E-HBA: Using Action Policies for Expert Advice and Agent Typification

Stefano V. Albrecht
The University of Edinburgh
Edinburgh, United Kingdom
s.v.albrecht@sms.ed.ac.uk

Jacob W. Crandall
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates
jcrandall@masdar.ac.ae

Subramanian Ramamoorthy
The University of Edinburgh
Edinburgh, United Kingdom
s.ramamoorthy@ed.ac.uk

Abstract

Past research has studied two approaches to utilise predefined policy sets in repeated interactions: as *experts*, to dictate our own actions, and as *types*, to characterise the behaviour of other agents. In this work, we bring these complementary views together in the form of a novel meta-algorithm, called *Expert-HBA* (E-HBA), which can be applied to any expert algorithm that considers the average (or total) payoff an expert has yielded in the past. E-HBA gradually mixes the past payoff with a predicted future payoff, which is computed using the type-based characterisation. We present results from a comprehensive set of repeated matrix games, comparing the performance of several well-known expert algorithms with and without the aid of E-HBA. Our results show that E-HBA has the potential to significantly improve the performance of expert algorithms.

1 Introduction

Many multiagent applications require an agent to quickly learn how to interact effectively with previously unknown other agents. Important examples include electronic markets, adaptive user interfaces, and robotic elderly care. Learning effective policies from scratch in such applications (e.g. using reinforcement learning or opponent modelling) can be difficult because of the essentially unconstrained nature of the interaction problem, by which we mean that the other agents may, in principle, have any kind of behaviours.

One approach to make this problem more tractable is to assume that we have access to a set of policies, or *experts*, which make action recommendations based on the current interaction history. Such experts may be specified by a human user or generated automatically from the problem specification. The goal, then, is to find the best expert through some repeated exploration strategy. Several such strategies, or *expert algorithms*, have been defined, e.g. (Crandall 2014; de Farias and Megiddo 2004; Auer et al. 1995).

Another approach is to use such policies, then called *types*, to characterise the behaviour of other agents. In this approach, the observed actions of an agent are compared with the predictions of the types, resulting in a posterior distribution which describes the relative likelihood that the agent implements any of the types. This distribution can then be used in a planning procedure to compute best-response actions, e.g. (Albrecht and Ramamoorthy 2014; Barrett, Stone, and Kraus 2011; Carmel and Markovitch 1999).

While both approaches have been shown to be effective under various circumstances, they have certain limitations: Most expert algorithms are *reactive* in the sense that they rely heavily on the past performance of experts, typically in the form of the average payoff an expert yielded. However, this can be problematic when interacting with adaptive agents, in which case past performance is not necessarily a good indicator of future performance. On the other hand, type-based methods such as HBA (Albrecht and Ramamoorthy 2014) are *proactive* in that they make explicit predictions about future interactions and choose actions accordingly. However, such methods are not designed for situations in which none of the types account for the observed behaviour of an agent (i.e. all posterior probabilities are zero or undefined).

To illustrate these limitations, suppose we are playing the Prisoner's Dilemma game against a simple adaptive opponent that cooperates only if we cooperated in the past 4 rounds, otherwise it defects. Assume we are given two experts, C and D, where C always cooperates and D always defects. A typical expert algorithm would try both experts and may find that D has a higher average payoff than C, if C is not tried for a sufficient number of consecutive rounds. Thus, it would favour D, which in the long-run is the worse expert. On the other hand, if the opponent's behaviour was provided to HBA as a type, it would eventually learn the true type and play optimally against it (that is, provided that its planning horizon is deep enough). However, if the true type is unknown to HBA, then the posterior may become undefined in the worst case and HBA would play randomly.

In this paper, we propose to address these limitations by combining the two approaches. Specifically, we present a novel meta-algorithm, called *Expert-HBA* (E-HBA), which can be applied to any expert algorithm that considers the average (or total) payoffs the experts yielded when following their recommendations. E-HBA gradually mixes the past payoff an expert yielded with a predicted future payoff, which is computed using the type-based approach. The mixing is gradual in that the weight of the predicted payoff is proportional to the *confidence* E-HBA has in the correctness of its predictions (to be made precise shortly). We present results

from a comprehensive set of repeated matrix games, comparing the performance of several well-known expert algorithms with and without the aid of E-HBA. Our results show that, if the true (or a similar) type of the opponent is in E-HBA's set of types, then it can significantly improve the performance of the expert algorithm, while in all other cases it performs similarly to the original expert algorithm.

