Island Model Genetic Algorithm for Hanabi

Andrea Cavallo
Politecnico di Torino
Computational Intelligence
Torino, Italy
s287965@studenti.polito.it

***Abstract*—This work describes the implementation of a genetic algorithm for the evolution of a player for the cooperative game Hanabi. The goal of the experiment is to find the best combination among a set of predefined rules to optimize the ability to play with other players. The work is inspired by [1], with some differences in the implementation of the genetic algorithm. The implemented rules, also inspired by [1], are described in Table IV.**

I. Introduction

Hanabi is a cooperative game where players share a common objective and should adapt their strategies to one another. As a matter of fact, each player knows the other players' cards but not his own, and should therefore provide wise suggestions to the others to allow them to play reasonably. This is a critical point for traditional approaches to games, since many algorithms are designed to exploit the opponent's weaknesses and are not able to achieve good cooperation. Many of the proposed approaches to the game of Hanabi are rule-based, i.e. players follow a set of hardcoded rules based on human experience. The aim of this work is, starting from a rule-based approach, to experiment the impact that genetic algorithms can have in the evolution of a player for Hanabi. In particular, a large set of possible predefined rules is implemented, according to the rules specified in [1], and the genetic algorithm finds the optimal ordering for them.

Parameter	Value
Genome length	29
Population size	25
Offspring size	50
Tournament size for parent selection	3
Mutation probability	0.1
Number of generations	100
Number of islands	4
Migration interval	10
Migration size	5
Elitism size	5

TABLE I: Parameters for the island model genetic algorithm

II. Method

The implemented method consists in a genetic algorithm using an island model to promote diversity among individuals. As a matter of fact, the initial experiments with a simple genetic algorithm proved that it was very difficult to escape the initial local optimum that was created by the random initialization of the population. Therefore, the island model was selected to allow a better exploitation of diversity.
The parameters used for the algorithm were chosen taking into consideration also the computational time required (a larger offspring size might have been beneficial, but the evaluation would have taken too long). The selected values are reported in Table I.

Since the goal is to find the best order among the available rules, the problem can be framed as a combinatorial problem, where the goal is to find the optimal combination of actions. Each rule is associated with an integer number, and a genotype is represented as a sequence of integers. The mutation of an individual is randomly selected between swap mutation, inversion mutation, scramble mutation and insert mutation. The crossover type is randomly selected between cycle crossover, partially mapped crossover and inver over. Parent selection occurs with a tournament. To avoid losing the best individuals, since the offspring size is not too large, also elitism is implemented (i.e. the best individuals are copied to the next generation).
The fitness of an individual is the average score it can achieve over 50 games. The games are played both with themselves (self-play) and with other players using a different strategy (mixed-play), to optimize the ability to adapt also to different kinds of players. In particular, the players implemented for mixed-play (i.e. those the genetic agent plays with in the evaluation games) are the main rule-based players in literature (as reported in [4]), described in the following.

- Osawa outer [2]: it has knowledge of what the other agents have been told already, to avoid repeating hints.

Rules
PlayIfCertain
DiscardUselessCard
TellPlayableCard
TellUnknownCard
DiscardRandomCard

- Cautios: it is derived from human gameplay. The agent plays cautiously, never losing a life.

Rules
PlayIfCertain
TellPlayableCard
DiscardUselessCard
DiscardRandomCard
RandomHintDiscard

- Piers [5]: it introduces a random play when the deck is empty, to try to achieve an additional point

Rules
PlayMostProbableIfDeckEmpty
PlayIfCertain
DiscardUselessCard
TellPlayableCard
TellUselessCard
DiscardUselessCard
TellRandomHint
DiscardRandomCard

- Van den Bergh [3]: it was created based on four main tasks:
  - If I'm certain enough that a card is playable, play it.
  - If I'm certain enough that a card is useless, discard it.
  - Give a hint if possible.
  - Discard a card

Rules
PlayProbablySafeCard(p>0.6)
PlayIfCertain
DiscardUselessCard
TellPlayableCard
TellUselessCard
TellMostInformation
DiscardProbablyUseless(p>0.4)
DiscardRandomCard
RandomHintDiscard

As a starting point, the performances of the rule-based players are evaluated using the average score over 100 games, both in self-play and in mixed-play. The results are reported in Table II.

Player	Score	
	self-play	*mixed-play*
Osawa outer	14.42	14.63
Cautios	15.22	15.01
Piers	14.84	14.76
Van den Bergh	14.40	14.73

TABLE II: Scores for rule-based players

To evolve the genetic player, the genetic algorithm is run four different times, once for each different number of players. In this way, the final strategy is optimized for a particular number of players. When a game starts, the implemented player automatically selects the most appropriate strategy depending on the number of players.

