Learning Communication for Multi-agent Systems

C. Lee Giles[1,2] and Kam-Chuen Jim[2]

[1] School of Information Sciences & Technology and Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16801 USA
giles@ist.psu.edu
[2] NEC Labs
4 Independence Way, Princeton, NJ 08540 USA
kamjim@research.nj.nec.com

Abstract. We analyze a general model of multi-agent communication in which all agents communicate simultaneously to a message board. A genetic algorithm is used to learn multi-agent languages for the predator agents in a version of the predator-prey problem. The resulting evolved behavior of the communicating multi-agent system is equivalent to that of a Mealy machine whose states are determined by the evolved language. We also constructed non-learning predators whose capture behavior was designed to take advantage of prey behavior known a priori. Simulations show that introducing noise to the decision process of the hard-coded predators allow them to significantly ourperform all previously published work on similar preys. Furthermore, the evolved communicating predators were able to perform significantly better than the hard-coded predators, which indicates that the system was able to learn superior communicating strategies not readily available to the human designer.

1 Introduction

Allowing agents to communicate and to learn what to communicate can significantly improve the flexibility and adaptiveness of a multi-agent system. This paper studies an ideal case where each agent has access to a small set of local information and through experience learns to communicate only the additional information that is important. While many researchers have shown the emergence of beneficial communication in multi-agent systems, very few have looked into how communication affects the behavior or representational power of the multi-agent system. This paper shows the relationship between the communication behavior of a multi-agent system and the finite state machine that completely describes this behavior. With this knowledge we demonstrate how evolved communication increases the performance of a multi-agent system.

W. Truszkowski, C. Rouff, M. Hinchey (Eds.): WRAC 2002, LNAI 2564, pp. 377–390, 2003.

1.1 Previous Work

Previous work has shown that beneficial communication can emerge in a multi-agent system. [1] show that agents can evolve to communicate altruistically in a track world even when doing so provides no immediate benefit to the individual. [2] use genetic algorithms to evolve finite state machines that cooperate by communicating in a simple abstract world. [3] study the emergence of conventions in multi-agent systems as a function of various hard-coded strategy update functions, including update functions where agents communicate to exchange memories of observed strategies by other agents. [4] show that vocabulary can evolve through the principle of self-organization. A set of agents create their own vocabulary in a random manner, yet self-organization occurs because the agents must conform to a common vocabulary in order to cooperate. [5] allow agents to communicate real-valued signals through continuous communication channels and evolved agents that communicate the presence of food in a food trail-following task. [6] showed that communication significantly improves performance of robot agents on tasks with little environmental communication, and that more complex communication strategies provide little or no benefit over low-level communication.

While many researchers have shown the emergence of beneficial communication, very few have analyzed the nature of the communication and how communication affects the behavior or representational power of the multi-agent system. [7] developed a "Recursive Modeling Method" to represent an agent's state of knowledge about the world and the other agents in the world. Furthermore, Gmytrasiewicz, Durfee, and Rosenchein used this framework to compute the expected utility of various speech acts by looking at the transformation the speech act induces on the agents' state of knowledge. [8] show that with certain assumptions, communication can be treated as an n-person game, and the optimal encoding of content by messages is obtained as an equilibrium maximizing the sum of the receiver's and speaker's expected utilities. More recently, we have shown that communication increases the representational power of multi-agent systems, and derived a method for estimating the language size for any multi-agent problem ([9], [10]).

2 Predator Prey Problem

The predator-prey pursuit problem is used in this paper because it is a general and well-studied multi-agent problem that still has not been solved, and it is a simplied version of problems seen in numerous applications such as warfare scenarios and computer games. The predator-prey pursuit problem was introduced by [11] and comprised four predator agents whose goal is to capture a prey agent by surrounding it on four sides in a grid-world. [12] used genetic programming to evolve predator strategies and showed that a linear prey (pick a random direction and continue in that direction for the rest of the trial) was impossible to capture reliably in their experiments because the linear prey avoids locality of movement. [13] studied a version of the predator prey problem in

