# Asymmetric Co-evolution for Imperfect-Information Zero-Sum Games

Ole Martin Halck and Fredrik A. Dahl

Norwegian Defence Research Establishment (FFI)
P.O. Box 25, NO-2027 Kjeller, Norway
{Ole-Martin.Halck,Fredrik-A.Dahl}@ffi.no

**Abstract.** We present an asymmetric co-evolutionary learning algorithm for imperfect-information zero-sum games. This algorithm is designed so that the fitness of the individual agents is calculated in a way that is compatible with the goal of game-theoretic optimality. This compatibility has been somewhat lacking in previous co-evolutionary approaches, as these have often depended on unwarranted assumptions about the absolute and relative strength of players. Our algorithm design is tested on a game for which the optimal strategy is known, and is seen to work well.

## 1 Introduction

Within the field of machine learning, learning to play games presents special challenges. Whereas other learning tasks usually involve a fixed problem environment, game environments are more variable, as a game-playing agent must expect to face different opponents. In imperfect-information games, a class of games that has received relatively little attention in machine learning, the challenges are even greater, due to the need of acting unpredictably. In addition to the challenges encountered during the learning itself, there are also difficulties connected to evaluating the success of the training procedure, as this evaluation will need to take into account the agent's performance against varying opposition.

One main approach that has been applied to the problem of learning to play games is *co-evolution*. In co-evolutionary learning, agents are evaluated and evolved in accordance to their performance in actual game-play against other evolving agents. The degree of success achieved by the co-evolution of agents has been variable; in this paper, we attempt to shed some light on the reasons for this.

The main contributions of this paper comprise a theoretical and a practical component. We argue that much previous research of machine learning in games reveals a need of theoretical awareness regarding the evaluation of game-playing agents in the two phases of the learning itself and the assessment of the success of learning. We attempt to address this need by presenting a theoretical evaluation criterion that is consistent with game theory. On a practical level, we use this theoretical viewpoint in reviewing different co-evolutionary learning methods, and present a new, asymmetric co-evolutionary design that solves some of the problems attached to more traditional approaches.

The remainder of the paper is laid out as follows: Section 2 treats the relationship between machine learning and game theory; here we present our evaluation criterion and discuss the goals of learning in games. In Section 3 we describe different designs for co-evolutionary learning in games – including our new algorithm – and examine their properties in a game-theoretical light. Section 4 describes experiments that illustrate the treatment given in Section 3. A discussion of our goals, method and results is given in Section 5, while Section 6 concludes the paper.

## 2 Machine Learning and Imperfect-Information Games

In game theory, a distinction is made between games with perfect and imperfect information. In perfect-information games, the players always have the same information about the game state; in imperfect-information games, the players have different state information. Poker is an example of an imperfect-information game – the players know their own cards, but not those of their opponents. A seemingly different source of information imperfection occurs in games with simultaneous actions, such as scissors–paper–rock. However, these games may be transformed into equivalent alternating-turn games with "normal" imperfect information (see e.g. [2]), and vice versa.

In the literature on machine learning in games, most of the focus has been on games with perfect information. Imperfect-information games seem to have been somewhat neglected in comparison, as noted and discussed in [4] and [2].

In this paper, we restrict our attention to two-player zero-sum games with imperfect information. The consequences of the zero-sum restriction, along with other important game-theoretical background, is explained in the following. We then turn to the significance this theory has for evaluating game-playing agents and for machine learning of games.

### 2.1 Theory of Imperfect-Information Zero-Sum Games

In the tradition of von Neumann and Morgenstern [10], a *game* is defined as a decision problem with two or more decision makers – *players* – where the outcome for each player may depend on the decisions made by all players. Each player evaluates possible outcomes in terms of his own utility function, and works to maximise his own expected utility only. Here, we restrict ourselves to games with two players; these players will be called Blue and Red.

A *pure strategy* for a player is a deterministic plan dictating his actions in every possible observed state of the game. A *mixed* or *randomized strategy* is a weighted average of pure strategies, where the weights are interpreted as the probability of choosing the associated pure strategy. A mixed strategy may also be specified in a behavioural way, by giving the probability distributions over available actions in each possible game state. Only finite games are considered in this paper, that is, we will assume that each player has a finite number of pure strategies, and that the payoffs are bounded.

