1-1-2012

Noise Tolerance for Real-time Evolutionary Learning of Cooperative Predator-Prey Strategies

Mark Wittkamp

Luigi Barone

Philip Hingston
Edith Cowan University

Lyndon While

10.1109/CIG.2012.6374134

Noise Tolerance for Real-time Evolutionary Learning of Cooperative Predator-Prey Strategies

Mark Wittkamp, Luigi Barone, Philip Hingston, and Lyndon While

***Abstract*—Learning team-based strategies in real-time is a difficult task, much more so in the presence of noise. In our previous work in the Prey and Predators domain we introduced an algorithm capable of evolving cooperative team strategies in real-time using fitness evaluations against a perfect opponent model. This paper continues our work within the same domain, training a team of predators to capture a prey. We investigate the effect of varying degrees of opponent model noise in our learning system. In the presence of and in the effort to mitigate the effects of such noise we present modifications to our baseline system in the forms of Rescaled Mutation, Conservative Replacement and a combination of the two techniques. The results of the modifications are extremely promising. The combined approach in particular real-times a vast improvement and decreased variance in the performance of our team of predators in the presence of opponent model noise. Additionally, the noise-mitigating strategies employed do not adversely affect the performance of the real-time team learning system in the absence of noise.**

I. Introduction

Games are often used as test-beds to further the development of computational intelligence techniques. They are suitable for this task because they involve similar problems to those encountered in real life, but are simpler and more clearly defined, generally with a well understood goal. Video games present a particularly interesting problem domain because they typically have a much larger number of actions available for players to make with these actions often having temporal significance. The development of adaptive behaviour using opponent modeling with evolutionary algorithms has been real-timed before [1], [2], but the problem becomes far more difficult when we require the learning to occur in real-time, as the game itself is being played.

Artificial players that train offline (generally by playing the game) can have a near limitless amount of training time available to them. The learning and fine tuning of artificial players could run continuously for many days or weeks until desirable behaviours have been found. Contrast this with real-time learning, where there is very little time to run simulations and the processor must also be shared with the game engine itself. Computational intelligence techniques require many iterations and many more test cases for the evolution process to yield desirable results. In order for a real-time approach to be feasible, standard computational intelligence techniques need to be sped up.

In our previous work [3] we created a learning framework that, through continuous adaptation and without any prior learning, learned to coordinate a team of predators to catch a single prey opponent. Beginning with randomly generated predator strategies, our system was capable of outperforming an offline approach in many tests by evolving and replacing predators one strategy at a time. Since our learning framework is able to learn and replace the predator strategies continuously, each individual strategy need not be overly complex — this is what helps make learning in such a short space of time possible. Through continuous adaption, many short term strategies piece together to form a more complex overall playing strategy.

Our previous work assumed a perfectly accurate model of the opponent's behaviour in order to learn a strategy for the next "time-slice". In this paper, we relax the assumption of a perfect opponent model and examine the behaviour of the system under varying amounts of error in the prediction of the opponent's location. Rather than aiming to reduce the amount of noise in the opponent model, in this paper we work towards improving the performance of our system in its presence. Creating an accurate opponent model (in this case, of the prey) is often difficult and obtaining a perfect opponent model may not always be possible – a model of a human player, for example. We will see in Section V that noise does indeed cause a problem for learning. Section VI discusses our approaches to learning in the presence of noise and real-times a great improvement in noise tolerance for our learning framework.

A. The Case For Real-Time Learning

Despite a large amount of research in the field of video game AI, the majority of AI strategy in commercial games is still in the form of scripted behaviour [4]. Developers turn to scripts for a number of reasons; they are understandable, predictable, easy to modify and extend, and are usable by non-programmers [5]. Scripts often have parameters that may be optimised using computational intelligence techniques offline, but the learning aspect is rarely a component in the released product [6].

While scripts can respond to the actions of human players, artificial agents (or "bots") are often inherently exploitable due to their inability to adapt. Once an agent's weakness has been discovered it can be exploited time and time again and soon the game fails to remain challenging or realistic and human players may lose interest. No matter how thorough the training process, in many modern games there are too many possible scenarios to expect that a hand-coded player will be able to handle them all equally well.

Scripted bots and their predetermined behaviour are susceptible to being overly repetitive or unrealistic, especially if the bots find themselves in a situation that the developers did not foresee. Stochastic systems can be used to introduce some

variety into the behaviour of artificial players, but they may offer only slight variation to some predetermined strategy. Too much variation has the potential for creating seemingly random or irrational behaviour which adversely affects a human player's sense of immersion in the game environment.

