On Convergence and Optimality of Best-Response Learning with Policy Types in Multiagent Systems

Stefano V. Albrecht
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
s.v.albrecht@sms.ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
s.ramamoorthy@ed.ac.uk

Abstract

While many multiagent algorithms are designed for homogeneous systems (i.e. all agents are identical), there are important applications which require an agent to coordinate its actions without knowing a priori how the other agents behave. One method to make this problem feasible is to assume that the other agents draw their latent policy (or type) from a specific set, and that a domain expert could provide a specification of this set, albeit only a partially correct one. Algorithms have been proposed by several researchers to compute posterior beliefs over such policy libraries, which can then be used to determine optimal actions. In this paper, we provide theoretical guidance on two central design parameters of this method: Firstly, it is important that the user choose a posterior which can learn the true distribution of latent types, as otherwise suboptimal actions may be chosen. We analyse convergence properties of two existing posterior formulations and propose a new posterior which can learn correlated distributions. Secondly, since the types are provided by an expert, they may be inaccurate in the sense that they do not predict the agents' observed actions. We provide a novel characterisation of optimality which allows experts to use efficient model checking algorithms to verify optimality of types.

1 INTRODUCTION

Many multiagent algorithms are developed with a homogeneous setting in mind, meaning that all agents use the same algorithm and are a priori aware of this fact. However, there are important applications for which this assumption may not be adequate, such as human-machine interaction, robot search and rescue, and financial markets. In such problems, it is important that an agent be able to effectively coordinate its actions without knowing *a priori* how the other agents

behave. The importance of this problem has been discussed in works such as [Albrecht and Ramamoorthy, 2013, Stone et al., 2010, Bowling and McCracken, 2005].

This problem is hard since the agents may exhibit a large variety of behaviours. General-purpose algorithms for multiagent learning are often impracticable, either because they take too long to produce effective policies or because they rely on prior coordination of behaviours [Albrecht and Ramamoorthy, 2012]. However, it has been recognised (e.g. [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011]) that the complexity of this problem can often be reduced by assuming that there is a latent set of policies for each agent and a latent distribution over these policies, and that a domain expert can provide informed guesses as to what the policies might be. (These guesses could also be generated automatically, e.g. using some machine learning method on a corpus of historical data.)

One algorithm that takes this approach is *Harsanyi-Bellman Ad Hoc Coordination* (HBA) [Albrecht and Ramamoorthy, 2013]. This algorithm maintains a set of user-defined types (by "type", we mean a policy or programme which specifies the behaviour of an agent) over which it computes posterior beliefs based on the agents' observed actions. The beliefs are then used in a planning procedure to compute expected payoffs for all actions (a procedure combining the concepts of Bayesian Nash equilibrium and Bellman optimality) and the best action is chosen. HBA was implemented as a reinforcement learning procedure and shown to be effective in both simulated and human-machine problems [Albrecht and Ramamoorthy, 2013]. Similar algorithms were studied in [Barrett et al., 2011, Carmel and Markovitch, 1999].

While works such as [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011, Carmel and Markovitch, 1999] demonstrate the practical usefulness of such methods, they provide no theoretical guidance on two central design parameters: Firstly, one may compute the posterior beliefs in various ways, and it is important that the user choose a posterior formulation which is able to accurately approximate the latent distribution of types. This is important as otherwise the expected payoffs may be inaccurate, in which case HBA may

choose suboptimal actions. In this paper, we analyse the convergence conditions of two existing posterior formulations and we propose a new posterior which can learn correlated type distributions. These theoretical insights can be applied by the user to choose appropriate posteriors.

Secondly, since the types are provided by the user (or generated automatically), they may be inaccurate in the sense that their predictions deviate from the agents' observed actions. This raises the need for a theoretical analysis of how much and what kind of inaccuracy is acceptable for HBA to be able to solve its task, by which we mean that it drives the system into a terminal state. (A different question pertains to payoff maximisation; we focus on task accomplishment as it already includes many practical problems.) We describe a methodology in which we formulate a series of desirable termination guarantees and analyse the conditions under which they are met. Furthermore, we provide a novel characterisation of optimality which is based on the notion of probabilistic bisimulation [Larsen and Skou, 1991]. In addition to concisely defining what constitutes optimal type spaces, this allows the user to apply efficient model checking algorithms to verify optimality in practice.

2 RELATED WORK

Opponent modelling methods such as case-based reasoning [Gilboa and Schmeidler, 2001] and recursive modelling [Gmytrasiewicz and Durfee, 2000] are relevant to the extent that they can complement the user-defined types by creating new types (the opponent models) on the fly. For example, [Albrecht and Ramamoorthy, 2013] used a variant of case-based reasoning and [Barrett et al., 2011] used a tree-based classifier to complement the user-defined types.

Plays and play books [Bowling and McCracken, 2005] are similar in spirit to types and type spaces. However, plays specify the behaviour of an entire team, with additional structure such as applicability and termination conditions, and roles for each agent. In contrast, types specify the action probabilities of a single agent and do not require commitment to conditions and roles.

Plans and plan libraries [Carberry, 2001] are conceptually similar to types and type spaces. However, the focus of plan recognition has been on identifying the goal of an agent (e.g. [Bonchek-Dokow et al., 2009]) and efficient representation of plans (e.g. [Avrahami-Zilberbrand and Kaminka, 2007]), while types are used primarily to compute expected payoffs and can be efficiently represented as programmes [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011].

I-POMDPs [Gmytrasiewicz and Doshi, 2005] and I-DIDs [Doshi et al., 2009] are related to our work since they too assume that agents have a latent type. These methods are designed to handle the full generality of partially observable states and latent types, and they explicitly model nested

beliefs. However, this generality comes at a high computational cost and the solutions are infeasible to compute in many cases. In contrast, we remain in the setting of fully observable states, and we implicitly allow for complex beliefs within the specification of types. This allows our methods to be computationally more tractable.

To the best of our knowledge, none of these related works directly address the theoretical questions considered in this paper. While our results apply to [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011, Carmel and Markovitch, 1999], we believe they could be generalised to account for some of the other related works as well. This includes the methodology described in Section 5.