We further limit our attention to *zero-sum* games, that is games where one player wins what the other loses, thus eliminating any incentive for co-operation between the players. Any finite two-player zero-sum game has a *value v*, a real number with the property that Blue has a strategy (possibly mixed) which guarantees that the expected payoff will be at least *v*, while Red has a strategy guaranteeing that Blue's payoff is at most *v* [9]. Clearly, these strategies are then *minimax strategies* or *solutions*, strategies that give the respective players their highest payoffs against their most dangerous respective opponents. Furthermore, when (and only when) both players employ minimax strategies, a *minimax equilibrium* or *solution* of the game occurs; the definition of such an equilibrium is that neither player gains by deviating from his strategy, assuming that the opponent does not deviate from his. The minimax equilibrium need not be unique, but in zero-sum games all such equilibria are associated with the same value.

In perfect-information games, there exist deterministic minimax equilibria, that is equilibria where each player can play optimally in the game-theoretic sense by employing a pure strategy. In games with imperfect information, however, mixed strategies are in general necessary. In scissors–paper–rock, for instance, the unique minimax strategy for each player is to choose randomly, with uniform probability, between the three pure strategies.

A more thorough treatment of these and other aspects of game theory can be found in e.g. [7].

## 2.2 Evaluating Performance

We now present a game-theoretic evaluation criterion for players of two-player zero-sum imperfect-information games. The set of all mixed strategies as defined above is denoted by *M*, a player is specified by the strategy it employs. Although this theoretical criterion is not practically applicable in games that have not been solved, it is crucial for a stringent treatment of game learning. For a further discussion of evaluation criteria in games, see [2].

The criterion we use is that of *equity against worst-case opponent*, denoted by *Geq*. For a given $P \in M$ it is defined as

$$Geq(P) = \inf_{Q \in M} \{E(P, Q)\}, \qquad (1)$$

where $E(P, Q)$ denotes the expected outcome of *P* when playing against *Q*. According to this definition, the *Geq* measure gives the expected outcome for *P* when playing against its most dangerous opposing strategy. In a game with value *v*, it is clear that $Geq(P) \leq v$ for all $P \in M$; *P* is a minimax solution if and only if $Geq(P) = v$. Thus, the *Geq* criterion has an immediate game-theoretic interpretation. It should also be noted that for a given *P*, there exists a *pure* opposing strategy *Q* that reaches the infimum, that is, there is a deterministic agent which makes *P* look the worst.

### 2.3 The Goals of Learning

The general goal of machine learning algorithms is to perform well in a problem domain by using information gained from experience within that domain. The agent typically trains itself on a limited set of domain data in order to become adept at handling situations that are not covered by the training set. If this is to succeed, it is clearly important that the feedback received during training corresponds to what we mean by good performance within the domain. In addition, the practitioner of machine learning needs to assess the degree to which the learning has been successful. Thus, performance evaluation is important both in the learning itself and in the assessment of the success of the learning procedure. Without evaluation, learning can neither be measured nor occur.

Machine learning in games presents special problems compared to other domains. The feedback received by an agent during game play depends critically on the agent it is playing against; that is, the environment is not fixed. Furthermore, for games that have not already been solved, it is difficult to define an objective evaluation criterion, and performance has to be measured in actual game play, which, again, depends on the opponents used.

Often, the goal of game-learning work is inadequately expressed. It is taken for granted that we want the resulting player to play "well", hopefully even "optimally", without any clear definition of what this entails – the idea of objective game-play quality is taken for granted. In some cases, agents are trained against and evaluated by the same opponents; this in essence turns the problem into a normal learning problem rather than a game-learning one. Sometimes, however, the goal is clearly stated, as in [6], where it is said: "In the game theory literature, the resolution of this dilemma is to eliminate the choice and evaluate each policy with respect to the opponent that makes it look the worst." According to this view, "optimal play" takes on the natural meaning of "game-theoretic solution", and the *Geq* criterion is the correct one for player evaluation. This is the view that will be used in the following.

