Maintaining Cooperation in Noisy Environments

Tsz-Chiu Au
Department of Computer Science
University of Maryland
College Park, Maryland, 20742
chiu@cs.umd.edu

Dana Nau
Department of Computer Science
University of Maryland
College Park, Maryland, 20742
nau@cs.umd.edu

Abstract

To prevent or alleviate conflicts in multi-agent environments, it is important to distinguish between situations where another agent has misbehaved intentionally and situations where the misbehavior was accidental. One situation where this problem arises is the Noisy Iterated Prisoner's Dilemma, a version of the Iterated Prisoner's Dilemma (IPD) in which there is a nonzero probability that a "cooperate" action will accidentally be changed into a "defect" action and vice versa. Tit-For-Tat and other strategies that do quite well in the ordinary (non-noisy) IPD can do quite badly in the Noisy IPD.

This paper presents a technique called *symbolic noise detection*, for detecting whether anomalies in player's behavior are deliberate or accidental. This idea to use player's deterministic behavior to tell whether an action has been affected by noise. We also present DBS, an algorithm that uses symbolic noise detection in the Noisy IPD. DBS constructs a model of the other agent's deterministic behavior, and watches for any deviation from this model. If the other agent's next action is inconsistent with this model, the inconsistency can be due either to noise or to a genuine change in their behavior; and DBS can often distinguish between two cases by waiting to see whether this inconsistency persists in next few moves. This technique is effective because many IPD players often have clear deterministic patterns of behavior.

We entered several different implementations of DBS in the 2005 Iterated Prisoner's Dilemma competition, in Category 2 (noisy environments). Out of the 165 contestants in this category, most of DBS implementations ranked among top ten. The best one ranked third, and it was beaten only by two "master-and-slaves strategy" programs that each had a large number of "slave" programs feeding points to them.

Introduction

The Iterated Prisoner's Dilemma (IPD) has become well known as an abstract model of a class of multi-agent environments in which agents accumulate payoffs that depend on how successful they are in their repeated interactions with other agents. An important variant of the IPD is the *Noisy* IPD, in which there is a small probability, called the *noise level*, that accidents will occur. In other words, the noise level is the probability of executing "cooperate" when "defect" was the intended move, or vice versa.

This work is based on an earlier work: *Accident or Intention: That Is the Question (in the Noisy Iterated Prisoner's Dilemma)*, in AAMAS'06 (May 8–12 2006) ©ACM, 2006.

Accidents can cause difficulty in cooperations with others in real-life situations, and the same is true in the Noisy IPD. Strategies that do quite well in the ordinary (non-noisy) IPD may do quite badly in the Noisy IPD (Axelrod & Dion 1988; Bendor 1987; Bendor, Kramer, & Stout 1991; Molander 1985; Mueller 1987; Nowak & Sigmund 1990). For example, if two players both use the well-known Tit-For-Tat (TFT) strategy, then an accidental defection may cause a long series of defections by both players as each of them punishes the other for defecting. How to cope with noise is an important issue for agents to maintain cooperation in noisy environments.

In this paper, we describe a technique called *symbolic noise detection*—the use of the other player's deterministic behavior to tell whether an action has been affected by noise. We describe an algorithm called DBS that implements this technique by building a symbolic model of how the other agent behaves, and watching for any deviation from this model. If the other agent's next move is inconsistent with their past behavior, this inconsistency can be due either to noise or to a genuine change in their behavior; and DBS can often distinguish between these two cases by waiting to see if this inconsistency persists or not in the next few iterations of the game. For more details, see (Au & Nau 2006)

In short, our strategy focuses on an important question for conflict prevention in noisy environments: *whether a misconduct is intentional or accidental.* In a noisy environment, a deviation from an agent's usual course of action can be explained in either way. If we form a wrong belief about which explanation is correct, our response may potentially destroy a good long-term relationship. If we ground our belief on evidence accumulated before and after the incident, we should be in a better position to identify the true cause and prescribe an appropriate solution. To accomplish this, DBS uses the following key techniques:

1. **Learning about the other player's strategy.** DBS uses an induction technique to create rules that model the other player's recent behavior. The rules give the probability that the player will cooperate under different situations. When a rule is consistent with all of the other player's recent behaviors, DBS changes it to a *deterministic* rule

that has either 0 or 1 as the probability of cooperation.