3 PRELIMINARIES

3.1 MODEL

Our analysis is based on the stochastic Bayesian game [Albrecht and Ramamoorthy, 2013]:

Definition 1. A *stochastic Bayesian game* (SBG) consists of

- discrete state space S with initial state $s^0 \in S$ and terminal states $\bar{S} \subset S$
- players $N = \{1, ..., n\}$ and for each $i \in N$:
  - set of actions A_i (where $A = A_1 \times ... \times A_n$)
  - type space Θ_i (where $\Theta = \Theta_1 \times ... \times \Theta_n$)
  - payoff function $u_i : S \times A \times \Theta_i \to \mathbb{R}$
  - strategy $\pi_i : \mathbb{H} \times A_i \times \Theta_i \to [0, 1]$
- state transition function $T : S \times A \times S \to [0, 1]$
- type distribution $\Delta : \Theta \to [0, 1]$

where $\mathbb{H}$ contains all *histories* $H^t = \langle s^0, a^0, s^1, a^1, ..., s^t \rangle$ with $t \geq 0$, $(s^\tau, a^\tau) \in S \times A$ for $0 \leq \tau < t$, and $s^t \in S$.

Definition 2. A SBG starts at time $t = 0$ in state s^0:

1. In state s^t, the types $\theta_1^t, ..., \theta_n^t$ are sampled from Θ with probability $\Delta(\theta_1^t, ..., \theta_n^t)$, and each player i is informed only about its own type θ_i^t.

2. Based on the history H^t, each player i chooses an action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t, \theta_i^t)$, resulting in the joint action $a^t = (a_1^t, ..., a_n^t)$.

3. The game transitions into a successor state $s^{t+1} \in S$ with probability $T(s^t, a^t, s^{t+1})$, and each player i receives an individual payoff given by $u_i(s^t, a^t, \theta_i^t)$.

This process is repeated until a terminal state $s^t \in \bar{S}$ is reached, after which the game stops.

3.2 ASSUMPTIONS

We make the following general assumptions in our analysis:

Assumption 1. We control player i, by which we mean that we choose the strategies π_i (using HBA). Hence, player i has only one type, θ_i, which is known to us.

We sometimes omit θ_i in u_i and π_i for brevity, and we use j and $-i$ to refer to the other players (e.g. $A_{-i} = \times_{j \neq i} A_j$).

Assumption 2. Given a SBG Γ, we assume that all elements of Γ are known except for the type spaces Θ_j and the type distribution Δ, which are *latent variables*.

Assumption 3. We assume *full observability* of states and actions. That is, we are always informed of the current history H^t before making a decision.

Assumption 4. For any type θ_j and history H^t, there exists a *unique* sequence $(\chi_{a_j})_{a_j \in A_j}$ such that $\pi_j(H^t, a_j, \theta_j) = \chi_{a_j}$ for all $a_j \in A_j$.

We refer to this as *external* randomisation and to the opposite (when there is no unique χ_{a_j}) as *internal* randomisation. Technically, Assumption 4 is implied by the fact that π_j is a function, which means that any input is mapped to exactly one output. However, in practice this can be violated if randomisation is used "inside" a type implementation, hence it is worth stating it explicitly. Nonetheless, it can be shown that under full observability, external randomisation is equivalent to internal randomisation. Hence, Assumption 4 does not limit the types we can represent.

Example 1. Let there be two actions, A and B, and let the expected payoffs for agent i be $E(A) > E(B)$. The agent uses ϵ-greedy action selection [Sutton and Barto, 1998] with $\epsilon > 0$. If agent i randomises *externally*, then the strategy π_i will assign action probabilities $\langle 1 - \epsilon/2, \epsilon/2 \rangle$. If the agent randomises *internally*, then with probability ϵ it will assign probabilities $\langle 0.5, 0.5 \rangle$ and with probability $1 - \epsilon$ it will assign $\langle 1, 0 \rangle$, which is equivalent to external randomisation.

3.3 ALGORITHM

Algorithm 1 gives a formal definition of HBA (based on [Albrecht and Ramamoorthy, 2013]) which is the central algorithm in this analysis. (Section 1 provides an informal description.) Throughout this paper, we will use Θ_j^* and Pr_j, respectively, to denote the user-defined type space and posterior for player j, where $\text{Pr}_j(\theta_j^* | H^t)$ is the probability that player j has type $\theta_j^* \in \Theta_j^*$ after history H^t. Furthermore, we will use Pr to denote the *combined* posterior, with $\text{Pr}(\theta_{-i}^* | H^t) = \prod_{j \neq i} \text{Pr}_j(\theta_j^* | H^t)$, and we sometimes refer to this simply as *the posterior*.

Note that the likelihood L in (1) is unspecified at this point. We will consider two variants for L in Section 4. The prior probabilities $P_j(\theta_j^*)$ in (1) can be used to specify prior beliefs about the distribution of types. It is convenient to specify $\text{Pr}_j(\theta_j^* | H^t) = P_j(\theta_j^*)$ for $t = 0$. Finally, note that (2)/(3) define an infinite regress. In practice, this may be implemented using stochastic sampling (e.g. as in [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011]) or by terminating the regress after some finite amount of time. In this analysis, we assume that (2)/(3) are implemented as given.

Algorithm 1 Harsanyi-Bellman Ad Hoc Coordination (HBA) [Albrecht and Ramamoorthy, 2013]

Input: SBG Γ, player i, user-defined type spaces Θ_j^*, history H^t, discount factor $0 \leq \gamma \leq 1$

Output: Action probabilities $\pi_i(H^t, a_i)$

1. For each $j \neq i$ and $\theta_j^* \in \Theta_j^*$, compute posterior probability

$$\text{Pr}_j(\theta_j^* | H^t) = \frac{L(H^t | \theta_j^*)\, P_j(\theta_j^*)}{\sum_{\hat{\theta}_j^* \in \Theta_j^*} L(H^t | \hat{\theta}_j^*)\, P_j(\hat{\theta}_j^*)} \quad (1)$$

2. For each $a_i \in A_i$, compute expected payoff $E_{s^t}^{a_i}(H^t)$ with

$$E_s^{a_i}(\hat{H}) = \sum_{\theta_{-i}^* \in \Theta_{-i}^*} \text{Pr}(\theta_{-i}^* | H^t) \sum_{a_{-i} \in A_{-i}} Q_s^{a_{i,-i}}(\hat{H}) \prod_{j \neq i} \pi_j(\hat{H}, a_j, \theta_j^*) \quad (2)$$

$$Q_s^a(\hat{H}) = \sum_{s' \in S} T(s, a, s') \left[u_i(s, a) + \gamma \max_{a_i} E_{s'}^{a_i}\left(\langle \hat{H}, a, s' \rangle\right) \right] \quad (3)$$

where $\text{Pr}(\theta_{-i}^* | H^t) = \prod_{j \neq i} \text{Pr}_j(\theta_j^* | H^t)$ and $a_{i,-i} \triangleq (a_i, a_{-i})$

3. Distribute $\pi_i(H^t, \cdot)$ uniformly over $\arg\max_{a_i} E_{s^t}^{a_i}(H^t)$

---

4 LEARNING THE TYPE DISTRIBUTION

This section is concerned with convergence and correctness properties of the posterior. The theorems in this section tell us if and under what conditions HBA will learn the type distribution of the game. As can be seen in Algorithm 1, this is important since the accuracy of the expected payoffs (2) depends crucially on the accuracy of the posterior (1).