2 Related Work

A number of expert algorithms have been defined. Some of the more well-known ones include the 'Weighted Majorities' algorithm (Littlestone and Warmuth 1994), Hedge (Freund and Schapire 1995), the Exp-family (Auer et al. 1995), the UCB-family (Auer, Cesa-Bianchi, and Fischer 2002), EEE (de Farias and Megiddo 2004), and S (Karandikar et al. 1998).

E-HBA itself is not an expert algorithm. Rather, it is a meta-algorithm that can be applied to any expert algorithm which considers the average (or total) payoff an expert has yielded. This includes all of the above algorithms, to which E-HBA can be applied in a straight-forward manner.

Our work is closest in spirit to (Crandall 2014), who proposed a meta-algorithm that prunes the set of selectable experts to those which it considers most promising. This is done using a variant of aspiration learning, similar to (Karandikar et al. 1998). E-HBA can be combined with Crandall's meta-algorithm since they target different aspects in expert algorithms: E-HBA modifies the average payoffs of experts while Crandall's algorithm prunes the set of experts.

Most expert algorithms are evaluated in terms of *regret* (e.g. Foster and Vohra 1999), which is the difference between the received payoffs and the payoffs of the best expert against the observed actions. However, it has been argued that this view of regret may be inadequate in interactive settings with adaptive agents (Crandall 2014; Arora, Dekel, and Tewari 2012). In this work, we evaluate expert algorithms in terms of the average payoffs they achieved.

E-HBA is based on HBA (Albrecht and Ramamoorthy 2014; 2013), which is informally described in Section 1 and formally defined in Section 3.3. Related methods were studied by Barrett, Stone, and Kraus (2011) and Carmel and Markovitch (1999). We focus on HBA because it is a relatively simple and general method with well-known guarantees (Albrecht and Ramamoorthy 2014).

The methods used in I-POMDPs (Gmytrasiewicz and Doshi 2005) are closely related to HBA. However, as they are designed to handle the full generality of partially observable states, complex nested beliefs, and subjective optimality criteria, their solutions can be very hard to compute. We focus on HBA because we are not aware of any expert algorithm that was specifically designed to handle partially observable states. However, we believe that our work can be extended to more complex models such as I-POMDPs.

3 Preliminaries

This section introduces some basic notation and definitions which we will use in the remainder of the paper.

3.1 Model

To simplify the exposition, we focus on 2-player repeated matrix games. However, our definitions can be extended to more complex models such as stochastic Bayesian games (Albrecht and Ramamoorthy 2014) and I-POMDPs (Gmytrasiewicz and Doshi 2005).

We use i to refer to our player and j to refer to the other player. At each time t in the game, each player $k \in \{i, j\}$ independently chooses an action $a_k^t \in A_k$ and receives a payoff $u_k(a_i^t, a_j^t) \in \mathbb{R}$. This process is repeated for a finite number of rounds. We assume we know A_i, A_j, and u_i.

3.2 Expert Algorithms

Let Φ_i be a set of *experts* for player i. Each expert $\phi_i \in \Phi_i$ specifies an action policy for player i. We write $\pi_i(H^t, a_i, \phi_i)$ to denote the probability that expert ϕ_i chooses action a_i after the history $H^t = \langle (a_i^0, a_j^0), ..., (a_i^{t-1}, a_j^{t-1}) \rangle$ of previous joint actions. Furthermore, we use $\bar{U}$ to denote the vector of average payoffs the experts have yielded when following their recommendations (i.e. one entry for each expert).

For the purposes of this exposition, we abstractly define an *expert algorithm* as a function $f(\bar{U})$ which returns a probability distribution over the set Φ_i of experts. This distribution specifies what experts to choose at any given time. We note that many expert algorithms use auxiliary statistics, such as the number of times an expert has been chosen. However, in the interest of clarity, we omit such details here.

3.3 Harsanyi-Bellman Ad Hoc Coordination

Our meta-algorithm is based on *Harsanyi-Bellman Ad Hoc Coordination* (HBA) (Albrecht and Ramamoorthy 2014). HBA utilises a set Θ_j^* of hypothesised *types* which specify complete action policies for player j. The types are hypothetical in the sense that player j may or may not implement one of the types in Θ_j^*, but we do not a priori know this.

We write $\pi_j(H^t, a_j, \theta_j^*)$ to denote the probability that player j chooses action a_j if it is of type θ_j^*, given the history H^t of previous joint actions. HBA computes the posterior belief that player j is of type θ_j^*, given H^t, as

$$\Pr_j(\theta_j^* | H^t) = \eta \, P_j(\theta_j^*) \prod_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j^*) \qquad (1)$$

where η is a normalisation constant and $P_j(\theta_j^*)$ denotes the prior belief (e.g. uniform) that player j is of type θ_j^*.

Using the posterior $\Pr_j$, HBA chooses an action a_i which maximises the expected average payoff $\frac{1}{h} E_h^{a_i}(H^t)$, where

$$E_h^{a_i}(\hat{H}) = \sum_{\theta_j^* \in \Theta_j^*} \Pr_j(\theta_j^* | H^t) \sum_{a_j \in A_j} \pi_j(\hat{H}, a_j, \theta_j^*) \, Q_{h-1}^{(a_i, a_j)}(\hat{H})$$

$$Q_h^{(a_i,a_j)}(\hat{H}) = u_i(a_i, a_j) + \begin{cases} 0 \text{ if } h = 0, \text{ else} \\ \max_{a_i'} E_h^{a_i'}\Big(\langle \hat{H}, (a_i, a_j) \rangle\Big) \end{cases} \qquad (2)$$

and h specifies the depth of the planning horizon. Note that H^t is the current history while $\hat{H}$ is used to construct all future trajectories in the game.

4 Expert-HBA

Expert-HBA (E-HBA) is a meta-algorithm that can be applied to any expert algorithm which is in the general form of $f(\bar{U})$. A schematic of E-HBA is provided in Algorithm 1.

4.1 Future Payoffs of Experts

E-HBA mixes the vector $\bar{U}$ of *observed* (past) average payoffs of each expert with a vector U^* of *expected* (future) average payoffs for each expert. In order to compute U^*, E-HBA uses an adapted version of HBA.

Similar to HBA, E-HBA maintains a posterior distribution $\Pr_j$ over a set Θ_j^* of hypothesised types for player j, which can be obtained in the same way as the expert set Φ_i. This distribution is computed using (1). To obtain the vector U^*, E-HBA computes the expected average payoff of each expert $\phi_i \in \Phi_i$, using a modified version of (2) which redefines

$$Q_h^{(a_i,a_j)} = u_i(a_i,a_j) + \begin{cases} 0 \text{ if } h = 0, \text{ else} \\ \sum_{a_i'} \pi_i(\hat{H}, a_i', \phi_i)\, E_h^{a_i'}\Big(\langle \hat{H}, (a_i,a_j)\rangle\Big) \end{cases}. \tag{3}$$

The difference between (3) and the original definition is that (3) only follows the expert when expanding future trajectories in the game, whereas the original definition has to consider all possible trajectories in order to implement the $\max$-operator. Depending on the stochasticity of the experts (i.e. the number of actions with positive probability), this may mean that (3) can be computed more efficiently than the original definition. Note that, as with the original definition, (3) may be approximated efficiently using Monte-Carlo Tree Search, e.g. (Kocsis and Szepesvári 2006).

4.2 Mixing with Confidence

Given the vectors $\bar{U}$ and U^*, E-HBA executes the expert algorithm as $f\big((1-C^t)\bar{U} + C^t U^*\big)$, where $C^t \in [0,1]$ is the key mixing factor that gradually combines the expert and type methodologies. Intuitively, C^t can be interpreted as the *confidence* E-HBA has at time t in the correctness of U^*, such that $C^t = 1$ corresponds to absolute confidence and $C^t = 0$ corresponds to no confidence at all.

Let the true type of player j be denoted by θ_j^+, and assume for simplicity that we have a single hypothesised type θ_j^* (any pair $(\Pr_j, \Theta_j^*)$ can be represented as a single type). Then, C^t can be viewed as quantifying the *similarity* between θ_j^+ and θ_j^*. However, given that we only observe H^t and θ_j^* but not θ_j^+, this can be an extremely difficult task. Indeed, even if we knew the true type θ_j^+, it would by no means be clear how to best quantify a similarity between θ_j^+ and θ_j^*.

In this preliminary work, we define confidence as the average weighted ratio of probabilities assigned to observed actions and maximum probabilities prescribed by the types, where the weight is given by the posterior $\Pr_j$. Formally, for $t > 0$,

$$C^t = \frac{1}{t}\sum_{\tau=0}^{t-1} \sum_{\theta_j^* \in \Theta_j^*} \Pr_j(\theta_j^* | H^\tau)\, \frac{\pi_j(H^\tau, a_j^\tau, \theta_j^*)}{\max_{a_j} \pi_j(H^\tau, a_j, \theta_j^*)} \tag{4}$$

Algorithm 1 Schematic of Expert-HBA (E-HBA)

Let $\bar{U}$ be vector of observed average payoffs of experts
Compute posterior $\Pr_j$ using (1)
Compute vector U^* using (2)/(3)
Compute confidence C^t, e.g. using (4)
Execute expert algorithm $f\big((1-C^t)\bar{U} + C^t U^*\big)$

where C^0 can be set arbitrarily in $[0,1]$, e.g. $C^0 = 1$ to indicate extreme optimism and $C^0 = 0$ for extreme pessimism. However, C^0 has typically no bearing because most expert algorithms choose the first expert randomly.

This definition of confidence is a useful baseline because it often approximates the average probability overlap between θ_j^* and θ_j^+, which is one possible quantification of similarity. Again assuming a single hypothesised type θ_j^*, the average probability overlap between θ_j^* and θ_j^+ at time $t > 0$ is

$$\frac{1}{t}\sum_{\tau=0}^{t-1} \sum_{a_j \in A_j} \min\left[\pi_j(H^\tau, a_j, \theta_j^*), \pi_j(H^\tau, a_j, \theta_j^+)\right].$$

However, we do note that (4) is not a perfect choice due to several shortcomings. For instance, (4) will always converge to 1 if θ_j^* converges to uniform action probabilities. Thus, more research is required to formulate a suitable theory around this notion of confidence.

4.3 Guarantees

Since E-HBA maintains the posterior $\Pr_j$ in the same way as HBA, it inherits all convergence guarantees of HBA. This includes Theorems 1 to 3 in (Albrecht and Ramamoorthy 2014). For example, Theorem 1 states that, if the true type of player j is included in Θ_j^* and if the prior beliefs P_j are positive (i.e. $P_j(\theta_j^*) > 0$ for all $\theta_j^* \in \Theta_j^*$), then E-HBA's predictions of future play will eventually be correct.

This reveals an interesting property of E-HBA: Once E-HBA makes correct future predictions, and for sufficiently high h, the predicted payoffs U^* will be the *true* average payoffs of the experts. Therefore, mixing $\bar{U}$ and U^* will be *more accurate* than $\bar{U}$ alone (unless $\bar{U} = U^*$), for any $C^t > 0$. Thus, under these circumstances, the mixing will not degrade the performance of the expert algorithm.

Of course, whether h is sufficient to produce accurate predictions U^* depends on the types in Θ_j^*. In the Prisoner's Dilemma example in Section 1, even if we knew the true type from the onset, a depth of $h = 3$ would predict higher average payoffs for the expert D, while C is more profitable in the long-term. Thus, a higher h would be needed.

Likewise, U^* may be inaccurate if the true type of player j is not included in Θ_j^*. However, in this case, the confidence C^t should adjust the portion of U^* used in the mix. As we show in our experiments, this can ensure that the expert algorithm only suffers minor (or no) degradations in its performance. Moreover, if there are types in Θ_j^* which are *similar* to the true type, in the sense that they assign similar probabilities to actions, then the accuracy of U^* often remains at a level which is proportional to the similarity.

4.4 Total Payoffs

E-HBA can be applied to any expert algorithm which considers the average payoffs that experts yielded in the past, i.e. $\bar{U}$. However, there are some expert algorithms that consider *total* rather than average payoffs, e.g. Hedge (Freund and Schapire 1995) and Exp3 (Auer et al. 1995).

One way in which E-HBA can be applied to such expert algorithms is to modify them to use average rather than total payoffs. This is possible because, when payoffs are observed iteratively, the total payoff can be mapped into the average payoff. However, it is likely that such a modification would invalidate the performance guarantees of the expert algorithm. For example, Hedge and Exp3 will not converge to the best expert if they are modified to use average payoffs.

Another way to apply E-HBA to such expert algorithms is to modify the definitions of E-HBA to use total rather than average payoffs. That is, we redefine $\bar{U}$ to be the *total* payoff the expert yielded when following their recommendations, and we compute U^* using (2)/(3) but without dividing by h. This is the approach we have chosen in our experiments.

However, there are two potential complications with the latter approach, both of which are due to the fact that $\bar{U}$ is the total payoff over (up to) t steps while U^* is the total payoff over h steps. Firstly, this means that there is no notion of accuracy of $\bar{U}$ and U^*, hence our correctness claim in Section 4.3 regarding the accuracy of the mixing does not hold anymore. Secondly, as the game proceeds with ever increasing t, the impact of U^* in the mix will diminish over time for $C^t < 1$. Nonetheless, as we show in our experiments, if C^t converges to 1, then E-HBA can still significantly improve the performance of the expert algorithm.

5 Experiments

We conducted experiments in a comprehensive set of repeated matrix games, comparing the performance of several well-known expert algorithms with and without E-HBA.

5.1 Games

We used a comprehensive set of benchmark games introduced by Rapoport and Guyer (1966), which consists of 78 repeated 2×2 matrix games (i.e. 2 players with 2 actions). The games are *strictly ordinal*, meaning that each player ranks each of the 4 possible outcomes from 1 (least preferred) to 4 (most preferred), and no two outcomes have the same rank. Furthermore, the games are *distinct* in the sense that no game can be obtained by transformation of any other game, which includes interchanging the rows, columns, and players (and any combination thereof) in the payoff matrix of the game.

The games can be grouped into 21 *no-conflict* games and 57 *conflict* games. In a no-conflict game, the two players have the same most preferred outcome, and so it is relatively easy to arrive at a solution that is best for both players. In a conflict game, the players disagree on the best outcome, hence they will have to find some form of a compromise.

5.2 Experts & Types

We used three automatic methods to generate parameterised sets of experts and types for a given game. The generated policies cover a reasonable spectrum of adaptive behaviours, including deterministic (CDT), randomised (CNN), and hybrid (LFT) policies. All parameter settings can be found in the appendix (Albrecht, Crandall, and Ramamoorthy 2015).

Leader-Follower-Trigger Agents (LFT) Crandall (2014) described a method to automatically generate sets of "leader" and "follower" agents that seek to play specific sequences of joint actions, called "target solutions". A leader agent plays its part of the target solution as long as the other player does. If the other player deviates, the leader agent punishes the player by playing a minimax strategy. The follower agent is similar except that it does not punish. Rather, if the other player deviates, the follower agent randomly resets its position within the target solution and continues play as usual. We augmented this set by a trigger agent which is similar to the leader and follower agents, except that it plays its maximin strategy indefinitely once the other player deviates.

Co-Evolved Decision Trees (CDT) We used genetic programming (Koza 1992) to automatically breed sets of decision trees. A decision tree takes as input the past n actions of the other player (in our case, $n = 3$) and deterministically returns an action to be played in response. The breeding process is co-evolutional, meaning that two pools of trees are bred concurrently (one for each player). In each evolution, a random selection of the trees for player 1 is evaluated against a random selection of the trees for player 2. The fitness criterion includes the payoffs generated by a tree as well as its dissimilarity to other trees in the same pool. This was done to encourage a more diverse breeding of trees, as otherwise the trees tend to become very similar or identical.

Co-Evolved Neural Networks (CNN) We used a string-based genetic algorithm (Holland 1975) to breed sets of artificial neural networks. The process is basically the same as the one used for decision trees. However, the difference is that artificial neural networks can learn to play stochastic strategies while decision trees always play deterministic strategies. Our networks consist of one input layer with 4 nodes (one for each of the two previous actions of both players), a hidden layer with 5 nodes, and an output layer with 1 node. The node in the output layer specifies the probability of choosing action 1 (and, since we play 2×2 games, of action 2). All nodes use a sigmoidal threshold function and are fully connected to the nodes in the next layer.

5.3 Expert Algorithms

The following expert algorithms were used: UCB1 (Auer, Cesa-Bianchi, and Fischer 2002), EEE (de Farias and Megiddo 2004), S (Karandikar et al. 1998), Hedge (Freund and Schapire 1995), and Exp3 (Auer et al. 1995).

For EEE, we used the parameter settings specified in (de Farias and Megiddo 2003). S was implemented as specified in Appendix A in (Crandall 2014), using the same parameter settings. For Hedge, we used the modified version provided in Section 3 in (Auer et al. 1995). Both Hedge and Exp3 used $\eta = 0.1$, and Exp3 used $\gamma = 0.1$.

As discussed in Section 4.4, Hedge and Exp3 are based on *total* rather than average payoffs, hence we adapted E-HBA

as specified in Section 4.4. That is, the variable G_i in Hedge and Exp3 (cf. Auer et al. 1995) was used in place of U^*. (Note that, in Hedge, G_i is the total payoff of *each* expert's recommendations, not just of those which we followed.) In addition, we applied a "booster" exponent b to U^* (i.e. $(U^*)^b$) to magnify the differences between experts. We used $b = 3$.

5.4 Experimental Procedure

We performed identical experiments for every expert/type generation method described in Section 5.2. Each of the 78 games was played 10 times using different random seeds, where each play lasted 5,000 rounds (this was unknown to the players to avoid "end-game" effects).

In each play, we randomly generated 5 unique experts for player 1 (controlled by E-HBA) and 5 unique types for player 2, and provided them to E-HBA as the sets Φ_1 and Θ_2^*, respectively. E-HBA used uniform prior beliefs and a planning horizon of $h = 5$, which constituted a good trade-off between computation time and accuracy in our experiments.

Every play was repeated in two modes: one in which player 2 was controlled by a randomly generated type, and one in which it was controlled by a fictitious player (Brown 1951). We used a fictitious player because it explicitly tries to learn the behaviour of E-HBA. While the generated types are adaptive as well, they do not create models of E-HBA's behaviour.

Finally, to isolate the effects of knowing the true type of player 2, we performed each experiment once for the case in which the true type of player 2 was included in Θ_2^* and once for the case in which it was not ($|\Theta_2^*| = 5$ in both cases).

5.5 Results

Figures 1 and 2 show a selection of results from a variety of scenarios (all results are given in the appendix document). Since most of the tested expert algorithms assume payoffs in the interval $[0, 1]$, we normalised the payoffs from $\{1, 2, 3, 4\}$ to $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$. The solid and dashed lines show the average payoffs of HBA (using the same parameters and types as E-HBA) and the best expert in each play, respectively. In the following, all 'significance' statements are based on paired two-sided t-tests with a significance level of 5%.

Figure 1 shows results for the case in which the true type of player 2 was included in Θ_2^*. As the plots show, E-HBA was able to significantly improve the performance of the expert algorithms. This was observed in all constellations of games, experts/types, and opponents. UCB1 and EEE benefited the most from E-HBA, with improvements of up to 10% and 15%, respectively. The improvements of the other algorithms were less substantial but still significant. Note that HBA performed the best in all experiments because it computes best-response actions at each point in time, whereas the expert algorithms (with and without E-HBA) must choose from a set of predefined expert policies that may not be optimal.

Another interesting observation is that E-HBA often meant the difference between performing worse and better than the best expert (e.g. Figures 1a and 1b). This is precisely a result of the fact that the tested expert algorithms rely heavily on the past performance of experts (as discussed in Section 1), while E-HBA also considers the future performance of experts at any given point. This allowed the modified expert algorithms to switch effectively between experts, which on average resulted in higher payoffs than persistently following any single expert.

It is important to note that the performance enhancements of E-HBA came at an increased computational cost. For instance, using a planning depth of $h = 5$ and the CDT types, the modified expert algorithms required roughly 10 times more time than the original algorithms. (However, we note that our implementation was not optimised for speed.) Therefore, whether or not the performance improvements are worth the additional costs is an important aspect that should be taken into account. Nonetheless, as discussed previously, the computational costs can often be reduced drastically by using efficient Monte-Carlo Tree Search methods.

Figure 2 shows results for the case in which the true type of player 2 was *not* included in Θ_2^*. Here, we observe that the modified algorithms performed similarly to the original algorithms, and substantially better than HBA (which often ended up playing randomly). This was due to the confidence C^t, which decreased quickly and remained low in many cases. Interestingly, there are cases in which the expert algorithms still benefited from E-HBA, especially with CDT and CNN in no-conflict games. In these cases, the set Θ_2^* included a type that was similar to the true type (in the sense that they assigned similar probabilities to actions) so that E-HBA and HBA were still able to make useful predictions.

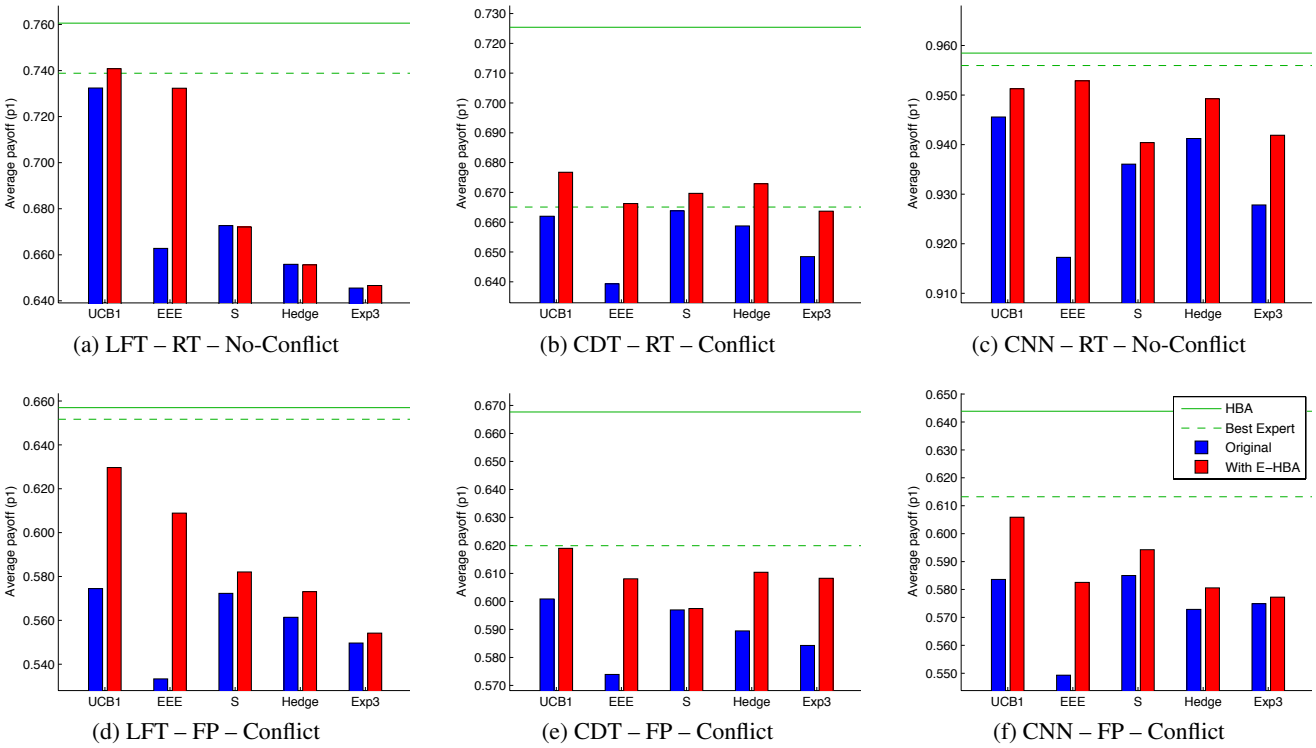
Finally, we note that, in some cases, Hedge and Exp3 performed significantly worse when combined with E-HBA. We found that this was mostly due to the problems described Section 4.4. Thus, while E-HBA works well in combination with expert algorithms that use average payoffs, it may not be optimal for expert algorithms that use total payoffs

6 Conclusion

Past research has studied two methods to utilise pre-defined policy sets in repeated interactions: as *experts*, to dictate our own actions, and as *types*, to characterise the behaviour of other agents. The contribution in this work is to bring these complementary views together, with the goal of combining their strengths and alleviating their weaknesses.

We have done so in the form of a meta-algorithm, E-HBA, which can be applied to any expert algorithm that considers the average (or total) payoff an expert yielded in the past. E-HBA mixes these observed payoffs with a predicted future payoff for each expert, using the type-based approach. Our experiments show that, if the true (or a similar) type of the opponent is known, then E-HBA can significantly improve the performance of expert algorithms, while in all other cases it performs similarly to the original expert algorithm.

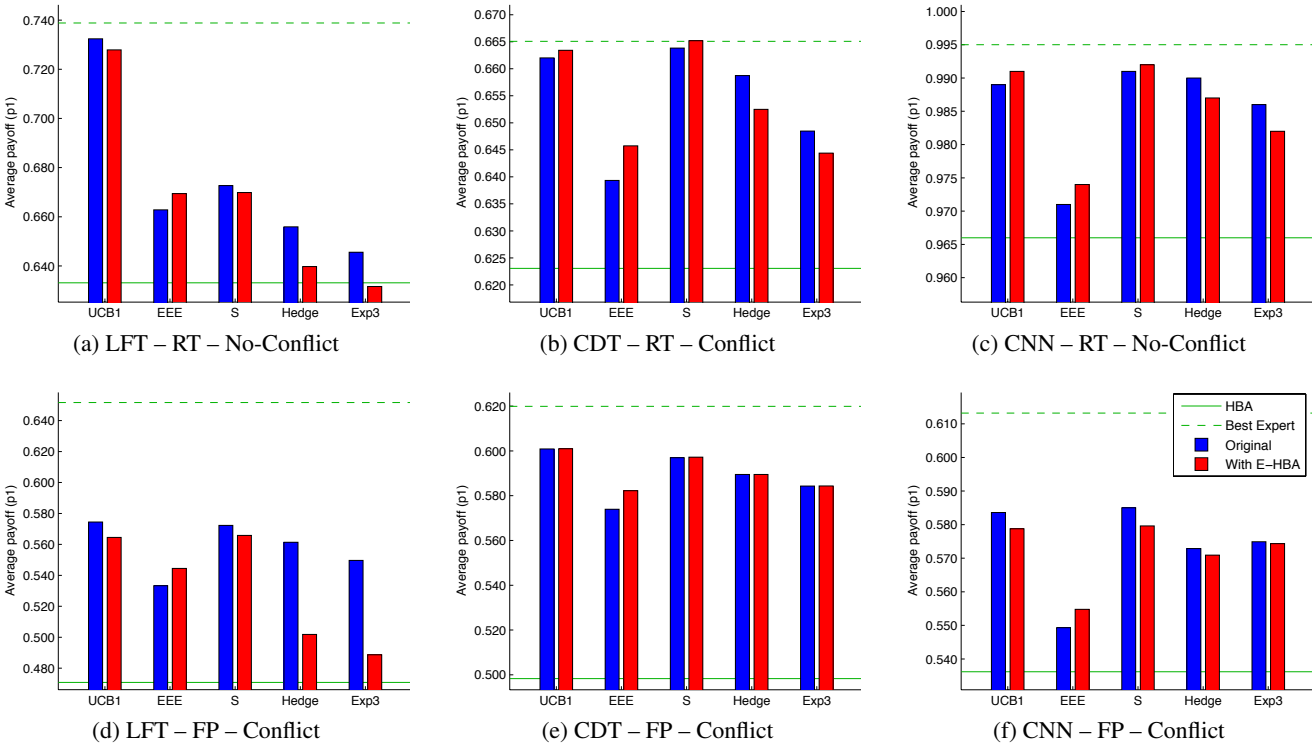
At the core of E-HBA is the confidence, C^t, which is used to regulate the mixing. We provide an intuitive definition of confidence which is simple to compute and works well in practice. However, we believe that other useful definitions of confidence exist, and in this sense we view E-HBA as a *family* of meta-algorithms. An important step in establishing this idea will be to develop a theory around this notion of confidence, similar to the theories on regret in expert algorithms and the theories on prior and posterior beliefs in the type-based approach.