## 3 Co-evolutionary Approaches

Within the framework of evolutionary computation, *co-evolution* has been used as a way of overcoming the problems presented by game-playing domains, see e.g. [1] and [12]. Here, an agent's fitness is measured by its performance in actual game play against other evolving agents, rather than how well it performs in a fixed environment. The idea is that the evolving players will drive each other toward the optimum by an evolutionary "arms race". In the following we discuss some basic forms of co-evolutionary learning and certain problems associated with these, and present an algorithm designed to overcome these problems. In all cases discussed, we consider two-population co-evolution, where each population contains players of one side in the game.

## 3.1 Basic Forms of Co-evolution

**Accumulated Fitness.** A seemingly natural way of evaluating the individuals in co-evolving populations is to play a tournament where each Blue individual plays each Red individual, accumulating the scores from the single games. An individual's fitness is then the total number of points scored against all members of the other population.

This approach may work in special cases, but fails in general. Several plausible reasons for this type of failure have been suggested, e.g. in [12], along with remedies for these problems. However, we see the main problem as lying in the "arms race" assumption mentioned above. This assumption is based on the idea that relative performance between players correlates well to the quality of the players as measured by the ultimate goal of the training (in our case a high *Geq* score), so that players beating each other in turn will get closer to this goal. Thus, a high degree of transitivity in the "who-beats-whom" relation is assumed.

Unfortunately, games generally display a lack of such transitivity, and this is especially true for imperfect-information games – scissors–paper–rock provides a trivial example. (This has also been recognised in e.g. [11].) Thus we see that the essential fault in this form of co-evolution is the discrepancy between the criterion used for giving feedback to the players and the criterion we evaluate them according to after training is done.

**Worst-Case Fitness.** With the above in mind, we naturally seek a better way of assigning fitness to the players during co-evolution, a way which corresponds better to our goal of a high *Geq* score. Since the *Geq* criterion tells us the expected performance when pitted against the most effective counter-strategy, it is tempting to let each individual's fitness be given by an estimate of its performance against the member of the other population which is most dangerous to the individual being evaluated.

Due to the mixed strategies of the agents, this calls for a more time-consuming tournament than in the case of accumulated fitness. With accumulated fitness, one game against each opponent gives an unbiased estimate of the fitness, as the expected value of the sum of the outcomes equals the sum of the expected values. Worst-case fitness, on the other hand, requires several games against each opponent, as the expected value of the minimum of the outcomes (which can be estimated by playing one game against each) is different from the minimum of the expected values, which is the fitness measure we want.

What is even worse, though, is that even if we play the number of games necessary for achieving good worst-case fitness estimates, this method cannot be expected to converge towards an optimal *Geq* score. The reason lies in the somewhat paradoxical nature of the minimax solution concept. At an equilibrium, where both sides play mixed strategies that are minimax solutions, neither side has anything to gain by deviating unilaterally. On the other hand, there is also nothing to *lose* by unilateral deviation, as long as only pure strategies present in the optimal mixture are used. Thus, even if the co-evolutionary procedure were to attain the optimum, this would not be a stable state.

## 3.2 An Algorithm for Asymmetric Co-evolution

We are now able to identify some conditions that should be met by a co-evolutionary game-learning algorithm if we are to expect convergence towards the game-theoretic optimum. First, the fitness evaluations should conform to the goal of the training – that is, they should be estimates of the *Geq* values of the individuals. Secondly, the minimax strategy – which is what we want – should be a stable state of the algorithm. We here propose an algorithm that is designed to meet these conditions.

**The Populations.** The most important feature of our algorithm is its *asymmetry*. Recall from Section 2.2 that among the most effective strategies against a given individual, there is always a pure one. Since we want the fitness of our resulting individuals to reflect the *Geq* criterion, we give one of the populations the task of being *Geq* estimators, and let it consist of *deterministic* agents rather than randomising ones. This also solves the problem of the minimax solution being unstable, as the solution is the only strategy that is not punished by any pure strategy.

Consequently, we let the Blue population be the one we train towards the optimal game-theoretic strategy. This population then consists of individuals with a representation that allows them to employ mixed strategies. In practice, this means that the output of each Blue agent in a game state should be a vector of nonnegative real numbers that sum to unity; this vector is interpreted as the agent's probability distribution for choosing between the available actions. When playing the game, the agent picks a random action using this distribution.