which the predators were allowed to move diagonally as well as orthogonally and the prey moved randomly. [14] used reinforcement learning and showed that cooperating predators that share sensations and learned policies amongst each other significantly outperforms non-cooperating predators. [15] study a simple non-communicating predator strategy in which predators move to the closest capture position, and show that this strategy is not very successful because predators can block each other by trying to move to the same capture position. Stephens and Merx also present another strategy in which 3 predators transmit all their sensory information to one central predator agent who decides where all predators should move. This central single-agent strategy succeeds for 30 test cases, but perhaps the success rate would be much lower if the agents were to move simultaneously instead of taking turns.

We use an implementation which is probably more difficult for the predators than in all previous work:

1. In our configuration, all agents are allowed to move in only four orthogonal directions. The predators cannot take shortcuts by moving diagonally to the prey, as they do in [13].
2. All agents have the same speed. The predators do not move faster than the prey, nor do they move more often than the prey, as they do in [12].
3. All agents move simultaneously. Because the agents do not take turns moving (e.g. [15]) there is some uncertainty in anticipating the result of each move. In addition, moving the agents concurrently introduces many potential conflicts, e.g. two or more agents may try to move to the same square.
4. The predators cannot see each other and do not know each other's location. If this information is essential then the predators will have to evolve a language that can represent such information.

The world is a two dimensional torus discretized into a 30x30 grid. If an agent runs off the left edge of the grid it would reappear on the right edge of the grid, and a similar behavior would be observed vertically. No two agents are allowed to occupy the same cell at the same time, and agents cannot move through each other. If two or more agents try to move to the same square they are blocked and remain in their current positions. At the beginning of each scenario the agents are randomly placed on different squares. Each scenario continues until either the prey is captured, or until 5000 time steps have occurred without a capture.

Two prey strategies are used in the simulations. The Random Prey chooses it's next action at each time step from the set N, S, E, W using a uniform random distribution. The Linear Prey picks a random direction at the beginning of a trial and continues in that direction for the duration of the scenario. It has been shown that the Linear Prey can be a difficult prey to capture [12, 4] because it does not stay localized in an area. In our simulations this is an even more difficult prey to capture because the prey and predators move at the same speed.

3 Non-communicating Predators

The first set of experiments in this section is done using predators with human-designed strategies, while the second set is done using evolved predators.

3.1 Hard-Coded Predators

The Follower agent moves in the direction that minimizes its Manhattan distance to the prey's current location. Ties are broken randomly. [13] used a similar greedy heuristic for a version of the predator prey problem where the agents take turns moving. [12] modified the algorithm to work in the predator prey problem with simultaneous moves. Both of these previous approaches are deterministic and thus suffer from deadlock.

The Herder agent is similar to the Follower agent but takes advantage of coordination at the initiation of each trial. At the beginning of each trial, each agent is assigned a role of either N, S, E, or W Herder. Their task is to move to the respective cell adjacent to the prey in their assigned direction, in effect herding the prey from four sides in an attempt to capture it. The Herder agent uses the same Manhattan distance heuristic as the Follower agent.

The HerderCounter agents attempt to herd the prey like the Herder agents, but takes advantage of *state information* by counting the number of times the prey has moved in the same direction. This count is used as an offset to predict the prey's future position, which is computed by projecting the prey in the direction of its previous move a number of steps equivalent to the count. The predators attempt to herd the prey using this predicted position. The count has a ceiling of 10, and when the prey changes direction the count is reset to zero. Since the agents all move at the same speed, it is impossible for the predators to catch up to a Linear prey once it is directly behind it. This modification was designed using knowledge of the Linear prey, and allows the predators to catch up to the prey by shortcutting to the prey's predicted position. We expect this modification to improve performance only against the Linear prey.

Deadlock occurs frequently when the predators and prey have deterministic strategies. For all three predator strategies in this section we vary the amount of noise that is introduced into the strategies in order to help prevent deadlock. In this paper, these agents will be named by appending the amount of noise to the name of it's strategy. For example, "Follower-10%" is a Follower agent which, for each move, has a 10% chance of choosing a random action.