2. **Detecting noise**. DBS uses the deterministic rules to detect anomalies that may be due either to noise or a genuine change in the other player's behavior. If a move is different from what those rules predict, this inconsistency triggers an *evidence collection process* to monitor the persistence of the inconsistency in the next few iterations of the game. The purpose of the evidence-collection process is to determine whether the violation is likely to be due to noise or to a change in the other player's policy.
3. **Temporarily tolerating possible misbehaviors by the other player.** Until the evidence-collection process finishes, DBS assumes that the other player's behavior is still as described by the deterministic rules. Once the evidence collection process has finished, DBS decides whether to believe the other player's behavior has changed, and updates the deterministic rules accordingly.
4. **Decision making based on the model of the other player.** DBS uses a game-tree search procedure to generate moves that will maximize DBS's utility if the other player continues to behave as DBS's model predicts.

Since DBS emphasizes the use of deterministic behaviors to distinguish noise from the change of the other player's behavior, it works well when the other player uses a pure (i.e., deterministic) strategy or a strategy that makes decisions deterministically most of the time. Fortunately, deterministic behaviors are abundant in the Iterated Prisoner's Dilemma. Many well-known strategies, such as TFT and GRIM, are pure strategies. Some strategies such as Pavlov or Win-Stay, Lose-Shift strategy (WSLS) (Kraines & Kraines 1989; 1993; 1995; Nowak & Sigmund 1993) are not pure strategies, but a large part of their behavior is still deterministic. The reason for the prevalence of determinism is discussed by Axelrod in (Axelrod 1984): clarity of behavior is an important ingredient of long-term cooperation. A strategy such as TFT benefits from its clarity of behavior, because it allows other players to make credible predictions of TFT's responses to their actions. We believe the success of our strategy in the competition is because this clarity of behavior also helps us to fend off noise.

An Outline of DBS

We call our strategy *Derived Belief Strategy (DBS)*. Figure 1 is an outline of the DBS procedure. The hypothesized policy π is a rule-based model of the other player's recent behavior; and to decide what moves to make, DBS does a game-tree search against this model's predictions. π includes probabilistic rules saying what moves the other player is likely to make under various conditions, and deterministic rules saying what moves the other player *will* make under various conditions. If the other player's last move contradicts a deterministic rule r in π, DBS will ignore this contradiction the first few times it happens, and will not update r. However, if the same contradiction occurs several times, DBS will substitute a probabilistic rule for r in π. Likewise, will DBS convert a probabilistic rule to a deterministic rule if all of the other player's recent moves are consistent with the deterministic rule; and otherwise DBS will update the probabilistic rule by updating the probability. For more details about this procedure, please read (Au & Nau 2006).

```
Procedure $\mathbf{DerivedBeliefStrategy}()$
 Initialize the hypothesized policy $\pi$ (e.g., TFT)
 Loop until the end of the game
  Generate a move $a$ based on $\pi$ (by game-tree search)
  Obtain the other player's move $b$
  If $b$ contradicts any deterministic rule $r$ in $\pi$
    & the contradiction has recently occurred several times
      replace $r$ in $\pi$ with a probabilistic rule
  Else if $b$ is consistent with a deterministic rule $r$ not in $\pi$
    & the consistency has recently occurred repeatedly
      replace the corresponding probabilistic rule in $\pi$ with $r$
```

Figure 1: An outline of the DBS procedure.

Table 1: Scores of the top ten programs, averaged over the five runs in Competition 2.