The Red population consists of individuals that are only able to play pure strategies, that is, in a given game state each Red individual always chooses the same action. Note that it is not necessary to devise another design and representation for this purpose. We may use the same as for Blue, and just change the interpretation of the output vector, so that the Red agent always chooses the action associated with the highest value. (In the case of ties between two or more actions, we may use an arbitrary policy for choosing between these, as long as it is consistent – this is necessary for maintaining the determinism of the agents.)

In order to ensure that the learning task for Blue gets monotonically more difficult over time, forcing it towards the optimum, we use a *hall of fame,* consisting of effective pure strategies found during training, for the Red population. This device has also been used for similar reasons in [12].

**The Algorithm.** The algorithm itself runs as follows: After initialising the populations with individuals having the properties described above, we use some method – such as a random draw, a heuristic or a simple tournament – for selecting a Blue individual that we designate as our nominee for the currently "best" Blue player. Then the following procedure is repeated (cf. Figure 1):

- Train the Red population for a few generations; the fitness measure for each individual is its performance against the Blue player currently nominated as best;
- Add the Red individual coming out on top after this training to the hall of fame;

- Train the Blue population for a few generations; the fitness measure for each individual is its performance against the member of the Red hall of fame which is most dangerous to that Blue individual;
- Nominate the Blue individual coming out on top after this training as the currently best Blue player.

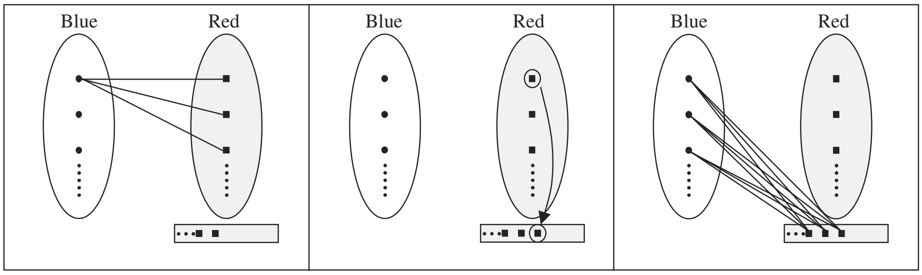

**Fig. 1.** Algorithm for asymmetric co-evolution

The goal of the Blue training is to find individuals that randomise between pure strategies in a way that makes it impervious to exploitation by the dangerous Red agents found; this drives the Blue agents towards the optimum. The Red training amounts to searching for a hole in the defence of the best Blue agent, thus giving the Blue population a chance to mend this flaw in the next training cycle. The metaphor of *hosts* and *parasites* [12] is particularly fitting in this setting, more so than in the symmetric cases where it is otherwise used. A host needs to guard itself against a broad variety of parasites, whereas a parasite is more than happy as long as it can break through a single host's defence. The parallel to the asymmetric layout of our algorithm should be obvious.

As for worst-case fitness (Section 3.1), it is necessary to play several games for each pair of players to obtain good performance estimates, due to the randomisation performed by the Blue agents (see also Section 5).

## 4 Experiments

The purpose of the experiments reported in this section is to illustrate the claims made about the different co-evolutionary designs discussed above. Therefore, we have applied the designs to a toy problem for which the solution is known, namely a modified version of the game Undercut. Furthermore, in order to factor out the effect of inaccurate performance estimates from our investigation of the designs themselves, we have used calculated expected results in our fitness assignments instead of sampled estimates.

Some standard terminology of evolutionary computation is used in the descriptions below; see e.g. [8] for definitions and explanations.

### 4.1 The Game of Zero-Sum Undercut

The two-player imperfect-information game of Undercut was invented by Douglas Hofstadter [3]. The rules are as follows: Each player selects a number between 1 and 5 inclusive. If the choice of one player is exactly one lower than that of the opponent (the player "undercuts" the opponent), the player receives a payoff equalling the sum of the two numbers. Otherwise, each player receives a payoff equalling his own choice. To make the game more challenging, we expand the available choices to the numbers from 1 through 30.

