Automated Playtesting in Collectible Card Games using Evolutionary Algorithms: a Case Study in HearthStone

Authored by:
P. Garcia-sanchez aDept. of Computer Engineering. University of C´adiz, Spain
Alberto Tonda UMR 782 GMPA, INRA, Thiverval-Grignon, France
Antonio M. Mora cDept. Computer Sciences and Technology, Universidad Internacional de La Rioja, Spain
Giovanni Squillero Politecnico di Torino, Italy
Juan J. Merelo Dept. of Computer Architecture and Computer Technology, University of Granada, Spain

Published in:
Knowledge-Based Systems, Volume 153, 1 August 2018, Pages 133-146

s1300164 Koki Nakagawa

Background

- Collectible Card Games (CCGs) are turn-based card games.

- Each players select cards from an extensive set of cards and build their decks in order to be able to make powerful combinations in the game.

- Cards available for the game are added on a regular basis and sometimes removed.

Background

- When new cards are added, a through analysis is necessary to maintain game balance.

- It is difficult to fully test the impact a new set of cards can have on all aspects of the game.

An automated testing tool (playtest tool) is needed to evaluate the impact of adding new cards in advance.

Purpose

- This paper proposes a playtest tool that can be used during the development of new content for a DCCG (digital version of CCG).
- They focused on automated deck building for a popular DCCG, HearthStone.
- They use ***Evolutionary Algorithm (EA)*** to automatically build and test viable and competitive decks.

Approach

- The approach uses an AI that automatically generates competitive decks using EA to evaluate the target metagame (the type of decks that one is expected to find in a specific ladder) and its effectiveness.

- Cards that appear frequently in the evolved decks or decks with extremely high win rates could be analyzed by experts and identified as unbalanced.

- Using a new evolutionary operator named ***smart mutation***, the deck is modified according to a human-like heuristic.

- For the ***fitness function***, each deck is evaluated in a simulated match against a wide and diverse set of human-made decks, selected from the most competitive in the metagame.

Method

EA

- EA is a program inspired by mechanisms of biological evolution and is usually applied to optimization problems.

- One of the advantage of EA is that it can obtain optimal, or near-optimal, solutions that are difficult for human experts to find.

- In this paper, this is used to identify outperforming cards.

Method

μGP

- μGP is a general-purpose EA designed to easily tackle different optimization problems out-of-the-box.

- An individual is defined as a deck of 30 cards taken at random from a specific card pool.

- μGP's evolution operator can change a card in a deck to another card or crossover two decks.

Method

Smart Mutation

- During deck building, human players swap cards that are close to the cost of use, to improve the deck while maintaining the nature of the deck.

- The evolution operator, called smart mutation in the µGP framework, replaces a random card in the parent deck with another card selected with uniform probability from among all available cards within +1/-1 of the original card's cost of use.

Method

Fitness evaluation

- The fitness function is divided into three parts.

1. number of errors (minimize):

   This measure takes into account whether the deck violates any of the rules of the game. Decks with this fitness value greater than 0 are not evaluated further and all remaining fitness values are set to the smallest possible value. This fitness value is minimized.

Method

Fitness evaluation

2. Number of Wins (Maximized):

   This is the total number of wins obtained after 16 games against each deck in the target metagame. This fitness value is maximized.

3. Standard deviation of victories (minimize):

   This value is calculated by computing the standard deviation of the number of wins against other opponents. If the deck has the same number of wins against all opponents, its standard deviation is optimal. This fitness value is minimized.

Method

Fitness evaluation

- When comparing two individuals

  First, the number of errors for each is considered, and the one with the lower number of errors is considered the winner.

  If both individuals have the same number of errors, the number of wins is used for comparison, and the one with the higher number of wins is chosen.

  Finally, the standard deviation is used only if they are tied.

Method

Pseudo-code

Algorithm 1 Pseudo-code of the EA adopted in the proposed approach.

```
population ← initializePopulation()
evaluate(population)
while stopping criterion not met do
  for child in offspring do
    parents ← tournamentSelection(population)
    choose evolutionary operator to apply
    if operator = crossover then
      offspring ← offspring + crossover(parentA,parentB)
    else if operator = mutation then
      offspring ← offspring + mutate(parentA)
    else if  operator = smartMutation then
      offspring ← offspring + smartMutation(parentA)
    end if
  end for
  evaluate(offspring)
  update internal parameters
  population ← population + offspring
  sort(population)
  reduce population to initial size by removing worst individuals
end while
```

Method

MetaStone

- MetaStone is an open-source HearthStone simulator.

- It can manually create decks and simulate matchups to obtain statistical metrics such as the number of turns it took to win, damage inflicted, etc.

