Best-response play in partially observable card games

Frans Oliehoek FAOLIEHO@SCIENCE.UVA.NL
Matthijs T. J. Spaan MTJSPAAN@SCIENCE.UVA.NL
Nikos Vlassis VLASSIS@SCIENCE.UVA.NL
Informatics Institute, Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

We address the problem of how to play optimally against a fixed opponent in a two-player card game with partial information like poker. A game theoretic approach to this problem would specify a pair of stochastic policies that are best-responses to each other, i.e., a Nash equilibrium. Although such a Nash-optimal policy guarantees a lower bound to the attainable payoff against any opponent, it may not necessarily be optimal against a fixed opponent. We show here that if the opponent's policy is fixed (either known or estimated by repeated play), then we can model the problem as a partially observable Markov decision process (POMDP) from the perspective of one agent, and solve it by dynamic programming. In particular, for a large class of card games including poker, the derived POMDP consists of a finite number of belief states and it can be solved exactly. The resulting policy is guaranteed to be optimal even against a Nash-optimal policy. We provide experimental results to support our claims, using a simplified 8-card poker game in which Nash-policies can be computed efficiently.

1. Introduction

A partially observable stochastic game (POSG) is general model that captures the sequential interaction of two or more agents under conditions of uncertainty. This model can be regarded as an extension of a stochastic game (Shapley, 1953), with parts of the state being hidden to at least one agent. It can also be viewed as an extension of a partially observable Markov decision process (POMDP) (Sondik, 1971), with state transitions being influenced by the combined actions of two or more agents. A POSG is also very closely related to the model of an extensive game with imperfect information (Kuhn, 1953).

The literature on POSGs is still relatively sparse. Hespanha and Prandini (2001) showed that a two-player finite-horizon POSG always has a Nash equilibrium in stochastic policies. Koller et al. (1994) demonstrated how to efficiently compute such a Nash equilibrium in the special case of a two-player zero-sum POSG with a tree-like structure, like the card games (e.g., poker) we consider here. Becker et al. (2003), Nair et al. (2003) and Emery-Montemerlo et al. (2004) have developed similar algorithms for computing solutions in the special case of common-interest POSGs. Only recently an algorithm for solving general (albeit still small) POSGs has been proposed (Hansen et al., 2004).

In this paper we consider the class of two-player zero-sum finite-horizon POSGs with a tree-like state structure, that includes many card games like poker. We depart from the game theoretic approach of computing Nash equilibria for these games, and instead deal with the problem of how to compute optimal policies (best-responses) against a fixed opponent. Our interest is motivated by recent suggestions to adopt an 'agent-centric' agenda in multiagent decision making, by which best-response learning algorithms (like Q-learning) replace classical game theoretic approaches to finding an optimal policy (Powers & Shoham, 2005). There are two strong arguments in favor of this approach: first, computing a Nash equilibrium is a difficult problem in general, and efficient algorithms exist only in special cases. Second, a Nash-policy is too conservative in the sense that it will not exploit possible weaknesses of opponents.

As we show below, in order to compute a best-response policy against a fixed opponent in a game like poker, we can model the game as a partially observable Markov decision process (POMDP) by defining a belief state for our protagonist agent. This reduction only requires knowing (e.g., estimating by repeated play) the

stochastic policy of the opponent. The POMDP can be subsequently solved, for instance by dynamic programming, deriving a best-response (deterministic) policy for the agent. In general, an approximate POMDP algorithm may be needed to deal with the continuous belief space (Spaan & Vlassis, 2004). For many card games however, including poker, the particular structure of the problem allows the POMDP to be solved exactly: there are a finite set of reachable beliefs in the POMDP, which allows mapping the POMDP model into a discrete-state MDP and then solve it with an exact dynamic programming method (e.g., value iteration). The resulting policy is optimal (gives the highest possible payoff) against the particular opponent. Moreover we can easily prove that it is no worse than the optimal Nash-policy when playing against a Nash-agent (an opponent that uses a Nash-optimal policy).