However, for this to be a well-posed learning problem, we have to assume that the posterior Pr can refer to the same elements as the type distribution Δ. Therefore, the results in this section pertain to a weaker form of *ad hoc coordination* [Albrecht and Ramamoorthy, 2013] in which the user knows that the latent type space Θ_j must be a subset of the user-defined type space Θ_j^*. Formally, we assume:

Assumption 5. $\forall j \neq i : \Theta_j \subseteq \Theta_j^*$

Based on this assumption, we simplify the notation in this section by dropping the * in θ_j^* and Θ_j^*. The general case in which Assumption 5 does *not* hold is addressed in Section 5.

We consider two kinds of type distributions:

Definition 3. A type distribution Δ is called *pure* if there is $\theta \in \Theta$ such that $\Delta(\theta) = 1$. A type distribution is called *mixed* if it is not pure.

Pure type distributions can be used to model the fact that each player has a fixed type throughout the game, e.g. as in [Barrett et al., 2011]. Mixed type distributions, on the other hand, can be used to model randomly changing types. This

was shown in [Albrecht and Ramamoorthy, 2013], where a mixed type distribution was used to model defective agents and human behaviour.

4.1 PRODUCT POSTERIOR

We first consider the product posterior:

Definition 4. The *product posterior* is defined as (1) with

$$L(H^t|\theta_j) = \prod_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j) \qquad (4)$$

This is the standard posterior formulation used in Bayesian games (e.g. [Kalai and Lehrer, 1993]) and was used in [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011].

It can be shown that, under a pure type distribution and if HBA does not a priori rule out any of the types in Θ_j^*, then it will learn to make correct future predictions. Let H^∞ be an infinite history with prefix H^τ, and denote by $P_\Delta(H^\tau, H^\infty)$ and $P_{\mathrm{Pr}}(H^\tau, H^\infty)$, respectively, the *true* probability (based on Δ) and the probability assigned by HBA (based on Pr) that H^τ will continue as prescribed by H^∞.

Theorem 1. Let Γ be a SBG with a pure type distribution Δ. If HBA uses a product posterior and if the prior probabilities P_j are positive ($\forall \theta_j^* \in \Theta_j^* : P_j(\theta_j^*) > 0$), then:
for any $\epsilon > 0$, there is a time t from which ($\tau \geq t$)
$P_{\mathrm{Pr}}(H^\tau, H^\infty)(1-\epsilon) \leq P_\Delta(H^\tau, H^\infty) \leq (1+\epsilon)P_{\mathrm{Pr}}(H^\tau, H^\infty)$
for all H^∞ with $P_\Delta(H^\tau, H^\infty) > 0$.

Proof. The proof extends the convergence result of [Kalai and Lehrer, 1993]. A full proof is provided in the appendix document [Albrecht and Ramamoorthy, 2014]. □

Unfortunately, there is a subtle but important asymmetry between making correct future predictions and knowing the true type distribution: while the latter implies the former, examples can be created to show that the reverse is not true in general. Therefore, while HBA is guaranteed to make correct future predictions after some time, it is not guaranteed to learn the type distribution of the game.

Note that Theorem 1 pertains to pure type distributions only. The following example shows that the product posterior may fail in SBGs with mixed type distributions:

Example 2. Consider a SBG with two players. Player 1 is controlled by HBA using a product posterior while player 2 has two types, θ_A and θ_B, which are assigned by a mixed type distribution Δ with $\Delta(\theta_A) = \Delta(\theta_B) = 0.5$. The type θ_A always chooses action A while θ_B always chooses action B. In this case, there will be a time t after which both types have been assigned at least once, and so both actions A and B have been played at least once by player 2. This means that from time t and all subsequent times $\tau \geq t$, we have $\mathrm{Pr}_2(\theta_A|H^\tau) = \mathrm{Pr}_2(\theta_B|H^\tau) = 0$ (that is, Pr_2 is undefined), and HBA will fail to make correct future predictions.

4.2 SUM POSTERIOR

We now consider the sum posterior:

Definition 5. The *sum posterior* is defined as (1) with

$$L(H^t|\theta_j) = \sum_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j) \qquad (5)$$

The sum posterior was introduced in [Albrecht and Ramamoorthy, 2013] to allow HBA to recognise changed types. In other words, the purpose of the sum posterior is to learn mixed type distributions. It is easy to see that a sum posterior would indeed learn the mixed type distribution in Example 2. However, we now give an example to show that, without additional requirements, the sum posterior does not necessarily learn any (pure or mixed) type distribution:

Example 3. Consider a SBG with two players. Player 1 is controlled by HBA using a sum posterior while player 2 has two types, θ_A and θ_{AB}, which are assigned by a pure type distribution Δ with $\Delta(\theta_A) = 1$. The type θ_A always chooses action A while θ_{AB} chooses actions A and B with equal probability. While product posterior converges to the correct probabilities Δ, the sum posterior converges to probabilities $\langle \frac{2}{3}, \frac{1}{3} \rangle$, which is incorrect.

Note that this example can be readily modified to use a mixed type distribution, with similar results. Therefore, we conclude that, in general, the sum posterior does not necessarily learn any type distribution.

Under what condition is the sum posterior guaranteed to learn the true type distribution? Consider the following two quantities, which can be computed from a given history H^t:

Definition 6. The *average overlap* of player j in H^t is

$$\mathrm{AO}_j(H^t) = \frac{1}{t} \sum_{\tau=0}^{t-1} \left[|\Lambda_j^\tau| \geq 2\right]_1 \sum_{\theta_j \in \Theta_j} \pi_j(H^\tau, a_j^\tau, \theta_j) |\Theta_j|^{-1}$$

$$\Lambda_j^\tau = \left\{\theta_j \in \Theta_j \mid \pi_j(H^\tau, a_j^\tau, \theta_j) > 0\right\} \qquad (6)$$

where $[b]_1 = 1$ if b is true, else 0.