Performance of Hard-Coded Predators Each predator strategy was tested on the Random and Linear preys. $10,000$ runs were performed on each predator-prey pairing. Each run terminates when either the prey is captured, or after $50,000$ cycles have elapsed, whichever occurs first. The results can be summarized by the following observations:

- *The Linear prey is more difficult to capture than the Random prey.* See Figure 1.

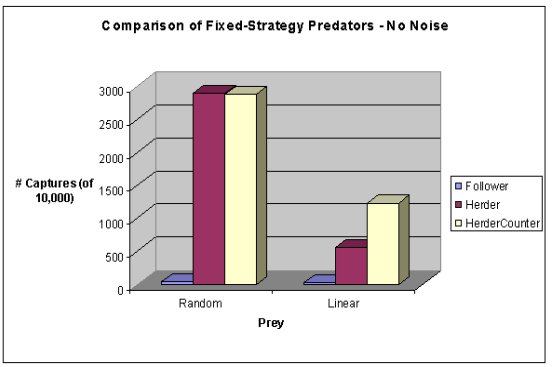


Fig. 1. Performance of Fixed-Strategy Predators on the Random and Linear Preys

- *Noise Significantly Improves Performance.* See Figures 2 and 3. In many cases, the predators cannot capture without noise. Using more noise can actually reduce the capture time, and unlike previous work which report results as percentage of captures this paper reports the average capture time because we observe 100 percent capture rate.
- *Coordination Can Improve Performance.* Herder and HerderCounter make use of implicit communication by coordinating their roles at the initiation of each run, and they both outperform the Follower strategy.
- *State Information Can Improve Performance.* HerderCounter agents use the count state information to offset the desired capture position. Figure 3 shows that of the three predator strategies in this section, the HerderCounter performs the best on the Linear prey. As expected, the HerderCounter strategy shows no significant improvement over the Herder strategy when attempting to capture the Random prey (see Figure 2). This shows that proper state information can be very important, but determining what state information is useful can be very problem specific.
- *Our noisy predators perform better than all previously published work to our knowledge.* A previous work whose experimental setup is most similar to our work is perhaps [12], although their setup makes the predators' job easier because they are allowed to move more frequently than the prey. Haynes and Sen and other previous work [13] on similar preys report results as a percentage of trials that lead to capture, whereas the results reported here show 100% capture rate.

3.2 Evolved Predators

This section describes how a genetic algorithm is used to evolve predator strategies, and compares the performance of these strategies with the fixed-strategy

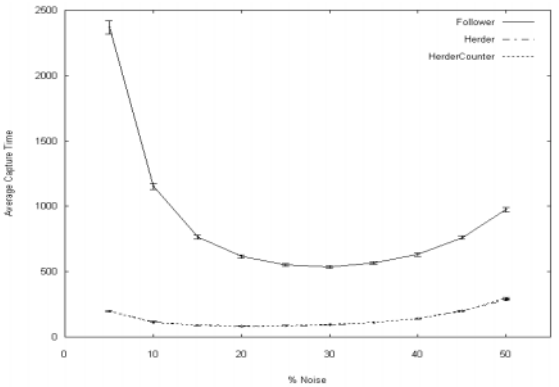


Fig. 2. Performance of Fixed-Strategy Noisy Predators on Random Prey. Error bars show 95% confidence intervals obtained by running 10000 simulations on each point

predators of the previous section. Each individual in the GA population represents a predator strategy that is used by all 4 homogenous predators in each scenario.

Encoding Predator Strategies The sensory information available to the predators comprise only the range and bearing of the prey. The range is measured in terms of the Manhattan distance. Both the range and bearing are discretized into $N_{range} = 8$ and $N_{bearing} = 8$ sectors. The predators can detect when the prey is 0, 1, 2, and 3+ cells away, measured in terms of the Manhattan distance. Ranges of 3 or more cells away are lumped under the same sector. The bearing of the prey from the predator is discretized into 8 equal sectors similar to the slices of a pizza pie. The 4 available actions are the moves $\{N, S, E, W\}$.