To illustrate the method, we have implemented a simplified 8-card poker game for which Nash equilibria can be computed (Koller & Pfeffer, 1997). Then, using the POMDP approach, we have computed best-response policies against Nash-agents. We have experimentally verified that a POMDP best-response policy can indeed reach the optimal Nash payoff when playing against a Nash-optimal agent.

In the following, we first describe our simplified poker game that we use as a running example throughout the paper (Section 2). Then we briefly outline the game-theoretic approach for solving such partially observable games (Section 3). In Section 4 we describe our POMDP-based approach to playing poker against a fixed opponent. We provide experimental results in Section 5, and Section 6 concludes.

2. A simplified poker game

Poker is an example of a stochastic game with partial (imperfect or incomplete) information. In poker a player cannot tell the exact state of the game (e.g., the card deal), and he does not know what policy his opponent will follow. Still, his (sequential) decision making must include comparing potential reward to the risk involved, trying to deceive the opponent by bluffing, dealing with unreliable information from the opponents' actions, and modeling the opponent. All these aspects make poker a very complex game.

There are many poker variants, which all share these properties (Billings et al., 2003). Most of these variants, however, are too large to analyze in an exact fashion because of the number of card combinations and possible betting sequences. Our goal in this paper is not to compute the ultimate strategy for playing general poker, but to address the problem how to play optimally against a fixed opponent. As the opponent strategy we consider can be an optimal Nash policy, we need to be able to compute the Nash policies. To this end, and as running example, we will use a small 8-card poker variant, described by Koller and Pfeffer (1997), which we Nash-solved by using their Gala system.

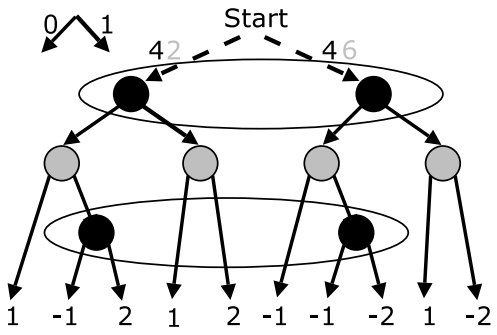


Figure 1. The partial game-tree of 8-card poker for the deals $(4, 2)$ and $(4, 6)$. Gambler's decision nodes are black, dealer's are grey. The payoffs are given for the gambler.

This 8-card poker is played by two players: a dealer and a gambler, who both own two coins. Before the game starts, each player puts one coin to the pot, the ante. Then the dealer deals both players one card out of a deck of eight cards (1 suit, ranks 1–8). After the players have observed their card, they are allowed to bet their remaining coin, starting with the gambler. If the gambler bets his coin, the dealer has the option to fold or call. If the dealer folds he loses the ante, and if he calls showdown follows. If the gambler does not bet, the dealer can choose to bet his coin. If the dealer does so, the gambler will have to decide whether to fold or call. If the game reaches the showdown (neither player bets or the bet of the dealer is called), the player with the highest card wins the pot.

3. Game-theoretic approach

In this section we will briefly outline the game theoretic approach for representing and solving such poker games.

3.1. Poker as an extensive form game

We can model 8-card poker as an *extensive form* game with partial (imperfect) information (Kuhn, 1953). The extensive form of a game is given by a tree, in which nodes represent game states and whose root is the starting state. There are two types of nodes: de-

cision nodes that represent points at which agents can make a move, and chance nodes which represent random transitions by 'nature'. In 8-card poker, the only chance node is the starting state, in which two cards are chosen uniformly at random from the 8-card deck and are dealt to the agents.

