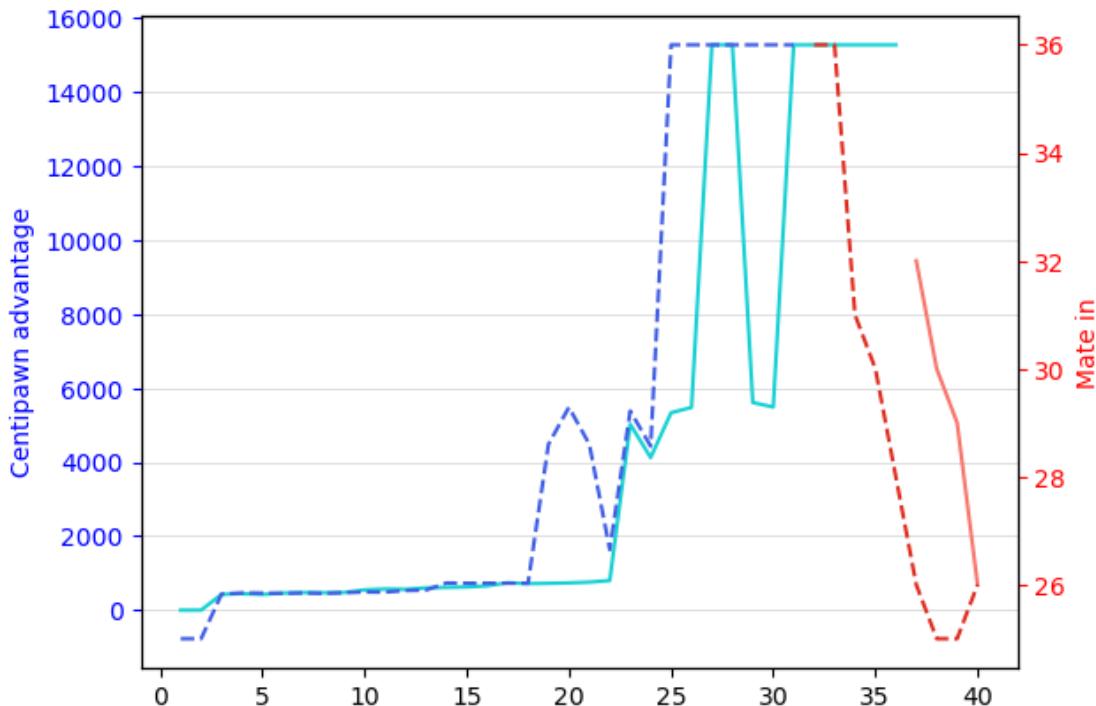
<IPython.core.display.HTML object>
({'type': 'cp', 'value': 740}, {'type': 'cp', 'value': 5480})

[ ]: chemin = os.path.join('plotevas', 'p5axis.pkl')
      with open(chemin, 'rb') as fichier:
          plot5axis = pickle.load(fichier)

      plotevas(plot5axis[0],plot5axis[1],plot5axis[2], 1)

MR verified at depth = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 23, 24, 27, 28, 31, 40]

```



It eventually converges at a certain point.

sim_diag

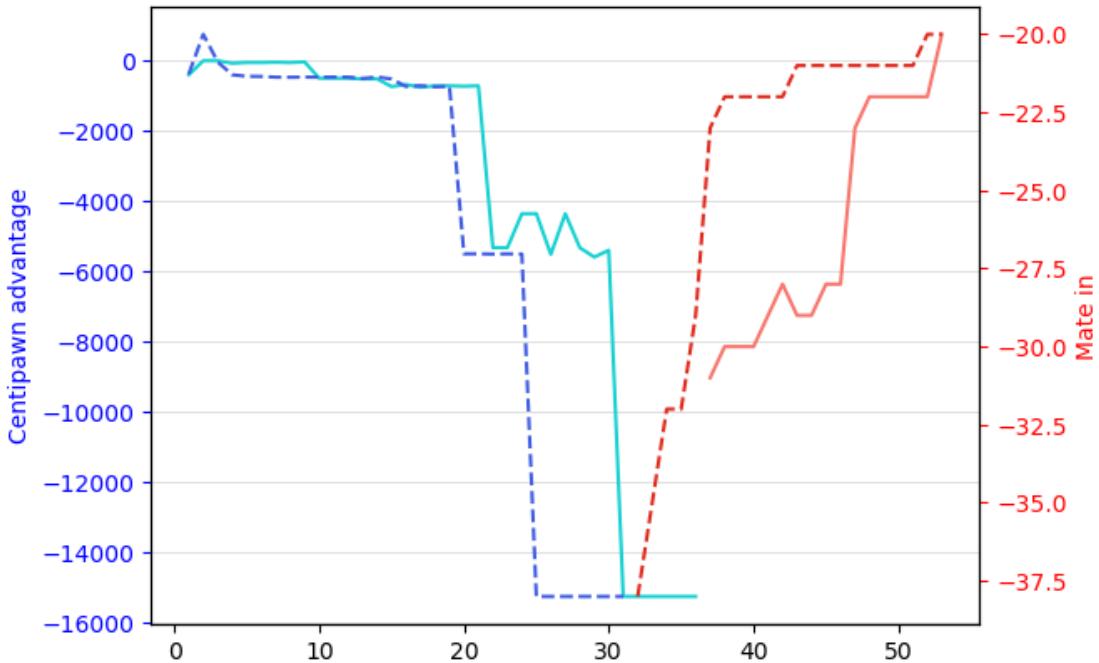
```
[ ]: p5diag = '3r4/8/8/R2K4/8/8/2N3r1/5k2 w'
show_pos([p5diag],300)
print(evaluationdouble(p5diag,sim_diag(p5diag),50000,20))

<IPython.core.display.HTML object>
({'type': 'cp', 'value': -727}, {'type': 'cp', 'value': -5502})

[ ]: chemin = os.path.join('plotevas', 'p5diag.pkl')
with open(chemin, 'rb') as fichier:
    plot5diag = pickle.load(fichier)

plotevas(plot5diag[0],plot5diag[1],plot5diag[2], 2)
```

MR verified at depth = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22, 23, 24, 31, 53]



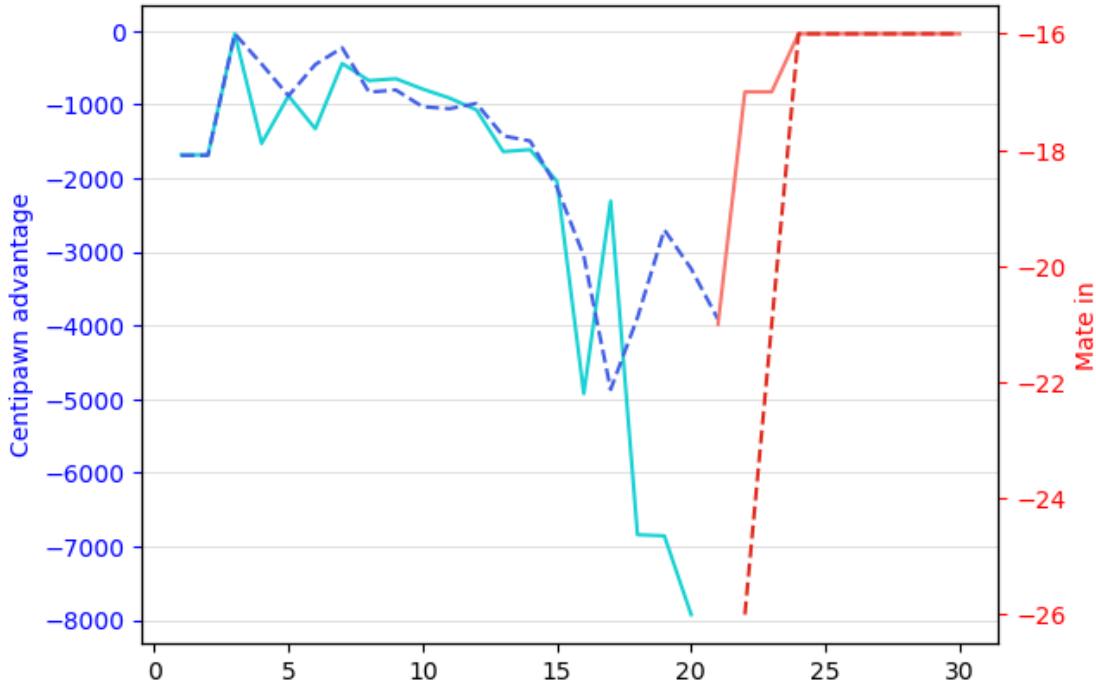
```
best_move
[ ]: p5best = 'q1B1R3/2k4R/8/pP6/2B5/r4q2/1R1B4/1K2R2q b'
show_pos([p5best,sim_axis(p5best)],300)
print(evaluation(p5best,50000,20)[1],evaluation(sim_axis(p5best),50000,20)[1])

<IPython.core.display.HTML object>
[{'Move': 'h1h7', 'Centipawn': -7925, 'Mate': None}, {'Move': 'a1a7', 'Centipawn': -3229, 'Mate': None}]
```

```
[ ]: chemin = os.path.join('plottevas', 'p5best.pkl')
with open(chemin, 'rb') as fichier:
    plot5best = pickle.load(fichier)

plottevas(plot5best[0],plot5best[1],plot5best[2], 4)
```

MR verified at depth = [2, 3, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 24, 25, 26, 27, 28, 29, 30]



0.9 Tests on the Carlsen-Nepo game

The article used the MR_mirror on the 20 first positions of a Carlsen-Nepo game, and it was violated on 4 positions. Experience is reproduced below.

```
[ ]: STOCKFISH16_LOCATION = "/home/mathieuacher/Downloads/
↳stockfish-ubuntu-x86-64-avx2/stockfish/stockfish-ubuntu-x86-64-avx2"
def carlsen_nepo_analysis(depth,v):
    if v==15:
        stockfish = Stockfish(STOCKFISH15_LOCATION)
    if v==16:
        stockfish = Stockfish(STOCKFISH16_LOCATION)
    stockfish.set_elo_rating(50000)
    stockfish.set_depth(depth)
    moves1 = ↳['d2d4', 'g8f6', 'g1f3', 'd7d5', 'g2g3', 'e7e6', 'f1g2', 'f8e7', 'e1g1', 'e8g8', 'b2b3', 'c7c5', 'd4c5']
```

```

index = 0
indexy = []
pos1s = []
pos2s = []
ev1s = []
ev2s = []
MR = []
epsilons = []
deltas = []
for move in moves1:
    indexy.append(index)
    print(index, end='', flush=True)
    print('\r', end='', flush=True)
    stockfish.make_moves_from_current_position([move])
    pos1s.append(stockfish.get_fen_position())
    ev1s.append(stockfish.get_evaluation())
    index+=1
    stockfish.set_fen_position('rnbqkbnr/pppppppp/8/8/8/8/PPPPPPP/RNBQKBNR b'
    ↪KQkq - 0 1')
    print(' II', end='', flush=True)
    print('\r', end='', flush=True)
    moves2 = [
    ↪['d7d5', 'g1f3', 'g8f6', 'd2d4', 'g7g6', 'e2e3', 'f8g7', 'f1e2', 'e8g8', 'e1g1', 'b7b6', 'c2c4', 'd5c4'
    index = 0
    for move in moves2:
        print(index, end='', flush=True)
        print('\r', end='', flush=True)
        stockfish.make_moves_from_current_position([move])
        pos2s.append(stockfish.get_fen_position())
        ev2s.append(stockfish.get_evaluation())
        index+=1
    for k in range(len(pos1s)):
        MR.append(MR_mirror(ev1s[k], ev2s[k], 0.25, 25))
        epsilons.append(abs(ev1s[k].get('value')+ev2s[k].get('value')))
        if ev1s[k].get('value')==0 and ev2s[k].get('value')==0:
            deltas.append(0)
        else:
            deltas.append(round(abs(ev1s[k].get('value')+ev2s[k].get('value'))/
    ↪(abs(ev1s[k].get('value'))+abs(ev2s[k].get('value'))), 3))

    d = {'index': indexy, 'moves1': moves1, 'evaluation pos1': ev1s,
          'moves2': moves2, 'evaluation pos2': ev2s, 'epsilons': ↪
    ↪epsilons, 'deltas': deltas, 'MR respected': MR}
    df = pd.DataFrame(data=d)
    print('Done!', end='', flush=True)
    print('\r', end='', flush=True)
return df

```

```
[ ]: carlsen_nepo_analysis(30,15)
```

Done!

```
[ ]:      index moves1           evaluation pos1 moves2 \
0       0   d2d4 {'type': 'cp', 'value': 27}   d7d5
1       1   g8f6 {'type': 'cp', 'value': 36}   g1f3
2       2   g1f3 {'type': 'cp', 'value': 28}   g8f6
3       3   d7d5 {'type': 'cp', 'value': 34}   d2d4
4       4   g2g3 {'type': 'cp', 'value': -19}  g7g6
5       5   e7e6 {'type': 'cp', 'value': 38}   e2e3
6       6   f1g2 {'type': 'cp', 'value': 28}   f8g7
7       7   f8e7 {'type': 'cp', 'value': 51}   f1e2
8       8   e1g1 {'type': 'cp', 'value': 30}   e8g8
9       9   e8g8 {'type': 'cp', 'value': 45}   e1g1
10     10   b2b3 {'type': 'cp', 'value': 0}    b7b6
11     11   c7c5 {'type': 'cp', 'value': 13}  c2c4
12     12   d4c5 {'type': 'cp', 'value': -2}  d5c4
13     13   e7c5 {'type': 'cp', 'value': 24}  e2c4
14     14   c2c4 {'type': 'cp', 'value': 7}   c7c5
15     15   d5c4 {'type': 'cp', 'value': 27}  d4c5
16     16   d1c2 {'type': 'cp', 'value': 2}   d8c7
17     17   d8e7 {'type': 'cp', 'value': 12}  d1e2
18     18   b1d2 {'type': 'cp', 'value': 8}   b8d7
19     19   b8c6 {'type': 'cp', 'value': 14}  b1c3
```

		evaluation pos2	epsilons	deltas	MR	respected
0	{'type': 'cp', 'value': -23}		4	0.080		True
1	{'type': 'cp', 'value': -37}		1	0.014		True
2	{'type': 'cp', 'value': -22}		6	0.120		True
3	{'type': 'cp', 'value': -38}		4	0.056		True
4	{'type': 'cp', 'value': 12}		7	0.226		True
5	{'type': 'cp', 'value': -42}		4	0.050		True
6	{'type': 'cp', 'value': -12}		16	0.400		True
7	{'type': 'cp', 'value': -45}		6	0.062		True
8	{'type': 'cp', 'value': -28}		2	0.034		True
9	{'type': 'cp', 'value': -36}		9	0.111		True
10	{'type': 'cp', 'value': -9}		9	1.000		True
11	{'type': 'cp', 'value': -36}		23	0.469		True
12	{'type': 'cp', 'value': 11}		9	0.692		True
13	{'type': 'cp', 'value': -9}		15	0.455		True
14	{'type': 'cp', 'value': 14}		21	1.000		True
15	{'type': 'cp', 'value': 0}		27	1.000		False
16	{'type': 'cp', 'value': 0}		2	1.000		True
17	{'type': 'cp', 'value': 0}		12	1.000		True
18	{'type': 'cp', 'value': -5}		3	0.231		True
19	{'type': 'cp', 'value': 0}		14	1.000		True

For some reasons we are getting only one error, and not 4 as the article got.

```
[ ]: carlsen_nepo_analysis(30,16)
```

Done!

```
[ ]:   index moves1           evaluation pos1 moves2 \
 0      0   d2d4  {'type': 'cp', 'value': 28}   d7d5
 1      1   g8f6  {'type': 'cp', 'value': 32}   g1f3
 2      2   g1f3  {'type': 'cp', 'value': 24}   g8f6
 3      3   d7d5  {'type': 'cp', 'value': 33}   d2d4
 4      4   g2g3  {'type': 'cp', 'value': 0}    g7g6
 5      5   e7e6  {'type': 'cp', 'value': 39}   e2e3
 6      6   f1g2  {'type': 'cp', 'value': 17}   f8g7
 7      7   f8e7  {'type': 'cp', 'value': 35}   f1e2
 8      8   e1g1  {'type': 'cp', 'value': 31}   e8g8
 9      9   e8g8  {'type': 'cp', 'value': 34}   e1g1
10     10   b2b3  {'type': 'cp', 'value': 10}   b7b6
11     11   c7c5  {'type': 'cp', 'value': 10}   c2c4
12     12   d4c5  {'type': 'cp', 'value': 0}    d5c4
13     13   e7c5  {'type': 'cp', 'value': 0}    e2c4
14     14   c2c4  {'type': 'cp', 'value': 0}    c7c5
15     15   d5c4  {'type': 'cp', 'value': -1}  d4c5
16     16   d1c2  {'type': 'cp', 'value': 0}    d8c7
17     17   d8e7  {'type': 'cp', 'value': 0}    d1e2
18     18   b1d2  {'type': 'cp', 'value': 0}    b8d7
19     19   b8c6  {'type': 'cp', 'value': 0}    b1c3

           evaluation pos2  epsilons  deltas  MR respected
 0  {'type': 'cp', 'value': -28}      0  0.000  True
 1  {'type': 'cp', 'value': -37}      5  0.072  True
 2  {'type': 'cp', 'value': -22}      2  0.043  True
 3  {'type': 'cp', 'value': -27}      6  0.100  True
 4  {'type': 'cp', 'value': 7}       7  1.000  True
 5  {'type': 'cp', 'value': -30}      9  0.130  True
 6  {'type': 'cp', 'value': -21}      4  0.105  True
 7  {'type': 'cp', 'value': -19}     16  0.296  True
 8  {'type': 'cp', 'value': -27}      4  0.069  True
 9  {'type': 'cp', 'value': -25}      9  0.153  True
10  {'type': 'cp', 'value': -7}       3  0.176  True
11  {'type': 'cp', 'value': -17}      7  0.259  True
12  {'type': 'cp', 'value': 0}        0  0.000  True
13  {'type': 'cp', 'value': 0}        0  0.000  True
14  {'type': 'cp', 'value': -3}      3  1.000  True
15  {'type': 'cp', 'value': 2}       1  0.333  True
16  {'type': 'cp', 'value': -1}      1  1.000  True
17  {'type': 'cp', 'value': 0}        0  0.000  True
18  {'type': 'cp', 'value': 0}        0  0.000  True
```

```
19      {'type': 'cp', 'value': 0}          0   0.000      True
```

Trying with Stockfish 16, and MR are not violated anymore.

0.10 Tests with Stockfish 16

Since the original paper was using Stockfish 15, the previous tests were conducted on this version. The following tests have been conducted with the latest version, Stockfish 16.

0.10.1 sim_mirror

depth = 10, entire dataset (Link V15)

```
[ ]: path = os.path.join('sim_mirror_d=10','evaluations50000_sim_mirror_d_10_v16.  
    ↪pkl')  
with open(path, 'rb') as file:  
    evaluations50000_sim_mirror_d_10_v16 = pickle.load(file)  
  
tests(evaluations50000_sim_mirror_d_10_v16,0,10, ', original dataset, SF16')
```

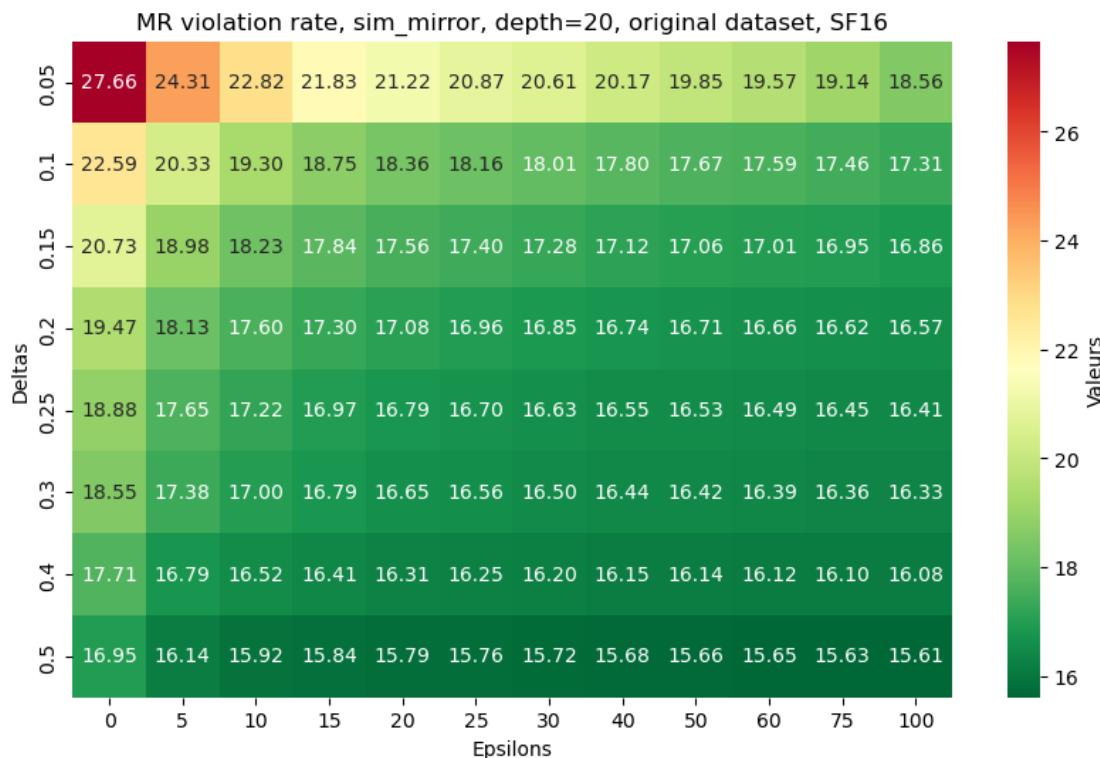


Note : the failure rate with Stockfish 15 was comprised between 10.48% and 36.15%.

depth = 20, entire dataset (Link V15)

```
[ ]: path = os.path.join('sim_mirror_d=20', 'evaluations50000_sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evaluations50000_sim_mirror_d_20_v16 = pickle.load(file)

tests(evaluations50000_sim_mirror_d_20_v16, 0, 20, ', original dataset, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 19.32% and 36.31%.

Similar behaviour: increasing depth isn't better on the entire dataset.

depth = 10, splitting (Link V15)

```
[ ]: print("Mates proportion : ", len(split_mate_cp(evaluations50000_sim_mirror_d_10_v16)[1]) / (len(split_mate_cp(evaluations50000_sim_mirror_d_10_v16)[0]) + len(split_mate_cp(evaluations50000_sim_mirror_d_10_v16)[1])))
```

Mates proportion : 0.22773591535548354

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_10_v16)[0], 0, 10, ', original dataset, cp evas only, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 0.93% and 36.74%.

```
[ ]: print('Mates failure rate :',  
      ↪ 100-MR(split_mate_cp(evaluations50000_sim_mirror_d_10_v16)[1],0,1,1), '%')
```

Mates failure rate : 25.492885307731868 %

Note : the mates failure rate with Stockfish 15 was ~34.6%.

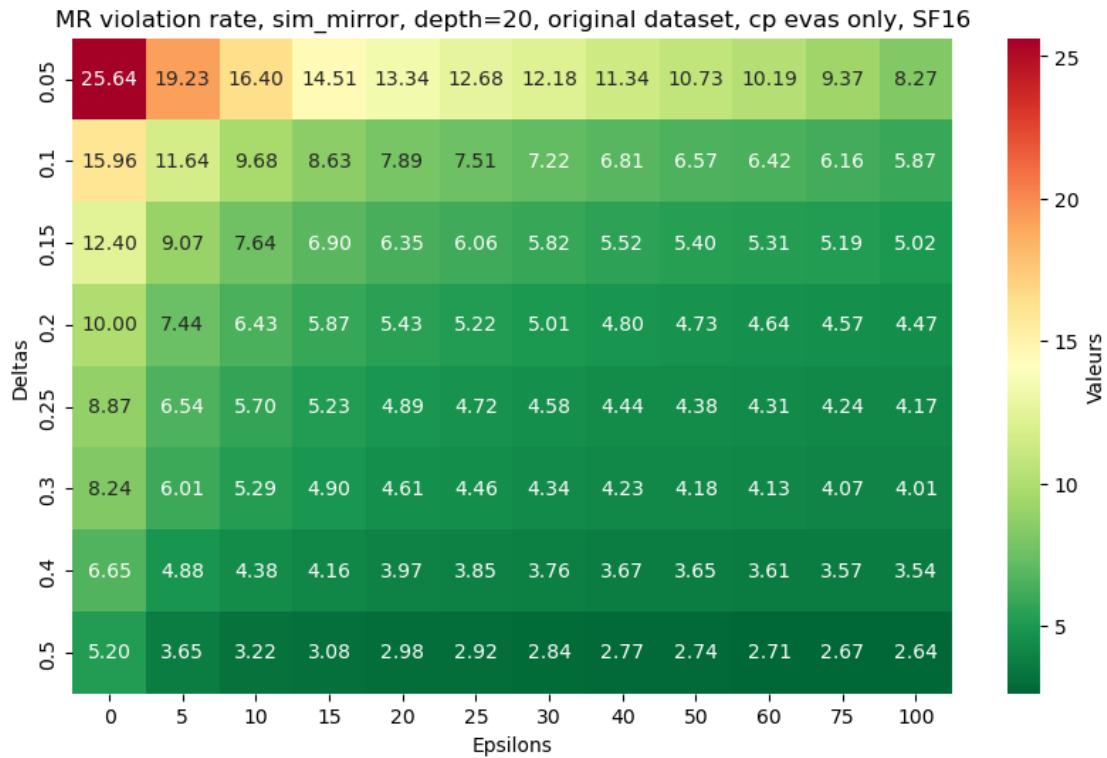
depth = 20, splitting (Link V15)

```
[ ]: print("Mates proportion : ",  
      ↪ len(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[1])/  
      ↪ (len(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[0])+len(split_mate_cp(evaluations5
```

Mates proportion : 0.47591514630620774

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[0],0,20, 'original',  
      ↪ dataset, cp evas only, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 2.01% and 38.67%.

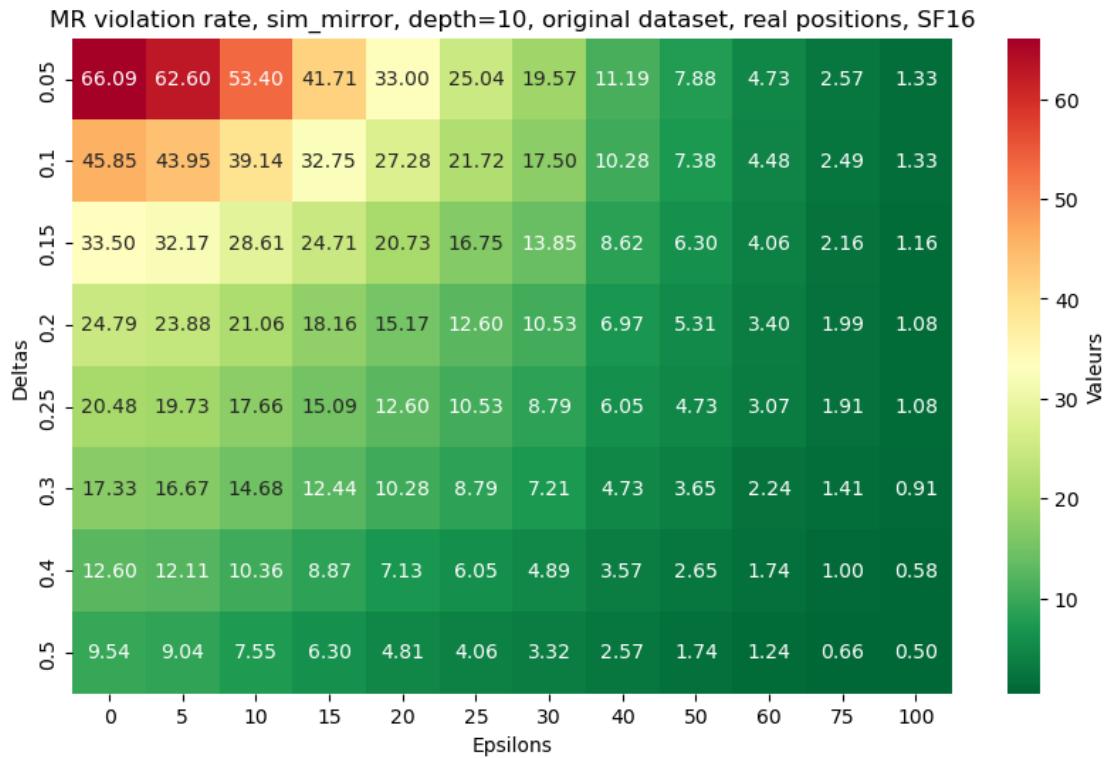
```
[ ]: print('Mates failure rate :', 100-MR(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[1], 0, 1, 1), '%')
```

Mates failure rate : 29.89656870792973 %

Note : the mates failure rate with Stockfish 15 was ~34.3%.

depth = 10, real positions (Link V15)

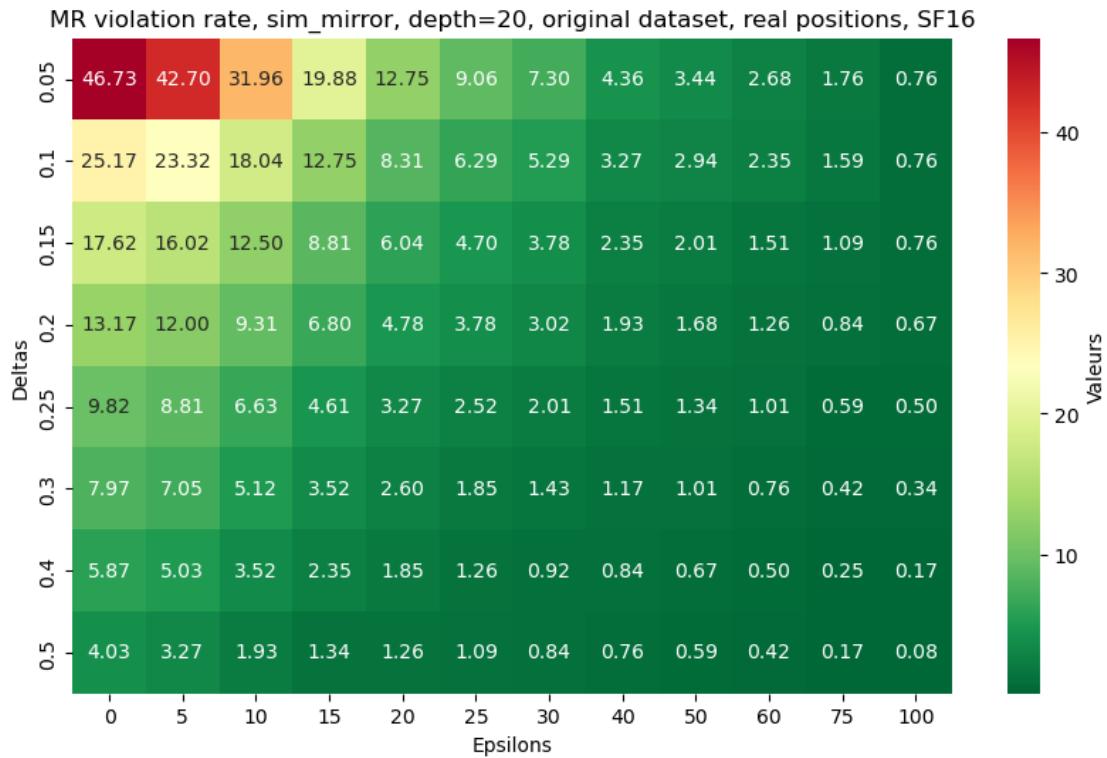
```
[ ]: evaluations50000_sim_mirror_d_10_v16_real = real_positions(evaluations50000_sim_mirror_d_10_v16[24000:])
evaluations50000_sim_mirror_d_10_v16_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_mirror_d_10_v16_real, 0, 10, 'original dataset, real positions, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 1.00% and 66.25%.

depth = 20, real positions (Link V15)

```
[ ]: evaluations50000_sim_mirror_d_20_v16_real = ↵
    ↵real_positions(evaluations50000_sim_mirror_d_20_v16[24000:])
evaluations50000_sim_mirror_d_20_v16_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_mirror_d_20_v16_real, 0, 20, ', original dataset, real ↵
    ↵positions, SF16!')
```



Note : the failure rate with Stockfish 15 was comprised between 0.25% and 50.67%.

0.10.2 sim_axis

depth = 10, entire dataset (Link V15)

```
[ ]: path = os.path.join('sim_axis_d=10','evaluations50000_sim_axis_d_10_v16.pkl')
      with open(path, 'rb') as file:
          evaluations50000_sim_axis_d_10_v16 = pickle.load(file)

      tests(evaluations50000_sim_axis_d_10_v16,1,10, ', original dataset, SF16')
```

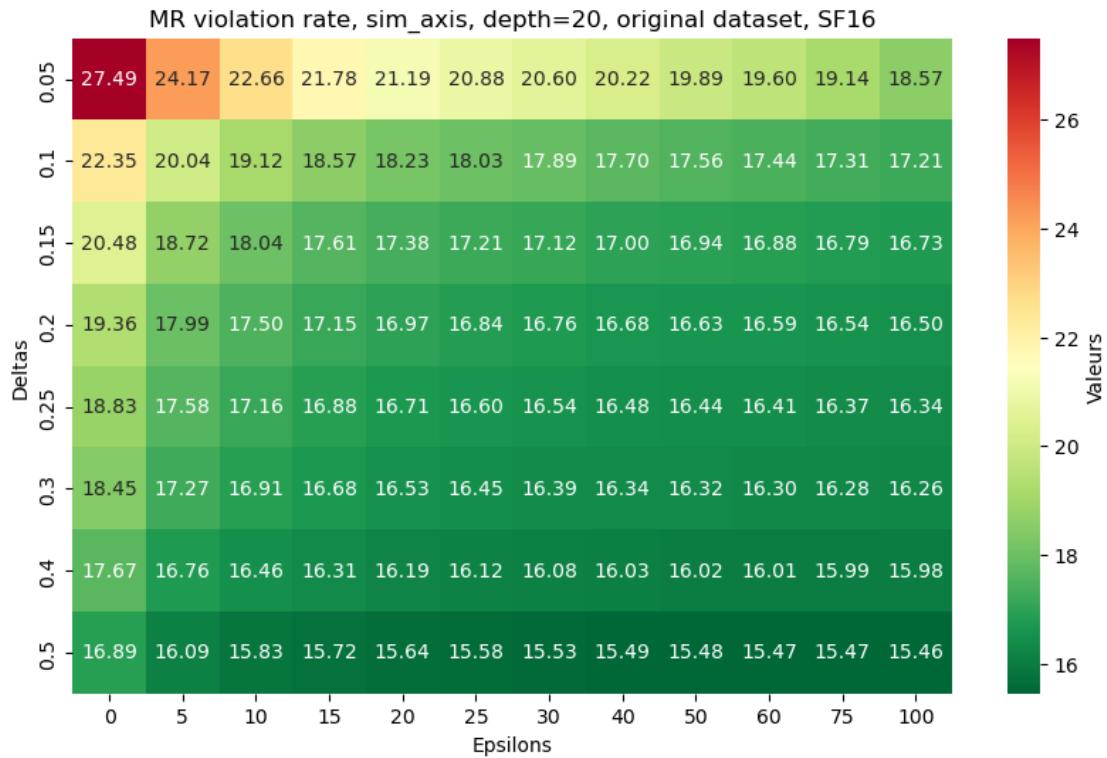


Note : the failure rate with Stockfish 15 was comprised between 10.42% and 35.90%.

depth = 20, entire dataset (Link V15)

```
[ ]: path = os.path.join('sim_axis_d=20','evaluations50000_sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evaluations50000_sim_axis_d_20_v16 = pickle.load(file)

      tests(evaluations50000_sim_axis_d_20_v16,1,20, ', original dataset, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 19.15% and 36.11%.

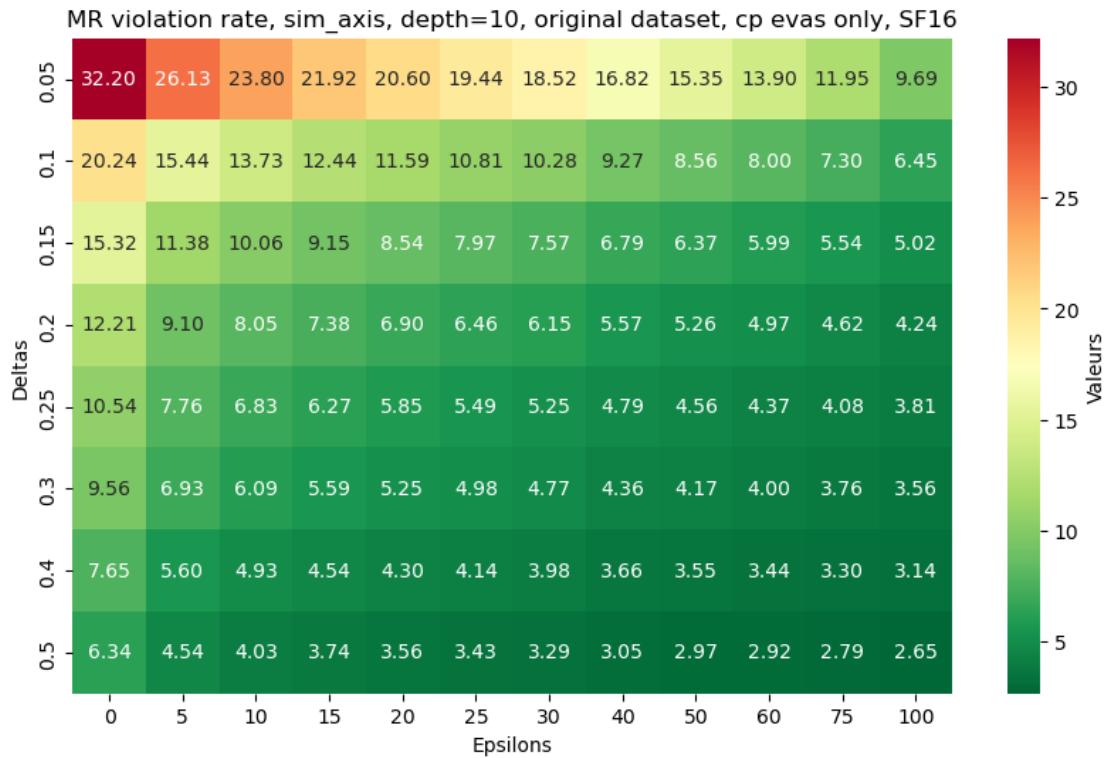
depth = 10, splitting (Link V15)

```
[ ]: print("Mates proportion : ", ↴
    ↴len(split_mate_cp(evaluations50000_sim_axis_d_10_v16)[1])/
    ↴(len(split_mate_cp(evaluations50000_sim_axis_d_10_v16)[0])+len(split_mate_cp(evaluations50000_sim_axis_d_10_v16)[1])))
```

Mates proportion : 0.22955350539141267

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_axis_d_10_v16)[0], 1, 10, 'original',
    ↴dataset, cp evas only, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 0.96% and 36.54%.

depth = 20, splitting (Link V15)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[1])/  
    ↪(len(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[0])+len(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[1]))
```

Mates proportion : 0.4768823277373116

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[0], 1, 20, ', original  
    ↪dataset, cp evas only, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 2.05% and 38.59%.

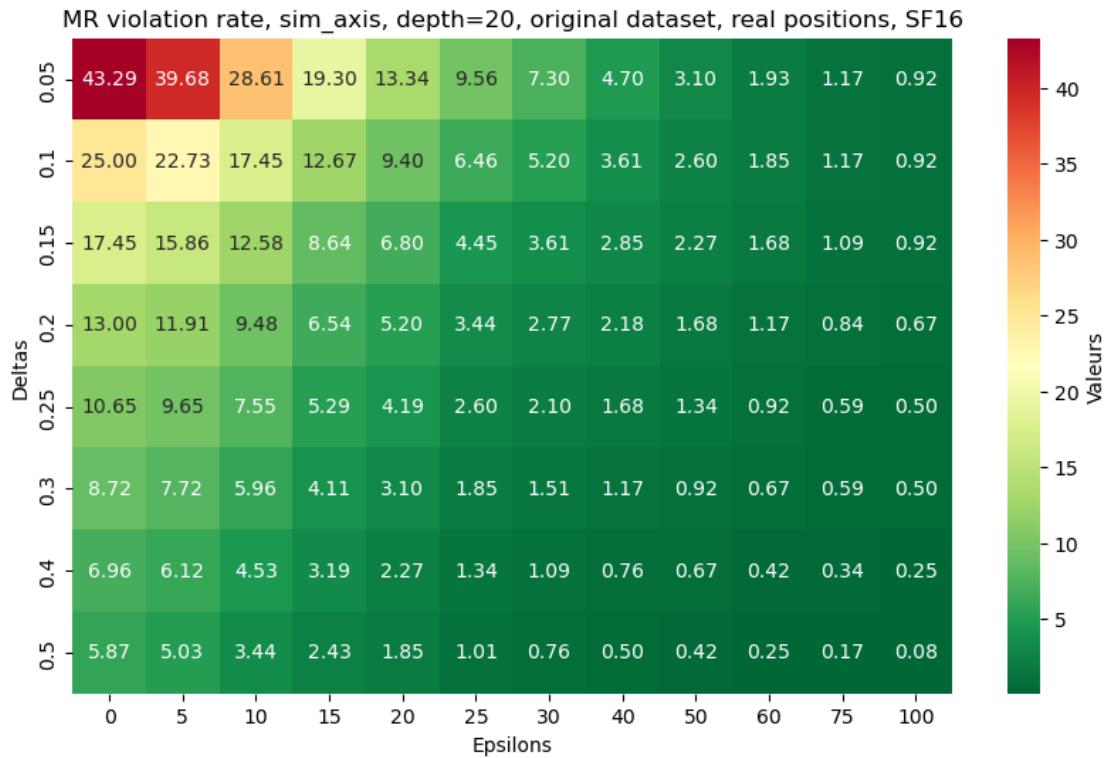
```
[ ]: print('Mates failure rate :')
      ↪', 100-MR(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[1], 1, 1, 1), '%')
```

Mates failure rate : 29.42906150453321 %

Note : the mates failure rate with Stockfish 15 was ~34.0%.

depth = 20, real positions (Link V15)

```
[ ]: evaluations50000_sim_axis_d_20_v16_real =
      ↪real_positions(evaluations50000_sim_axis_d_20_v16[24000:])
evaluations50000_sim_axis_d_20_v16_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_sim_axis_d_20_v16_real, 1, 20, ', original dataset, real
      ↪positions, SF16')
```



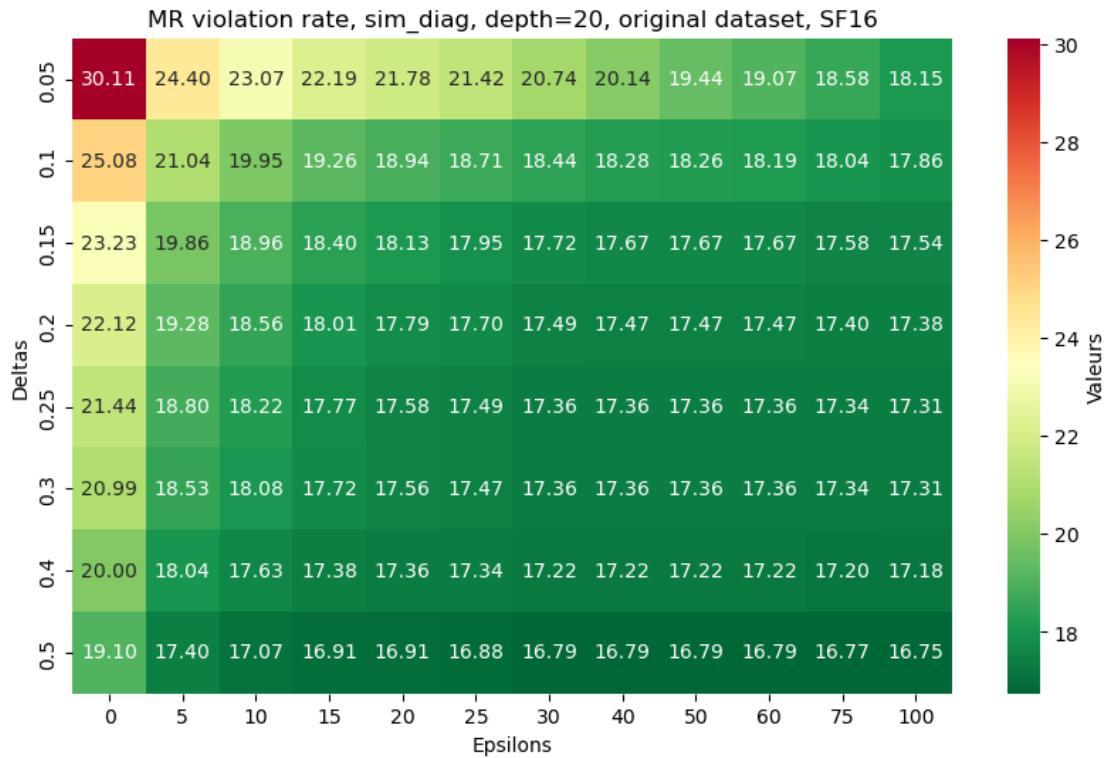
Note : the failure rate with Stockfish 15 was comprised between 0.25% and 50.04%.

0.10.3 sim_diag

depth = 20, entire dataset (Link V15)

```
[ ]: path = os.path.join('sim_diag_d=20', 'evaluations50000_sim_diag_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evaluations50000_sim_diag_d_20_v16 = pickle.load(file)

      tests(evaluations50000_sim_diag_d_20_v16, 2, 20, ', original dataset, SF16')
```



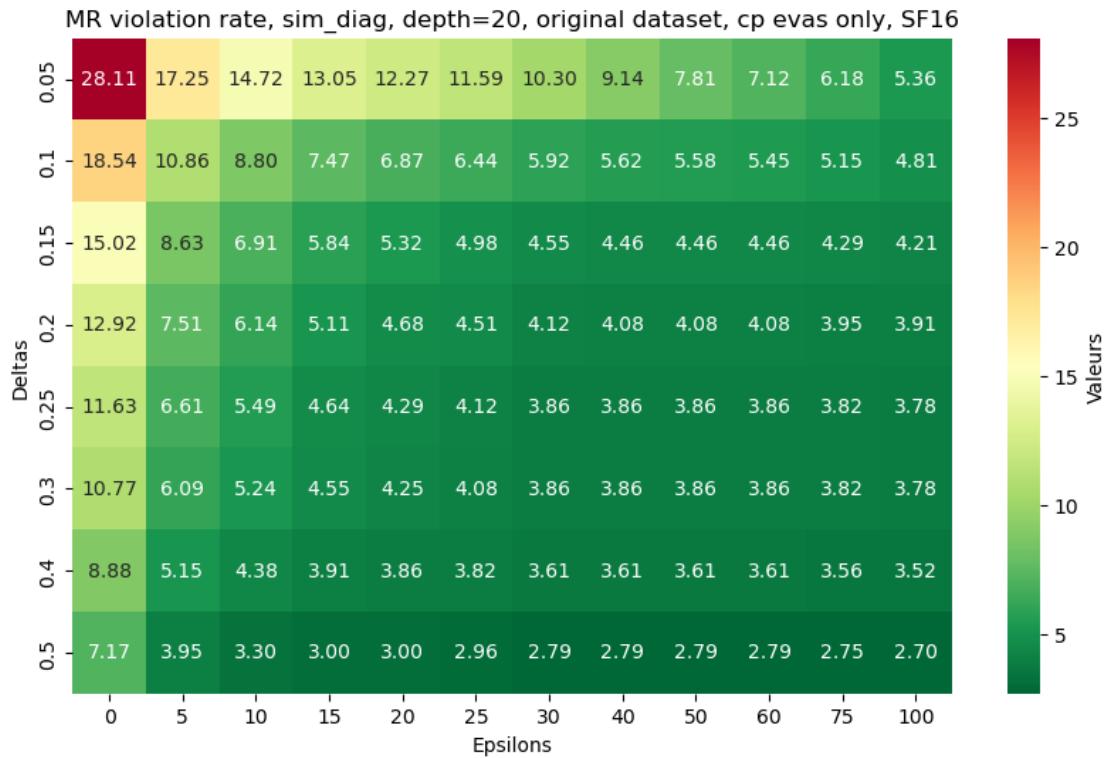
Note : the failure rate with Stockfish 15 was comprised between 22.28% and 41.36%. ##### depth = 20, splitting (Link V15)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_sim_diag_d_20_v16)[1])/  
    ↪(len(split_mate_cp(evaluations50000_sim_diag_d_20_v16)[0])+len(split_mate_cp(evaluations50000_sim_diag_d_20_v16)[1]))
```

Mates proportion : 0.47404063205417607

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_sim_diag_d_20_v16)[0],2,20, ', original  
    ↪dataset, cp evas only, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 2.41% and 48.55%.

```
[ ]: print('Mates failure rate :')
    ↵',100-MR(split_mate_cp(evaluations50000_sim_diag_d_20_v16)[1],2,1,1),'%')
```

Mates failure rate : 32.33333333333334 %

Note : the mates failure rate with Stockfish 15 was ~38.5%.

0.10.4 replace

depth = 20, entire dataset (Link V15)

```
[ ]: path = os.path.join('replace_d=20','evaluations50000_replace_d_20_v16.pkl')
with open(path, 'rb') as file:
    evaluations50000_replace_d_20_v16 = pickle.load(file)

tests(evaluations50000_replace_d_20_v16,3,20, ', original dataset, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 1.29% and 2.06%.

depth = 20, splitting (Link V15)

```
[ ]: print("Mates proportion : ", ↴
    ↴len(split_mate_cp(evaluations50000_replace_d_20_v16)[1])/(
    ↴len(split_mate_cp(evaluations50000_replace_d_20_v16)[0])+len(split_mate_cp(evaluations50000_replace_d_20_v16)[1]))
```

Mates proportion : 0.663164039696439

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_replace_d_20_v16)[0], 3, 20, ', original',
    ↴dataset, cp evas only, SF16')
```



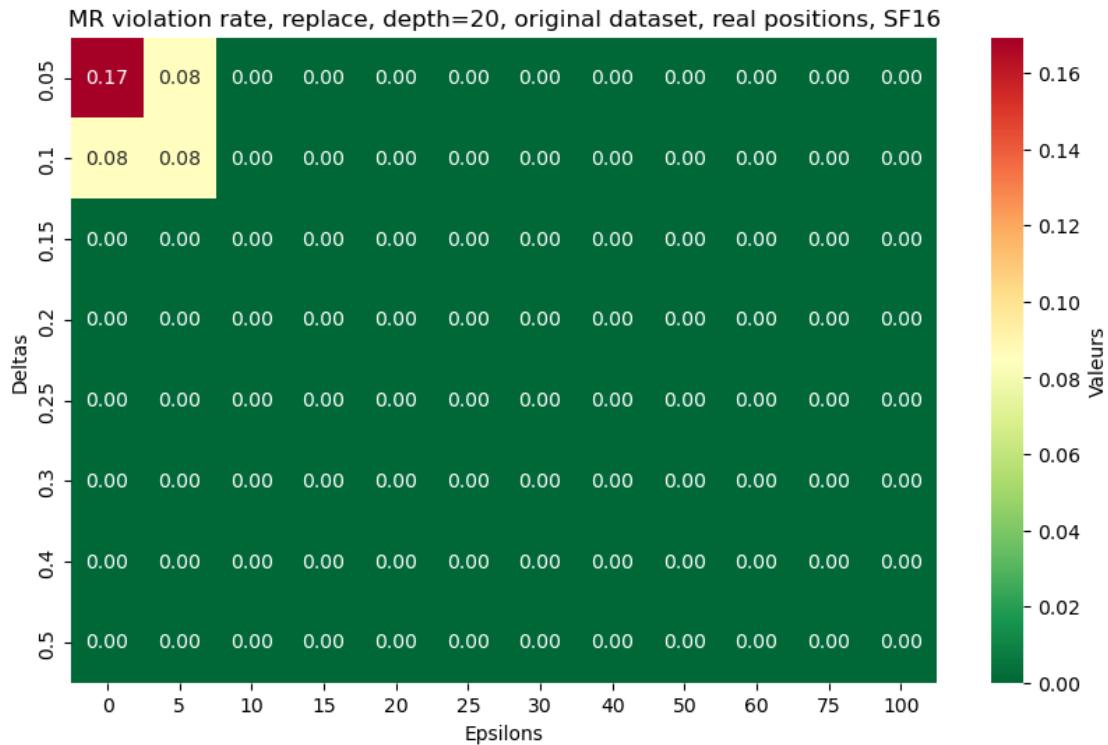
Note : the failure rate with Stockfish 15 was comprised between 0.22% and 2.86%.

```
[ ]: print('Mates failure rate :')
    ↵',100-MR(split_mate_cp(evaluations50000_replace_d_20_v16)[1],3,1,1),'%')
```

Mates failure rate : 1.464468629961587 %

depth = 20, real positions (Link V15)

```
[ ]: evaluations50000_replace_d_20_v16_real =
    ↵real_positions(evaluations50000_replace_d_20_v16[17000:])
evaluations50000_replace_d_20_v16_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_replace_d_20_v16_real,3,20, 'original dataset, real
    ↵positions, SF16')
```

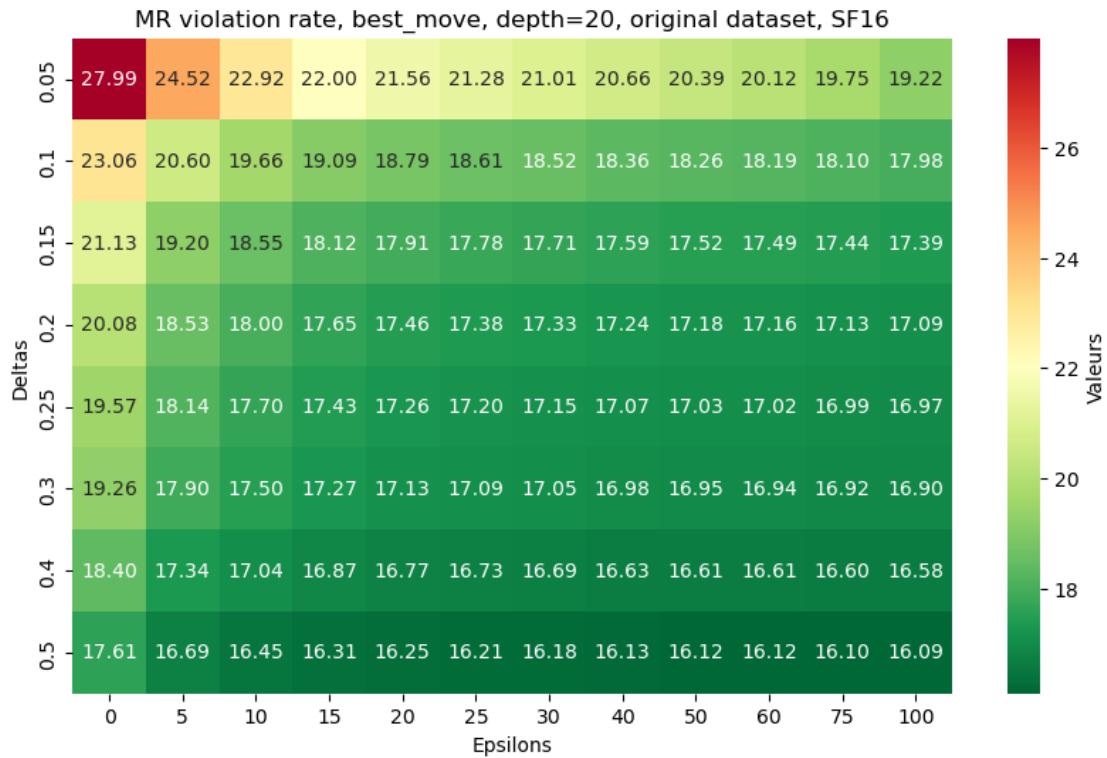


0.10.5 best_move

depth = 20, entire dataset (Link V15)

```
[ ]: path = os.path.join('best_move_d=20', 'evaluations50000_best_move_d_20_v16.pkl')
with open(path, 'rb') as file:
    evaluations50000_best_move_d_20_v16 = pickle.load(file)

tests(evaluations50000_best_move_d_20_v16, 4, 20, ', original dataset, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 19.68% and 36.68%.

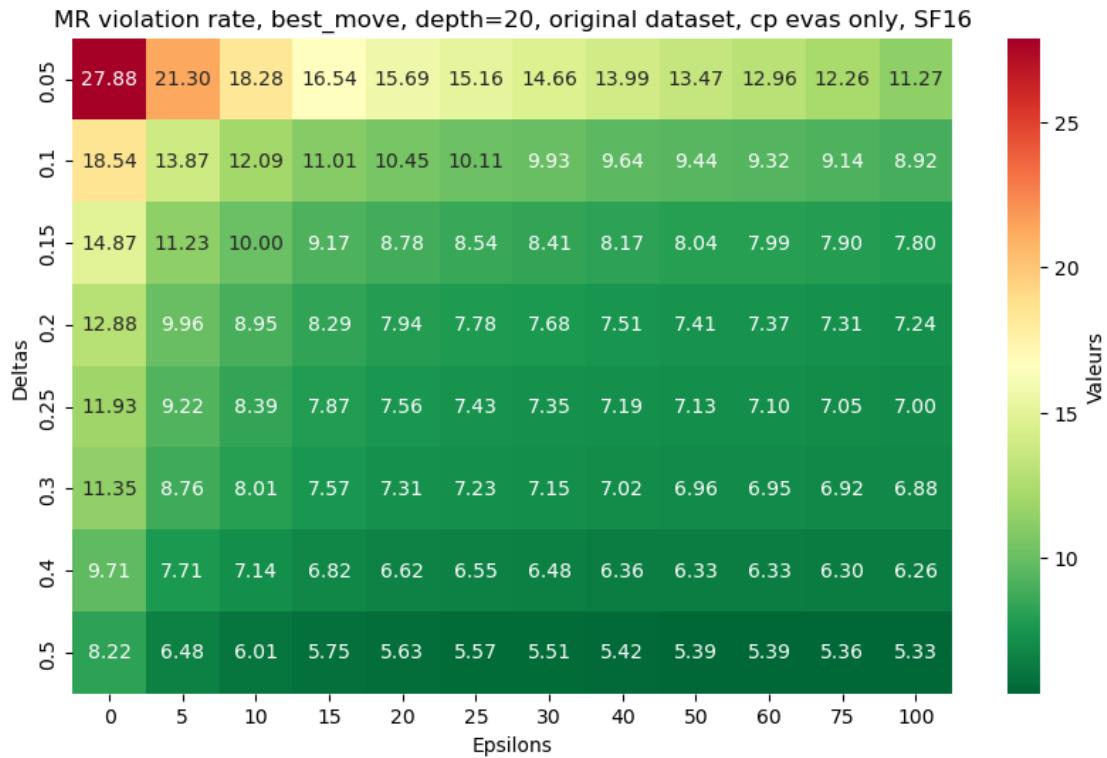
depth = 20, splitting (Link V15)

```
[ ]: print("Mates proportion : ",  
    ↪len(split_mate_cp(evaluations50000_best_move_d_20_v16)[1])/  
    ↪(len(split_mate_cp(evaluations50000_best_move_d_20_v16)[0])+len(split_mate_cp(evaluations50000_best_move_d_20_v16)[1])))
```

Mates proportion : 0.4718529330545584

Evaluations with a difference in centipawns

```
[ ]: tests(split_mate_cp(evaluations50000_best_move_d_20_v16)[0],4,20, ', original  
    ↪dataset, cp evas only, SF16')
```



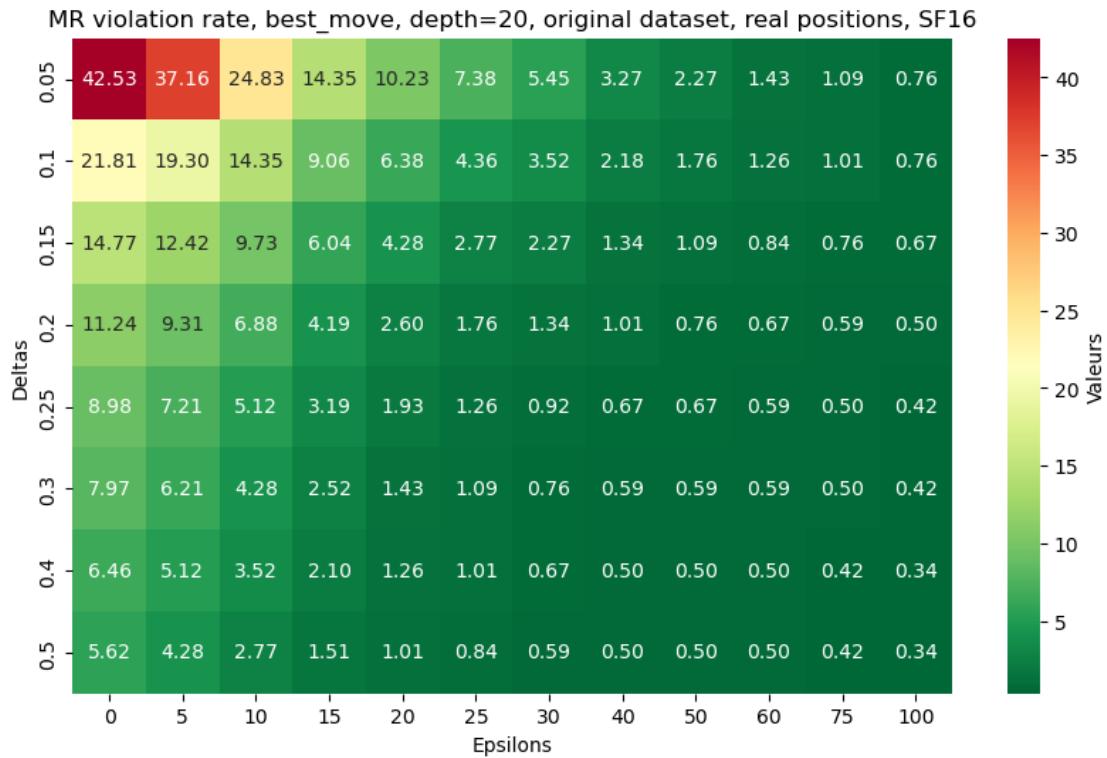
Note : the failure rate with Stockfish 15 was comprised between 5.81% and 41.58%.

