Final Project Report: The Hanabi Game

Xiuhan Wang, Jing Wei

January 10, 2021

1 Introduction

The Hanabi game is a benchmark challenge in fully cooperative games of imperfect information. It's very different from adversarial two-player zero-sum games which have been studied more. But non-zero-sum scenarios are more important in real life.

In this game of limited communication, players convey information by choosing actions from a few options. Therefore to understand others' messages, the player must reason about beliefs and intentions of other players. Such theory of mind reasoning is crucial in more general collaborative efforts, especially those with human partners. (Bard et al., 2020)

2 Rules of the Hanabi Game

Hanabi is a game for two to five players. Each player holds four cards in his hand (or five if less than four players). Everyone can see all others' cards but not that of himself. Each card has a rank (1–5) and a color (red, green, blue, yellow or white). There are three 1s, two 2s, 3s, 4s and a 5 of each color; hence 50 cards in total. The players are fully cooperative—they want to play cards to build five stacks, one for each color, going consecutively from 1. The goal is to maximize the score, which is the total number of cards in the stack.

Players take turns to play. In one's turn, he must take exactly one of the following actions:

- **Hint.** To give a hint to others, the player chooses a color or a rank, then point out all cards matching that color or rank in another player's hand. Only ranks and colors that are present in the other player's hand can be chosen. To limit the number of hints, the group has initially eight information tokens and a token is consumed when giving a hint.
- **Discard.** When there are less than eight information tokens, he can discard a card from his hand, draw a new card from the deck and recover an information token. The discarded card is visible to all players and the newly drawn card is visible only to all other players.
- **Play.** The player can choose a card from his hand and try to put it in the stacks. It's successful if the rank is exactly one larger than the top of the stack corresponding to the color. If it's successful, put the card on the top of that stack. Recover one information token if there are less than eight and the rank played is 5. If the play is unsuccessful, discard the card and the group loses one life. When three lives are lost, the game ends immediately. Whether successful or not, the player draws a new card from the deck.

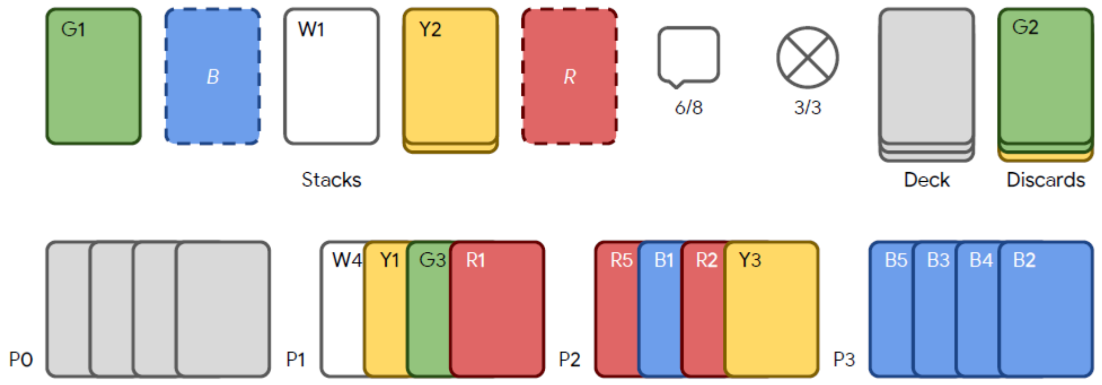

Figure 1: An example of a four player Hanabi game from the point of view of player 0 (cited from Bard et al. (2020)).

3 Previous Methods on Hanabi

The Hanabi game was proved NP-hard even if all hidden information is revealed. (Baffier et al., 2017) The hand-coded mathematical "hat guessing game" strategy (Cox et al., 2015) held state-of-the-art results from March 2016 until December 2019. In five-player games, it almost achieved the best performance resulted by a cheating strategy where each player sees his own cards and uses the information strategy. However, it performed badly in the two-player setting (Bouzy, 2017); and it is not generalizable to similar games. A Monte-Carlo tree search algorithm is given by Walton-Rivers et al. (2017); but the performance is still not very good.