III. Results

Figure 1 reports the fitness for the different islands during the evolution of the genetic algorithm. It can be observed that the island model provides benefits to all islands: those that perform worse at the beginning achieve results similar to the best ones at the end. It should be noticed that the evaluation of an individual may vary at different generations, since the score is based on the result of 50 games. This may lead to having a lower best fitness in a subsequent generation, even though elitism is implemented.
Tables V, VI, VII, VIII report the strategies corresponding to the best individuals for each number of players.
The performances of the evolved algorithm are evaluated using the average score over 100 games with different number of players, both in self-play and in mixed-play. The results are reported in Table III. It is possible to notice that the performance are significantly better with respect to the initial rule-based players, especially for self-play.

Number of players	Score	
	self-play	*mixed-play*
2 players	18.32	17.29
3 players	17.69	16.62
4 players	16.79	15.67
5 players	15.27	14.67
overall	17.02	16.06

TABLE III: Scores for evolved player

IV. Conclusion

We have experimented the impact of a genetic algorithm in finding the optimal strategy to play a game of Hanabi. The results show that it is possible to improve the scores that rule-based players are able to achieve.
The island model for the genetic algorithm provides an improvement in the exploitation of diversity, since fit individuals in one island positively affect individuals in the others.

References

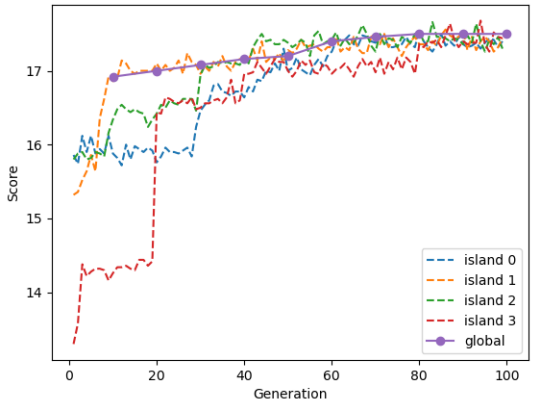
[1] R. Canaan, H. Shen, R. Torrado, J. Togelius, A. Nealen and S. Menzel, "Evolving Agents for the Hanabi 2018 CIG Competition," 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018, pp. 1-8, doi: 10.1109/CIG.2018.8490449.

[2] H. Osawa, "Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information." in AAAI workshop: Computer Poker and Imperfect Information, 2015, pp. 37–43.

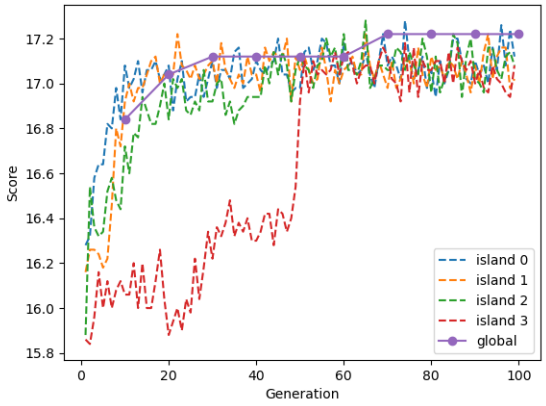
[3] M. J. van den Bergh, A. Hommelberg, W. A. Kosters, and F. M. Spieksma, "Aspects of the cooperative card game hanabi," in Benelux Conference on Artificial Intelligence. Springer, 2016, pp. 93–105.

[4] J. Walton-Rivers, P. R. Williams, R. Bartle, D. Perez-Liebana, and S. M. Lucas, "Evaluating and modelling hanabi-playing agents," in Evolutionary Computation (CEC), 2017 IEEE Congress on. IEEE, 2017, pp. 1382–1389.

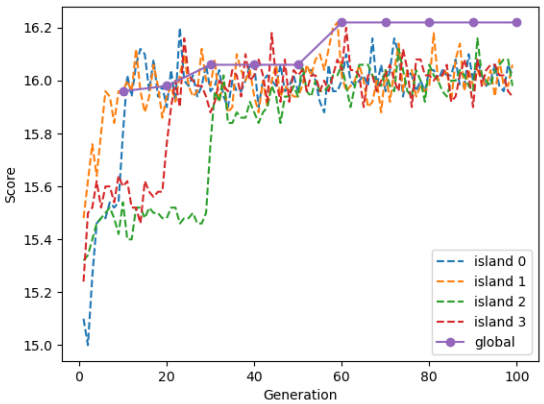
[5] J. Walton-Rivers, "Fireworks agent competition," access: 05/14/2018. [Online]. Available: hanabi.aiclash.com