```
[ ]: print('Mates failure rate :')
    ↪', 100-MR(split_mate_cp(evaluations50000_best_move_d_20_v16)[1], 4, 1, 1), '%')
```

Mates failure rate : 28.124738734219548 %

depth = 20, real positions (Link V15)

```
[ ]: evaluations50000_best_move_d_20_v16_real = ↪
    ↪real_positions(evaluations50000_best_move_d_20_v16[22000:])
evaluations50000_best_move_d_20_v16_real.reset_index(drop=True, inplace=True)
tests(evaluations50000_best_move_d_20_v16_real, 4, 20, ', original dataset, real
    ↪positions, SF16')
```



Note : the failure rate with Stockfish 15 was comprised between 0.34% and 48.95%.

0.10.6 Differences in evaluations between SF15 and SF16

We'll also show grids with using MR_equi to check the variations between SF15 and SF16 on the both original and mutated positions, by merging two dataframes and keeping only the columns about the positions we want, using the functions below.

```
[ ]: def merge(evas1,evas2):
    evas1 = evas1[['pos1','evaluation pos1']]
    evas2 = evas2[['pos1','evaluation pos1']]

    evas2 = evas2.rename(columns={'evaluation pos1': 'evaluation pos2'})

    evas1.reset_index(drop=True, inplace=True)
    evas2.reset_index(drop=True, inplace=True)

    merged_dataset = pd.merge(evas1, evas2, on='pos1', how='inner')
    merged_dataset.insert(0, 'index', range(1, len(merged_dataset) + 1))

    return merged_dataset
```

```

def merge2(evas1,evas2):
    evas1 = evas1[['pos2','evaluation pos2']]
    evas2 = evas2[['pos2','evaluation pos2']]

    evas1 = evas1.rename(columns={'pos2': 'pos1'})
    evas2 = evas2.rename(columns={'pos2': 'pos1'})

    evas1 = evas1.rename(columns={'evaluation pos2': 'evaluation pos1'})

    evas1.reset_index(drop=True, inplace=True)
    evas2.reset_index(drop=True, inplace=True)

    merged_dataset = pd.merge(evas1, evas2, on='pos1', how='inner')
    merged_dataset.insert(0, 'index', range(1, len(merged_dataset) + 1))

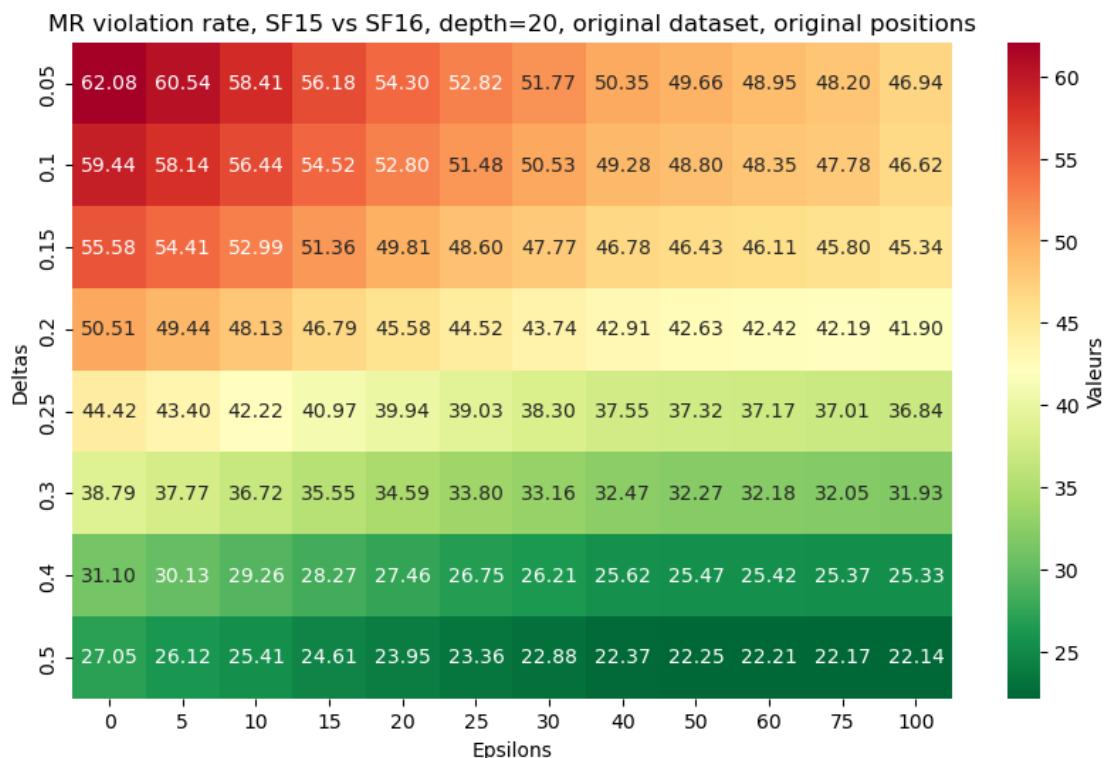
    return merged_dataset

```

Entire dataset

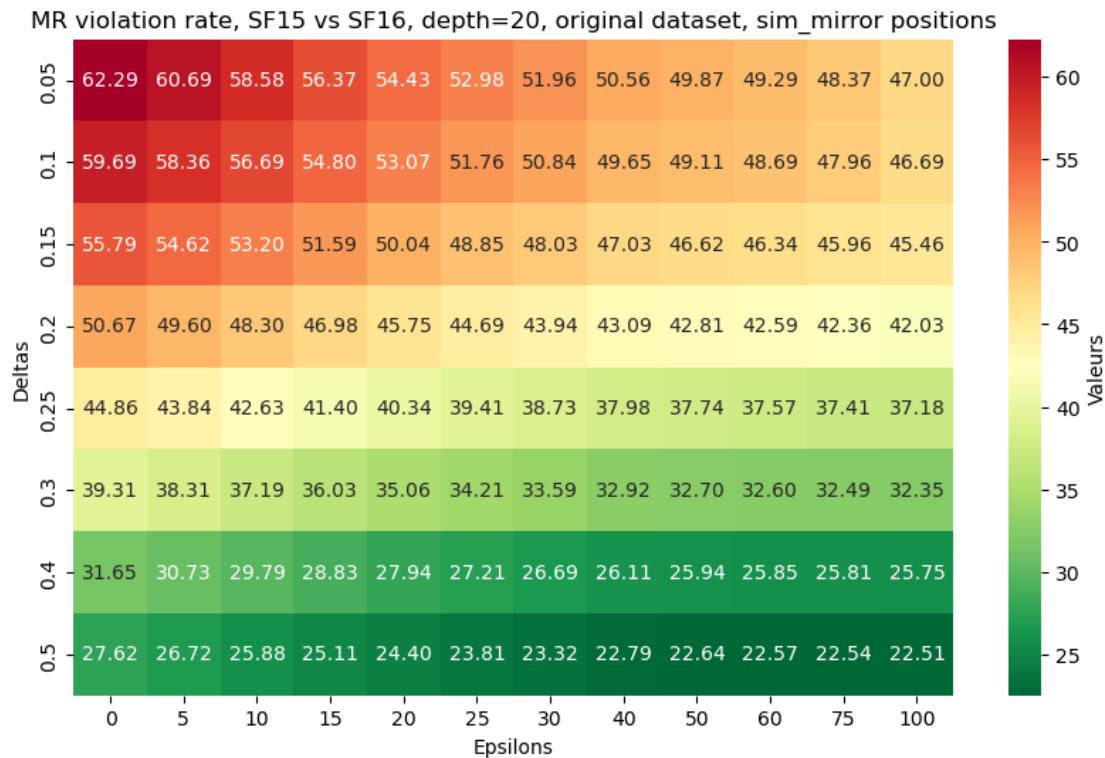
Original Positions

[]: tests(merge(evaluations50000_sim_mirror_d_20_v16,evaluations50000_sim_mirror_d_20),5,20,�, original dataset, original positions)



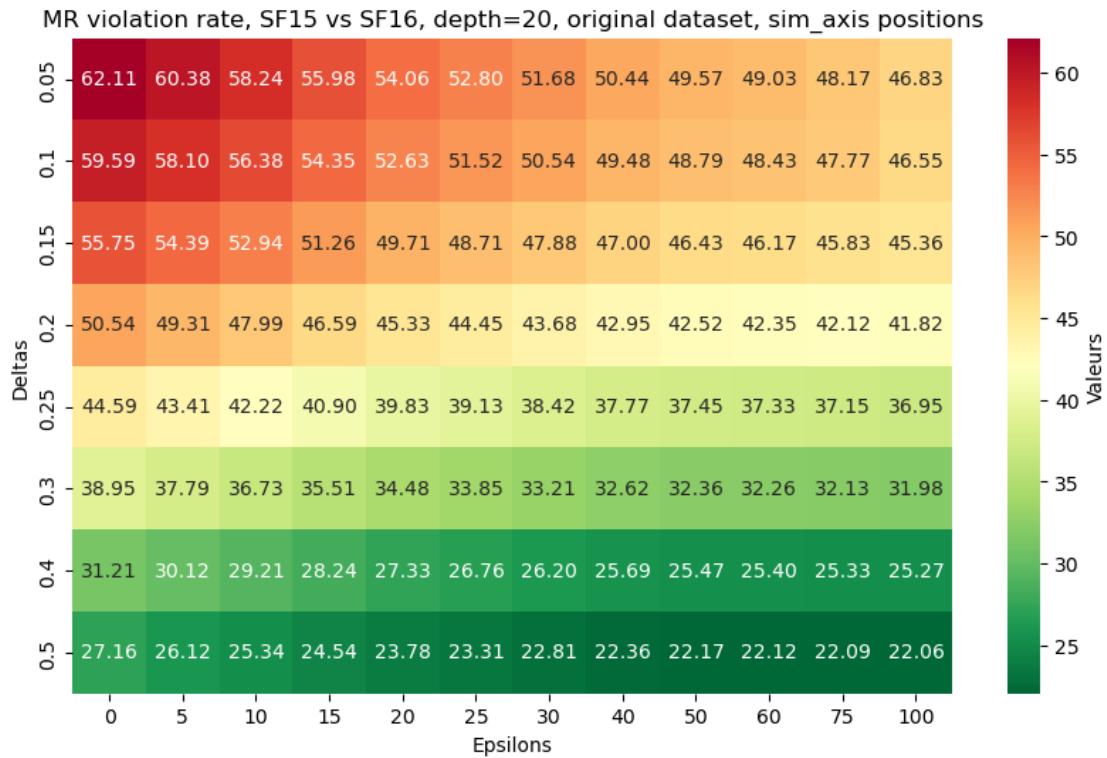
sim_mirror

```
[ ]: tests(merge2(evaluations50000_sim_mirror_d_20_v16,evaluations50000_sim_mirror_d_20),5,20,✉  
↪', original dataset, sim_mirror positions')
```



sim_axis

```
[ ]: tests(merge2(evaluations50000_sim_axis_d_20_v16,evaluations50000_sim_axis_d_20),5,20,✉  
↪', original dataset, sim_axis positions')
```

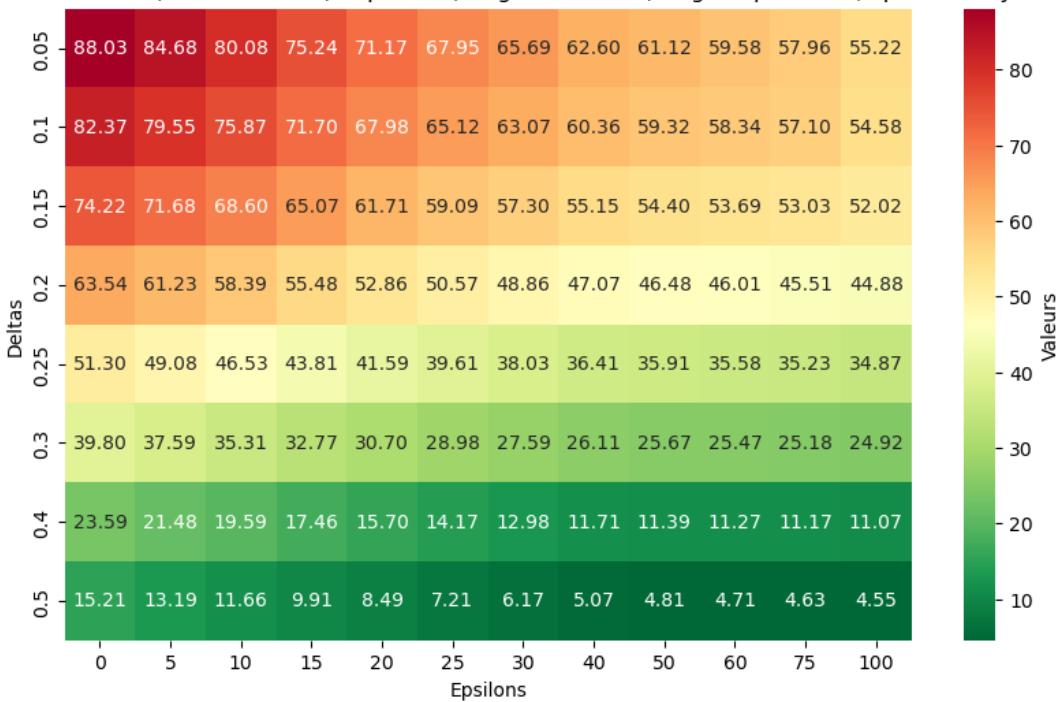


Splitting, positions with evaluations in centipawns only

Original Positions

```
[ ]: tests(merge(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[0],split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[1],original dataset, original positions, cp evas only'))
```

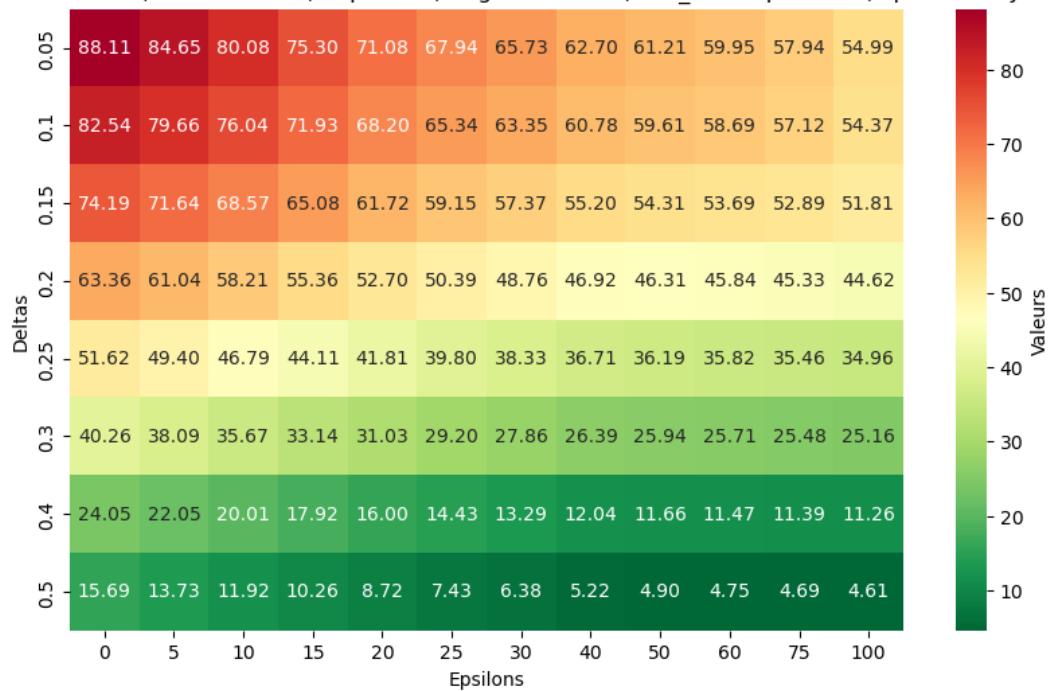
MR violation rate, SF15 vs SF16, depth=20, original dataset, original positions, cp evas only



sim_mirror

```
[ ]: tests(merge2(split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[0],split_mate_cp(evaluations50000_sim_mirror_d_20_v16)[1]), original dataset, sim_mirror positions, cp evas only')
```

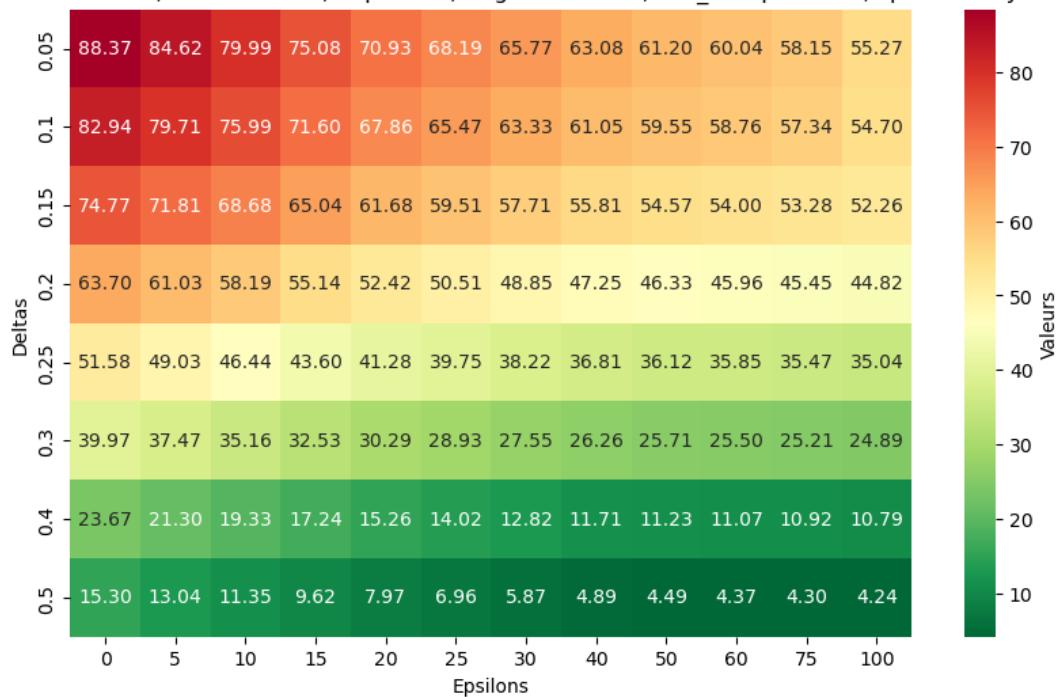
MR violation rate, SF15 vs SF16, depth=20, original dataset, sim_mirror positions, cp evas only



sim_axis

```
[ ]: tests(merge2(split_mate_cp(evaluations50000_sim_axis_d_20_v16)[0],split_mate_cp(evaluations50000_sim_axis_d_20_v16)[1],original dataset, sim_axis positions, cp evas only'))
```

MR violation rate, SF15 vs SF16, depth=20, original dataset, sim_axis positions, cp evas only

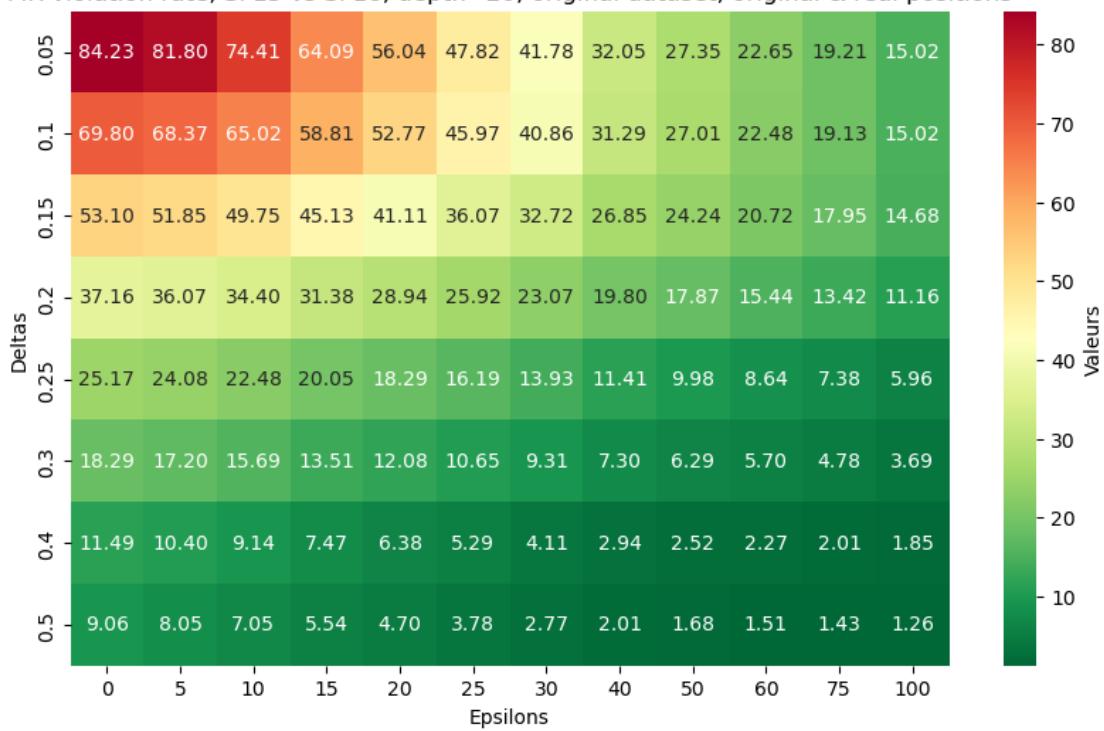


Real positions

Original Positions

```
[ ]: tests(merge(evaluations50000_sim_mirror_d_20_v16_real, evaluations50000_sim_mirror_d_20_real),
      ↵', original dataset, original & real positions')
```

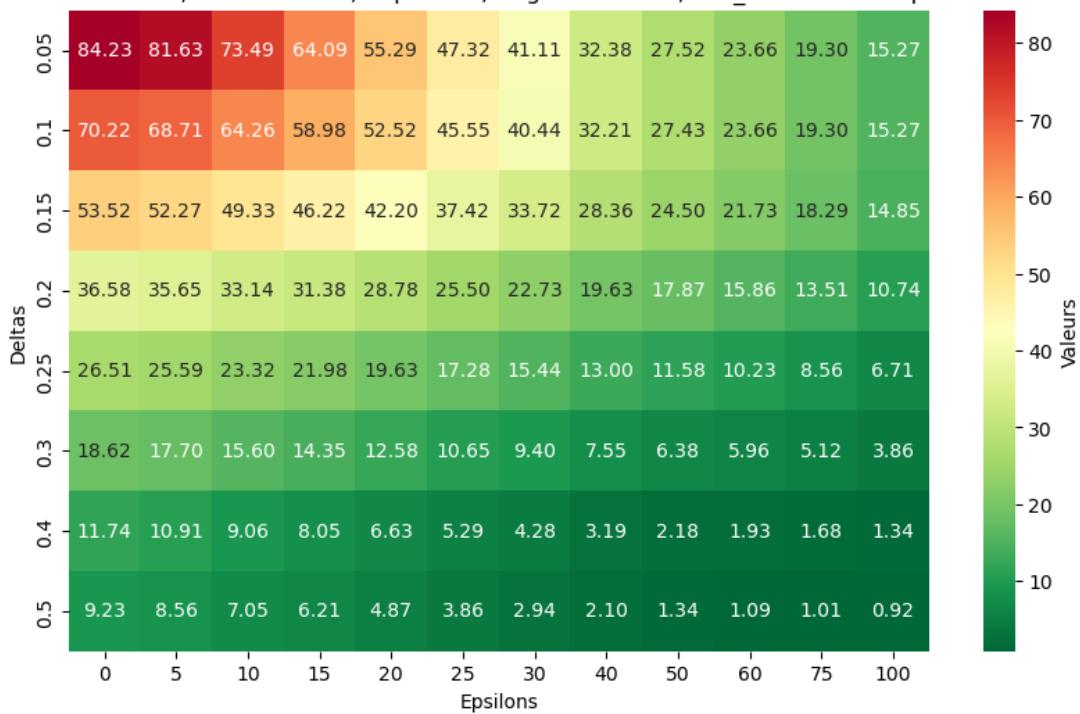
MR violation rate, SF15 vs SF16, depth=20, original dataset, original & real positions



sim_mirror

```
[ ]: tests(merge2(evaluations50000_sim_mirror_d_20_v16_real,evaluations50000_sim_mirror_d_20_real),
      ↵', original dataset, sim_mirror on real positions')
```

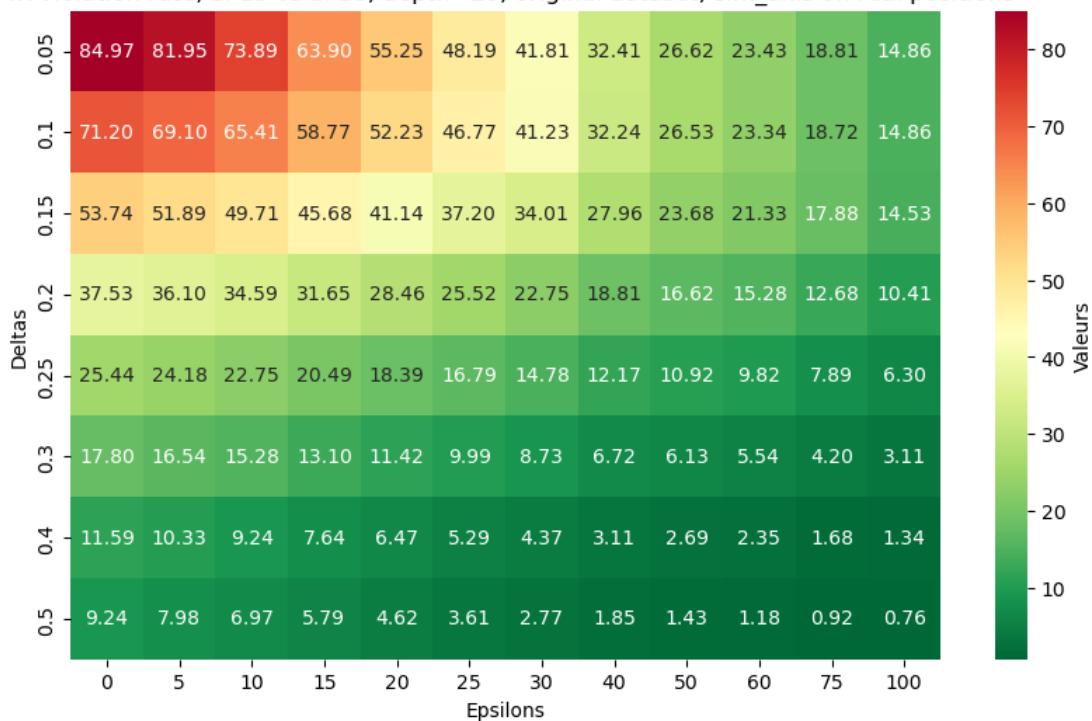
MR violation rate, SF15 vs SF16, depth=20, original dataset, sim_mirror on real positions



sim_axis

```
[ ]: tests(merge2(evaluations50000_sim_axis_d_20_v16_real,evaluations50000_sim_axis_d_20_real),5,20
↳ ' , original dataset, sim_axis on real positions')
```

MR violation rate, SF15 vs SF16, depth=20, original dataset, sim_axis on real positions



0.10.7 Conclusions with Stockfish 16

This newer version shows slightly better results.

0.11 Tests with positions from lichess, Stockfish 15

We took games from the lichess database, and picked position at move 10, 20, 30 and 40. We then split the positions by elo, and also added Carlsen's games. We are using Stockfish 15 there. Moreover, we'll also show grids for positions where castling isn't possible anymore ; the original paper decided to remove the ability to castle on every positions, but if we does that on real positions, even if they still look real, some may actually not be anymore.

0.11.1 sim_mirror

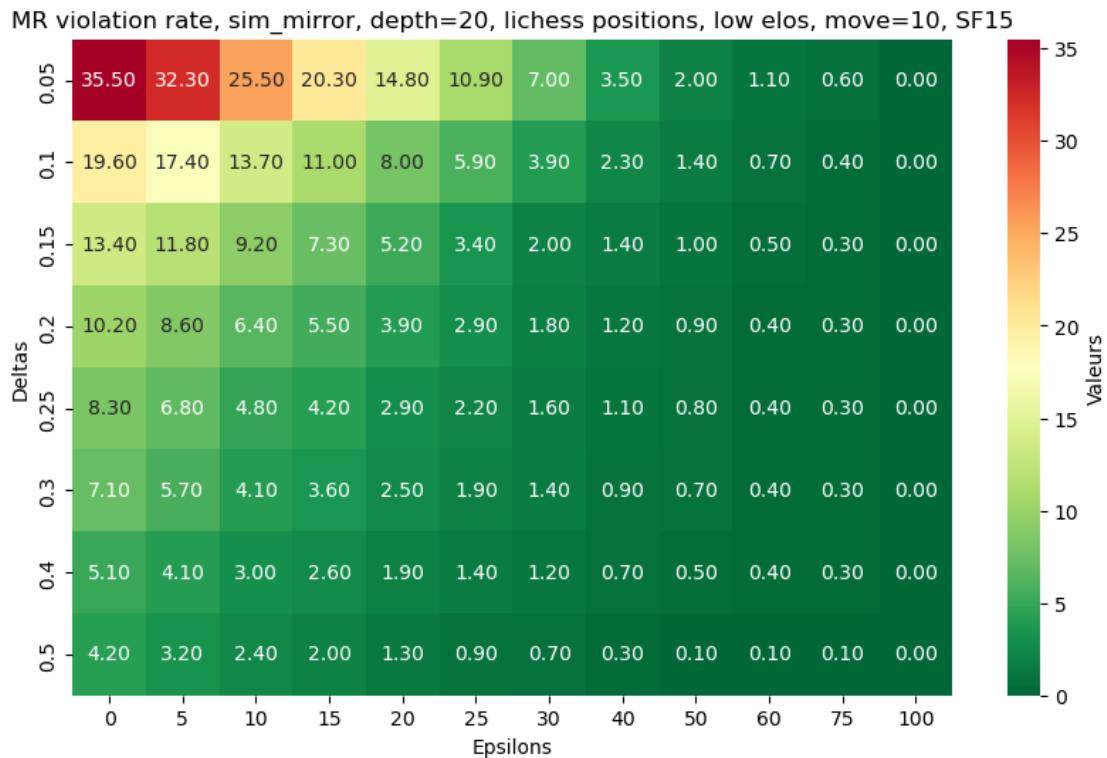
Openings (move=10)

Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-10sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs, 0, 20, ', lichess positions, low elos, move=10, SF15')
```

```
move = 10
```



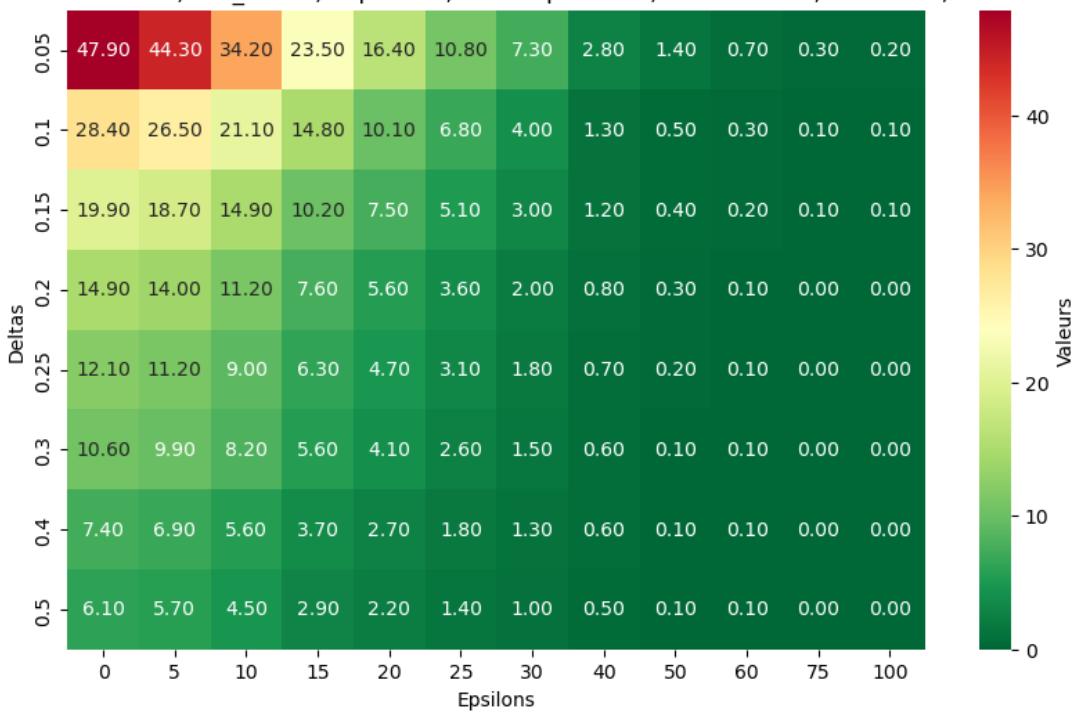
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1500-2000-10sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, ', lichess positions, medium elos, move=10, SF15')
```

```
move = 10
```

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=10, SF15

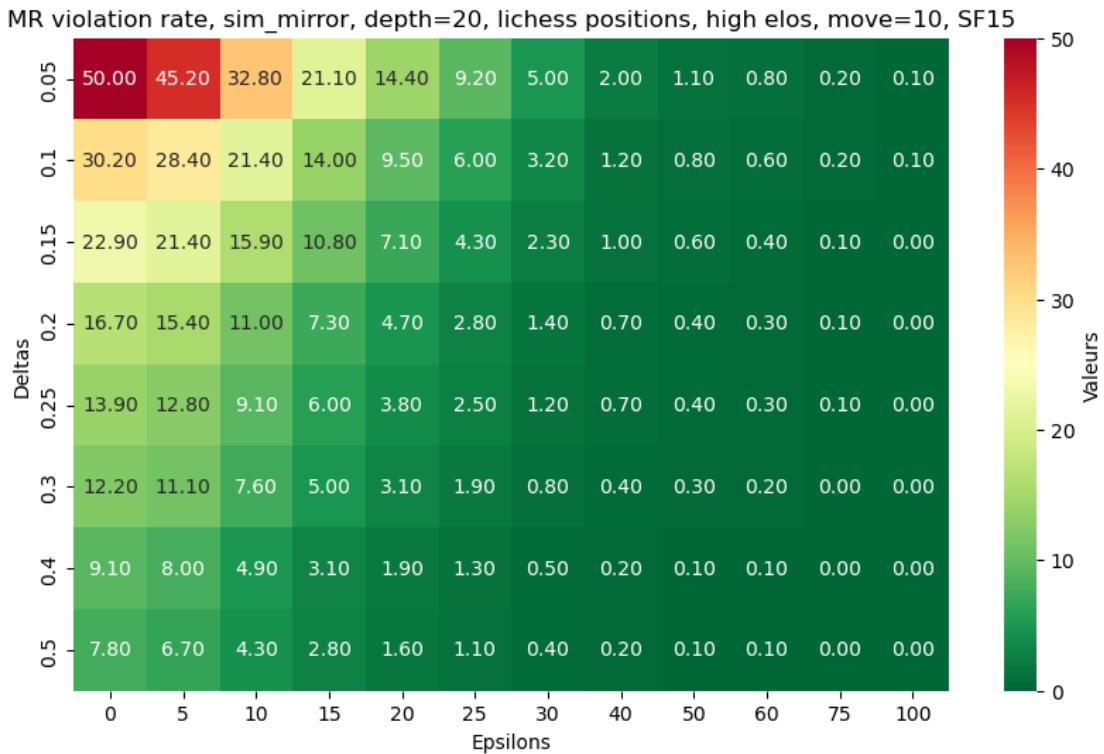


Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, ' lichess positions, high elos, move=10, SF15')
```

move = 10



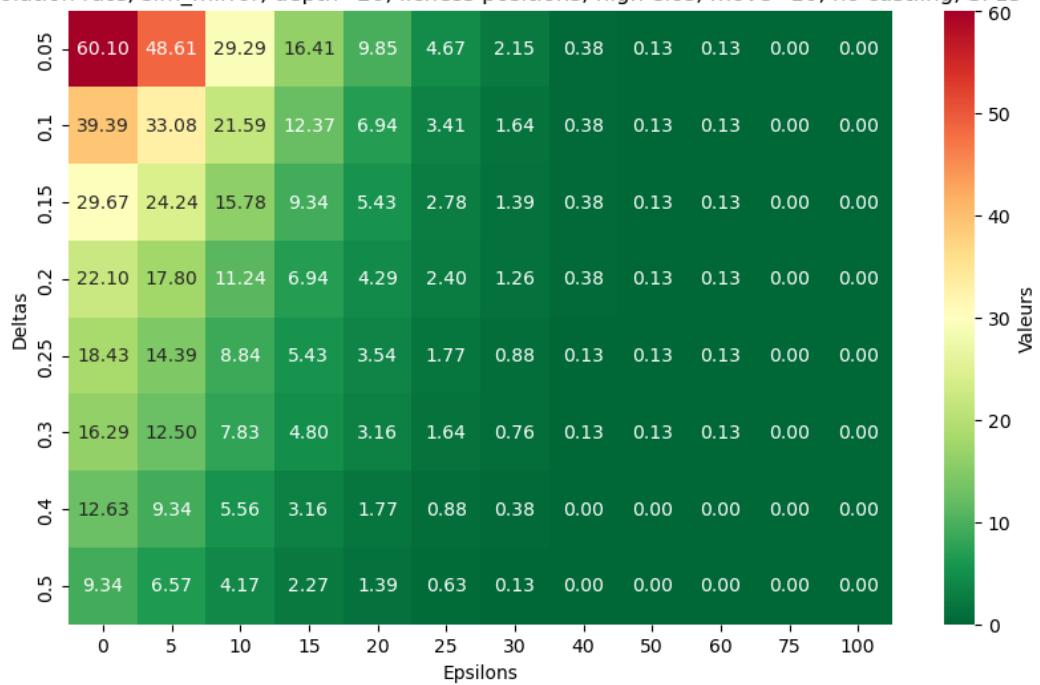
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10_nocastlingsim_mirror_d_20.
    ↵pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, ', lichess positions, high elos, move=10, no castling, SF15')
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=10, no castling, SF15

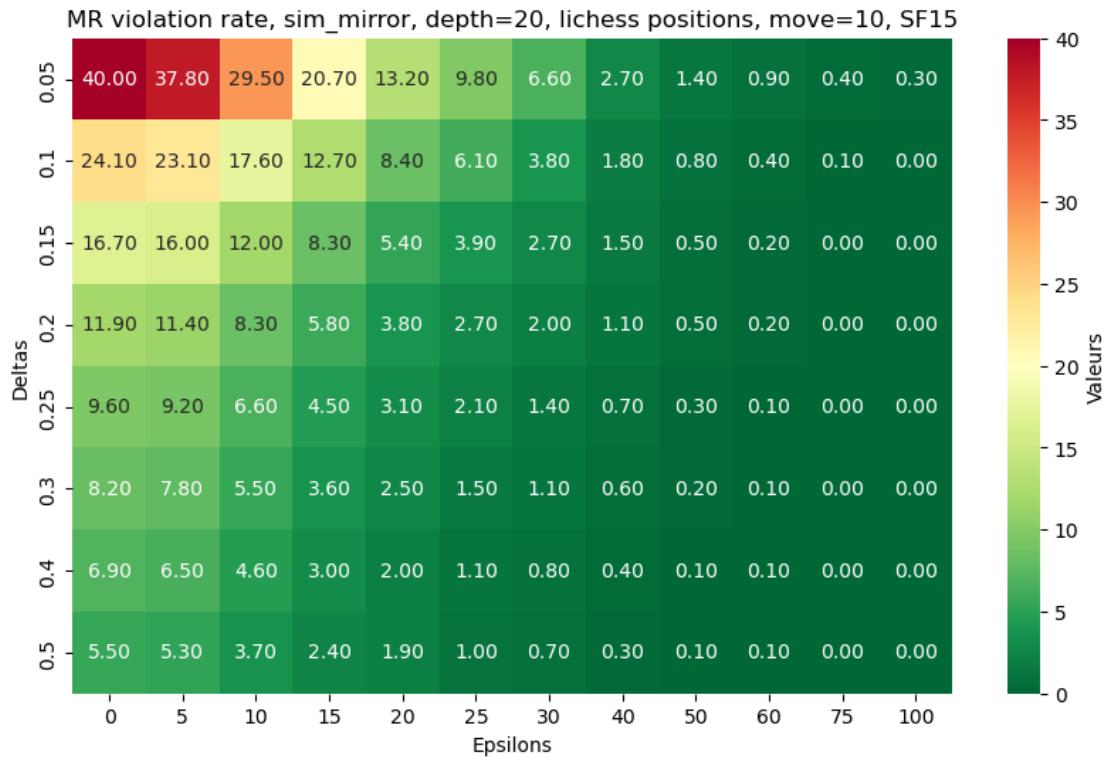


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, ' lichess positions, move=10, SF15')
```

move = 10



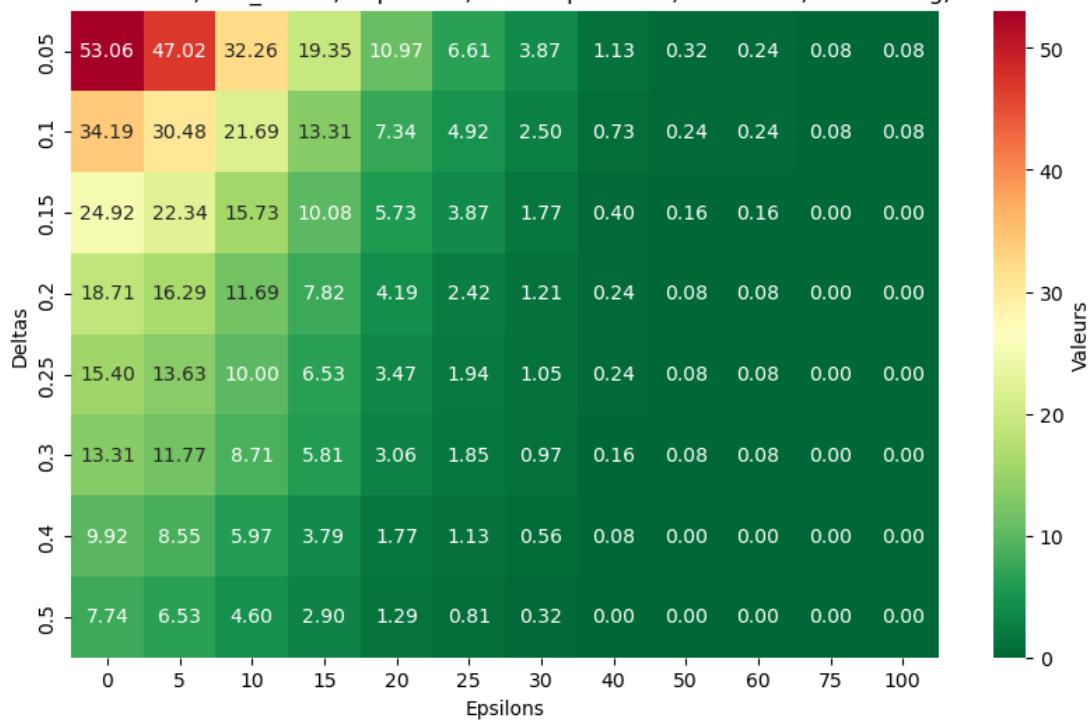
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10_nocastlingsim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, ' lichess positions, move=10, no castling, SF15')
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, move=10, no castling, SF15

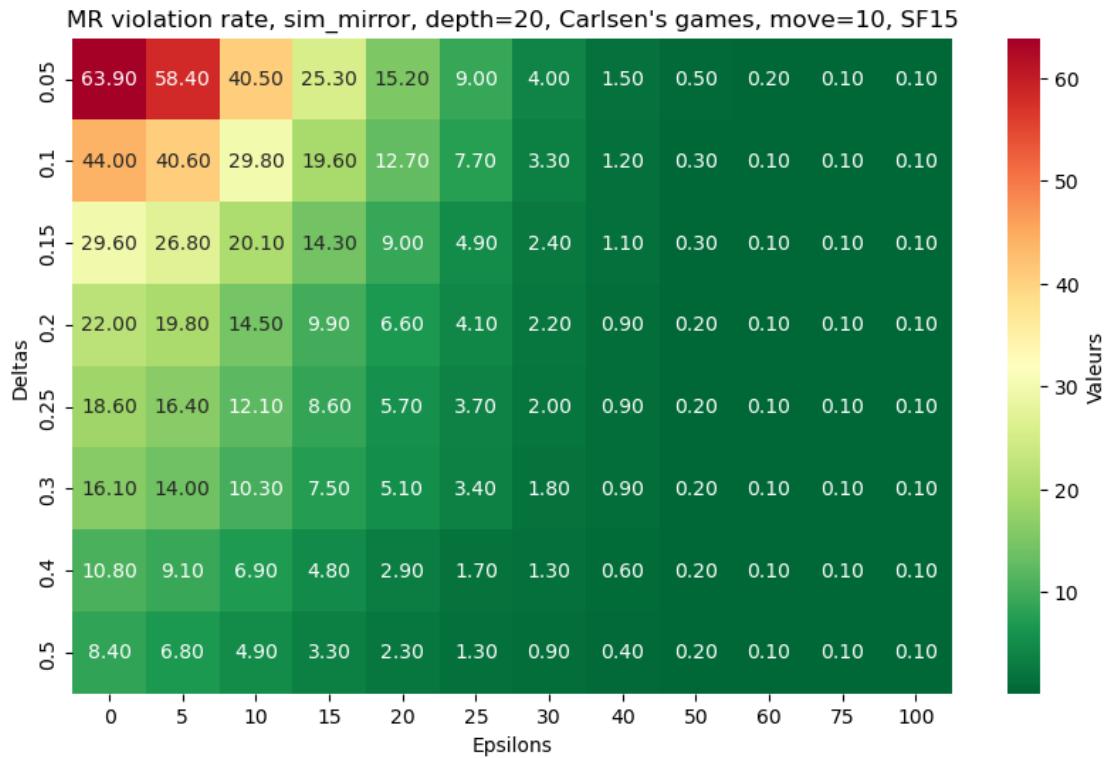


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, " Carlsen's games, move=10, SF15")
```

move = 10



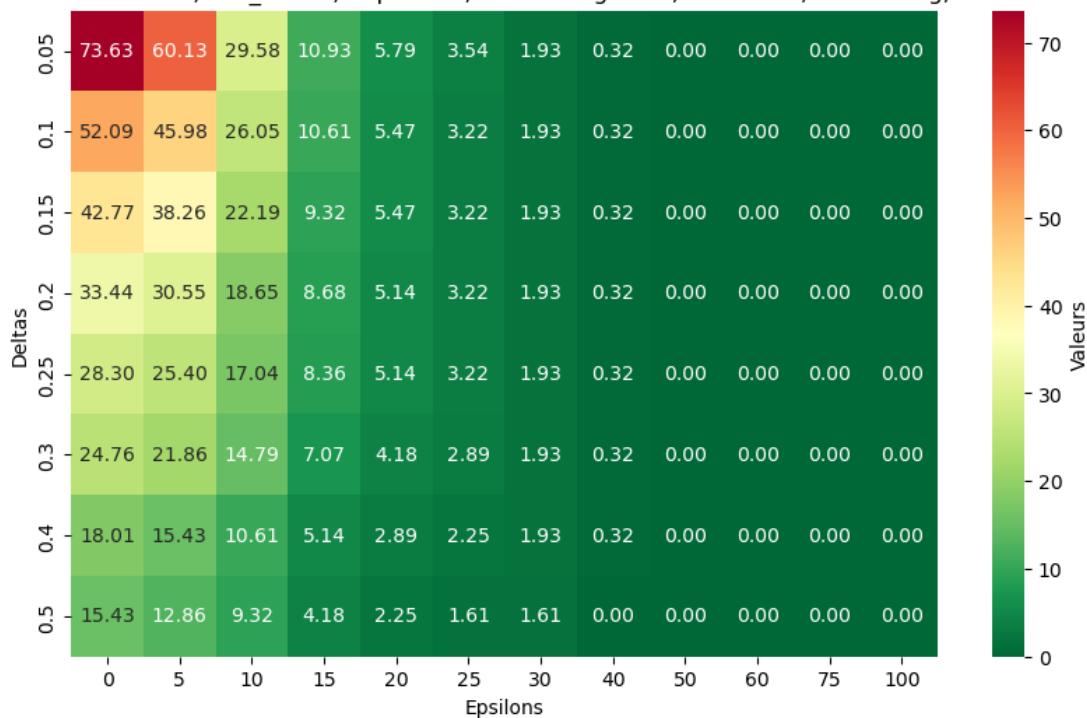
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10_nocastlingsim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs01510 = pickle.load(file)

      print('move = 10')
      tests(evs01510, 0, 20, "Carlsen's games, move=10, no castling, SF15")
```

move = 10

MR violation rate, sim_mirror, depth=20, Carlsen's games, move=10, no castling, SF15



Middlegames (move=20)

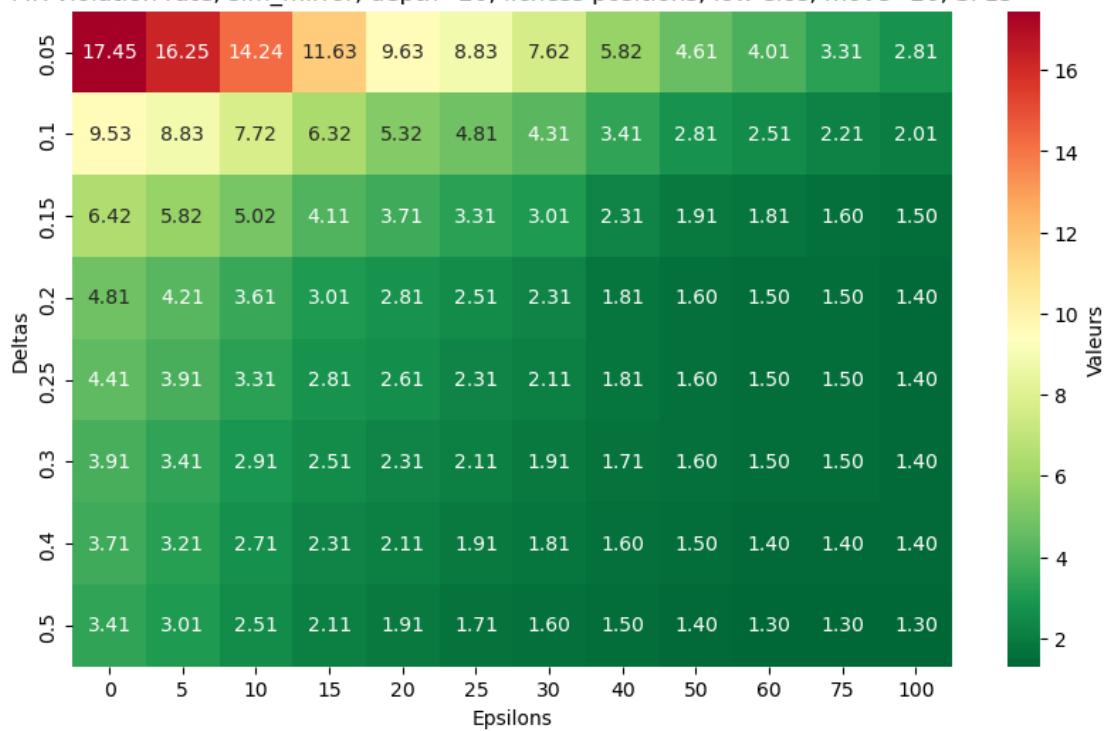
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-20sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 0, 20, ' lichess positions, low elos, move=20, SF15')
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=20, SF15



Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-20sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs,0,20, ' lichess positions, medium elos, move=20, SF15')
```

move = 20



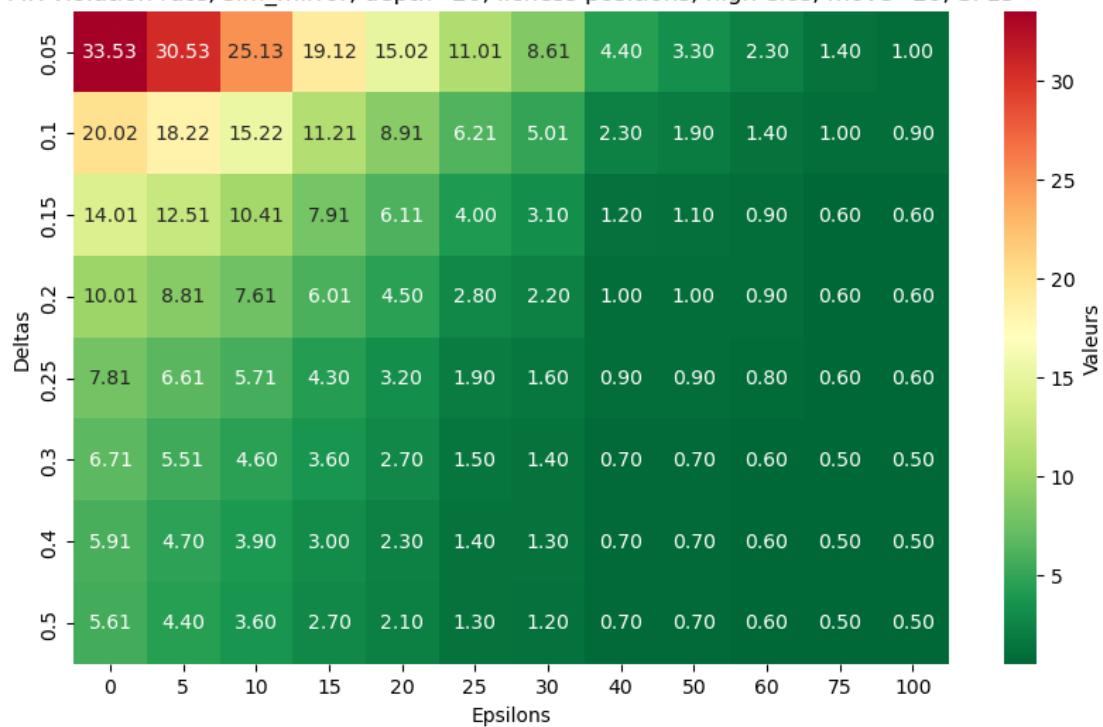
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, ' lichess positions, high elos, move=20, SF15')
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=20, SF15