Method

MetaStone

- The AI engine can select different heuristics based on the score given to the actions evaluated in each turn, taking into account the combination of card weights used.

- ❏ Play Random Behavior (PRB): Actions to be played are randomly selected.
- ❏ Greedy Optimize Move (GOM): AI selects each move in order of score.
- ❏ Greedy Optimize Move (GOT): AI selects the move with the highest score, calculated considering the current game situation, among all possible move combinations.
- ❏ Flat Monte Carlo Tree (FMC): During a specified number of iterations, the AI simulates random moves as far as possible until the end of the game and calculates the score considering the future game state.

Experimental evaluation

- In this paper, nine human-designed decks were considered as opponents.

- To estimate how well MetaStone could play against the prepared decks, a first tournament was conducted using all four possible AI combinations, with each deck playing 32 games against all other decks.

Experimental evaluation

Table 1: Number of games won by the human-made decks in our preliminary analysis (of a total of 11,520) aggregated by deck, after confronting all deck/AI combinations (every deck was played using all the available AIs against the others).

Deck name	Games Won	Percentage of victories
Midrange Hunter	1,669	14.48%
Aggro Paladin	1,502	13.03%
Mage Tempo	1,494	12.96%
Control Warrior	1,295	11.24%
Midrange Druid	1,286	11.16%
Mech Shaman	1,233	10.70%
Shadow Madness Priest	1,122	9.73%
Warlock MalyLock	1,101	9.55%
Oil Rogue	809	7.02%

Experimental evaluation

Table 2: Number of games won in our preliminary analysis (of a total of 11,520) of all decks/AI combinations.

Deck	FMC	GOM	GOT	PRB
Aggro Paladin	302	420	579	201
Control Warrior	187	395	359	354
Mage Tempo	281	442	648	123
Mech Shaman	183	399	417	234
Midrange Druid	304	341	470	171
Midrange Hunter	417	437	561	254
Oil Rogue	52	289	419	49
Shadow Madness Priest	90	445	510	77
Warlock MalyLock	95	462	357	187
Total by AI	1911	3630	4320	1650

Experimental evaluation

- This paper sets AI GOT as the AI to beat in the remaining experiments.

Table 3: Number of games won using the GOT AI with each deck. Each deck shown in this table played against other decks using the same AI (256 games per deck).

Deck name	Games Won	Games Lost	Percentage of victories
Aggro Paladin	182	74	71.09%
Mage Tempo	177	79	69.14%
Shadow Madness Priest	152	104	59.37%
Midrange Hunter	143	113	55.85%
Mech Shaman	119	137	46.48%
Oil Rogue	106	150	41.40%
Control Warrior	104	152	40.62%
Midrange Druid	85	171	33.20%
Warlock MalyLock	83	173	32.42%

Parameters

- μGP has been configured with the parameters reported in Table 4 for all the experiments.

Table 4: Parameters used by the EA (μGP). The activation probabilities of the operators are self-adapted. For more information on the parameters, see [20] or visit `https://sourceforge.net/p/ugp3/wiki/Home/`.

Parameter	Meaning	Value
μ	Population size	10
λ	Operators applied	10
α	Self-adapting inertia	0.9
σ	Initial mutation strength	0.9
τ	Size of the tournament selection	[2-4]
G	Number of generations	200
S	Strategy	$(\mu + \lambda)$
R	Replacement mechanism	Generational
e	Number of opponent decks	8
t	Number of games per opponent deck	16
Operators used	`singleParameterAlterationMutation`	
	`onePointCrossover`	
	`twoPointCrossover`	
	`Smart Mutation`	

Result

- This paper examined the percentage of wins for each generated deck by the method proposed in this paper (with and without using Smart Mutation) and the original human-designed decks.

Result

Table 5: Percentage of wins obtained by the human-designed decks in the preliminary analysis (Subsection 5.2), versus the percentage of wins obtained by the best individuals evolved using our method with and without Smart Mutation at the end the run for each class. Best results are highlighted.

Class	Human	No Smart Mutation	Smart Mutation
Paladin	71.094 %	**82.813** %	81.250 %
Mage	69.141 %	76.563 %	**81.250** %
Priest	59.375 %	78.906 %	**83.594** %
Hunter	55.859 %	76.563 %	**78.906** %
Shaman	46.484 %	**72.656** %	68.750 %
Rogue	41.406 %	67.188 %	**77.344** %
Warrior	40.625 %	67.188 %	**73.438** %
Druid	33.203 %	78.125 %	**80.469** %
Warlock	32.422 %	60.938 %	**67.188** %

Result

- All decks generated by the method in this paper had a higher win rate than the original human-designed deck.

- The use of smart mutation resulted in improvements in all classes except shaman and paladin.

- In the case of the Paladin and Shaman, the version of the EA using Smart Mutation lags only a few percentage points behind the other EAs.