The behavior of each evolved predator is represented by a binary chromosome string. The number of binary bits required to represent the 4 actions are $b_{actions} = 2$. The total length c of the GA chromosome is given by the following equation:

$$\begin{aligned} c &= N_{range} N_{bearing} b_{actions} \\ &= 128 \end{aligned}$$

Evaluating Fitness The fitness of each evolved strategy is determined by testing it on 100 randomly generated scenarios. Each scenario specifies unique starting locations for the predator and prey agents. The maximum number of cycles per scenario is 5000, after which the scenario times out and the predators are considered to have failed. Since the initial population is randomly generated, it is very unlikely that the first few generations will be able to capture the

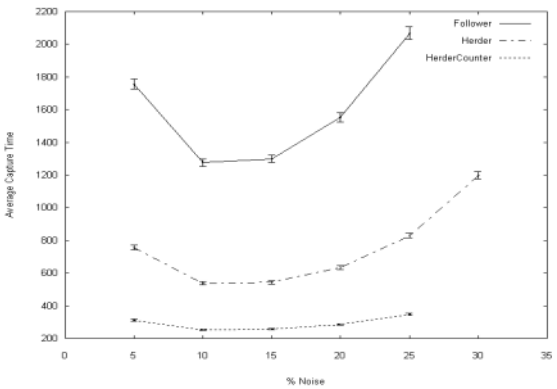


Fig. 3. Performance of Fixed-Strategy Noisy Predators on the Linear Prey. Error bars show 95% confidence intervals obtained by running 10000 simulations on each point

prey. We attempt to speed up the evolution of fit strategies by rewarding those strategies that at least stay near the prey and are able to block the prey's path. The fitness f_i of individual i is computed at the end of each generation as follows, where $N_{\max} = 5000$ is the maximum number of cycles per scenario, $T = 100$ is the total number of scenarios for each individual, and n_c is the number of captures:

- If $n_c = 0$, $f_i = \frac{0.4}{d_{avg} + 0.6\frac{n_b}{N_{\max} T}}$ where d_{avg} is the average distance of the all 4 predators from the prey during the scenarios, and n_b is the cummulative # of cycles where the prey's movement was blocked by an adjacent predator during T scenarios. The fitness of non-capture strategies can never be greater than 1.
- If $0 < n_c < T$, $f_i = n_c$.
- If $n_c = T$, $f_i = T + \frac{10000T}{\sum_{j=0}^{T} t_j}$, where $t_j$is the number of cycles required to capture the prey at scenario j.

GA Setup The following parameters of the GA algorithm were found experimentally to be most effective. The population size of each generation is fixed at 100 individuals. The mutation rate is set at 0.01. We use 2-point crossover with a crossover probability of 0.4. The idea behind multi-point crossover is that parts of the chromosome that contribute to the fit behavior of an individual may not be in adjacent substrings. Also, the disruptive nature of multi-point crossover may result in a more robust search by encouraging exploration of the search space rather than early convergence to highly fit individuals. For a discussion of 2-point crossover and generalized multi-point crossover schemes see [16]. A Tournament selection scheme [17] with a tournament size $Tour$ of 5 is used to select

the parents at each generation. In Tournament selection, *Tour* individuals are chosen randomly from the population and the best individual from this group is selected as a parent. This is repeated until enough parents have been chosen to produce the required number of offsprings for the next generation. The larger the tournament size, the greater the selection pressure, which is the probability of the best individual being selected compared to the average probability of selection of all individuals.