Foerster et al. (2018) proposed a method of Bayesian action decoder (BAD). The key idea is a joint public belief over players' private features to resolve recursive belief over beliefs, and sampling a deterministic partial policy to resolve the dilemma between informative actions and stochastic actions for exploration. BAD worked well in the two-player setting. Hu & Foerster (2020) continued to optimize BAD to SAD and incorperated search into SPARTA (Lerer et al., 2020) to achieve a new state-of-the-art result.

Then they come up with the method called "other-play" (OP) (Hu et al., 2020) that work well on the zero-shot coordination scenario of the game. Previous studies on MARL usually focus on the self-play (SP) setting where agents are trained and tested both against with themselves, but agents naively trained by SP could do arbitrarily bad when paired with a partner they're not trained with. OP trains the agent with a copy of itself that is randomly permuted according to the symmetries in the game; hence it reduces the tendency towards arbitrary symmetry breaking. Agents trained with OP perform well when tested by cross-play, and also cooperate well with human.

Our work aims to be a better substitute for OP. Intuitively, the optimal strategies for symmetric games should be the same. As an analog, the convolutional neural network uses a symmetric network for training. We apply this idea to the network and reduce the network size by the symmetry group size.

The diagram below shows the relations of the previous methods.

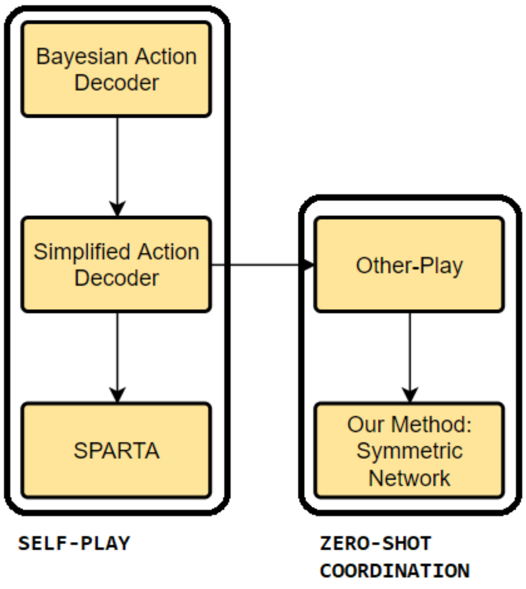

Figure 2: A diagram of relations of previous methods.

4 Zero-Shot Coordination

Zero-shot coordination is the scenario where players are paired with strangers they're not trained with, and must quickly coordinate to cooperate. Consider the following *lever coordination game*: two players, who don't know each other, will choose two levers independently. If they choose the same, they can get the reward of that lever. Otherwise, the reward is 0. In Figure 3 (a), obviously the only strategy is to pick a lever at random, with expected reward 0.1. However in (b), if they both pick a 1-lever, the expected reward is 0.11. But if they both pick the 0.9-lever, they get higher reward 0.9.

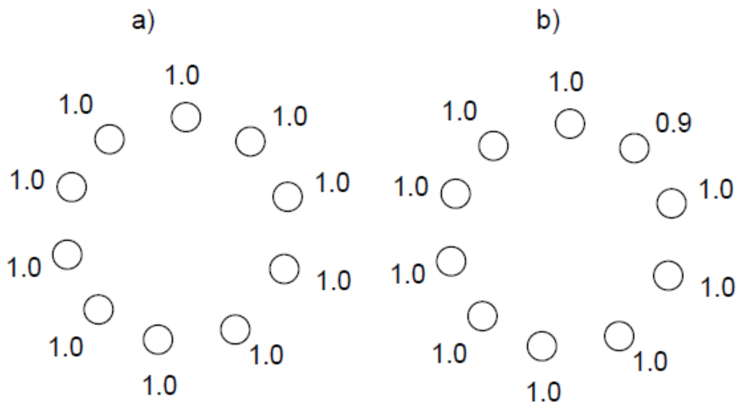

Figure 3: An example of a lever coordination game (cited from Hu et al. (2020)).