Undercut is clearly not zero-sum; we make a zero-sum version by changing the payoff structure somewhat. A player undercutting his opponent receives the sum of the choices from the opponent; if there is no undercut, the player with the highest choice receives the *difference* between the choices from the opponent. If, for example, Blue plays 14 and Red 22, Red wins 8 from Blue (i.e. Blue gets payoff –8, Red gets 8); if Blue plays 26 and Red 27, Blue wins 53 from Red. As the game is symmetric, its value is clearly zero; thus, the optimal *Geq* evaluation is also zero. The worst possible *Geq* score, incidentally, belongs to the strategy of always playing 30; the most effective counter-strategy is always playing 29, and the minimum score is –59.

The game can be solved using techniques like linear programming [14] or fictitious play [7]; the probability distribution of the solution is given in Table 1. (Choices not appearing in the table should not be played.)

**Table 1.** Solution of zero-sum Undercut with 30 choices

| **Choice** | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|
| **Probability** | 0.095 | 0.084 | 0.151 | 0.117 | 0.161 | 0.110 | 0.135 | 0.069 | 0.078 |

### 4.2 Experimental Setup

For the experiments reported here, the behaviour of each individual was specified by a string of 30 real numbers in $(-1,1)$. These numbers naturally represent the probability of making the corresponding choices; to map the string into a valid probability vector, all negative entries are set to zero and the rest normalised to sum to unity. (Note that this does not affect the string itself.)

The population sizes were set to 50; 500 generations were completed for each population. Tournament selection was used for selecting parents for the genetic operations. For each pair of parents a genetic operator was chosen at random to produce two offspring; the operations and probabilities used were:

- Uniform crossover (probability ½): for each position in the string, distribute the two parent values randomly between the children;
- Average crossover (probability ¼): for each position in the string, let $p$ and $q$ be the two parent values, and set the offspring values to $(2p+q)/3$ and $(p+2q)/3$.
- Mutation (probability ¼): the children are copies of the parents, except that each string position is changed to a random number in $(-1,1)$ with probability 1/15.

Elitism was used; the two individuals with the highest fitness survived from one generation to the next.

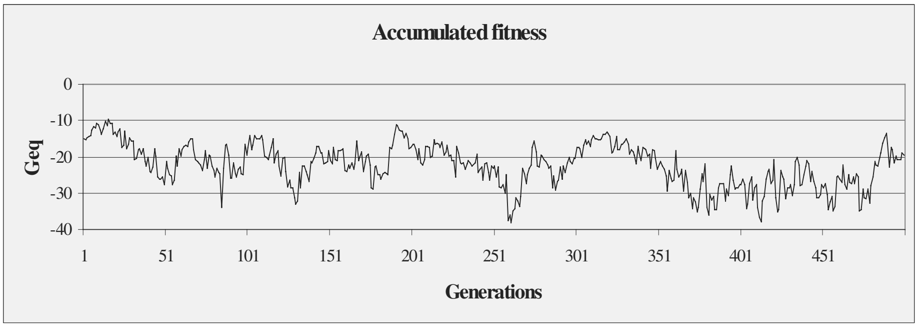

**Fig. 2.** *Geq* for the best individual of each Blue generation, using accumulated fitness

## 4.3 Results

We now present the results of applying the different co-evolutionary designs to the game of zero-sum, 30-value Undercut. We evaluate the training using the *Geq* criterion; the optimal value is then zero.

**Accumulated and Worst-Case Fitness.** Figure 2 shows the *Geq* of the best Blue individual of each generation when using symmetric co-evolution with accumulated fitness, averaged over five runs.
It is clear that this form of learning does not work given our goal; the reason is the lack of transitivity between strategies, as described in Section 3.1. Simply put, there is no incentive to move towards the optimum for either population, as long as the most effective strategy for exploiting the vulnerabilities of the opposing population is itself equally vulnerable.

When using worst-case fitness, the co-evolution produces better individuals than in the case of accumulated fitness, but still does not converge towards the optimum (Figure 3; notice the difference in scale compared to Figure 2).

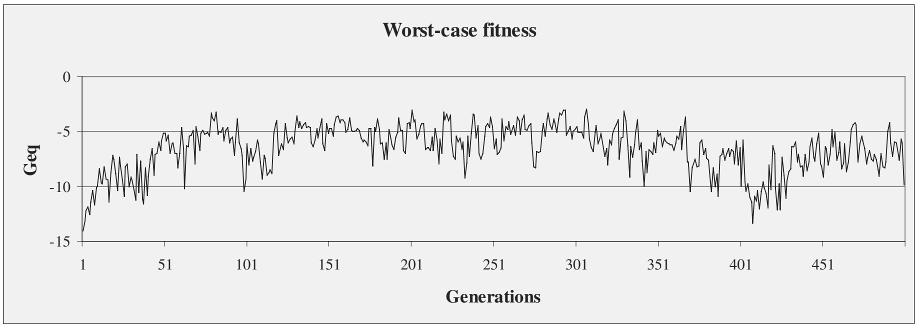

**Fig. 3.** *Geq* for the best individual of each Blue generation, using worst-case fitness

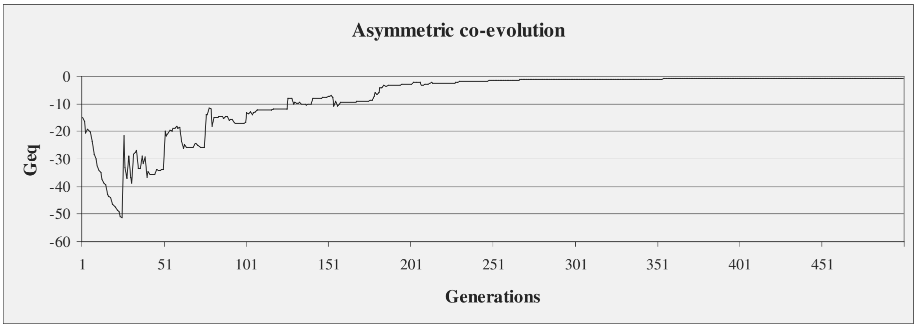

**Fig. 4.** *Geq* for the best individual of each Blue population, using asymmetric co-evolution

The reason for the improved performance is that worst-case fitness corresponds far better to game-theoretic evaluation than does accumulated fitness. On the other hand, the non-coerciveness of minimax play hinders a stable improvement of the agents.

**Asymmetric Co-evolution.** In the case of our asymmetric design of Section 3.2, we let each population train for 25 generations within each main iteration, and ran 20 of these iterations, so that the total number of generations for each side was the same as for the other designs. The results for the best Blue individual of each generation, again averaged over five runs, are shown in Figure 4.
The algorithm clearly pushes the Blue population towards better performance; the improvement gets more monotonic the more members are present in the Red hall of fame. This is due to the increased correspondence over time between Blue fitness and the *Geq* measure, as the Red hall of fame is filled with parasites that are dangerous to the various Blue strategies that may occur.

Note also that although the number of generations for each side is the same as for the symmetric designs, the total number of actual Blue–Red match-ups is much smaller than in those cases. Each Red agent always trains against *one* Blue strategy instead of a whole population, while the Blue agents are trained against a hall of fame, the size of which starts at one and increases over time. Of course, when training proceeds further, the number of opponents for each Blue agent will grow.

## 5 Discussion

The results of our experiments bear out what was said in Section 3 about the various designs for co-evolution of game-playing agents. In particular, they show that the idea that co-evolution works by setting up an "arms race" between the populations is not necessarily sound – for an arms race to take place and give the desired results, we require games in which there is a good correspondence between the true strength of the players (measured game-theoretically) and who beats whom. This is often not the case; in imperfect-information games this correspondence can be especially poor.

Therefore, we require a mode of fitness evaluation that enforces such a correspondence; our asymmetric design has this property.

In the experiments, we used a simple game where the solution is known, and used the calculated expected results of the match-ups in the fitness calculations, instead of results from actual game play. This was done to give a noise-free validation of our claims about the different co-evolutionary designs; for our method to be of practical interest – i.e. in games where the solution is not known – we obviously need to estimate the expected results by playing repeated games for each match-up. This, along with the fact that the number of matches in each generation increases, makes the algorithm relatively expensive in computational terms. This is, of course, a general problem with evolutionary algorithms, as these are rather blind searches compared to methods that glean information about the fitness terrain in more systematic ways.