The following pseudocode describes the methodology:

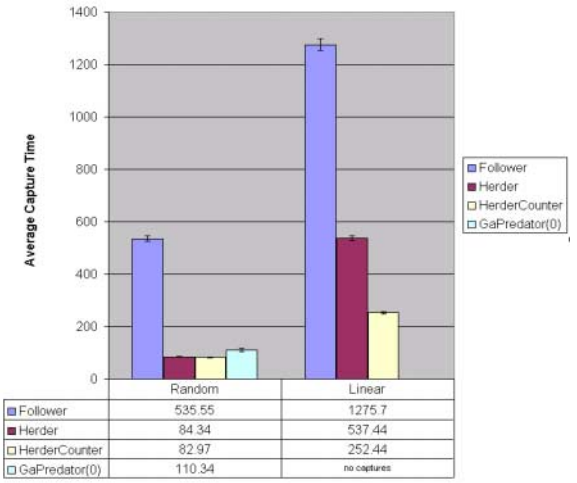
1. Repeat the following for 10 trials on selected prey:
   (a) Randomly generate a population of 100 individuals.
   (b) Repeat the following until the predators show no improvement after 200 generations:
      i. Simulate each predator strategy on 100 scenarios and evaluate its fitness based on the performance on those scenarios.
      ii. Select 100 individuals from the current population using Tournament selection, pair them up, and create a new population by using 2-point crossover with mutation.
   (c) The best strategy found over all generations is used as the solution of this trial. The fitness of this strategy is then recomputed by testing on 1000 new randomly generated scenarios.
2. The strategy that performed best over all 10 trials is used as the solution to this prey.

Performance of GA Predators The results of the best GA predators evolved for each prey are shown in Figure 4. Also shown in the figure are results of the fixed predator strategies at their optimal noise levels, which were obtained by taking the best average capture times found for each predator in Figures 2 and 3. The GA predator strategy is denoted as *GaPredator(0)*, where *(0)* indicates that there is no communication.

The performance of GaPredator(0) against the Random prey is comparable to the performance of the Herder and HerderCounter predators (though taking on average 30 cycles longer to capture), and significantly better than the performance of the Follower predators. However, none of the evolved predators were able to reliably capture the Linear prey. In the next Section we explore whether or not allowing the predators to evolve communication would improve their performance against the Linear prey.

4 Communication

All predator agents communicate simultaneously to a *message board* (Figure 5). At every iteration, each predator speaks a string of symbols which is stored on the message board. Each agent then reads all the strings communicated by all the predators and determines the next move and what to say next. Strings are restricted to have equal length l. We vary the length l of the strings and study the effect on performance.



	Random	Linear
Follower	535.55	1275.7
Herder	84.34	537.44
HerderCounter	82.97	252.44
GaPredator(0)	110.34	no captures

Fig. 4. Performance of best non-communicating predator strategies. Error bars show 95% confidence intervals

4.1 Communicating Agents as One FSM

This type of communication may be represented as shown in Figure 5, where $\{A_m\}$ is the set of homogenous predator agents, $\{O_m\}$ are the actions of the predators, and $\{I_{mn}\}$ is the set of environmental inputs, where n is the number of inputs and m is the number of communicating agents. The message board can be interpreted as a set of *state nodes*.

The entire set of agents can be viewed as one finite state machine (FSM) with the set of possible states specified by the state nodes $\{S_{ml}\}$. The whole multi-agent system is equivalent to a finite state automaton with output, otherwise known as a finite state transducer. One type of finite state transducer is the Mealy finite state machine, in which the output depends on both the state of the machine and its inputs. A Mealy machine can be characterized by a quintuple $M = (\Sigma, Q, Z, \delta, \lambda)$, where Σ is a finite non-empty set of input symbols, Q is a finite non-empty set of states, Z is a finite non-empty set of output symbols, δ is a "next-state" function which maps $Q \times \Sigma \rightarrow Q$, and λ is an output function which maps $Q \times \Sigma \rightarrow Z$.