Agents trained by self-play would quickly reach a convention to pick the same lever. This gives the highest payoff 1 and is fine when playing with the same partner they're trained with. We see, in self-play, agents often use coordinated symmetry breaking to get higher payoffs, but this doesn't work and will be disastrous in the zero-shot coordination setting.

Similarly in Hanabi, the 5 colors are symmetric and thus indistinguishable. Self-play bots' strategies often involves conventions like "hinting yellow means discarding the first card". This would coordinate badly with another agent whose convention is "hinting blue means discarding the first card", which has the same chance of being adopted by self-play as the strategy.

5 Other Play

Arbitrary symmetries breaking is harmful in zero-shot coordination. Hu et al. (2020) proposed a method called *other-play* (OP) to alleviate this problem.

The symmetries in the MDP can be described as a isomorphism over states $\mathcal{S}$, observations $\mathcal{A}$ and actions $\mathcal{O}$ that keeps the transition function P, the reward R and the observation function O invariant.

$$\begin{aligned}\phi = (\phi_{\mathcal{S}}, \phi_{\mathcal{O}}, \phi_{\mathcal{A}}) \in \Phi \Leftrightarrow P(\phi(s')|\phi(s), \phi(a)) &= P(s'|s, a) \\ \wedge\ R(\phi(s), \phi(a), \phi(s')) &= R(s, a, s') \\ \wedge\ O_i(\phi(o)|\phi(s), \phi(a)) &= O_i(o|s, a), \quad \forall s, a, s'\end{aligned}$$

Clearly, Φ is a permutation group that acts on $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{O}$.

In the lever game, the 9 levers of 1 are indistinguishable; hence there is S_9 symmetry. In Hanabi, the 5 colors and the positions of cards in the players' hands are symmetric; the corresponding permutation group is $S_{\text{num-colors}} \times (S_{\text{hand-size}})^{\text{num-players}}$.

Let π^1, π^2 be the policies of the two players and let $J(\pi^1, \pi^2)$ be the reward of their match. The self-play learning is optimizing the joint policy

$$\pi^* = \arg\max_{\pi} J(\pi^1, \pi^2).$$

Other-play modifies the training of self-play, such that for each run, uniformly iid choose a random permutation $\phi_i \in \Phi$ for each agent i, and let agent i play the permuted game $\phi_i(\mathcal{S}, \mathcal{O}, \mathcal{A})$. In other words, agents observes and act in different permutations of the same environment.

Therefore, other-play training is optimizing the target

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\phi\sim\Phi} J(\pi^1, \phi(\pi^2)).$$

Because all $\phi(\pi)$ are not differentiated by the MDP, each of these policies must have the same chance being adopted by the partner. Hence other-play is the best meta-quilibrium learning rule, that is, neither agent can improve the payoff by adopted another learning rule instead.

OP's implementation is like a kind of domain randomization, it aims to make the policy invariant to how the partner breaks the symmetries.

6 Symmetric Networks

6.1 Theoretical Method

What OP result is essentially a model that doesn't break the symmetry arbitrarily. We can achieve this goal more straightforwardly by designing the networks to be symmetric according to this permutation group.

Formally speaking, we restrict our policy to statify

$$\phi(\pi) = \pi \ \forall \phi \in \Phi.$$

Because in OP's learning target, every policy behaves like the uniform mixture of its permutations, this restriction doesn't harm the effectiveness.

In order to satisfy this constraint, we make our neural networks symmetric. In other words, the permutation group Φ of MDP can also act on neurons and connections in the networks.

- For every neuron h in the network, there should also be neurons $\phi(h)$ for all $\phi \in S_5$.
- h and $\phi(h)$ should have the same bias. If h takes its input from a neuron g with weight α, then $\phi(h)$ takes input from $\phi(g)$ with the same weight.

Therefore, we see neurons can be seen as devided into equivalence classes $\Phi(h) = \{\phi(h) \mid \phi \in \Phi\}$, where they share a same set of weights and bias among the same class. The sizes of these equivalent classes are at most $|\Phi|$.

6.2 Implementation

We construct a network in which neurons are labeled by $h_{i,\tau}$ and let $\phi(h_{i,\tau}) = h_{i,\phi(\tau)}$. Here, i is the index of the equivalent class, and the fingerprint τ marks the neuron's index in the equivalent class and is drawn from a set T_Φ that Φ can act on.

For the common symmetry group S_n (like S_9 in the lever game), let T_n^k be the set of k-length permutations of 1 to n, and let Φ naturally act on this set. For the permutation group that is a Cartesian product $\Phi = \Phi_1 \times \Phi_2$ (like that of Hanabi), let T also be the Cartesian product $T_\Phi = T_{\Phi_1} \times T_{\Phi_2}$.

Neurons in T_n^k equivalence class stands for the features that distinguish some k labels out of the n symmetric labels in the game. For example, "the probability of my first card being red" can be represented by a neuron in $T_5^1 \times T_4^1$ class, "the advantage of discarding a rank-4 green card over a rank-2 yellow card" represented in T_5^2, and "the expected number of information tokens in the next step" represented in $T_5^0 \times T_4^0 = \{1\}$ (a singleton neuron).

In each layer, place some instances of these various sizes of classes together. For example, SAD has 512 LSTM cells in a hidden layer, we can have about 256 T_5^0 classes of cells, 32 T_5^1 classes and 8 T_5^2 classes to get the similar performance. When n is not small, large k may make the size of equivalence class very large, but that is not needed. Most considered features are in small k.

Then let's consider the weight of the connection from a neuron $h_{i,\tau}$ to a neuron $h_{j,\sigma}$ in the next layer, where $\tau = (t_1, t_2, \ldots, t_k) \in T_n^k,\ \sigma = (s_1, s_2, \ldots, s_m) \in T_n^m$.

Let $\text{mask}(\tau, \sigma)$ be a k-length sequence whose i-th position is

$$\text{mask}(\tau, \sigma)_i = \begin{cases} j, & \text{if } t_i = s_j \\ 0, & \text{if } t_i \text{ doesn't appear in } \sigma \end{cases}.$$

Then there should be weights $w_{\text{mask}(\tau,\sigma)}$, and

$$h_{j,\sigma} = f\Big(\sum_{i \in \text{prev-layer}} \sum_{\tau \in T_i} w_{\text{mask}(\tau,\sigma)} h_{i,\tau} \ + \ b_j \Big)$$

where f is the non-linearity.

Note that this way of constructing a network is compatible with various forms of neurons including LSTM cells.

6.3 Related Works

Convolutional neural networks appears to be similar to our work in the nature of making the structure of the network resemble the symmetry. The convolution reflects the transition invariance of the input. What we proprose is a more general method of studying the underlying symmetries in the problem and designing a network according to it.

By designing the translation-invariant structure of the network and reuse parameters, CNN dramastically reduces the number of parameters, and is hence easy to train, and converges both faster and to more reasonable results. Our method also reduces the number of parameters, hence will converge faster and to a certainly symmetry-invariant policy.

7 Experiments

7.1 Lever game

(Optional)

7.2 Hanabi

First we trained an OP agent according to Hu et al. (2020) to reproduce their results. Following is our training result.

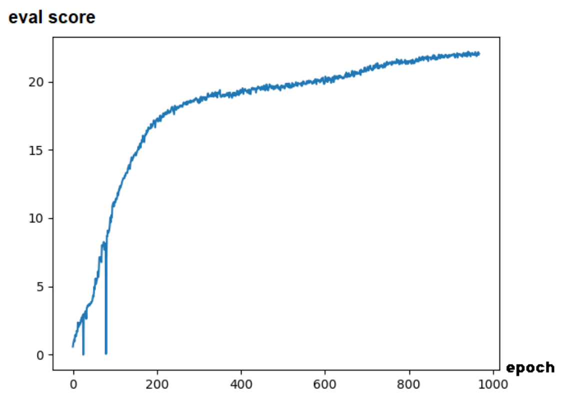

Figure 4: The self-play score during training an OP agent.

Hu et al. (2020)'s agents reached more than 24 scores in self-play. Due to our limited access to computation resources, our agent is still far from convergance, even though we spent 3 whole days of training. However to see a significant result, we have to train multiple agents using different seeds. That requires a lot of time, and money to rent GPUs.

Our code is available at `https://github.com/Aeterrno-Amora/hanabi_SAD` and `https://github.com/Aeterrno-Amora/hanabi-learning-environment`. See r2d2.py and sym_utils.py in the `hanabi_SAD` repository for the implementation of our symmetric networks. We implemented symmetric linear layers and symmetric LSTM layers according to symmetry groups as described in the *Implementation* section, and combined them into a SAD model.

But there was lot of codes to modify to make our model run with the given framework. The input of observations, the output of q-values and several other intermediate data have to be rearranged in a color-major order, so that the color symmetries in them can be correctly recognized by the model. To do this we have to read 50KB of code and rewrite them to nearly double amount. There is still some more to do to study how to optimize our code with pytorch JIT scripts, or it won't run with the given batch runner.

We have already eliminated a lot of bugs. But now we're suffering from a bug of accidentally accessing memory beyond allocated range. It's supposed to be an out-of-range vector operation. But it's very hard to figer out where it happens.

8 Conclusion

The Hanabi game is quite a challenge. Inspired from convolutional neural networks, we have come up with the symmetric networks. We remark that this method can be applied to most of the cases where OP is applicable. It is also fundamentally compatible with any type of optimization method. The idea of designing a network according to symmetries can even be apply to wider topics to produce interesting models for various problem with symmetry.

Unfortunately, due to the hardness of debugging and the limitations of computation resources, we are unable to finish the experiments. We believe our approach will work well. We are sorry about the incompleteness of our work.

There are still some further directions to research on. Both OP and our approach require the symmetric group to be given. To generalize to an arbitrary game in which symmetries are hard to tell by a human, we can consider how to combine our method or OP with MDP homomorphism methods which can "learn" the underlying symmetries in MDP.

References

Baffier, J.-F., Chiu, M.-K., Diez, Y., Korman, M., Mitsou, V., van Renssen, A., … Uno, Y. (2017). Hanabi is np-hard, even for cheaters who look at their cards . *Theoretical Computer Science*, *675*, 43–55.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., … Bowling, M. (2020). The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, *280*, 103216.

Bouzy, B. (2017). Playing hanabi near-optimally. In *Advances in computer games* (pp. 51–62).

Cox, C., Silva, J. D., Deorsey, P., Kenter, F. H. J., Retter, T., & Tobin, J. (2015). How to make the perfect fireworks display: Two strategies for hanabi. *Mathematics Magazine*, *88*(5), 323–336.

Foerster, J. N., Song, H. F., Hughes, E., Burch, N., Dunning, I., Whiteson, S., … Bowling, M. (2018). Bayesian action decoder for deep multi-agent reinforcement learning. In *International conference on machine learning* (pp. 1942–1951).

Hu, H., & Foerster, J. N. (2020). Simplified action decoder for deep multi-agent reinforcement learning. In *Iclr 2020 : Eighth international conference on learning representations.*

Hu, H., Lerer, A., Peysakhovich, A., & Foerster, J. (2020). "other-play" for zero-shot coordination. *arXiv preprint arXiv:2003.02979.*

Lerer, A., Hu, H., Foerster, J. N., & Brown, N. (2020). Improving policies via search in cooperative partially observable games. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(5), 7187–7194.

Walton-Rivers, J., Williams, P. R., Bartle, R., Perez-Liebana, D., & Lucas, S. M. (2017). Evaluating and modelling hanabi-playing agents. In *2017 ieee congress on evolutionary computation (cec)* (pp. 1382–1389).