Current generation average score: 1113.968
Best strategy: (-0.44662478418885376, 10, -0.022483782974163313, 9)
Score of best strategy: 54992.0
Average score of this generation: 1113.968
File to save stats: stats.txt

1) Using holes + roughness only, I achieved 54992.0 average 1st generation. The issue turned out to be with a bug in my Tetris modeling, where when I was clearing rows I just subtracted the number of rows from the heights and did not account for rows where there were spaces. Using my original code with the corrected model, I managed to obtain the best strategy within 7 or 8 generations within approximately 1.5 hours(45 minutes at home → save generation → 45 minutes during English)

Stats:
File name: origsucess.pkl
Best strategy: (0.7113145494400512, 0.10932544485773943, 0.3155569190486618, -0.13813796345077067, 0.9192070099383098, -0.8641858829168829)
Score of best strategy: 69488.0
Average score of this generation: 13078.96
Weights in heuristic:

```
def heurestic(board,strategy,rowsCleared,columns):
    """
    Board shall be stored in reverse row major order
    """
    if board == "GAME OVER":
        return -40000000
    value = 0
    a,b,c,d,e,f = strategy
    avg_val,max_val,wells,diff,numHoles = highest_heap(board,columns)
    value += a * avg_val # average height
    value += b * max_val # maximum height
    value += c * wells # minimum_height
    value += d * diff # difference between maximum and minimum height
    value += e * rowsCleared # rows cleared
    value += f * numHoles # number of holes
    return value
```

- The average height, rows cleared, and number of holes are most important (have the highest coefficient).

- As consistent with canonical Tetris strategy, a board with many holes is penalized in the strategy

3) I did not enjoy the assignment because most of the time was spent optimizing the modeling function.