It is easy to show that the multi-agent system is a Mealy machine by describing the multi-agent system in terms of the quintuple M. The input set Σ is obtained from the set $\{I_{00}I_{01}...I_{0n}I_{10}I_{11}...I_{mn}\}$ of all possible concatenated sensor readings for the predator agents (for all possible values of I). The states Q are represented by concatenation of all symbols in the message board. Since the communication strings comprise binary symbols $\{0, 1\}$, the maximum number of states N_{states} in the Mealy machine is therefore determined by the number of communicating agents m and by the length l of the communication strings: $N_{states} = 2^{lm}$. The output set Z is obtained from the

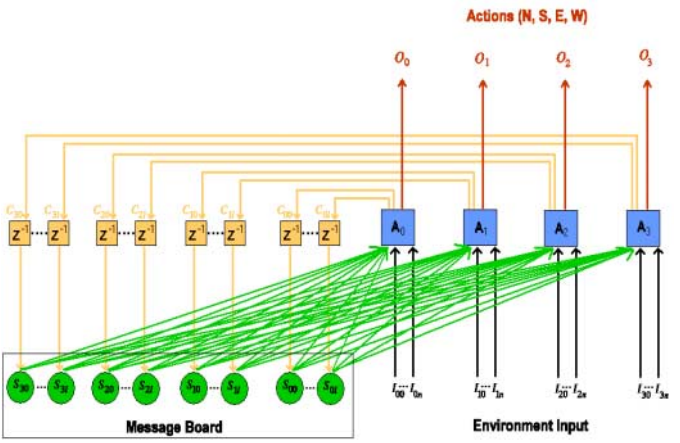


Fig. 5. Multi-agent Communication as a single Finite State Machine

set $\{O_{00}O_{01}..O_{0p}O_{10}O_{11}...O_{mp}\}$ of all possible concatenated actions for all the communicating agents, where p is the number of bits required to encode the possible actions for each agent (for all possible values of O). In the general case where the actions do not have to be encoded as binary bits, the output set is simply the set $\{O_0O_1...O_m\}$ of all possible concatenated actions for the m communicating agents. The next state function δ and output function λ are determined by the agents' action and communication strategies. The strategies themselves may be FSMs or something with even more representational power, in such a case the multi-agent FSM is a hierarchical FSM.

4.2 States and Partially Observable Environments

From Figure 5 it is clear that communication allows the agents to use state information. This state information is contributed by all communicating agents and represents the state of the multi-agent system. Although each individual agent may maintain its own state information, such information will be limited by the available sensors of the agent. Communication allows agents to "tell" each other environmental information that may have been observable only to a subset of the agents. Obviously, communication will be of little use in this respect in the limit when the same set of environmental information is observable to all agents. It is rare for all agents to have access to the same amount of information. This is due to the fact that an individual agent will usually have its own internal state that is not observable by other agents. If an agent's internal state helps determine its behavior, communication may be instrumental in allowing the agents to converge on an optimal plan of action.

4.3 Experimental Setup

A genetic algorithm is used to evolve predators that communicate. This section describes sets of experiments with strings of varying length l. As the length l increases, the number of strings that are available for communicative acts increases exponentially.

Encoding Predator Strategies The sensory information available to the predators include the range and bearing of the prey as discussed in Section 3.2. In addition, the predator agents have access to the contents of the message board. Since each agent speaks a string of length l at each time step, the number of symbols on the message board is ml, where m is the number of predator agents.

The behavior of each predator is represented by a binary string. The number of binary bits required to represent the 4 movements are $b_{moves} = 2$. In addition, each agent speaks a string of length l at each iteration. Thus, the total number of action bits is

$$b_{actions} = b_{moves} + l \tag{1}$$

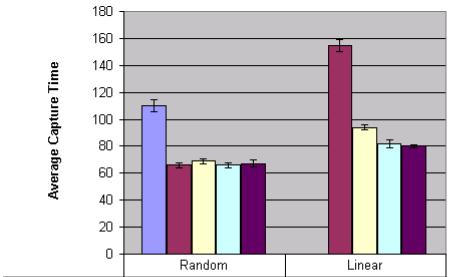
The range and bearing are discretized to $N_{range} = 8$ and $N_{bearing} = 8$ sectors. In addition, since there are ml binary symbols on the message board, the message board can have $N_{messages} = 2^{ml}$ possible messages. The total number of states that can be sensed by a predator is $N_{states} = N_{range}N_{bearing}N_{messages}$ This provides the following equation for the chromosome length c_{ml} of a GA predator:

$$\begin{aligned} c_{ml} &= b_{actions}N_{states} \\ c_{ml} &= N_{range}N_{bearing}(2+l)2^{ml} \end{aligned} \tag{2}$$

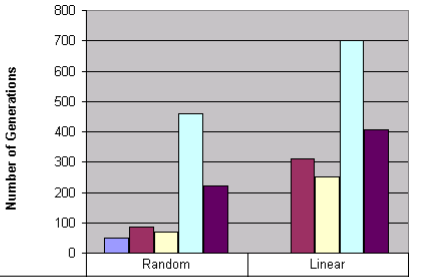
so the chromosome length increases exponentially with communication string length and number of agents.

Growing GA Predators - Coarse to Fine Search To improve efficiency, it would be useful to *grow* the predators. Growing means taking a population of predators that have already evolved a language from a set of possible strings, and evolving them further after increasing the set of possible strings they are allowed to communicate. This re-uses the knowledge acquired by predators that were limited to a smaller language. This is effectively a coarse-to-fine search. By starting with a smaller set of possible strings (and therefore smaller search space) the agents are forced to evolve a minimalistic language to communicate the most important state information. As we increase the search space by increasing the number of possible strings, the agents can refine the language and communicate other useful, but possibly less critical, information.

When a population of GA predators with chromosome length c_{ml} is grown to a length of $c_{m(l+1)}$, each new chromosome is encoded such that the behavior of the new predator is initially identical to that of the chromosome it was grown from. The portions of the larger chromosome that are new are not visited initially



	Random	Linear
GaPredator(0)	110	no captures
GaPredator(1)	66	155
GaPredator(0->1)	69	94
GaPredator(2)	66	82
GaPredator(1->2)	67	80



	Random	Linear
GaPredator(0)	50	no captures
GaPredator(1)	85	310
GaPredator(0->1)	70	250
GaPredator(2)	460	700
GaPredator(1->2)	220	405

Fig. 6. Best capture times and the corresponding number of evolutionary generations required to evolve the communicating predators against the Random and Linear preys, at communication string lengths 0, 1, and 2. Error bars on the capture times show 95% confidence intervals using the Student's t distribution, obtained by running 1000 simulations at each point

because the predator is making exactly the same decisions as before and will therefore see the same set of sensory states. As the evolutionary process begins, new sensory states will be visited and the agent will evolve accordingly.

The population size of the grown $c_{m(l+1)}$ predators is always twice the population size of the c_{ml} predators they were grown from. Half of the population of $c_{m(l+1)}$ predators are grown from the c_{ml} predators, the other half are generated randomly. In this manner the population of grown predators do not have to rely solely on mutation for introducing new genetic material to the genes that were copied from the c_{ml} predators. They can obtain new genetic material through crossover with the randomly generated individuals.

Setup In the sections that follow, the GA predators are labelled as GaPredator(l), where l is the length of the communication strings. $l = 0$ means the predators are not communicating, and is identical to the GaPredator discussed in Section 3.2. Grown predators are labelled as GaPredator($l_0 \rightarrow l_1$), where l_0 is the communication string length before the agent is grown, and $l_1$is the length it was grown to. Five predator populations GaPredator(0), GaPredator(1), GaPredator(2),GaPredator($0 \rightarrow 1$), and GaPredator($1 \rightarrow 2$) are matched against the Random and Linear preys. Each matchup is performed similarly to the set-up described in Section 3.2, except that each predator population is evolved until there is no further improvement in 200 generations. The initial GaPredator($0 \rightarrow 1$) population that is matched up against the Linear prey is grown from the GaPredator(0) population with the best average fitness against the Linear prey.

4.4 Results

Figure 6 shows the best average capture times, and the number of evolutionary generations that were required to achieve those capture times. The number of generations reported for the grown predators are recursively cumulative with the number of generations needed to evolve them before they were grown. Below is a summary of the results:

- Communication improves capture performance.
- As the length of the communication string increases, the capture time decreases. However, the best performance of GaPredator(1) against the Random prey is comparable to the best performance of GaPredator(2) and GaPredator($1 \rightarrow 2$), which indicates that a communication string of length 1 is sufficient against the Random prey.
- The evolutionary generations required increases with the length of the communication string.
- The performance of grown predators is comparable to that of the equivalent non-grown predators but requires significantly less evolution time.
- The evolved communicating predators perform better than all previously published work to our knowledge, and better than the noisy, hard-coded predators presented in the previous Section.