Another common limitation of current game AI is that teams of agents tend to be overly self-interested. While many good agents may be useful for a team, this is very different from team-interested agents who can understand and prioritise the good of the team over individual gain. Without team based learning, artificial players run the risk of being overly "greedy" to the detriment of the team. No matter how well the individual parts may be tuned, certain team strategies may never arise — a self-interested individual would not sacrifice itself to draw fire away from team-mates or to lead opponents into an ambush, for example. Team based learning is useful where the goal to be accomplished is too complex to be achieved by individuals lacking team coordination, RoboCup soccer [7] is a good example.

The real-time learning and continuous adaptation of a team of artificial agents is desirable for a number of reasons. An agent capable of real-time learning would be inherently robust just as strategies learnt offline are inherently exploitable. Ideally, an adapting agent could be expected to perform in situations never considered by the game developers. Our previous work in a prey and predators domain [3] learned cooperative predator strategies in real-time using a perfect opponent model of the prey for simulations. This paper takes the next step and investigates to what extent team learning is possible in real-time when faced with the more realistic scenario where only an imperfect opponent model is available.

II. The Iterative Real-time Team Learning Framework

Our real-time learning framework is a novel implementation of an Evolutionary Algorithm, designed to run in parallel with the game environment and to iteratively evolve a team of agents via an analogy of Darwinian selection. Learning takes place continuously within discretised time slices; during each time slice, a role is selected for training.

The system first looks ahead to the predicted state at the start of the next time-slice (ES_{t+1}). This state is used to determine which role to train and from which population (each role maintains its own population). Each time-slice, a single role is trained in a round-robin fashion. How these roles map to the agents is up to the implementation, but for this study we use a direct one-to-one mapping of each role to a unique predator so the terms "role" and "predator" strategy may be used interchangeably. It may be advantageous to organise the mapping of roles to predators in a more meaningful way (such as by distance to the prey) and then automatically switch the strategies used by predators as their circumstances change, but we ignore these considerations in this work.

The look-ahead state (ES_{t+1}) contains the expected state of the environment and all agents one time-slice into the future. The learning framework has access to the behaviour of the other predators in the teams but not the prey strategy — only to a noisy approximation of the prey is available. When training a particular role, the role is replaced in the look-ahead state and then a simulation one time-slice length in the future from the look-ahead state (until ES_{t+2}) is completed. Even though only a single predator is training during any given time slice, the fitness measure used evaluates the team as a whole rather than sanctioning the individual directly. The individuals in the population are each evaluated by their contribution to the predator team's predicted performance at the end of the next time slice.

The evolutionary process takes place in real-time, in parallel with actual events in the game environment. The fittest individual is used to replace the role currently undergoing training in the next time slice. The evolutionary algorithm uses this performance data to create successive generations of strategies for that predator as long as time permits — that is, until the game reaches S_{t+1}. We use the same fitness function as that of [8] as described below where d_0 is the sum of all predator's starting distances to the prey, and d_e is the sum of the ending distances. The framework is depicted visually in Figure 1 and written up as pseudo-code in Algorithm II.1.

$$f = \begin{cases} d_0 - d_e/10 & \text{if prey not caught} \\ 200 - d_e/10 & \text{if prey caught} \end{cases}$$

In this paper, we compare a number of modifications to the baseline approach presented in [3]. The baseline system uses an elitist selection scheme where the top half of the population reproduces by one-point cross-over and mutation to replace the bottom half of the population. Mutation is applied randomly to a single weight of the individual, with 0.1 strength. We cap our simulation time at 2000 game ticks (roughly 40 seconds). Though we hope to complete a capture far sooner, we allow the simulations to run up to this length for data collection purposes and to give our predators an opportunity to learn in the presence of high levels of noise in the opponent model.

While playing, the predators do not explicitly communicate. They learn to cooperate implicitly via the fitness function which rewards an individual strategy based on the performance of the team rather than some measure of individual contribution. For the look-ahead and training simulations the predator currently undergoing a learning cycle has implicit access to each other predator's model. It also has access to a "noisy" opponent model representing the strategy of the prey — this is described in detail in Section III-B.

In our previous work, we experimented with various time-slice lengths. We found that with longer time-slice lengths (with a perfect opponent model) the system was able to produce more effective capture strategies — often learning to capture with only 2 trained predator strategies cooperating with a 3rd, randomly initialised, predator strategy. The trade-off is that it takes longer for each predator to be given an opportunity to learn and a longer wait between strategies

