Improving Policies via Search in Cooperative Partially Observable Games

Adam Lerer
Facebook AI Research
alerer@fb.com

Hengyuan Hu
Facebook AI Research
hengyuan@fb.com

Jakob Foerster
Facebook AI Research
jnf@fb.com

Noam Brown
Facebook AI Research
noambrown@fb.com

Abstract

Recent superhuman results in games have largely been achieved in a variety of zero-sum settings, such as Go and Poker, in which agents need to compete against others. However, just like humans, real-world AI systems have to coordinate and communicate with other agents in cooperative partially observable environments as well. These settings commonly require participants to both interpret the actions of others and to act in a way that is informative when being interpreted. Those abilities are typically summarized as *theory of mind* and are seen as crucial for social interactions. In this paper we propose two different search techniques that can be applied to improve an arbitrary agreed-upon policy in a cooperative partially observable game. The first one, *single-agent search*, effectively converts the problem into a single agent setting by making all but one of the agents play according to the agreed-upon policy. In contrast, in *multi-agent search* all agents carry out the same common-knowledge search procedure whenever doing so is computationally feasible, and fall back to playing according to the agreed-upon policy otherwise. We prove that these search procedures are theoretically guaranteed to at least maintain the original performance of the agreed-upon policy (up to a bounded approximation error). In the benchmark challenge problem of Hanabi, our search technique greatly improves the performance of every agent we tested and when applied to a policy trained using RL achieves a new state-of-the-art score of 24.61 / 25 in the game, compared to a previous-best of 24.08 / 25.

Introduction

Real-world situations such as driving require humans to coordinate with others in a partially-observable environment with limited communication. In such environments, humans have a mental model of how other agents will behave in different situations (theory of mind). This model allows them to change their beliefs about the world based on why they think an agent acted as they did, as well as predict how their own actions will affect others' future behavior. Together, these capabilities allow humans to search for a good action to take while accounting for the behavior of others.

Despite the importance of these cooperative settings to real-world applications, most recent progress on AI in large-scale games has been restricted to zero-sum settings where agents compete against each other, typically rendering communication useless. Search has been a key component in reaching professional-level performance in zero-sum games, including backgammon (Tesauro 1994), chess (Campbell, Hoane Jr, and Hsu 2002), Go (Silver et al. 2016; Silver et al. 2017; Silver et al. 2018), and poker (Moravčík et al. 2017; Brown and Sandholm 2017; Brown and Sandholm 2019).

Inspired by the success of search techniques in these zero-sum settings, in this paper we propose methods for agents to conduct search given an agreed-upon 'convention' policy (which we call the *blueprint policy*) in cooperative partially observable games. We refer to these techniques collectively as *Search for Partially Observing Teams of Agents (SPARTA)*. In the first method, a single agent performs search assuming all other agents play according to the blueprint policy. This allows the search agent to treat the known policy of other agents as part of the environment and maintain beliefs about the hidden information based on others' actions.

In the second method, multiple agents can perform search simultaneously but must simulate the search procedure of other agents in order to understand why they took the actions they did. We propose a modification to the multi-agent search procedure - *retrospective belief updates* - that allows agents to fall back to the blueprint policy when it is too expensive to compute their beliefs, which can drastically reduce the amount of computation while allowing for multi-agent search in most situations.

Going from just the blueprint policy, to single-agent search, to multi-agent search each empirically improves performance at the cost of increased computation. Additionally, we prove that SPARTA cannot result in a lower expected value than the blueprint policy except for an error term that decays in the number of Monte Carlo (MC) rollouts.

We test these techniques in Hanabi, which has been proposed as a new benchmark challenge problem for AI research (Bard et al. 2019). Hanabi is a popular fully cooperative, partially observable card game with limited communication. Most prior agents for Hanabi have been developed using handcrafted algorithms or deep reinforcement learning (RL). However, the Hanabi challenge paper itself remarks that "humans approach Hanabi differently than current learning algorithms" because while RL algorithms perform exploration to find a good joint policy (convention), humans instead typically start with a convention and then individually search for the best action assuming that their partners will play the convention (Bard et al. 2019).

Applying SPARTA to an RL blueprint (that we train in self-play) establishes a new state-of-the-art score of 24.61 / 25 on 2-player Hanabi, compared to a previous-best of 24.08 / 25. Our search methods also achieve state-of-the-art scores in 3, 4, and 5-player Hanabi (Table 1). To our knowledge, this is the first application of theoretically sound search in a large partially observable cooperative game.

We provide code for single- and multi-agent search in Hanabi as well as a link to supplementary material at `https://github.com/facebookresearch/Hanabi_SPARTA`

Related Work

Search has been necessary to achieve superhuman performance in almost every benchmark game. For fully observable games, examples include two-ply search in backgammon (Tesauro 1994), alpha-beta pruning in chess (Campbell, Hoane Jr, and Hsu 2002), and Monte Carlo tree search in Go (Silver et al. 2016; Silver et al. 2017; Silver et al. 2018). The most prominent partially observable benchmark game is poker, where search based on beliefs was key to achieving professional-level performance (Moravčík et al. 2017; Brown and Sandholm 2017; Brown and Sandholm 2019). Our search technique most closely resembles the one used in the superhuman multi-player poker bot *Pluribus*, which conducts search given each agents' presumed beliefs and conducts MC rollouts beyond the depth limit of the search space assuming all agents play one of a small number of blueprint policies.

Cooperative multi-agent settings have been studied extensively under the DEC-POMDP formalism (Oliehoek, Spaan, and Vlassis 2008). Finding optimal policies in DEC-POMDPs is known to be NEXP-hard (Bernstein et al. 2002), so much prior work studies environments with structure such as factorized local interactions (Oliehoek et al. 2008), hierarchical policies (Amato et al. 2015), or settings with explicit costs of communication (Goldman and Zilberstein 2003).

There has been a good deal of prior work developing agents in Hanabi. Notable examples of hand-crafted bots that incorporate human conventions include SmartBot (O'Dwyer 2019) and Fireflower (Wu 2018a). Alternatively, so called 'hat-coding' strategies (Wu 2018b), which are particularly successful for N $>$ 2 players, instead use information theory by communicating instructions to all players via hints using modulo-coding, as is commonly employed to solve 'hat-puzzles'. More recent work has focused on tackling Hanabi as a learning problem (Bard et al. 2019; Foerster et al. 2019). In this domain, the Bayesian Action Decoder learning method uses a public belief over private features and explores in the space of deterministic partial policies using deep RL (Foerster et al. 2019), which can be regarded as a scalable instantiation of the general ideas presented in (Nayyar, Mahajan, and Teneketzis 2013). The Simplified Action Decoder algorithm established the most recent state of the art in Hanabi (Hu and Foerster 2020).

Lastly, there is recent work on ad-hoc team play in Hanabi, in which agents get evaluated against a pool of different teammates (Canaan et al. 2019; Walton-Rivers et al. 2017).

Background

We consider a Dec-POMDP with N agents. The Markov state of the environment is $s \in \mathcal{S}$ and we use i to denote the agent index. At each time step, t, each agent obtains an observation $o^i = Z(s, i)$, where Z is a deterministic observation function, and takes an action $a^i \in \mathcal{A}$, upon which the environment carries out a state transition based on the transition function, $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, \mathbf{a})$, and agents receive a team-reward of $r_t = \mathcal{R}(s_t, \mathbf{a})$, where $\mathbf{a}$ indicates the joint action of all agents. We use $\tau_t = \{s_0, \mathbf{a}_0, r_0, ...s_t\}$ to denote the game history (or 'trajectory') at time t. Each agent's policy, π^i, conditions on the agent's *action-observation history* (AOH), $\tau^i_t = \{o^i_0, a^i_0, r_0, ...o^i_t\}$, and the goal of the agents is to maximise the total expected return, $J_\pi = \mathbb{E}_{\tau \sim P(\tau|\pi)} R_0(\tau)$, where $R_0(\tau)$ is the forward looking return of the trajectory, $R_t(\tau) = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$, and γ is an (optional) discount factor. In this paper we consider both deterministic policies, $a^i = \pi(\tau^i_t)$, typically based on heuristics or search, and stochastic policies, $a^i \sim \pi^\theta(a^i|\tau^i_t)$, where θ are the parameters of a function approximator, *e.g.*, a deep neural network.