(a) LFT – RT – No-Conflict (b) CDT – RT – Conflict (c) CNN – RT – No-Conflict

(d) LFT – FP – Conflict (e) CDT – FP – Conflict (f) CNN – FP – Conflict

Figure 1: **True type of player 2 included in Θ_2^*.** X–Y–Z format means that experts and types were generated by X, player 2 was controlled by Y, and results are shown for Z games. RT denotes random type and FP denotes fictitious player.



(a) LFT – RT – No-Conflict (b) CDT – RT – Conflict (c) CNN – RT – No-Conflict

(d) LFT – FP – Conflict (e) CDT – FP – Conflict (f) CNN – FP – Conflict

Figure 2: **True type of player 2 *not* included in Θ_2^*.** X–Y–Z format means that experts and types were generated by X, player 2 was controlled by Y, and results are shown for Z games. RT denotes random type and FP denotes fictitious player.

References

Albrecht, S., and Ramamoorthy, S. 2013. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems (extended abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 1155–1156.

Albrecht, S., and Ramamoorthy, S. 2014. On convergence and optimality of best-response learning with policy types in multiagent systems. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 12–21.

Albrecht, S.; Crandall, J.; and Ramamoorthy, S. 2015. E-HBA: Using action policies for expert advice and agent typification – Appendix. *http://rad.inf.ed.ac.uk/data/publications/2015/mipc15app.pdf*.

Arora, R.; Dekel, O.; and Tewari, A. 2012. Online bandit learning against an adaptive adversary: From regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning*, 1503–1510.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Symposium on the Foundations of Computer Science*, 322–331.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Barrett, S.; Stone, P.; and Kraus, S. 2011. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, 567–574.

Brown, G. 1951. Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13(1):374–376.

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

Crandall, J. 2014. Towards minimizing disappointment in repeated games. *Journal of Artificial Intelligence Research* 49:111–142.

de Farias, D., and Megiddo, N. 2003. How to combine expert (or novice) advice when actions impact the environment. In *Advances in Neural Information Processing Systems 16*, 815–822.

de Farias, D., and Megiddo, N. 2004. Exploration-exploitation tradeoffs for experts algorithms in reactive environments. In *Advances in Neural Information Processing Systems 17*, 409–416.

Foster, D., and Vohra, R. 1999. Regret in the on-line decision problem. *Games and Economic Behavior* 29(1):7–35.

Freund, Y., and Schapire, R. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, 23–37. Springer.

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

Holland, J. 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. The MIT Press.

Karandikar, R.; Mookherjee, D.; Ray, D.; and Vega-Redondo, F. 1998. Evolving aspirations and cooperation. *Journal of Economic Theory* 80(2):292–331.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, volume 4212 of Lecture Notes in Computer Science, pages 282–293*. Springer.

Koza, J. 1992. *Genetic programming: On the programming of computers by means of natural selection*. The MIT Press.

Littlestone, N., and Warmuth, M. 1994. The weighted majority algorithm. *Information and Computation* 108(2):212–261.

Rapoport, A., and Guyer, M. 1966. A taxonomy of 2×2 games. *General Systems: Yearbook of the Society for General Systems Research* 11:203–214.