In a partial information game, an agent may be uncertain about the true state of the game. In particular, an 8-card poker agent may not be able to discriminate between two nodes in the tree. The nodes that an agent cannot tell apart are grouped in *information sets*. In Fig. 1 a part of the game-tree of 8-card poker is drawn. At the root of tree ('Start' node) a card is dealt to each agent. At each decision node the agents can choose between action 1 (*bet*), and action 0 (*fold*). Fig. 1 shows the partial game tree for two deals: in the first the dealer receives card 2, in the second he receives card 6. The gambler receives card 4 in both cases. Therefore the gambler cannot discriminate between the two deals. This is illustrated by the information sets indicated by ovals. The leaves of the tree represent the outcomes of the game and the corresponding payoffs. In the figure we only show the payoff of the gambler; the payoff of the dealer is exactly the opposite. Games in which payoffs add up to zero, as is the case here, are called *zero-sum* games.

3.2. Solving poker

Solving poker means computing the optimal *policies* the agents should follow. In 8-card poker, a policy for an agent is a mapping from his information sets to actions. A *pure* policy specifies, for each information set, an action that should be taken with probability one. A *stochastic* policy is a probability distribution over pure policies: it specifies, for each information set, an action that should be taken with some specific probability. For example, in the 8-card poker game shown in Fig. 1, a stochastic policy for the gambler could specify that he should bet with probability 0.4 after having received card 4.

The solution of a game specifies how each agent should play *given that the opponent also follows this advise*. As such, it provides an optimal policy for each agent. The solution of a game is given by one or more of its Nash equilibria. Let π_i denote a policy for agent i. A pair of policies $\pi = (\pi_1, \pi_2)$ induce expected payoff $H_i(\pi)$ to agent i, where the expectation is over over the chance nodes in the game; in a zero-sum game $H_1(\pi) = -H_2(\pi)$. When π is a Nash equilibrium, $H_1(\pi)$ is the *value of the game*.

Definition 1. A tuple of policies $\pi = (\pi_1, \pi_2)$ is a Nash equilibrium if and only if it holds that:

$$\begin{aligned} &\forall_{\pi_1'} \left(H_1(\pi_1, \pi_2) \geq H_1(\pi_1', \pi_2)\right) \wedge \\ &\forall_{\pi_2'} \left(H_2(\pi_1, \pi_2) \geq H_2(\pi_1, \pi_2')\right) \end{aligned} \tag{1}$$

That is, for each agent i, playing π_i gives an equal or higher expected payoff than playing π_i'. So both policies are best responses to each other.

In two-player zero-sum games, a Nash policy is a security policy and the value of the game is the security value for an agent. The latter means that the expected payoff of an agent who plays a Nash policy cannot be lower than the value of the game. In other words, a Nash policy gives the payoff that an agent can maximally guarantee for himself, given that the opponent will act in a best-response manner in order to minimize this payoff.

Nash (1951) and Kuhn (1953) have shown that any extensive-form game with perfect recall[1] has at least one Nash equilibrium in stochastic policies. Moreover, in two-player zero-sum games, all Nash equilibria specify the same value for the game (Osborne & Rubinstein, 1994). Therefore, *any* Nash policy is optimal against a best-response opponent, in the sense that it guarantees the security value of the game.

3.3. Computing Nash equilibria

To find the Nash equilibria of an extensive form game, the game can be first transformed into its *normal form*. This is a matrix representation of all pure policies available to the agents. Entry (i, j) of the matrix gives the expected payoff of agent 1's policy i versus agent 2's policy j. Consequently, when converting from the extensive to the normal form, the tree-like structure of the game is discarded. Moreover, the normal form representation of an extensive game can be very large. Note that every pure policy is a mapping from information sets to actions, therefore the number of pure policies is exponential in the size of the game tree.

To compute the Nash equilibria from the normal form we can use linear programming. The normal form of a two-player zero-sum game defines a linear program whose solutions are the Nash-equilibria of the game. However, transforming an extensive form game to its normal form results in very large games, making such an approach for computing Nash equilibria impractical.

Koller and Pfeffer (1997) proposed the 'Gala' system, which solves games in an alternative representation

[1] Perfect recall implies that an agent remembers all actions that he has taken in the past.

called the *sequence* form, whose size is polynomial with the number of nodes in the game tree. This allows one to solve larger games, but real-life games are typically still too large to solve. For example consider Texas Hold-em poker[2], whose game-tree contains $O(10^{18})$ nodes (Billings et al., 2003) and therefore computing Nash equilibria is computationally infeasible.