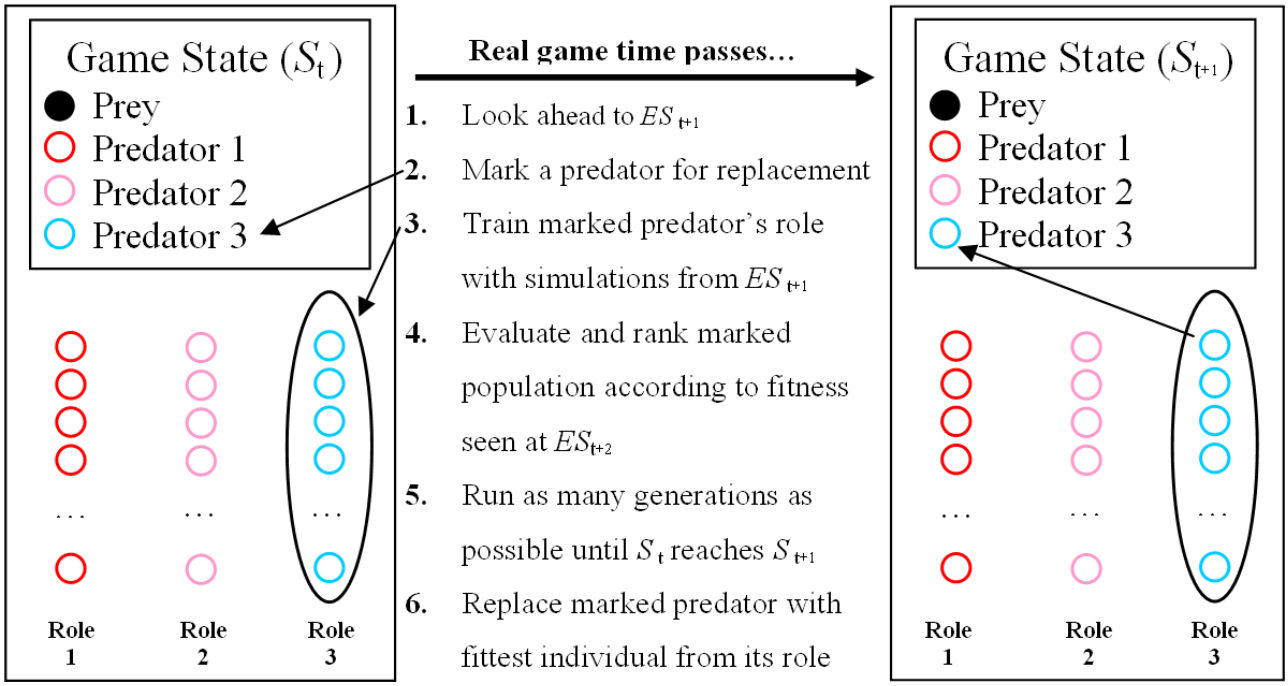

Fig. 1. Pictorial representation of the real-time learning framework

being updated.

At each game-tick, the opponent model gives an approximation of the prey's position which may be out by some degree. Both the look-ahead and simulation phases are affected by noise in the opponent model — the longer the runs, the more potential there is for the predicted behaviour of the prey to vary from its true behaviour. Using an imperfect opponent model would introduce a further disadvantage to having a long time-slice length and would make comparisons more difficult. We experimented with the idea of having a variable time-slice length to deal with varying levels of noise, but the optimal time-slice was highly variable upon other factors such as the strength of the prey opponent. The experiments in this paper all use a fixed time-slice length of 60 game-ticks.

III. Experimental Domain

We have developed a system for learning effective team strategies in real-time as a game is being played. We allow for no prior offline learning; all learning takes place while the game is being played. To test our system, we use the prey and predators domain studied in [9], [8]. We are evolving a team of predator strategies to coordinate their movements to trap and capture the prey in real-time. In this paper we extend the problem by no longer allowing a perfect opponent model. We experiment with only providing the learning system with an imperfect model of the prey, building upon our previous work in [3]. The aim is to broaden the effectiveness and generality of our learning framework into situations where it may be difficult or infeasible to construct an accurate opponent model.

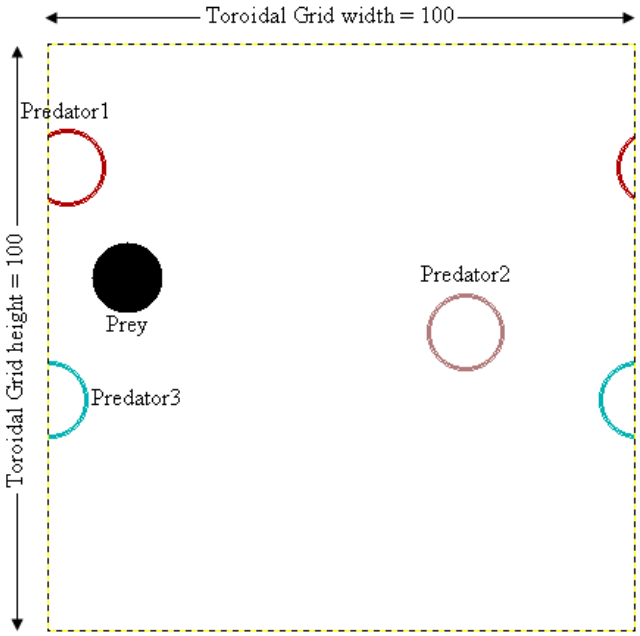

Fig. 2. The prey and predators environment

A. Prey and Predators Environment

The game environment we use is closely modelled from that of [8]. In this predators-prey environment, we have a single prey and a team of 3 predators. The goal of the predators is to catch (making contact with) the prey. The prey's aim is simple; avoid being caught by the predators.