Definition 7. The *average stochasticity* of player j in H^t is

$$\mathrm{AS}_j(H^t) = \frac{1}{t} \sum_{\tau=0}^{t-1} |\Theta_j|^{-1} \sum_{\theta_j \in \Theta_j} \frac{1 - \pi_j(H^\tau, \hat{a}_j^\tau, \theta_j)}{1 - |A_j|^{-1}} \qquad (7)$$

where $\hat{a}_j^\tau \in \arg\max_{a_j} \pi_j(H^\tau, a_j, \theta_j)$.

Both quantities are bounded by 0 and 1. The average overlap describes the similarity of the types, where $\mathrm{AO}_j(H^t) = 0$ means that player j's types (on average) never chose the same action in history H^t, whereas $\mathrm{AO}_j(H^t) = 1$ means that they behaved identically. The average stochasticity describes the uncertainty of the types, where $\mathrm{AS}_j(H^t) = 0$ means that player j's types (on average) were fully deterministic in the action choices in history H^t, whereas

$AS_j(H^t) = 1$ means that they chose actions randomly with uniform probability.

We can show that, if the average overlap and stochasticity of player j converge to zero as $t \to \infty$, then the sum posterior is guaranteed to learn any pure or mixed type distribution:

Theorem 2. Let Γ be a SBG with a pure or mixed type distribution Δ. If HBA uses a sum posterior, then, for $t \to \infty$: If $AO_j(H^t) = 0$ and $AS_j(H^t) = 0$ for all players $j \neq i$, then $\Pr(\theta_{-i}|H^t) = \Delta(\theta_{-i})$ for all $\theta_{-i} \in \Theta_{-i}$.

Proof. Throughout this proof, let $t \to \infty$. The sum posterior is defined as (1) where L is defined as (5). Given the definition of L, both the numerator and the denominator in (1) may be infinite. We invoke L'Hôpital's rule which states that, in such cases, the quotient $\frac{u(t)}{v(t)}$ is equal to the quotient $\frac{u'(t)}{v'(t)}$ of the respective derivatives with respect to t. The derivative of L with respect to t is the average growth per time step, which in general may depend on the history H^t of states and actions. The average growth of L is

$$L'(H^t|\theta_j) = \sum_{a_j \in A_j} F(a_j|H^t)\, \pi_j(H^t, a_j, \theta_j) \quad (8)$$

where

$$F(a_j|H^t) = \sum_{\theta_j \in \Theta_j} \Delta(\theta_j)\, \pi_j(H^t, a_j, \theta_j) \quad (9)$$

is the probability of action a_j after history H^t, with $\Delta(\theta_j)$ being the marginal probability that player j is assigned type θ_j. As we will see shortly, we can make an asymptotic growth prediction irrespective of H^t. Given that $AO_j(H^t) = 0$, we can infer that whenever $\pi_j(H^t, a_j, \theta_j) > 0$ for action a_j and type θ_j, then $\pi_j(H^t, a_j, \theta'_j) = 0$ for all other types $\theta'_j \neq \theta_j$. Therefore, we can write (8) as

$$L'(H^t|\theta_j) = \Delta(\theta_j) \sum_{a_j \in A_j} \pi_j(H^t, a_j, \theta_j)^2 \quad (10)$$

Next, given that $AS_j(H^t) = 0$, we know that there exists an action a_j such that $\pi_j(H^t, a_j, \theta_j) = 1$, and therefore we can conclude that $L'(H^t|\theta_j) = \Delta(\theta_j)$. This shows that the history H^t is irrelevant to the asymptotic growth rate of L. Finally, since $\sum_{\theta_j \in \Theta_j} \Delta(\theta_j) = 1$, we know that the denominator in (1) will be 1, and we can ultimately conclude that $\Pr_j(\theta_j|H^t) = \Delta(\theta_j)$. $\square$

Theorem 2 explains why the sum posterior converges to the correct type distribution in Example 2. Since the types θ_A and θ_B always choose different actions and are completely deterministic (i.e. the average overlap and stochasticity are always zero), the sum posterior is guaranteed to converge to the type distribution. On the other hand, in Example 3 the types θ_A and θ_{AB} produce an overlap whenever action A is chosen, and θ_{AB} is completely random. Therefore, the
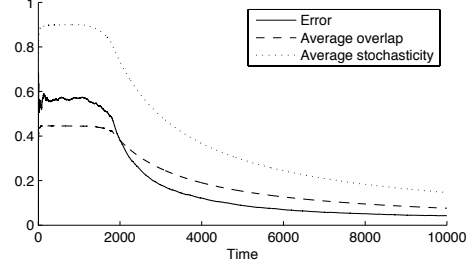

Figure 1: Example run in random SBG with 2 players, 10 actions, and 100 states. Player j has 3 reinforcement learning types with ϵ-greedy action selection (decreasing linearly from $\epsilon = 0.7$ at $t = 1000$, to $\epsilon = 0$ at $t = 2000$). The error at time t is $\sum_{\theta_j} |\Pr_j(\theta_j|H^t) - \Delta(\theta_j)|$ where $\Pr_j$ is the sum posterior.

average overlap and stochasticity are always positive, and an incorrect type distribution was learned.

The assumptions made in Theorem 2, namely that the average overlap and stochasticity converge to zero, require practical justification. First of all, it is important to note that it is only required that these converge to zero *on average* as $t \to \infty$. This means that in the beginning there may be arbitrary overlap and stochasticity, as long as these go to zero as the game proceeds. In fact, with respect to stochasticity, this is precisely how the exploration-exploitation dilemma [Sutton and Barto, 1998] is solved in practice: In the early stages, the agent randomises deliberately over its actions in order to obtain more information about the environment (*exploration*) while, as the game proceeds, the agent becomes gradually more deterministic in its action choices so as to maximise its payoffs (*exploitation*). Typical mechanisms which implement this are ϵ-greedy and Softmax/Boltzmann exploration [Sutton and Barto, 1998]. Figure 1 demonstrates this in a SBG in which player j has 3 reinforcement learning types. The payoffs for the types were such that the average overlap would eventually go to zero.

Regarding the average overlap converging to zero, we believe that this is a property which should be guaranteed *by design*, for the following reason: If the user-defined type space Θ_j^* is such that there is a constantly high average overlap, then this means that the types $\theta_j^* \in \Theta_j^*$ are in effect very similar. However, types which are very similar are likely to produce very similar trajectories in the planning step of HBA (cf. $\hat{H}$ in (2)) and, therefore, constitute redundancy in both time and space. Therefore, we believe it is advisable to use type spaces which have low average overlap.

4.3 CORRELATED POSTERIOR

An implicit assumption in the definition of (1) is that the type distribution Δ can be represented as a product of n independent factors (one for each player), so that $\Delta(\theta) = \prod_j \Delta_j(\theta_j)$. Therefore, since the sum posterior is in the form of (1), it is in fact only guaranteed to learn *independent* type distributions. This is opposed to *correlated*

type distributions, which cannot be represented as a product of n independent factors. Correlated type distributions can be used to specify constraints on type combinations, such as "player j can only have type θ_j if player k has type θ_k". The following example demonstrates how the sum posterior fails to converge to a correlated type distribution:

Example 4. Consider a SBG with 3 players. Player 1 is controlled by HBA using a sum posterior. Players 2 and 3 each have two types, θ_A and θ_B, which are defined as in Example 2. The type distribution Δ chooses types with probabilities $\Delta(\theta_A, \theta_B) = \Delta(\theta_B, \theta_A) = 0.5$ and $\Delta(\theta_A, \theta_A) = \Delta(\theta_B, \theta_B) = 0$. In other words, player 2 can never have the same type as player 3. From the perspective of HBA, each type (and hence action) is chosen with equal probability for both players. Thus, despite the fact that there is zero overlap and stochasticity, the sum posterior will eventually assign probability 0.25 to all constellations of types, which is incorrect. This means that HBA fails to recognise that the other players never choose the same action.

In this section, we propose a new posterior which can learn any correlated type distribution:

Definition 8. The *correlated posterior* is defined as

$$\Pr(\theta_{-i}|H^t) = \eta \, P(\theta_{-i}) \sum_{\tau=0}^{t-1} \prod_{\theta_j \in \theta_{-i}} \pi_j(H^\tau, a_j^\tau, \theta_j) \quad (11)$$

where P specifies prior probabilities (or beliefs) over Θ_{-i} (analogous to P_j) and η is a normalisation constant.

The correlated posterior is closely related to the sum posterior. In fact, in converges to the true type distribution under the same conditions as the sum posterior:

Theorem 3. Let Γ be a SBG with a *correlated* type distribution Δ. If HBA uses the correlated posterior, then, for $t \to \infty$: If $AO_j(H^t) = 0$ and $AS_j(H^t) = 0$ for all players $j \neq i$, then $\Pr(\theta_{-i}|H^t) = \Delta(\theta_{-i})$ for all $\theta_{-i} \in \Theta_{-i}$.

Proof. Proof is analogous to proof of Theorem 2. $\square$

It is easy to see that the correlated posterior would learn the correct type distribution in Example 4. Note that, since it is guaranteed to learn any correlated type distribution, it is also guaranteed to learn any independent type distribution. Therefore, the correlated posterior would also learn the correct type distribution in Example 2. This means that the correlated posterior is *complete* in the sense that it covers the entire spectrum of pure/mixed and independent/correlated type distributions. However, this completeness comes at a higher computational complexity. While the sum posterior is in $O(n \max_j |\Theta_j|)$ time and space, the correlated posterior is in $O(\max_j |\Theta_j|^n)$ time and space. In practice, however, the time complexity can be reduced drastically by computing the probabilities $\pi_j(H^\tau, a_j^\tau, \theta_j)$ only once for each j and $\theta_j \in \Theta_j$ (as in the sum posterior), and then reusing them in subsequent computations.

5 INACCURATE TYPE SPACES

Each user-defined type θ_j^* in Θ_j^* is a hypothesis by the user regarding how player j might behave. Therefore, Θ_j^* may be *inaccurate* in the sense that none of the types therein accurately predict the observed behaviour of player j. This is demonstrated in the following example:

Example 5. Consider a SBG with two players and actions L and R. Player 1 is controlled by HBA while player 2 has a single type, θ_{LR}, which chooses L,R,L,R, etc. HBA is provided with $\Theta_j^* = \{\theta_R^*, \theta_{LRR}^*\}$, where θ_R^* always chooses R while θ_{LRR}^* chooses L,R,R,L,R,R etc. Both user-defined types are inaccurate in the sense that they predict player 2's actions in only $\approx 50\%$ of the game.

Two important theoretical questions in this context are how closely the user-defined type spaces Θ_j^* have to approximate the real type spaces Θ_j in order for HBA to be able to *(1)* solve the task (i.e. bring the SBG into a terminal state), and *(2)* achieve maximum payoffs. These questions are closely related to the notions of *flexibility* and *efficiency* [Albrecht and Ramamoorthy, 2013] which, respectively, correspond to the probability of termination and the average payoff per time step. In this section, we are primarily concerned with question 1, and we are concerned with question 2 only in so far as that we want to solve the task in minimal time. (Since reducing the time until termination will increase the average payoff per time step, i.e. increase efficiency.) This focus is formally captured by the following assumption, which we make throughout this section:

Assumption 6. Let player i be controlled by HBA, then $u_i(s, a, \theta_i) = 1$ iff. $s \in \bar{S}$, else 0.

Assumption 6 specifies that we are only interested in reaching a terminal state, since this is the only way to obtain a none-zero payoff. In our analysis, we consider discount factors γ (cf. Algorithm 1) with $\gamma = 1$ and $\gamma < 1$. While all our results hold for both cases, there is an important distinction: If $\gamma = 1$, then the expected payoffs (2) correspond to the actual probability that the following state can lead to (or is) a terminal state (we call this the *success rate*), whereas this is not necessarily the case if $\gamma < 1$. This is since $\gamma < 1$ tends to prefer shorter paths, which means that actions with lower success rates may be preferred if they lead to faster termination. Therefore, if $\gamma = 1$ then HBA is solely interested in termination, and if $\gamma < 1$ then it is interested in *fast* termination, where lower γ prefers faster termination.

5.1 METHODOLOGY OF ANALYSIS

Given a SBG Γ, we define the *ideal process*, X, as the process induced by Γ in which player i is controlled by HBA and in which HBA always knows the current and all future types of all players. Then, given a posterior Pr and user-defined type spaces Θ_j^* for all $j \neq i$, we define the *user process*, Y, as the process induced by Γ in which player i

is controlled by HBA (same as in X) and in which HBA uses Pr and Θ_j^* in the usual way. Thus, the only difference between X and Y is that X can always predict the player types whereas Y approximates this knowledge through Pr and Θ_j^*. We write $E_{s^t}^{a_i}(H^t|C)$ to denote the expected payoff (as defined by (2)) of action a_i in state s^t after history H^t, in process $C \in \{X, Y\}$.

The idea is that X constitutes the ideal solution in the sense that $E_{s^t}^{a_i}(H^t|X)$ corresponds to the *actual* expected payoff, which means that HBA chooses the truly best-possible actions in X. This is opposed to $E_{s^t}^{a_i}(H^t|Y)$, which is merely the *estimated* expected payoff based on Pr and Θ_j^*, so that HBA may choose suboptimal actions in Y. The methodology of our analysis is to specify what relation Y must have to X to satisfy certain guarantees for termination.

We specify such guarantees in PCTL [Hansson and Jonsson, 1994], a probabilistic modal logic which also allows for the specification of time constraints. PCTL expressions are interpreted over infinite histories in labelled transition systems with atomic propositions (i.e. Kripke structures). In order to interpret PCTL expressions over X and Y, we make the following modifications without loss of generality: Firstly, any terminal state $\bar{s} \in \bar{S}$ is an *absorbing* state, meaning that if a process is in $\bar{s}$, then the next state will be $\bar{s}$ with probability 1 and all players receive a zero payoff. Secondly, we introduce the atomic proposition `term` and label each terminal state with it, so that `term` is true in s if and only if $s \in \bar{S}$.

We will use the following two PCTL expressions:

$$F_{\succ p}^{\leq t}\texttt{term}, \ F_{\succ p}^{< \infty}\texttt{term}$$

where $t \in \mathbb{N}$, $p \in [0, 1]$, and $\succ \in \{>, \geq\}$.

$F_{\succ p}^{\leq t}\texttt{term}$ specifies that, given a state s, with a probability of $\succ p$ a state s' will be reached from s within t time steps such that s' satisfies `term`. The semantics of $F_{\succ p}^{< \infty}\texttt{term}$ are similar except that s' will be reached in arbitrary but finite time. We write $s \models_C \phi$ to say that a state s satisfies the PCTL expression ϕ in process $C \in \{X, Y\}$.

5.2 CRITICAL TYPE SPACES

In the following section, we sometimes assume that the user-defined type spaces Θ_j^* are *uncritical*:

Definition 9. The user-defined type spaces Θ_j^* are *critical* if there is $S^c \subseteq S \setminus \bar{S}$ which satisfies:

1. For each $H^t \in \mathbb{H}$ with $s^t \in S^c$, there is $a_i \in A_i$ such that $E_{s^t}^{a_i}(H^t|Y) > 0$ and $E_{s^t}^{a_i}(H^t|X) > 0$

2. There is a positive probability that Y may eventually get into a state $s^c \in S^c$ from the initial state s^0

3. If Y is in a state in S^c, then with probability 1 it will always be in a state in S^c (i.e. it will never leave S^c)

We say Θ_j^* are *uncritical* if they are not critical.

Intuitively, critical type spaces have the potential to lead HBA into a state space in which it *believes* it chooses the right actions to solve the task, while other actions are *actually* required to solve the task. The only effect that its actions have is to induce an infinite cycle, due to a critical inconsistency between the user-defined and true type spaces. The following example demonstrates this:

Example 6. Recall Example 5 and let the task be to choose the same action as player j. Then, Θ_j^* is uncritical because HBA will always solve the task at $t = 1$, regardless of the posterior and despite the fact that Θ_j^* is inaccurate. Now, assume that $\Theta_j^* = \{\theta_{RL}^*\}$ where θ_{RL}^* chooses actions R,L,R,L etc. Then, Θ_j^* is critical since HBA will always choose the opposite action of player j, thinking that it would solve the task, when a different action would actually solve it.

A practical way to ensure that the type spaces Θ_j^* are (eventually) uncritical is to include methods for opponent modelling in each Θ_j^* (e.g. as in [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011]). If the opponent models are guaranteed to learn the correct behaviours, then the type spaces Θ_j^* are guaranteed to become uncritical. In Example 6, any standard modelling method would eventually learn that the true strategy of player j is θ_{LR}. As the model becomes more accurate, the posterior gradually shifts towards it and eventually allows HBA to take the right action.

5.3 TERMINATION GUARANTEES

Our first guarantee states that if X has a positive probability of solving the task, then so does Y:

Property 1. $s^0 \models_X F_{>0}^{< \infty}\texttt{term} \Rightarrow s^0 \models_Y F_{>0}^{< \infty}\texttt{term}$

We can show that Property 1 holds if the user-defined type spaces Θ_j^* are uncritical and if Y only chooses actions for player i with positive expected payoff in X.

Let $\mathbb{A}(H^t|C)$ denote the set of actions that process C may choose from in state s^t after history H^t, i.e. $\mathbb{A}(H^t|C) = \arg \max_{a_i} E_{s^t}^{a_i}(H^t|C)$ (cf. step 3 in Algorithm 1).

Theorem 4. Property 1 holds if Θ_j^* are uncritical and

$$\forall H^t \in \mathbb{H} \ \forall a_i \in \mathbb{A}(H^t|Y) : E_{s^t}^{a_i}(H^t|X) > 0 \quad (12)$$

Proof. Assume $s^0 \models_X F_{>0}^{< \infty}\texttt{term}$. Then, we know that X chooses actions a_i which *may* lead into a state s' such that $s' \models_X F_{>0}^{< \infty}\texttt{term}$, and the same holds for all such states s'. Now, given (12) it is tempting to infer the same result for Y, since Y only chooses actions a_i which have positive expected payoff in X and, therefore, could truly lead into a terminal state. However, (12) alone is not sufficient to infer $s' \models_Y F_{>0}^{< \infty}\texttt{term}$ because of the special case in which Y chooses actions a_i such that $E_{s^t}^{a_i}(H^t|X) > 0$ but without ever reaching a terminal state. This is why we require that the user-defined type spaces Θ_j^* are uncritical, which prevents this special case. Thus, we can infer that $s' \models_Y F_{>0}^{< \infty}\texttt{term}$, and hence Property 1 holds. $\square$

The second guarantee states that if X always solves the task, then so does Y:

Property 2. $s^0 \models_X F_{\geq 1}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq 1}^{<\infty}\texttt{term}$

We can show that Property 2 holds if the user-defined type spaces Θ_j^* are uncritical and if Y only chooses actions for player i which lead to states into which X may get as well.

