# Solving the Generalized Form of the Game of Set Efficiently

Steven Takeshita
Adviser: Zachary Kincaid

## Motivation and Goal

The Game of Set was created in 1974 and published in 1991. The game consists of $3^4 = 81$ unique cards. Each card has 4 properties and one of three values. A valid set of three cards is one in which for each property, they all either have the same or different value. At the beginning of the game, 12 cards are shown, and players must locate valid sets. Once a set is found or no set exists, three new cards are added. The most number of sets collected at the ends win. Therefore, a natural extension of this problem is to determine how fast can we find sets. The goal of my project is to create a solver that locates a set efficiently in practice for a game with $p$ properties and $v$ values. The Game of Set is an interesting problem for dynamic algorithms in that when three new cards are added, the algorithm should be able to build off of previous knowledge. A way to utilize this past information could be helpful in applying similar principles to other problems that could lend itself to dynamic programming.

## Problem Background and Related Work

The Game of Set has been researched thoroughly for education and as a motivating example for cap sets, but little research exists about solving a generalized version. Chaudhuri et al (2003) proved that the generalized version of Set is in fact NP-Complete through a reduction from perfect-Dimensional Matching, a known NP-Hard problem [1].

This paper proves computational complexity bounds, but does not solve the problem in practice. Solvers do exist on the internet and can be readily found [3]. Some solvers even employ image processing to identify sets [2]. However, the solvers are constrained to have four properties and three values. Therefore, the solver I will build will be able to solve the generic case of Set in which it has $p$ properties and $v$ values.

## Problem Definition

A deck will contain $v^p$ possible cards where the cards have $p$ properties and $v$ values. Initially, $v * p$ cards will be outputted as the starting layout. Identify an arbitrary set of $p$ cards, in which for all properties the values are either the same or all different. Remove this set, or if no set exists, add $p$ more cards from the remaining $v^p - p$ cards. If no set exists, add $p$ more cards. Find a total of $n$ sets, where $n \leq v^{p-1}$.

By removing the first set of cards seen, this will simulate real gameplay, in which the goal is to find sets as fast as possible. By finding a generalized $n$ number of sets, as cards are added

and removed, the dynamic algorithm will hopefully see some speed up. This further simulates gameplay in which the goal is to find a collection of sets and it is likely that there might not exists sets or sets will be found causing new cards to be added.

# Approach

Generalizing the Game of Set has been done academically, but no practical solution to locate sets exists. A reduction to SAT creates a solver to theoretically always verify whether a set exists, but this does not verify a set within reasonable time if we consider the the poly-time transformation and the fact that it needs to recalculate a set every time new cards are added, not using information from previous rounds to verify it faster. Therefore, my approach will use previous knowledge of the absence of sets to quickly discover sets in a new showing of cards. Set's changing probabilities of discovering a Set also lends itself to the dynamic algorithm approach. Norvig (2017) showed that on a fresh layout of the game with 12 cards, the ratio of games in which there exists a Set to those that do not contain a set, is 29:1. But as the game plays on, the ratio drops to 15:1, a 50% less chance of finding a set. For 15 cards, the ratio drops to 4% of its original ratio [4]. Therefore, as the game is iteratively played in which Sets are taken out, the game becomes immensely harder. With a dynamic algorithm approach, the solver I build should be able to beat the SAT solver especially with this added difficulty as Sets are removed from the board.

# Plan

The project will be divided into two major and one minor parts. The first is to create a randomization algorithm to create a random starting board ($v * p$ cards) with $p$ properties and $v$ values. These cards need to be randomly selected from $v^p$ possible cards and when sets of size $v$ are iteratively removed, they must be able to retrieve $v$ new cards such that the new cards are random and without replacement. Both the SAT based solver and dynamic algorithm must be able to take this output as input.

The second part is finding a reduction to the NP-Complete problem, SAT. Then I will code the algorithm up to solve the equation and find a Set in the board for varying values of $p$, $v$, and number of sets to find.

The last part will be to create a more efficient algorithm. This algorithm will use a dynamic algorithm to speed up the search for iterative rounds. I will then compare this algorithm against the SAT solver program for varying values of $p$, $v$, and number of sets to find.

**Tentative Plan:**

3/2 - Create randomization algorithm and begin reduction to SAT. Write up reduction to SAT.

3/9 - Finish reduction to SAT solver and implement it. Run correctness/stress tests. Write up motivation and goal.

3/16 - Write up problem background and related work. Begin work on more efficient algorithm implementation.

3/30 - Continue work on dynamic algorithm. Start writing up approach as the algorithm takes shape.

4/6 - Finish more efficient algorithm implementation. Run correctness/stress tests. Finish writing approach and start writing implementation.

4/13 - Fix lingering bugs in code. Ensure randomization works with implementations. Finish writing implementation section.

4/22 - Finish slides for Oral Presentation and practice. Finish writing results section. Ensure that code is sound and works for all cases.

4/27 - Finish writing conclusion section. Revise all previous parts.

5/3 - Finish final report and send to Professor Kincaid to review

5/7 - Finish revisions and submit written proposal.

# Evaluation

To test the randomization algorithm for choosing cards, I will randomly generate many $n * p$ sized decks, and calculate the distribution of this starting deck. To verify that the algorithm to pick the next $p$ cards is uniform from the remaining cards, I will find the distribution of $p$ cards that are iteratively chosen for a random beginning deck and sample this many times. The randomization will be successful if both are uniform distributions.

With two built solvers for the same problem, I will time how long it takes to iteratively find a given number of sets. I will adjust the number of sets, values and properties. Hopefully at some levels, most likely higher number of sets, values, and properties, the dynamic algorithm solver will finish faster than the SAT based solver.

# References


[1] Kamalika Chaudhuri, Brighten Godfrey, David Ratajczak, and Hoeteck Wee. On the complexity of the game of set, 2003.

[2] Fernando San Martin Jorquera and Amanda Legge. Set card game solver using image processing techniques on smart-phone photos, 2013.

[3] Steve Nolte. Javascript set game solver.

[4] Peter Norvig. The odds of finding a set in the card game set, 2017.