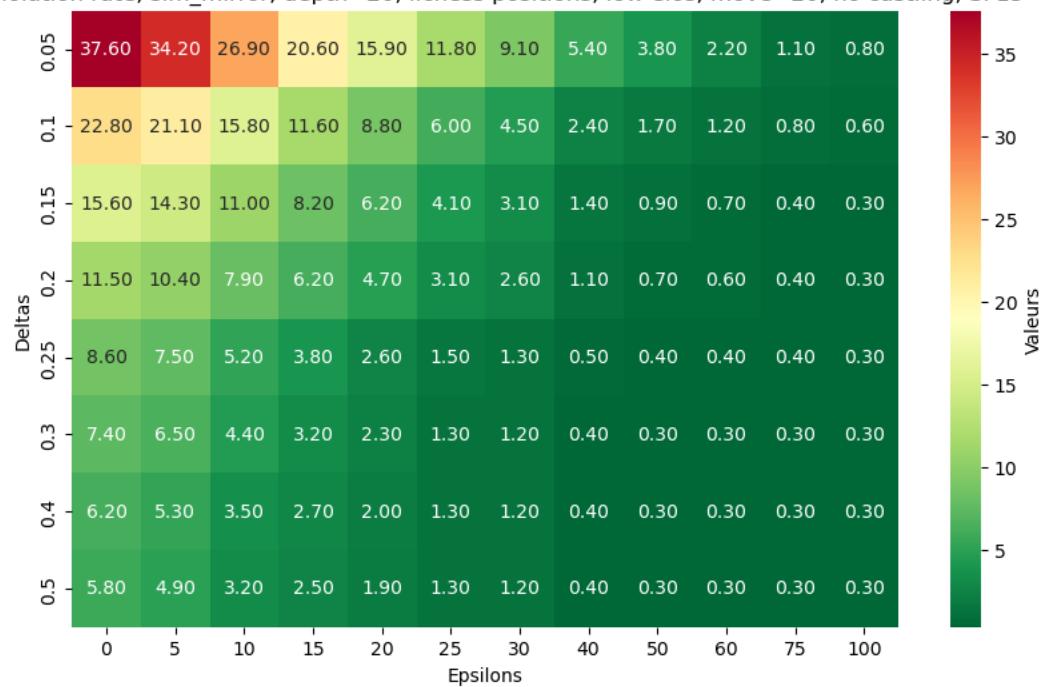
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20_nocastlingsim_mirror_d_20.
    ↪pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, ' lichess positions, low elos, move=20, no castling, SF15')
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=20, no castling, SF15

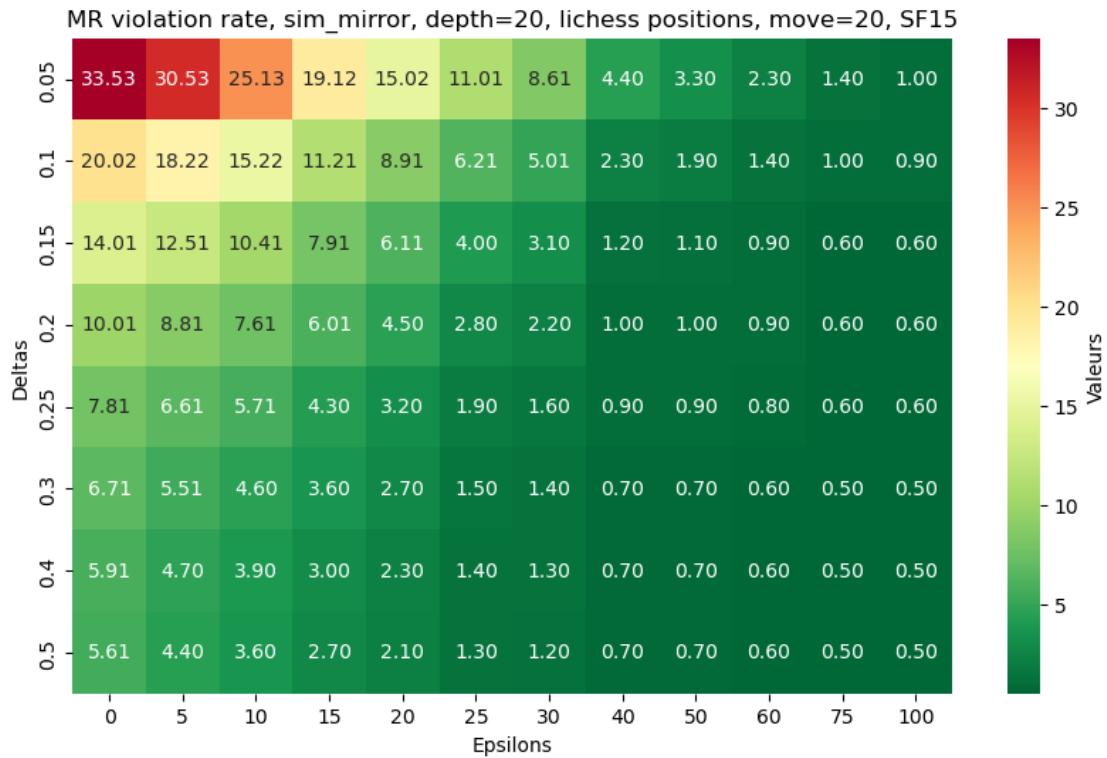


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, ' lichess positions, move=20, SF15')
```

move = 20



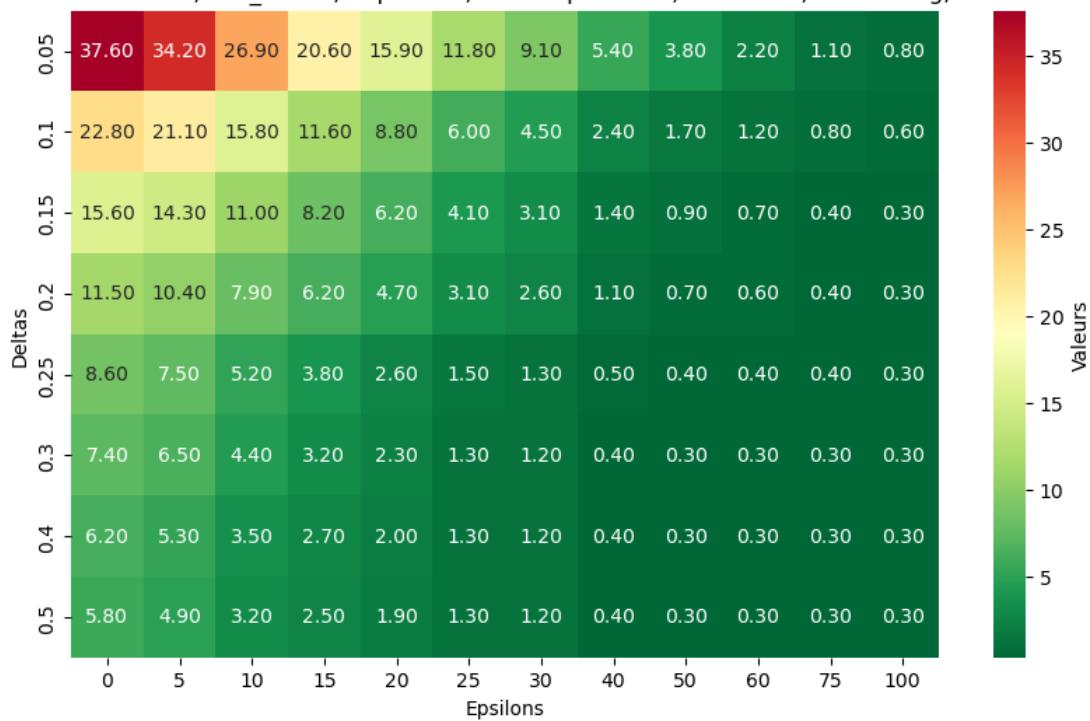
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20_nocastlingsim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 0, 20, ' lichess positions, move=20, no castling, SF15')
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, move=20, no castling, SF15

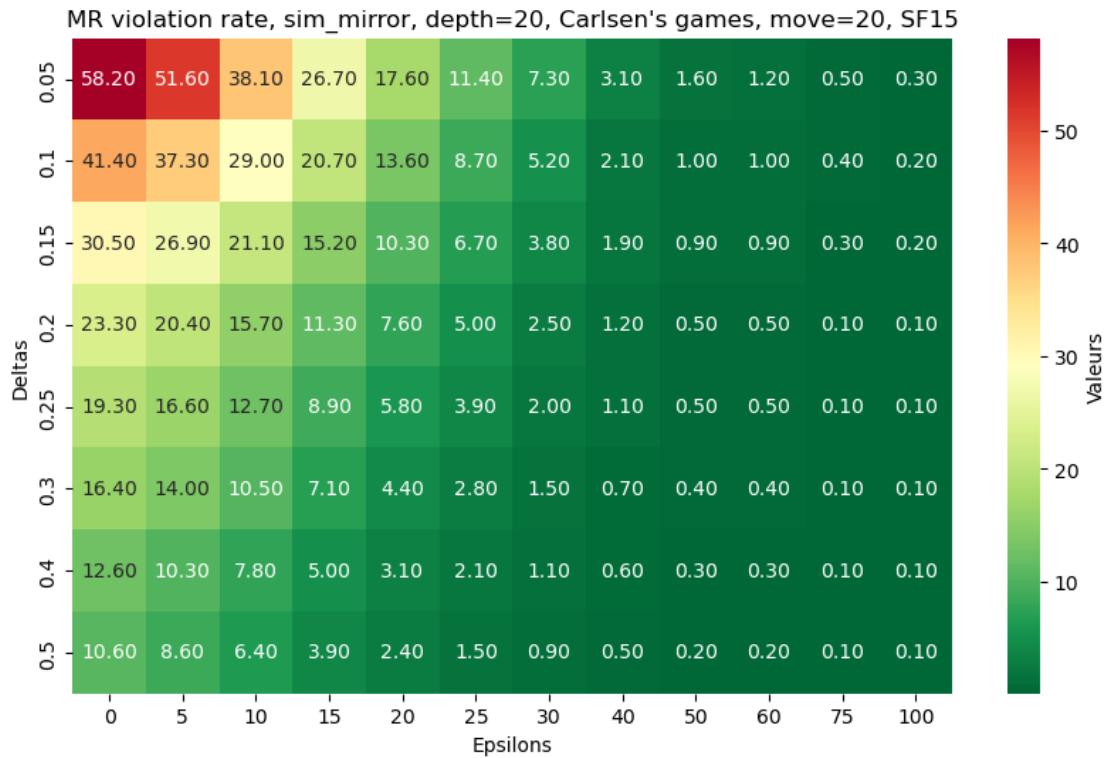


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, " Carlsen's games, move=20, SF15")
```

move = 20

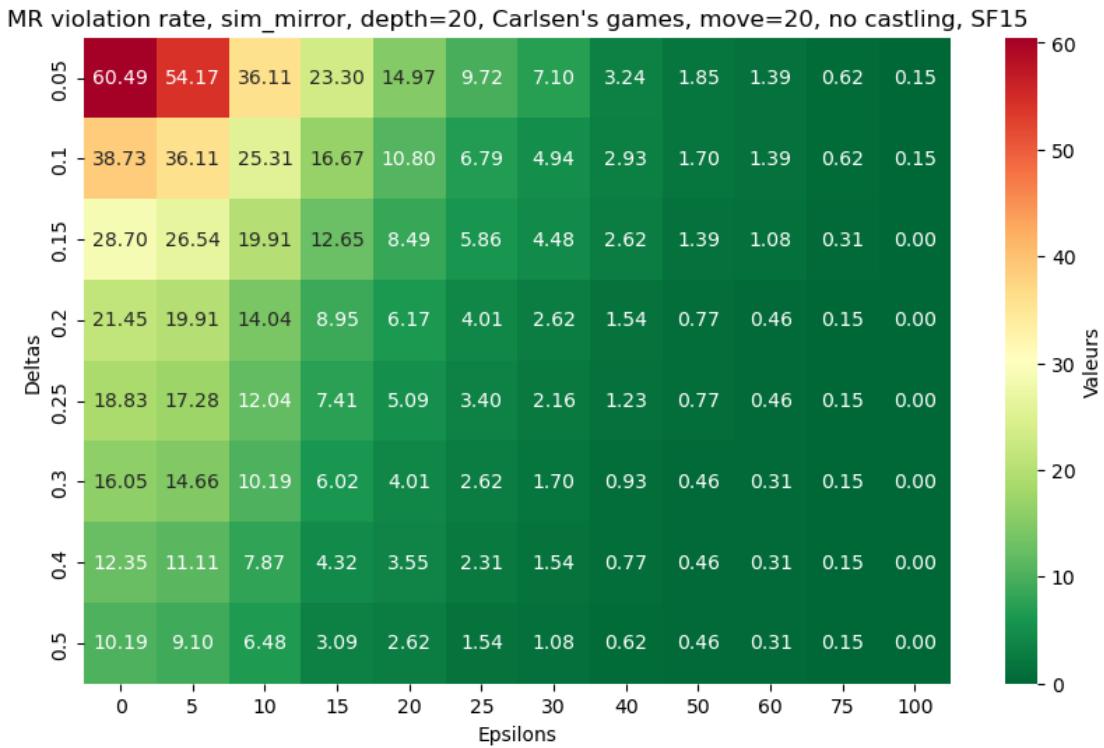


Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20_nocastlingsim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs01520 = pickle.load(file)

      print('move = 20')
      tests(evs01520, 0, 20, "Carlsen's games, move=20, no castling, SF15")
```

move = 20



Middle/Endgames (move=30)

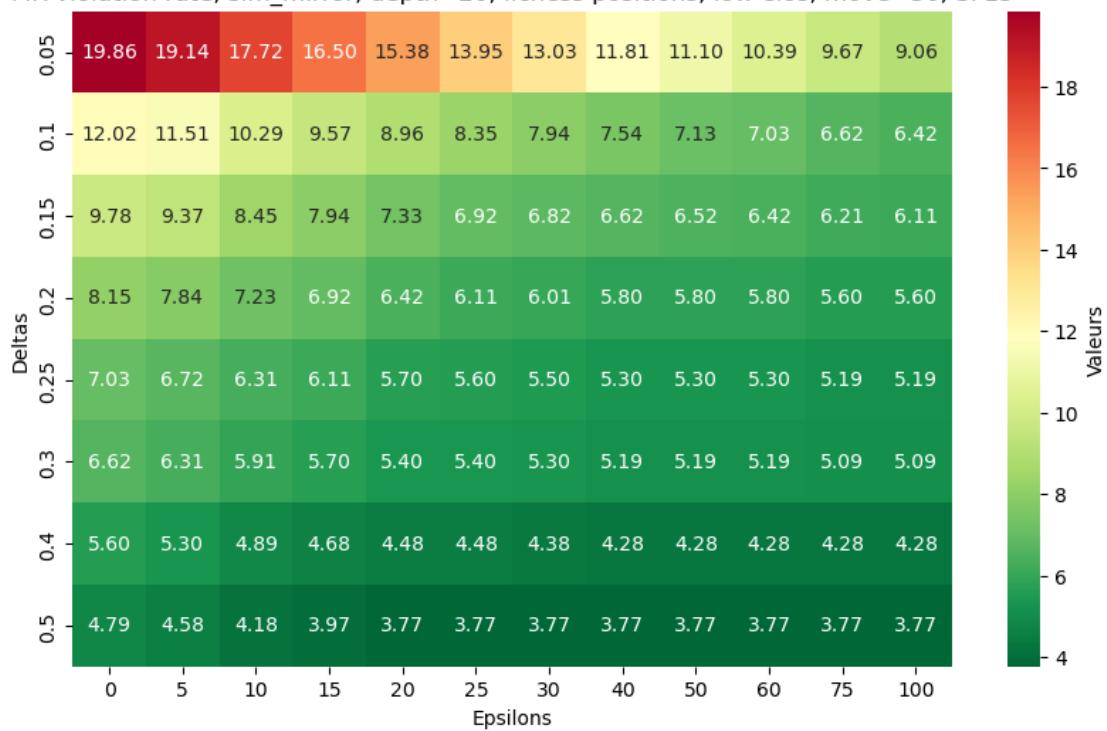
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-30sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 0, 20, " lichess positions, low elos, move=30, SF15")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=30, SF15

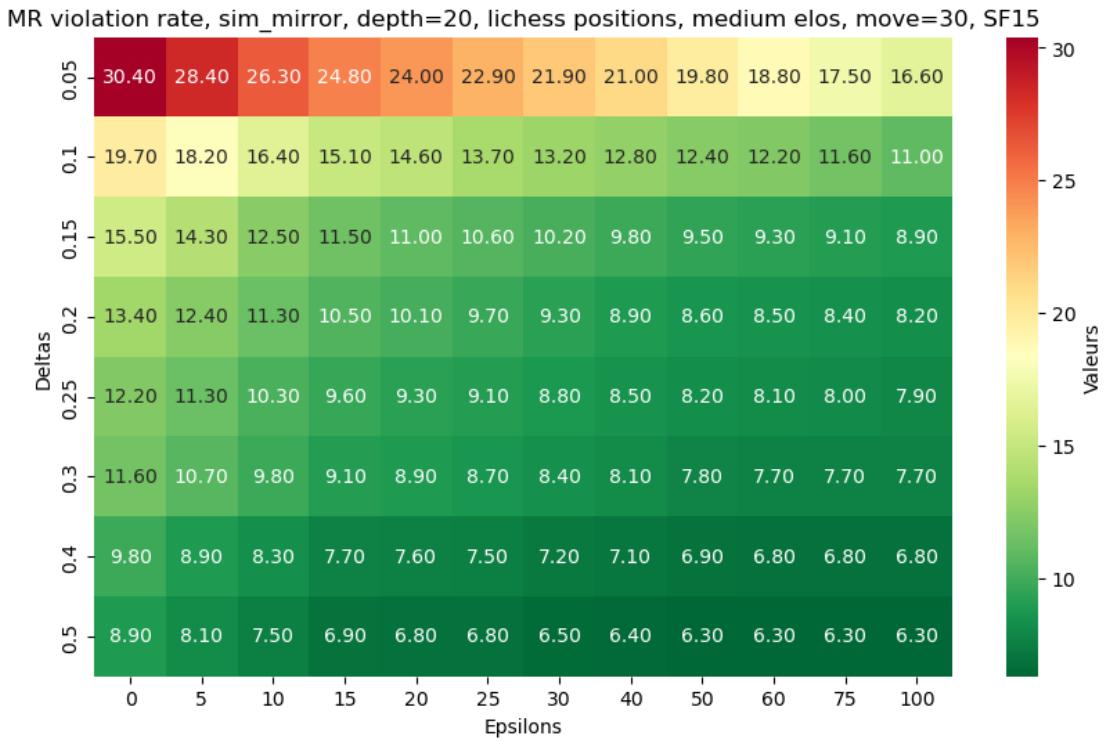


Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-30sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs,0,20, " lichess positions, medium elos, move=30, SF15")
```

move = 30



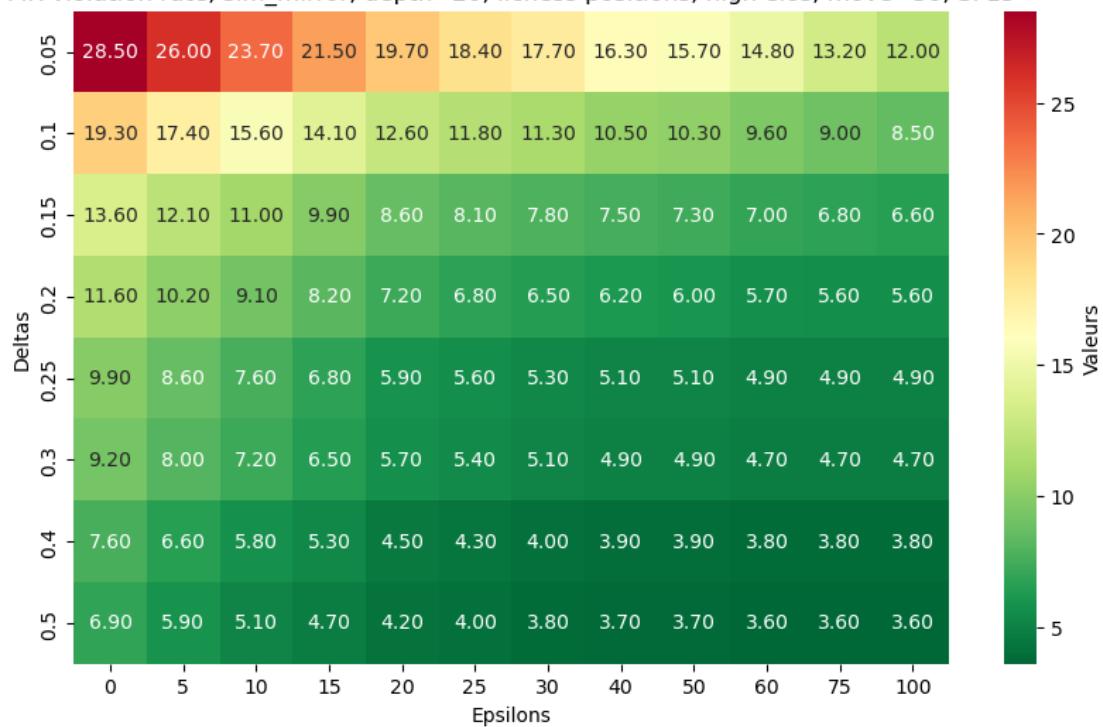
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, high elos, move=30, SF15")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=30, SF15



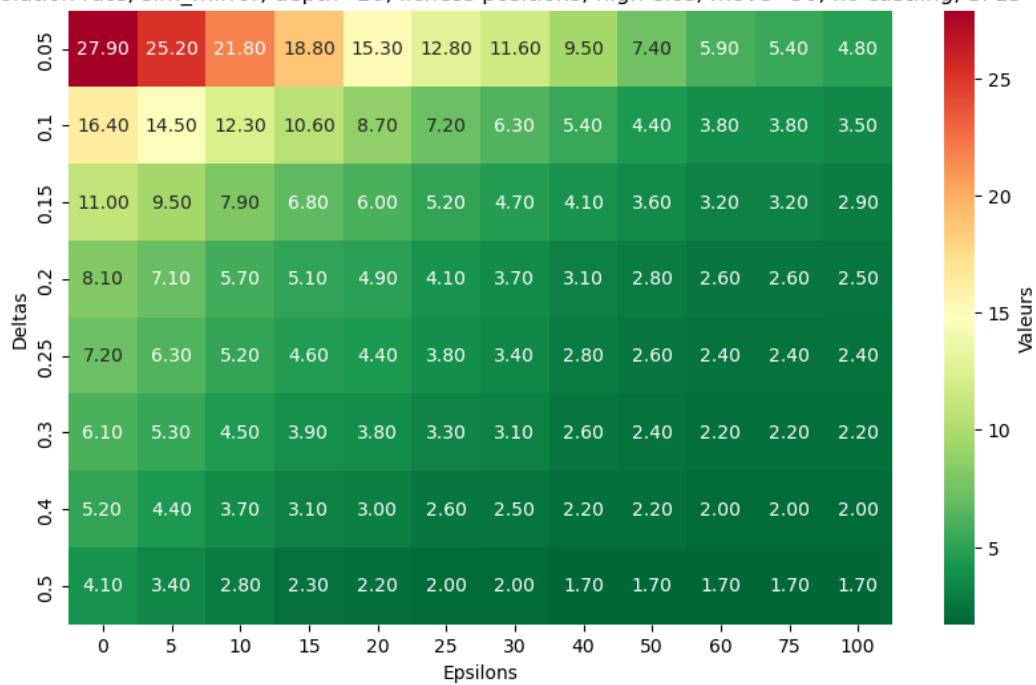
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30_nocastlingsim_mirror_d_20.
    ↪pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, high elos, move=30, no castling, SF15")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=30, no castling, SF15



Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, move=30, SF15")
```

move = 30



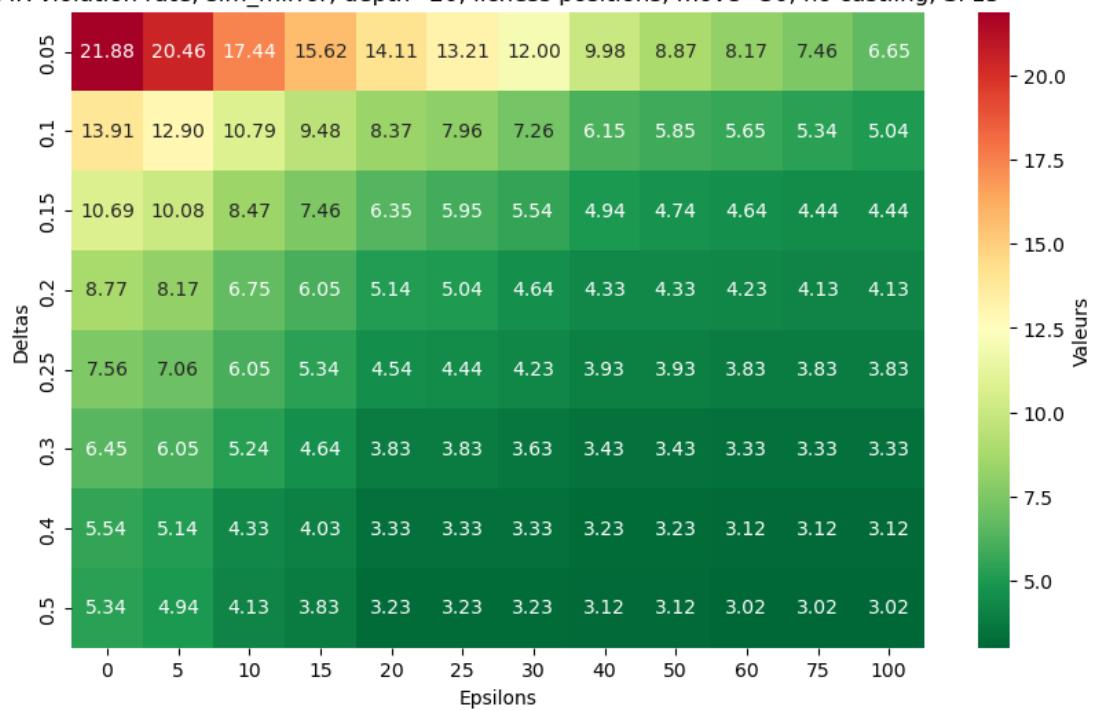
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30_nocastlingsim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, move=30, no castling, SF15")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, move=30, no castling, SF15

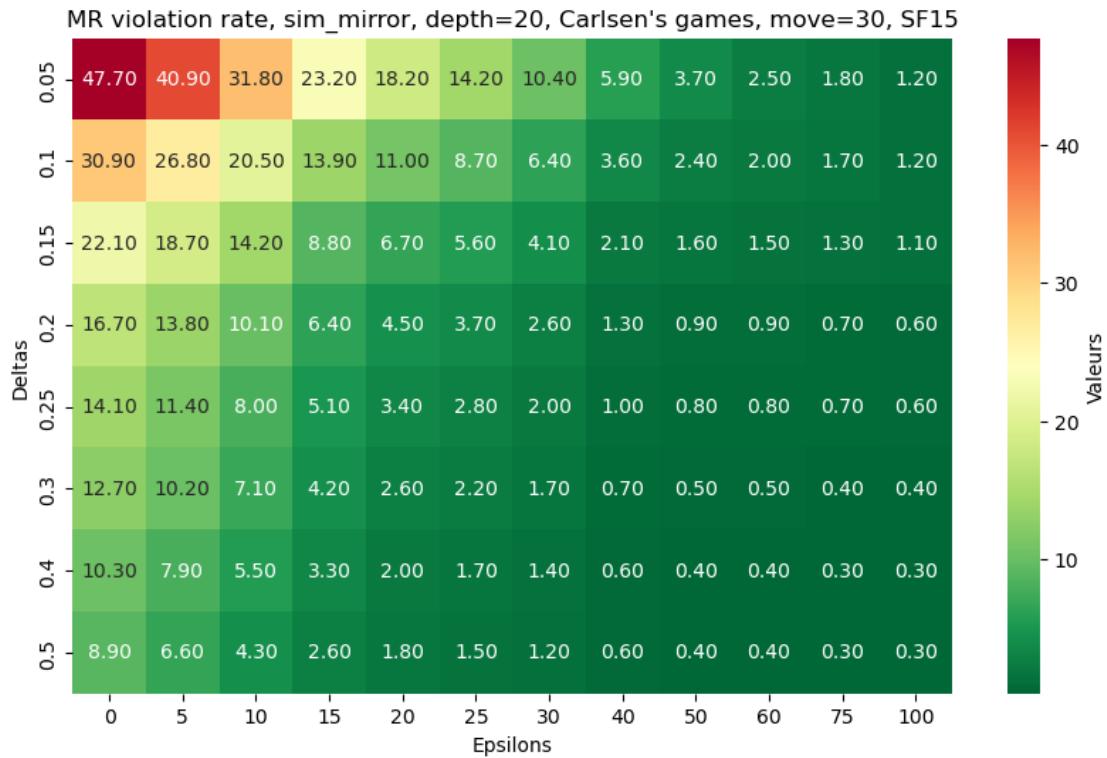


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " Carlsen's games, move=30, SF15")
```

move = 30

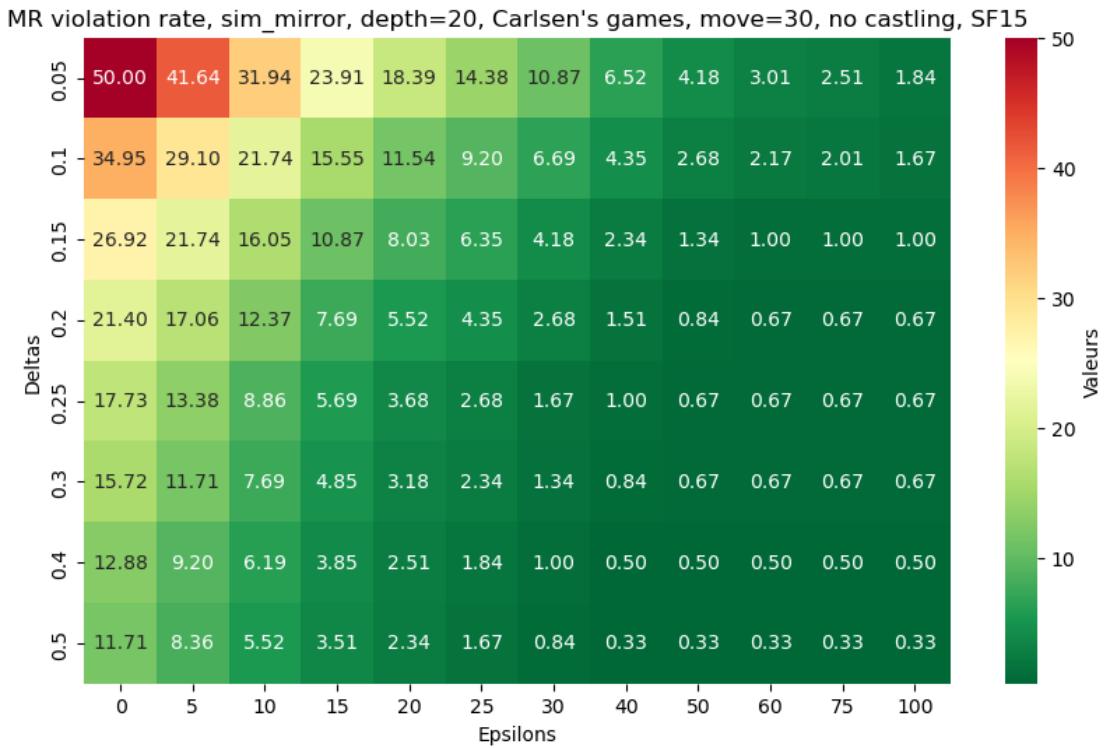


Carlsen's games, no castling available

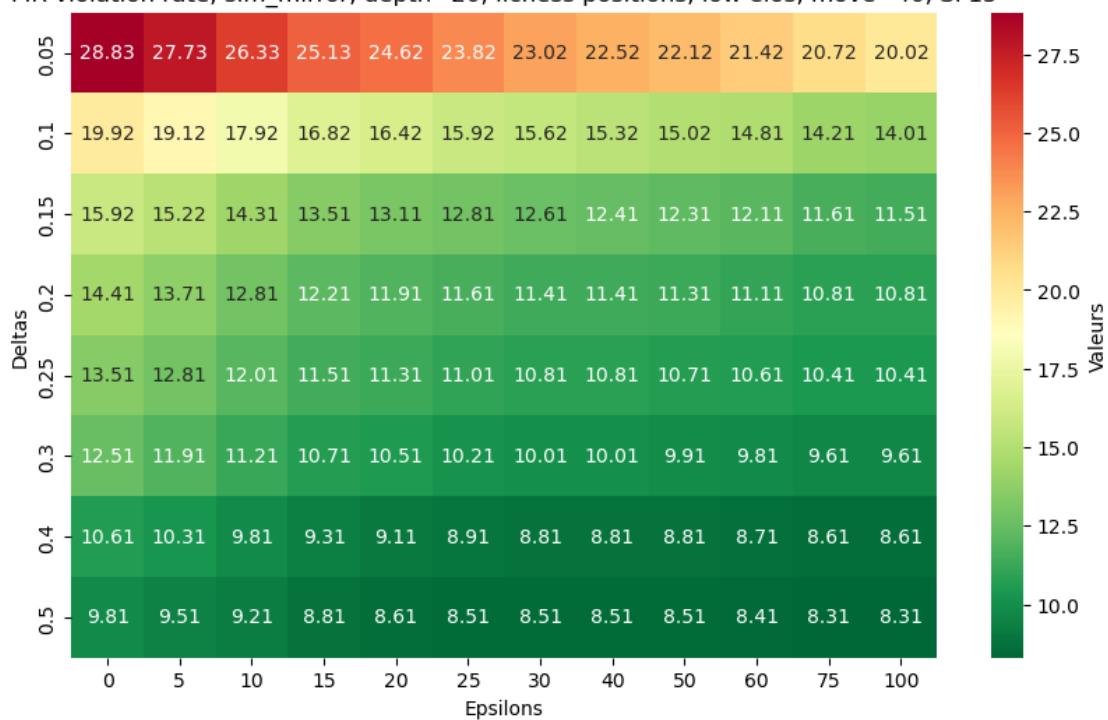
```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30_nocastlingsim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs01530 = pickle.load(file)

      print('move = 30')
      tests(evs01530, 0, 20, "Carlsen's games, move=30, no castling, SF15")
```

move = 30



MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=40, SF15



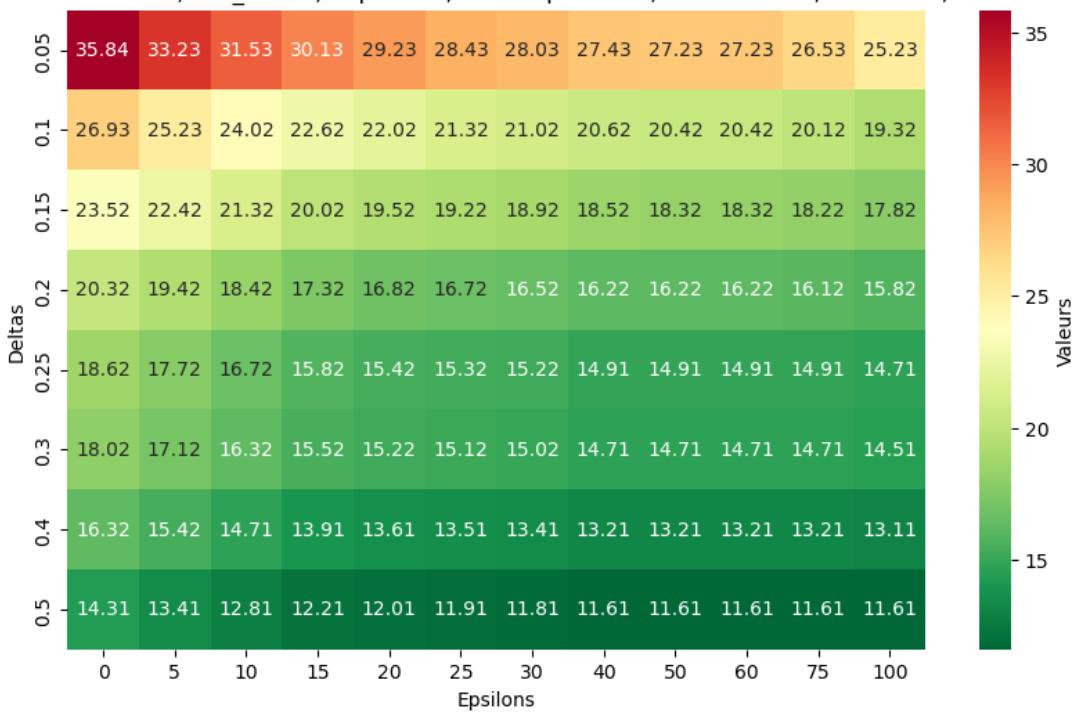
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1500-2000-40sim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 0, 20, " lichess positions, medium elos, move=40, SF15")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=40, SF15



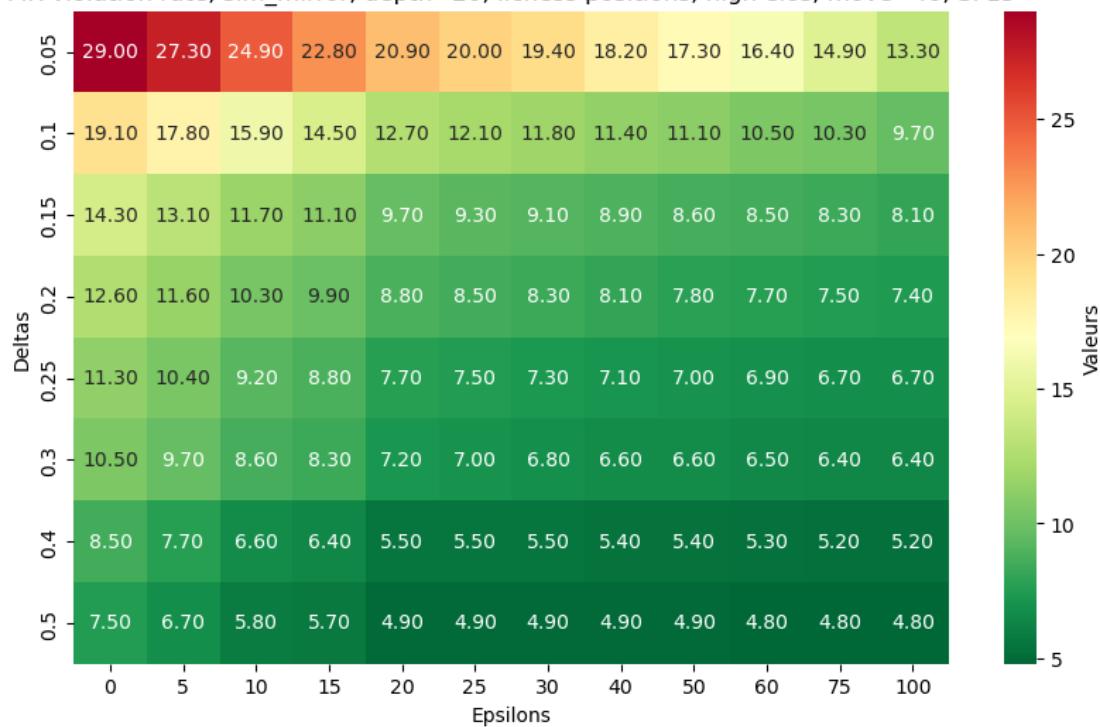
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " lichess positions, high elos, move=40, SF15")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=40, SF15



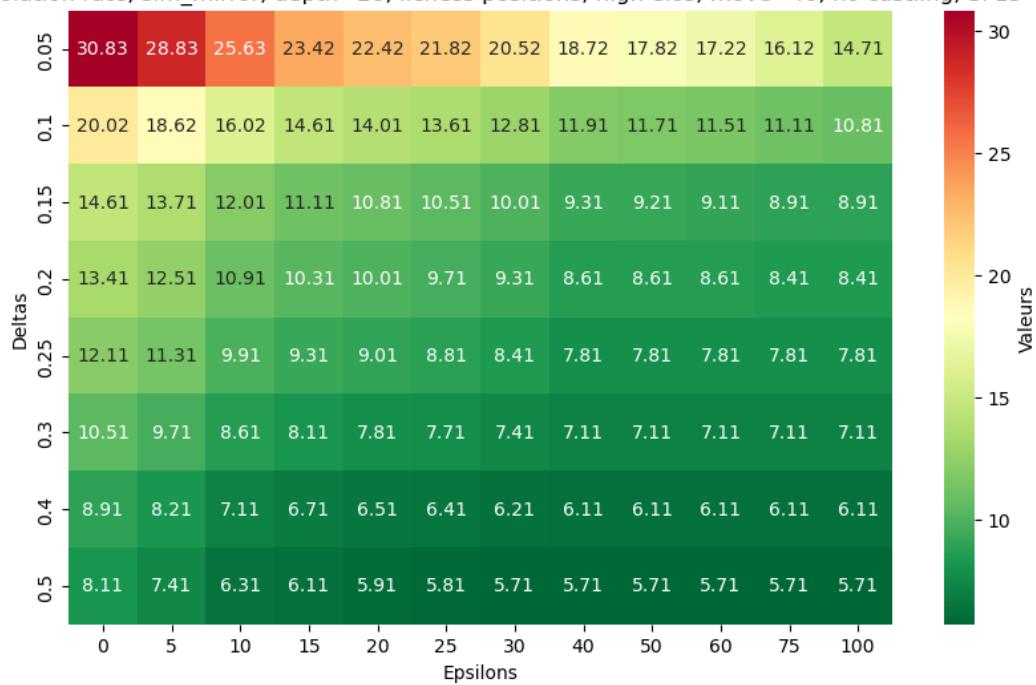
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40_nocastlingsim_mirror_d_20.
    ↵pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " lichess positions, high elos, move=40, no castling, SF15")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=40, no castling, SF15

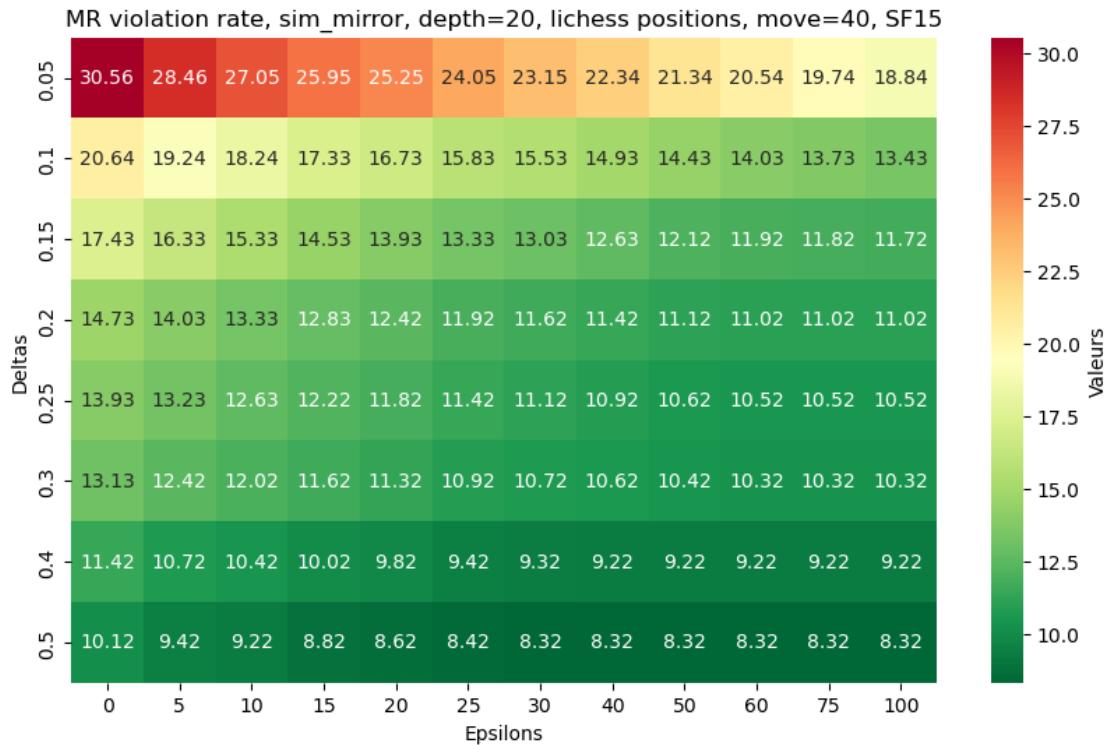


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " lichess positions, move=40, SF15")
```

move = 40

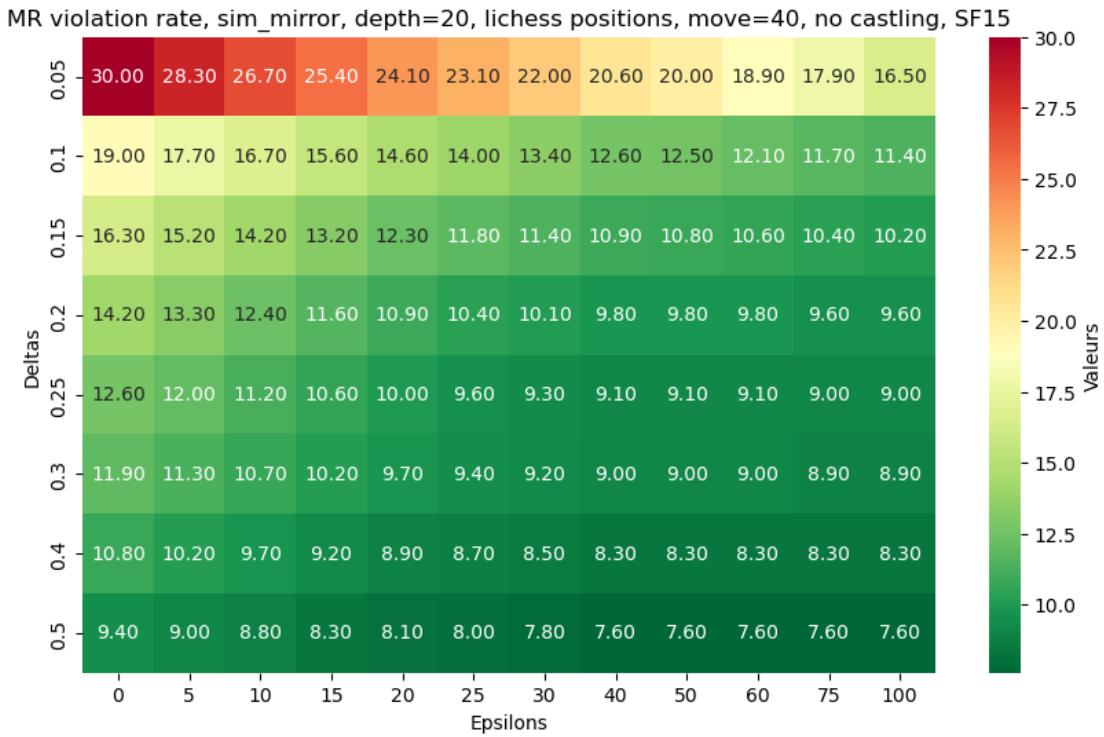


Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40_nocastlingsim_mirror_d_20.
    ↪pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " lichess positions, move=40, no castling, SF15")
```

move = 40

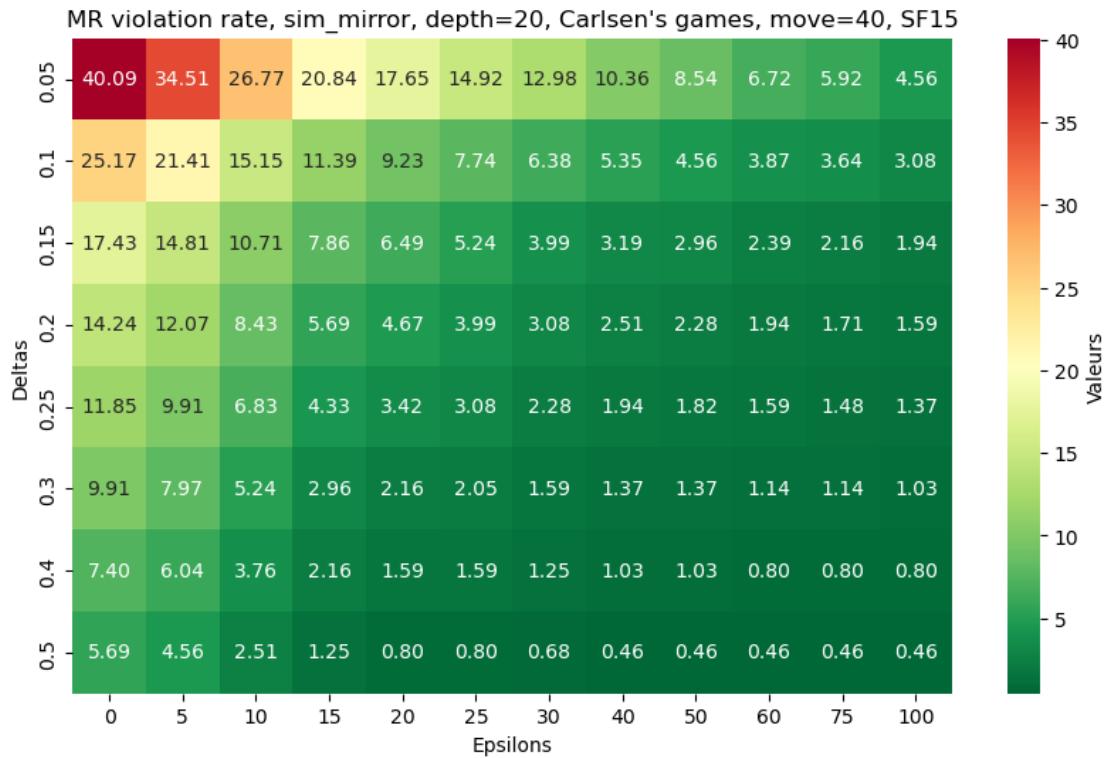


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40sim_mirror_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " Carlsen's games, move=40, SF15")
```

move = 40



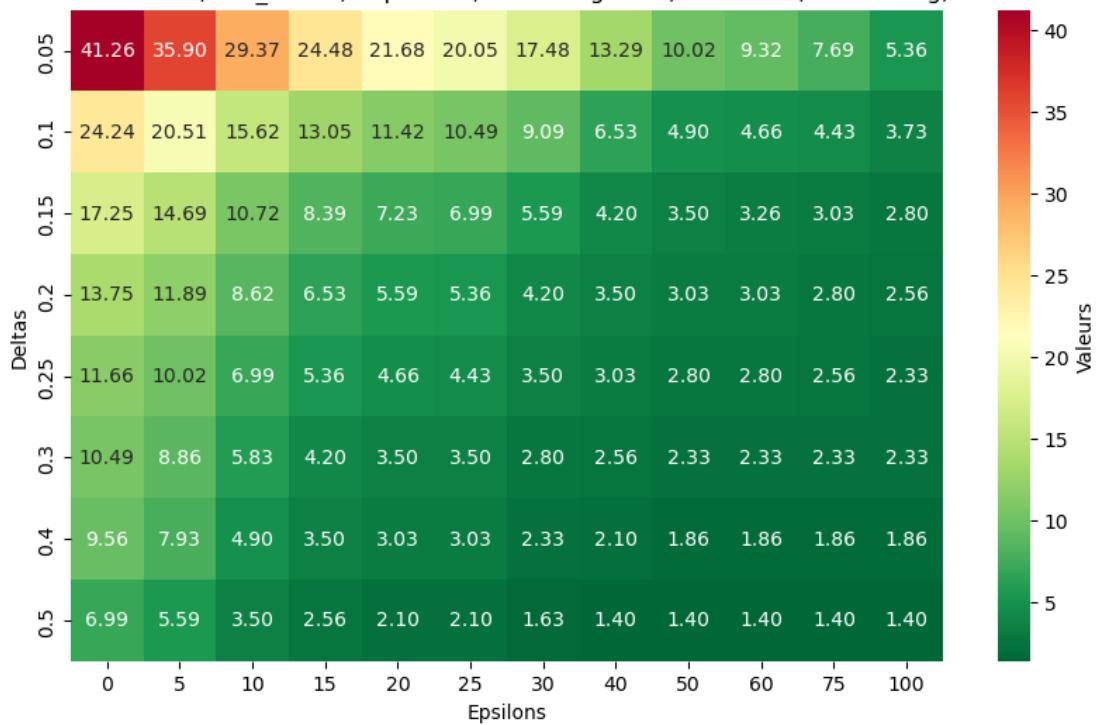
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40_nocastlingsim_mirror_d_20.pkl')
      with open(path, 'rb') as file:
          evs01540 = pickle.load(file)

      print('move = 40')
      tests(evs01540, 0, 20, "Carlen's games, move=40, no castling, SF15")
```

move = 40

MR violation rate, sim_mirror, depth=20, Carlen's games, move=40, no castling, SF15



0.11.2 sim_axis

Openings (move=10)

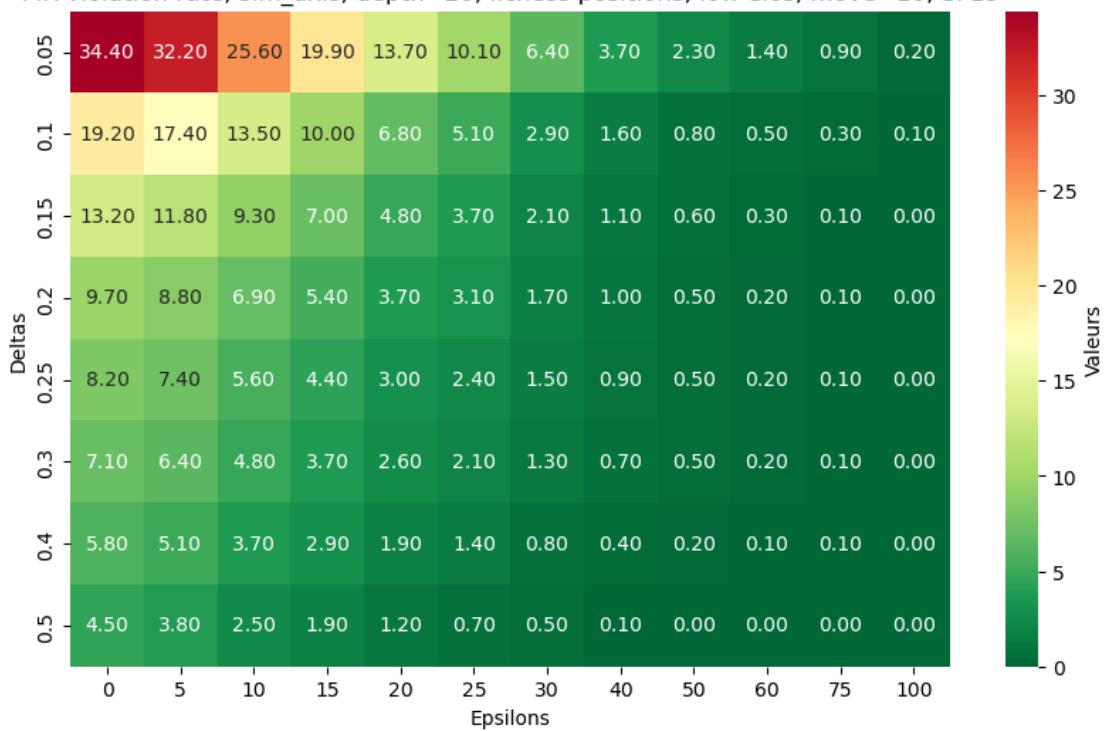
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-10sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, " lichess positions, low elos, move=10, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=10, SF15