Thus, the use of smart mutation still appears to be beneficial.

Discussion

- A histogram of the card cost of each deck (also called the mana curve) is shown in Figure 2.

  With a few exceptions, most of the decks generated have a curve shifted to the left, which is clearly a sign of an aggressive deck.

Discussion

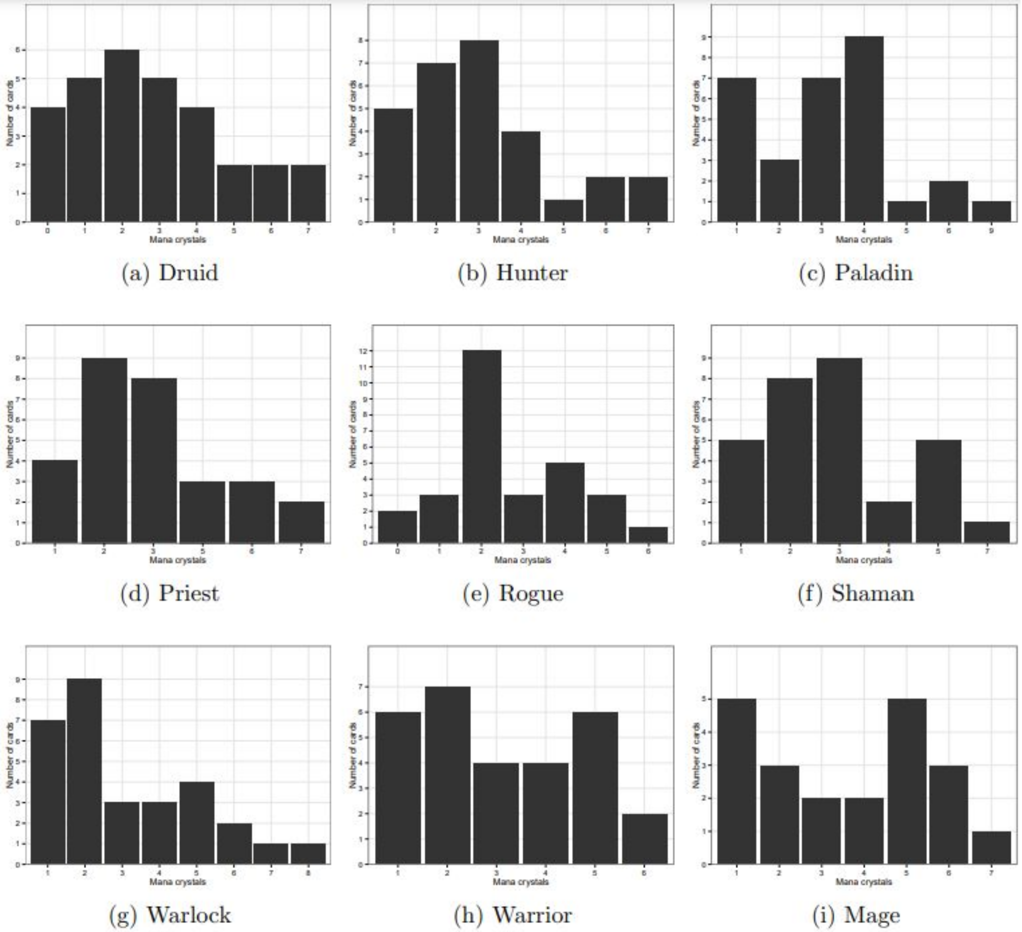

Figure 2: Number of cards by cost, also called *Mana Curves*, of each generated deck. As it can be seen, they are balanced to the left, meaning the generated decks tend to be aggressive and quick to play, as they have a larger number of low-resources (crystals) cards.

Discussion

- Even lacking any specific instructions to do so, the EA is able to create decks with multiple copies of the same card.

- The evolved decks feature a large number of cards that are considered effective by human players.

These are actions that make sense to build a deck with a high win rate.

Discussion

- Several cards widely considered less than optimal are also in the best evolved decks.

This is likely because, even if they were not a threat to humans, they were a threat to greedy AI.

Discussion

- While the proposed approach seems effective, there’s an important weakness to consider

- ❏ While MetaStone is an effective solution for assessing suitability, it is not up to the level of human players.
- ❏ It is difficult to claim that an AI using a greedy approach will generate decks suitable for playing against humans, who can deploy extremely diverse strategies.

Conclusion

- This paper proposes a method to automatically find unbalanced cards and card combinations by applying an Evolutionary Algorithm (EA) to optimize decks of HearthStone, a Collectible Card Game (CCG).

- All decks were evaluated against the best human-designed decks for a given season using the framework MetaStone AI to simulate the game.

Thank you for your attention!