General game-playing applied to 2048

Paul Lisker and Ahmed Ahmed

December 16, 2016

1 Introduction

General Game Playing (GGP) is a subfield of Artificial Intelligence that seeks to develop agents that are capable of playing many games rather than being tailored and limited to particular game. GGP has been a popular subject of AI research, particularly over the past decade. The goal of GGP is to create programs that are capable of successfully playing more than one game successfully. As such, a successful GGP agent will not have algorithms tailored to a particular game, but rather generalized algorithms that can be successful when applied to various different games (though occassionally of the same domain—that is, games that are similar in nature). For an illustrative example on the difference between a regular game playing agent and a GGP agent, we can look at IBM's famous Deep Blue computer. This computer ran an AI program that was tailored to the goal of winning in a game of chess; however, this program would have been unusable for other games. On the other end of the spectrum, successful GGP programs can play—albeit perhaps less effectively as a dedicated algorithm—more games than just one, or play games that would otherwise be intractable for specialized algorithms.

The requirements of General Game Playing, then, necessitate the use of algorithms that are not domain-specific. in other words, it must use algorithms that can operate with a broad spectrum of games without being dependent on shared rules between said games. Among this class of algorithms is the notable GGP algorithm of Monte Carlo Tree Search (MCTS), a class of algorithms for finding optimal choices in a decision space by taking random sample simulations and building a search tree according to their results. We explore this algorithm with more in Section 4.

In this project, we apply MCTS to the domain of games including the popular puzzle game 2048, analyze its performance, and compare it to other successful AI approaches.

Therefore, our first goal is to develop a program that can win the game. In good competitive spirit, it naturally follows the secondary goal of maximizing scores in order to compete with the better 2048 programs that have been previously published. Subsequently,

we analyze the performance of our implementation with and without the use of game-specific heuristics. Finally, we also analyze the flexibility afforded by our use of GGP techniques, by applying our implementation to an expanded class of similar games we call *generalized-2048* created by allowing variation in some of the rules of standard 2048.

The specific algorithm we chose to implement is Upper Confidence Bounds for Trees (UCT), a common flavor of Monte Carlo Tree Search (MCTS). There has been a boom in research on MCTS algorithms in the past decade after MCTS AI players were successful at the classical Chinese game and universal challenge for AI game-playing, Go. MCTS provides an alternative to the traditional approach to combinatorial games, minimax algorithms, especially in games of nontrivial size for which no reliable heuristic has been developed situations in which minimax often fails. However, in cases where reliable domain-specific heuristics have been developed - as is the case for 2048 - either MCTS or minimax may be applicable.

MCTS is essentially a class of techniques for traversing the game-trees. Such game-trees are an especially convenient model for *combinatorial games* such as 2048 - that is, sequential, discrete, full-information games. Therefore, this project most directly draws off of the concepts of uninformed and heuristic search that we explored earlier in this course. Additionally, MCTS does have a Monte Carlo portion - that is, it uses sampling in order to approximately inform its search. This concept of using of sampling methods to find approximate answers to intractable problems was explored in this course in the context of particle filters in Bayes Nets.

2 Background and Related Work

2.1 Introduction to 2048

2048 is a puzzle game developed by Gabrielle Cirulli that was a viral hit in 2014. Each state in the original game consists of a 4×4 grid, each cell of which is either empty or contains a number that is a power of 2. The player choose to slide all tiles in one of four directions, and whenever two tiles of the same number collide, they combine to form the tile of their sum. After each move, a random tile is generated. The game ends when there are no more available moves for the player i.e. all cells are non-empty and there are no available merges. The ultimate goal is to continue merging tiles until a 2048 tile is created.

2.2 Published solutions

As the game went viral in 2014, the AI game-playing community went to work at publishing several automated 2048 players. Most approaches that the authors are aware of have

been minimax implementations, so most focused on developing useful heuristics and optimizing for faster run times in order to increase the depth of their minimax searches within reasonable time, with one of the deepest being a depth 7 search published by XXXXX in XXXX. This solution was also one of the most, as it reaches. Our implementation distinguishes itself from these solutions, in that it doesn't use minimax at all, instead using MCTS as our primary method.

2.3 Similar applications of Monte Carlo Tree Search

XXX applied MCTS with great success to the combinatoric puzzle game sudoku and XXXX. Blablabla.

3 Problem Specification

In this project, we aim to build an artificial intelligence program that can play well at an arbitrary game in the class of games, **generalized-2048**. We define generalized-2048 as the set of games with rules equivalent to those of standard 2048, except: x y z.

1. It may have an arbitrary board size $n \times m$ instead of just the 4×4 board in standard 2048.
2. It may have an arbitrary score function R, instead of the standard 2048 instead of the cumulative score of merged tile values.

4 Approach

- Introduce our approach: general game playing + lessen the generality
- Explain how this allows us to expand our domain to larger class of games
- (AUTHORS) showed that the class of 2048 games with only the first generalization are NP-complete (SOURCE).

Algorithm 1 Here is the algorithm.

procedure MYALGORITHM(b)
  $a \leftarrow 10$
end procedure

	Score
MCTS	XXX
MCTS with heuristic	YYY
Heuristic search	ZZZ

Table 1: A performed better than B and C at standard 2048.

5 Experiments

Analysis, evaluation, and critique of the algorithm and your implementation. Include a description of the testing data you used and a discussion of examples that illustrate major features of your system. Testing is a critical part of system construction, and the scope of your testing will be an important component in our evaluation. Discuss what you learned from the implementation.

5.1 Results

For algorithm-comparison projects: a section reporting empirical comparison results preferably presented graphically.

6 Discussion

Summary of approach and results. Major takeaways? Things you could improve in future work?

A System Description

Appendix 1 A clear description of how to use your system and how to generate the output you discussed in the write-up. *The teaching staff must be able to run your system.*

B Group Makeup

Appendix 2 A list of each project participant and that participants contributions to the project. If the division of work varies significantly from the project proposal, provide a

brief explanation. Your code should be clearly documented.