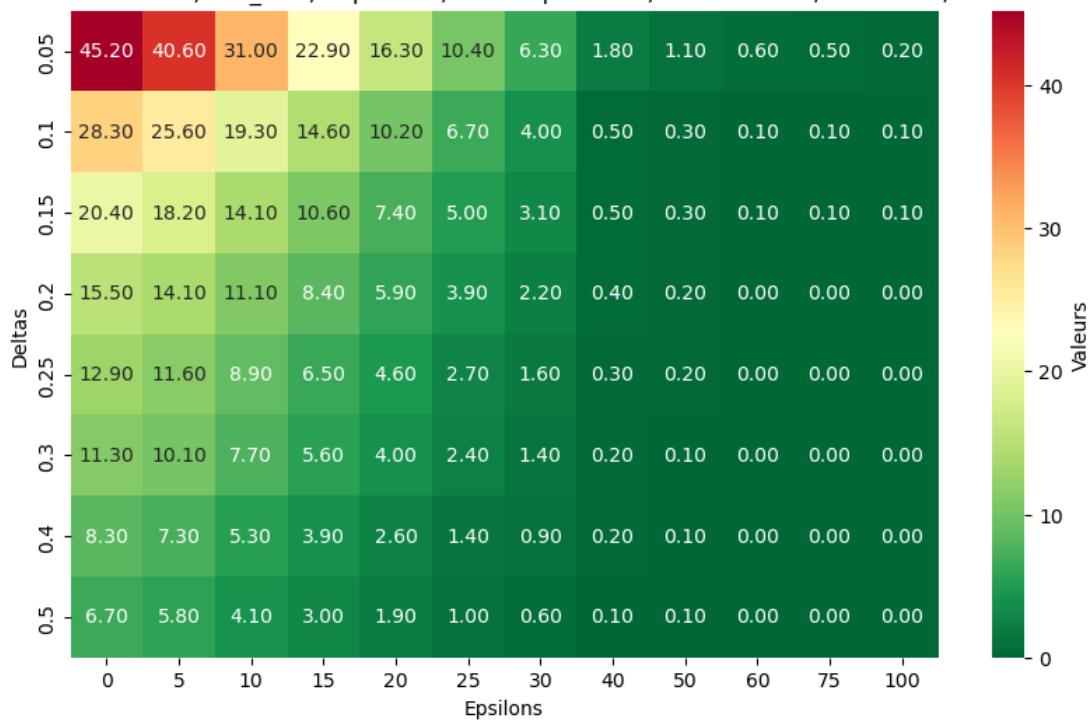
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-10sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,1,20, " lichess positions, medium elos, move=10, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=10, SF15



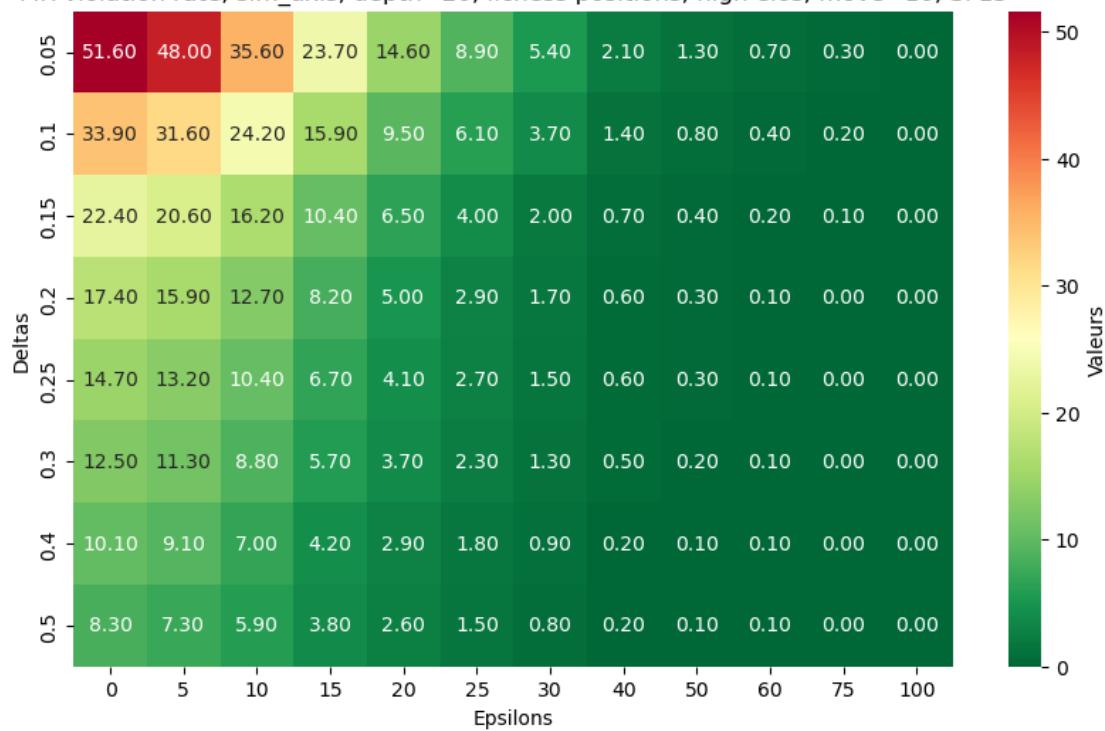
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, " lichess positions, high elos, move=10, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=10, SF15



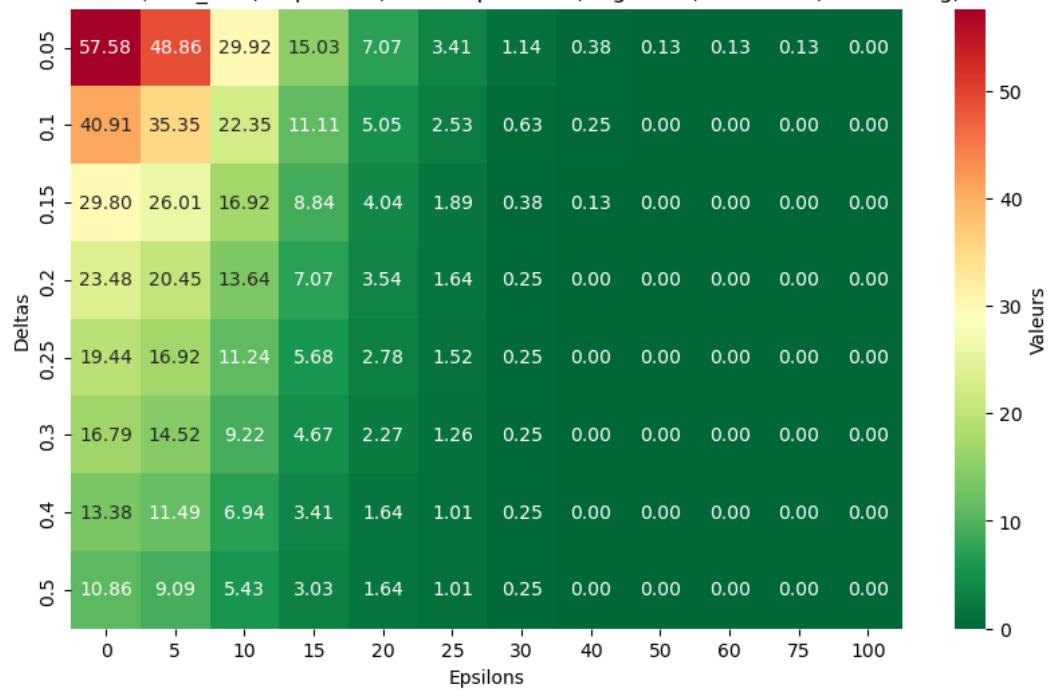
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals','ev_lichess2000-2500-10_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,1,20, " lichess positions, high elos, move=10, no castling, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=10, no castling, SF15

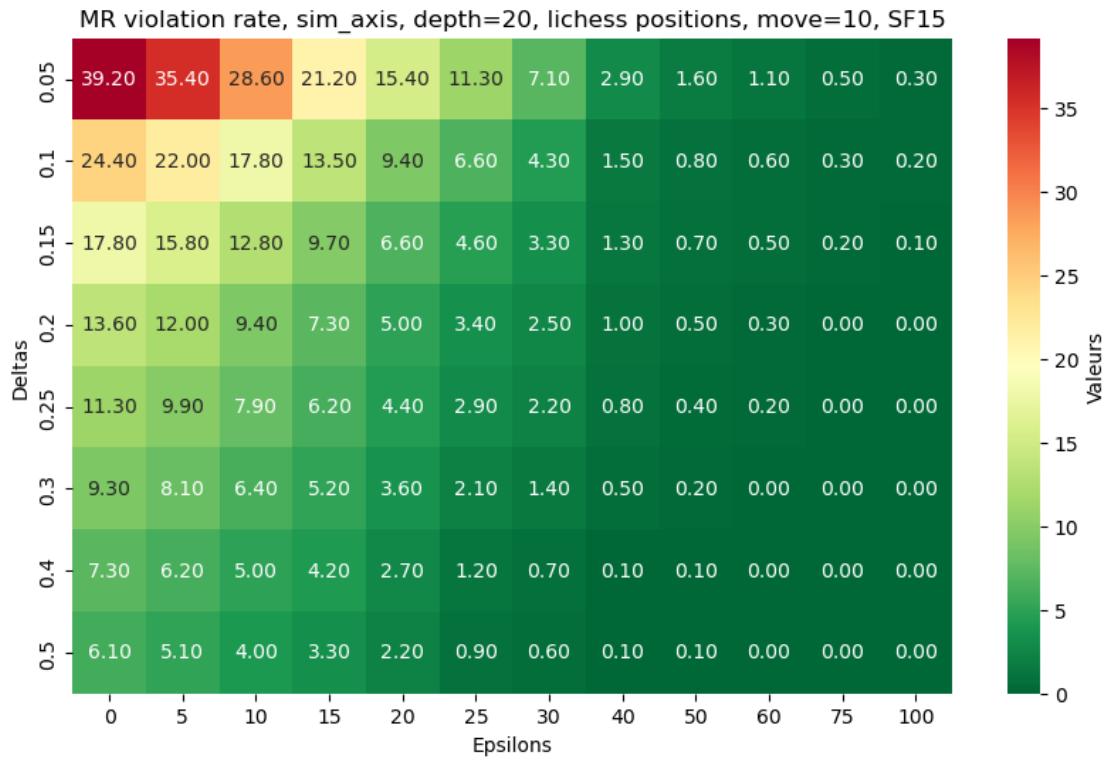


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, ", lichess positions, move=10, SF15")
```

move = 10



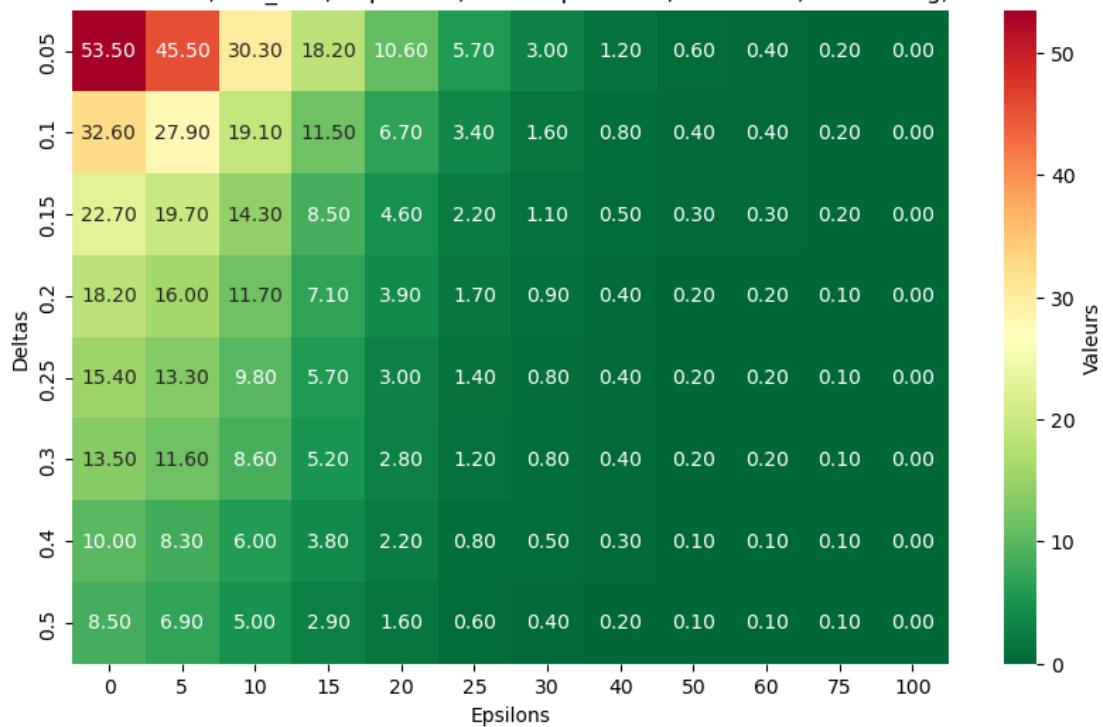
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals','ev_lichessAllElos-10_nocastlingsim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs,1,20, " lichess positions, move=10, no castling, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, move=10, no castling, SF15

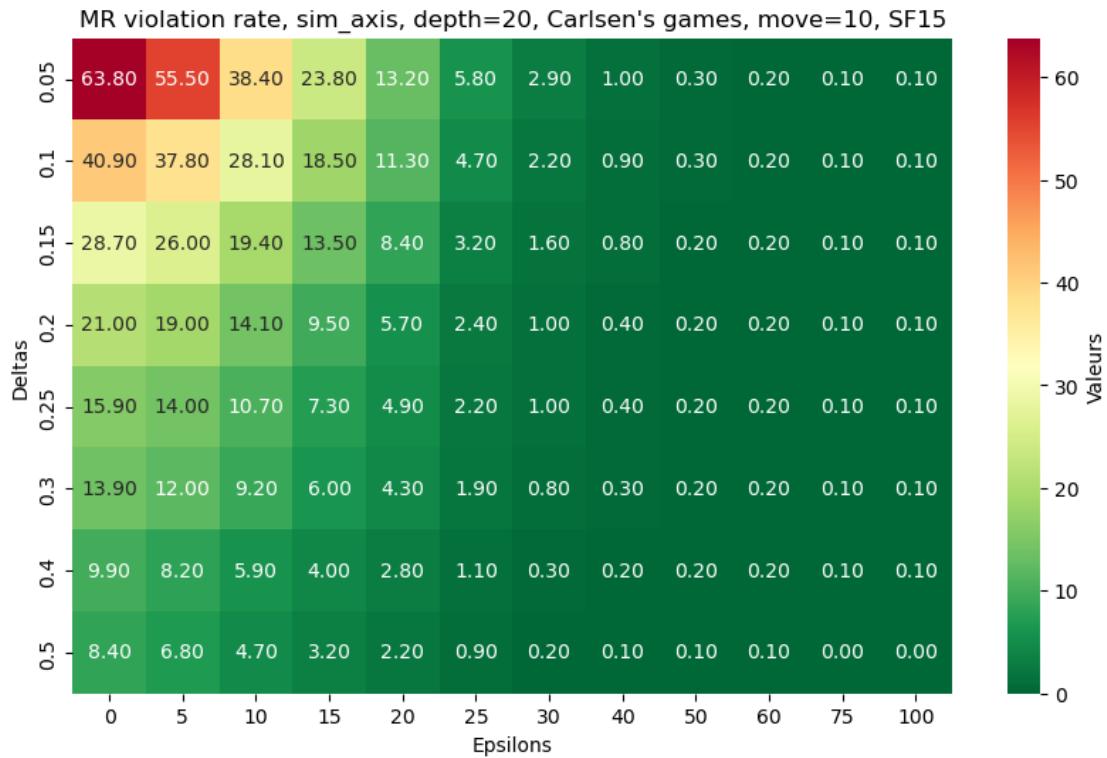


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, " Carlsen's games, move=10, SF15")
```

move = 10



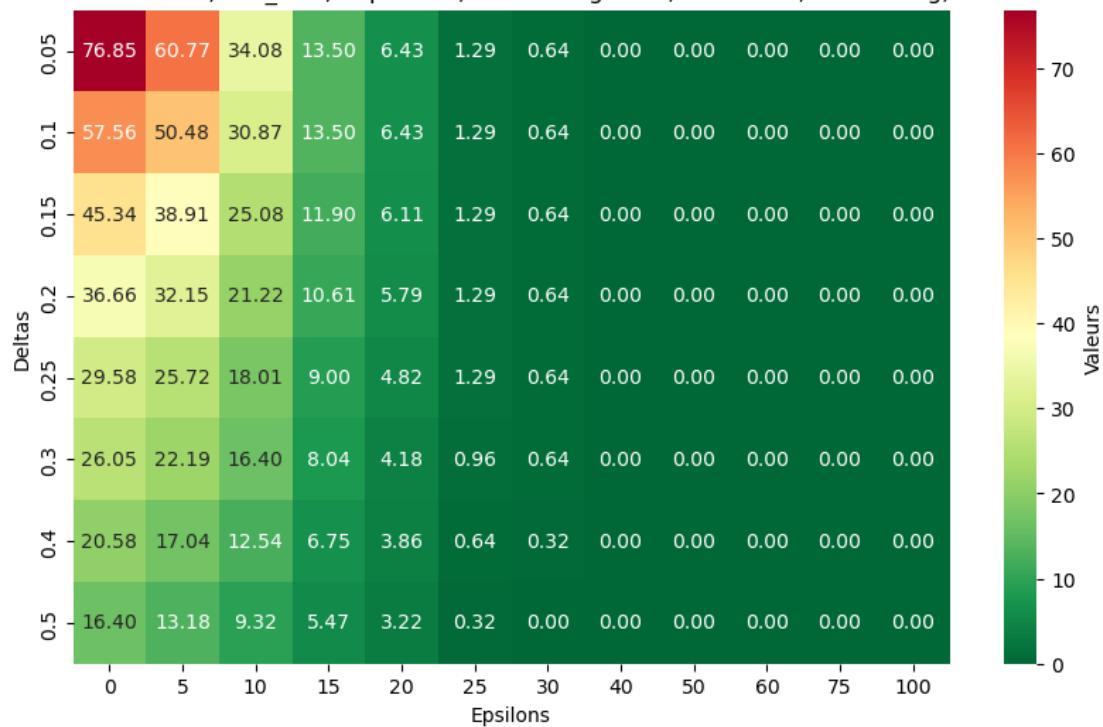
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs11510 = pickle.load(file)

      print('move = 10')
      tests(evs11510, 1, 20, " Carlsen's games, move=10, no castling, SF15")
```

move = 10

MR violation rate, sim_axis, depth=20, Carlsen's games, move=10, no castling, SF15



Middlegames (move=20)

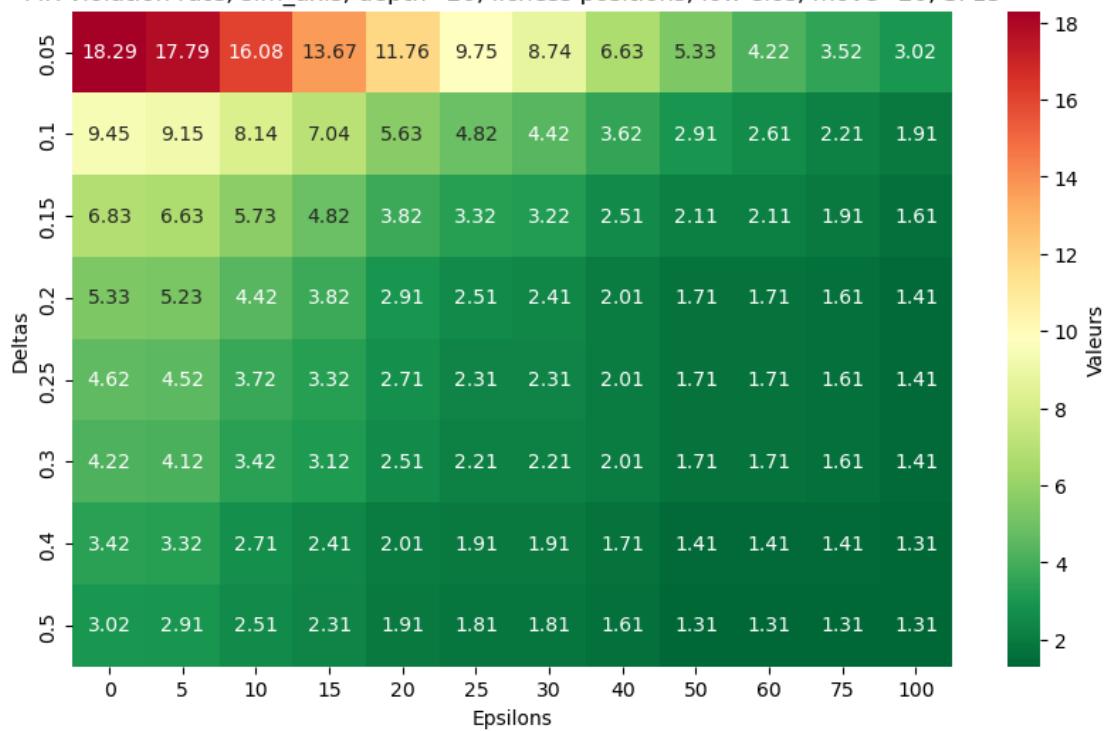
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-20sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " lichess positions, low elos, move=20, SF15")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=20, SF15



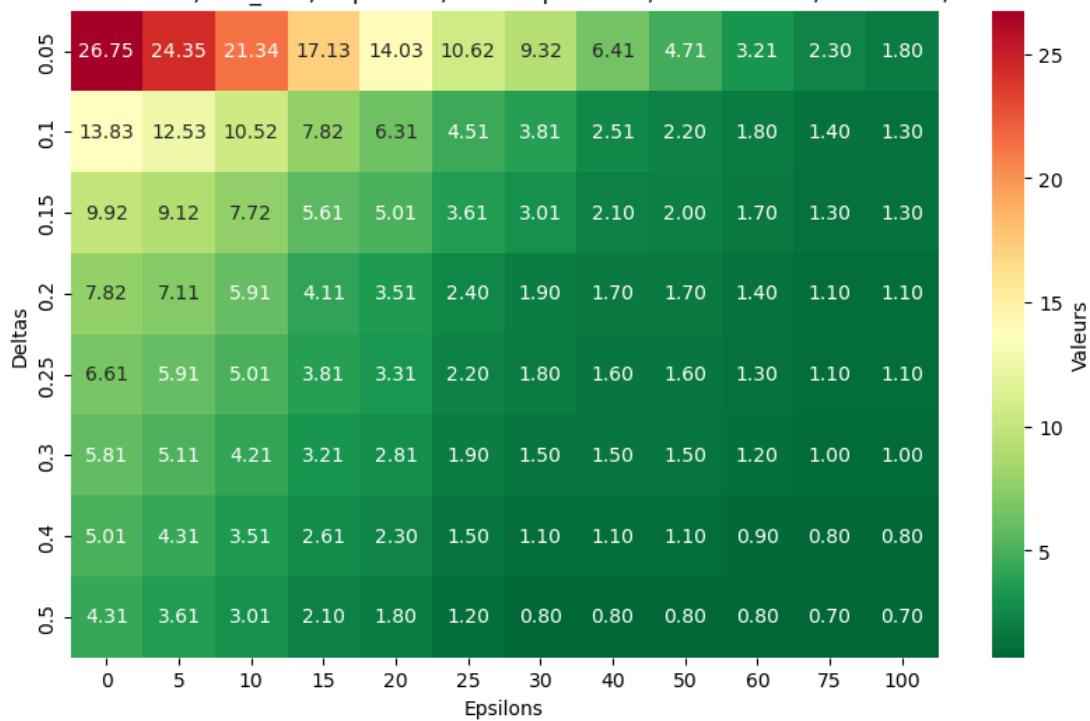
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-20sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs,1,20, " lichess positions, medium elos, move=20, SF15")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=20, SF15



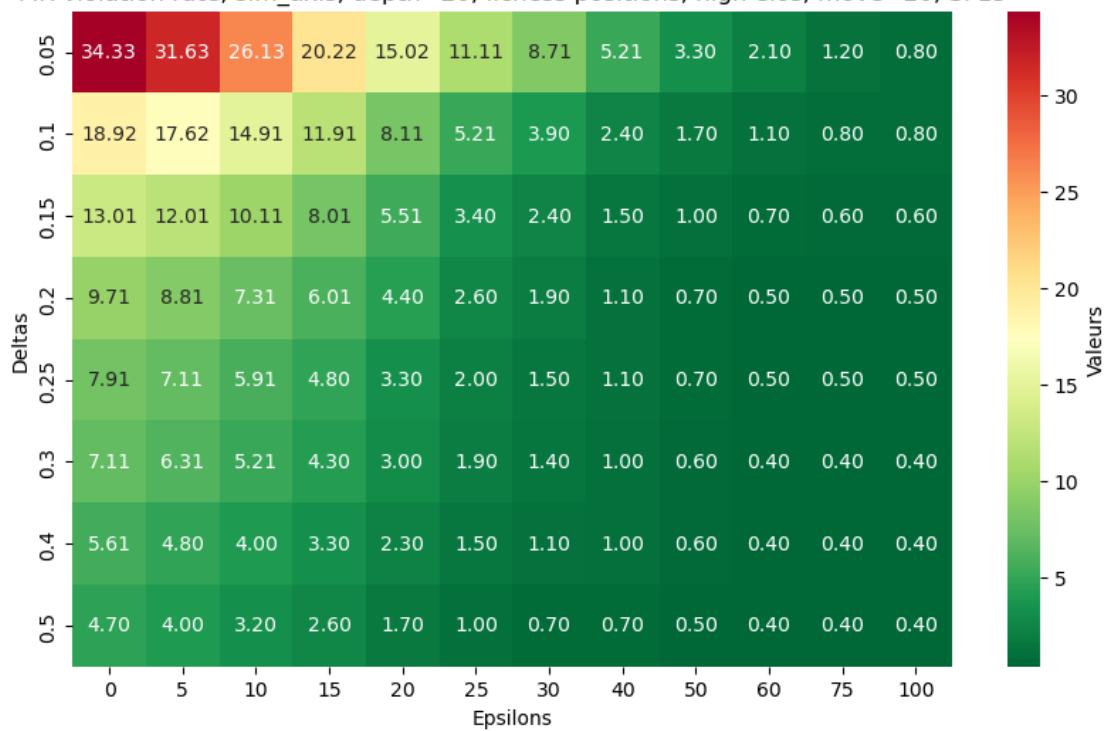
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " lichess positions, high elos, move=20, SF15")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=20, SF15



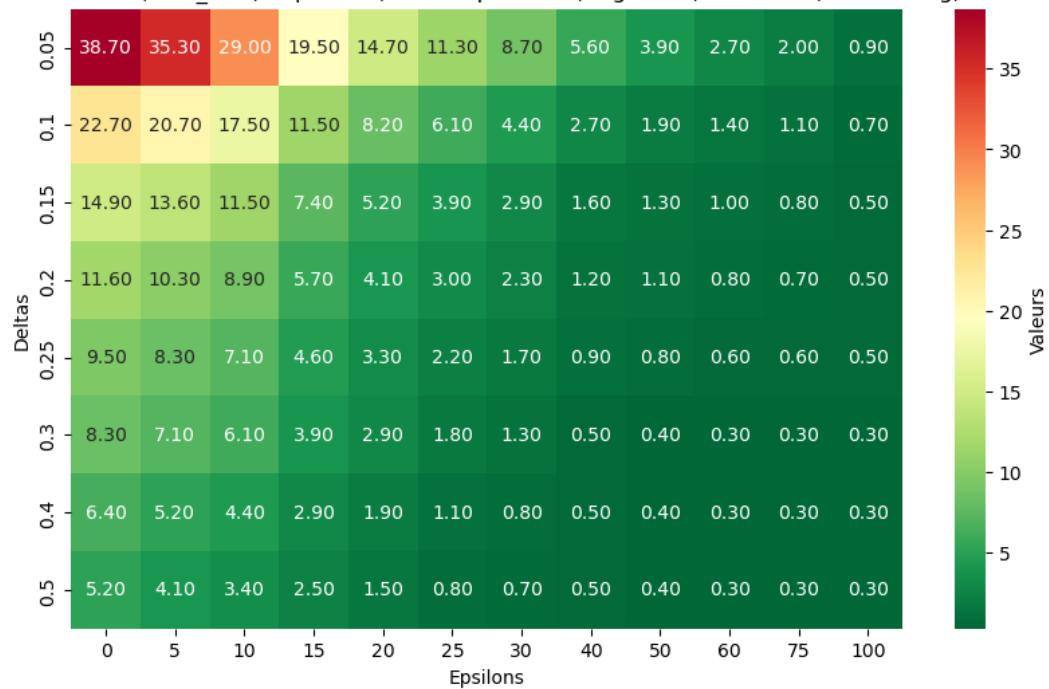
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 1, 20, " lichess positions, high elos, move=20, no castling, SF15")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=20, no castling, SF15

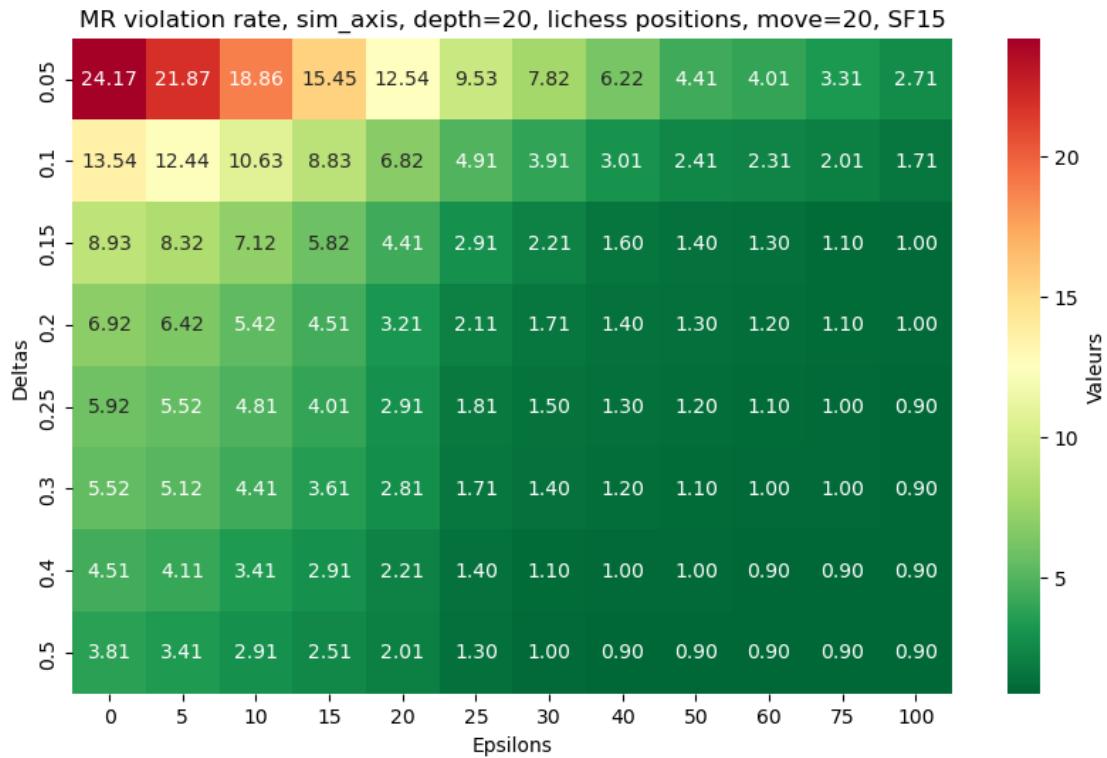


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-20sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " lichess positions, move=20, SF15")
```

move = 20

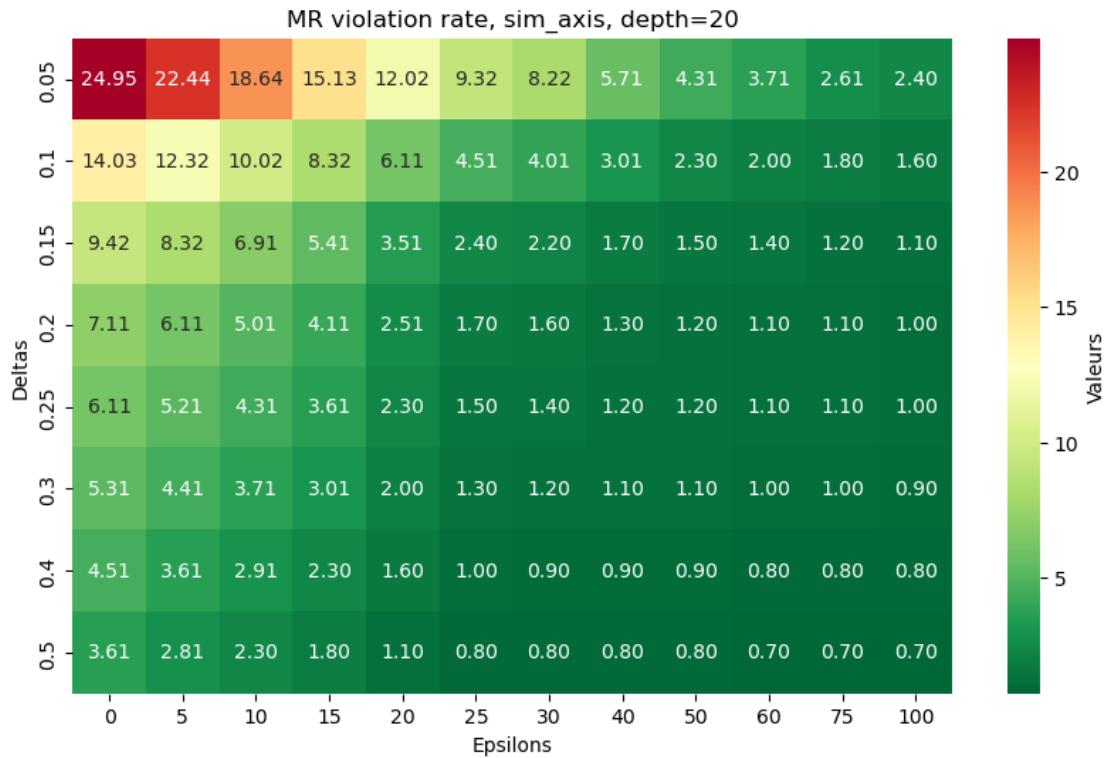


Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals','ev_lichessAllElos-20_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs,1,20)
```

move = 20

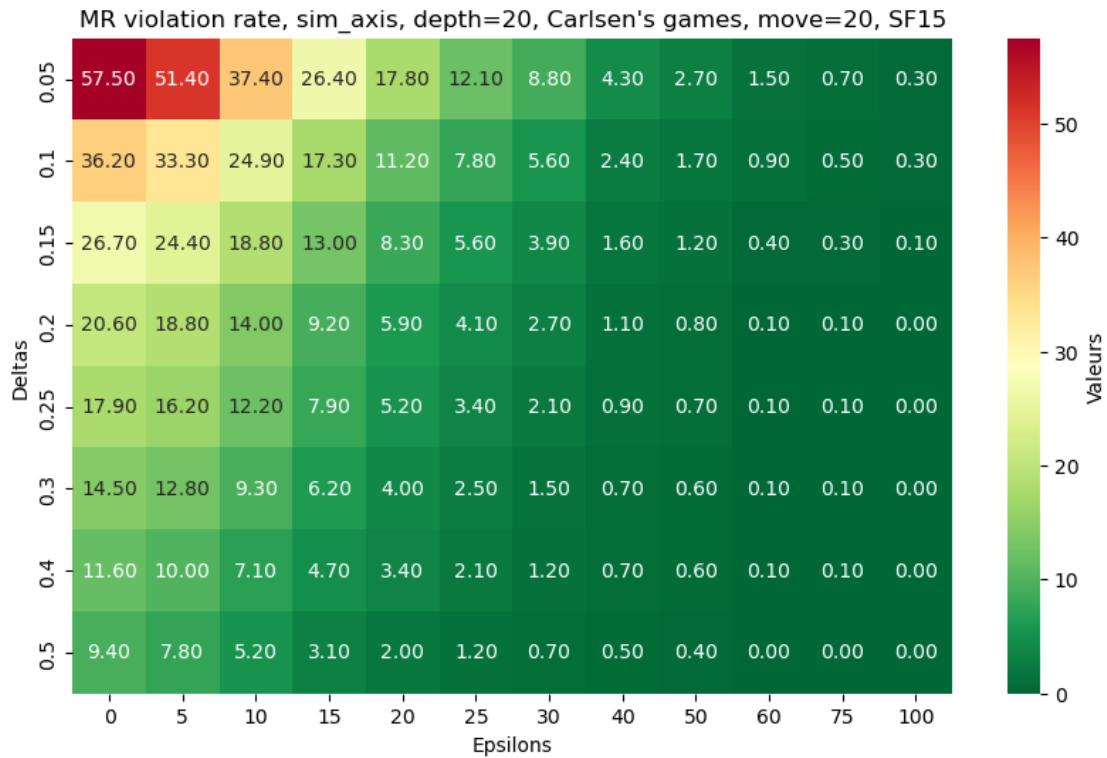


Carlsen's games

```
[ ]: path = os.path.join('reals','ev_Carlsen-20sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs,1,20, " Carlsen's games, move=20, SF15")
```

move = 20



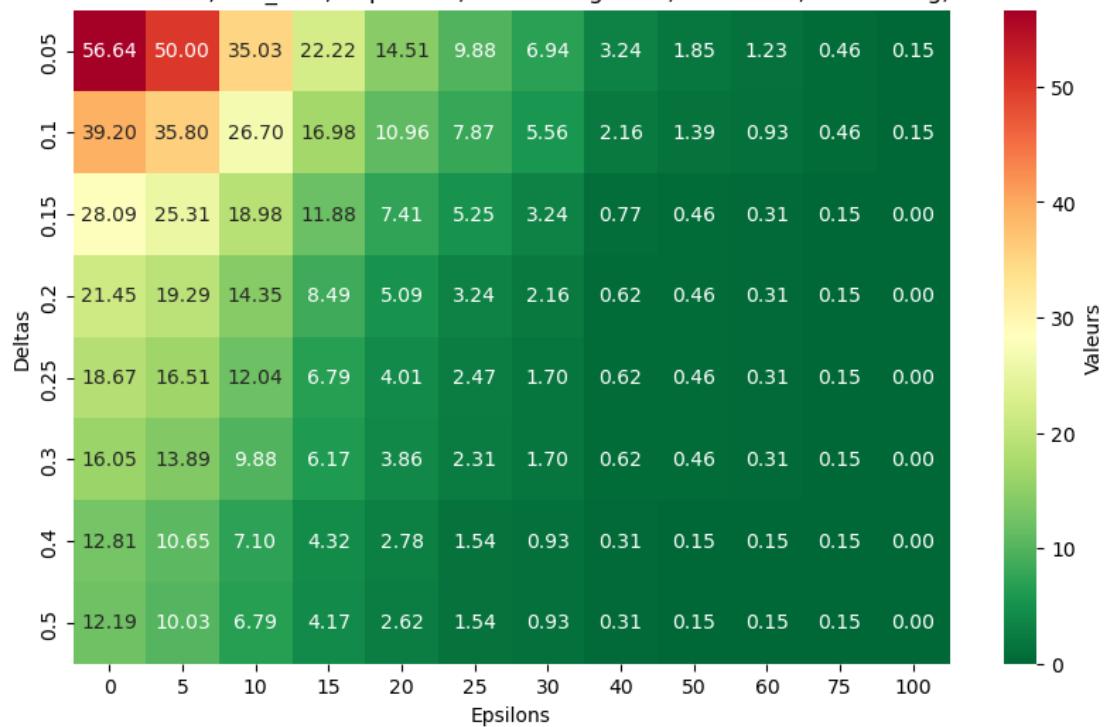
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs11520 = pickle.load(file)

      print('move = 20')
      tests(evs11520, 1, 20, "Carlsen's games, move=20, no castling, SF15")
```

move = 20

MR violation rate, sim_axis, depth=20, Carlsen's games, move=20, no castling, SF15



Middle/Endgames (move=30)

Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-30sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 1, 20, " lichess positions, low elos, move=30, SF15")
```

move = 30



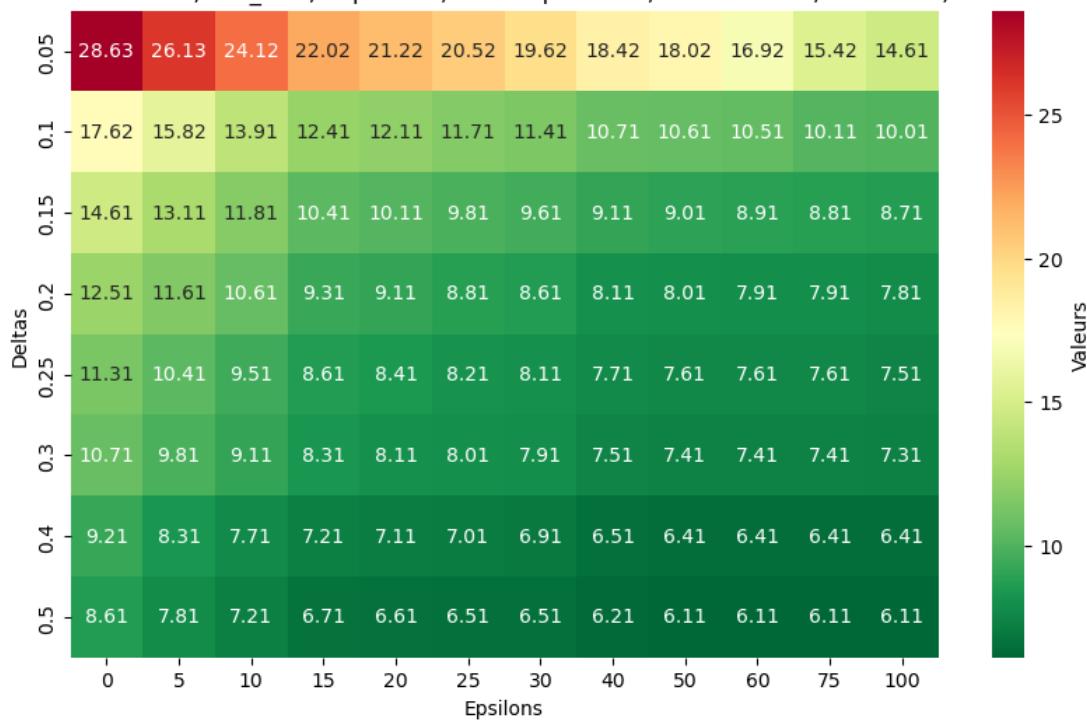
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-30sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs,1,20, " lichess positions, medium elos, move=30, SF15")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=30, SF15



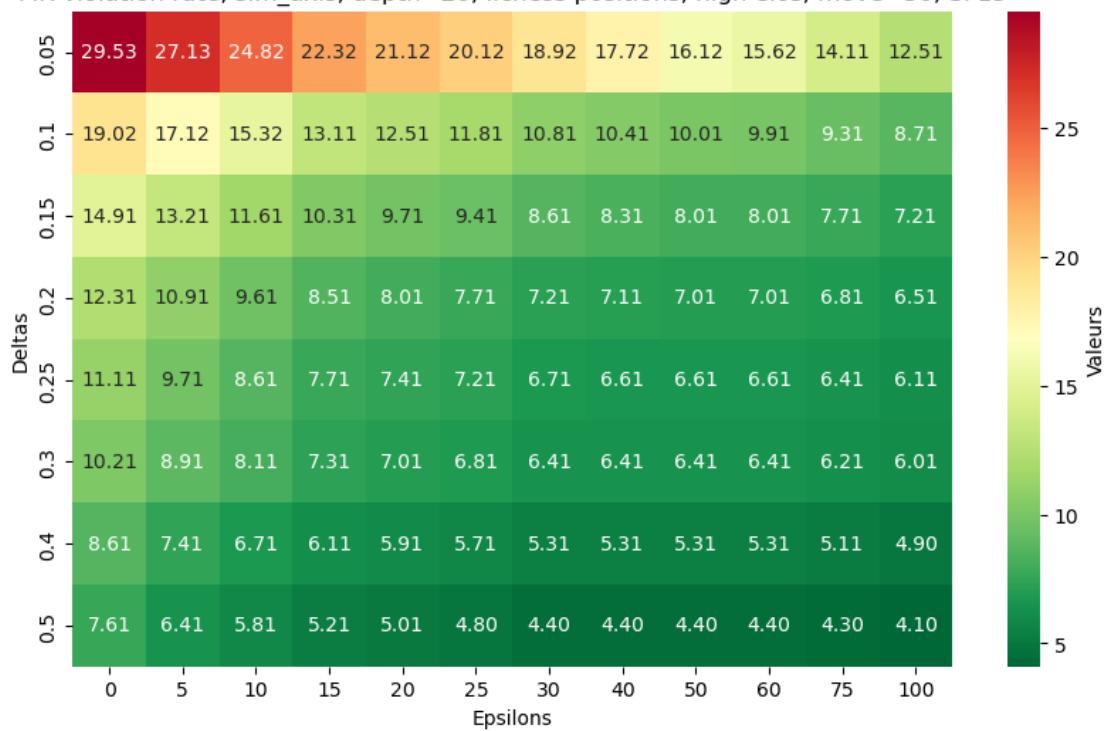
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 1, 20, " lichess positions, high elos, move=30, SF15")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=30, SF15

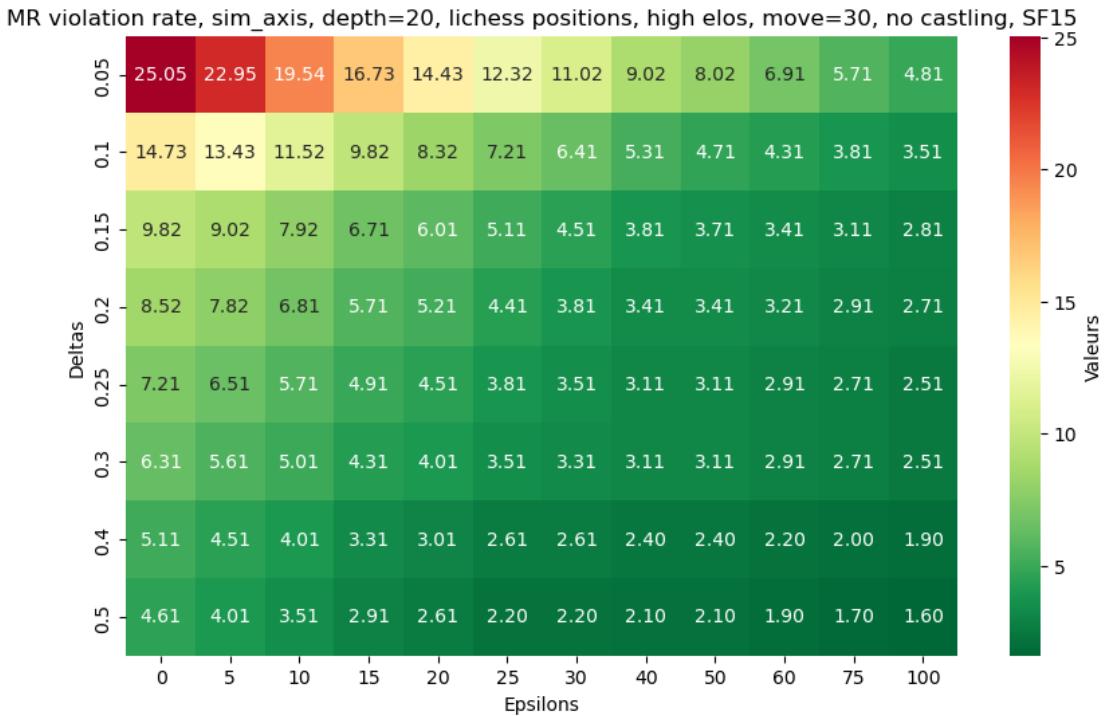


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals','ev_lichess2000-2500-30_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs,1,20, " lichess positions, high elos, move=30, no castling, SF15")
```

move = 30

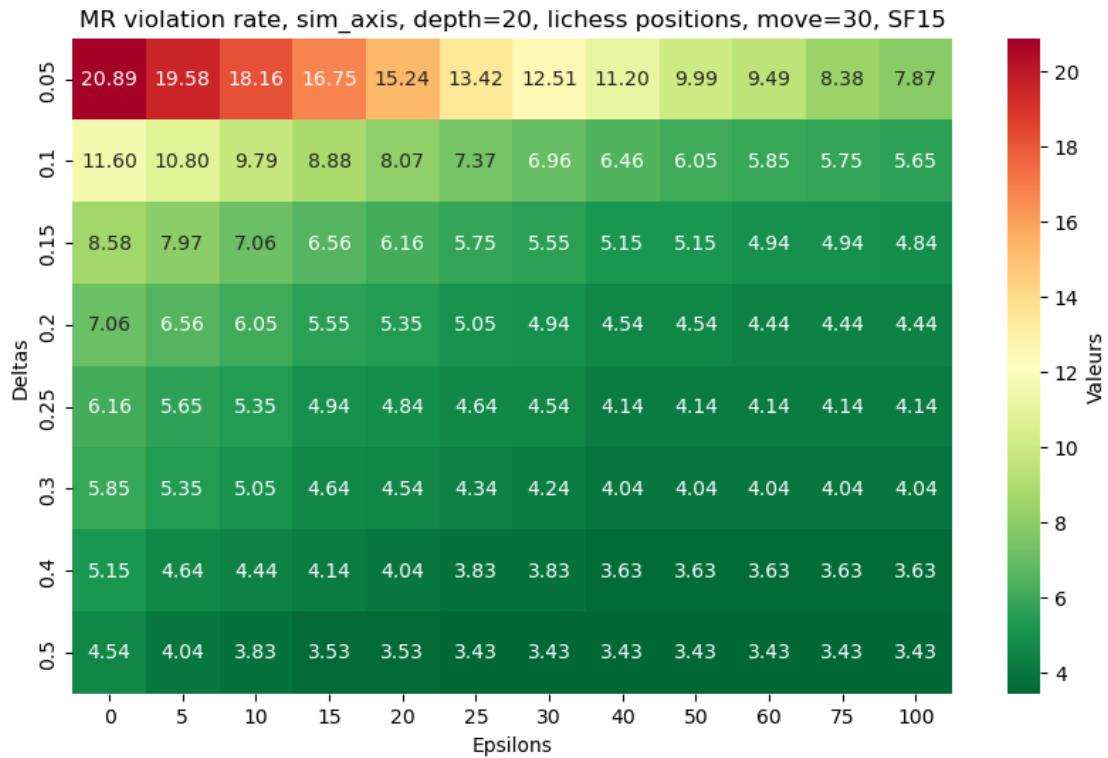


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 1, 20, ", lichess positions, move=30, SF15")
```

move = 30



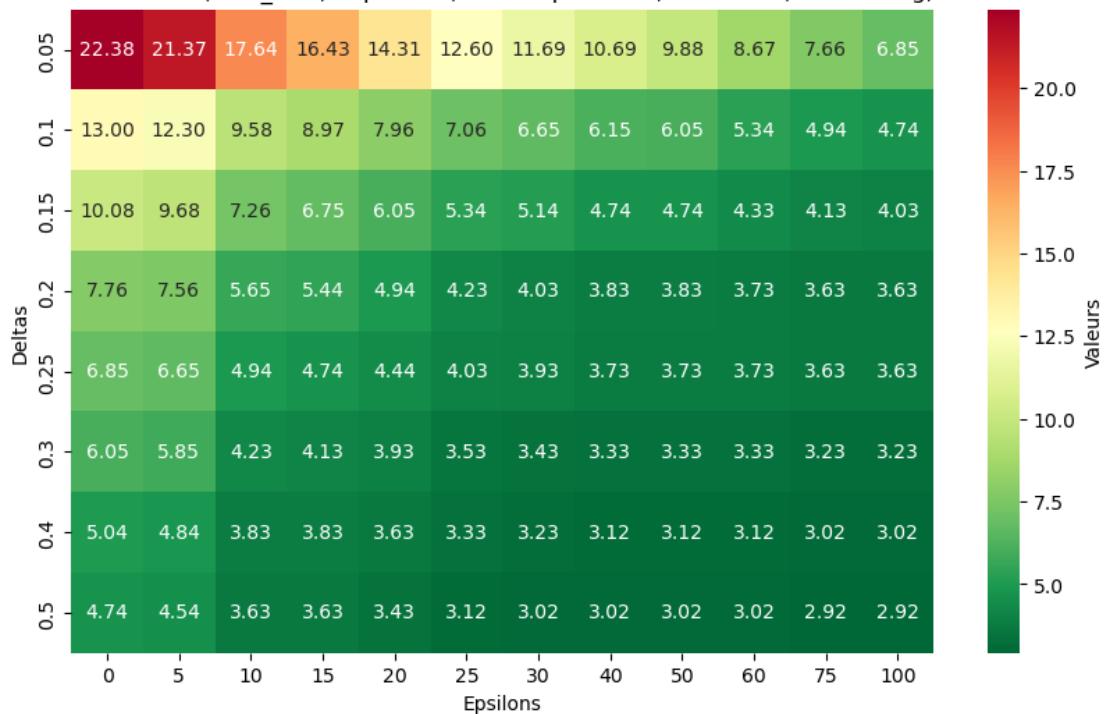
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals','ev_lichessAllElos-30_nocastlingsim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs,1,20, " lichess positions, move=30, no castling, SF15")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, move=30, no castling, SF15

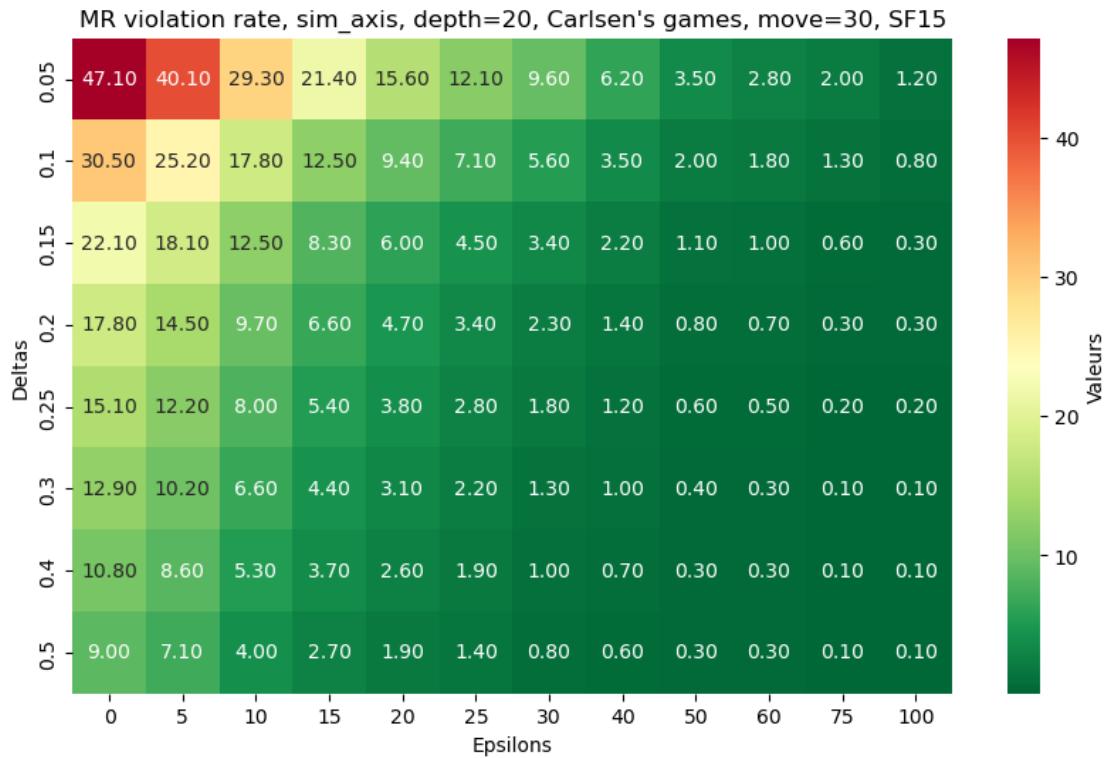


Carlsen's games

