Tetris Meets StackRank

Paul Chesnais (pmc85) and Sam “Sven” Svenningsen (sjs382)

Goal

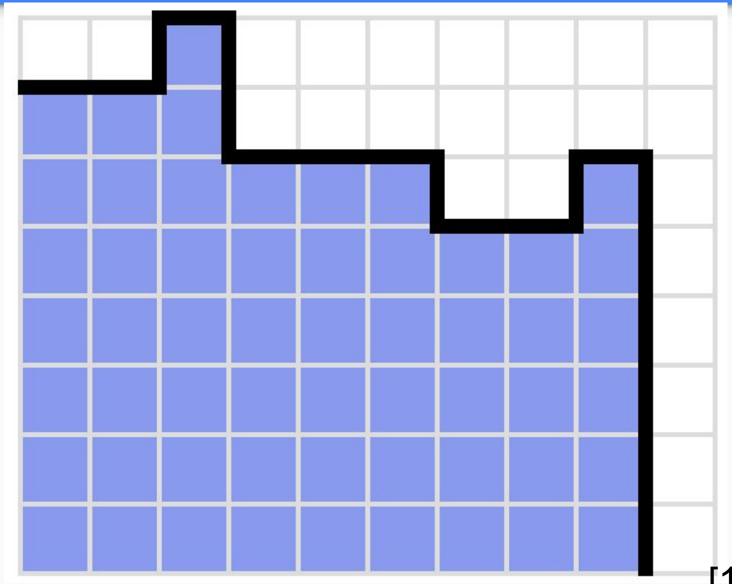
- Getting a computer to play Tetris
  - Tetris is NP-Complete.
  - We approximate a solution that plays well according to human standards.
- We do so by ranking the contour (the shape of the top of the stack) then choosing the sequence of Tetris pieces that lead to the highest ranking stack.

StackRank Algorithm

- Given arbitrary stack, calculate its rank given the rank of the stacks it can reach and the stacks that reach it.
- To solve this we did three things:
  - Created contours to simplify the game state representation
  - Mapping to connect the contours that are reachable from one another relate to each other
    - Similar to the idea of web pages that link to one another in Google's PageRank
  - Ranking to rank each contour and decide where to put each piece

Contours

- Represent a stack by only keeping track of the shape of the stack's contour. The stack's contour is defined by the height of each individual column minus the height of the shortest column.
  - Ignore absolute difference greater than 4 since it does not affect our choices
- We maintain the guarantee that the stack is always full, i.e. never has holes, and one of the outermost columns is always kept empty.



[1]

0,1,-2,0,0,-1,0,1 -> 45244345_9

Mapping Stage

- The total set of stacks is a graph, with a directed edge connecting stack s to stack s' if and only if s can reach s'.
- The contour approximation of a stack’s features can only represent 9^8 stacks, and as such, this is how many stacks the algorithm will attempt to rank.

Mapping Stage (continued)

- The Mapping Stage computes the complete graph of stacks as a map between:
  - (stack s, tetromino t) -> [stack s’,stack s”,...]
- The precomputation is required because this graph is quite large

Ranking Stage

- Given a stack s, and an array a the ranks of the stacks that it can reach, compute rank(s, a).
- What is a good measure of a stack's value?

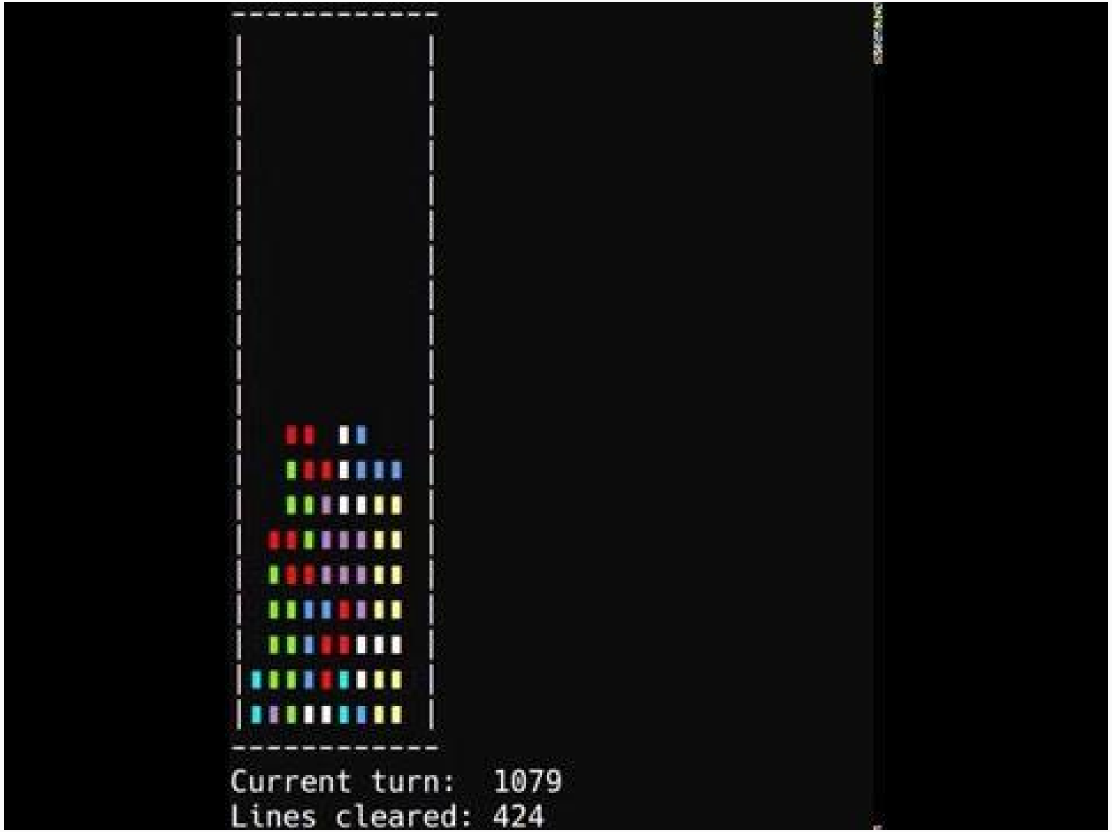
Ranking Stage (continued)

- **Naive Connectivity**: For a given stack, its rank is the sum of ranks of all the stacks that it can reach.
- **Average Connectivity**: For a given stack, its rank is the average rank of all the stacks that it can reach.
- **Majority Rules**: For a given stack, its rank is defined by the following: 0 if for more than half of the tetrominoes cannot be placed on this stack without creating a hole, else the sum of the ranks that this stack can reach over the number of tetrominoes, i.e. 7.
- **Pessimistic Evaluation**: (current) was designed to account for the absolute worst case scenario. It is quite similar to Majority Rules, but sets the score to 0 when a single tetromino cannot be placed on the stack

Game Tree Search

- A rank for all possible stacks is equivalent to a board/stack evaluation function
- This means that the agent can play the game using regular Game Tree Search techniques
- Unfortunately, the branching factor is at 190, meaning that the search depth must be limited to 4 for the player to make its decision in a reasonable amount of time

Current Version



Conclusion

- Beats the NES world record by 500 pieces on average
- We believe that every possible optimization was made to this player’s strategy, within the bounds of the problem we set for ourselves
- In order words, we’ve reached the maximum potential for a ranking strategy that only uses a naive Game Tree search and an approximation of the board evaluation function.

Sources

[1] http://www.ryanheise.com/tetris/tetris_artificial_intelligence.html