In order to represent variable sequence lengths trajectories, τ^i_t, Deep RL in partially observable settings typically uses recurrent networks (RNNs) (Hausknecht and Stone 2015). These RNNs can learn implicit representations of the sufficient statistics over the Markov state given τ^i_t. In contrast, in our work we will use explicit *beliefs* to represent the probability distribution over possible trajectories[1]. The private belief of agent i that they are in trajectory τ_t at time step t is $B^i(\tau_t) = P(\tau_t|\tau^i_t)$. We also define $B(\tau_t) = P(\tau_t|\tau^{\mathcal{G}}_t)$ and $B(\tau^i_t) = P(\tau^i_t|\tau^{\mathcal{G}}_t) = \sum_{\tau_t \in \tau^i_t} B(\tau_t)$, which is the probability that agent i is in AOH τ^i_t conditional only on the *common knowledge* (CK) of the entire group of agents $\tau^{\mathcal{G}}_t$. Here CK are things that all agents know that all agents know ad infinitum. Please see (Osborne and Rubinstein 1994) for a formal definition of CK and (Foerster et al. 2018; Foerster et al. 2019) for examples of how CK can arise and be used in multi-agent learning. Practically it can be computationally challenging to exactly compute common-knowledge beliefs due a large number of possible states and trajectories. However, in many settings, *e.g.*, poker (Brown and Sandholm 2019), the CK-belief can be factorized across public features and a set of private features associated with the different agents. In these settings the CK trajectory is simply the history of public features.

[1]Maintaining exact beliefs in large POMDPs is typically considered intractable. However we find that even in an environment with large state spaces such as card games, the number of states that have non-zero probability conditional on a player's observations is often much smaller; in the case of Hanabi, this set of possible states can be stored and updated explicitly.

In our methods we will commonly need to carry out computations over each of the possible trajectories that have non-zero probability given an agent's AOH or all agents' common knowledge. We refer to these as the *trajectory range* $\beta_t^i = \{\tau_t | B^i(\tau_t) > 0\}$ and $\beta_t = \{\tau_t | B(\tau_t) > 0\}$. We refer to the vector of probabilities in the trajectory range on timestep t by $\mathbf{B}_t^i$ and $\mathbf{B_t}$. We also commonly need to carry out computations over each of an agent's AOHs that have non-zero probability given the common knowledge of the entire group of agents, which we refer to as the *AOH range* $\chi_t^i = \{\tau_t^i | B(\tau_t^i) > 0\}$. We refer to the entire vector of probabilities in the AOH range of agent i on timestep t by $\mathbf{C}_t^i$.

We further define the typical conditional expectations ('value functions'),

$$V_\pi(\tau_t) = \mathbb{E}_{\tau'_T \sim P(\tau'_T | \pi, \tau_t)} R_t(\tau'_T) \tag{1}$$

$$Q_\pi(\tau_t, a^i) = \mathbb{E}_{\tau'_T \sim P(\tau'_T | \pi, \tau_t, a^i)} R_t(\tau'_T) \tag{2}$$

Given the range defined above we also introduce expectations conditioned on the AOHs, τ_t^i:

$$V_\pi(\tau_t^i) = \sum_{\tau_t \in \beta_t^i} B^i(\tau_t) V_\pi(\tau_t) \tag{3}$$

$$Q_\pi(\tau_t^i, a^i) = \sum_{\tau_t \in \beta_t^i} B^i(\tau_t) Q_\pi(\tau_t, a^i) \tag{4}$$

Even though the optimal policies in the fully cooperative setting are deterministic, we consider stochastic policies for the purpose of reinforcement learning.

We further assume that a deterministic *blueprint* policy π_b, defining what each player should do for all possible trajectories, is common knowledge amongst all players. Player i's portion of the blueprint policy is denoted π_b^i and the portion of all players other than i as π_b^{-i}. In some settings, when actually playing, players may choose to not play according to the blueprint and instead choose a different action determined via online search. In that case π_b differs from π, which denotes the policy that is actually played.

In our setting all of the past actions taken by all agents and the observation functions of all players are common knowledge to all agents. As such, if an agent is known to be playing according to a given policy, each action taken by this agent introduces a belief update across all other agents and the public belief. Suppose agent i has a current belief B_{t-1}^i and next observes (a_t^j, o_t^i), where we have broken up the observation to separate out the observed partner action. Then,

$$B^i(\tau_t) = P(\tau_t | \tau_t^i) = P(\tau_t | \tau_{t-1}^i, o_t^i, a_t^j) \tag{5}$$

$$= \frac{B^i(\tau_{t-1}) \pi^j(a_t^j | \tau_{t-1}) P(o_t^i | \tau_{t-1}, a_t^j)}{\sum\limits_{\tau'_{t-1}} B^i(\tau'_{t-1}) \pi^j(a_t^j | \tau'_{t-1}) P(o_t^i | \tau'_{t-1}, a_t^j)} \tag{6}$$

In other words, the belief update given (a_t^j, o_t^i) consists of two updates: one based on the partner's known policy π^j, and the other based on the dynamics of the environment. The common knowledge belief B is updated in the same way, using the common-knowledge observation $\tau^{\mathcal{G}}$ rather than τ^i.

Method

In this section we describe SPARTA, our online search algorithm for cooperative partially-observable games.

Single-Agent Search

We first consider the case in which only one agent conducts online search. We denote the searching agent as agent i. Every other agent simply plays according to the blueprint policy (and we assume agent i knows all other agents play according to the blueprint).

Since agent i is the only agent determining her policy online while all other agents play a fixed common-knowledge policy, this is effectively a single-agent POMDP for agent i. Specifically, agent i maintains a belief distribution $\mathbf{B}_t^i$ over trajectories she might be in based on her AOH τ_t^i and the known blueprint policy of the other agents. Each time she receives an observation or another agent acts, agent i updates her belief distribution according to (6) (see Figure 1, left). Each time i must act, she estimates via Monte Carlo rollouts the expected value $Q_{\pi_b}(\tau_t^i, a^i)$ of each action assuming *all* agents (including agent i) play according to the joint blueprint policy π_b for the remainder of the game following the action (Figure 1, right). A precise description of the single-agent search algorithm is provided in the appendix.

As described, agent i calculates the expected value of only the next action. This is referred to as 1-ply search. One could achieve even better performance by searching further ahead, or by having the agent choose between multiple blueprint policies for the remainder of the game. However, the computational cost of the search would also increase, especially in a game like Hanabi that has a large branching factor due to chance. In this paper all the experiments use 1-ply search.

Since this search procedure uses exact knowledge of all other agents' policies, it cannot be conducted correctly by multiple agents independently. That is, if agent j conducts search on a turn after agent i conducted search on a previous turn, then agent j's beliefs are incorrect because they assume agent i played π_b^i while agent i actually played the modified policy π^i. If more than one agent independently performs search assuming that others follow the blueprint, policy improvement cannot be guaranteed, and empirical performance is poor (Table 4, Appendix).