```
[ ]: path = os.path.join('reals','ev_Carlsen-30sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs,1,20, " Carlsen's games, move=30, SF15")
```

move = 30



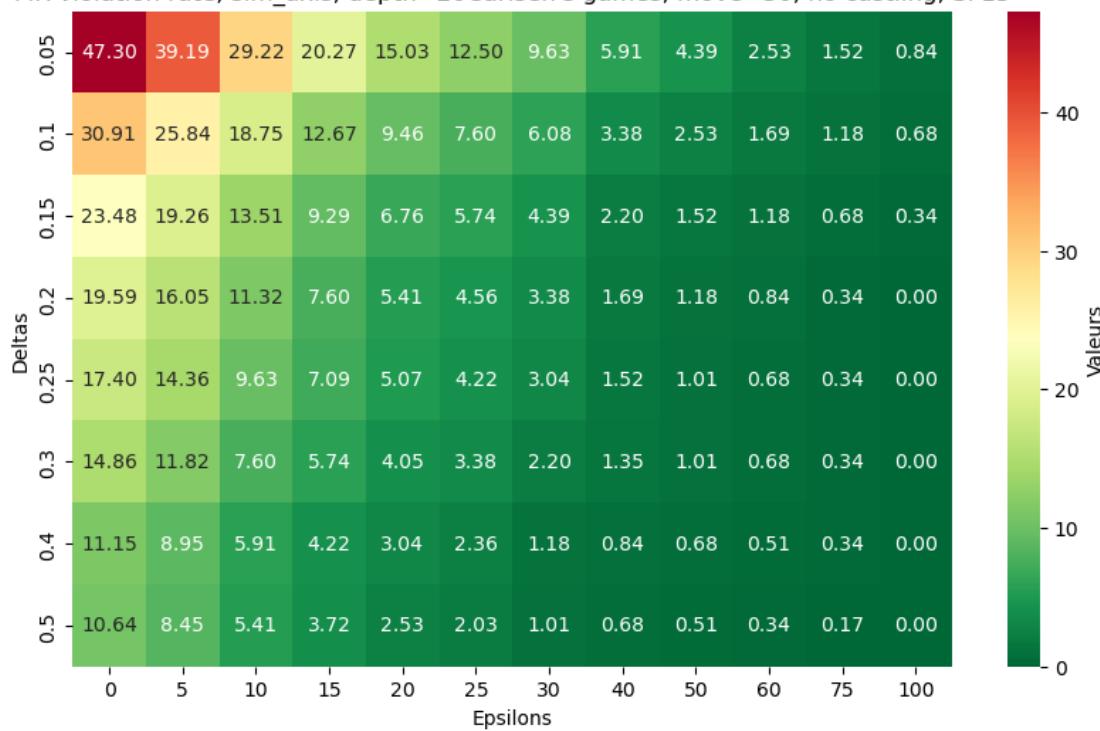
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs11530 = pickle.load(file)

      print('move = 30')
      tests(evs11530, 1, 20, "Carlsen's games, move=30, no castling, SF15")
```

move = 30

MR violation rate, sim_axis, depth=20Carlsen's games, move=30, no castling, SF15



Endgames (move=40)

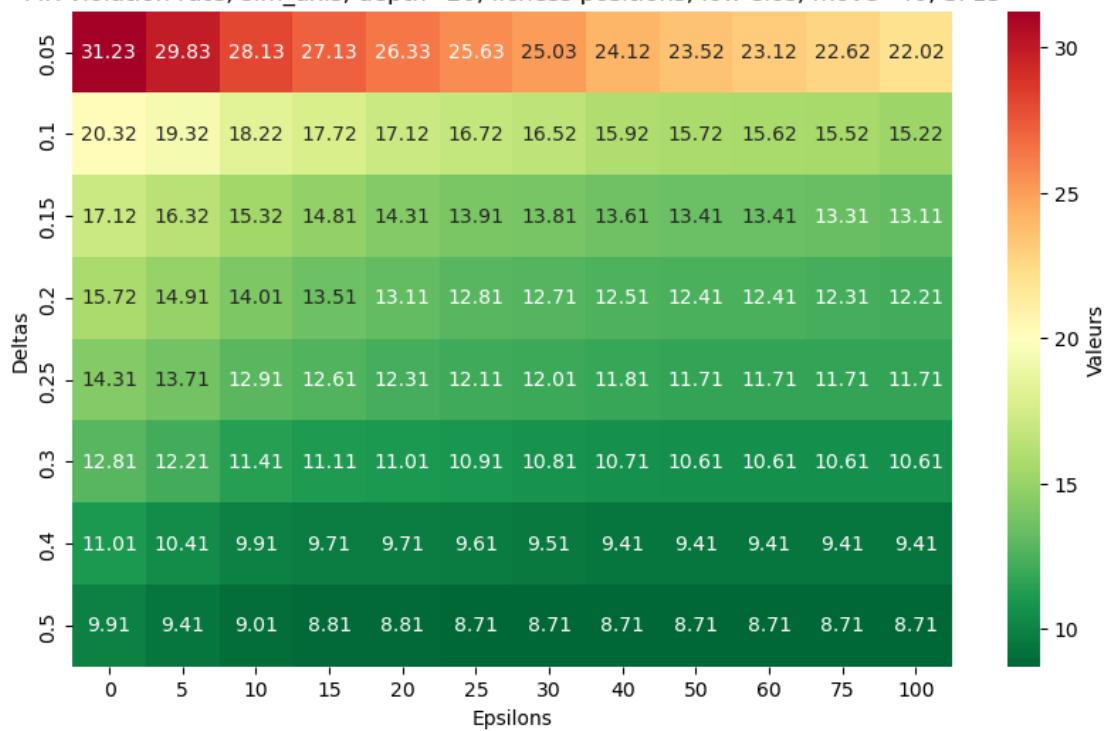
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-40sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 1, 20, " lichess positions, low elos, move=40, SF15")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=40, SF15



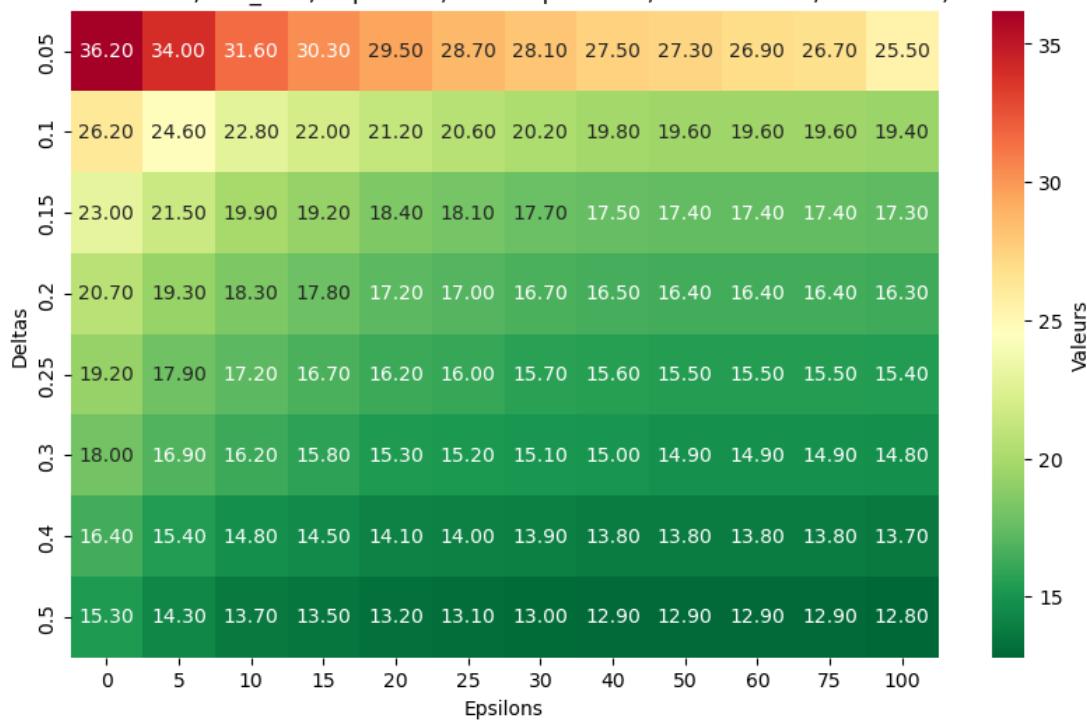
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-40sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs,1,20, " lichess positions, medium elos, move=40, SF15")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=40, SF15



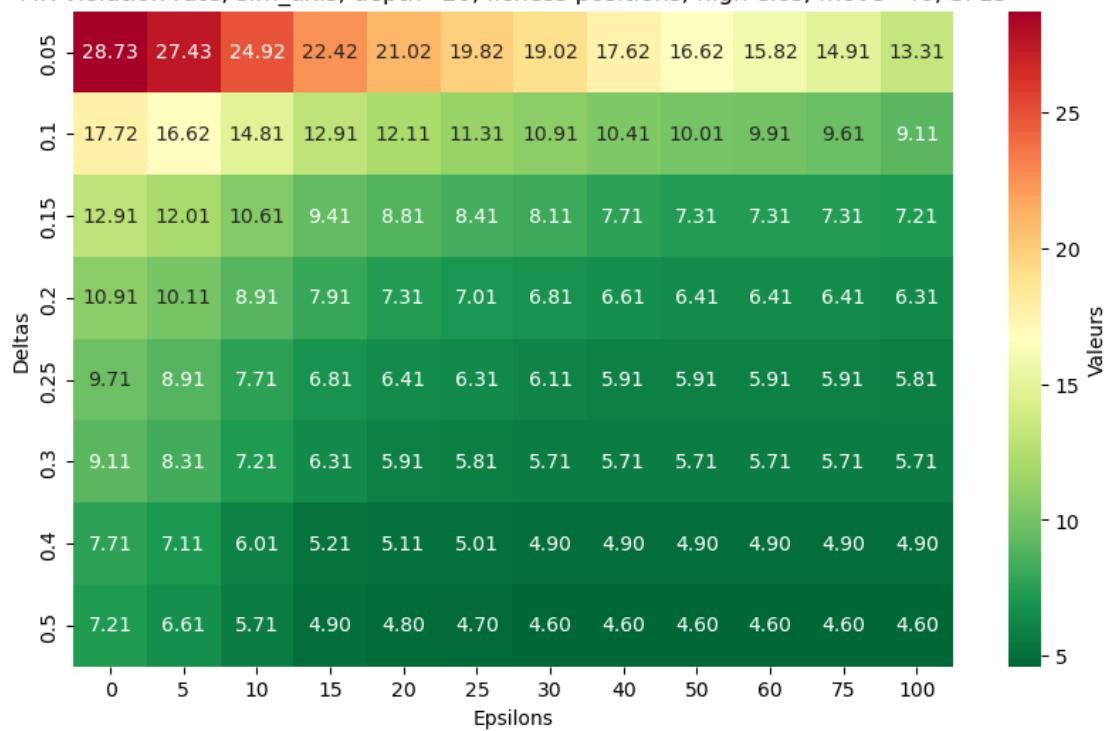
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 1, 20, " lichess positions, high elos, move=40, SF15")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=40, SF15

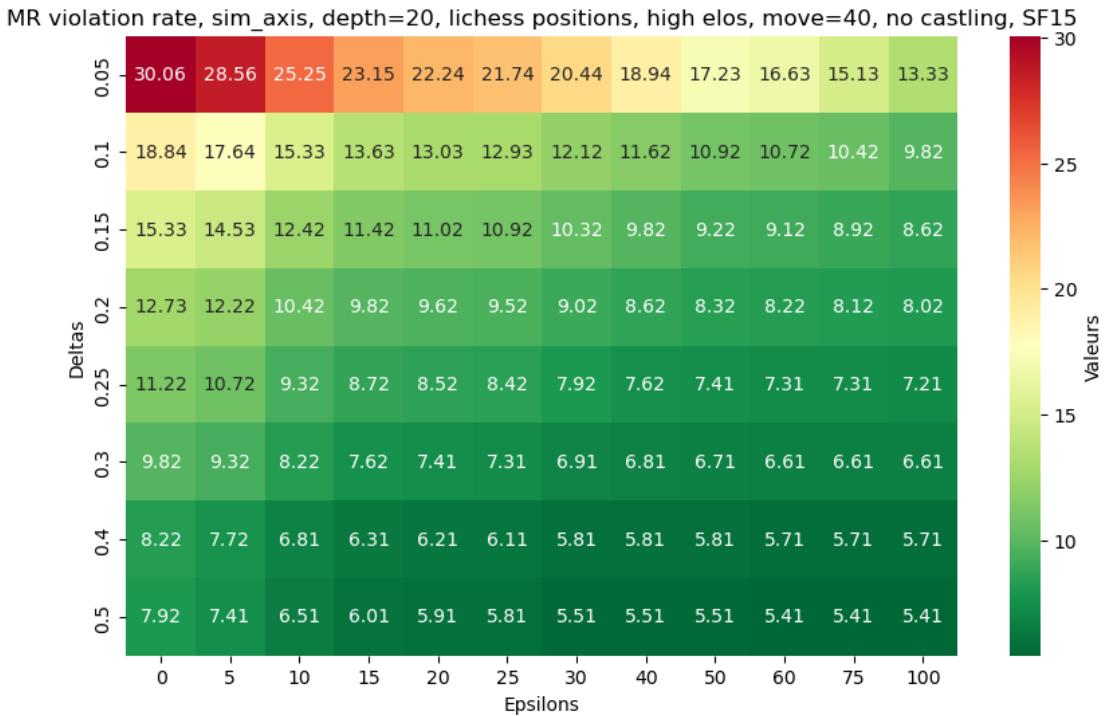


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals','ev_lichess2000-2500-40_nocastlingsim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs,1,20, " lichess positions, high elos, move=40, no castling, SF15")
```

move = 40

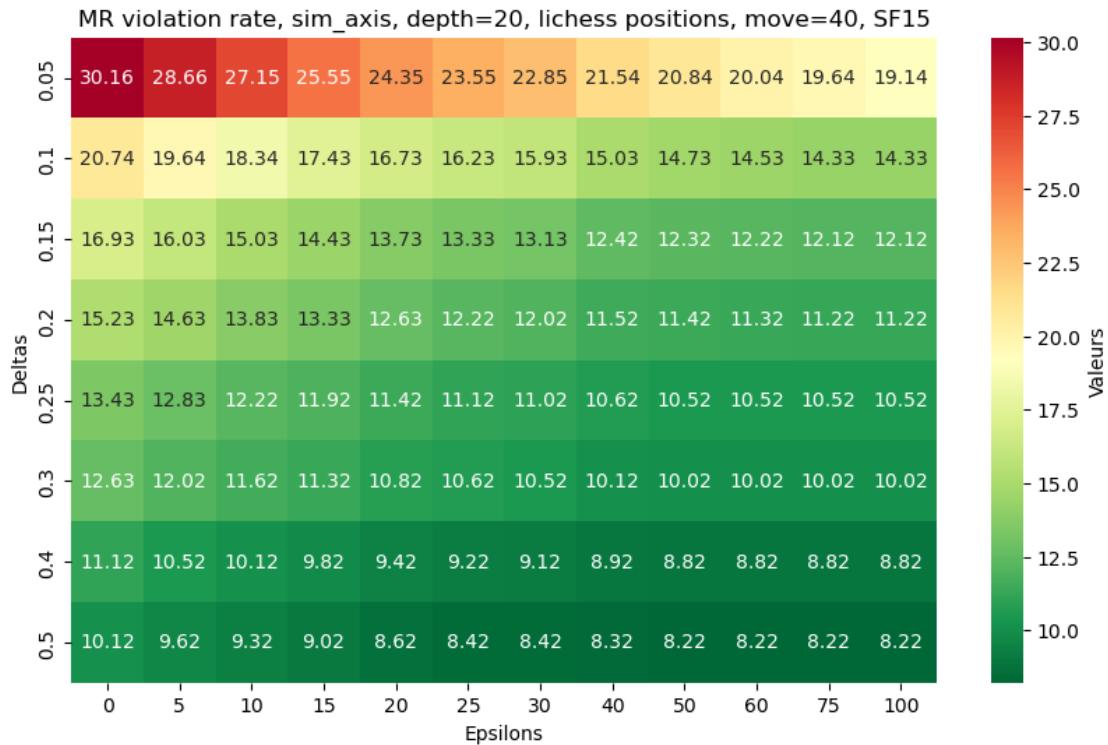


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 1, 20, ", lichess positions, move=40, SF15")
```

move = 40

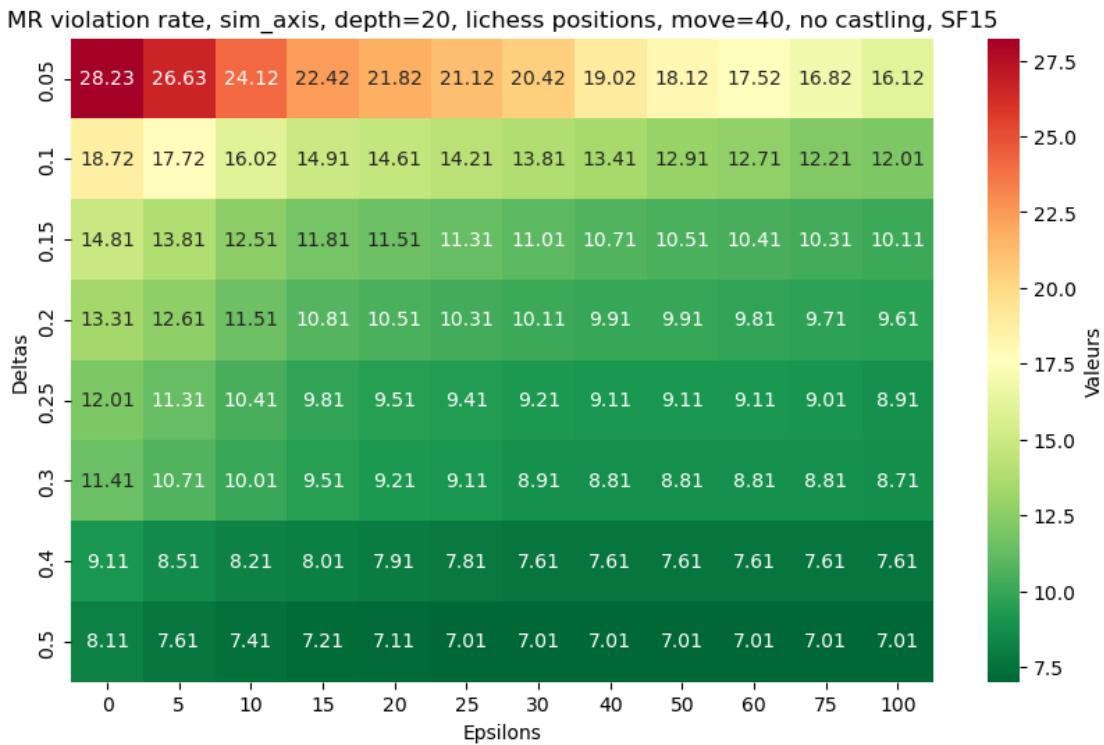


Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40_nocastlingsim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 1, 20, ", lichess positions, move=40, no castling, SF15")
```

move = 40

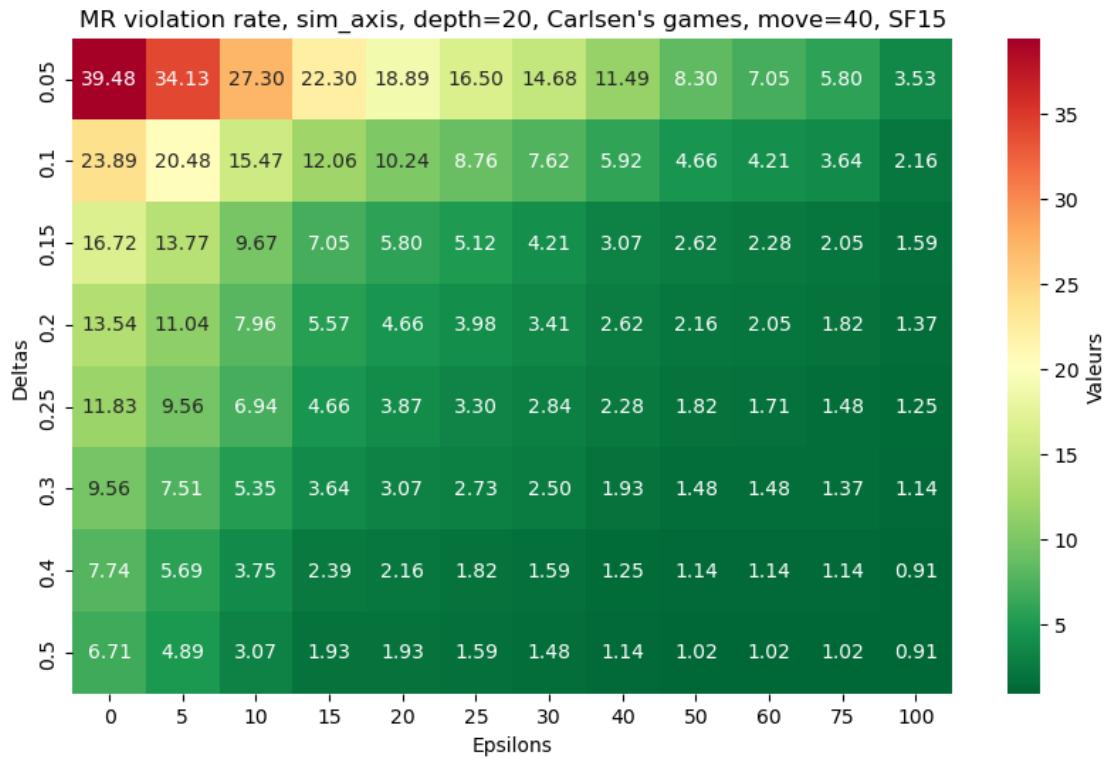


Carlsen's games

```
[ ]: path = os.path.join('reals','ev_Carlsen-40sim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs,1,20, ", Carlsen's games, move=40, SF15")
```

move = 40



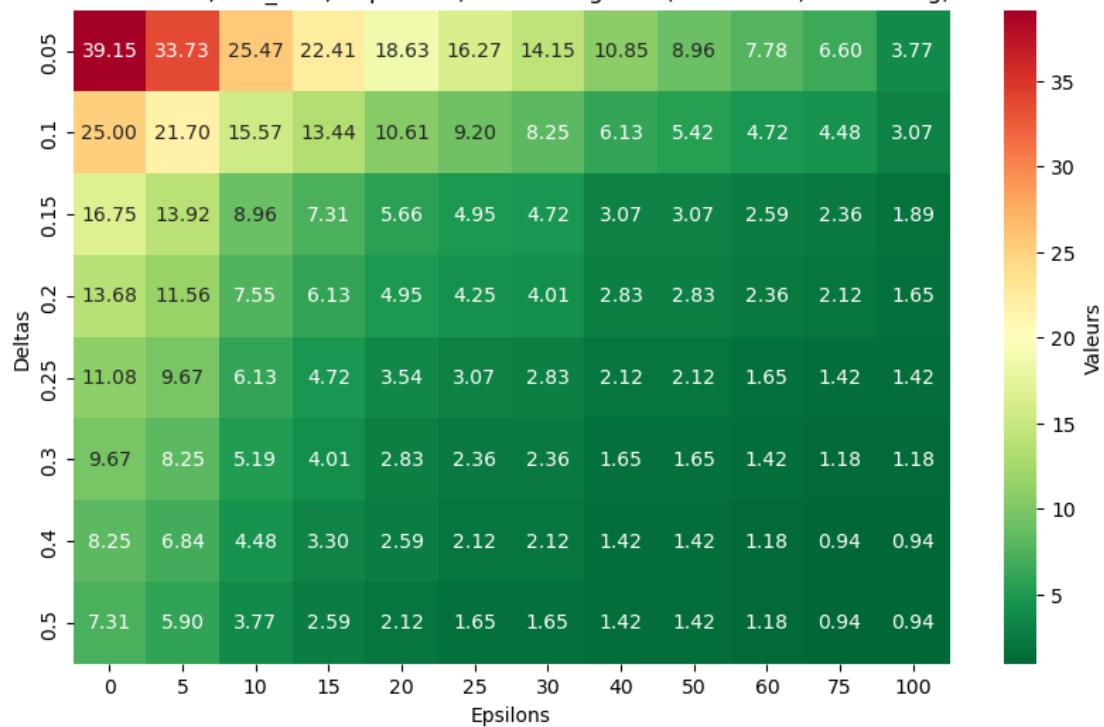
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40_nocastlingsim_axis_d_20.pkl')
      with open(path, 'rb') as file:
          evs11540 = pickle.load(file)

      print('move = 40')
      tests(evs11540, 1, 20, " Carlsen's games, move=40, no castling, SF15")
```

move = 40

MR violation rate, sim_axis, depth=20, Carlsen's games, move=40, no castling, SF15



0.12 Stockfish 16, lichess

0.12.1 sim_mirror

Openings (move=10)

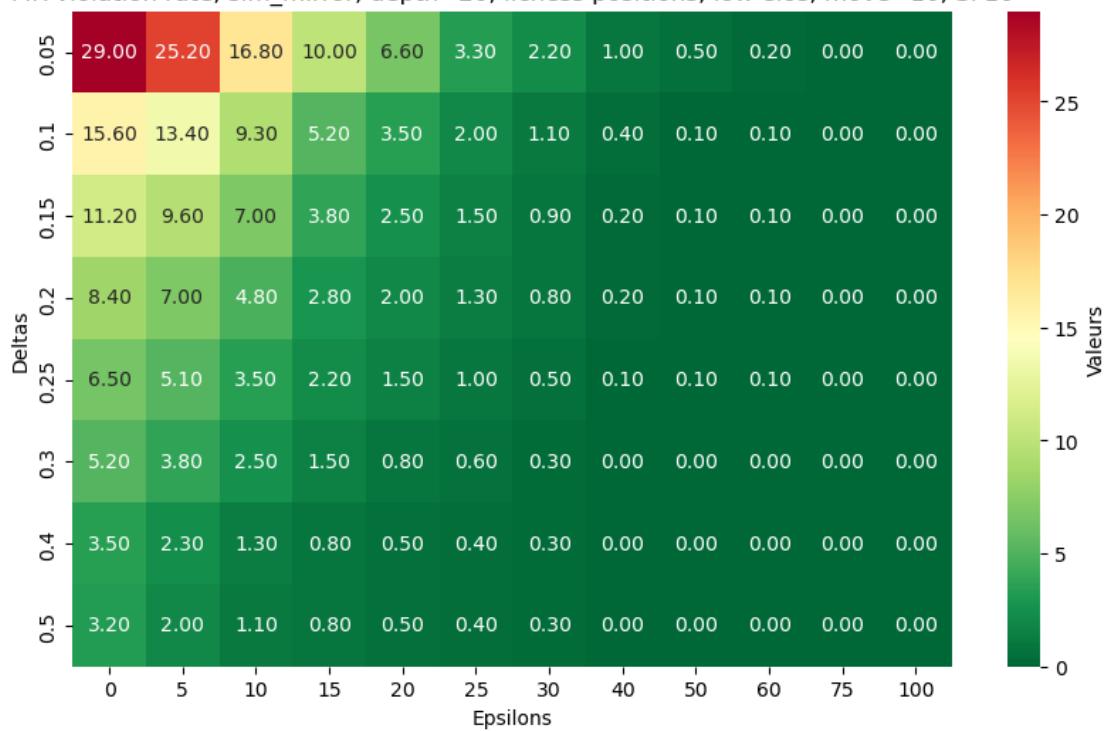
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-10sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs, 0, 20, " lichess positions, low elos, move=10, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=10, SF16



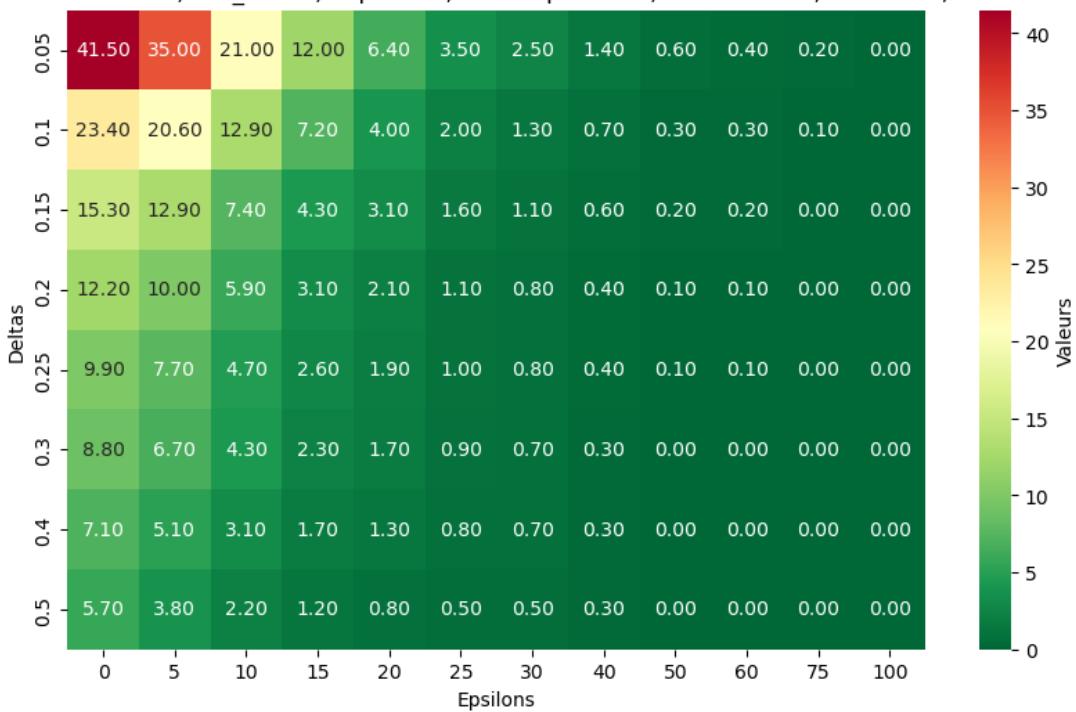
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-10sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,0,20, " lichess positions, medium elos, move=10, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=10, SF16



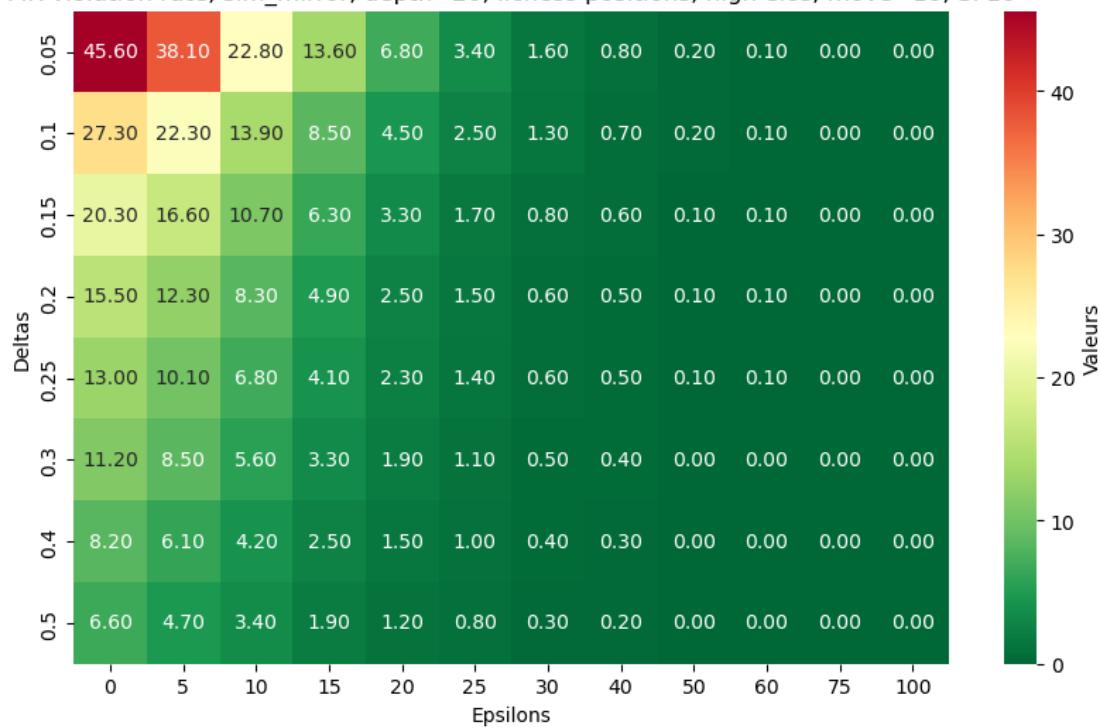
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, " lichess positions, high elos, move=10, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=10, SF16

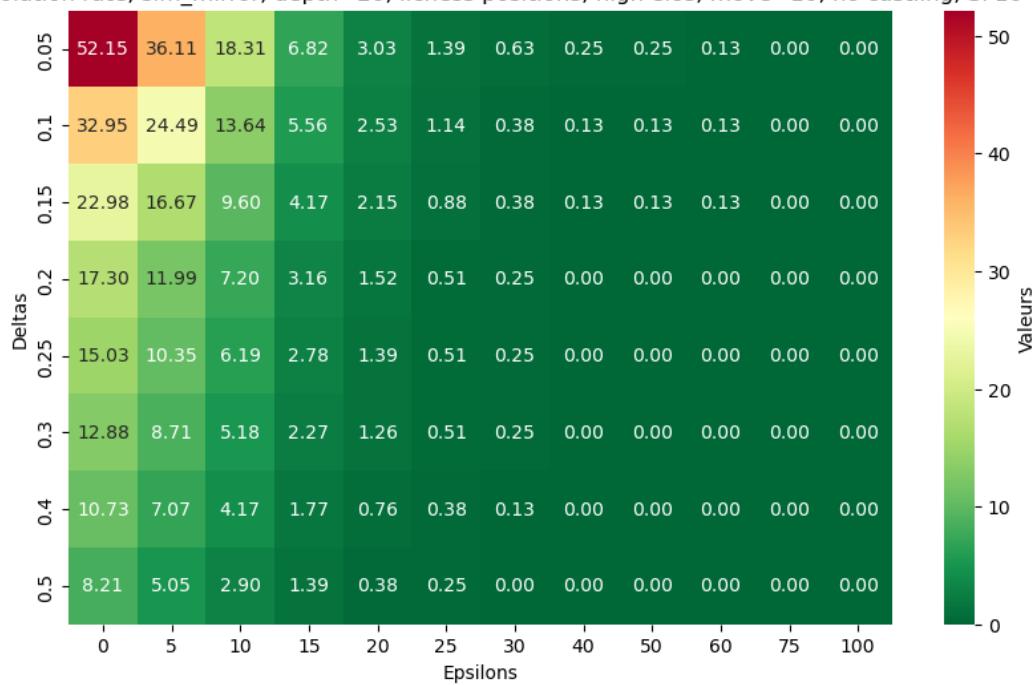


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.  
    ↪join('reals', 'ev_lichess2000-2500-10_nocastlingsim_mirror_d_20_v16.pkl')  
    with open(path, 'rb') as file:  
        evs = pickle.load(file)  
  
    print('move = 10')  
    tests(evs, 0, 20, " lichess positions, high elos, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=10, no castling, SF16

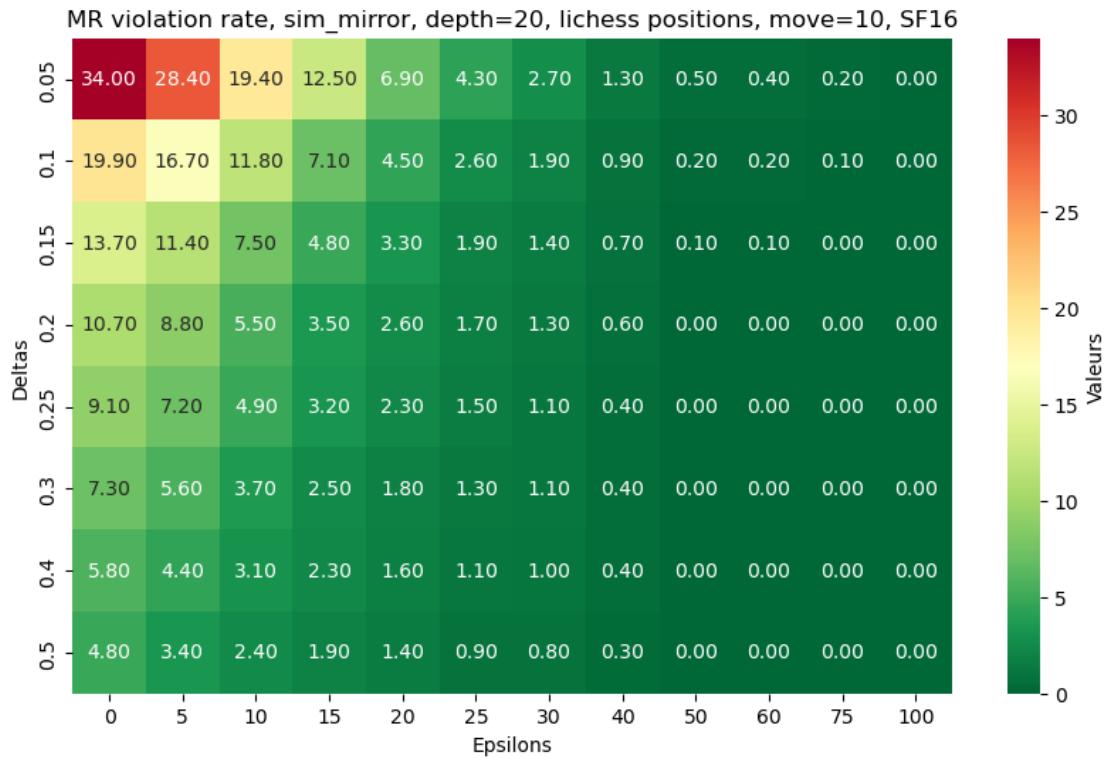


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, " lichess positions, move=10, SF16")
```

move = 10



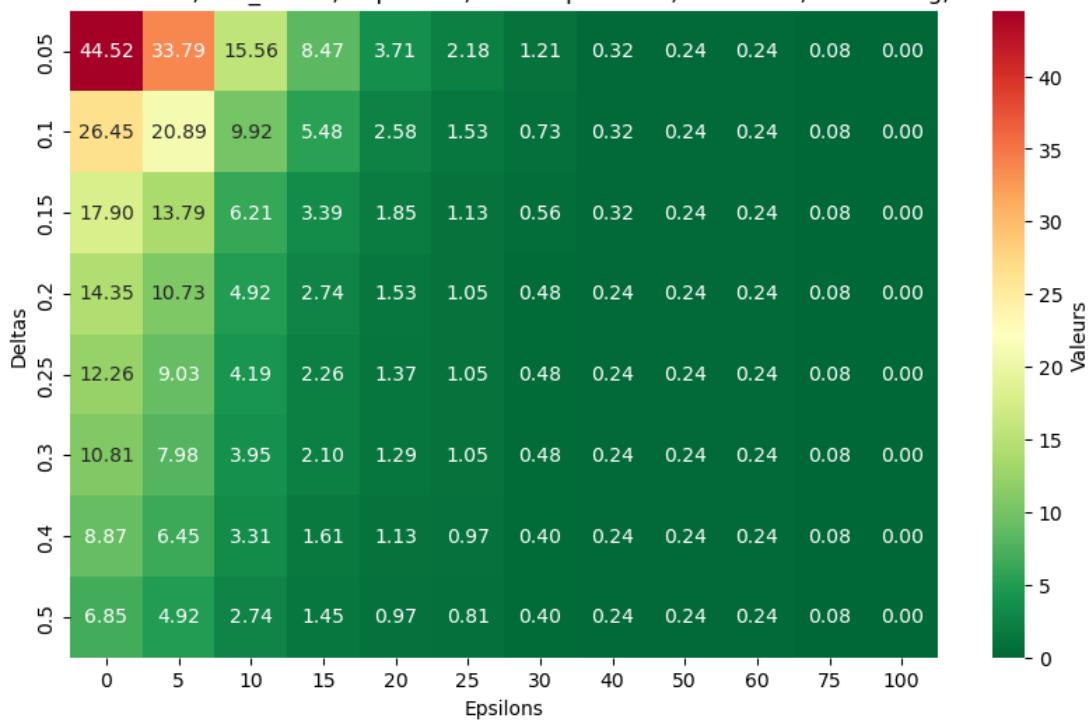
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs, 0, 20, " lichess positions, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, lichess positions, move=10, no castling, SF16

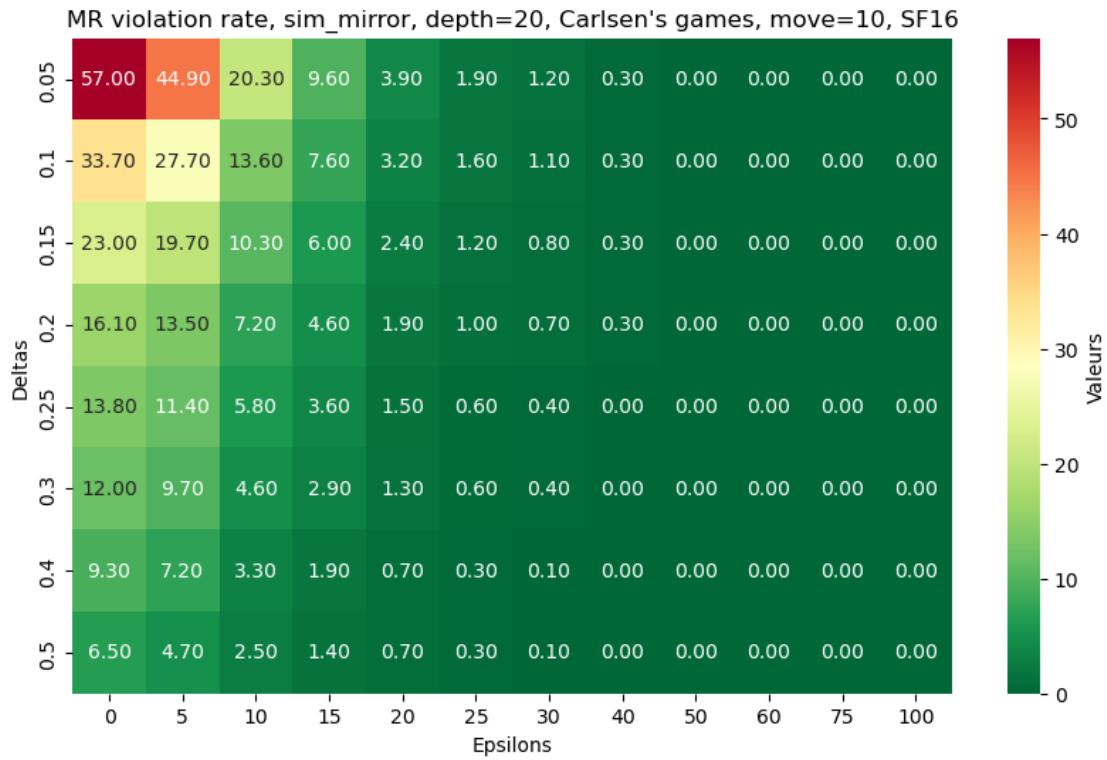


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 0, 20, " Carlsen's games, move=10, SF16")
```

move = 10



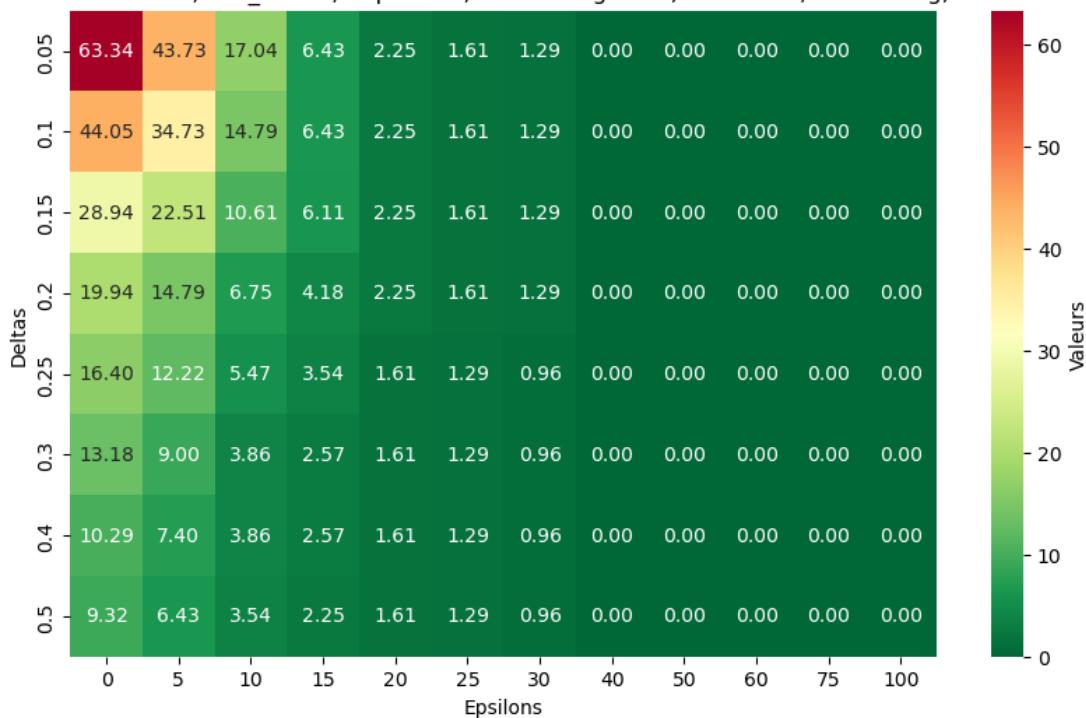
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs01610 = pickle.load(file)

      print('move = 10')
      tests(evs01610, 0, 20, "Carlsen's games, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_mirror, depth=20, Carlsen's games, move=10, no castling, SF16



Middlegames (move=20)

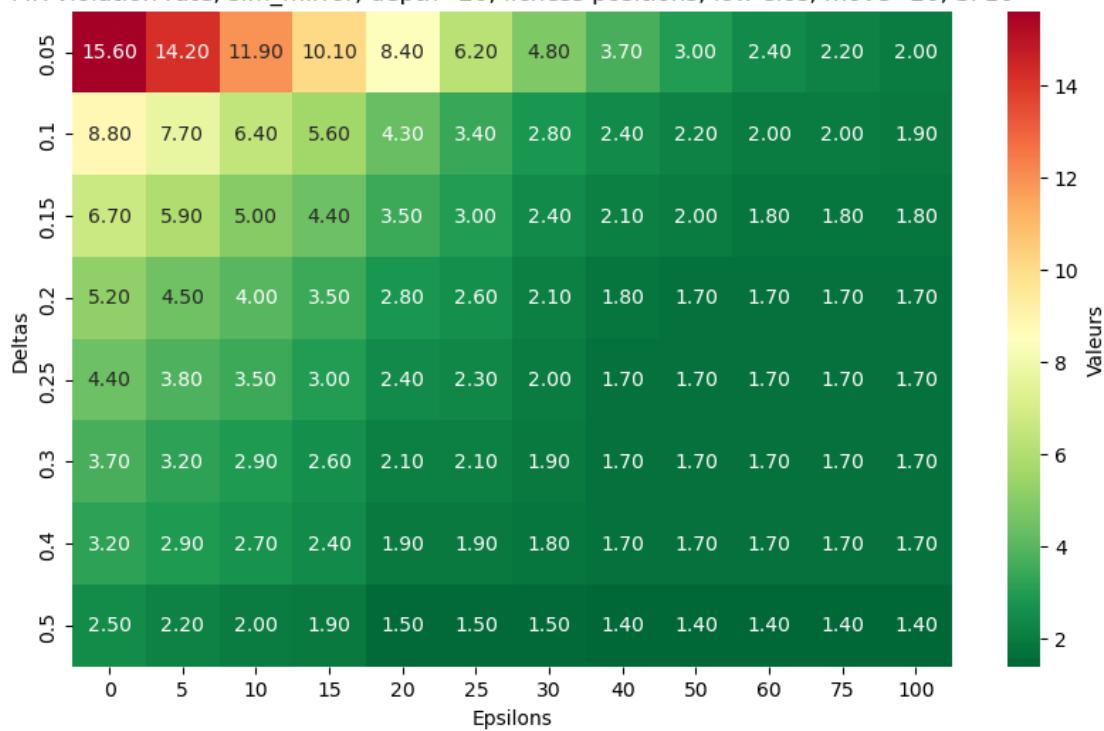
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-20sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 0, 20, " lichess positions, low elos, move=20, SF16")
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=20, SF16



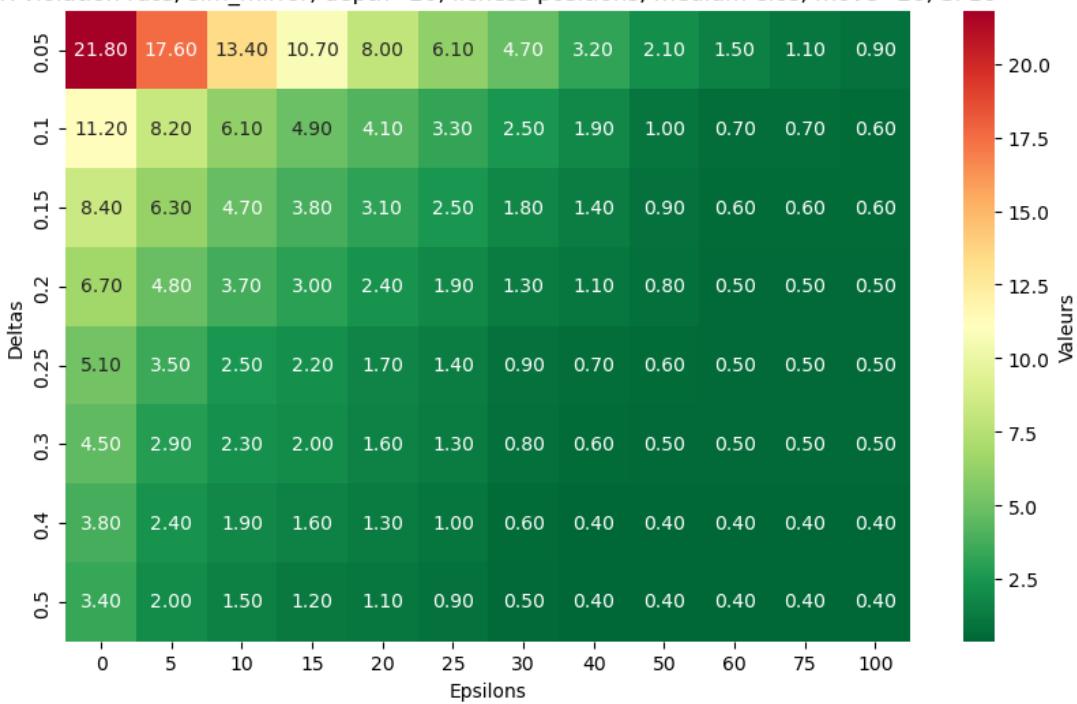
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-20sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs,0,20, " lichess positions, medium elos, move=20, SF16")
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=20, SF16