Rank	Program	Avg. score
1	BWIN	433.8
2	IMM01	414.1
3	DBSz	408.0
4	DBSy	408.0
5	DBSpl	407.5
6	DBSx	406.6
7	DBSf	402.0
8	DBStft	401.8
9	DBSd	400.9
10	lowESTFT_classic	397.2

Competition Results

To test DBS's performance, we entered nine different versions of it as contestants in Category 2 (the Noisy IPD) of the 2005 IPD competition (Kendall, Darwen, & Yao 2005). Each version of DBS had a different set of parameters or different implementation. Table 1 shows the average scores of the top ten programs. The competition website at http://www.prisoners-dilemma.com gives a more extensive set of tables showing each program's ranking in each run.

DBS performed impressively in the competition: all nine versions of DBS were in the top 25, and seven of them placed among top ten. The best version, DBSz, placed third; and it lost only to BWIN and IMM01.

BWIN and IMM01 both used *master-and-slaves* strategies, an approach that took advantage of the fact that each participant in the competition was allowed to submit up to 20 programs as contestants. BWIN was a *master* program with 19 *slave* programs feeding points to it. Whenever BWIN and one of its slaves played, BWIN would defect and the slave would cooperate so that BWIN got 5 points and the slave got nothing. The same was true for IMM01.

In constrast, DBS does not use a master-and-slaves strategy, nor does it conspire with other programs in any other way. Nonetheless, DBS remained competitive with the master-and-slaves strategies. Furthermore, if we average the score of each master with the scores of its slaves, the average score for BWIN and its slaves is 379.9, and the average score for IMM01 and its slaves is 351.7, both of which are less than the scores of any of our versions of DBS. Furthermore, a more extensive analysis (Au & Nau 2005) shows that if the size of each master-and-slaves team had been limited to less than 10, DBSz would have outperformed both BWIN and IMM01, even without averaging their scores with their slaves' scores.

Related Work on the IPD

Early studies of the effect of noise in the Iterated Prisoner's Dilemma focused on how TFT, a highly successful strategy in noise-free environments, would do in the presence of noise. TFT is known to be vulnerable to noise; for instance, if two players use TFT at the same time, noise would trigger long sequences of mutual defections (Molander 1985). A number of people confirmed the negative effects of noise to TFT (Axelrod & Dion 1988; Bendor 1987; Bendor, Kramer, & Stout 1991; Molander 1985; Mueller 1987; Nowak & Sigmund 1990). Axelrod found that TFT was still the best decision rule in the rerun of his first tournament with a one percent chance of misperception (Axelrod 1984, page 183), but TFT finished sixth out of 21 in the rerun of Axelrod's second tournament with a 10 percent chance of misperception (Donninger 1986). In Competition 2 of the 2005 IPD competition, the noise level was 0.1, and TFT's overall average score placed it 33rd out of 165.

The oldest approach to remedy TFT's deficiency in dealing with noise is to be more forgiving in the face of defections. A number of studies found that more forgiveness promotes cooperation in noisy environments (Bendor, Kramer, & Stout 1991; Mueller 1987). For instance, Tit-For-Two-Tats (TFTT) retaliates only when it receives two defections in two previous iterations. TFTT can tolerate isolated instances of defections caused by noise and is more readily to avoid long sequences of mutual defections. However, TFTT is susceptible to exploitation of its generosity and was beaten in Axelrod's second tournament by TESTER, a strategy that may defect every other move. In Competition 2 of the 2005 IPD Competition, TFTT ranked 30—a slightly better ranking than TFT's. In contrast to TFTT, DBS can tolerate not only an isolated defection but also a sequence of defections caused by noise, and at the same time DBS monitors the other player's behavior and retaliates when exploitation behavior is detected (i.e., when the exploitation causes a change of the hypothesized policy, which initially is TFT).