Multi-Agent Search

In order for an agent to conduct search effectively, her belief distribution must be accurate. In the case of single-agent search, this was achieved by all agents agreeing beforehand on a blueprint policy, and then also agreeing that only one agent would ever conduct search and deviate from the blueprint. In this section we instead assume that all agents agree beforehand on both a blueprint policy and on what search procedure will be used. When agent i acts and conducts search, the other agents exactly replicate the search procedure conducted by agent i (including the random seed) and compute agent i's resulting policy accordingly. In this way, the policy played so far is always common knowledge.

Since the other agents do not know agent i's private observations, we have all agents (including agent i) conduct search and compute agent i's policy for *every possible* AOH that agent i might be in based on the common-knowledge observations. Specifically, all agents conduct search for every AOH $\tau_t^{\prime i} \in \chi_t^i$. When conducting search for a particular $\tau_t^{\prime i}$ as part of this loop, the agents also compute what $\mathbf{B}_i^t$ would be assuming agent i's AOH is $\tau_t^{\prime i}$ and compute $Q_{\pi_b}(\tau_t^{\prime i}, a^i)$ for every action a^i based on this $\mathbf{B}_i^t$. We refer to this loop of search over all $\tau_t^{\prime i} \in \chi_t^i$ as *range-search*.

Agent i must also compute her policy via search for every $\tau_t^{\prime i} \in \chi_t^i$ (that is, conduct range-search) even though she knows τ_t^i, because the search procedure of other agents on future timesteps may be based on agent i's policy for $\tau_t^{\prime i} \neq \tau_t^i$ where $\tau_t^{\prime i} \in \chi_t^i$ and it is necessary for all agents to be consistent on what that policy is to ensure that future search procedures are replicated identically by all agents.

In a game like two-player Hanabi, $|\chi_t^i|$ could be nearly 10 million, which means the range-search operation of multi-agent search could be 10 million times more expensive than single-agent search in some situations and therefore infeasible. Fortunately, the actual number of positive-probability AOHs will usually not be this large. We therefore have all agents agree beforehand on a budget for range-search, which we refer to as a *max range* (abbreviated *MR*). If $|\chi_t^i| > MR$ on timestep t where agent i is the acting agent, then agent i does not conduct search and instead simply plays according to the blueprint policy π_b. Since χ_t^i and the max range are common knowledge, it is also common knowledge when an agent does not search on a timestep and instead plays according to the blueprint.

Using a max range makes multi-agent search feasible on certain timesteps, but the fraction of timesteps in which search can be conducted may be less than single-agent search (which in a balanced two-player game is 50%). The next section describes a way to use a max range while guaranteeing that there's always at least one agent who can perform search.

Retrospective Belief Updates

As discussed in the previous section, χ_t^i on some timesteps may be too large to conduct search on. Using a max range mitigates this problem, but may result in search only rarely being conducted. Fortunately, in many domains more common knowledge information is revealed as the game progresses, which reduces the number of positive-probability AOHs on previous timesteps.

For example, at the start of a two-player game of Hanabi in which agent i acts first (and then agent j), $|\chi_0^i|$ might be nearly 10 million. If agent i were to use search to choose an action at this point, it would be too expensive for the agents to run range-search given the magnitude of $|\chi_0^i|$, so the other agents would not be able to conduct search on future timesteps.

However, suppose the action chosen from agent i's search results in agent i giving a hint to agent j. Given this new common-knowledge information, it might now be known that only 100,000 of the 10 million seemingly possible AOHs at the first timestep were actually possible. It may now be feasible for the agents to run range-search on this reduced set of 100,000 possible AOHs. In this way, agent i is able to conduct search on a timestep t where the max range is exceeded, and the agents can execute range-search at some later timestep t' once further observations have reduced the size of χ_t^i below the max range.

We now introduce additional notation to generalize this idea. Agent i's belief at timestep t' that the trajectory at some earlier timestep t was (or is) τ_t is $B_{t'}^i(\tau_t) = P(\tau_t|\tau_{t'}^i)$. The public belief at timestep t', which conditions only on the common knowledge of the entire group of agents at timestep t', that the trajectory at timestep t was (or is) τ_t is $B_{t'}(\tau_t) = P(\tau_t|\tau_{t'}^{\mathcal{G}})$ and $B_{t'}(\tau_t^i) = P(\tau_t^i|\tau_{t'}^{\mathcal{G}}) = \sum_{\tau_t \in \tau_t^i} B_{t'}(\tau_t)$. The trajectory range at timestep t' of the trajectories at timestep t is $\beta_{t,t'} = \{\tau_t | B_{t'}(\tau_t) > 0\}$ and $\beta_{t,t'}^i = \{\tau_t | B_{t'}^i(\tau_t) > 0\}$. The AOH range at timestep t' of the agent i AOHs at timestep t is $\chi_{t,t'}^i = \{\tau_t^i | B_{t'}(\tau_t^i) > 0\}$.

Again, the key idea behind retropective updates is that an agent can delay running range-search on a timestep until that range shrinks based on subsequent observations. When it is an agent's turn to act, she conducts search if and only if she has run range-search for each previous timestep t where one of the *other* agents played search (because this means she knows her belief distribution). Otherwise, she plays according to the blueprint policy.

Specifically, all agents track the oldest timestep on which search was conducted but range-search was not conducted. This is denoted t^*. Assume the agent acting at t^* was agent i. If at any point $|\chi_{t^*,t}^i| \leq$ *MR* then all agents conduct range-search for timestep t^* and t^* is incremented up to the next timestep on which search was conducted but range-search was not conducted (but obviously not incremented past the current timestep t). If $|\chi_{t^*,t}^i| \leq$ *MR* for this new t^* then all agents again conduct range-search and the process repeats. Since $\chi_{t^*,t}^i$ depends only on common knowledge, all agents conduct range-search at the same time and therefore t^* is always consistent across all agents. When it is an agent's turn to act on timestep t, she conducts search if and only if she was the agent to act on timestep t^* or if $t = t^*$. An important upshot of this method is that it's always possible for at least one agent to use search when acting.

If *MR* is set to zero then multi-agent search with retrospective updates is identical to single-agent search, because the first agent to act in the game will conduct search and she will continue to be the only agent able to conduct search on future timesteps. As *MR* is increased, agents are able to conduct search on an increasing fraction of timesteps. In two-player Hanabi, setting *MR* $=$ 10,000 makes it possible to conduct search on 86.0% of timesteps even though in the worst case $|\chi_t^i| \approx$ 10,000,000.

Soundness and Convergence Bound for Search

We now prove a theorem that applies to all three SPARTA variants described in this section. Loosely, it states that applying search on top of a blueprint policy cannot reduce the expected reward relative to the blueprint, except for an error term that decays as $O(1/\sqrt{N})$, where N is the number of Monte Carlo rollouts.

Theorem 1. *Consider a Dec-POMDP with N agents, actions $\mathcal{A}$, reward bounded by $r_{min} \leq R(\tau) < r_{max}$ where*

$r_{max} - r_{min} \leq \Delta$, game length bounded by T, and a set of blueprint policies $\pi_b \equiv \{\pi_b^0, \ldots, \pi_b^N\}$. If a Monte Carlo search policy π_s is applied using N rollouts per step, then

$$V_{\pi_s} - V_{\pi_b} \geq -2T\Delta|\mathcal{A}|N^{-1/2} \tag{7}$$

The proof is provided in the Appendix.

Experimental Setup

We evaluate our methods in the partially observable, fully cooperative game Hanabi, which at a high level resembles a cooperative extension of solitaire. Hanabi has recently been proposed as a new frontier for AI research (Bard et al. 2019) with a unique focus on theory of mind and communication.