We are interested in training the team of predators in

Algorithm II.1: REAL-TIME EVOLUTIONARY TEAM LEARNING SYSTEM( )

comment: Initialise a population (P_r) of individuals for each identified role (r)

for each $r \in Environment.Roles$
do $\{P_r \leftarrow$ CREATEPOPULATIONOFINDIVIDUALS( )

for each $t \in Time - slices$
do
- **comment:** Capture the current state of $Environment$ to S_t
- $S_t \leftarrow Environment.$GETSTATE( )
- **comment:** Look ahead from the captured state to the next expected state ES_{t+1}
- $ES_{t+1} \leftarrow$ LOOKAHEAD$(S_t, OpponentModel)$
- $MarkedRole \leftarrow$ CHOOSEROLE(ES_{t+1})
- **for** $g \leftarrow 1$ **to** $NumGenerations$
  - **do**
    - **in parallel for each** individual $i \in P_{MarkedRole}$
      - **do**
        - $StartStates[i] \leftarrow ES_{t+1}$
        - $StartStates[i].$REPLACEROLE$(Environment.Roles[MarkedRole], P_{MarkedRole}[i])$
        - $ES_{t+2}[i] \leftarrow$ RUNSIMULATION$(StartStates[i])$
    - **comment:** Evaluate $P_{MarkedRole}$ by inspecting expected end states (ES_{t+2})
    - Fittest $\leftarrow$ EVALUATE$(P_{MarkedRole}, ES_{t+2})$
    - **comment:** Evolve the next generation of individuals for P_r
    - EPOCH(P_r)
- $Environment.$REPLACEROLE$(Environment.Roles[MarkedRole], Fittest)$

real-time to cooperate with each other towards the goal of catching the prey. The prey and predators move at the same speed, thus making the task of capturing any competent prey impossible without some degree of cooperation. We use the hand-coded *Repelled* prey described in Section III-B as a training partner for our team of predators in all experiments.

The environment for all experiments is a 100×100 toroidal grid without obstacles; agents (prey and predators) are represented by circles of radius 6. In this environment a simple hand-coded prey could quite easily evade 2 predators indefinitely, thus the task of capturing the prey will need the cooperative actions of all 3 predators working together. The initial setup places the 3 predators in a corner of the toroid grid (being a toroid, they are all one and the same) and the prey is randomly positioned such that it is not initially in contact with any predator. The speed of each agent in the game is fixed — the prey and all predators are either moving at this speed or stationary; there is nothing in between.

B. Hand-coded Prey Controllers

In our previous work, we created hand-coded opponents capable of evading the predators to varying ability. The simplest prey was based on the description provided in [8] – it was able to move at the same speed as the predators and its strategy was always to head in the direction opposite to the nearest predator. We found that when using this prey as a training partner, the predators easily captured the prey very quickly and our learning system was not sufficiently challenged. We created a more complex player which we named the *Repelled* prey — a vector based approach that avoids all predators proportionate to their proximity which we confirmed provided a more difficult capture task.

In this work, we use the *Repelled* prey as a training partner for all experiments. The algorithm used by the Repelled prey is as follows. For each of the 3 predators, the Repelled prey applies a force of repulsion equal to $1/d^2$ in the direction of the predator, where d is the minimum toroidal distance from the prey to that predator. This prey moves at the same speed as the predators, heading in a direction determined by the

sum of the repulsive forces (but always moving at maximum speed).

To simulate noise, we have a "noisy" version of the Repelled prey with an adjustable level of adherence to the prescribed behaviour of the original. The noise is in the form of a Gaussian random variable of zero mean and a standard deviation as a proportion of π applied to the direction in which the prey is headed (measured in radians) — we call this proportion the *noise amplitude* of the opponent model. This is how we simulate opponent model noise in our game — the game in progress is one against the Repelled prey, but our learning algorithm only has access to our prey opponent model. For example, when we describe an opponent model having a noise amplitude of 0.2π, this means that a Gaussian random variable of 0.2π (or $36degrees$) is added to the model's direction at each and every decision point (game tick).