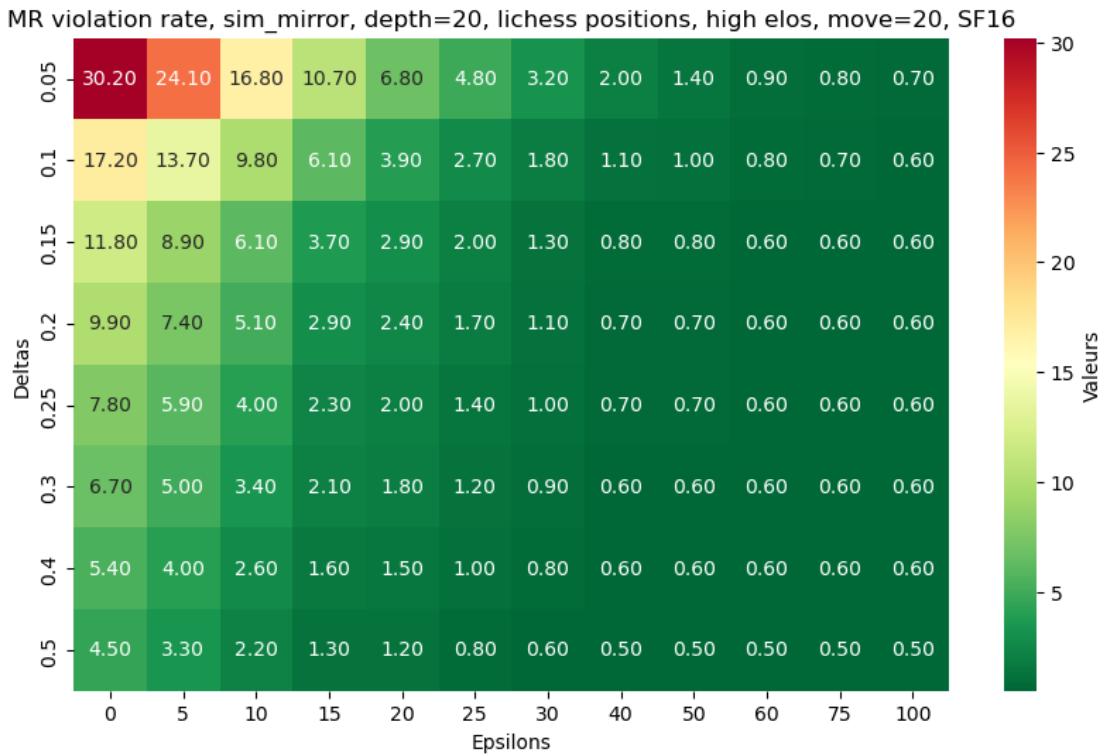


Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 0, 20, " lichess positions, high elos, move=20, SF16")
```

move = 20

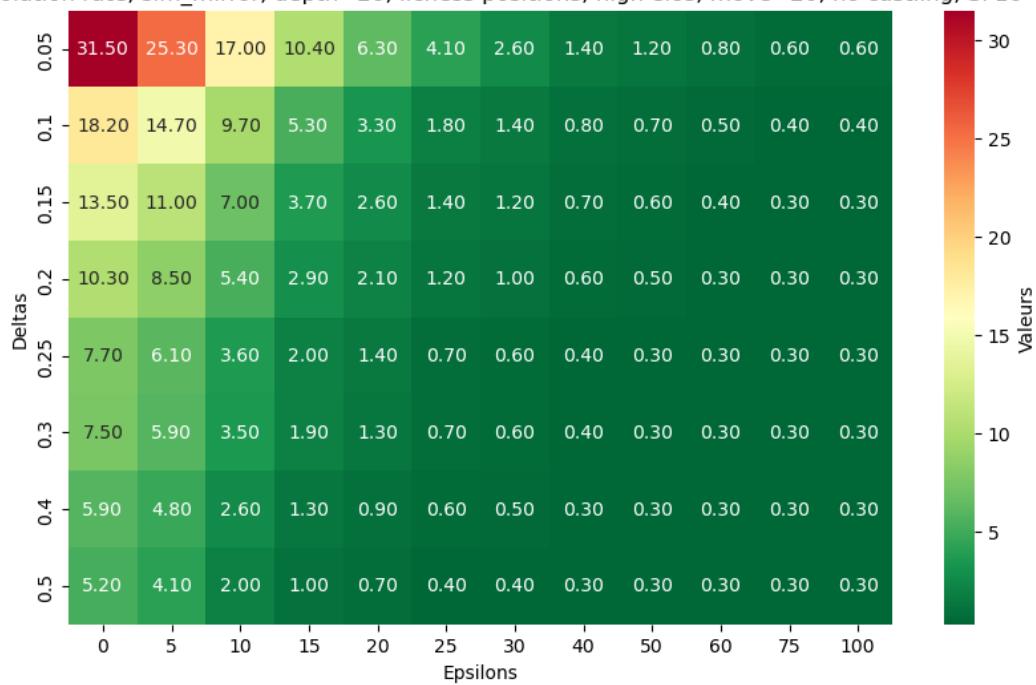


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.  
    ↪join('reals', 'ev_lichess2000-2500-20_nocastlingsim_mirror_d_20_v16.pkl')  
    with open(path, 'rb') as file:  
        evs = pickle.load(file)  
  
    print('move = 20')  
    tests(evs, 0, 20, " lichess positions, high elos, move=20, no castling, SF16")
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=20, no castling, SF16

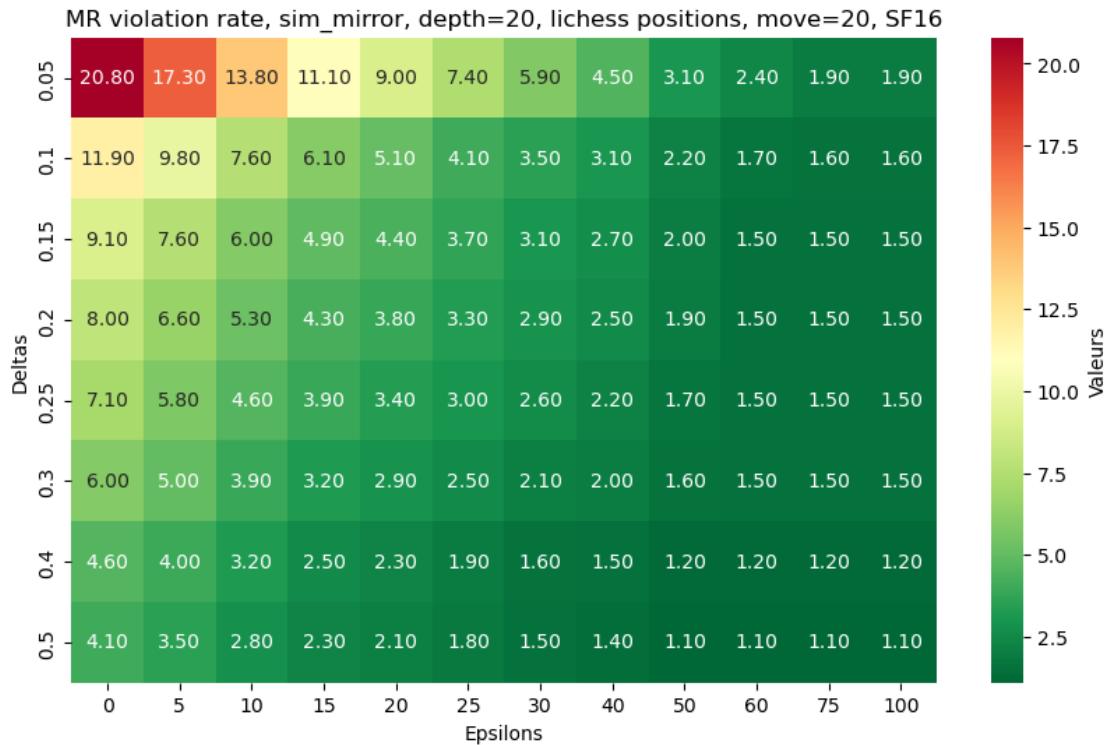


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-20sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, " lichess positions, move=20, SF16")
```

move = 20



Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-20_nocastlingsim_mirror_d_20_v16.  
    ↪pkl')  
with open(path, 'rb') as file:  
    evs = pickle.load(file)  
  
print('move = 20')  
tests(evs, 0, 20, " lichess positions, move=20, no castling, SF16")
```

move = 20

MR violation rate, sim_mirror, depth=20, lichess positions, move=20, no castling, SF16

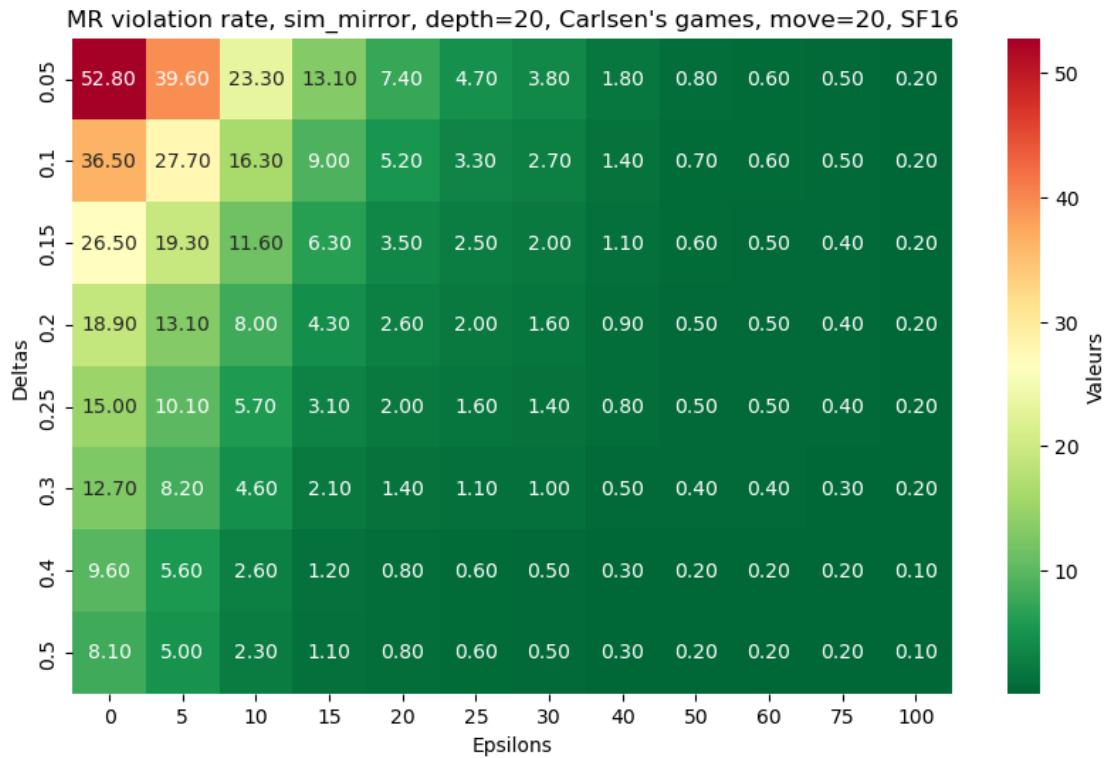


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 0, 20, " Carlsen's games, move=20, SF16")
```

move = 20



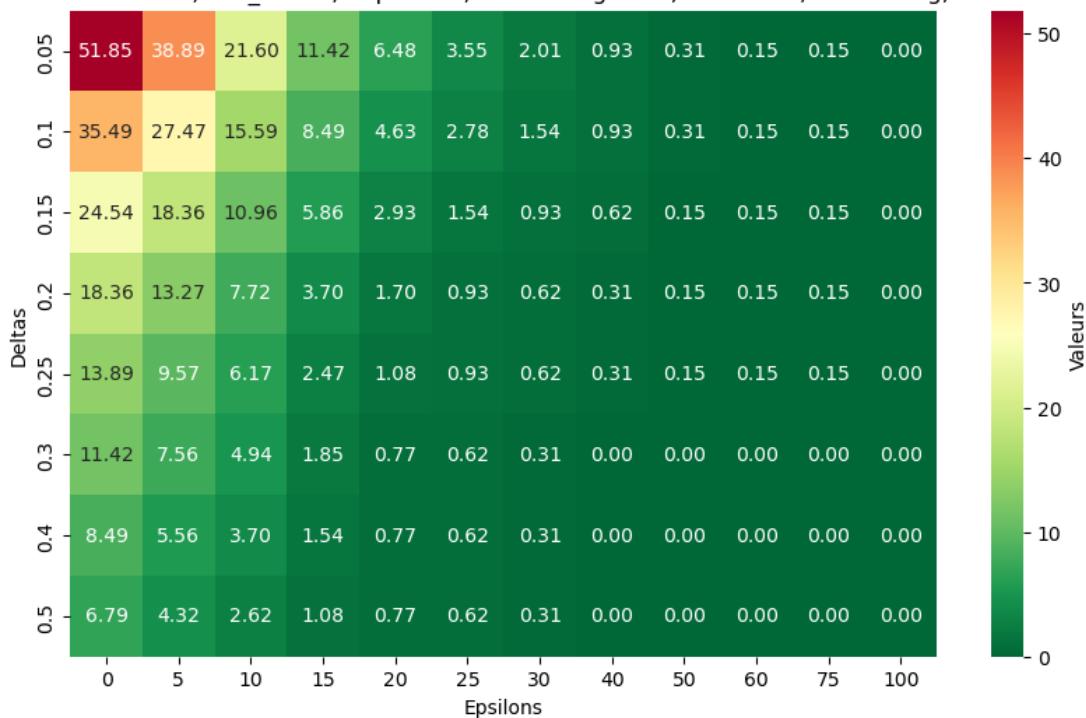
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs01620 = pickle.load(file)

      print('move = 20')
      tests(evs01620, 0, 20, "Carlsen's games, move=20, no castling, SF16")
```

move = 20

MR violation rate, sim_mirror, depth=20, Carlsen's games, move=20, no castling, SF16



Middle/Endgames (move=30)

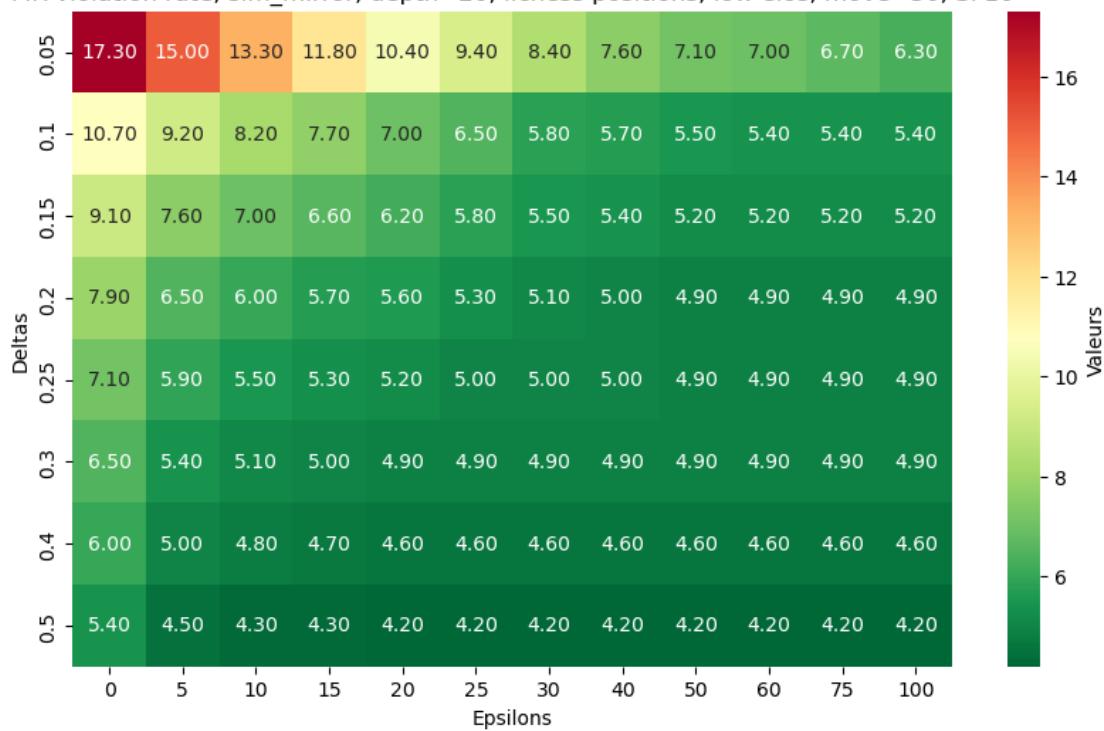
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-30sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 0, 20, " lichess positions, low elos, move=30, SF16")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=30, SF16



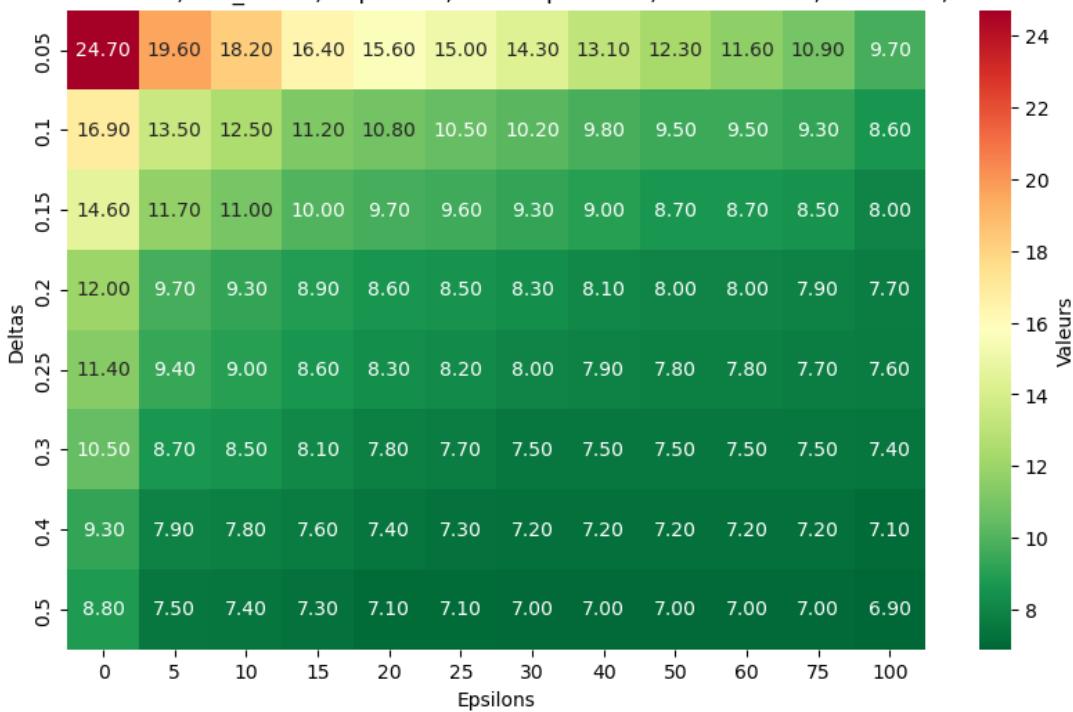
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-30sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs,0,20, " lichess positions, medium elos, move=30, SF16")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=30, SF16

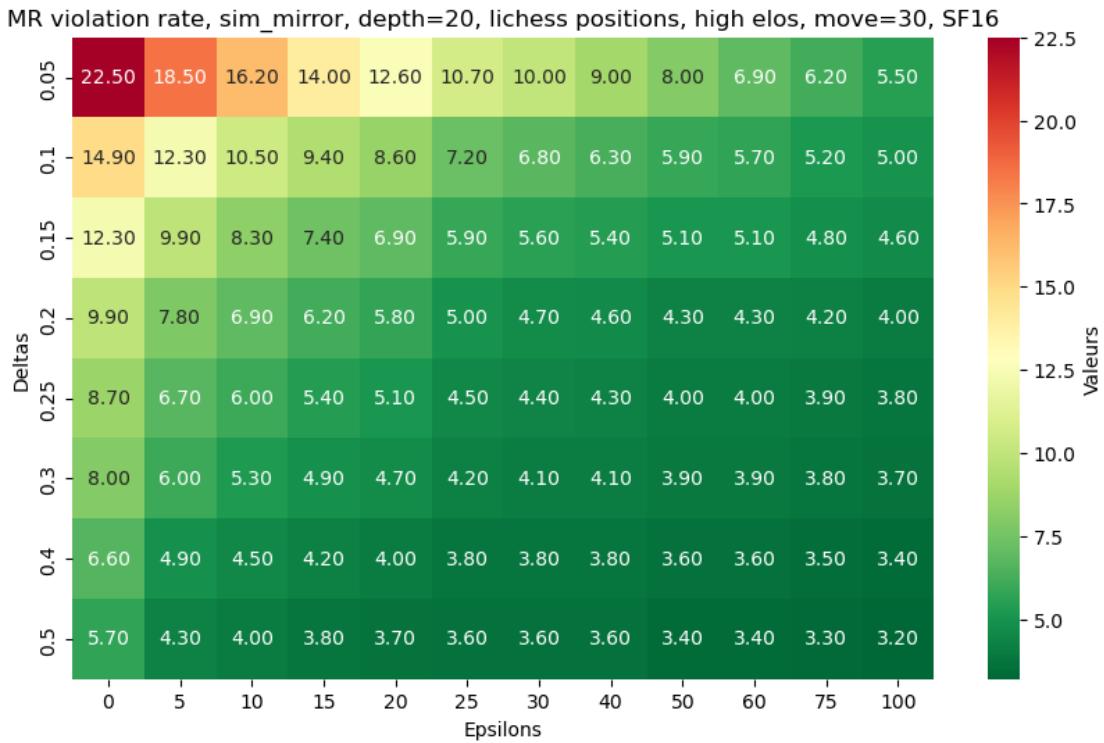


Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, high elos, move=30, SF16")
```

move = 30

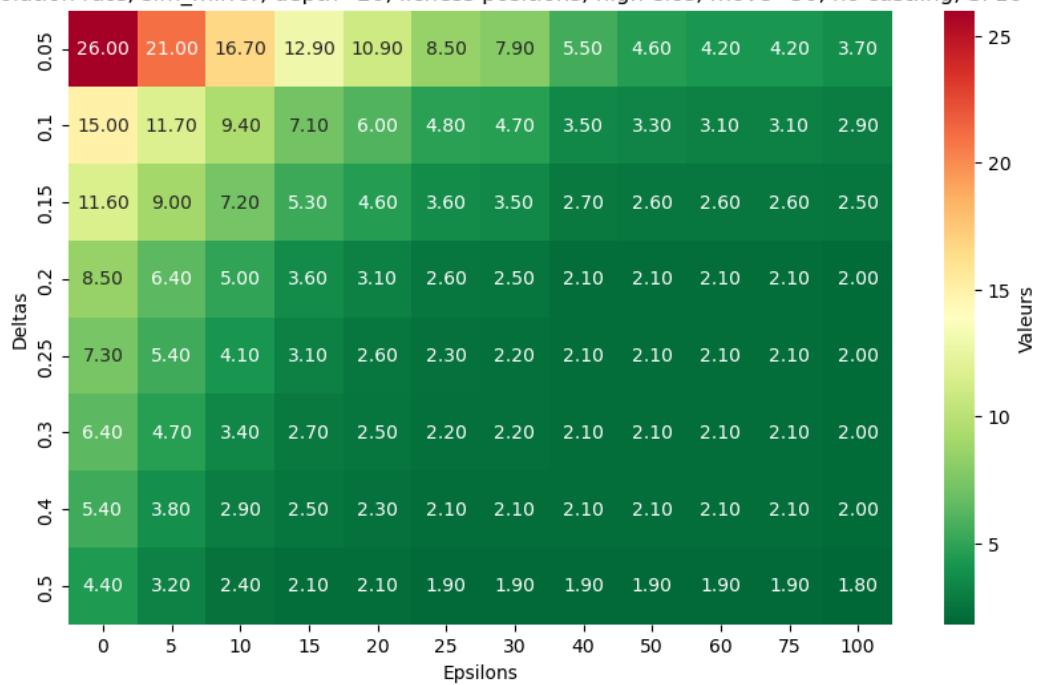


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.  
    ↪join('reals', 'ev_lichess2000-2500-30_nocastlingsim_mirror_d_20_v16.pkl')  
with open(path, 'rb') as file:  
    evs = pickle.load(file)  
  
print('move = 30')  
tests(evs, 0, 20, " lichess positions, high elos, move=30, no castling, SF16")
```

move = 30

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=30, no castling, SF16

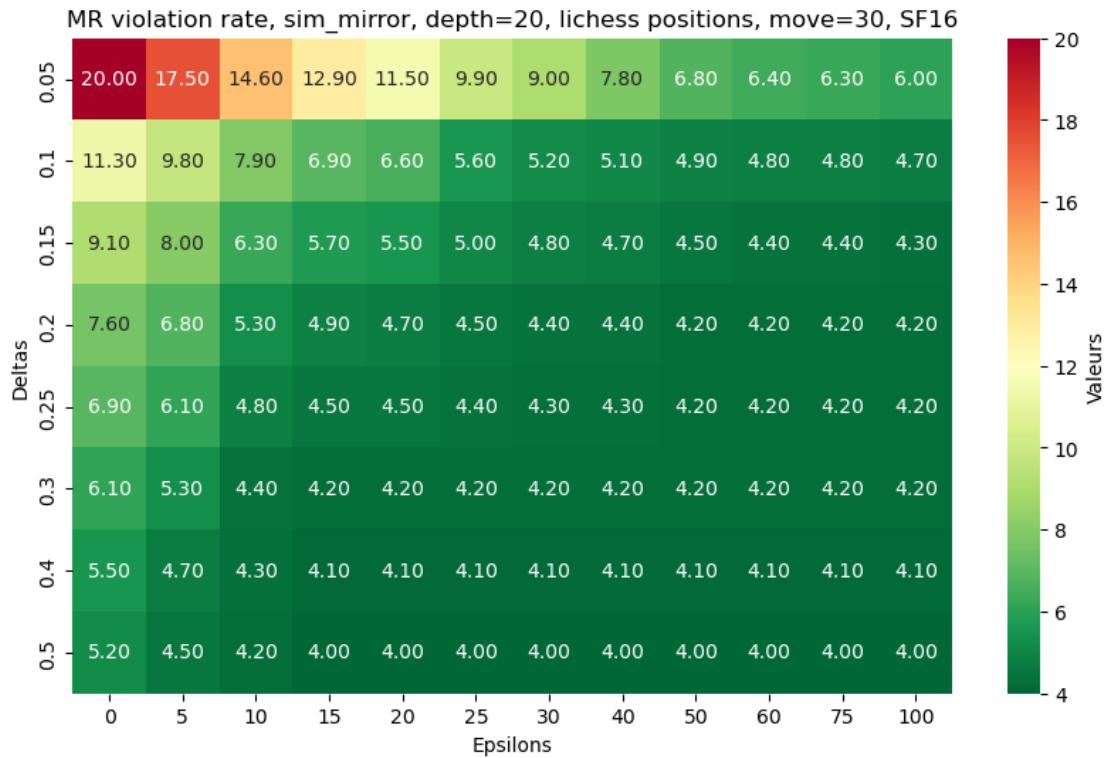


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " lichess positions, move=30, SF16")
```

move = 30

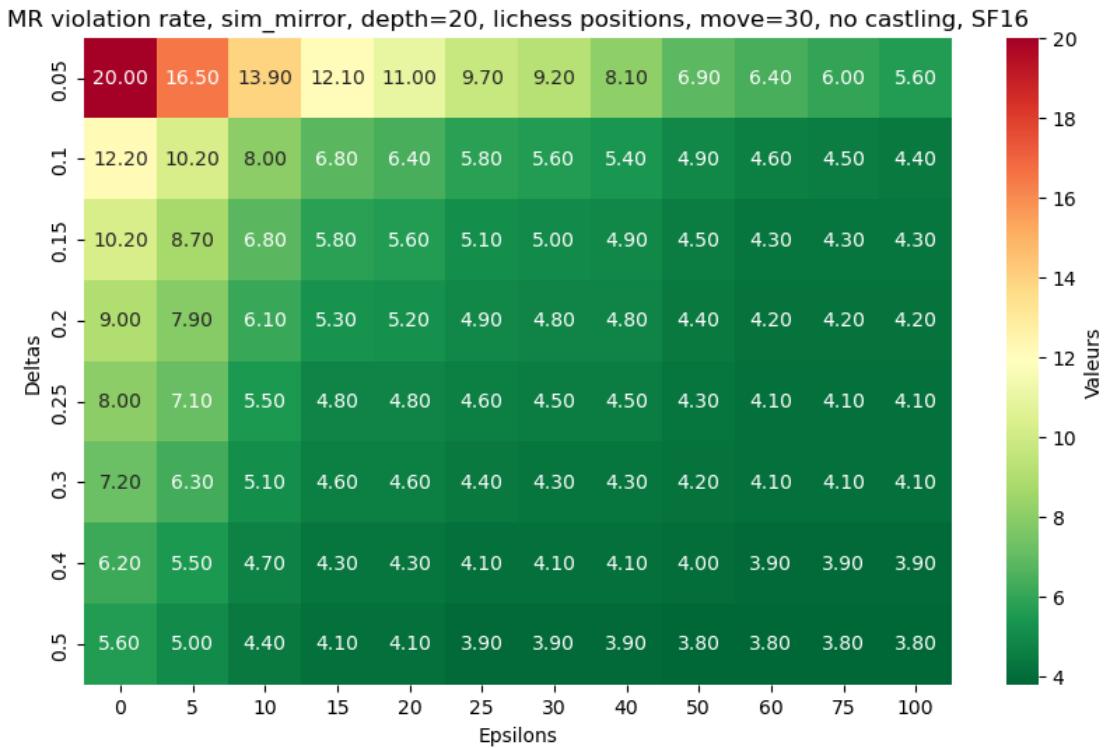


Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 0, 20, " lichess positions, move=30, no castling, SF16")
```

move = 30

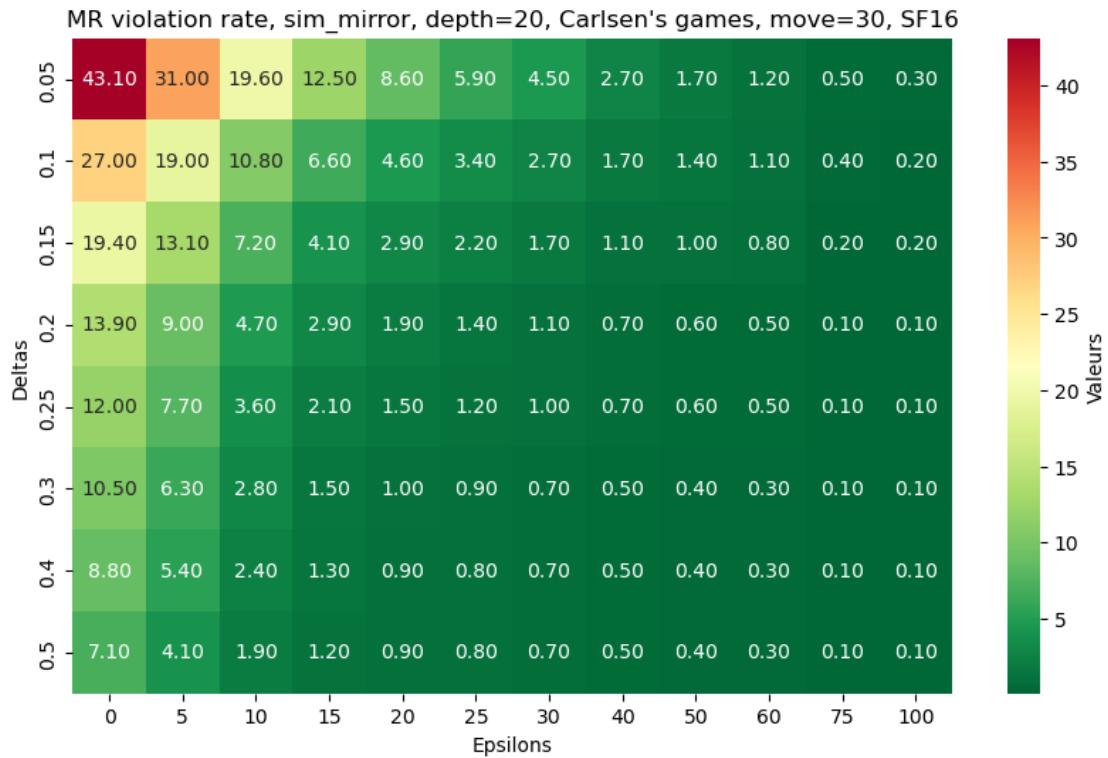


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 0, 20, " Carlsen's games, move=30, SF16")
```

move = 30



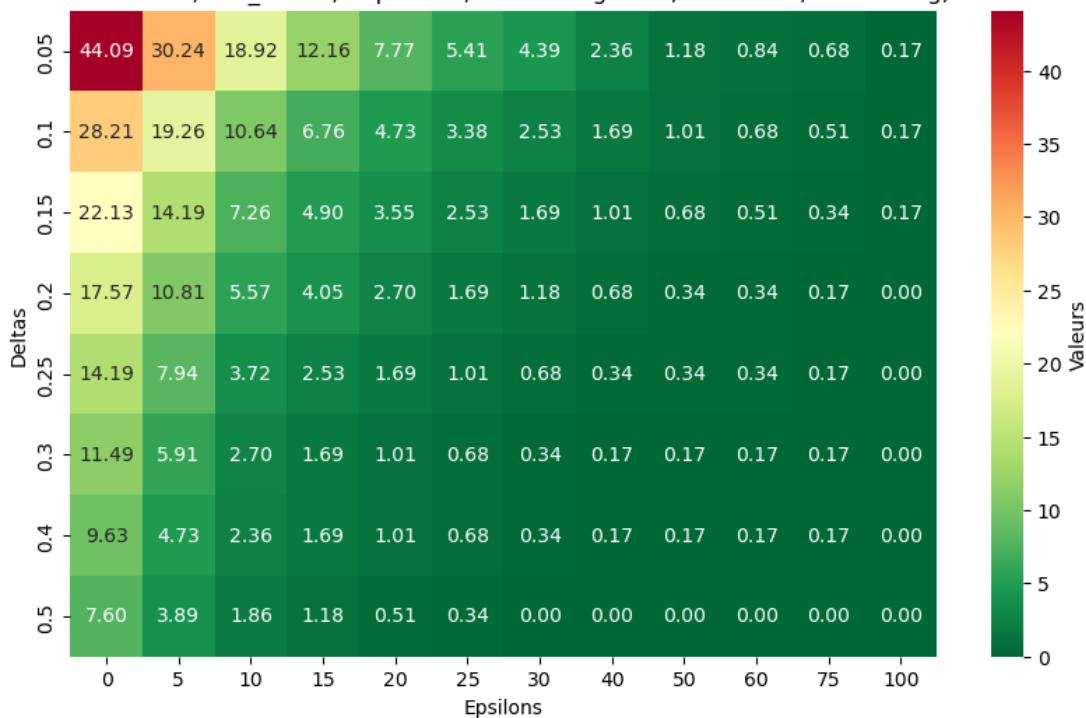
Carlsen's games, no castling avaialble

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs01630 = pickle.load(file)

      print('move = 30')
      tests(evs01630, 0, 20, "Carlsen's games, move=30, no castling, SF16")
```

move = 30

MR violation rate, sim_mirror, depth=20, Carlsen's games, move=30, no castling, SF16



Endgames (move=40)

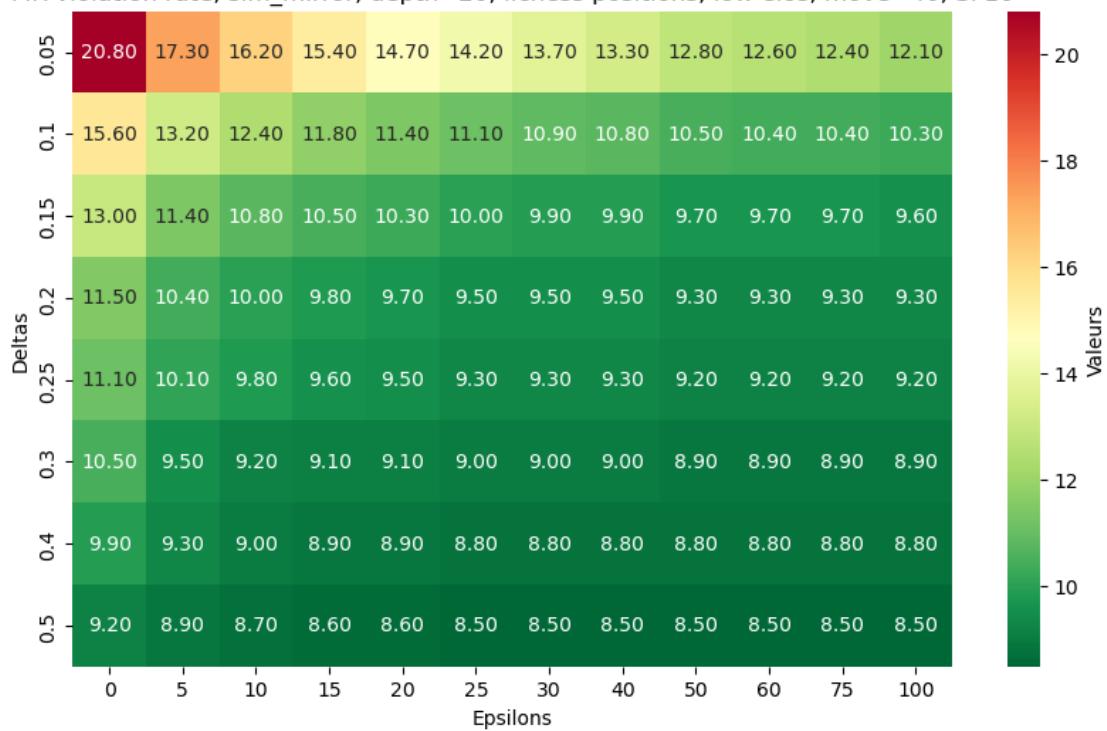
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-40sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 0, 20, " lichess positions, low elos, move=40, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, low elos, move=40, SF16



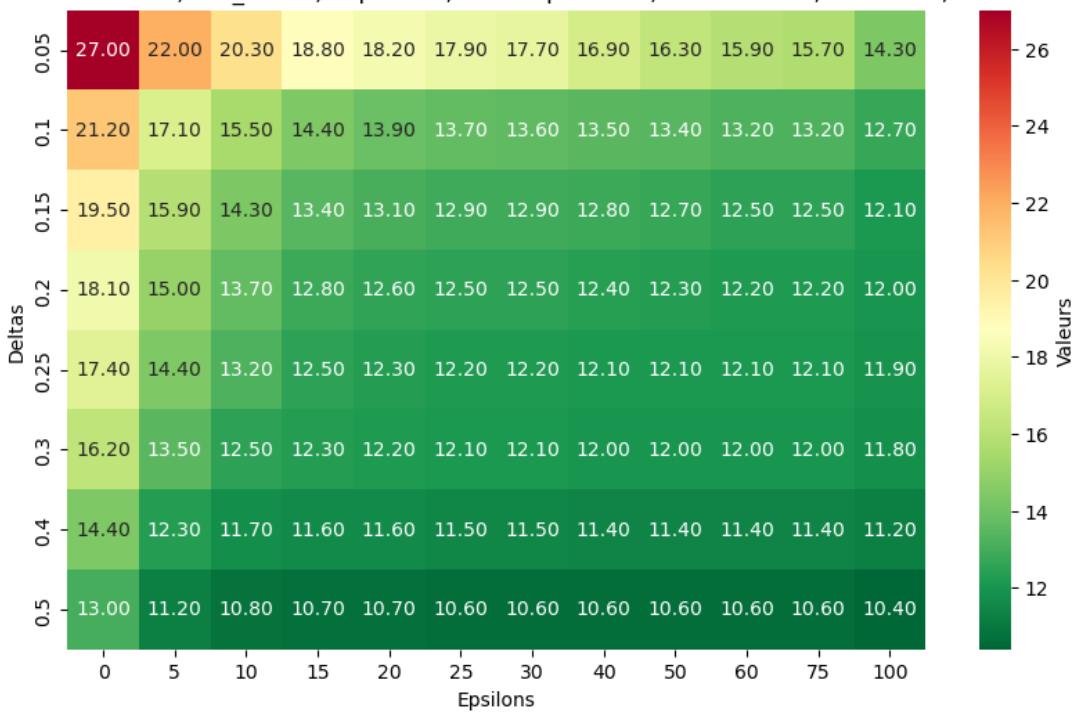
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-40sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs,0,20, " lichess positions, medium elos, move=40, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, medium elos, move=40, SF16



Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " lichess positions, high elos, move=40, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=40, SF16



Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.  
    ↪join('reals', 'ev_lichess2000-2500-40_nocastlingsim_mirror_d_20_v16.pkl')  
with open(path, 'rb') as file:  
    evs = pickle.load(file)  
  
print('move = 40')  
tests(evs, 0, 20, " lichess positions, high elos, move=40, no castling, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, high elos, move=40, no castling, SF16



Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40sim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 0, 20, " lichess positions, move=40, SF16")
```

move = 40



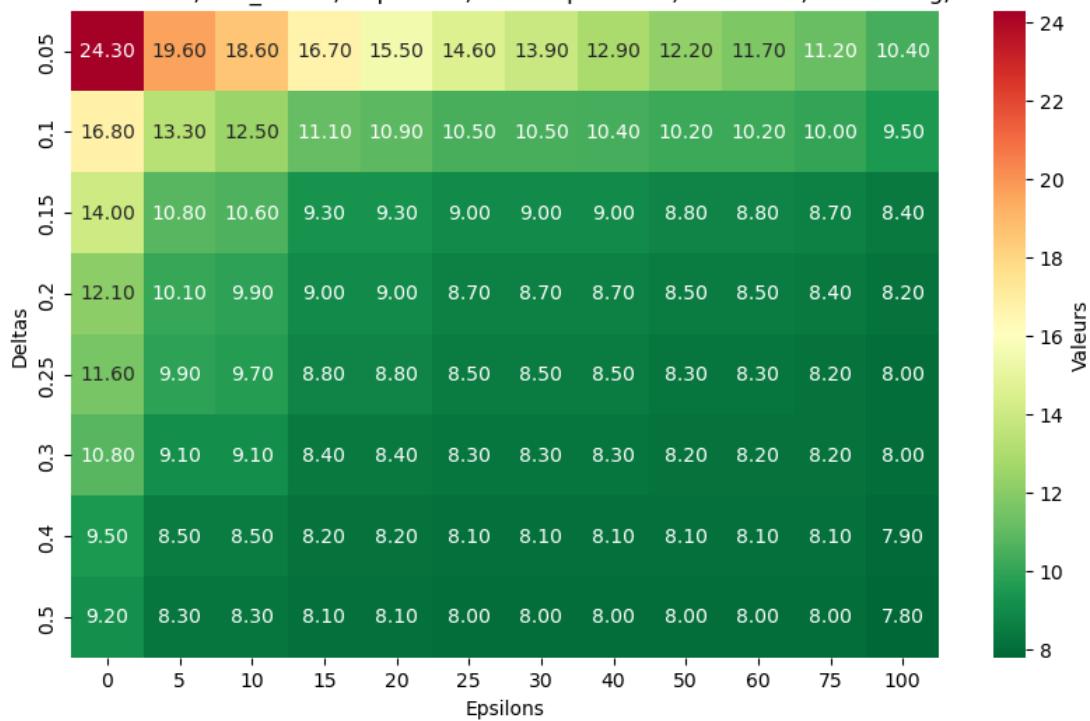
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 0, 20, " lichess positions, move=40, no castling, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, lichess positions, move=40, no castling, SF16

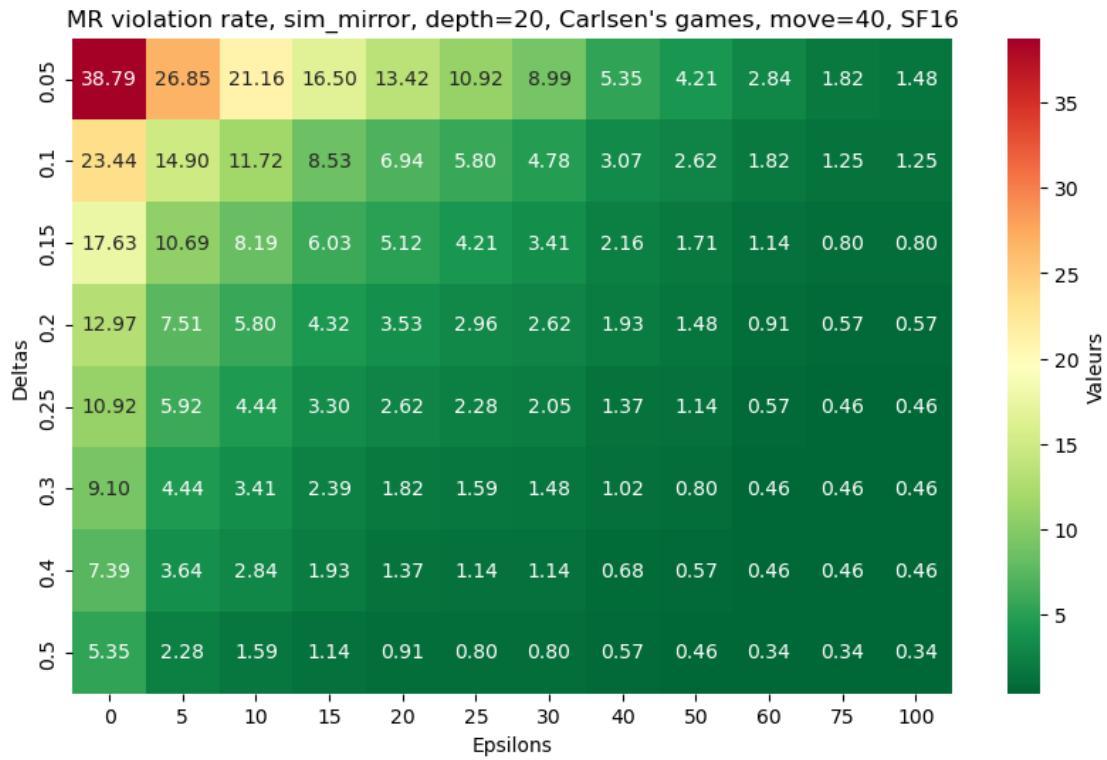


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40sim_mirror_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 0, 20, " Carlsen's games, move=40, SF16")
```

move = 40



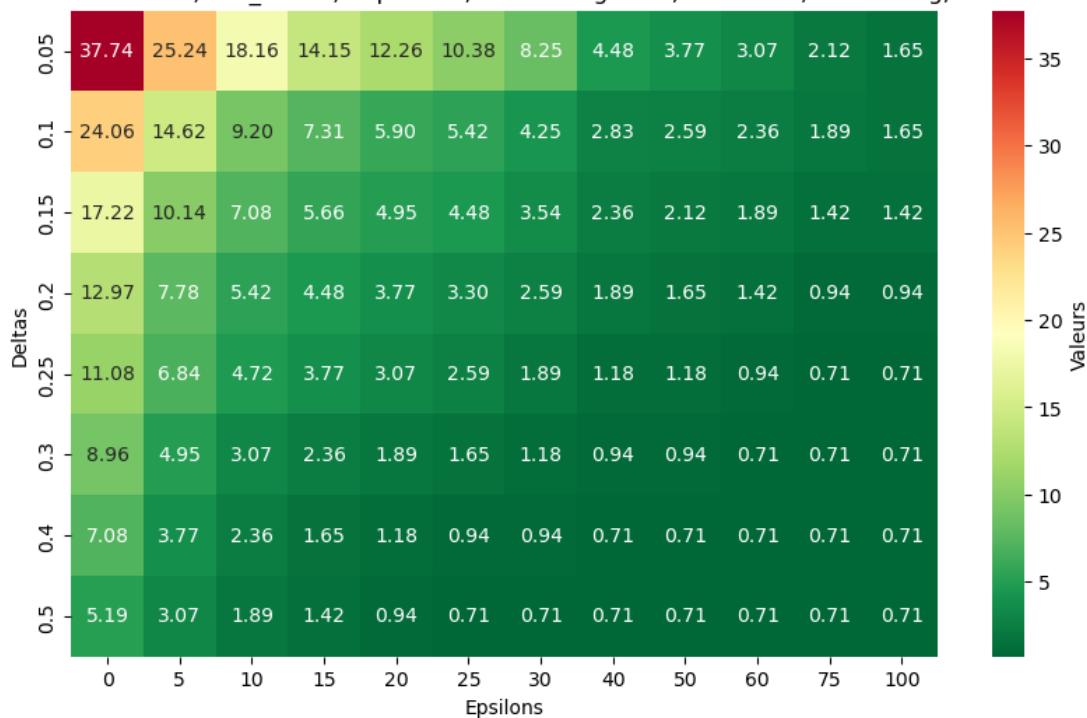
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40_nocastlingsim_mirror_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs01640 = pickle.load(file)

      print('move = 40')
      tests(evs01640, 0, 20, "Carlsen's games, move=40, no castling, SF16")
```

move = 40

MR violation rate, sim_mirror, depth=20, Carlsen's games, move=40, no castling, SF16



0.12.2 sim_axis

Openings (move=10)

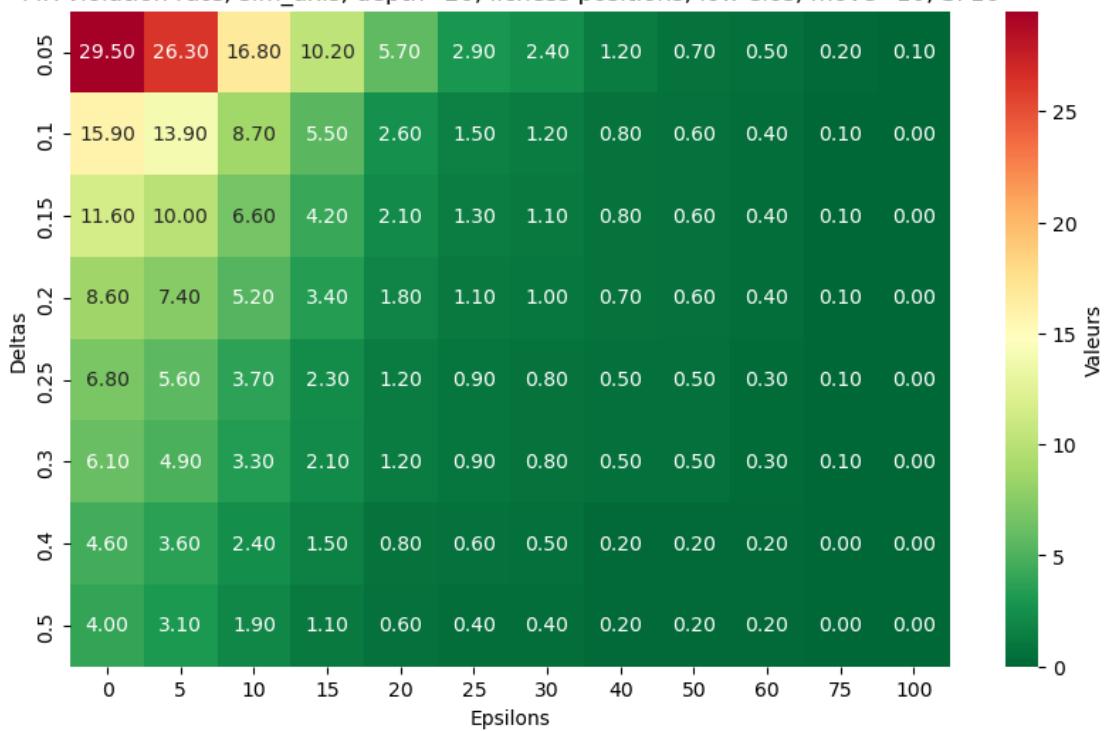
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1000-1500-10sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,1,20, " lichess positions, low elos, move=10, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=10, SF16



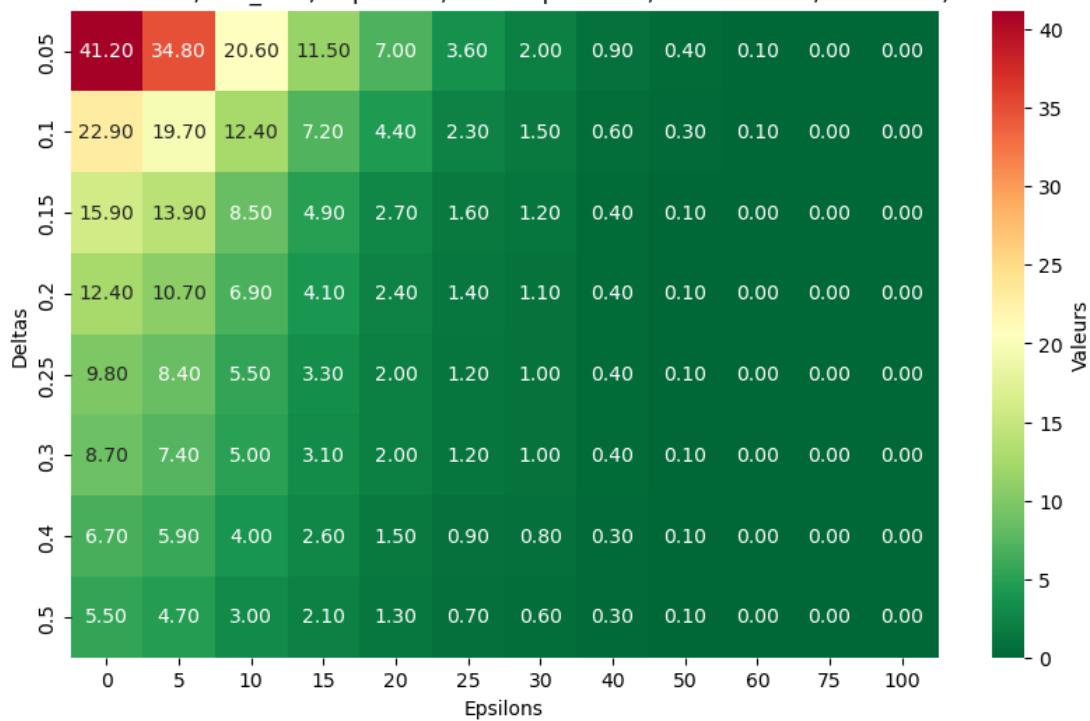
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-10sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,1,20, " lichess positions, medium elos, move=10, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=10, SF16



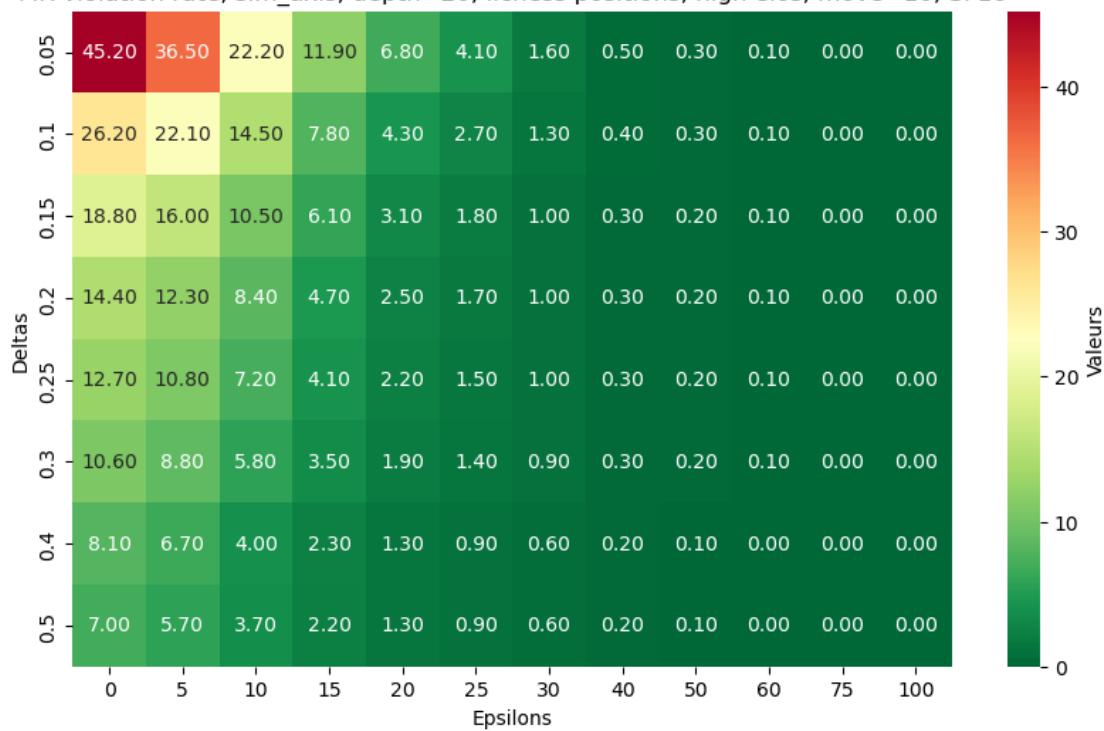
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs, 1, 20, " lichess positions, high elos, move=10, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=10, SF16



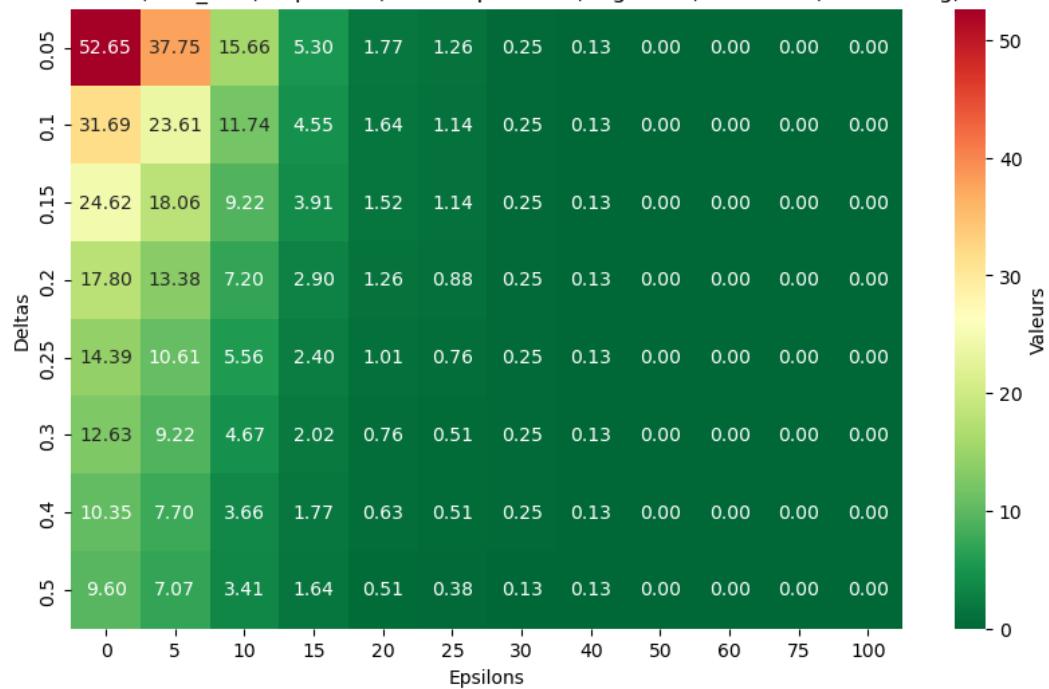
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-10_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs, 1, 20, " lichess positions, high elos, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=10, no castling, SF16