5 Conclusions

Introducing noise to the decision process of our human-designed predator strategies allows the predators to overcome deadlock and thus outperform predator strategies both programmed and learned in all previously published work. Furthermore, a genetic algorithm can evolve communicating predators that outperform all (noisy) human-designed predator strategies reported in this paper.

A multi-agent system in which all the agents communicate simultaneously is equivalent to a Mealy machine whose states are determined by the concatenation of the strings in the agents' communication language. The simulations show that increasing the language size, and thus increasing the number of possible states in the equivalent Mealy machine, can significantly improve the performance of the predators.

References

[1] Ackley, D.H., Littman, M.L.: Altruism in the evolution of communication. In Brooks, R.A., Maes, P., eds.: Artificial Life IV: Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems, MIT Press (1994) 40–49 378

[2] MacLennan, B.J., Burghardt, G.M.: Synthetic ethology and the evolution of cooperative communication. Adaptive Behavior **2** (1993) 161–188 378

[3] Walker, A., Wooldridge, M.: Understanding the emergence of conventions in multi-agent systems. In Lesser, V., Gasser, L., eds.: Proceedings of the First International Conference on Multi-Agent Systems, Menlo Park, CA, AAAI Press (1995) 384–389 378
[4] Steels, L.: Self-organizing vocabularies. In Langton, C., ed.: Proceedings of Alife V, Nara, Japan (1996) 378, 379
[5] Saunders, G.M., Pollack, J.B.: The evolution of communication schemes over continuous channels. In Maes, P., Mataric, M., Meyer, J., Pollack, J., eds.: From Animals to Animats 4: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior, MIT Press (1996) 580–589 378
[6] Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Autonomous Robots **1** (1994) 27–52 378
[7] Gmytrasiewicz, P.J., Durfee, E.H., Rosenschein, J.: Toward rational communicative behavior. In: AAAI Fall Symposium on Embodied Language, AAAI Press (1995) 378
[8] Hasida, K., Nagao, K., Miyata, T.: A game-theoretic account of collaboration in communication. In Lesser, V., ed.: Proceedings of the First International Conference on Multi–Agent Systems (ICMAS), San Francisco, CA, MIT Press (1995) 140–147 378
[9] Jim, K., Giles, C.L.: Talking helps: Evolving communicating agents for the predator-prey pursuit problem. Artificial Life **6(3)** (2000) 237–254 378
[10] Jim, K., Giles, C.L.: How communication can improve the performance of multiagent systems. In: 5th International Conference on Autonomous Agents. (2001) 378
[11] Benda, M., Jagannathan, V., Dodhiawalla, R.: On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA (1985) 378
[12] Haynes, T., Sen, S.: Evolving behavioral strategies in predators and prey. In Wei, G., Sen, S., eds.: Adaptation and Learning in Multiagent Systems. Springer Verlag, Berlin (1996) 113–126 378, 379, 380, 381
[13] Korf, R.E.: A simple solution to pursuit games. In: Working Papers of the 11th International Workshop on Distributed Artificial Intelligence, Glen Arbor, Michigan (1992) 183–194 378, 379, 380, 381
[14] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proc. of 10th ICML. (1993) 330–337 379
[15] Stephens, L.M., Merx, M.B.: The effect of agent control strategy on the performance of a dai pursuit problem. In: Proceedings of the 10th International Workshop on DAI, Bandera, Texas (1990) 379
[16] Jong, K.A.D., Spears, W.M.: A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Mathematics and Artificial Intelligence Journal **5** (1992) 1–26 383
[17] Goldberg, D., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G., ed.: Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, San Mateo, CA (1991) 69–93 383