Our system's knowledge of the starting position of the opponent is never guaranteed to be completely accurate. At the beginning of each time-slice, the system knows the exact state of the game. However, since our system is training for play beginning in the *next* time-slice, the starting point involves one time-slice of potential error during the look-ahead phase. Due to noise in the opponent model, the perceived and true states deviate further still as the training simulations progress.

C. Predator Controller

A predator takes the form of a randomly initialised feed-forward neural network with 2 inputs, 5 outputs, and a hidden layer of size 10. The only inputs to the predator's network are its x and y displacement relative to the prey. The predator's actual x and y coordinates on the toroidal grid do not factor into its decision making process, nor need it be given the homogenous nature of the environment. The outputs of the network are *North*, *South*, *East*, *West* and *Stationary*.

Predators do not have any explicit communication or knowledge of where their team mates are. This allows the representation of an individual to be very simple and allows a smaller search-space to cover, which is particularly useful given that our system learns in real time. Information regarding the location of the predator's team mates is implicitly provided by means of the fitness function. Recall from Section II that an individual's fitness is based on how well the whole team performs and cooperates together with this new individual.

A predator will remain still if the *Stationary* output exceeds that of all other outputs. Otherwise, the difference between the East and West outputs determines the x component of the predator's direction vector and the difference between North and South determines y. While this network representation could be used to define an agent that is capable of varying its speed, here we are only using it to describe the predator's direction, not magnitude. The predators all travel at a fixed speed, equal to that of the prey. Like the prey, predators are either completely motionless or travelling at their fixed speed.

IV. The Baseline Learning Framework

Our previous work [3] tested the performance of a learning framework for the real-time learning of strategies in the prey and predators domain against a hand-coded prey opponent. We showed that the resulting real-time team strategies are able to capture hand-coded prey of varying degrees of difficulty without any prior learning. The system is highly adaptive to change, capable of handling many different situations, and quickly learning to function in situations that it has never seen before.

In developing our system we first developed a version that ran in "simulated real time", where the game would pause and allow the evolution of a strategy to complete to a predetermined number of generations. When extending our system into real-time (where the number of generations is severely limited), we observed no statistically significant drop in performance for any but one experiment. Since our learning framework is able to learn and replace the predator strategies continuously, each individual strategy need not be overly complex — this reduces the total search-space and is what helps make learning in such a short space of time possible.

In [3] we compared our framework with that of an offline approach described in [8], in the prey and predators domain. It soon became apparent that the prey previously discussed was too simple a task for our system — a 100% capture rate was routinely observed in most experiment scenarios. We then went on to real-time the framework's effectiveness against 2 more types of prey, one of which being able to travel at triple the speed of the predators. These successes gave us confidence in our framework but the assumption of a perfect opponent model is one that needed to be addressed. A learning framework that only works upon first ensuring a perfect opponent model is not particularly useful in many real-world scenarios. While it is common to strive towards having accurate opponent models to train against, attaining one that is absolutely perfect is often infeasible or even impossible.

V. Learning in the Presence of Opponent Model Noise

The problem of noise makes learning difficult. There are many forms of noise — for example, noise in the environment model, noise in the opponent model, and noise in the fitness function. In this paper we experiment with noise in the opponent model. Obviously it is desirable to have as accurate an opponent model as possible, but opponent modeling to a high degree of accuracy may often be difficult or even impossible. Additionally, results gained from such an exercise have a risk of being overly domain specific and difficult to generalise. Rather than focusing on improving the opponent model of our prey, this paper will focus on building up tolerance to noise in the opponent model of our real-time learning system.

Compensating for noise is a difficult problem [10], [11]. Some general approaches to learning in the presence of

noise involve resampling, avoiding convergence by reducing elitism and to encourage diversity by widening the search space. We have tried a number of different "off-the-shelf" approaches to modifying our system in an attempt to make it more tolerant to noise in the opponent model as well as more specialised variations or combinations of approaches. In this paper we report on two of our approaches as well as their combined effect in improving our learning system's tolerance to opponent model noise.

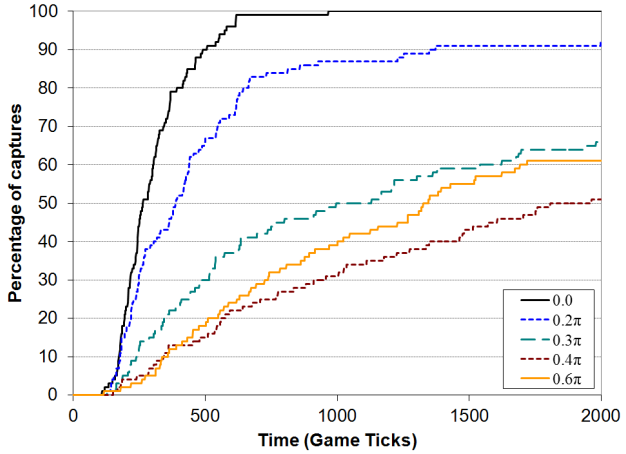

Fig. 3. The effect of opponent model noise for the baseline system

Figure 3 real-times the deterioration of performance of the baseline system in the presence of increasing amounts of opponent model noise. The plot for 100% accuracy real-times results similar to those reported in [3], with the other plots demonstrating that level of performance is no longer achievable in the presence of opponent model noise. The baseline learning system seems to be at least somewhat tolerant to noise in the opponent model — that is, the system doesn't completely fall apart in the presence of noise. At a noise amplitude of 0.2π opponent model accuracy, the capture rate falls from 100% to a little over 90%. We repeated the experiment with even smaller amounts of noise and found that even a very small amount of opponent model noise such as 0.04π results in an observable decrease in performance (albeit a small one). Performance suffers a rapid degradation when the noise amplitude is raised to 0.3π with a resulting capture rate of only 66%.

For the baseline system, the variance in performance increases with the amount of noise in the opponent model. Our plots only show up to a noise amplitude of 0.6π because any more noise than this drives the variance up so high as to make meaningful comparisons difficult. The next section reports on our attempts to mitigate the effects of opponent model noise. Our aim is to modify our existing framework such that it will be more tolerant to noise in the opponent model.

VI. Compensating for the Effects of Noise

In this section, we report on modifications to our baseline system designed to improve our Iterative Real-time Team Team Learning System in the presence of opponent model noise. We report on our implementations of Rescaled Mutation [], Conservative Replacement [], and a combination of the two techniques.

A. Rescaled Mutation

Rescaled Mutation [12] is based on the principle of "mutate large, inherit small". Individuals are mutated by a large amount so that we may sample more distant strategies in an attempt to avoid converging towards local optima when the opponent model noise is high. When these distant strategies are found to be successful they do not, however, become part of the population. Rather, they act as a way of potentially driving the evolution towards these strategies. If we consider a successful distant mutation to be a vector deviating from the original, then the same vector of lesser magnitude (a "lesser mutant") would join the population on its behalf. Our implementation of this approach applies a Gaussian mutation of standard deviation 0.4 to a randomly selected weight — four times as large as the amount of mutation used in the baseline approach and for the "lesser mutant" that ultimately may join the population.

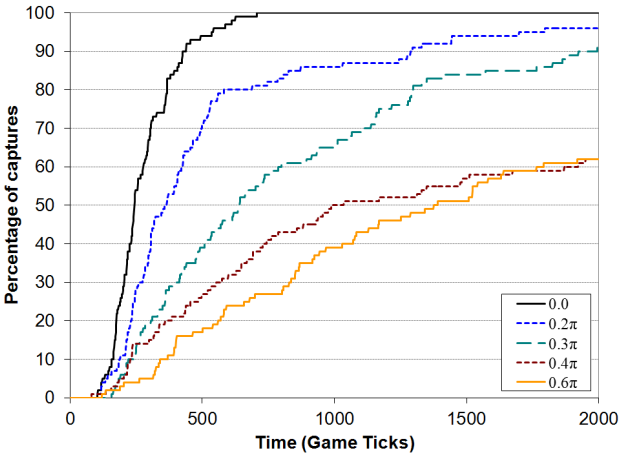

Fig. 4. The effect of opponent model noise using Rescaled Mutation

Somewhat surprisingly, the Rescaled Mutation system performs as well as the baseline approach when a perfect opponent model is available. This suggests that our learning system using Rescaled Mutation is an absolute improvement to the baseline rather than some special feature that needs to be carefully toggled on or off at varying degrees of perceived opponent model noise. This is a fortunate result because the need to detect and automate a switch according to the level of noise would introduce its own share of problems.

Rescaled Mutation has made our real time team learning system far more tolerant to opponent model noise. Figure 4 real-times a significant improvement over the baseline system. At a noise amplitude of 0.2π, we observe a slight

improvement when using Rescaled Mutation compared to the baseline system. The largest improvement, however, is real-timed in our experiment where the noise amplitude is raised to 0.3π. Here the Rescaled Mutation system achieves an improved prey capture rate; 24% higher than the baseline approach.

B. Conservative Replacement

In the Conservative Replacement approach we begin with the idea that the longer an individual has survived in the population, the more robust it is likely to be to the presence of opponent model noise. In order to replace a more mature individual a new individual must not only be better than an existing individual, but be better by at least predefined threshold. This is to prevent "lucky" individuals from replacing good strategies. If a new individual performs well in a particular run, it may not necessarily imply that it is a better individual — the individual may have be reporting a misleadingly high performance result due to noise.

This approach [] where new individuals do not compete on a level playing field with the rest of the population is often applied in the opposite direction [13], [14] — to protect newly created and potentially innovative but immature strategies from having to compete with the more developed strategies. However, in the presence of high levels of noise, it makes sense not to "trust" new individuals based on a single run. One would expect that after multiple performance evaluations, the system could be more confident in trusting that an individual's fitness is a more accurate reflection of its true ability.

In our implementation of Conservative Replacement, we provide a fitness reward to existing individuals relative to their age by the formula: $reward = |fitness \times \log(age + 1)^2)/200|$ A logarithmic reward function is chosen so that a large reward is given to an individual that has survived one generation as opposed to a new individual, with a slightly lesser additional reward for having survived two generations and so on.

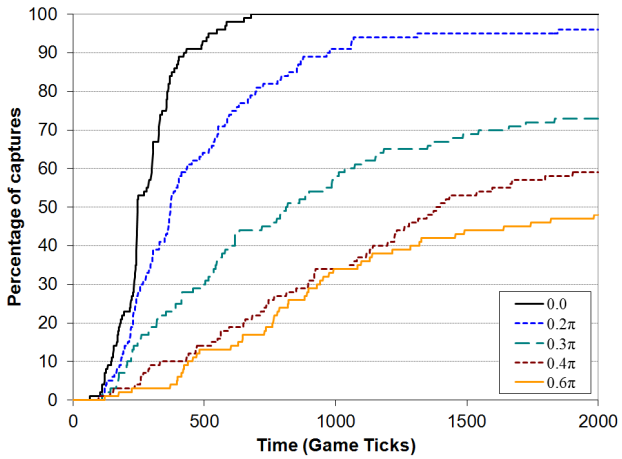

Fig. 5. The effect of opponent model noise using Conservative Replacement

Figure 5 shows the average performance of our system when implemented using Conservative Replacement performs as well as the baseline approach when a perfect opponent model is available. These results suggest that there is never a good reason not to use Conservative Replacement for our system. This is consistent with our result in Section VI-A and those yet to be discussed in Section VI-C. At a noise level of 0.2π this system performs roughly 5% better than the baseline approach.

Our system using Conservative Replacement performs equal or better than the baseline system. At a noise amplitude of 0.3π, the Conservative Replacement system performs about 7% better than the baseline but far short of the impressive result that Rescaled Mutation was able to boast at this level. For the experiments with opponent models of accuracy below 0.4π the results are not statistically any better or worse than that of the baseline system though the performance variance has decreased.

C. Conservative Replacement with Rescaled Mutation

For this experiment we combine both approaches from Sections VI-A and VI-B to dealing with opponent model noise. In this system, we perform Rescaled Mutation as in Section 4, but then only replace an individual if the "lesser mutant" outperforms it by at least some threshold depending on the age of the existing individual and as described in Section VI-B.

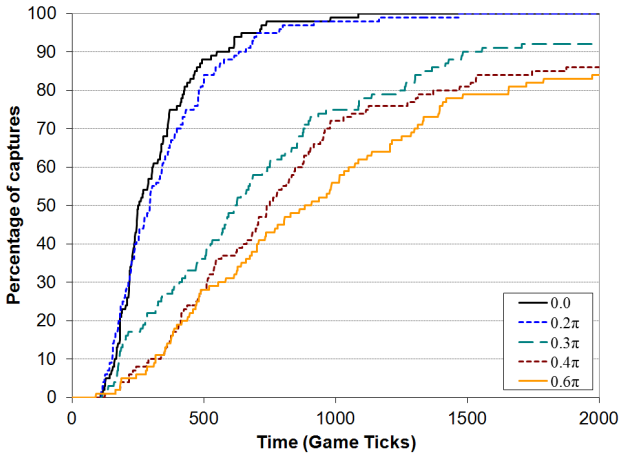

Fig. 6. The effect of opponent model noise using Conservative Replacement with Rescaled Mutation

Figure 6 real-times the performance of our system using the combined Conservative Replacement with Rescaled Mutation approach. This system is a great improvement over our other methods in several areas. Firstly, this combined Conservative Replacement with Rescaled Mutation does not affect performance when there is no opponent model noise. The combined approach allows our system to achieve a perfect capture rate when using an opponent model with a noise amplitude of 0.2π accuracy. This is a huge improvement, indicating that the two approaches are complementary.

For low levels of noise, this combined system performs like a "best case" scenario of our other approaches, managing to achieve the high performance of the Rescaled Mutation approach. At 0.3π noise amplitude the system achieves a capture rate of over 90% which is about the same as what was achieved with Rescaled Mutation but and roughly 25% and 20% higher more than the baseline and Conservative Replacement approaches respectively.