The main goal of the team of agents is to complete 5 stacks of cards, one for each color, in a legal sequence, starting with a **1** and finishing with a **5**. The defining twist in Hanabi is that while players can observe the cards held by their teammates, they cannot observe their own cards. As such, players need to exchange information with their teammates in order to decide which cards to play. Hanabi offers two different means for doing so. First, players can take costly hint actions that reveal part of the state to their teammates. Second, since all actions are observed by all players, each action (such as playing a card or discarding a card) can itself be used to convey information, in particular if players agree on a set of conventions before the game. For further details on the state and action space in Hanabi please see (Bard et al. 2019).

In this work we are focused on the *self-play* part of the challenge, in which the goal is to find a set of policies that achieve a high score when playing together as a team.

For the blueprint strategy used in the experiments, we experimented with open-sourced handcrafted bots WTFWThat (Wu 2018b) and SmartBot (O'Dwyer 2019). We also created two of our own blueprint strategies. One was generated from scratch using deep reinforcement learning, which we call RLBot. The other, which we call CloneBot, was generated by conducting imitation learning using deep neural networks on the policy produced by single-agent search on top of SmartBot. The details for the generation of both bots are given in the appendix.

All experiments except the imitation learning of CloneBot and the reinforcement learning of RLBot were conducted on CPU using machines with Intel® Xeon® E5-2698 CPUs containing 40 cores each. A game of Hanabi requires about 2 core-hours for single-agent search and 90 core-hours for retrospective multi-agent search using the SmartBot blueprint policy with a max range of 10,000. We parallelize the search procedure over multiple cores on a single machine.

Implementing SPARTA for Hanabi

The public and private observations in Hanabi are factorizable into public and private features.[2] Specifically, each player's hidden information consists of the cards held by other agents, so the beliefs for each player can be represented as a distribution over possible hands that player may be holding. The initial private beliefs can be constructed based on the card counts. In addition to updates based on the actions of other agents, updates based on observations amount to (a) adjusting the probabilities of each hand based on the modified card count as cards are revealed, and (b) setting the probability of hands inconsistent with hints to 0.

The public beliefs for 2-player Hanabi can be factored into independent probability distributions over each player's hand (conditional on the common knowledge observations). These public beliefs are identical to the private beliefs except that the card counts are not adjusted for the cards in the partner's hand. Given one player's hand, the private beliefs over the other player's hand (which is of course no longer independent of the other player's hand) can be computed from the public beliefs by adjusting the card counts for the privately-observed cards.[3]

Estimating Action Expected Values via UCB

In order to reduce the number of MC rollouts that must be performed during search, we use a UCB-like procedure that skips MC rollouts for actions that are presumed not to be the highest-value action with high confidence. After a minimum of 100 rollouts per action is performed, the reward sample mean and its standard deviation is computed for each action. If the expected value for an action is not within 2 standard deviations of the expected value of the best action, its future MC rollouts are skipped.

Furthermore, we use a configurable threshold for deviating from the blueprint action. If the expected value of the action chosen by search does not exceed the value of the blueprint action by more than this threshold, the agent plays the blueprint action. We use a threshold of 0.05 in our experiments.

The combination of UCB and the blueprint deviation threshold reduces the number of rollouts required per timestep by $10\times$, as shown in Figure 3 in the appendix.

Bootstrapping Search-Based Policies via Imitation Learning

Search can be thought of as a policy improvement operator, i.e. an algorithm that takes in a joint policy and outputs samples from an improved joint policy. These samples can be used as training data to learn a new policy via imitation learning. This improved policy approximates the effect of search on the blueprint policy while being cheaper to execute, and search can be run on this learned policy, in effect bootstrapping the search procedure. In principle, repeated application of search and learning could allow the effect of single-agent search to be applied on multiple agents, and could allow the benefits of search to extend beyond the depth limit of the search procedure. However, there is no guarantee that this process would eventually converge to an optimal policy, even in the case of perfect function approximation.

We refer to the policy learned in this manner as CloneBot. We provide details of the training procedure in the Appendix and evaluate its performance in Table 1.

[2]The class of games of this form has recently been formalized as 'Factorized Observation Games' in (Kovařík et al. 2019).

[3]This conversion from public to private beliefs is what we call ConditionOnAOH() in the algorithm listing in the Appendix.

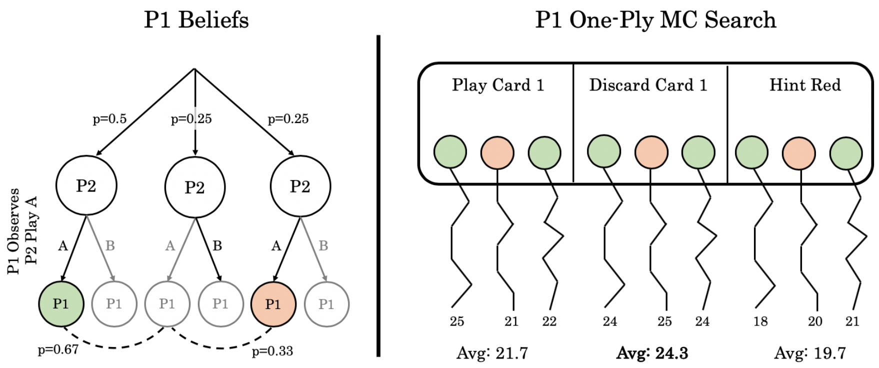

Figure 1: An illustration of the SPARTA Monte Carlo search procedure for an agent P1. **Left:** The MDP starts in one of three states and P2 plays A. P1 observes that P2 played A, ruling out the middle state in which P2 would have played B. This leaves P1 with a probability distribution (beliefs) over two possible states. **Right:** For each legal action ('Play Card 1', ...) P1 performs rollouts from states drawn from the belief distribution, and picks the action with the highest average score: Discard Card 2.

Results

Table 1 shows that adding SPARTA leads to a large improvement in performance for all blueprint policies tested in two-player Hanabi (the most challenging variant of Hanabi for computers) and achieves a new state-of-the-art score of 24.61 / 25 compared to the previous-best of 24.08 / 25.[4] Much of this improvement comes from adding single-agent search, though adding multi-agent search leads to an additional substantial improvement.

Unfortunately there are no reliable statistics on top human performance in Hanabi. Discussions with highly experienced human players has suggested that top players might achieve perfect scores in 2-player Hanabi somewhere in the range of 60% to 70% of the time when optimizing for perfect scores. Our strongest agent optimizes for expected value rather than perfect scores and still achieves perfect scores 75.5% of the time in 2-player Hanabi.

Table 2 shows the benefits of single-agent search also extend to the 3, 4, and 5-player variants of Hanabi as well. For these variants, the *SAD* agent (Hu and Foerster 2020) was state-of-the-art among learned policies, while an information-theoretic hat-coding policy (WTFWThat) achieves close to perfect scores for 4 and 5 players (Wu 2018b). Applying single-agent search to either SAD or WTFWThat improves the state-of-the-art scores for 3, 4, and 5-player variants (Table 2, Table 5 in Appendix). Applying multi-agent search in Hanabi becomes much more expensive as the number of players grows because the number of cards each player observes becomes larger.

Table 3 examines the performance and cost in number of rollouts for single-agent search and multi-agent search with different values of the max range (MR) parameter, using SmartBot as the blueprint agent. When MR is set to zero, we are performing single-agent search, so search is conducted on 50% of timesteps, which requires about 10^5 rollouts per game[5]. As the max range is increased, search is conducted more often. At a max range of 10,000, search is conducted on 86% of timesteps even though the maximum possible range for a timestep in two-player Hanabi is nearly 10 million. However, this still requires about 1,000× as many rollouts as single-agent search.

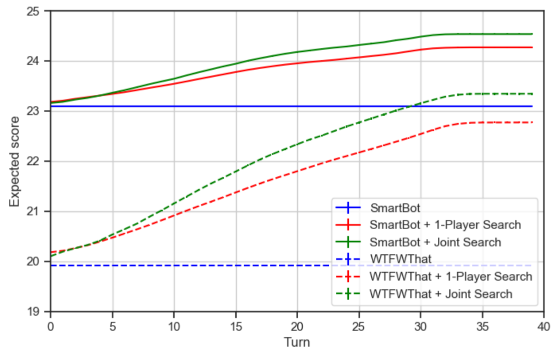

Figure 2: Average score predicted by MC search rollouts at different points in the game, for two different blueprints. Expected score by turn is averaged over replicates for each condition (error bars are included but are too small to be visible). If a game ends before 40 turns, the final score is propagated for all subsequent turns.

Figure 2 plots the average MC search prediction of the expected payoff of the best move ($Q_{\pi_b}(\tau^i, a^*)$) at different points in the game, for two blueprint policies. As predicted by the theory, the expected score starts at the blueprint expected score and increases monotonically as search is applied at each move of the game.

Conclusions

In this paper we described approaches to search in partially observable cooperative games that improves upon an arbitrary blueprint policy. Our algorithms ensure that any agent

[4]It is impossible to achieve a perfect score for some shuffles of the deck. However, the highest possible average score is unknown.

[5]SPARTA also performs counterfactual belief updates using the blueprint, using about 2×10^6 policy evaluations per game, which corresponds to about 4×10^4 games worth of policy evaluations. This cost is dominated by search rollouts for all SPARTA variants.

Blueprint Strategy	No Search	Single-Agent Search	Multi-Agent Search
RL (ACHA)	22.73 ± 0.12	-	-
(Bard et al. 2019)	15.1%	-	-
BAD	23.92 ± 0.01	-	-
(Foerster et al. 2019)	58.6%	-	-
SmartBot	22.99 ± 0.001	**24.21 ± 0.002**	**24.46 ± 0.01**
(O'Dwyer 2019)	29.6%	**59.1%**	**64.8%**
CloneBot	23.32 ± 0.001	**24.37 ± 0.02**	-
(ours)	40.6%	**64.6%**	-
DQN	23.45 ± 0.01	**24.30 ± 0.02**	**24.49 ± 0.02**
(ours)	46.8%	**63.5%**	**67.7%**
SAD	24.08 ± 0.01	**24.53 ± 0.01**	**24.61 ± 0.01**
(Hu and Foerster 2020)	56.1%	**71.1%**	**75.5%**

Table 1: Results for various policies in 2-player Hanabi coupled with single-agent and retrospective multi-agent search (SPARTA). The top row is the average score (25 is perfect), the bottom row is the perfect-score percentage. Single-agent and multi-agent search monotonically improve performance. For multi-agent search, a max range of 2,000 is used. Results in bold beat the prior state of the art of 24.08.

# Players	No Search	Single-Agent Search
2	24.08 ± 0.01	**24.53 ± 0.01**
	56.1%	**71.1%**
3	23.99 ± 0.01	**24.61 ± 0.01**
	50.4%	**74.5%**
4	23.81 ± 0.01	**24.49 ± 0.01**
	41.5%	**64.2%**
5	23.01 ± 0.01	**23.91 ± 0.01**
	13.9%	**32.9%**

Table 2: Hanabi performance for different numbers of players, applying single-agent search (SPARTA) to the 'SAD' blueprint agent which achieved state-of-the-art performance in 3, 4, and 5-player Hanabi among learned policies. Results shown in bold are state-of-the-art (for learned policies). Higher scores in 3+ player Hanabi can be achieved using hard-coded 'hat-counting' techniques; Table 5 in the Appendix shows that search improves those strategies too.

Max Range	% search	# Rollouts	Average Score
0	50.0%	1.5×10^5	24.21 ± 0.002
80	56.2%	3.0×10^6	24.30 ± 0.02
400	65.1%	1.3×10^7	24.40 ± 0.02
2000	77.2%	5.2×10^7	24.46 ± 0.01
10000	86.0%	1.8×10^8	24.48 ± 0.01

Table 3: Multi-agent retrospective search with a SmartBot blueprint for different values of max range. As max range increases, search is performed more often leading to a higher score, but the rollout cost increases dramatically.

conducting search always has an accurate belief distribution over the possible trajectories they may be in, and provide an alternative in case the search procedure is intractable at certain timesteps. We showed that in the benchmark domain of Hanabi, both search techniques lead to large improvements in performance to all the policies we tested on, and achieves a new state of the art score of 24.61 / 25 compared to a previous-best of 24.08 / 25. We also proved that applying our search procedure cannot hurt the expected reward relative to the blueprint policy, except for an error term that shrinks with more Monte Carlo rollouts. This result fits a theme, also shown in other games such as Chess, Go, and Poker, that search leads to dramatically improved performance compared to learning or heuristics alone.

The performance improvements from search come at a computational cost. Our search procedure involves tracking the belief probability of each AOH that a player may be in given the public information, which for Hanabi is no more than 10 million probabilities. Search also requires computing a large number of MC rollouts of the policy, especially for multi-agent search where the number of rollouts scales with the size of the belief space. Conducting search in partially observable games, whether cooperative or competitive, with many more AOHs per instance of public information remains an interesting challenge for future work and may be applicable to settings like Bridge and environments with visual inputs like driving.

This work currently assumes perfect knowledge of other agents' policies. This is a reasonable assumption in settings involving centralized planning but decentralized execution, such as self-driving cars that are created by a single company or robots working together as a team in a factory. In general however, an agent's model of others may not be perfect. Investigating how search performs in the presence of an imperfect model of the partner, and how to make search more robust to errors in that model, are important directions for future work as well.

Acknowledgments

We would like to thank Pratik Ringshia for developing user interfaces used to interact with Hanabi agents.

References

[Amato et al. 2015] Amato, C.; Konidaris, G.; Cruz, G.; Maynor, C. A.; How, J. P.; and Kaelbling, L. P. 2015. Planning for decentralized control of multiple robots under uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1241–1248.

[Bard et al. 2019] Bard, N.; Foerster, J. N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H. F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. 2019. The Hanabi challenge: A new frontier for AI research. *arXiv preprint arXiv:1902.00506*.

[Bernstein et al. 2002] Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of markov decision processes. *Mathematics of operations research* 27(4):819–840.

[Brown and Sandholm 2017] Brown, N., and Sandholm, T. 2017. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

[Brown and Sandholm 2019] Brown, N., and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science* eaay2400.

[Campbell, Hoane Jr, and Hsu 2002] Campbell, M.; Hoane Jr, A. J.; and Hsu, F.-h. 2002. Deep Blue. *Artificial intelligence* 134(1-2):57–83.

[Canaan et al. 2019] Canaan, R.; Togelius, J.; Nealen, A.; and Menzel, S. 2019. Diverse agents for ad-hoc cooperation in hanabi. *arXiv preprint arXiv:1907.03840*.

[Foerster et al. 2018] Foerster, J. N.; de Witt, C. A. S.; Farquhar, G.; Torr, P. H.; Boehmer, W.; and Whiteson, S. 2018. Multi-agent common knowledge reinforcement learning. *arXiv preprint arXiv:1810.11702*.

[Foerster et al. 2019] Foerster, J.; Song, F.; Hughes, E.; Burch, N.; Dunning, I.; Whiteson, S.; Botvinick, M.; and Bowling, M. 2019. Bayesian action decoder for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 1942–1951.

[Goldman and Zilberstein 2003] Goldman, C. V., and Zilberstein, S. 2003. Optimizing information exchange in cooperative multi-agent systems. In *Autonomous Agents and Multiagent Systems*, 137–144. ACM.

[Hausknecht and Stone 2015] Hausknecht, M., and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

[Horgan et al. 2018] Horgan, D.; Quan, J.; Budden, D.; Barth-Maron, G.; Hessel, M.; van Hasselt, H.; and Silver, D. 2018. Distributed prioritized experience replay. *CoRR* abs/1803.00933.

[Hu and Foerster 2020] Hu, H., and Foerster, J. 2020. Simplified action decoder for deep multi-agent reinforcement learning. In *Submitted to International Conference on Learning Representations*. under review.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kovařík et al. 2019] Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisỳ, V. 2019. Rethinking formal models of partially observable multiagent decision making. *arXiv preprint arXiv:1906.11110*.

[Moravčík et al. 2017] Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337):508–513.

[Nayyar, Mahajan, and Teneketzis 2013] Nayyar, A.; Mahajan, A.; and Teneketzis, D. 2013. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control* 58(7):1644–1658.

[O'Dwyer 2019] O'Dwyer, A. 2019. Hanabi. `https://github.com/Quuxplusone/Hanabi`.

[Oliehoek et al. 2008] Oliehoek, F. A.; Spaan, M. T.; Whiteson, S.; and Vlassis, N. 2008. Exploiting locality of interaction in factored dec-pomdps. In *Autonomous Agents and Multiagent Systems-Volume 1*, 517–524.

[Oliehoek, Spaan, and Vlassis 2008] Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research* 32:289–353.

[Osborne and Rubinstein 1994] Osborne, M. J., and Rubinstein, A. 1994. *A course in game theory*. MIT press.

[Rigollet 2015] Rigollet, P. 2015. High-dimensional statistics.

[Schaul et al. 2015] Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized Experience Replay. *arXiv e-prints* arXiv:1511.05952.

[Silver et al. 2016] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484.

[Silver et al. 2017] Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

[Silver et al. 2018] Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419):1140–1144.

[Sutton 1988] Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Mach. Learn.* 3(1):9–44.

[Tesauro 1994] Tesauro, G. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* 6(2):215–219.

[van Hasselt, Guez, and Silver 2015] van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep reinforcement learning with double q-learning. *CoRR* abs/1509.06461.

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

[Wang, de Freitas, and Lanctot 2015] Wang, Z.; de Freitas, N.; and Lanctot, M. 2015. Dueling network architectures for deep reinforcement learning. *CoRR* abs/1511.06581.

[Wu 2018a] Wu, D. 2018a. A rewrite of hanabi-bot in scala. `https://github.com/lightvector/fireflower`.

[Wu 2018b] Wu, J. 2018b. Hanabi simulation in rust. `https://github.com/WuTheFWasThat/hanabi.rs`.

Search Algorithm Listing

Algorithm 1 Single Agent Search

```
function SINGLEAGENTSEARCH($i$, $\pi_0 \dots \pi_N$)
    Initialize $B_0^i$, agent $i$'s private beliefs.
    for $t = 1, \dots, T$ do
        $o_t \leftarrow$ GETENVIRONMENTOBSERVATION()
        $B_t^i \leftarrow$ STEPBELIEFS($B_{t-1}^i, o_t$)
        $j \leftarrow$ WHOSETURN($t$)
        if $j = i$ then
            $a_t \leftarrow$ MCSEARCH($B_t^i, i, \pi_0 \dots \pi_N$)
            Play $a_t$
        else
            $a_t \leftarrow$ GETTEAMMATEACTION()                 ▷ Observe the teammate's action
            for $(\tau, p_\tau) \in \beta_t^i$ do                ▷ $\beta_t^i$ are the non-zero elements of $B_t^i$
                $a_t' \leftarrow \pi_j(\tau^j)$                  ▷ Assuming deterministic $\pi_j$
                If $a_t' \neq a_t$, set $p_\tau$ to 0.

function MCSEARCH($B^i$, $i$, $\pi_0 \dots \pi_N$)
    stats$[a] \leftarrow 0$ for $a \in A$
    while not converged do                                       ▷ We don't describe UCB here
        $\tau \sim B^i$                                          ▷ Sample $\tau$ from the beliefs according to $p_\tau$
        for $a \in A$ do
            score $\leftarrow$ ROLLOUTGAME($\tau$, $a$, $\pi_0 \dots \pi_N$)
            stats$[a] \leftarrow$ stats$[a]$ + score
    return $\operatorname{argmax}_{a \in A}$ stats[a]
```

Algorithm 2 Multi-Agent Search

```
function RetrospectiveMultiAgentSearch($i, \pi_0, \ldots, \pi_N, maxRange$)
    Initialize $B_0$, the common knowledge beliefs at $t = 0$.
    $searcher \leftarrow 0$                              ▷ The player that is *currently* allowed to perform search
    $actionQ \leftarrow \varnothing$                     ▷ The set of actions for which the beliefs have not been updated
    for $t = 1, \ldots, T$ do
        $o_t \leftarrow$ GetEnvironmentObservation()
        $j \leftarrow$ WhoseTurn($t$)
        $B_t \leftarrow$ StepBeliefs($B_{t-1}, o_t^{\mathcal{G}}$)     ▷ $o_t^{\mathcal{G}}$ is the common-knowledge observation
        RetrospectiveBeliefUpdate($\pi_0, \ldots, \pi_N, o_t^{\mathcal{G}}, actionQ, maxRange$)
        if $actionQ = \varnothing$ then
            $searcher = j$
        if $j = i$ then
            $a_t \leftarrow$SelectMove($B_t, i, \pi_0, \ldots, \pi_N, searcher$)
            Play $a_t$
        else
            $a_t \leftarrow$ GetTeammateAction()                ▷ Observe the teammate's action
        $\chi_t^j \leftarrow$GetAohs($B_t, j$)
        if $j = searcher$ then
            $actionQ.push\_back((t, a_t, j, B_t, \chi_t^j))$
        else
            for $(\tau^j, p_{\tau^j}) \in \chi_t^j$ do          ▷ Update beliefs immediately based on blueprint $\pi_j$
                $a'_t \leftarrow \pi_j(\tau^j)$                    ▷ Assuming deterministic $\pi_j$
                If $a'_t \neq a_t$, set $p_{\tau^j}$ to 0.

function SelectMove($B_t, i, \pi_0, \ldots, \pi_N, searcher$)
    if $i = searcher$ then
        $B_t^i \leftarrow$ ConditionOnAOH($B_t, \tau^i$)
        $a \leftarrow$ MCSearch($B_t^i, i, \pi_0, \ldots, \pi_N$)
    else
        $a \leftarrow \pi_i(\tau^i)$
    return $a$

function RetrospectiveBeliefUpdate($\pi_0, \ldots, \pi_N, o_t^{\mathcal{G}}, actionQ, maxRange$)
    for $(a_{t'}, j, B_{t'}, \chi_{t',t}^j) \in actionQ$ do
        Update $\chi_{t',t}^j$ based on $o_t^{\mathcal{G}}$
    while $actionQ \neq \varnothing$ do
        $(a_{t'}, j, B_{t'}, \chi_{t',t}^j) \leftarrow actionQ.front()$
        if $|\chi_{t',t}^j| > maxRange$ then
            break
        for $(\tau^j, p_{\tau^j}) \in \chi_{t',t}^j$ do
            $B_{t'}^j \leftarrow$ ConditionOnAOH($B_{t'}, \tau^j$)
            $a'_{t'} \leftarrow$ MCSearch($B_{t'}^j, j, \pi_0 \ldots \pi_N$)
            if $a'_{t'} \neq a_{t'}$ then
                for $(a_{t'}, j, B_{t'}, \chi_{t',t}^j) \in actionQ$ do     ▷ Prune beliefs at subsequent times
                    Remove any $\tau \in \chi_{t',t}^j$ for which $\tau^j$ is a prefix of $\tau$
        $actionQ.pop\_front()$
```