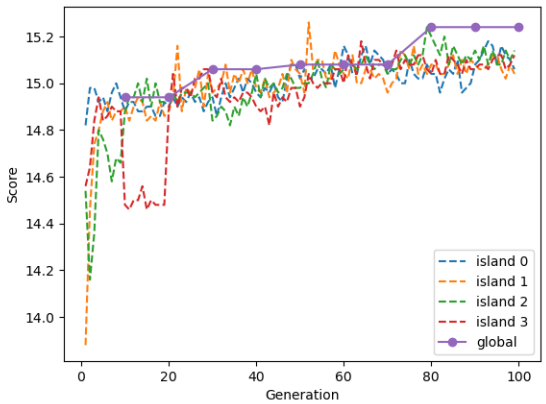
(a) 2 players



(b) 3 players



(c) 4 players



(d) 5 players

Fig. 1: Fitness for the different genetic algorithms

ID	Rule	Description
0	PlayIfCertain	Play a card if it is certainly playable
1,2,3	PlayProbablySafeCard	Play a card with a probability higher than p
4,5,6	PlayProbablySafeCardWithLives	Play a card with a probability higher than p if there are at least 2 remaining lives
7,8,9,10	PlayRecentlyHinted	Play a card hinted in the last turn if its probability of being playable is higher than p and there are at least 1 lives remaining
11	PlayMostProbableIfDeckEmpty	Play card with highest probability of being playable if there are no more cards in the deck
12	CompleteTellPlayableCard	Give a hint about a playable card such that the owner gets to know everything about it
13	TellAboutOnes	Tell a player about the ones
14	TellAboutFives	Tell a player about the fives
15	TellPlayableCard	Give a hint to any player about a playable card
16	TellUselessCard	Give a hint to any player about a useless card
17	TellMostInformation	Give the hint that maximizes the information acquired by the player receiving it (i.e. how many possible values of his cards are eliminated)
18	TellRandomHint	Give a random hint
19	TellUnknownCard	Give a hint about a card a player doesn't know all about
20	TellUnambiguous	Give a hint according to a scoring function that maximizes the probability of being playable for playable cards and minimizes the probability of being playable for non-playable cards
21	DiscardRandomCard	Discard a random card
22	DiscardUselessCard	Discard a card that is known to be useless
23	DiscardHighestKnown	Discard the highest known card
24	DiscardUnidentifiedCard	Discard a card with no knowledge about
25,26,27	DiscardProbablyUseless	Discard a card with a probability of being useless higher than p
28	RandomHintDiscard	Randomly give a hint or discard a card (it is always possible to do at least one of the two)

TABLE IV: Description of the implemented rules

Rules
PlayProbablySafeCard(p>0.8)
TellPlayableCard
PlayProbablySafeCardWithLives(p>0.4)
DiscardUselessCard
DiscardUnidentifiedCard
DiscardProbablyUseless(p>0.8)
DiscardRandomCard
PlayRecentlyHinted(p>0.01,l>1)
PlayRecentlyHinted(p>0.2,l>1)
DiscardProbablyUseless(p>0.4)
PlayProbablySafeCard(p>0.6)
PlayRecentlyHinted(p>0.01,l>0)
PlayMostProbableIfDeckEmpty
PlayProbablySafeCardWithLives(p>0.6)
DiscardProbablyUseless(p>0.6)
TellUselessCard
PlayProbablySafeCardWithLives(p>0.8)
DiscardHighestKnown
TellMostInformation
TellAboutOnes
PlayProbablySafeCard(p>0.4)
RandomHintDiscard

TABLE V: Evolved rules for 2 players

Rules
PlayProbablySafeCardWithLives(p>0.6)
CompleteTellPlayableCard
PlayIfCertain
PlayRecentlyHinted(p>0.01,l>1)
DiscardUselessCard
TellPlayableCard
PlayMostProbableIfDeckEmpty
PlayProbablySafeCard(p>0.8)
PlayProbablySafeCardWithLives(p>0.8)
DiscardProbablyUseless(p>0.6)
DiscardUnidentifiedCard
TellUnambiguous
RandomHintDiscard

TABLE VI: Evolved rules for 3 players

Rules
PlayRecentlyHinted(p>0.2,l>1)
PlayProbablySafeCard(p>0.8)
DiscardUselessCard
TellPlayableCard
DiscardRandomCard
TellAboutOnes
PlayProbablySafeCardWithLives(p>0.6)
DiscardProbablyUseless(p>0.8)
PlayProbablySafeCardWithLives(p>0.8)
PlayProbablySafeCardWithLives(p>0.4)
CompleteTellPlayableCard
TellUselessCard
PlayRecentlyHinted(p>0.01,l>1)
DiscardHighestKnown
PlayIfCertain
PlayRecentlyHinted(p>0.01,l>0)
TellMostInformation
PlayProbablySafeCard(p>0.6)
TellUnknownCard
RandomHintDiscard

TABLE VII: Evolved rules for 4 players

Rules
PlayProbablySafeCardWithLives(p>0.8)
PlayRecentlyHinted(p>0.2,l>1)
PlayIfCertain
PlayMostProbableIfDeckEmpty
PlayProbablySafeCard(p>0.8)
CompleteTellPlayableCard
TellPlayableCard
DiscardUnidentifiedCard
PlayRecentlyHinted(p>0.01,l>0)
TellAboutFives
TellMostInformation
RandomHintDiscard

TABLE VIII: Evolved rules for 5 players