4. Playing against a fixed opponent

In this section we depart from the game-theoretic approach, and address the problem of how to play optimally against a fixed opponent. It turns out that, if the (stochastic) policy of the opponent agent j can be summarized by a model in the form $p(a_j|s, a_i)$, where a_j denotes his action at some state s and a_i our action, then we can model the poker game as a partially observable Markov decision process (POMDP) *from the perspective* of our protagonist agent i, and compute an optimal (deterministic) policy for it.

According to our POMDP model, at any time step the game is in a state $s \in S$, where the state space S contains all chance nodes in the game-tree, as well as all nodes in which our protagonist agent i takes a decision (the black nodes in Fig. 1 if agent i is the gambler). The game starts at the 'Start' state (chance node). At any other state (decision-node) s, our protagonist agent i takes an action $a_i \in A_i = \{bet, fold\}$, and the opponent takes an action a_j according to a stochastic policy $\pi_j = p(a_j|s, a_i)$. Then the game switches to a new state s' as a result of the two actions (a_1, a_2) and according to a stochastic joint transition model $p(s'|s, a_i, a_j)$. Using the policy of the opponent, we can compute a single transition model for our agent i as follows:

$$p(s'|s, a_i) = \sum_{a_j} p(s'|s, a_i, a_j)p(a_j|s, a_i). \quad (2)$$

This allows us to treat the game as a POMDP from the perspective of our protagonist agent i. In our 8-card poker, the joint transition model $p(s'|s, a_i, a_j)$ is stochastic only in the 'Start' state and deterministic elsewhere (given the actions of the two agents), since there are no other chance nodes in the game-tree. However, as we assume a stochastic policy for agent j, from the viewpoint of agent i the game is stochastic in every state.

In partial information games the agents cannot directly observe the true state of the game, but they receive observations (clues) about the state. In our POMDP model of the game, in each state s our protagonist agent i perceives an observation $o_i \in O_i$ that is related to the current state s and his last action a_i through a stochastic observation model $p(o_i|s, a_i)$. The first observation o_i agent i receives indicates the hand that is dealt to it, consecutive observations signal the last action a_j of the opponent. Moreover, the observation model is deterministic: after the cards are dealt an agent observes $o_i \in \{1, \ldots, 8\}$ with probability one, and in consecutive states he perceives $o_i = a_j$ with full certainty.

At every time step an agent k receives an individual scalar reward signal $r_k(s, a_i, a_j, s')$ based on the previous game state s, current state s' and the joint action (a_i, a_j). Using the policy of the opponent agent j we can compute the reward our agent i receives as follows:

$$r_i(s, a_i, s') = \sum_{a_j} r_i(s, a_i, a_j, s')p(a_j|s, a_i). \quad (3)$$

In poker the reward is 0 except for transitions into one of the end-states, i.e., when one of the agents has folded or the game reaches showdown (Section 2).

As all sets S, O_i, and A_i are discrete and finite in a poker game, we can convert the discrete POMDP model in a continuous belief-state Markov decision process (MDP) (Sondik, 1971), in which the agent summarizes all information about its past using a *belief* vector $b(s)$. The belief b is a probability distribution over S, and grants the agent perfect recall. Our agent i starts with an initial belief b_0, which in our poker setting is set to a Dirac distribution on the 'Start' state. Every time agent i takes an action a_i and observes o_i, it's belief is updated by Bayes' rule:

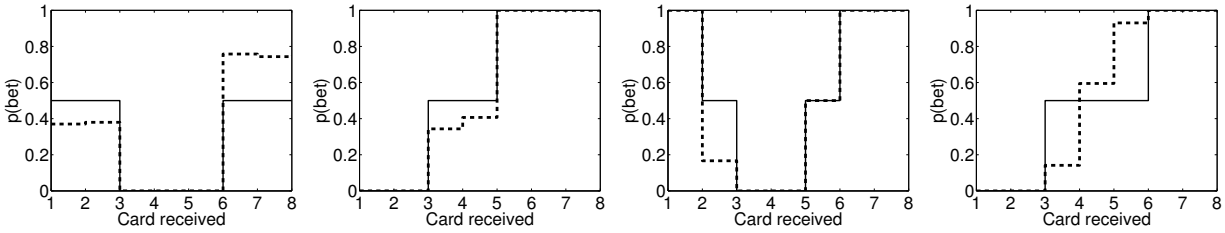
$$b^{a_i}_{o_i}(s') = \frac{p(o_i|s', a_i)}{p(o_i|a_i, b)} \sum_{s \in S} p(s'|s, a_i)b(s), \text{ where} \quad (4)$$

$$p(o_i|a_i, b) = \sum_{s' \in S} p(o_i|s', a_i) \sum_{s \in S} p(s'|s, a_i)b(s) \quad (5)$$

is a normalizing constant. To solve the belief-state MDP in general a large range of POMDP solution techniques can be applied, including exact (Sondik, 1971) or approximate ones (Spaan & Vlassis, 2004).

It turns out, however, that in the class of two-player card games that we are considering only a relatively small *finite* set of beliefs B can ever be experienced by the agent. The set is finite as the problem has a finite horizon, and small because the horizon is low and the sets O_i and A_i are small. Furthermore, after the card dealing, in each state only one of two observations is possible (the action of the opponent agent), reducing the branching factor of the tree of beliefs. In partic-

[2]Texas Hold-em is a popular poker variant in which each player gets two private cards and share 5 public cards.



(a) Gambler policy at start of game.

(b) Gambler policy after seeing a bet.

(c) Dealer policy after seeing a pass.

(d) Dealer policy after seeing a bet.

Figure 2. POMDP and Nash policies for 8-card poker. Dashed lines indicate a Nash policy computed by Gala and solid lines the best-response POMDP policy against that Nash policy, computed by solving the belief-state MDP. The x-axis denotes the card dealt to the gambler ((a),(b)) or the dealer ((c),(d)), and the y-axis indicates the probability of betting.

ular, each information set of agent i in the game-tree will induce a single belief.

Given a finite set B we can compute a finite belief-state MDP from the continuous belief-state MDP, by taking each $b \in B$ as a possible state. Computing a transition model $p(b'|b, a_i)$ specifying how our agent i switches from a particular belief b to another belief b' when taking action a_i is straightforward:

$$p(b'|b, a_i) = p(o_i|a_i, b), \tag{6}$$

where $p(o_i|a_i, b)$ is giving by (5). The reward $r_i(b, a_i, b')$ is defined as:

$$r_i(b, a_i, b') = \sum_{s,s'} r_i(s, a_i, s')b(s)b'(s'). \tag{7}$$

This MDP can now be solved in an exact fashion using standard dynamic programming techniques, for instance using value iteration (Sutton & Barto, 1998). The result is a deterministic best-response policy π_i for our protagonist agent, that maps beliefs it encounters to optimal actions. Value iteration allows us to compute the value of the initial belief state, which equals the expected reward of the game when agent i follows π_i and agent j behaves according to π_j.

Note that the derived best-response policy for the protagonist agent is a deterministic one: we know that the optimal policy of a POMDP is always deterministic (Puterman, 1994). Although this may seem a limiting factor, we can use the following result from game theory: a stochastic policy is a best-response policy against some opponent if and only if all the deterministic policies to which it assigns nonzero probability are also best-response policies to this opponent. Our POMDP derived policies are best-response policies because our agent maximizes his payoff *exactly* over the space of his deterministic policies, and therefore they *must* be in the support of a best-response stochastic policy. In other words, a deterministic POMDP policy is equally good to any best-response stochastic policy in terms of achieved payoff.