Experimental Details for Reinforcement Learning

We train the reinforcement learning baseline with our own implementation of Ape-X DQN (Horgan et al. 2018) on the Hanabi Learning Environment (HLE) (Bard et al. 2019). Essentially, Ape-X DQN is a distributed Q-learning framework with a large number of asynchronous actors feeding transitions into a shared prioritized replay buffer (Schaul et al. 2015) in parallel, and a centralized learner that samples from the replay buffer to update the agent. It also incorporates several techniques for improved performance such as n-step return targets (Sutton 1988), double Q-learning (van Hasselt, Guez, and Silver 2015) and dueling network architecture (Wang, de Freitas, and Lanctot 2015). There are two notable differences between our implementation and the one proposed in the original paper. First, instead of having 360 actors each running on a single environment on a CPU, we use 80 actors while each of them running on a vector of 20 environments. We loop over 20 environments and batch their observations together. Then the actor acts on the batch data using GPU. With this modification, our implementation can run under moderate computation resources with 20 CPU cores (40 Hyper-Threads) and 2 GPUs where one GPUs is used for training and the other is shared by all 80 asynchronous actors. Second, we synchronize all actors with learner every 10 mini-batches while in the original paper each actor independently synchronize with the learner every 400 environment steps.

We modify the input features of HLE by replacing card knowledge section with the V0-Belief proposed in (Foerster et al. 2019). To avoid the agent being overly cautious, we do not zero out reward for games in which all life tokens are exhausted during training, even though we report numbers according to the counting scheme at test time. The agent uses a 3 layer fully connected network with 512 neurons each layer, followed by two-stream output layers for value and advantage respectively. Similar to the original Ape-X paper, each actor executes an ϵ_i-greedy policy where $\epsilon_i = \epsilon^{1+\frac{1}{N-1}\alpha}$ for $i \in \{0, ..., N-1\}$ but with a smaller $\epsilon = 0.1$ and $\alpha = 7$. The discount factor γ is set to 0.999. The agent is trained using Adam optimizer (Kingma and Ba 2014) with learning rate $= 6.25 \times 10^{-5}$ and $\epsilon = 1.5 \times 10^{-5}$. Each mini-batch contains 512 transitions sampled from the prioritized replay buffer with priority exponent of 0.6 and importance sampling exponent equal to 0.4.

Experimental Details for Imitation Learning Applied to Search

We train a supervised policy for one agent using sampled AOH-action pairs from 1.3 million games where single-agent search was applied to the SmartBot blueprint policy. The model is a 2-layer, 128-unit LSTM, which takes as input a representation of the AOH as described in the previous section, as well as the action the blueprint agent plays at this AOH. This latter input is available at inference time and is crucial to achieve good performance, presumably because the neural model can “fall back” to the blueprint policy, which may be hard to approximate in some situations (as it’s not a neural policy).

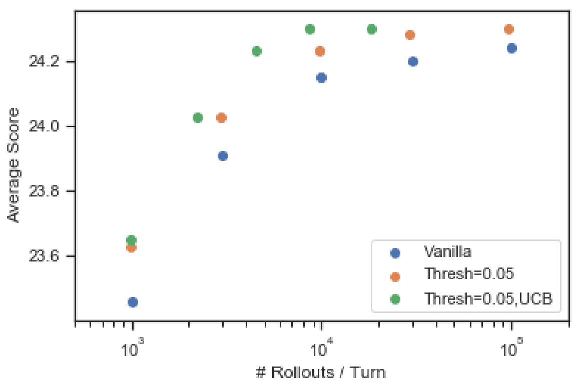

Figure 3: Effect of blueprint deviation threshold and UCB-like search pruning on average score as a function of total number of rollouts per turn when using SmartBot as the blueprint.

The model outputs a softmax policy π_{clone} and is optimized using the objective function is the expected reward $\sum_a \pi_{clone}(\tau^i, a)Q(\tau^i, a)$. At test time we execute the action with the highest probability under the policy. We found that this objective led to a stronger agent than either predicting $Q(\tau^i, a)$ directly with an MSE loss - which wastes model capacity predicting Q for unplayed actions - or predicting a directly with a cross-entropy loss - which forces the model to select between actions with nearly identical value.

We train the model for 100,000 SGD steps, a batch size of 256, using the RMSprop optimizer with a learning rate of 5×10^{-4}.

Additional Results

Figure 3 shows the effect of the blueprint deviation threshold and UCB-like search pruning on average score. Interestingly, a threshold of 0.05 improves the total score even for a large number of rollouts; perhaps this is because the move chosen by SmartBot is more useful for future search optimizations later in the game. UCB reduces the number of rollouts per turn without affecting the average score.

Table 4 lists average scores in 2-player Hanabi using *independent multi-agent search*. In this variant, both players independently perform search (incorrectly) assuming that the partner is playing the blueprint. Without modification, this approach leads to undefined situations where the partner’s beliefs contain no states. To solve this, we add some uncertainty to the model of the partner’s strategy.

In the table, we sweep two parameters: the uncertainty about the partner’s strategy in the belief update, and the threshold minimum difference in expected reward at which the agent will deviate from the blueprint. The rationale for the threshold is that any time a player deviates from the blueprint, they are corrupting their partner’s beliefs, so this should only be done when it is substantially beneficial. Under some settings, independent joint search performs better than the blueprint (22.99) but never outperforms single-agent search.

Belief	ΔR **Threshold**			
Uncertainty	0	0.1	0.2	0.5
0.05	14.41	23.62	24.18	24.02
0.1	14.22	23.48	24.12	24.01
0.2	13.69	22.41	23.57	23.83

Table 4: Average scores in 2-player Hanabi for *independent multi-agent search* using the SmartBot blueprint strategy.

Table 5 shows new state-of-the-art results in 3, 4, and 5 player Hanabi by applying search on top of the information-theoretic 'WTFWThat' hat-coding policy. This hat-coding policy achieves near-perfect scores but is not learned and considered somewhat against the spirit of the game, since it does not use grounded information at all. Nevertheless, we show here that even these policies can be improved with single-agent search.

# Players	**No Search**	**Single-Agent Search**
2	19.45 ± 0.001 0.28%	22.78 ± 0.02 10.0%
3	24.20 ± 0.01 49.1%	$\mathbf{24.83 \pm 0.006}$ **85.9%**
4	24.83 ± 0.01 87.2%	$\mathbf{24.96 \pm 0.003}$ **96.4%**
5	24.89 ± 0.00 91.5%	$\mathbf{24.94 \pm 0.004}$ **95.5%**

Table 5: Hanabi performance for different numbers of players, using the hard-coded 'WTFWThat' hat-counting blueprint agent which achieved state-of-art performance in 3, 4, and 5-player Hanabi. Results shown in bold are state-of-the-art for that number of players.

Proof of Main Theorem

Theorem 2. *Consider a Dec-POMDP with N agents, actions $\mathcal{A}$, reward bounded by $r_{min} \leq R(\tau) < r_{max}$ where $r_{max} - r_{min} \leq \Delta$, game length bounded by T, and a set of blueprint policies $\pi_b \equiv \{\pi_b^0 \dots \pi_b^N\}$. If a MC search policy π_s is applied using N rollouts per step, then*

$$V_{\pi_s} - V_{\pi_b} \geq -2T\Delta|\mathcal{A}|N^{-1/2} \tag{8}$$

Proof. Consider agent i at some (true) trajectory τ. This agent has an exact set of beliefs $B^i(\tau^i)$, i.e. the exact probability distribution over trajectories it could be in given its action-observation history τ^i.

Suppose that an agent at some point τ acts according to the search procedure described in Section (it is not relevant whether single-agent or joint search is performed). For each action $a \in \mathcal{A}$, the agent collects $N/|\mathcal{A}|$ i.i.d. samples of the reward R, sampling trajectories from $\tau \sim B^i(\tau^i)$ and assuming that agent i plays some action a at τ^i and all agents play according to π_b thereafter. We denote a single MC rollout (which is an MC estimate of $Q_{\pi_b}(\tau^i, a)$) as $\hat{Q}^k_{\pi_b}(\tau^i, a)$, and the mean of the rollouts for a as $\hat{Q}_{\pi_b}(\tau^i, a)$. The agent then plays $\hat{a}^* \equiv \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \hat{Q}_{\pi_b}(\tau^i, a)$. We denote the *true* optimal action as $a^* \equiv \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, Q_{\pi_b}(\tau^i, a)$.

$$\begin{aligned}
Q_{\pi_b}(\tau^i, \hat{a}^*) - V_{\pi_b}(\tau^i) &= Q_{\pi_b}(\tau^i, \hat{a}^*) - Q_{\pi_b}(\tau^i, \pi_b(\tau^i)) &(9)\\
&= Q_{\pi_b}(\tau^i, \hat{a}^*) - \hat{Q}_{\pi_b}(\tau^i, \hat{a}^*) + \hat{Q}_{\pi_b}(\tau^i, \hat{a}^*) &(10)\\
&\qquad - \hat{Q}_{\pi_b}(\tau^i, \pi_b(\tau^i)) + \hat{Q}_{\pi_b}(\tau^i, \pi_b(\tau^i)) - Q_{\pi_b}(\tau^i, \pi_b(\tau^i)) &\\
&\geq Q_{\pi_b}(\tau^i, \hat{a}^*) - \hat{Q}_{\pi_b}(\tau^i, \hat{a}^*) + \hat{Q}_{\pi_b}(\tau^i, \pi_b(\tau^i)) - Q_{\pi_b}(\tau^i, \pi_b(\tau^i)) &(11)\\
&\geq -2 \max_{a\in\mathcal{A}} \left|\hat{Q}_{\pi_b}(\tau^i, a) - Q_{\pi_b}(\tau^i, a)\right| &(12)
\end{aligned}$$

Line (10) adds canceling terms and line (11) simplifies the expression using the definition of $\hat{a}^*$.

We will now use a concentration inequality to bound (12). To do so we will need refer to some facts about *subgaussian* random variables, which loosely means those whose tails die off at least as fast as a Gaussian.

Definition: X is σ-subgaussian for some $\sigma > 0$ if $\forall s \in \mathbb{R}$, $\mathbb{E}\left[e^{sX}\right] \leq e^{\frac{\sigma^2 s^2}{2}}$.

Lemma 1 (Hoeffding's lemma (1963))**.** *Let X be a random variable with mean 0 and $X \in [a, b]$, $\Delta = b - a$. Then X is $\Delta/2$-subgaussian.*

The proof is in (Rigollet 2015), pg. 20.

Lemma 2. *Suppose that $X_1, \dots, X_n$ are independent and σ-subgaussian. Then their mean is $\frac{\sigma}{\sqrt{n}}$-subgaussian.*

Proof of Lemma 2.

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\frac{s}{n}\sum_{1\leq i\leq n} X_i\right)\right] &= \mathbb{E}\left[\prod_{1\leq i\leq n} \exp\left(sX_i/n\right)\right]\\
&= \prod_{1\leq i\leq n} \mathbb{E}\left[\exp\left(sX_i/n\right)\right] \qquad \text{by independence}\\
&= \prod_{1\leq i\leq n} \exp\left(\frac{\sigma^2 s^2}{2n^2}\right)\\
&= \exp\left(\frac{s^2}{2n^2}\sum_{1\leq i\leq n} \sigma^2\right)\\
&= \exp\left(\frac{\left(\sigma/\sqrt{n}\right)^2 s^2}{2}\right)
\end{aligned}$$

$\square$

Lemma 3. *Suppose that $X_1, \dots, X_n$ are each σ-subgaussian. Then*

$$\mathbb{E}\left[\max_{1 \le i \le n} |X_i|\right] \le \sigma\sqrt{2\log(2n)} \tag{13}$$

The proof is in (Rigollet 2015), pg. 25.

Now we can apply these results to our setting. $\hat{Q}^k_{\pi_b}(\tau^i, a) - Q_{\pi_b}(\tau^i, a)$ has mean 0 and bounded width Δ, therefore by Lemma 1 it is $\Delta/2$-subgaussian. The search procedure performs $N/|\mathcal{A}|$ rollouts per action, so from Lemma 2 the mean $\hat{Q}_{\pi_b}(\tau^i, a) - Q_{\pi_b}(\tau^i, a)$ is $\left(\frac{\Delta}{2}\sqrt{\frac{|\mathcal{A}|}{N}}\right)$-subgaussian. Finally, applying Lemma 3 to the bound in (12), we have

$$Q_{\pi_b}(\tau^i, \hat{a}^*) - V_{\pi_b}(\tau^i) \ge -2\left(\frac{\Delta}{2}\sqrt{\frac{|\mathcal{A}|}{N}}\right)\sqrt{2\log(2|\mathcal{A}|)} \tag{14}$$

We can simplify this a bit using the fact that $\log(X) \le X$ for $X \ge 1$:

$$Q_{\pi_b}(\tau^i, \hat{a}^*) - V_{\pi_b}(\tau^i) \ge -2\Delta|\mathcal{A}|N^{-1/2} \tag{15}$$

We can now prove the main theorem. We denote the policy that follows the MC search procedure up to time t and the blueprint π_b thereafter as $\pi_{s\to t}$. We will prove by induction on t that

$$V_{\pi_{s\to t}}(\tau_0) - V_{\pi_b} \ge -2t\Delta|\mathcal{A}|N^{-1/2}. \tag{16}$$

The base case is satisfied by definition, since $\pi_{s\to 0} \equiv \pi_b$.

Suppose that at some time t, Equation 16 is satisfied.

$$\begin{aligned}
V_{\pi_{s\to(t+1)}} &= \mathbb{E}_{\tau_{t+1}\sim\mathcal{P}(\tau_0,\pi_{s\to t})}\left[\mathbb{E}_{a\sim\pi_s(\tau^i_{t+1})}\left[Q_\pi(\tau^i_{t+1}, a)\middle|\,\tau_{t+1}\right]\right] && && (17)\\
&= \mathbb{E}_{\tau_{t+1}\sim\mathcal{P}(\tau_0,\pi_{s\to t}),a\sim\pi_s(\tau^i_{t+1})}\left[Q_{\pi_b}(\tau^i_{t+1}, a)\right] && \text{(Law of Total Exp.)} && (18)\\
&\ge \mathbb{E}_{\tau_{t+1}\sim\mathcal{P}(\tau_0,\pi_{s\to t})}\left[V_{\pi_b}(\tau^i_{t+1})\right] - 2\Delta|\mathcal{A}|N^{-1/2} && \text{(Eq. 15)} && (19)\\
&= V_{\pi_{s\to t}}(\tau^i_t) - 2\Delta|\mathcal{A}|N^{-1/2} && && (20)\\
&\ge V_{\pi_b} - 2(t+1)\Delta|\mathcal{A}|N^{-1/2} && \text{(Eq. 16)} && (21)
\end{aligned}$$

Rearranging,

$$V_{\pi_{s\to(t+1)}} - V_{\pi_b} \ge -2(t+1)\Delta|\mathcal{A}|N^{-1/2} \tag{22}$$

This completes the proof by induction. Since the game length is bounded by T, $V_{\pi_{s\to T}} \equiv V_{\pi_s}$ and the proof is complete. □