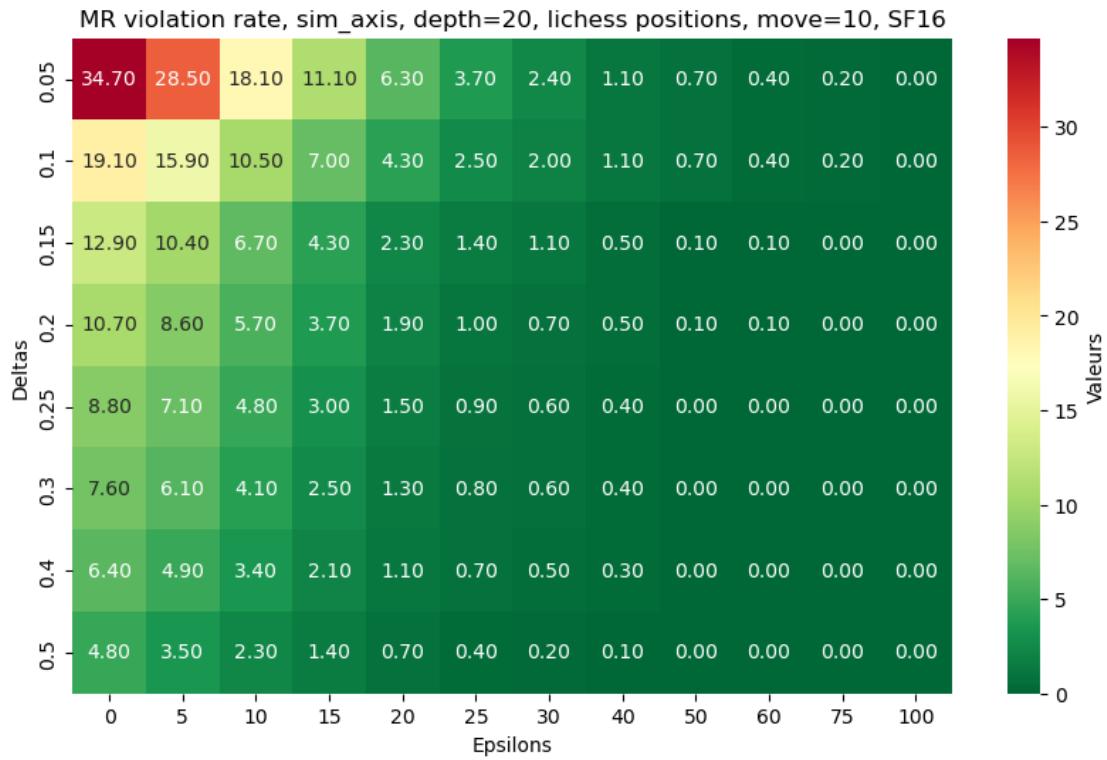


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-10sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, ", lichess positions, move=10, SF16")
```

move = 10



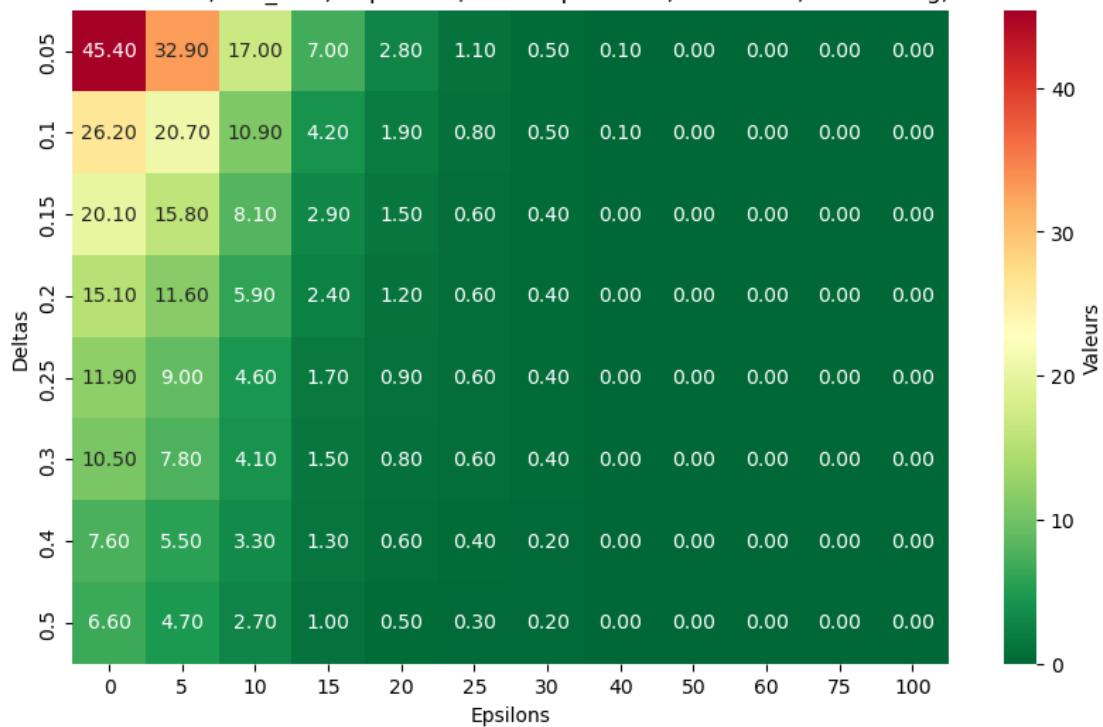
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals','ev_lichessAllElos-10_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 10')
      tests(evs,1,20, " lichess positions, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, lichess positions, move=10, no castling, SF16

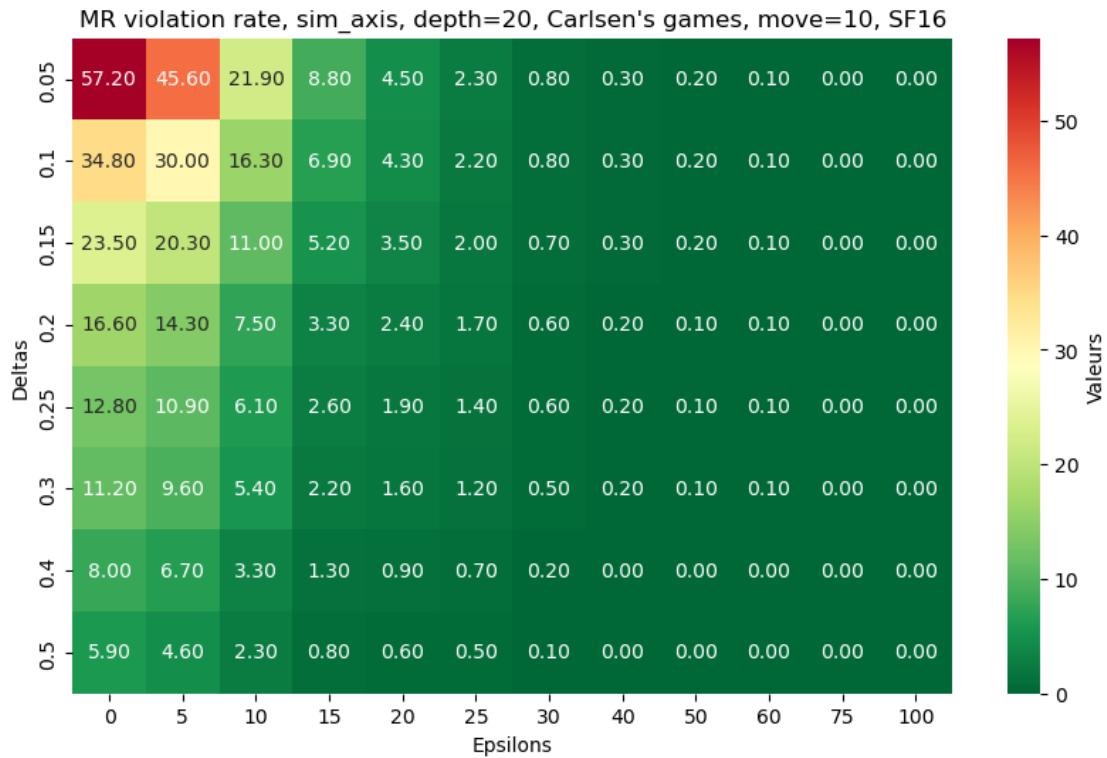


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 10')
tests(evs, 1, 20, " Carlsen's games, move=10, SF16")
```

move = 10



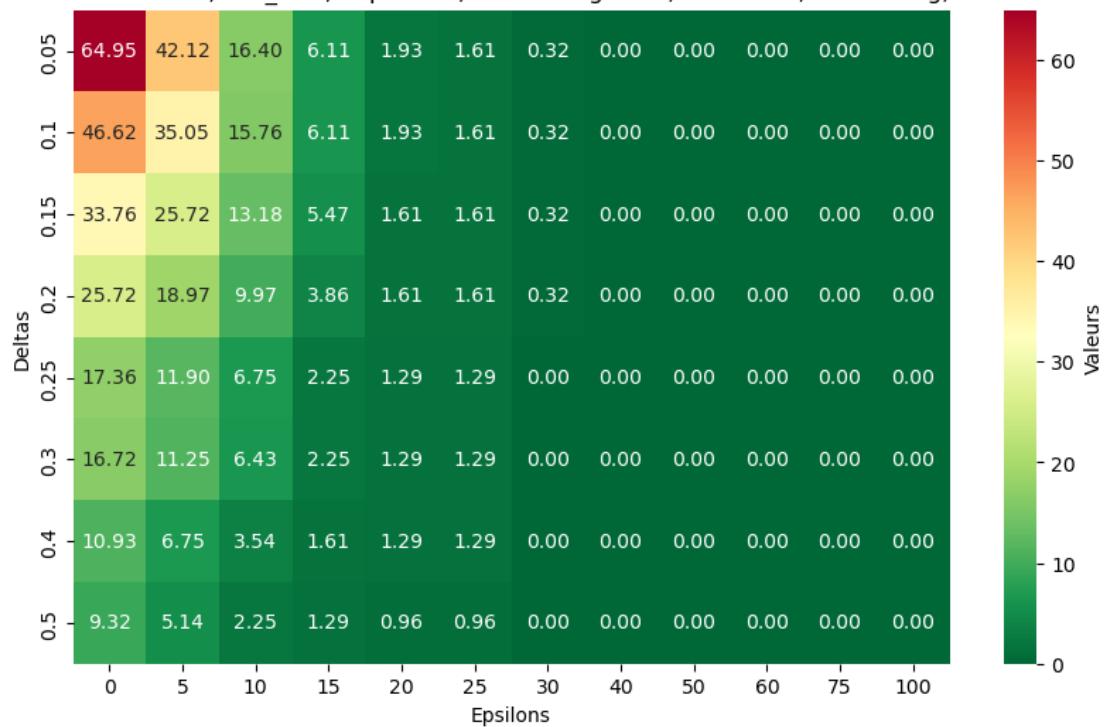
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-10_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs11610 = pickle.load(file)

      print('move = 10')
      tests(evs11610, 1, 20, " Carlsen's games, move=10, no castling, SF16")
```

move = 10

MR violation rate, sim_axis, depth=20, Carlsen's games, move=10, no castling, SF16



Middlegames (move=20)

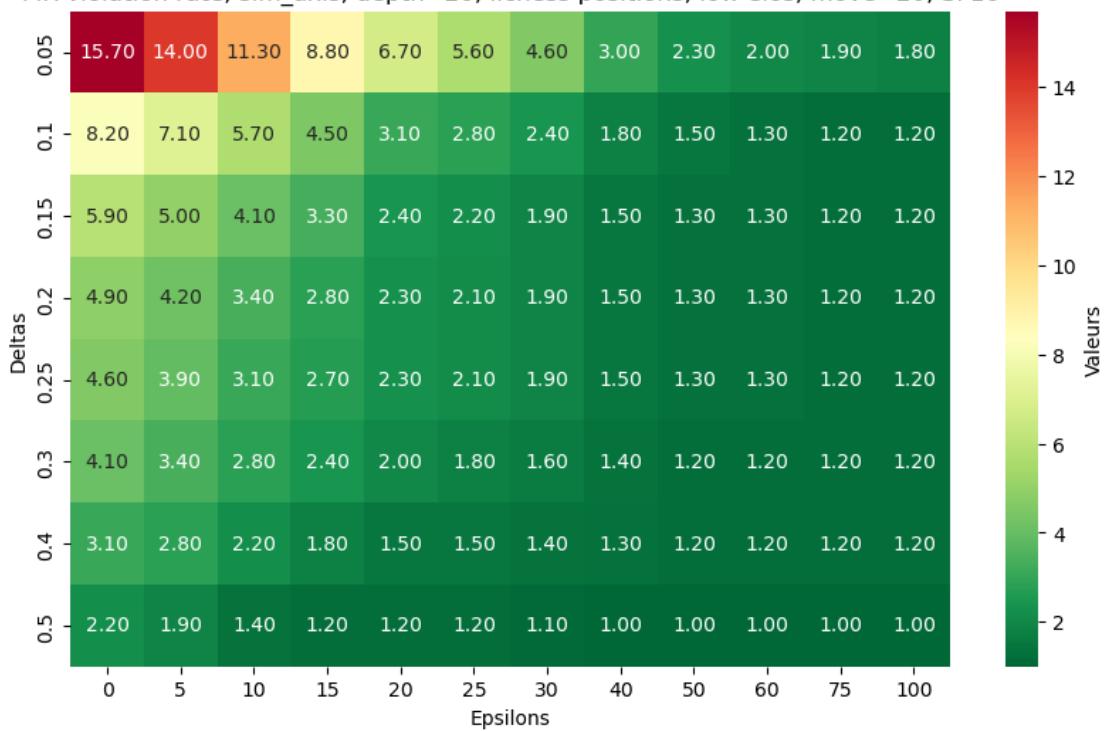
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-20sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " lichess positions, low elos, move=20, SF16")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=20, SF16



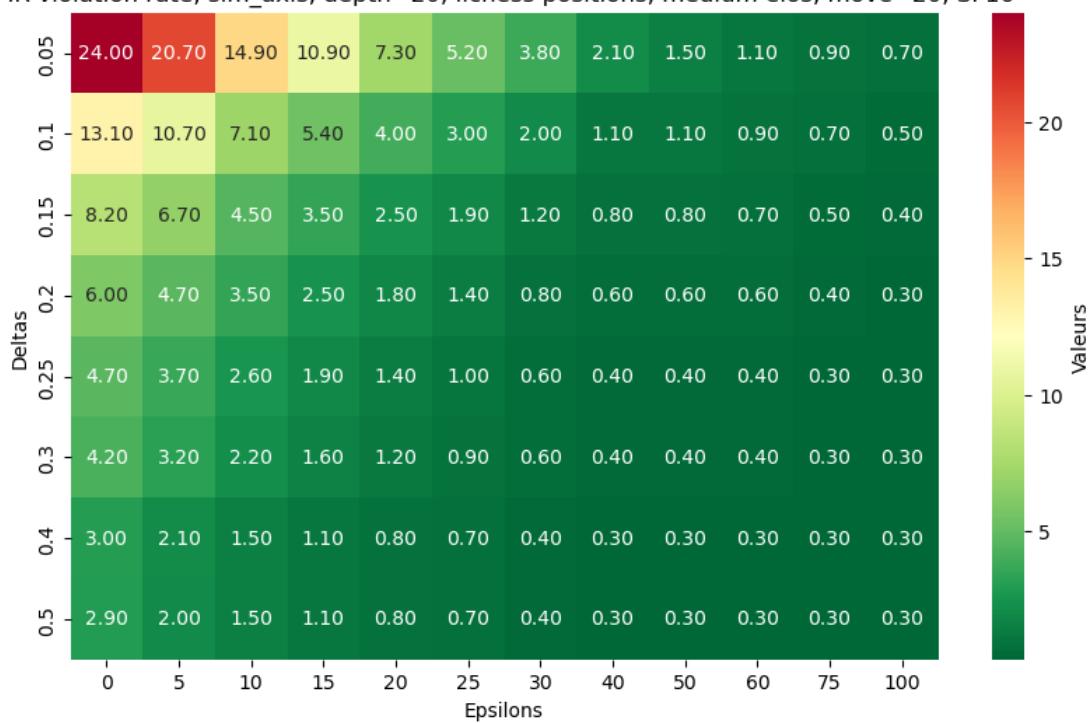
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-20sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs,1,20, " lichess positions, medium elos, move=20, SF16")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=20, SF16



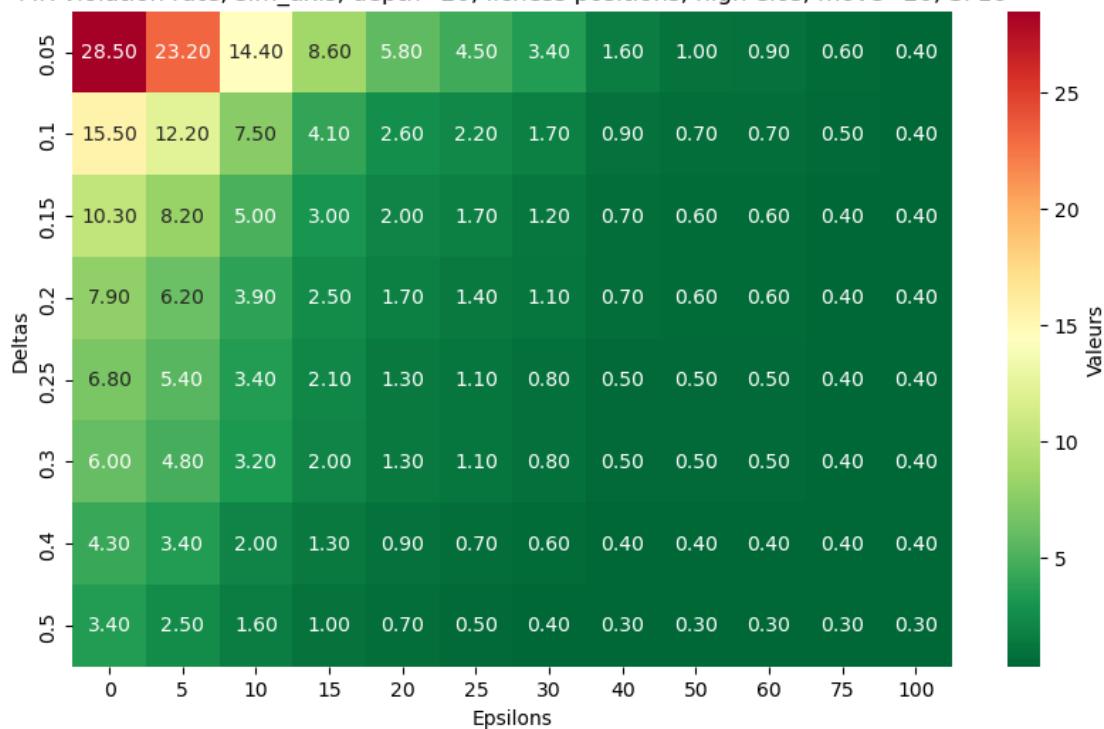
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-20sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " lichess positions, high elos, move=20, SF16")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=20, SF16



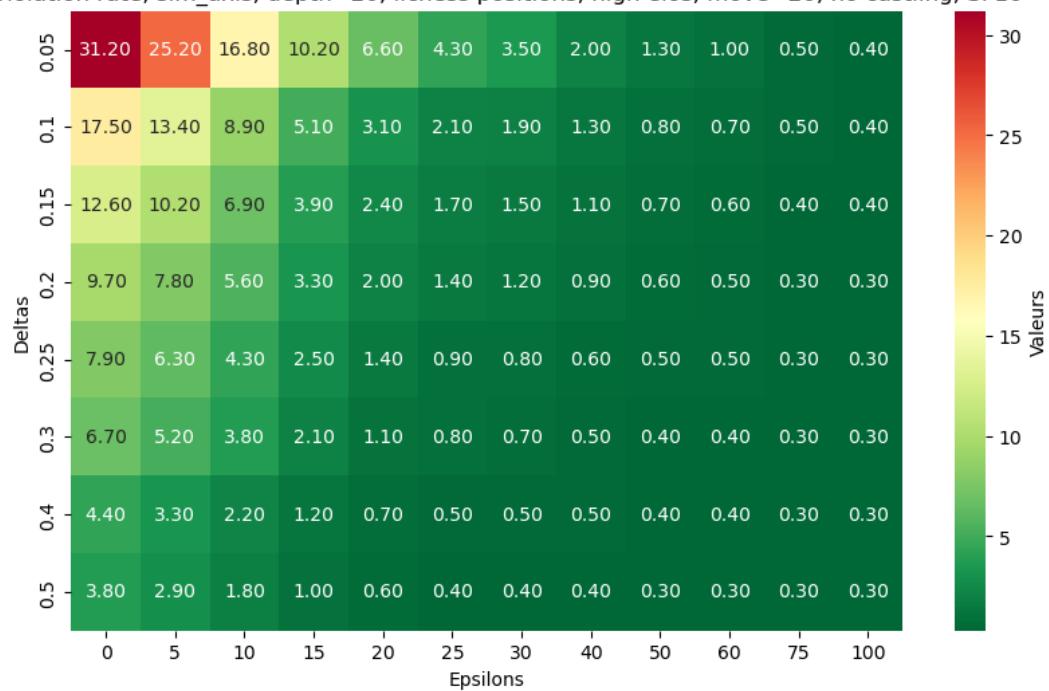
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals','ev_lichess2000-2500-20_nocastlingsim_axis_d_20_v16.pkl')
       with open(path, 'rb') as file:
           evs = pickle.load(file)

       print('move = 20')
       tests(evs,1,20, " lichess positions, high elos, move=20, no castling, SF16")
```

move = 20

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=20, no castling, SF16

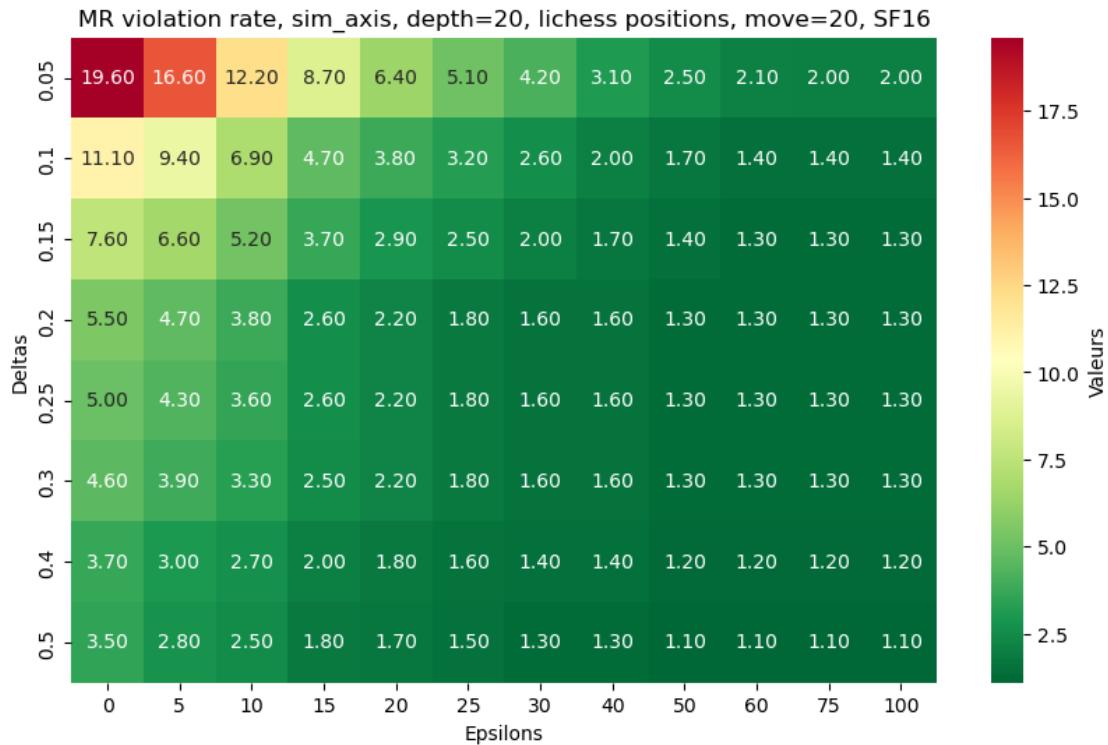


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-20sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 20')
      tests(evs, 1, 20, " lichess positions, move=20, SF16")
```

move = 20

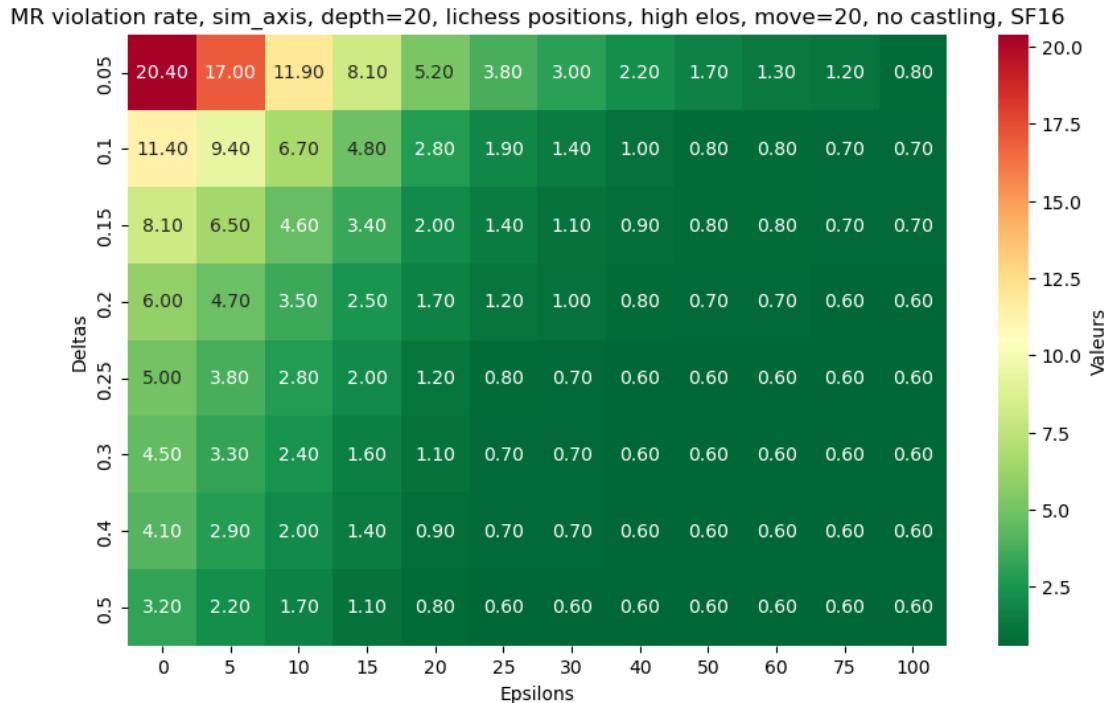


Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-20_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

          print('move = 20')
          tests(evs, 1, 20, " lichess positions, high elos, move=20, no castling, SF16")
```

move = 20

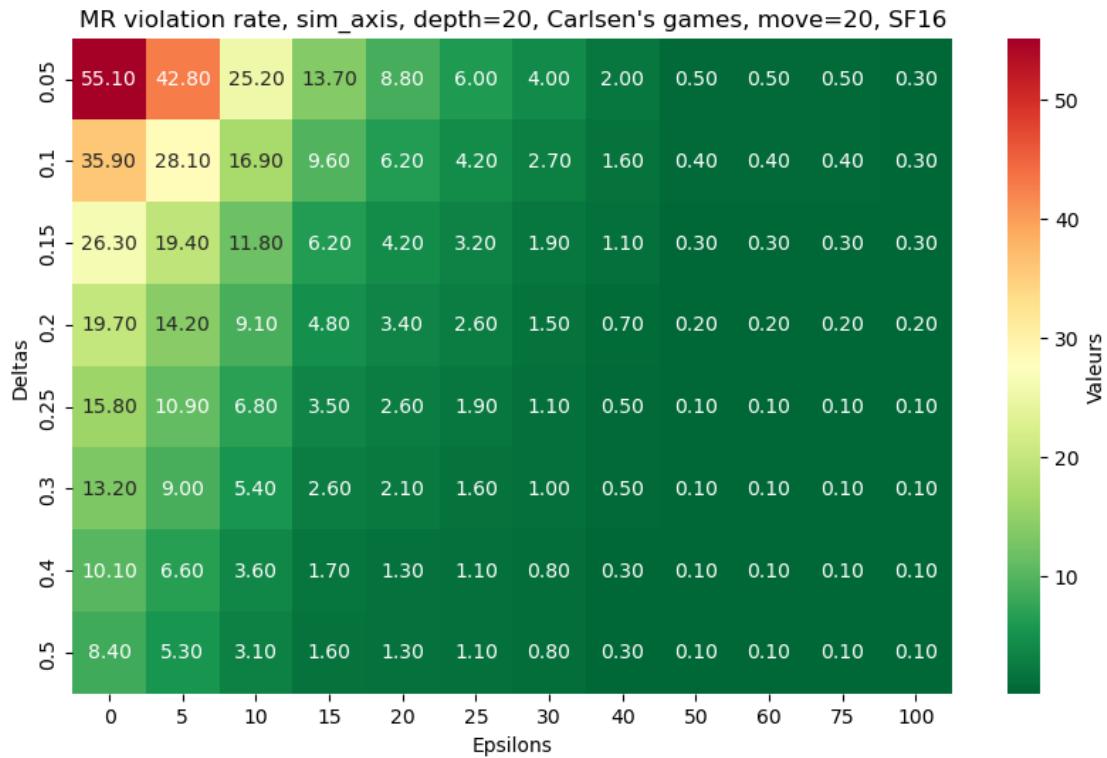


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 20')
tests(evs, 1, 20, " Carlsen's games, move=20, SF16")
```

move = 20



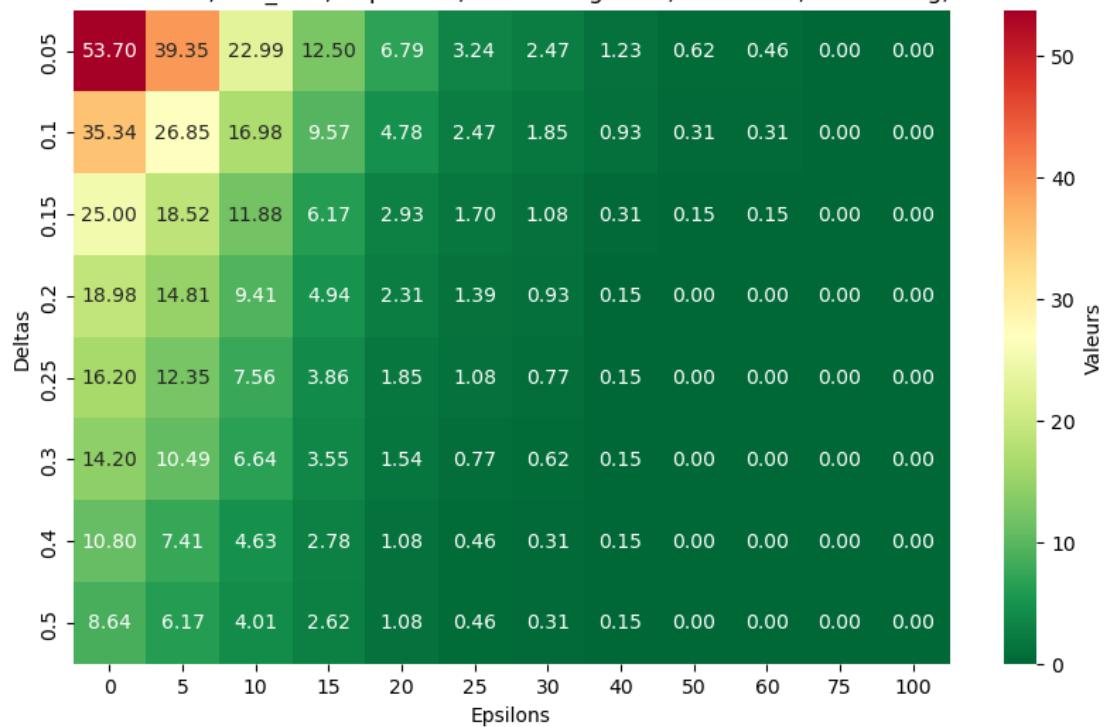
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-20_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs11620 = pickle.load(file)

      print('move = 20')
      tests(evs11620, 1, 20, " Carlsen's games, move=20, no castling, SF16")
```

move = 20

MR violation rate, sim_axis, depth=20, Carlsen's games, move=20, no castling, SF16



Middle/Endgames (move=30)

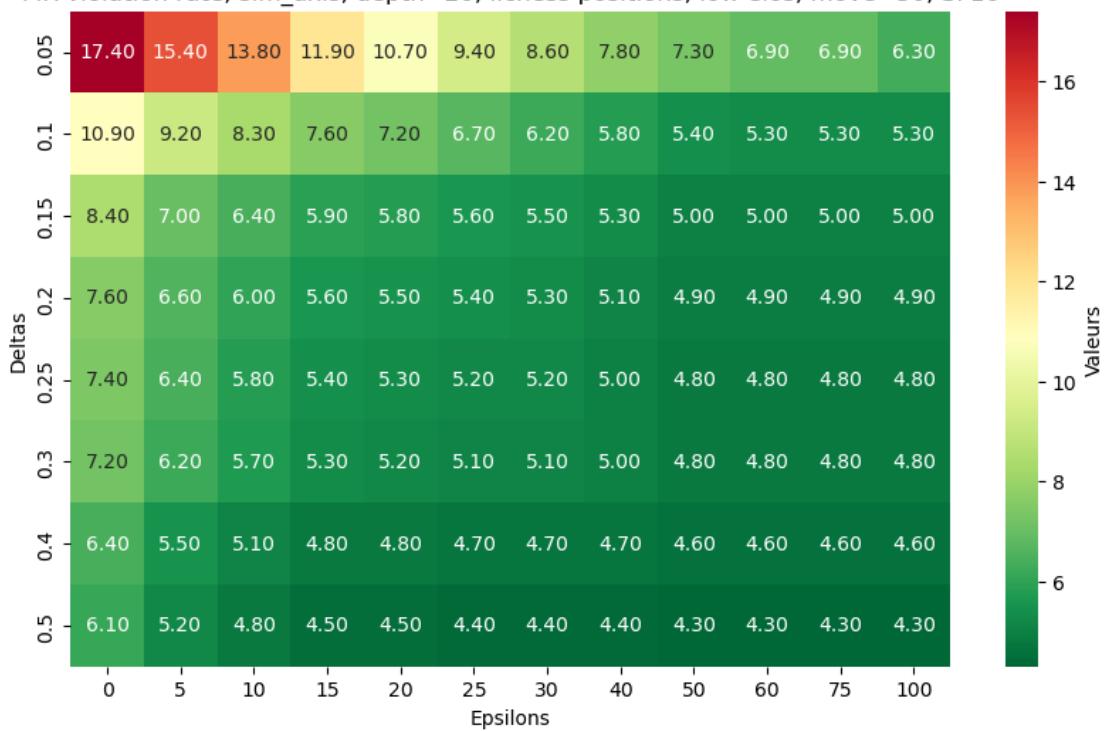
Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-30sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 1, 20, " lichess positions, low elos, move=30, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, low elos, move=30, SF16



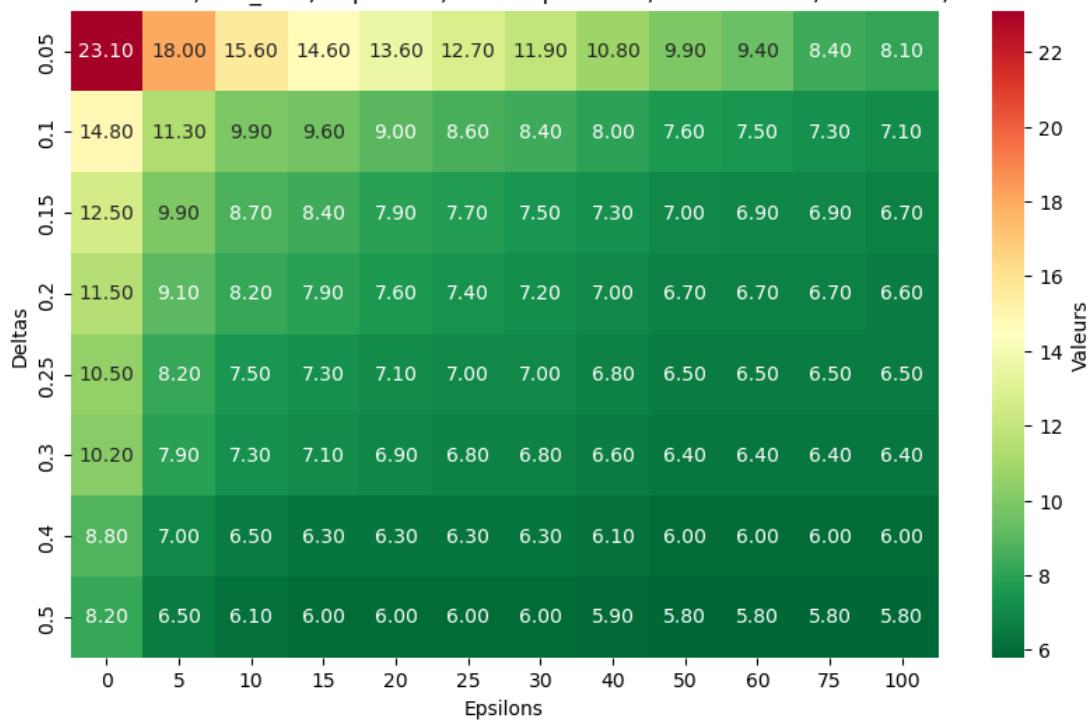
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-30sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs,1,20, " lichess positions, medium elos, move=30, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=30, SF16



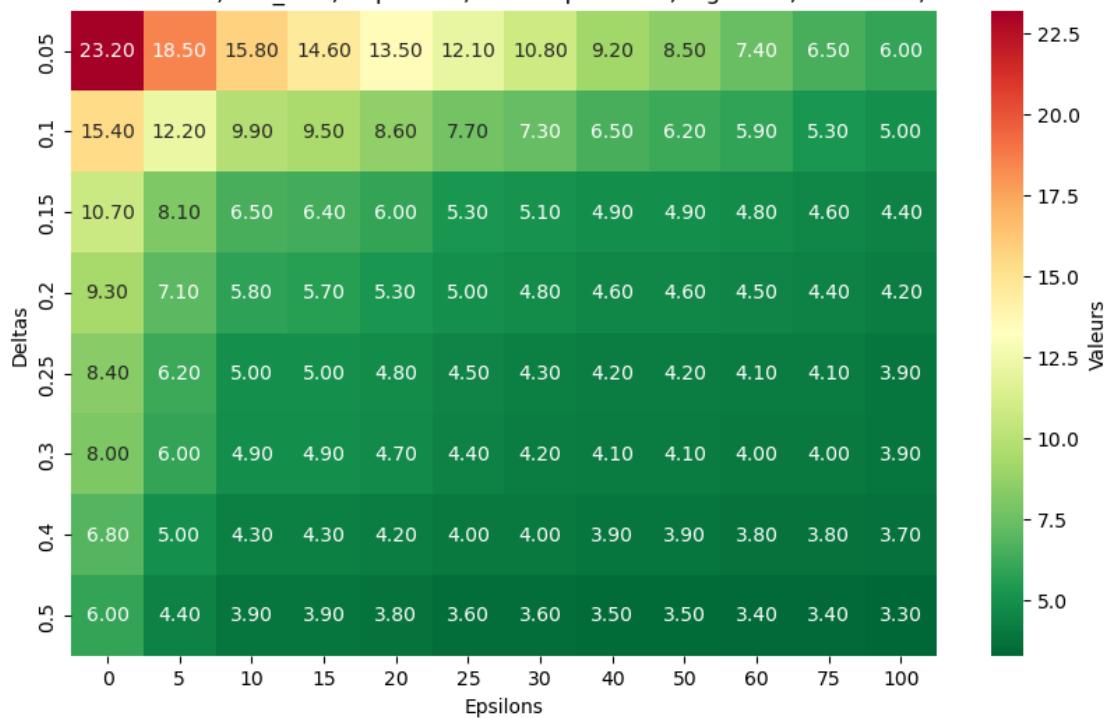
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 1, 20, " lichess positions, high elos, move=30, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=30, SF16



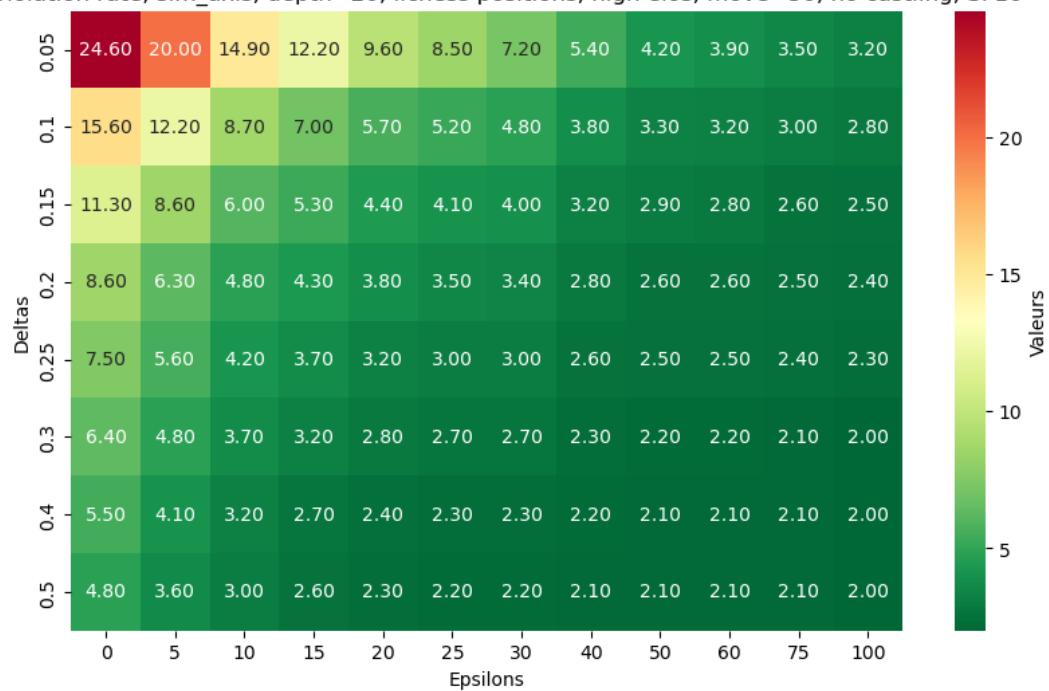
Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-30_nocastlingsim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 1, 20, " lichess positions, high elos, move=30, no castling, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=30, no castling, SF16

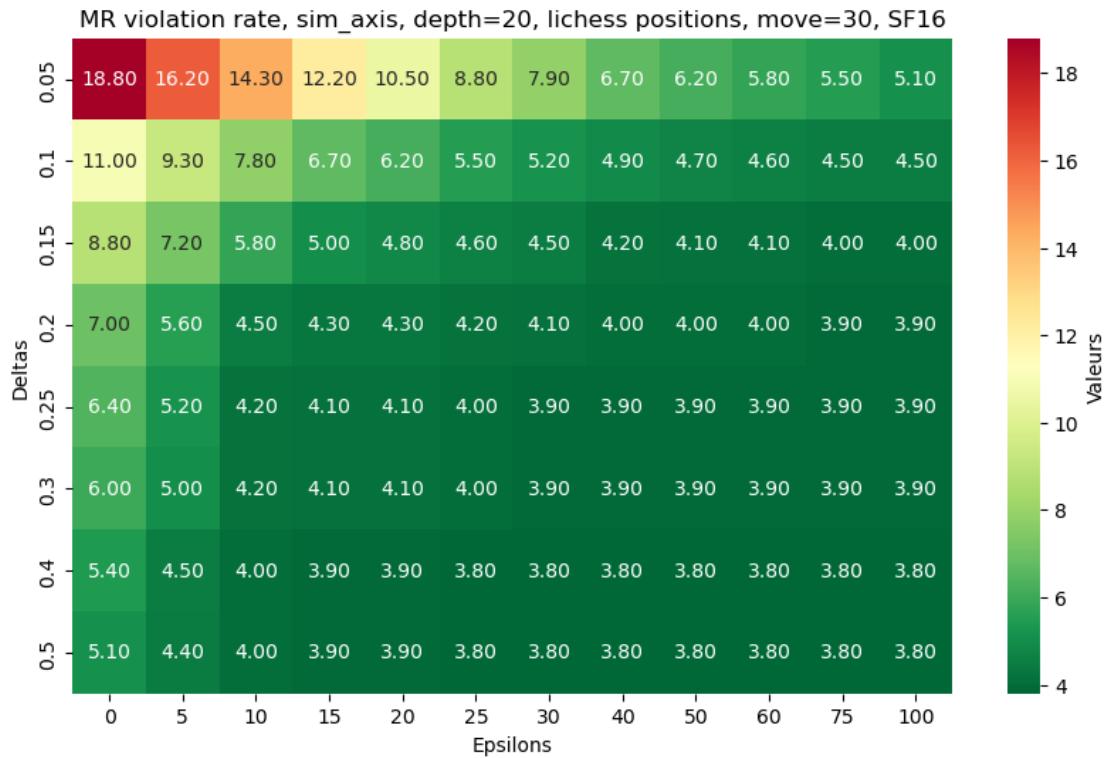


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 30')
tests(evs, 1, 20, ", lichess positions, move=30, SF16")
```

move = 30



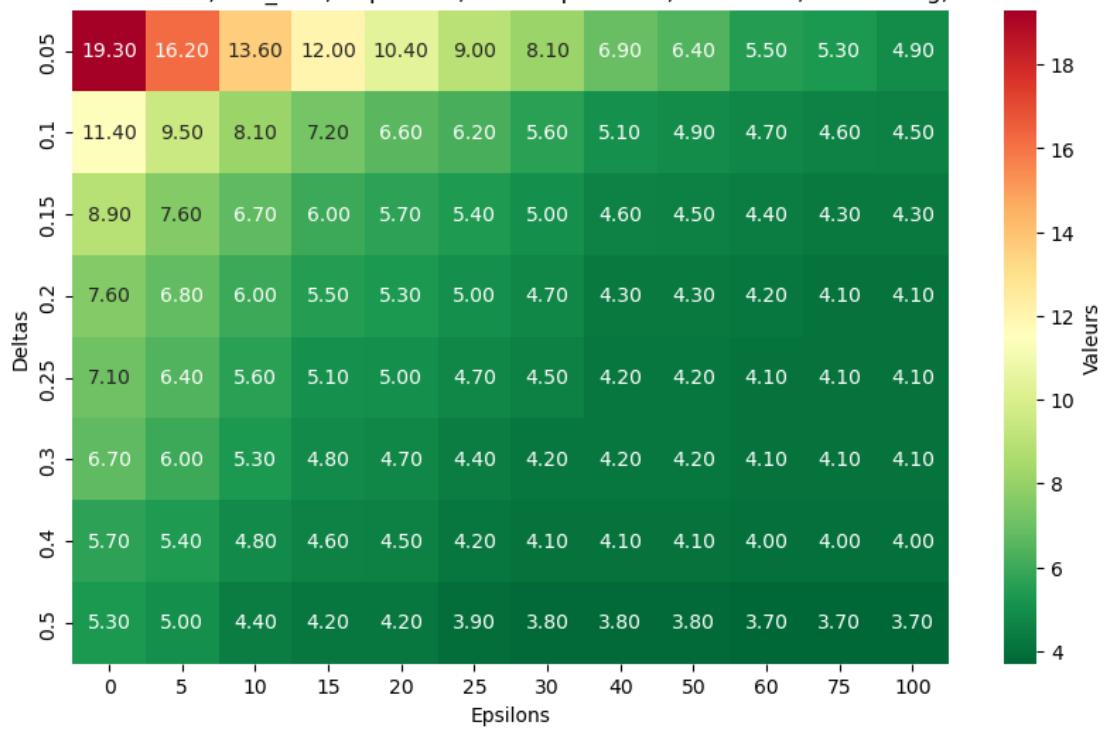
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-30_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 1, 20, " lichess positions, move=30, no castling, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, lichess positions, move=30, no castling, SF16

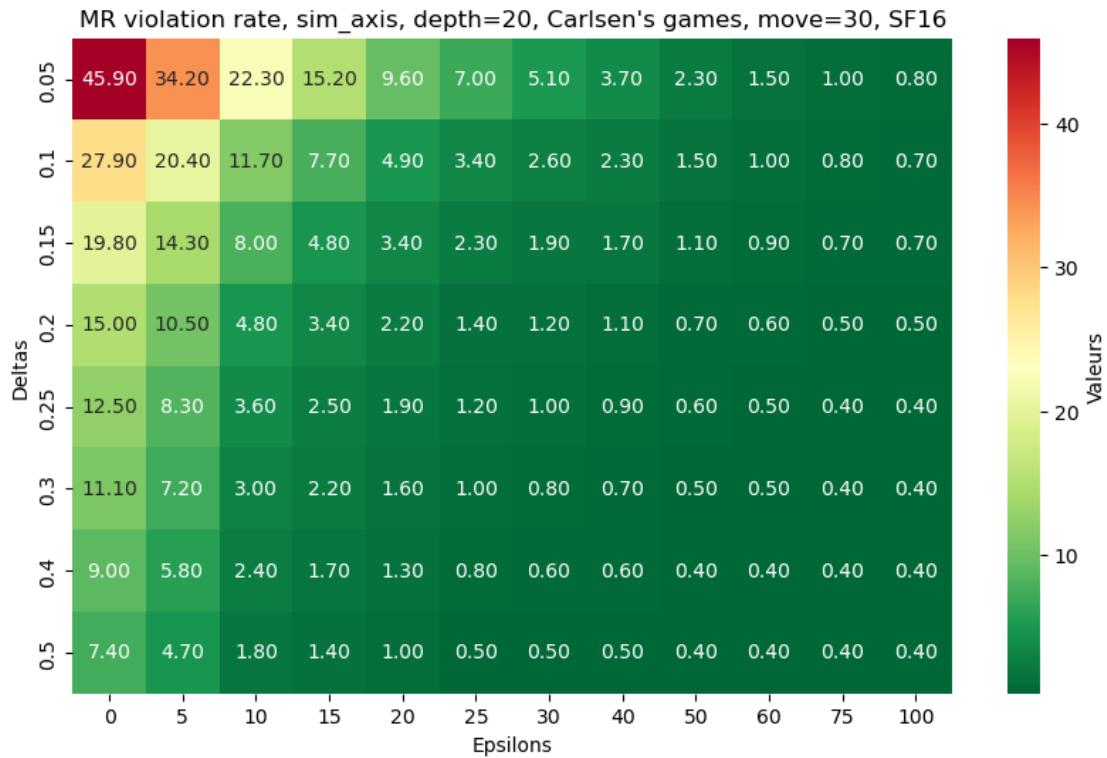


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 30')
      tests(evs, 1, 20, " Carlsen's games, move=30, SF16")
```

move = 30



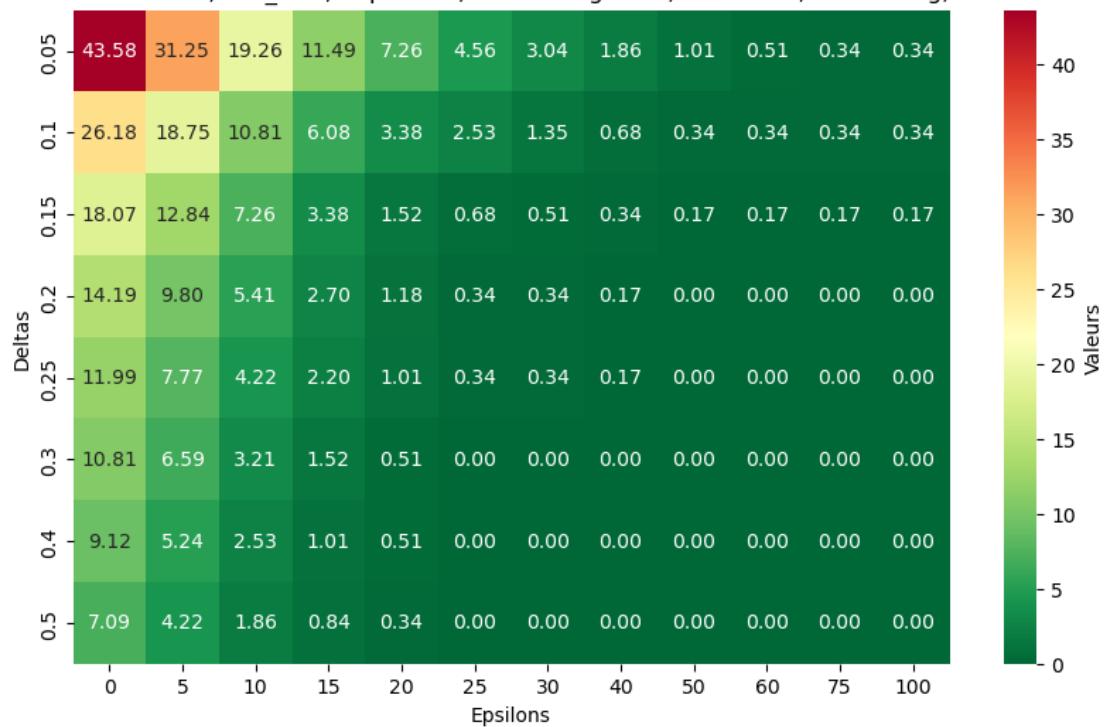
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-30_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs11630 = pickle.load(file)

      print('move = 30')
      tests(evs11630, 1, 20, " Carlsen's games, move=30, no castling, SF16")
```

move = 30

MR violation rate, sim_axis, depth=20, Carlsen's games, move=30, no castling, SF16



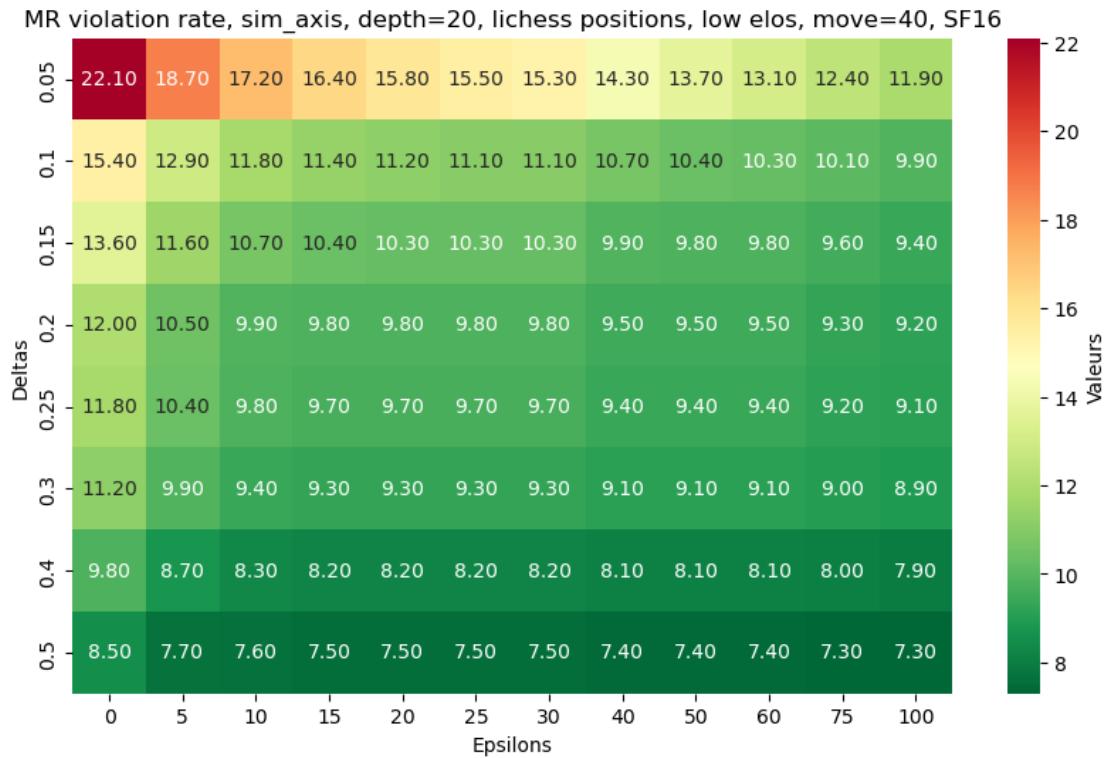
Endgames (move=40)

Lichess positions - 1000 to 1500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess1000-1500-40sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 1, 20, " lichess positions, low elos, move=40, SF16")
```