Let $\mu(H^t, s|C)$ be the probability that process C transitions into state s from state s^t after history H^t, i.e. $\mu(H^t, s|C) = \frac{1}{|\mathbb{A}|}\sum_{a_i\in\mathbb{A}}\sum_{a_{-i}} T(s^t, \langle a_i, a_{-i}\rangle, s)\prod_j \pi_j(H^t, a_j, \theta_j^t)$ with $\mathbb{A} \equiv \mathbb{A}(H^t|C)$, and let $\mu(H^t, S'|C) = \sum_{s\in S'}\mu(H^t, s|C)$ for $S' \subset S$.

Theorem 5. Property 2 holds if Θ_j^* are uncritical and

$$\forall H^t\in\mathbb{H} \,\forall s\in S : \mu(H^t, s|Y) > 0 \Rightarrow \mu(H^t, s|X) > 0 \tag{13}$$

Proof. The fact that $s^0 \models_X F_{\geq 1}^{<\infty}\texttt{term}$ means that, throughout the process, X only transitions into states s with $s \models_X F_{\geq 1}^{<\infty}\texttt{term}$. As before, it is tempting to infer the same result for Y based on (13), since it only transitions into states which have maximum success rate in X. However, (13) alone is not sufficient since Y may choose actions such that (13) holds true but Y will never reach a terminal state. Nevertheless, since the user-defined type spaces Θ_j^* are uncritical, we know that this special case will not occur, and hence Property 2 holds. $\square$

We note that, in both Properties 1 and 2, the reverse direction holds true regardless of Theorems 4 and 5. Furthermore, we can combine the requirements of Theorems 4 and 5 to ensure that both properties hold.

The next guarantee subsumes the previous guarantees by stating that X and Y have the same minimum probability of solving the task:

Property 3. $s^0 \models_X F_{\geq p}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq p}^{<\infty}\texttt{term}$

We can show that Property 3 holds if the user-defined type spaces Θ_j^* are uncritical and if Y only chooses actions for player i which X might have chosen as well.

Let $R(a_i, H^t|C)$ be the *success rate* of action a_i, formally $R(a_i, H^t|C) = E_{s^t}^{a_i}(H^t|C)$ with $\gamma = 1$ (so that it corresponds to the actual *probability* with which a_i may lead to termination in the future). Define $X_{\min}$ and $X_{\max}$ to be the processes which for each H^t choose actions $a_i \in \mathbb{A}(H^t|X)$ with, respectively, minimal and maximal success rate $R(a_i, H^t|X)$.

Theorem 6. If Θ_j^* are uncritical and

$$\forall H^t\in\mathbb{H} : \mathbb{A}(H^t|Y) \subseteq \mathbb{A}(H^t|X) \tag{14}$$

then

(i) for $\gamma = 1$: Proposition 3 holds in both directions

(ii) for $\gamma < 1$: $s^0 \models_X F_{\geq p}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq p'}^{<\infty}\texttt{term}$ with $p_{\min} \leq q \leq p_{\max}$ for $q \in \{p, p'\}$, where $p_{\min}$ and $p_{\max}$ are the highest probabilities such that $s^0 \models_{X_{\min}} F_{\geq p_{\min}}^{<\infty}\texttt{term}$ and $s^0 \models_{X_{\max}} F_{\geq p_{\max}}^{<\infty}\texttt{term}$.

Proof. (i): Since $\gamma = 1$, all actions $a_i \in \mathbb{A}(H^t|X)$ have the same success rate for a given H^t, and given (14) we know that Y's actions always have the same success rate as X's actions. Provided that the type spaces Θ_j^* are uncritical, we can conclude that Property 3 must hold, and for the same reasons the reverse direction must hold as well.

(ii): Since $\gamma < 1$, the actions $a_i \in \mathbb{A}(H^t|X)$ may have different success rates. The lowest and highest chances that X solves the task are precisely modelled by $X_{\min}$ and $X_{\max}$, and given (14) and the fact that Θ_j^* are uncritical, the same holds for Y. Therefore, we can infer the common bound $p_{\min} \leq \{p, p'\} \leq p_{\max}$ as defined in Theorem 6. $\square$

Properties 1 to 3 are *indefinite* in the sense that they make no restrictions on time requirements. Our fourth and final guarantee subsumes all previous guarantees and states that if there is a probability p such that X terminates *within* t time steps, then so does Y for the same p and t:

Property 4. $s^0 \models_X F_{\geq p}^{\leq t}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq p}^{\leq t}\texttt{term}$

We believe that Property 4 is an adequate criterion of optimality for the type spaces Θ_j^* since, if it holds, Θ_j^* must approximate Θ_j in a way which allows HBA to plan (almost) as accurately — in terms of solving the task — as the "ideal" HBA in X which always knows the true types.

What relation must Y have to X to satisfy Property 4? The fact that Y and X are processes over state transition systems means we can draw on methods from the model checking literature to answer this question. Specifically, we will use the concept of *probabilistic bisimulation* [Larsen and Skou, 1991], which we here define in the context of our work:

Definition 10. A *probabilistic bisimulation* between X and Y is an equivalence relation $B \subseteq S \times S$ such that

(i) $(s^0, s^0) \in B$

(ii) $s_X \models_X \texttt{term} \Leftrightarrow s_Y \models_Y \texttt{term}$ for all $(s_X, s_Y) \in B$

(iii) $\mu(H_X^t, \hat{S}|X) = \mu(H_Y^t, \hat{S}|Y)$ for any histories H_X^t, H_Y^t with $(s_X^t, s_Y^t) \in B$ and all equivalence classes $\hat{S}$ under B.

Intuitively, a probabilistic bisimulation states that X and Y do (on average) match each other's transitions. Our definition of probabilistic bisimulation is most general in that it does not require that transitions are matched by the same action or that related states satisfy the same atomic propositions other than termination. However, we do note that other definitions exist that make such additional requirements, and our results hold for each of these refinements.
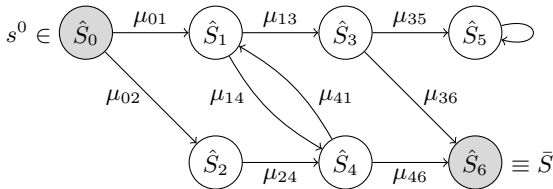
The main contribution of this section is to show that the

optimality criterion expressed by Property 4 holds in *both directions* if there is a probabilistic bisimulation between X and Y. Thus, we offer an alternative formal characterisation of optimality for the user-defined type spaces Θ_j^*:

Theorem 7. Property 4 holds in both directions if there is a probabilistic bisimulation between X and Y.

