Report on the development of 2048 AI Agent

1. **Overview**

   This paper describes the evolution strategy, the architecture, and the performance aspect of MyAgent.py AI player with which I participated in the 2048 Game Contest. The agent was constructed in a heuristic, as opposed to search tree, manner, where selecting moves is prioritized by heavier evaluation on the value of a move rather than the search tree such as the Expectimax. This was a strategy that was efficient in computation bearing in mind the 0.5 second restriction that the server had in moving.

2. **Agent Strategy**

   The essence of the agent lies on the heuristic evaluation instead of complete game-tree searching. The findMove function enlists all possible legal moves, calculates the resulting game states with a custom evaluation function and selects as the move with the best score. In order to avoid crashes, the agent defaults to using a legal move at the start (setMove(legal[0]).

   The important elements of the strategy are:

   Tradition: fail-safe fallback move at start.

   The non-profound estimation of heuristics.

   Ordering of moves to expose probable beneficial moves at the beginning of the assessment.

3. **Heuristic Evaluation**

   In order to determine a utility value to each of the possible states, the evaluate() function adds up a number of weighted features:

   Features Used:

   Empty Tiles: Promotes having an open board.

   Merge Count: Reward movements which permit merging of neighbouring tiles. Monotonicity: Encourages the rows and the columns with increasing/decreasing values.

   Maximum Tile: Favors This is the one where the maximum tile can be added here.

   Corner Bonus: Rewards more points when it is possible to place the maximum tile on the top-left corner.

   Weights Used: python Copy Edit { 'empty': 100, 'merge': 40, 'monotonicity': 20, 'max_tile': 2500, 'corner': 500 }

   We tuned these weights by trial runs so as to balance exploration (free tiles) and exploitation (merging tiles, corner control).

4. **Move Ordering**

   To enhance the quality of moves and offer higher probabilities of selecting best moves within short time:
   The priorities in moves are according to:
   Amount of empty tiles that were the result.
   Or the max tile is up left.
   The scores of every move are ranked and ordered in descending order before evaluation.
   Such an rankings also highly accelerates evaluation and sees that that this order considers moves that preserve control to be considered first.

5. **Improvements and Performance**

   Random or Static Agent Initial Baseline
   Standard Score: 1,000-2,000
   Max Score: -5,000
   Post-Heuristic-Based Tuning
   Mean: 15777.52
   High Score: 50, 236
   The lowest score: 15,054
   Matches: 50
   The agent has illustrated:
   Powerful cornering control and merge uniformity.
   Very repeatable test run performance.
   Superb tile control with features of the monotonicity + merge count.

6. **Fallback Mechanisms**

   The agent will be well guarded against:
   Move selection crashes.
   Illegal move returns when the timing is very tight.

7. **Future Work**

   Although the existing agent is working well, further improvement can be as follows: Expectimax or Alpha-Beta with ad-aptive depth.
   Adaptive heuristic modification at the gameplay phases.
   Speedups by pattern-based caching of repeated board configurations.
   Heuristic tuning using training data in a learning sense.

8. **GitHub and Submission**

   The Improvements and all the versions were pushed into the GitHub Classroom and this document was added as Development.pdf.