The question, then, is when and why we should use co-evolutionary approaches, rather than more informed methods? One obvious answer is that they may be useful when other approaches fail, for instance when it is difficult to find agent representations amenable to other machine-learning techniques. Another situation in which co-evolution (and, indeed, evolution in general) is useful is when we specifically desire to use a certain representation that does not lend itself well to other approaches. As an example, we mention that we have work in progress on a far more complex game, where the individuals are small computer programs for playing the game, and the method of evolution is genetic programming [5]. The point of using this non-parametric representation is to evolve game-playing policies that are semantically understandable to humans; neural-net training, for instance, does not produce this kind of information in a readily accessible way.

All of our claims and conclusions in this paper are based on the goal of training agents that are strong in the game-theoretic sense; their ability to randomise strategies in a minimax-like way is the criterion for evaluation. We have already touched upon certain problems with this view, in particular the instabilities connected to the defensive nature of minimax strategies. This defensive approach may seem counter-intuitive to humans, as the goal of these strategies is to randomise in such a way as to be invulnerable to a possibly more intelligent opponent. Furthermore, they do not use information about their opponents, for instance information gleaned from previous games. A minimax strategy has only a weak ability of punishing vulnerable opponents; in fact, it is only expected to win if the opponent performs actions that are not a part of the optimal mixed strategy. Some other research, such as the work on poker reported in e.g. [13], has the more ambitious goal of using opponent modelling for exploiting the weaknesses of other agents. While there are certain problems with this approach, such as the lack of theoretically sound performance measures, the work is indeed very interesting. In a nutshell we can say that an agent trained in this way assumes that is can become more intelligent than its opponents, and thus be able to beat them, while a minimax-trained agent assumes that it will meet more intelligent strategies, and prepares for the worst.

## 6 Conclusion

We have presented an asymmetric co-evolutionary learning algorithm for imperfect-information zero-sum games. This algorithm has been designed so that the fitness of the individual agents is calculated in a way that is compatible with the goal of game-theoretic optimality. This compatibility has been somewhat lacking in previous co-evolutionary approaches, as these have often depended on unwarranted assumptions about the absolute and relative strength of players. Our algorithm is seen to work well on a toy problem for which the optimal strategy is known.

## References

1. Angeline, P.J., Pollack, J.B.: Competitive environments evolve better solutions for complex tasks. In: Forrest, S. (ed.): *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo (1993) 264–270.
2. Halck, O.M., Dahl, F.A.: On classification of games and evaluation of players – with some sweeping generalizations about the literature. In: Fürnkranz, J., Kubat, M. (eds.): *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing*, Jozef Stefan Institute, Ljubljana (1999).
3. Hofstadter, D.R.: *Metamagical Themas. Questing for the Essence of Mind and Pattern*. Basic Books, New York (1985).
4. Koller, D., Pfeffer, A.: Representations and solutions for game-theoretic problems. *Artificial Intelligence* **94**(1) (1997) 167–215.
5. Koza, J.R.: *Genetic Programming. On the Programming of Computers by Means of Natural Selection.* MIT press, Cambridge, Massachusetts (1992).
6. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, New Brunswick (1994) 157–163.
7. Luce, R.D., Raiffa, H.: *Games and Decisions*. Wiley, New York (1957).
8. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended edition.* Springer-Verlag, Berlin–Heidelberg–New York (1996).
9. von Neumann, J.: Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen* **100** (1928) 295–320.
10. von Neumann, J., Morgenstern, O.: *Theory of Games and Economic Behavior*. Princeton University Press, Princeton (1944).
11. Pollack, J.B., Blair, A.D.: Co-evolution in the successful learning of backgammon strategy. *Machine Learning* **32**(3) (1998) 225–240.
12. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. *Evolutionary Computation* **5**(1) (1997) 1–29.
13. Schaeffer, J., Billings, D., Peña, L., Szafron, D.: Learning to play strong poker. In: Fürnkranz, J., Kubat, M. (eds.): *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing*, Jozef Stefan Institute, Ljubljana (1999).
14. Strang, G.: *Linear Algebra and Its Applications. Second Edition.* Harcourt Brace Jovanovich, Orlando (1980).