Proof. First of all, we note that, strictly speaking, the standard definitions of bisimulation (e.g. [Baier, 1996, Larsen and Skou, 1991]) assume the Markov property, which means that the next state of a process depends only on the current state of the process. In contrast, we consider the more general case in which the next state may depend on the history H^t of previous states and joint actions (since the player strategies π_j depend on H^t). However, one can always enforce the Markov property *by design*, i.e. by augmenting the state space S to account for the relevant factors of the past. In fact, we could postulate that the histories as a whole constitute the states of the system, i.e. $S = \mathbb{H}$. Therefore, to simplify the exposition, we assume the Markov property and we write $\mu(s, \hat{S}|C)$ to denote the cumulative probability that C transitions from state s into any state in $\hat{S}$.

Given the Markov property, the fact that B is an equivalence relation, and $\mu(s_X, \hat{S}|X) = \mu(s_Y, \hat{S}|Y)$ for $(s_X, s_Y) \in B$, we can represent the dynamics of X and Y in a common graph, such as the following one:

The nodes correspond to the equivalence classes under B. A directed edge from $\hat{S}_a$ to $\hat{S}_b$ specifies that there is a positive probability $\mu_{ab} = \mu(s_X, \hat{S}_b|X) = \mu(s_Y, \hat{S}_b|Y)$ that X and Y transition from states $s_X, s_Y \in \hat{S}_a$ to states $s_X', s_Y' \in \hat{S}_b$. Note that s_X, s_Y and s_X', s_Y' need not be equal but merely equivalent, i.e. $(s_X, s_Y) \in B$ and $(s_X', s_Y') \in B$. There is one node ($\hat{S}_0$) that contains the initial state s^0 and one node ($\hat{S}_6$) that contains all terminal states $\bar{S}$ and no other states. This is because once X and Y reach a terminal state they will always stay in it (i.e. $\mu(s, \bar{S}|X) = \mu(s, \bar{S}|Y) = 1$ for $s \in \bar{S}$) and since they are the only states that satisfy term. Thus, the graph starts in $\hat{S}_0$ and terminates (if at all) in $\hat{S}_6$.

Since the graph represents the dynamics of both X and Y, it is easy to see that Property 4 must hold in both directions. In particular, the probabilities that X and Y are in node $\hat{S}$ at time t are identical. One simply needs to add the probabilities of all directed paths of length t which end in $\hat{S}$ (provided that such paths exist), where the probability of a path is the product of the μ_{ab} along the path. Therefore,

X and Y terminate with equal probability, and on average within the same number of time steps. □

Some remarks to clarify the usefulness of this result: First of all, in contrast to Theorems 4 to 6, Theorem 7 does not explicitly require Θ_j^* to be uncritical. In fact, this is implicit in the definition of probabilistic bisimulation. Moreover, while the other theorems relate Y and X for identical histories H^t, Theorem 7 relates Y and X for *related* histories H_Y^t and H_X^t, making it more generally applicable. Finally, Theorem 7 has an important practical implication: it tells us that we can use efficient methods for model checking (e.g. [Baier, 1996, Larsen and Skou, 1991]) to verify optimality of Θ_j^*. In fact, it can be shown that for Property 4 to hold (albeit not in the other direction) it suffices that Y be a *probabilistic simulation* [Baier, 1996] of X, which is a coarser preorder than probabilistic bisimulation. However, algorithms for checking probabilistic simulation (e.g. [Baier, 1996]) are computationally much more expensive (and fewer) than those for probabilistic bisimulation, hence their practical use is currently limited.

6 CONCLUSION

This paper complements works such as [Albrecht and Ramamoorthy, 2013, Barrett et al., 2011, Carmel and Markovitch, 1999] — with a focus on HBA due to its generality — by providing answers to two important theoretical questions: "Under what conditions does HBA learn the type distribution of the game?" and "How accurate must the user-defined type spaces be for HBA to solve its task?" With respect to the first question, we analyse the convergence properties of two existing posteriors and propose a new posterior which can learn correlated type distributions. This provides the user with formal reasons as to which posterior should be chosen for the problem at hand. With respect to the second question, we describe a methodology in which we analyse the requirements of several termination guarantees, and we provide a novel characterisation of optimality which is based on the notion of probabilistic bisimulation. This gives the user a formal yet practically useful criterion of what constitutes optimal type spaces. The results of this work improve our understanding of how a method such as HBA can be used to effectively solve agent interaction problems in which the behaviour of other agents is not a priori known.

There are several interesting directions for future work. For instance, it is unclear what effect the prior probabilities P_j have on the performance of HBA, and if a criterion for optimal P_j could be derived. Furthermore, since our convergence proofs in Section 4 are asymptotic, it would be interesting to know if useful finite-time error bounds exist. Finally, our analysis in Section 5 is general in the sense that it applies to any posterior. This could be refined by an analysis which commits to a specific posterior.

References

S. Albrecht and S. Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012.

S. Albrecht and S. Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 2013.

S. Albrecht and S. Ramamoorthy. On convergence and optimality of best-response learning with policy types in multiagent systems – Appendix, 2014. *http://rad.inf.ed.ac.uk/data/publications/2014/uai14app.pdf*.

D. Avrahami-Zilberbrand and G. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 2007.

C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *Proceedings of the 8th International Conference on Computer Aided Verification, Lecture Notes in Computer Science*, volume 1102, pages 38–49. Springer, 1996.

S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

E. Bonchek-Dokow, G. Kaminka, and C. Domshlak. Distinguishing between intentional and unintentional sequences of actions. In *Proceedings of the 9th International Conference on Cognitive Modeling*, 2009.

M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

D. Carmel and S. Markovitch. Exploration strategies for model-based learning in multi-agent systems: Exploration strategies. *Autonomous Agents and Multi-Agent Systems*, 2(2):141–172, 1999.

P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive POMDPs: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3):376–416, 2009.

I. Gilboa and D. Schmeidler. *A theory of case-based decisions*. Cambridge University Press, 2001.

P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24(1):49–79, 2005.

P. Gmytrasiewicz and E. Durfee. Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 3(4):319–350, 2000.

H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

E. Kalai and E. Lehrer. Rational learning leads to Nash equilibrium. *Econometrica*, pages 1019–1045, 1993.

K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.

R. Sutton and A. Barto. *Reinforcement learning: An introduction*. The MIT Press, 1998.