move = 40



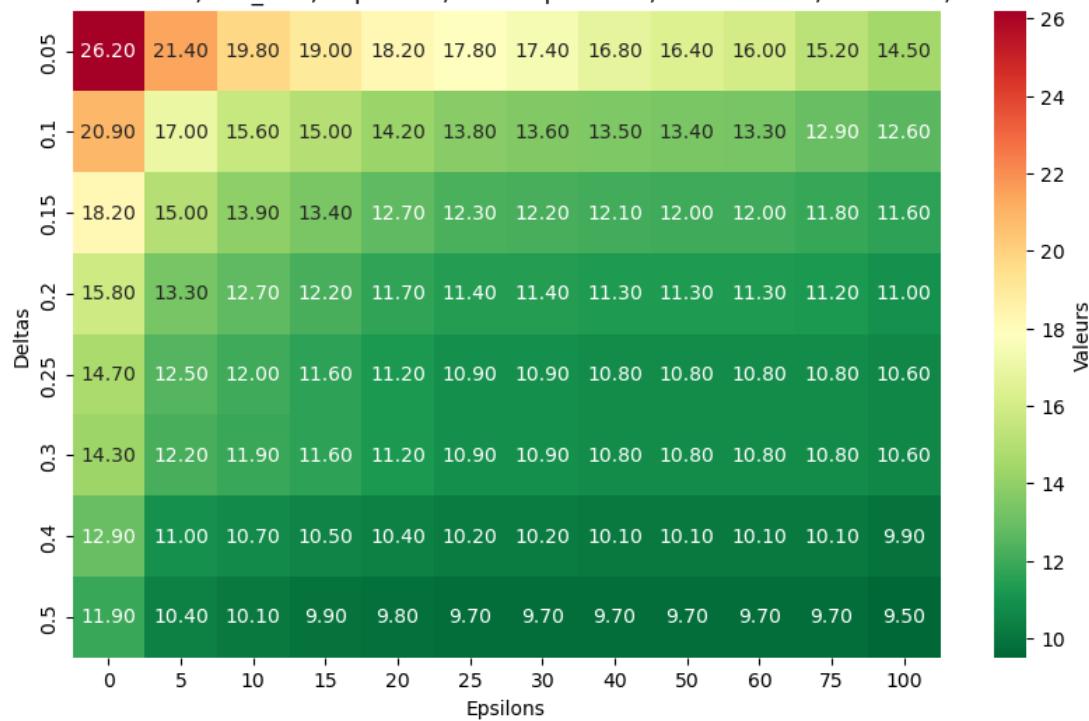
Lichess positions - 1500 to 2000 elo rating

```
[ ]: path = os.path.join('reals','ev_lichess1500-2000-40sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs,1,20, " lichess positions, medium elos, move=40, SF16")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, medium elos, move=40, SF16



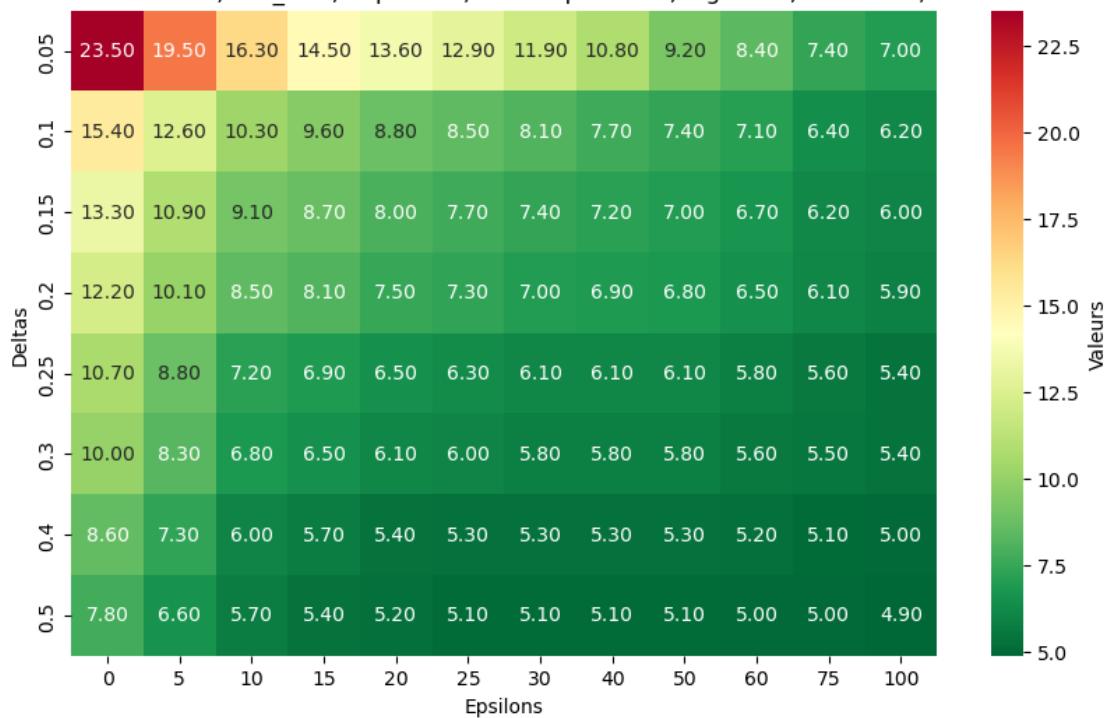
Lichess positions - 2000 to 2500 elo rating

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 1, 20, " lichess positions, high elos, move=40, SF16")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, high elos, move=40, SF16

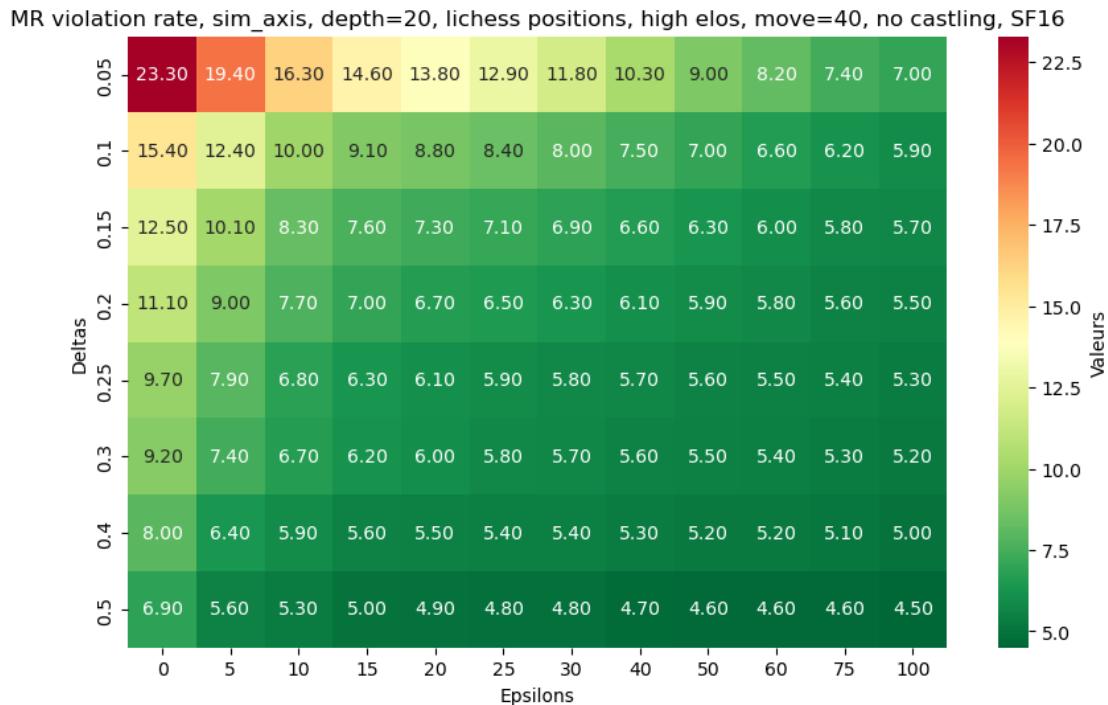


Lichess positions - 2000 to 2500 elo rating, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichess2000-2500-40_nocastlingsim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 1, 20, " lichess positions, high elos, move=40, no castling, SF16")
```

move = 40

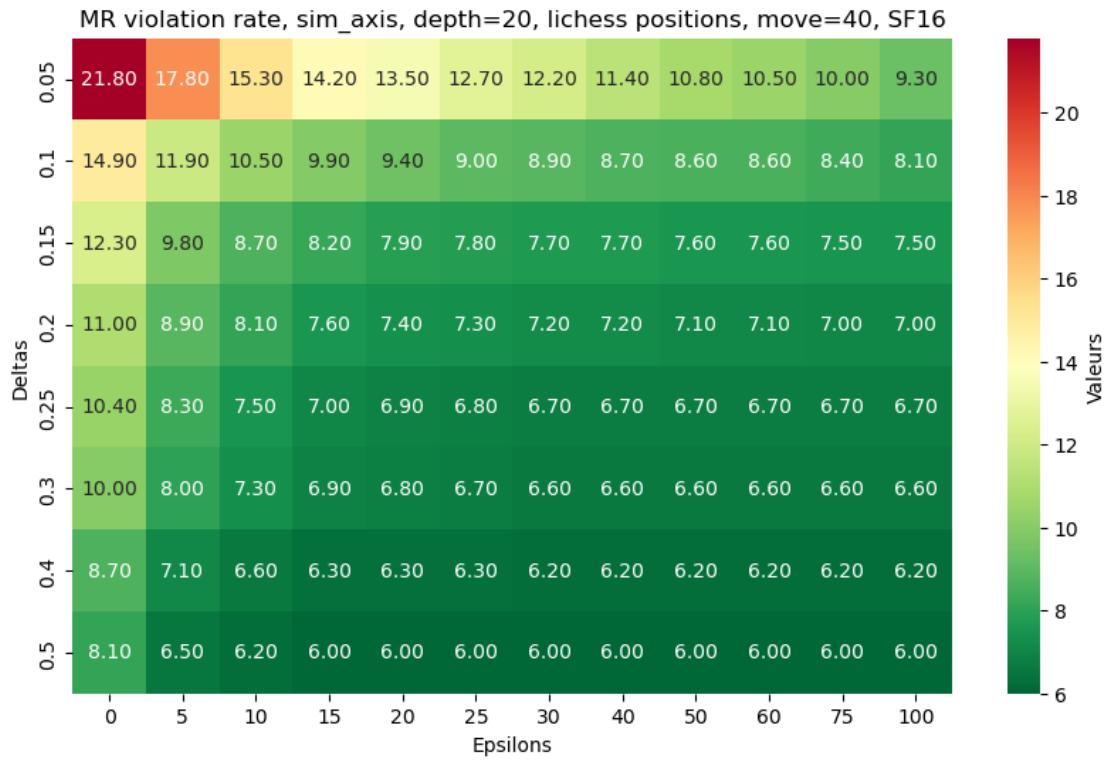


Lichess positions - All elos

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40sim_axis_d_20_v16.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs, 1, 20, " lichess positions, move=40, SF16")
```

move = 40



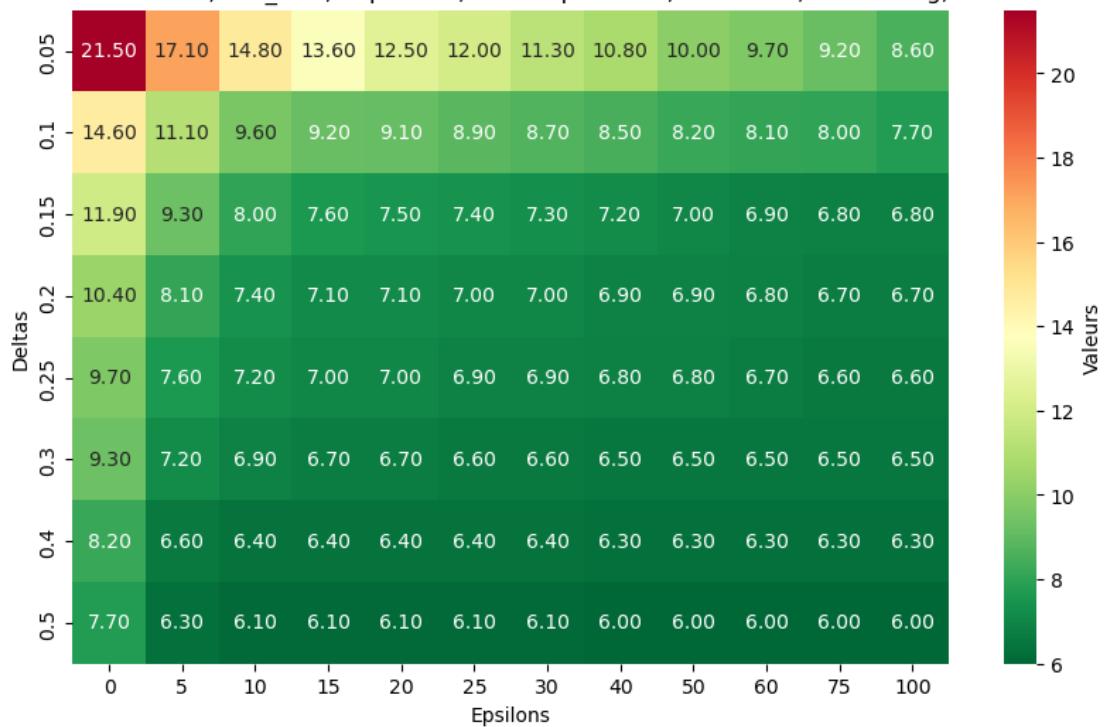
Lichess positions - All elos, no castling available

```
[ ]: path = os.path.join('reals', 'ev_lichessAllElos-40_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 1, 20, " lichess positions, move=40, no castling, SF16")
```

move = 40

MR violation rate, sim_axis, depth=20, lichess positions, move=40, no castling, SF16

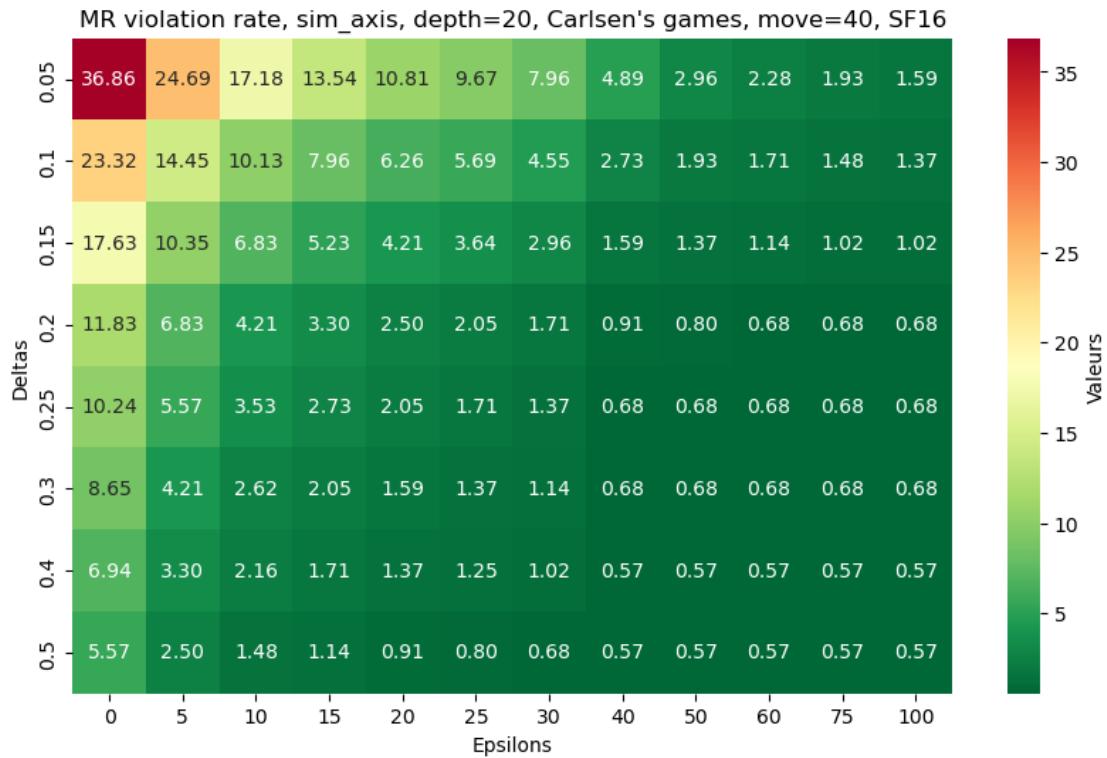


Carlsen's games

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40sim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs = pickle.load(file)

      print('move = 40')
      tests(evs, 1, 20, " Carlsen's games, move=40, SF16")
```

move = 40



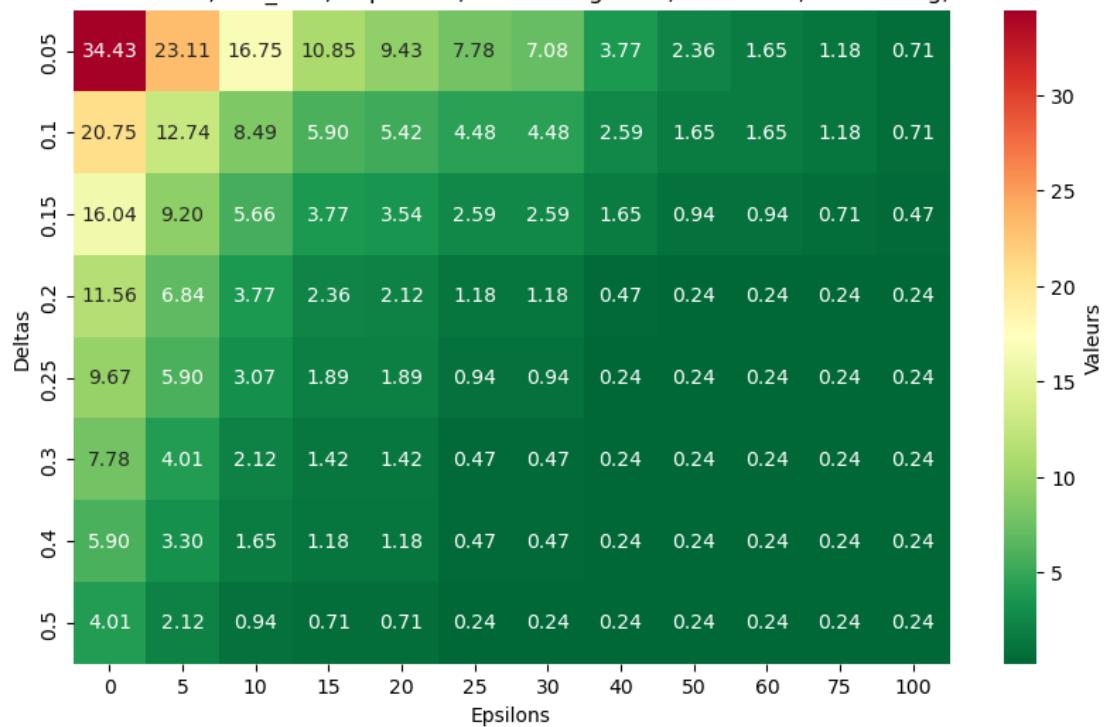
Carlsen's games, no castling available

```
[ ]: path = os.path.join('reals', 'ev_Carlsen-40_nocastlingsim_axis_d_20_v16.pkl')
      with open(path, 'rb') as file:
          evs11640 = pickle.load(file)

      print('move = 40')
      tests(evs11640, 1, 20, " Carlsen's games, move=40, no castling, SF16")
```

move = 40

MR violation rate, sim_axis, depth=20, Carlsen's games, move=40, no castling, SF16



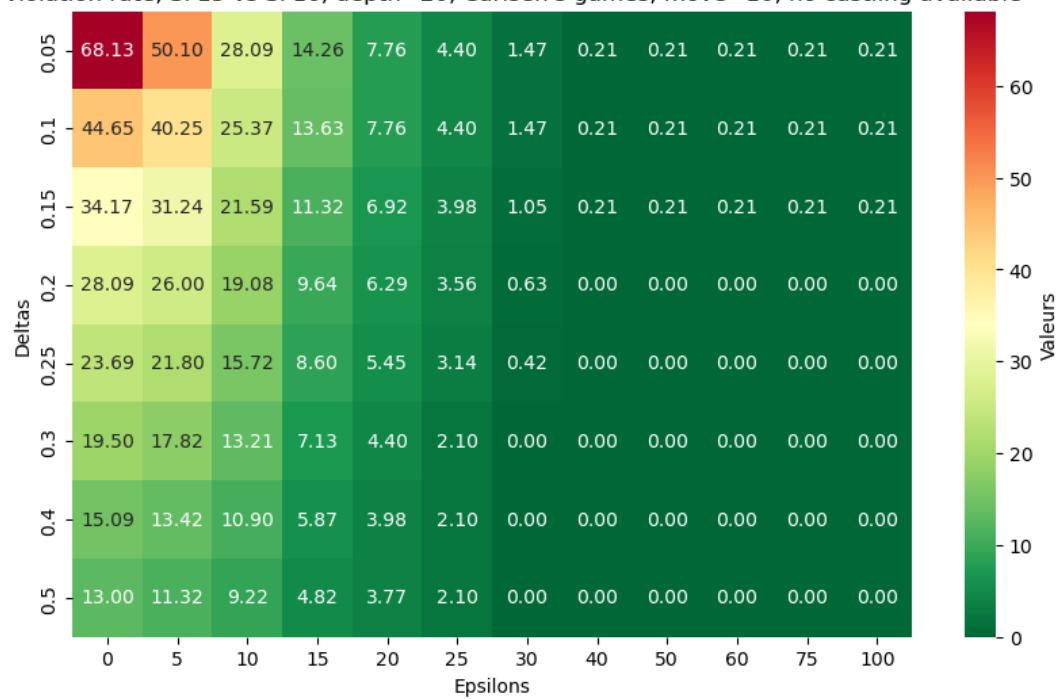
0.12.3 Differences between SF15 and SF16

For this part, we'll show the grids of gaps between SF15 and SF16, on Carlsen's games.

move=10

```
[ ]: tests(merge(evs11610,evs11510),5,20, "", Carlsen's games, move=10, no castling,
  ↴available")
```

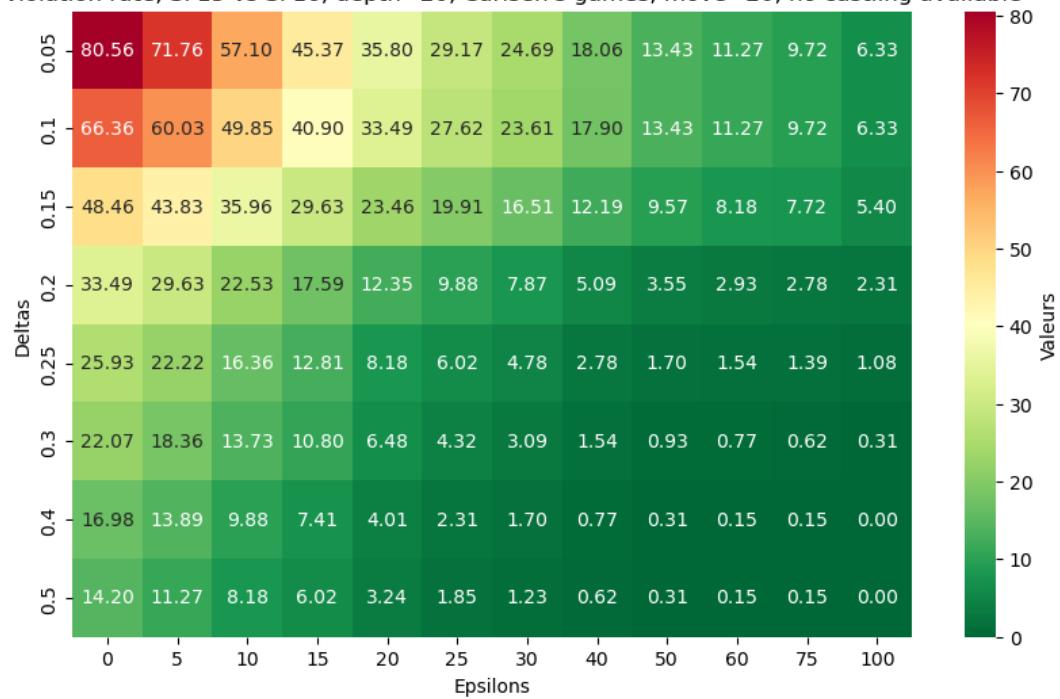
MR violation rate, SF15 vs SF16, depth=20, Carlsen's games, move=10, no castling available



move=20

```
[1]: tests(merge(evs11620,evs11520),5,20, "", Carlsen's games, move=20, no castling
         ↵available")
```

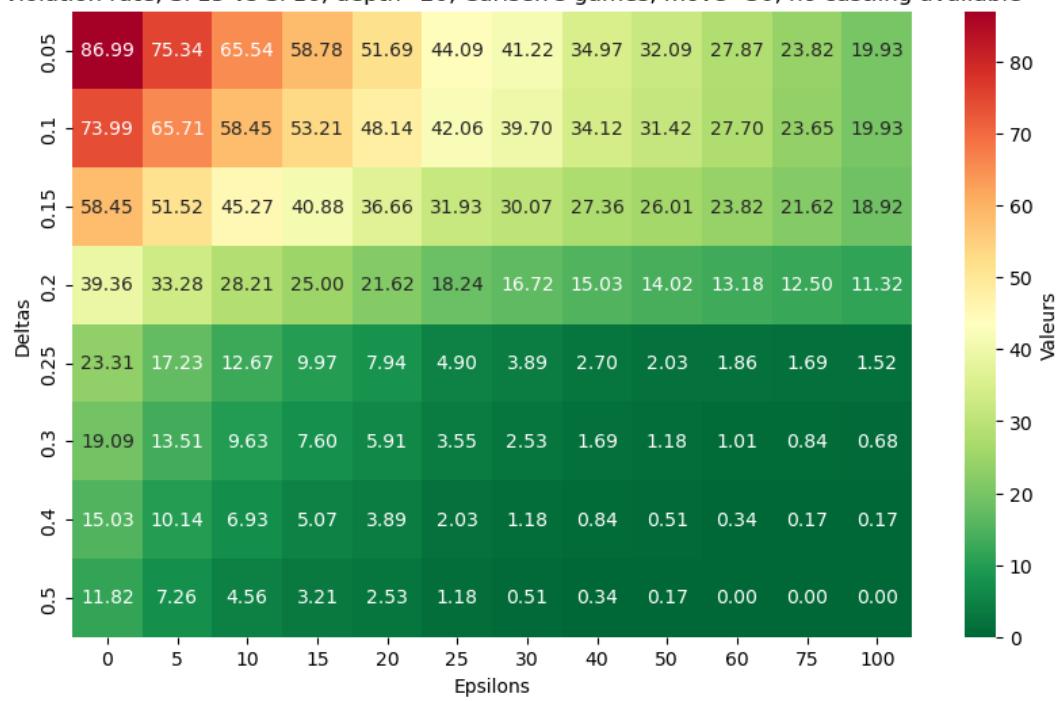
MR violation rate, SF15 vs SF16, depth=20, Carlsen's games, move=20, no castling available



move=30

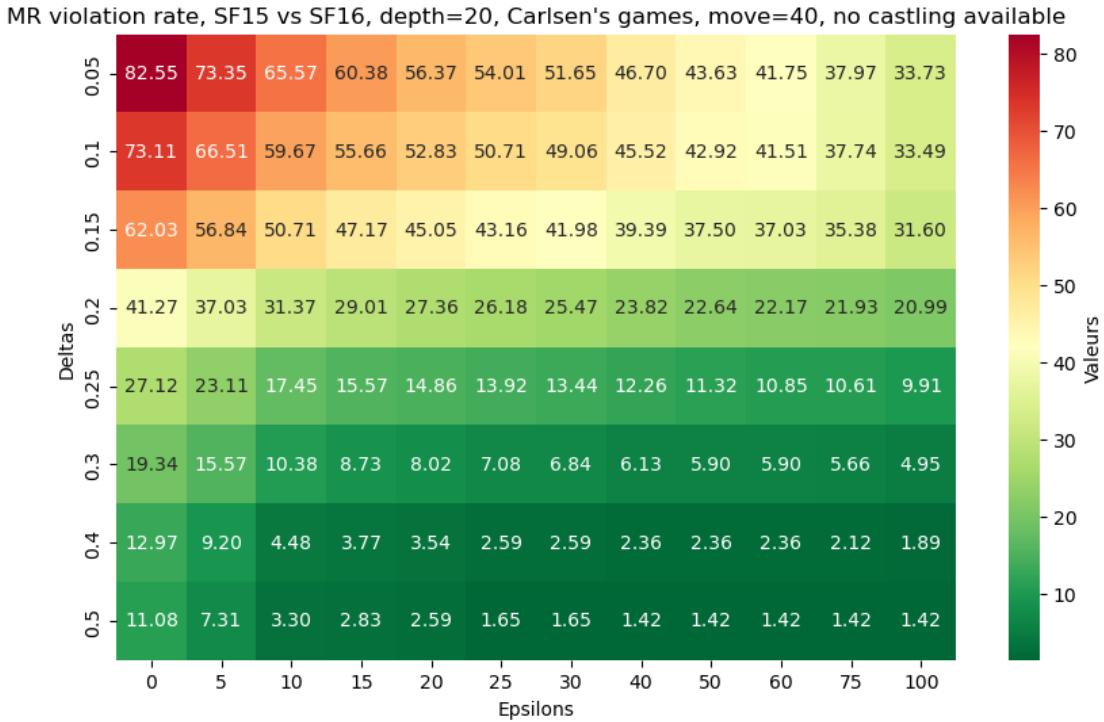
```
[1]: tests(merge(evs11630,evs11530),5,20, "", Carlsen's games, move=30, no castling
 ↵available")
```

MR violation rate, SF15 vs SF16, depth=20, Carlsen's games, move=30, no castling available



move=40

```
[1]: tests(merge(evs11640,evs11540),5,20, "", Carlsen's games, move=40, no castling
 ↵available")
```



Comments on SF15 vs SF16 As we can see, SF16 is slightly better and SF15 and SF16 tend to have bigger gaps on endgame positions.

0.12.4 Analysis on some gaps

We can observe that Stockfish tends to violate MRs more frequently in endgame positions. This is due to the higher rate of mate-positions where Stockfish doesn't return the same mate value when the position is mutated. Actually, it may be interesting to try what happens if we consider that as long as the mate is for the same side on both positions, MRs are not violated.

Editing MR_equi as shown below:

```
[ ]: #for sim_axis and sim_diag
def MR_equi(o1, o2, delta, epsilon):
    final = False
    if o1.get('type') != o2.get('type'):
        final = False
    elif o1.get('type') == o2.get('type') == 'mate' and o1.get('value')*o2.
        get('value')<0:
        final = False
    elif o1.get('type') == o2.get('type') == 'cp' and dif1(o1.get('value'), o2.
        get('value')) > delta and dif2(
            o1.get('value'), o2.get('value')) >= epsilon:
        final = False
```

```

    else:
        final = True
    return final

```

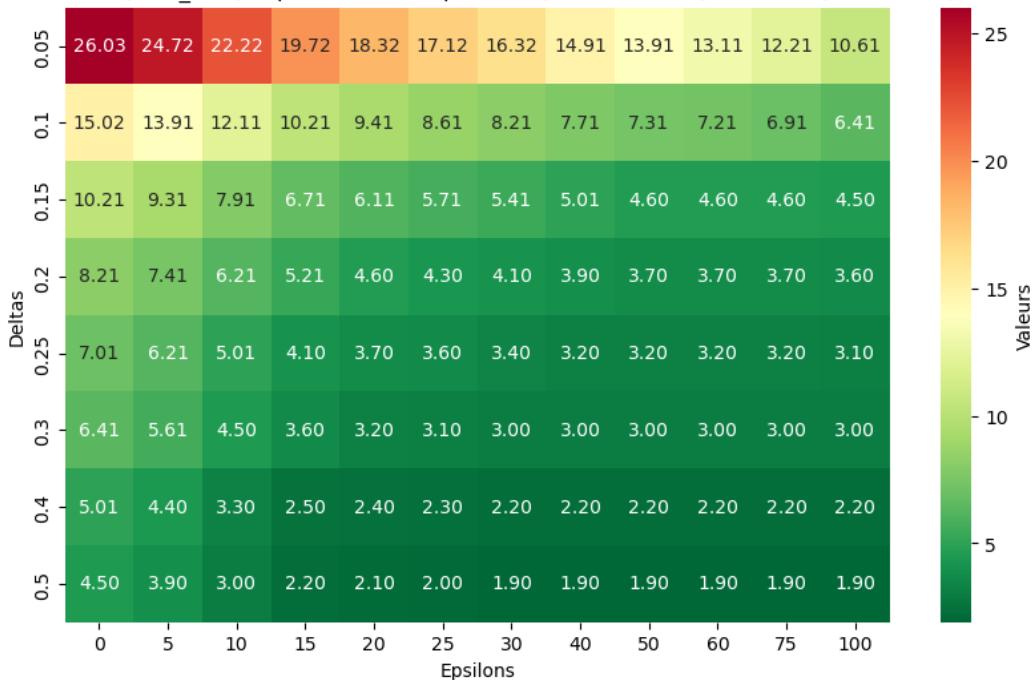
Move 40, Lichess positions - 1500 to 2000 elo rating This grid originally showed more than 10% of failure.

```
[ ]: path = os.path.join('reals','ev_lichess2000-2500-40sim_axis_d_20.pkl')
with open(path, 'rb') as file:
    evs = pickle.load(file)

print('move = 40')
tests(evs,1,20, "lichess positions, medium elos, move=40, MR edited, SF15")
```

move = 40

MR violation rate, sim_axis, depth=20lichess positions, medium elos, move=40, MR edited, SF15



The difference is due to the relatively high percentage of mate positions.

```
[ ]: print("Mates proportion : ", len(split_mate_cp(evs))[1]/
    ↪(len(split_mate_cp(evs))[0]+len(split_mate_cp(evs)[1])))
print('Mates failure rate : ', 100-MR(split_mate_cp(evs)[1],1,1,1), '%')
```

Mates proportion : 0.13713713713713
Mates failure rate : 12.408759124087581 %

We'll now print evaluations as a function of depth on biggest gaps.

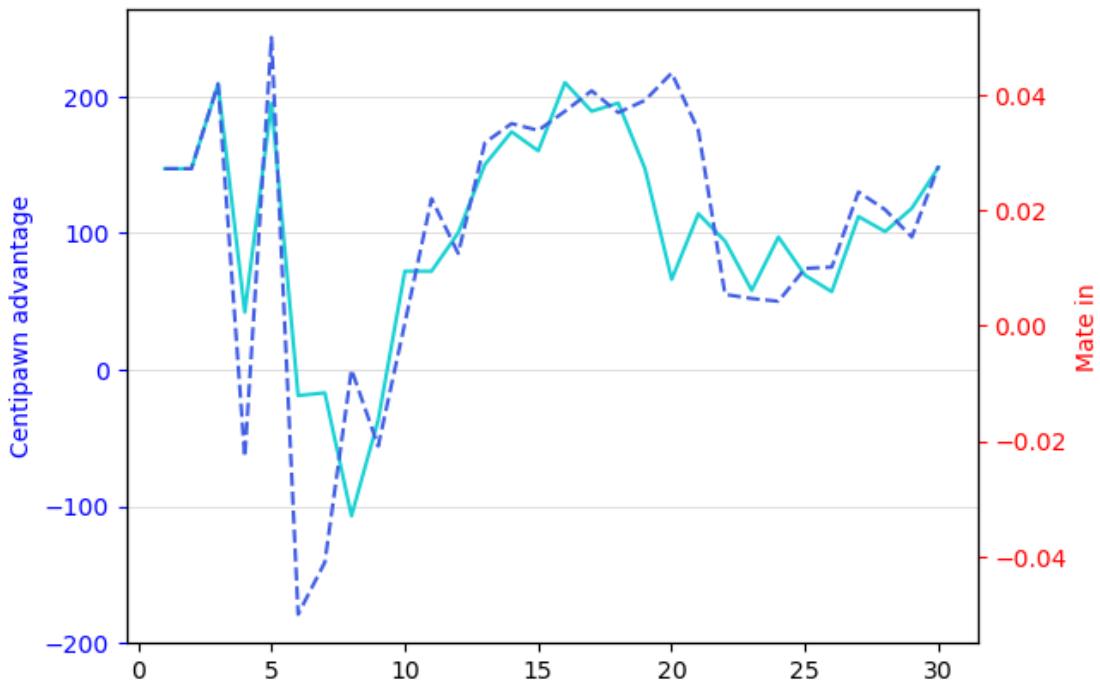
sim_mirror

Carlsen positions, move 10

```
[ ]: carlsen10 = 'r2qkb1r/pp2p2p/n4pN1/1Npn3Q/8/8/PPPP1PPP/R1B1K2R b'  
  
show_pos([carlsen10],300)  
print(evaluationdouble(carlsen10,sim_mirror(carlsen10),50000,20))  
  
<IPython.core.display.HTML object>  
({'type': 'cp', 'value': 66}, {'type': 'cp', 'value': -217})
```

```
[ ]: chemin = os.path.join('plotevas', 'calsen10.pkl')  
with open(chemin, 'rb') as fichier:  
    carlsen10p = pickle.load(fichier)  
  
plotevas(carlsen10p[0],carlsen10p[1],carlsen10p[2], 0)
```

MR verified at depth = [2, 3, 5, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 25, 26, 27, 28, 29, 30]



sim_axis

Lichess positions - 1500 to 2000 elo rating, move 40

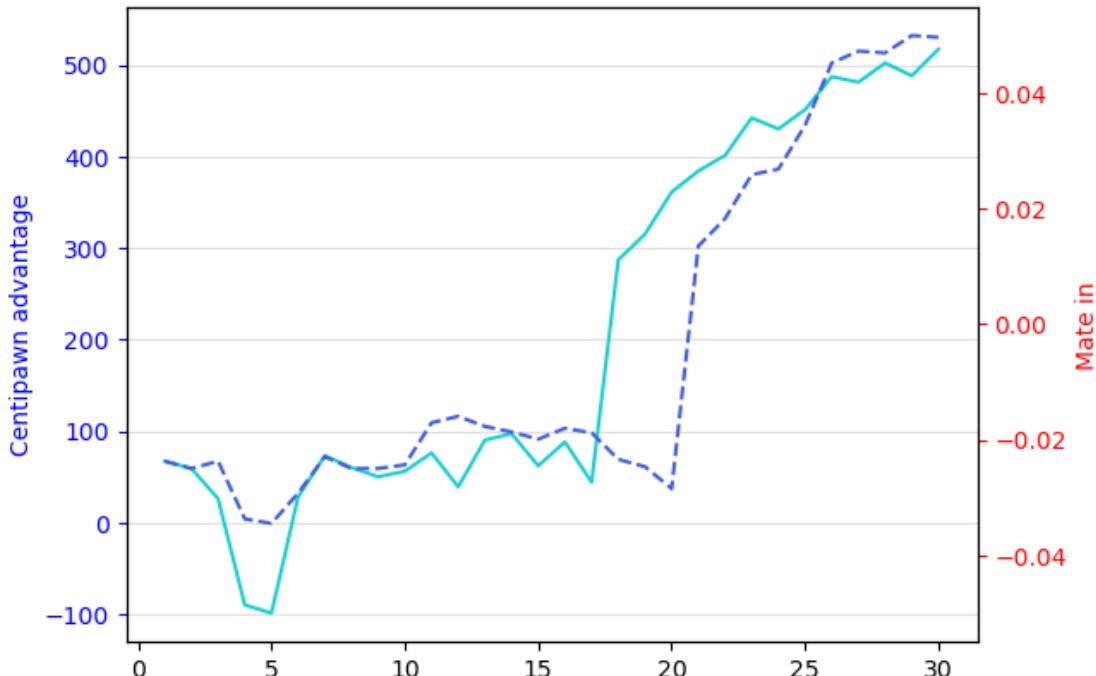
```
[ ]: pos = '8/1p5p/p3k3/4p1P1/P2bP3/7N/2p4P/2B4K b'
show_pos([pos],300)
print(evaluationondouble(pos,sim_axis(pos),50000,20))

<IPython.core.display.HTML object>
({'type': 'cp', 'value': 361}, {'type': 'cp', 'value': 37})
```

```
[ ]: chemin = os.path.join('plotevas', 'L152040.pkl')
with open(chemin, 'rb') as fichier:
    l152040 = pickle.load(fichier)

plotevas(l152040[0],l152040[1],l152040[2], 1)
```

MR verified at depth = [2, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]



```
[ ]: sim_axis(pos)
```

```
[ ]: '8/p5p1/3k3p/1P1p4/3Pb2P/N7/P4p2/K4B2 b'
```

```
[ ]: sim_axis('1N5k/3Kb2q/1NP3p1/1R4B1/3R4/1B6/8/n3Q3 w')
```

```
[ ]: 'k5N1/q2bK3/1p3PN1/1B4R1/4R3/6B1/8/3Q3n w'
```

```
[ ]: with open('massplotevas/evas_axis_1.pkl','rb') as fichier:  
    loaded = pickle.load(fichier)
```

```
[ ]: tableau = [0] * 25  
for k in range(len(loaded)):  
    for l in loaded[k][2]:  
        if l<len(tableau):  
            tableau[l]+=1  
tableau
```

```
[ ]: [0,  
      0,  
     100,  
     91,  
     83,  
     67,  
     71,  
     69,  
     70,  
     72,  
     74,  
     85,  
     78,  
     83,  
     88,  
     88,  
     86,  
     87,  
     87,  
     92,  
     89,  
     92,  
     96,  
     96,  
     96]
```

```
[ ]: import pandas as pd  
pd.__version__
```

```
[ ]: '1.5.3'
```

```
[ ]: evaluations50000_sim_mirror_d_20_real
```

```
[ ]:      index                                pos1  \\\n0      r1bq1r1k/p1pnbp1/1p2p3/6p1/3PB3/5N2/PPPQ1PPP/...  
1      r1bq1r1k/p1pnbp1/1p2p3/6p1/3PB3/5N2/PPPQ1PPP/...  
2      r1b2rk1/1p1nbppp/pq1p4/3B4/P2NP3/2N1p3/1PP3PP/...
```

```

3      3   r1b2rk1/1p1nbppp/pq1p4/3B4/P2NP3/2N1p3/1PP3PP/...
4      4   r2r1k2/p3qpp1/1p1ppn1p/n5B1/P1PNbP2/2P3Q1/4B1P...
...
1187   ...   ...
1187   1187  3r1r1k/1p3qb1/1p1p2pp/1P1Pp3/R2nN1BP/3P2P1/Q4P...
1188   1188  3r1r1k/1p3qb1/1p1p2pp/1P1Pp3/R2nN1BP/3P2P1/Q4P...
1189   1189           8/2Q3pk/pB5p/P7/7P/2bq2P1/5PK1/8 w
1190   1190           8/2Q3pk/pB5p/P7/7P/2bq2P1/5PK1/8 b
1191   1191  1rbq1rk1/ppp1p1bp/5np1/3Ppp2/2P5/1QN3P1/PP2PPB...

                      evaluation pos1 \
0      {'type': 'cp', 'value': 120}
1      {'type': 'cp', 'value': -451}
2      {'type': 'cp', 'value': 281}
3      {'type': 'cp', 'value': -117}
4      {'type': 'cp', 'value': 247}
...
1187   ...   ...
1187   {'type': 'cp', 'value': 173}
1188   {'type': 'cp', 'value': 113}
1189   {'type': 'cp', 'value': 181}
1190   {'type': 'cp', 'value': 143}
1191   {'type': 'cp', 'value': 137}

                      pos2 \
0      2kr3r/pppq1ppp/5n2/3pb3/6P1/1P2P3/P1PNBPP1/R1B...
1      2kr3r/pppq1ppp/5n2/3pb3/6P1/1P2P3/P1PNBPP1/R1B...
2      r2q1r1k/1pp3pp/2n1P3/p2np3/3b4/PQ1P4/1P1NBPPP/...
3      r2q1r1k/1pp3pp/2n1P3/p2np3/3b4/PQ1P4/1P1NBPPP/...
4      4rrk1/4b1pp/2p3q1/p1pnBp2/N5b1/1P1PPN1P/P3QPP1...

...
1187   ...   ...
1187   1r6/q4p1k/3p2p1/r2Nn1bp/1p1pP3/1P1P2PP/1P3QB1/...
1188   1r6/q4p1k/3p2p1/r2Nn1bp/1p1pP3/1P1P2PP/1P3QB1/...
1189           8/5pk1/2BQ2p1/7p/p7/Pb5P/2q3PK/8 b
1190           8/5pk1/2BQ2p1/7p/p7/Pb5P/2q3PK/8 w
1191   r1b2rk1/pp2ppbp/1qn3p1/2p5/3pPP2/5NP1/PPP1P1BP...

                      evaluation pos2
0      {'type': 'cp', 'value': -211}
1      {'type': 'cp', 'value': 469}
2      {'type': 'cp', 'value': -324}
3      {'type': 'cp', 'value': 125}
4      {'type': 'cp', 'value': -259}
...
1187   ...   ...
1187   {'type': 'cp', 'value': -166}
1188   {'type': 'cp', 'value': -147}
1189   {'type': 'cp', 'value': -182}
1190   {'type': 'cp', 'value': -96}
1191   {'type': 'cp', 'value': -135}

```

```
[1192 rows x 5 columns]
```

```
[ ]: import pandas as pd
import numpy as np

# Assuming 'df' is your original dataframe
# Extract the 'value' from the 'evaluation pos1' column where 'type' is 'cp', ↴
# else set to NaN
evaluations50000_sim_mirror_d_20_real['eval_pos1'] = ↴
    evaluations50000_sim_mirror_d_20_real['evaluation pos1'].apply(lambda x: ↴
        x['value'] if x['type'] == 'cp' else np.nan)

# Extract the 'value' from the 'evaluation pos2' column where 'type' is 'cp', ↴
# else set to NaN
evaluations50000_sim_mirror_d_20_real['eval_pos2'] = ↴
    evaluations50000_sim_mirror_d_20_real['evaluation pos2'].apply(lambda x: ↴
        x['value'] if x['type'] == 'cp' else np.nan)

# Compute the 'ratio_eval' column as |x - (-y)| / (|x| + |y|) where x is ↴
# eval_pos1 and y is eval_pos2
evaluations50000_sim_mirror_d_20_real['ratio_eval'] = ↴
    abs(evaluations50000_sim_mirror_d_20_real['eval_pos1'] - ↴
        (-evaluations50000_sim_mirror_d_20_real['eval_pos2'])) / ↴
    (abs(evaluations50000_sim_mirror_d_20_real['eval_pos1']) + ↴
        abs(evaluations50000_sim_mirror_d_20_real['eval_pos2']))

# Compute the 'cpdiff_eval' column as |x - (-y)| where x is eval_pos1 and y is ↴
# eval_pos2
evaluations50000_sim_mirror_d_20_real['cpdiff_eval'] = ↴
    abs(evaluations50000_sim_mirror_d_20_real['eval_pos1'] - ↴
        (-evaluations50000_sim_mirror_d_20_real['eval_pos2']))

# when eval_pos1 is 0 (or eval_pos2 is 0) the ratio is 1... which is stupid!
evaluations50000_sim_mirror_d_20_real.sort_values(by="ratio_eval", ↴
    ascending=False, na_position='last')

evaluations50000_sim_mirror_d_20_real.sort_values(by="cpdiff_eval", ↴
    ascending=False, na_position='last')
```

```
[ ]:      index                                pos1 \
100     100          7b/8/kq6/8/8/1N2R3/K2P4/8 w
116     116          r1b2rk1/pp1p2pR/8/1pb2p2/5N2/7Q/qPPB1PPP/6K1 w
40      40          8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w
88      88          8/2p1k3/3p3p/2PP1pp1/1P1K1P2/6P1/8/8 w
8       8           3q1r1k/1b3ppp/p1n5/1p1pPB2/2rP4/P6N/1P2Q1PP/R4...
```

```

...
679    679  1rbq1rk1/1p3pbp/p2p1np1/2pP4/Pn2PP2/2N1B3/1P1...
873    873  2r5/6bk/q4np1/1N2pp1p/PBb1p3/Q3P1PP/5PB1/3R2K1 b
107    107          8/3R1P2/1ppP1p2/3r4/8/K7/p4k2/8 b
117    117      r1b2rk1/pp1p2pR/8/1pb2p2/5N2/7Q/qPPB1PPP/6K1 b
245    245  3r3r/k1p2pb1/B1b2q2/2RN3p/3P2p1/1Q2B1Pn/PP3PKP...

            evaluation pos1 \
100  {'type': 'cp', 'value': -478}
116  {'type': 'cp', 'value': 362}
40   {'type': 'cp', 'value': 197}
88   {'type': 'cp', 'value': -1}
8    {'type': 'cp', 'value': 468}
...
679    {'type': 'cp', 'value': 0}
873    {'type': 'cp', 'value': 0}
107  {'type': 'mate', 'value': -4}
117  {'type': 'mate', 'value': -2}
245  {'type': 'mate', 'value': -1}

            pos2 \
100          8/k2p4/1n2r3/8/8/KQ6/8/7B b
116  6k1/Qppb1ppp/7q/5n2/1PB2P2/8/PP1P2Pr/R1B2RK1 b
40   5k2/5p2/1p4pp/1p2P3/1Ppn4/P4NPP/5PK1/8 b
88   8/8/6p1/1p1k1p2/2pp1PP1/3P3P/2P1K3/8 b
8    r4rk1/1p2q1pp/p6n/2Rp4/1P1Ppb2/P1N5/1B3PPP/3Q1...
...
679  r2q1rk1/1p1nb1pp/2n1b3/pN2pp2/2Pp4/P2P1NP1/1P3...
873  3r2k1/5pb1/q3p1pp/pbB1P3/1n2PP1P/Q4NP1/6BK/2R5 w
107          8/P4K2/k7/8/3R4/1PPp1P2/3r1p2/8 w
117  6k1/Qppb1ppp/7q/5n2/1PB2P2/8/PP1P2Pr/R1B2RK1 w
245  5r2/pp3pkp/1q2b1pN/3p2P1/2rn3P/b1B2Q2/K1P2PB1/...

            evaluation pos2  eval_pos1  eval_pos2  ratio_eval \
100  {'type': 'cp', 'value': 0}     -478.0       0.0  1.000000
116  {'type': 'cp', 'value': -824}    362.0    -824.0  0.389545
40   {'type': 'cp', 'value': -513}    197.0    -513.0  0.445070
88   {'type': 'cp', 'value': -277}    -1.0    -277.0  1.000000
8    {'type': 'cp', 'value': -199}    468.0    -199.0  0.403298
...
679    {'type': 'cp', 'value': 0}     0.0       0.0    NaN
873    {'type': 'cp', 'value': 0}     0.0       0.0    NaN
107  {'type': 'mate', 'value': 4}     NaN       NaN    NaN
117  {'type': 'mate', 'value': 2}     NaN       NaN    NaN
245  {'type': 'mate', 'value': 1}     NaN       NaN    NaN
```

cpdiff_eval

```

100      478.0
116      462.0
40       316.0
88       278.0
8        269.0
..
679      ...
873      0.0
107      NaN
117      NaN
245      NaN

```

[1192 rows x 9 columns]

```

[ ]: 
[ ]: show_pos(["8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w"], 300)
<IPython.core.display.HTML object>
[ ]: evaluation("8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w", 50000, 10)
[ ]: ({'type': 'cp', 'value': 145},
      [{'Move': 'd5b6', 'Centipawn': 202, 'Mate': None}])
[ ]: evaluationdouble("8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w", "5k2/5p2/1p4pp/
    ↪1p2P3/1Ppn4/P4NPP/5PK1/8 b", 50000, 32)
[ ]: ({'type': 'cp', 'value': 653}, {'type': 'cp', 'value': -847})
[ ]: evaluation("8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w", 50000, 20)
[ ]: ({'type': 'cp', 'value': 197},
      [{'Move': 'd5f6', 'Centipawn': 199, 'Mate': None}])
[ ]: import subprocess

# Function to extract the evaluation scores and best moves from Stockfish output
def extract_info(output):
    info_lines = re.findall(r"info depth (\d+) seldepth \d+ multipv (\d+) score_
    ↪cp (-?\d+)(?: .+)? pv (.+)", output)

    scores = []
    best_moves = []
    evs = []
    for depth, multipv, score, pv in info_lines:
        d = int(depth)
        m = int(multipv)

```

```

    if m == 1:
        scores[d] = int(score)
        best_moves[d] = pv.split()[0]
        evs[d] = scores[d]
    return best_moves, scores, evs

# analyze fen
# depth is max depth and actually we get the number (scores) from 1 to depth
def analyze_fen(fen, depth):

    command = STOCKFISH15_LOCATION # f"docker run -i --rm {IMAGE_NAME}:
    ↪{version}"

    input_str = f"position fen {fen}\ngo depth {depth}\nucinewgame\nquit\n"

    try:
        process = subprocess.run(
            command, input=input_str, text=True, capture_output=True, shell=True
        )
        output = process.stdout
        # print(output)
    except Exception as e:
        print("Error or timeout...", e)
        return

    results = []
    best_moves, scores, evs = extract_info(output)
    for d in range(1, depth + 1):
        result = {
            "type" : "cp",
            "value" : evs[d] if d in evs else None
        }
        results.append(result)
    return results

def analyze_fen_evolution(fen, fen_t, depth_max):
    ev1 = analyze_fen(fen, depth_max)
    ev2 = analyze_fen(fen_t, depth_max)
    ev1 = [{"value" : i["value"] * (-1), "type" : i["type"]} for i in ev1] # ↪
    ↪hack?
    return (ev1, ev2, [i for i in range(1, depth_max + 1)])

def analyze_evo(fen_base, fen_transform=None, depth=10):

```

```

if fen_transform is None:
    fen_transform = sim_mirror(fen_base)
print(fen_base, "mirror", fen_transform)
rs = analyze_fen_evolution(fen_base, fen_transform, depth)
plotevas(rs[0], rs[1], rs[2], 0)
show_pos([fen_base, fen_transform], 300)

# fen_base = "r2qkb1r/pp2p2p/n4pN1/1Npn3Q/8/8/PPPP1PPP/R1B1K2R b" # "8/5pk1/
# ↪p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w"
# analyze_evo(fen_base, None, depth=15)

i = 2
analyze_evo(evaluations50000_sim_mirror_d_20_real.sort_values(by="cpdiff_eval", ↪
    ascending=False, na_position='last').
    ↪iloc[i]['pos1'], evaluations50000_sim_mirror_d_20_real.
    ↪sort_values(by="cpdiff_eval", ascending=False, na_position='last').
    ↪iloc[i]['pos2'], depth=40)

```

8/5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w mirror
5k2/5p2/1p4pp/1p2P3/1Ppn4/P4NPP/5PK1/8 b

```

-----
KeyboardInterrupt                                     Traceback (most recent call last)
Cell In[422], line 69
      65 # fen_base = "r2qkb1r/pp2p2p/n4pN1/1Npn3Q/8/8/PPPP1PPP/R1B1K2R b" # "8/
      ↪5pk1/p4npp/1pPN4/1P2p3/1P4PP/5P2/5K2 w"
      66 # analyze_evo(fen_base, None, depth=15)
      68 i = 2
---> 69 ↪analyze_evo(evaluations50000_sim_mirror_d_20_real.sort_values(by="cpdiff_eval", , ascending=False,
      ↪iloc[i]['pos1'], evaluations50000_sim_mirror_d_20_real.
      ↪sort_values(by="cpdiff_eval", ascending=False, na_position='last').
      ↪iloc[i]['pos2'], depth=40)

Cell In[422], line 61, in analyze_evo(fen_base, fen_transform, depth)
      59     fen_transform = sim_mirror(fen_base)
      60 print(fen_base, "mirror", fen_transform)
---> 61 rs = analyze_fen_evolution(fen_base, fen_transform, depth)
      62 plotevas(rs[0], rs[1], rs[2], 0)
      63 show_pos([fen_base, fen_transform], 300)

Cell In[422], line 51, in analyze_fen_evolution(fen, fen_t, depth_max)
      50 def analyze_fen_evolution(fen, fen_t, depth_max):
---> 51     ev1 = analyze_fen(fen, depth_max)
      52     ev2 = analyze_fen(fen_t, depth_max)
      53     ev1 = [{"value" : i["value"] * (-1), "type" : i["type"]} for i in ↪
      ↪ev1] # hack?

Cell In[422], line 30, in analyze_fen(fen, depth)
      27 input_str = f"position fen {fen}\ngo depth {depth}\nucinewgame\nquit\n"
```

```

  29 try:
--> 30     process = subprocess.run(
  31     command, input=input_str, text=True, capture_output=True, shell=True
  32 )
  33     output = process.stdout
  34     # print(output)

File /usr/lib64/python3.11/subprocess.py:550, in run(input, capture_output, timeout, check, *popenargs, **kwargs)
  548 with Popen(*popenargs, **kwargs) as process:
  549     try:
--> 550         stdout, stderr = process.communicate(input, timeout=timeout)
  551     except TimeoutExpired as exc:
  552         process.kill()

File /usr/lib64/python3.11/subprocess.py:1209, in Popen.communicate(self, input, timeout)
  1206     endtime = None
  1208 try:
-> 1209     stdout, stderr = self._communicate(input, endtime, timeout)
  1210 except KeyboardInterrupt:
  1211     # https://bugs.python.org/issue25942
  1212     # See the detailed comment in .wait().
  1213     if timeout is not None:

File /usr/lib64/python3.11/subprocess.py:2113, in Popen._communicate(self, input, endtime, orig_timeout)
  2106     self._check_timeout(endtime, orig_timeout,
  2107                           stdout, stderr,
  2108                           skip_check_and_raise=True)
  2109     raise RuntimeError( # Impossible :
  2110         '_check_timeout(..., skip_check_and_raise=True) '
  2111         'failed to raise TimeoutExpired.')
-> 2113 ready = selector.select(timeout)
  2114 self._check_timeout(endtime, orig_timeout, stdout, stderr)
  2116 # XXX Rewrite these to use non-blocking I/O on the file
  2117 # objects; they are no longer using C stdio!

File /usr/lib64/python3.11/selectors.py:415, in _PollLikeSelector.select(self, timeout)
  413 ready = []
  414 try:
--> 415     fd_event_list = self._selector.poll(timeout)
  416 except InterruptedError:
  417     return ready

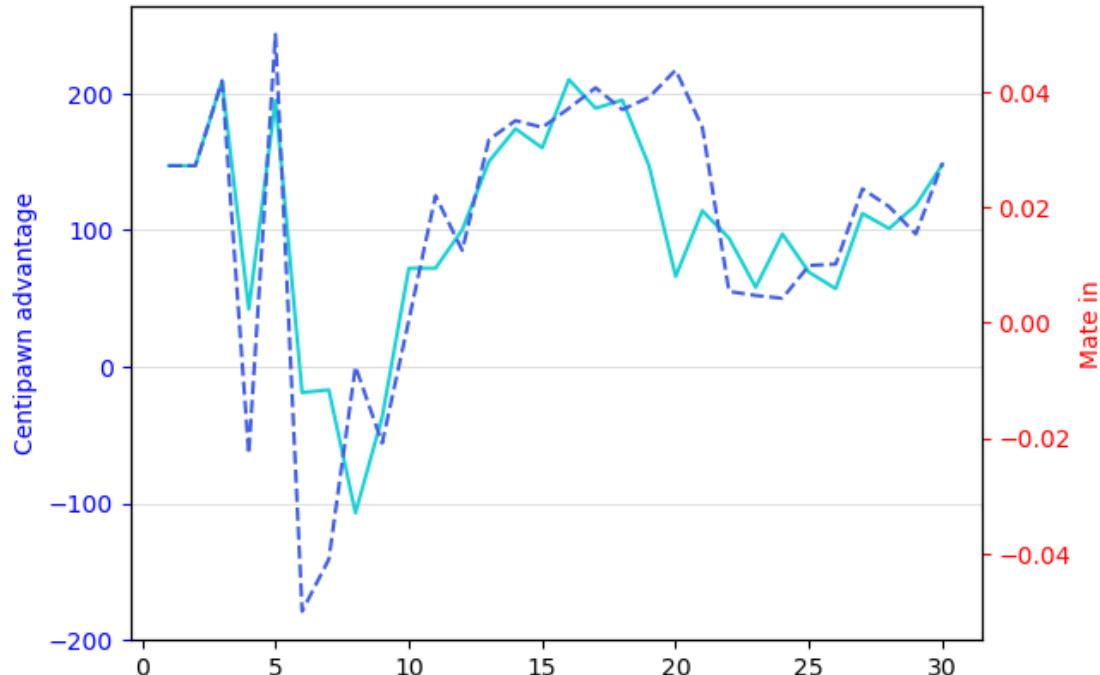
```

KeyboardInterrupt:

[]:

[]: plotevas(carlsen10p[0], carlsen10p[1], carlsen10p[2], 0)

MR verified at depth = [2, 3, 5, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 25, 26, 27, 28, 29, 30]



[]: