CS 182 Final Project Proposal

Aaron Sachs, Everett Sussman & Jan Geffert

November 1, 2017

Problem: *2048* is a puzzle game which gained popularity in 2014. Played on a 4x4 grid, the aim is to attain puzzle pieces with high values by moving the tiles on the board in intelligent ways [1]

The game mechanics are interesting as they incorporate both deterministic actions shifting the tiles to the top, left, bottom, right and probabalistic elements, that is randomly appearing tiles.

Solutions: We intend to build several agents that solve *2048*, including:

1. a **greedy search** agent that picks the move which maximizes the highest-valued tile,
2. a **greedy search** agent that picks the move which maximizes the number of empty squares (empty square),
3. a **heuristics-based expectimax** agent with a finely tuned evaluation function that is a linear combination of features. We might want to try to find the optimal weigthing of the different features by running local search algorithms on top of expectimax,
4. a pure **Monte Carlo simulation search** agent,
5. an **MDP** agent with knowledge about the underlying probability distributions determing the placement and value of new tiles,
6. a **Q-learning reinforcement learning agent** that learns these distributions / is robust to changes,
7. a **Deep Q-learning** reinforcement learning agent (TODO: what exactly does it do),
8. a **Monte Carlo Tree Search agent** (TODO: specify what that means, I am not sure),
9. and possibly, an agent learning from successful human play.

[1] Try it for yourself on `http://gabrielecirulli.github.io/2048/`

Metrics: For each algorithm, we plan to measure the following characteristics:

- What is the performance of the algorithm, i.e. the distribution of **final scores** and the distribution of the **values of the maximum tile** at the end of a game?
- What is the distribution of the **number of moves** that the algorithm takes to achieve the final solution.
- Are their certain **patterns** or **particular strategies** that emerge?
- How **complex** are the algorithm and the underlying data structures? What is the runtime, theoretically and practically, and how much space is required.
- In the case of learning algorithms: How many **iterations** are **needed** to reliably reach certain scores?

We furthermore plan to conduct a non-representative study of the average level of human play to be used as a benchmark. (TODO Should we? :D)

Development: The game mechanics are fairly simple allowing us to re-implement the game in a local Python testing environment. Having done that, we will first focus on designing a highly performant *2048* player and then try to optimize the algorithm do reduce runtime and/or space usage.

Resources:

- The official *2048* implementation which is available under the MIT license at https://github.com/gabrielecirulli/2048.
- The relevant chapters of AIMA
- Silver, David (2009). Reinforcement Learning and Simulation-Based Search in Computer Go `http://papersdb.cs.ualberta.ca/~papersdb/uploaded_files/1029/paper_thesis.pdf`
- Silver, David; Huang, Aja; Maddison, Chris J.; Guez, Arthur; Sifre, Laurent; van den Driessche, George; Schrittwieser, Julian; Antonoglou, Ioannis; Panneershelvam, Veda (2016-01-28). Mastering the game of Go with deep neural networks and tree search. `http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html`

TODO: "3. examples of expected behavior of the system or the types of problems the algorithms you investigate are intended to handle"

TODO: "5. and a list of papers or other resources you intend to use inform your project effort. This list will form the core of your project report reference list. If your project includes anything unusual (such as having significant systems demands), please state this as well."