The achievement that stands out the most of this system is the huge gain in performance that it achieves when the noise level is high. In the experiments of the combined Conservative Replacement with Rescaled Mutation approach we real-time a gain in performance to over 80% capture for noise amplitudes of 0.4π and 0.6π. At these high levels of noise, the baseline approach manages captures of around 50 to 60%. The combined approach greatly outperforms both the individual Rescaled Mutation and Conservative Replacement approaches on their own — a little over 50% capture rate and 60% respectively. For the combined approach, the variance of performance is also far lower.

VII. Conclusions

In our previous work, we presented a system for the real-time evolution of team predator strategies in the Prey and Predators domain assuming the use of a perfect opponent model. This paper extends upon this by relaxing this unrealistic assumption and implanting noise of varying degrees into the opponent model of the prey. Our learning system trains using an approximated model of the prey that the team of predators is aiming to capture.

We real-time the deterioration in performance of our baseline system in the presence of noise and compare this with a number of proposed improvements. Even small amounts of noise are reflected in a decrease in performance with larger aount of noise seeing a drop in performance from 100% down to roughly 60%. The increased level of opponent model noise negatively impacts the reliability of the baselines system, with an increased performance variance.

Our attempts at creating a more noise-tolerant system have yielded some great success. When using Conservative Replacement, we witnessed a performance increase in low levels of noise. Some improvement in the higher levels of noise is seen, but the variance at these levels of noise is too high to make meaningful comparisons. The improvements witnessed under the Rescaled Mutation approach were more impressive than the Conservative Replacement approach. An interesting point to note is that at no levels of opponent model noise did either of these proposed ideas negatively impact our learning system.

A combined approach, with Conservative Replacement used in conjunction with Rescaled Mutation was also trialled. This approach yielded extremely positive results. At a noise level of 2π, the combined approach was able to attain a capture rate of 100% compared to 92%, 97% and 96% of the baseline, Conservative Replacement, and Rescaled Mutation approaches respectively. A far greater improvement was achieved for greater levels of noise in the opponent model. At 0.6π (the highest amount of noise in our trial), the system under combined approach was able to achieve a capture rat of over 80% where the other methods only managed around 60%.

Our attempt at achieving a level of noise tolerance for our real-time team learning system has been met with considerable success. This success greatly expands the potential applications for our learning system. The somewhat unrealistic assumption of a perfect opponent model has been removed from our system, leaving in its place a more robust and reliable system in the face of opponent model noise. Future work may involve experimenting with other forms of noise or testing our system in a more difficult problem domain.

References

[1] M. Wittkamp and L. Barone, "Evolving adaptive play for the game of spoof using genetic programming," in *In Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games*, 2006.

[2] M. Wittkamp, L. Barone, and L. While, "A comparison of genetic programming and look-up table learning for the game of spoof," in *In Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, 2007.

[3] M. Wittkamp, L. Barone, P. Hingston, and L. While, "Real-time evolutionary learning of cooperative predator-prey strategies," in *In Proceedings of the 2012 Australian Computer Science Week (ACSW)*, 2012.

[4] M. L. Berger, *Scripting: overview and code generation, in AI Game Programming Wisdom*. MIT Press, 2002.

[5] P. Tozour, *The Perils of AI Scripting*. Charles River Media, Inc., 1995.

[6] D. Charles, C. Fyfe, D. Livingstone, and S. McGlinchey, "Biologically inspired artificial intelligence for computer games," in *Medical Information Science Reference*, 2007.

[7] H. Kitano, A. Minoru, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," *In Proceedings Workshop on Entertaintment Life*, 1995.

[8] C. Yong and R. Miikkulainen, "Cooperative coevolution of multi-agent systems," 2001. [Online]. Available: citeseer.ist.psu.edu/yong01cooperative.html

[9] R. A., R. P., and M. R., "Constructing competitive and cooperative agent behavior using coevolution," in *In Proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games*, 2010.

[10] A. D. Pietro, L. Barone, and L. While, "A comparison of different adaptive learning techniques for opponent modelling in the game of guess-it," in *In Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games*, 2006.

[11] P. Mertikopoulos and A. L. Moustakas, "Learning in the presence of noise," in *Proceedings of the First ICST international conference on Game Theory for Networks*, ser. GameNets'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 308–313. [Online]. Available: http://dl.acm.org/citation.cfm?id=1689499.1689539

[12] H. georg Beyer, "Mutate large, but inherit small! on the analysis of rescaled mutations in (1,λ)-es with noisy fitness data," in *In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, 1998.

[13] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001. [Online]. Available: http://dx.doi.org/10.1162/106365601750190398

[14] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1162/106365602320169811