(Molander 1985) proposed to mix TFT with ALLC to form a strategy which is now called Generous Tit-For-Tat (GTFT) (Nowak & Sigmund 1992). Like TFTT, GTFT avoids an infinite echo of defections by cooperating when it receives a defection in certain iterations. The difference is that GTFT forgives randomly: for each defection GTFT receives, it randomly chooses to cooperate with a small probability (say 10%) and to defect otherwise. DBS, however, does not make use of forgiveness explicitly as GTFT does. DBS's decisions are based entirely on the hypothesized policy that it has learned. But temporary tolerance can be deemed a form of forgiveness, since DBS does not retaliate immediately when a defection occurs in a mutual cooperation situation. This form of forgiveness is carefully planned and there is no randomness in it.

A family of strategies called "Pavlovian" strategies, or simply called Pavlov, was found to be more successful than TFT in noisy environments (Kraines & Kraines 1989; 1993; 1995; Nowak & Sigmund 1993). When an accidental defection occurs, Pavlov can resume mutual cooperation in a smaller number of iterations than TFT (Kraines & Kraines 1989; 1993). Pavlov learns by conditioned response through rewards and punishments; it adjusts its probability of cooperation according to the previous outcome. Like Pavlov, DBS learns from its past experience. DBS, however, has an intermediate step between learning from experience and decision making: it maintains a model of the other player's behavior, and uses this model to reason about nosie.

Relationships to Other Areas within AI

There are a number of games similar to the Iterated Prisoner's Dilemma that are used in studying behavior among self-interested agents (Axelrod 1984; Dawkins 1990). Thus, the ideas we have described in this paper are likely to be useful in research on computer games, and in research in multi-agent systems. Below we discuss each of those areas in more detail.

Game Playing

The use of opponent modeling is important in games of imperfect information such as Poker (Barone & While 2000; Billings *et al.* 2003; Davidson *et al.* 2000) and RoShamBo (Egnor 2000). There have been many works on learning opponent's strategy in the non-noisy IPD (Dyer 2004; Hingston & Kendall 2004). A common approach to opponent modeling is to model the opponent's strategy as a probabilistic finite automaton, assuming the opponent's next move depends only on the outcomes of the last several iterations. Then the probabilities in the automata are learnt using various learning methods. In contrast, DBS does not aim at learning the other player's strategy completely; instead, it learns the other player's recent behavior, since the other player may alter its strategy in the middle of a game.

To the best of our knowledge, ours is the first piece of work on using opponent models in the IPD to detect errors in the execution of other agent's actions. We believe our technique may also be useful in other kinds of environments.

Multi-Agent Systems

There have been many works on the study of cooperation in multi-agent systems (Ferber 1999; Weiss 2000; Wooldridge 2002). Noise can have a destructive effect to the formation and maintenance of coordination relationships among agents. The agents in a general multi-agent environment are usually more capable than those in the IPD, and therefore they can cope with noise by a variety of methods. For instance, if agents can communicate with each other, the agents may detect noise by a predefined communication protocol. However, if the agents cannot completely trust each other, then we believe that there is no protocol that is guaranteed to tell which action has been affected by noise. It would be interesting to see how symbolic noise detection could enhance these methods or vice versa.

Summary

For conflict prevention in noisy environments, a critical problem is to distinguish between situations where another player has misbehaved intentionally and situations where the misbehavior was accidental. That is the problem that DBS was formulated to deal with.

DBS's impressive performance in the 2005 Iterated Prisoner's Dilemma competition occurred because DBS was better able to maintain cooperation in spite of noise than any other program in the competition.

Since clarity of behavior is an important ingredient of long-term cooperation in the IPD, most IPD programs have behavior that follows clear deterministic patterns. The clarity of these patterns made it possible for DBS to construct policies that were good approximations of the other players' strategies, and to use these policies to fend off noise.

We believe that clarity of behavior is also likely to be important in other multi-agent environments in which agents have to cooperate with each other. Thus it seems plausible that techniques similar to those used in DBS may be useful in those domains.

In the Noisy IPD competition, there was no way for an agent to tell whether an execution of an action was affected by noise or not. In other environments, agents may be able to obtain partial information about whether noise has occurred. There may be some interesting ways to utilize that information within symbolic noise detection.

Acknowledgments

This work was supported in part by ISLE contract 0508268818 (subcontract to DARPA's Transfer Learning program), UC Berkeley contract SA451832441 (subcontract to DARPA's REAL program), and NSF grant IIS0412812. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funders.

References

Au, T.-C., and Nau, D. 2005. An Analysis of Derived Belief Strategy's Performance in the 2005 Iterated Prisoner's Dilemma Competition. Technical Report CSTR-4756/UMIACS-TR-2005-59, University of Maryland.

Au, T.-C., and Nau, D. 2006. Accident or intention: That is the question (in the noisy iterated prisoner's dilemma).

Axelrod, R., and Dion, D. 1988. The further evolution of cooperation. *Science* 242(4884):1385–1390.

Axelrod, R. 1984. *The Evolution of Cooperation*. Basic Books.

Barone, L., and While, L. 2000. Adaptive learning for poker. In *GECCO-2000*, 566–573.

Bendor, J.; Kramer, R. M.; and Stout, S. 1991. When in doubt... cooperation in a noisy prisoner's dilemma. *The Journal of Conflict Resolution* 35(4):691–719.

Bendor, J. 1987. In good times and bad: Reciprocity in an uncertain world. *American Journal of Politicial Science* 31(3):531–558.

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; and Schaeffer, J. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI-2003*, 661–668.

Davidson, A.; Billings, D.; Schaeffer, J.; and Szafron, D. 2000. Improved opponent modeling in poker. In *ICAI'2000*, 1467–1473.

Dawkins, R., ed. 1990. *The Selfish Gene*. Oxford University Press.

Donninger, C. 1986. *Paradoxical Effects of Social Behavior*. Heidelberg: Physica Verlag. chapter Is it always efficient to be nice?, 123–134.

Dyer, D. W. 2004. Opponent modelling and strategy evolution in the iterated prisoner's dilemma. Master's thesis, School of Computer Science and Software Engineering, The Univ. of Western Australia.

Egnor, D. 2000. Iocaine powder explained. *ICGA Journal* 23(1):33–35.

Ferber, J., ed. 1999. *Multi-Agent Systems: An Introduction to Distrbuted Artificial Intelligence*. Addison-Wesley Professional.

Hingston, P., and Kendall, G. 2004. Learning versus evolution in iterated prisoner's dilemma. In *GECCO-2004*.

Kendall, G.; Darwen, P.; and Yao, X. 2005. The iterated prisoner's dilemma competition. http://www.prisoners-dilemma.com.

Kraines, D., and Kraines, V. 1989. Pavlov and the prisoner's dilemma. *Theory and Decision* 26:47–79.

Kraines, D., and Kraines, V. 1993. Learning to cooperate with pavlov an adaptive strategy for the iterated prisoner's dilemma with noise. *Theory and Decision* 35:107–150.

Kraines, D., and Kraines, V. 1995. Evolution of learning among pavlov strategies in a competitive environment with noise. *The Journal of Conflict Resolution* 39(3):439–466.

Molander, P. 1985. The optimal level of generosity in a selfish, uncertain environment. *The Journal of Conflict Resolution* 29(4):611–618.

Mueller, U. 1987. Optimal retaliation for optimal cooperation. *The Journal of Conflict Resolution* 31(4):692–724.

Nowak, M., and Sigmund, K. 1990. The evolution of stochastic strategies in the prisoner's dilemma. *Acta Applicandae Mathematicae* 20:247–265.

Nowak, M. A., and Sigmund, K. 1992. Tit for tat in heterogeneous populations. *Nature* 355:250–253.

Nowak, M., and Sigmund, K. 1993. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature* 364:56–58.

Weiss, G., ed. 2000. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press.

Wooldridge, M. 2002. *Introduction to MultiAgent Systems*. John Wiley & Sons Ltd.