5. Experiments

We will now present an experiment performed in the 8-card poker setting, in which we solved 8-card poker as described in Section 2 using the Gala system. The results from this were a pair of optimal Nash policies and the value of the game, which is $+0.0625$ coins per game in favor of the dealer. The next step was to create a POMDP model for the gambler, using the found optimal Nash policy for the dealer by incorporating the dealer's policy in the POMDP transition model, as described in Section 4. This was also done with the roles reversed. From this POMDP model a finite belief-state MDP was extracted which we solved using value iteration. To construct the policy, the action with the highest expected payoff for a belief was selected. When for a certain belief the expected values for both actions are equal, these actions are taken with equal probability.

The resulting policies are shown in Fig. 2. The expected payoff for the POMDP policies is equal to the value attained by the optimal policies ($+0.0625$ for the dealer and -0.0625 for the gambler) and as these are the best payoffs obtainable, the policies are clearly best-response policies. Furthermore, we see that the computed POMDP policies are quite similar to the Nash policies. In particular, betting with probability 1 or 0 happens in exactly the same situations as in the Nash policies. However there are situations in which the two policies differ: the cases in which the POMDP policy is indifferent between both actions and which are assigned probability 0.5.

6. Conclusions

In this paper we addressed the problem of computing a best-response policy for an agent playing against a fixed opponent in a partially observable card game like poker. In such a card game an agent only receives partial information regarding the true state of the game, i.e., the cards dealt to each agent. An agent can only observe its own hand and the action the other agent has executed. A second source of uncertainty is the unknown policy of the other agent. A game-theoretic approach to solving such games would be to compute a pair of stochastic policies that are best-responses to each other, i.e., a Nash equilibrium. Unfortunately, computing Nash equilibria is a difficult problem in general and such a Nash policy is secure but conservative: it will not exploit possible weaknesses of an opponent.

However, when we assume the opponent agent has a fixed policy (known or estimated by repeated play), we can model the game as partially observable Markov decision process (POMDP) from the perspective of our protagonist agent. We have shown that by solving the resulting POMDP model we can compute a deterministic best-response policy for our agent. We focused on a simplified 8-card poker game in which Nash equilibria can be computed. We have argued and experimentally verified that the computed POMDP best-response policy can indeed reach the optimal Nash payoff when playing against a Nash-optimal agent. Avenues of future research include investigating more compact state representations, tackling larger poker variations and considering more general partially observable stochastic games.

References

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2003). Transition-independent decentralized Markov decision processes. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*.

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., & Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker. *Proc. Int. Joint Conf. on Artificial Intelligence*. Acapulco, Mexico.

Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*.

Hansen, E., Bernstein, D., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. *Proc. 19th National Conf. on Artificial Intelligence (AAAI-04)*. San Jose.

Hespanha, J., & Prandini, M. (2001). Nash equilibria in partial-information games on Markov chains. *Proc. of the 40th Conf. on Decision and Control*.

Koller, D., Megiddo, N., & von Stengel, B. (1994). Fast algorithms for finding randomized strategies in game trees. *Proc. of the 26th ACM Symposium on Theory of Computing (STOC)* (pp. 750–759).

Koller, D., & Pfeffer, A. (1997). Representations and solutions for game-theoretic problems. *Artificial Intelligence*, *94*, 167–215.

Kuhn, H. (1953). Extensive games and the problem of information. *Annals of Mathematics Studies*, *28*, 193–216.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D., & Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. *Proc. Int. Joint Conf. on Artificial Intelligence*. Acapulco, Mexico.

Nash, J. F. (1951). Non-cooperative games. *Annals of Mathematics*, *54*, 286–295.

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. MIT Press.

Powers, R., & Shoham, Y. (2005). New criteria and a new algorithm for learning in multi-agent systems. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*. Cambridge, MA: MIT Press.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc.

Shapley, L. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, *39*, 1095–1100.

Sondik, E. J. (1971). *The optimal control of partially observable Markov decision processes*. Doctoral dissertation, Stanford University.

Spaan, M. T. J., & Vlassis, N. (2004). *Perseus: randomized point-based value iteration for POMDPs* (Technical Report IAS-UVA-04-02). Informatics Institute, University of Amsterdam.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.