

Evolution of Artificial Intelligence in the Game of Tag

Schoon Ha

Greg Turk

Georgia Institute of Technology

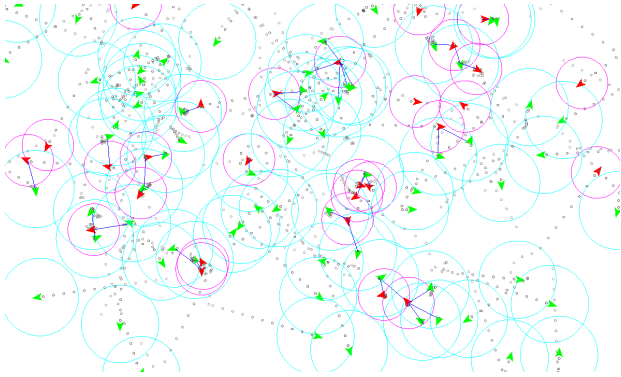


Figure 1: *The initial state of the simulator. Red agents are it (predator) and Green agents are non-it (prey) The ratio between the number of it and the number of non-it is 4:6.*

Abstract

This paper describes evolution of artificial intelligent in the Game of Tag. In our simulator, artificial agents are classified as 'it' (predator) and 'non-it' (prey) and assigned symmetrical goals : pursuit and evasion respectively. Instead of using explicit fitness functions, predators and preys are evolved solely based on their pursuit and evasion records. Each agent has own artificial intelligent which is represented as a set of lisp-style predicates including sensor data, conditional statement and numerical operations. Even though we have two separate species, 'it' and 'non-it', we initialize and mutate their A.I. in the same method and probability distribution. The evolved 'it' and 'non-it' have their own style of strategies and show agile pursuit and evasion behaviors.

1 Introduction

The game of tag is a popular playground game for children which has many other names and variational rules. [TGT] The game starts from deciding who is going to be one or multiple 'it'. Then 'it' players chase 'non-it' players who all try to escape. If 'it' player succeeded to get close to 'non-it' player, 'it' tags 'non-it' and their roles are swapped. This game need players to have a good pursuit and escape behaviors which is frequently observed in nature.

Our goal is evolving intelligent behavior of 'it' and 'non-it' without any explicit fitness function. This approach is more convincing than having explicit fitness functions since there is no such a function in nature and only controlled by survival of the fittest. Moreover, we are trying to avoid explicit parameters such as speed or sensor range in gene representation because we assume that it can

make the direction of evolution biased towards user intension. In our simulation, A.I. of agents takes only raw sensor data as an input and use them in non-straight forward manner. Thus, our evolution is not biased and more resemble to that of nature.

2 Related Work

Our work is originally inspired by classic research of Karl Sims [Sims 1994]. He represents a genetic language with nodes and connections, and fitness of each virtual creature is calculated by different types of measures, such as speed of walking or height of jumping. Another paper on [Reynolds 1994] the Game of Tag is more related to our research, especially in terms of representation of genes. He uses a lisp-style language to approximate a vector field like algorithm, and directly crossover lisp-style genes in "random cut and paste" manner. He also measures semi-explicit fitness value of creatures and apply Steady-State Genetic Programming (SSGP) for evolution.

Avoiding explicit fitness functions has been studied to achieve evolution in more natural way. One of prior works is [Ventrella 1996] which simultaneously and continuously simulates a lot of swimmers which has genes and a state machine. The recent work of Greg Turk [Turk 2010] also demonstrates artificial evolution of physically simulated creatures which can adjust friction coefficients of own feet. Both researches successfully show that natural evolution is possible without any type of artificial fitness functions.

3 Simulation

3.1 Agent

In this work, each agent has a sensor which reports a position of the closest enemy (the closest predator for a prey agent, and vice versa). This position information is defined in each one's local coordinate as shown in Fig.2. After each player calculates the next direction using its own algorithm described in section 3.2, it moves forward a fixed distance along its new direction. Our model is completely kinetic and there is no mass, inertia, momentum or friction at all, so each agent can turn its direction as much as it wants. In our configuration, a predator has 25% faster speed but 30% shorter sensor range comparing to a prey.

3.2 Representation of Genes

To represent behavior of each agent, we use a set of lisp predicates and terminals (Table.1) including sensor data. Terminals consists of 'local-x', 'local-y' and some random floating constants between 0 to 1. There is no limitation of the number of sensor data terminals ('local-x' and 'local-y') in each program, thus there is a possibility of totally blinded agents. The operator predicates such as 'plus' or 'minus' and the 'ifte' macros provides manipulation of data and conditional execution and enable more complex program to agents. To prevent exponential growth of A.I., we set the limitation to the number of predicates in one A.I. program.

In mutation, there is no crossover operation since we only allow an agent to reproduce itself alone (Parthenogenesis) . When reproduction occurs, an agent selects one of its 'victim' subtree of A.I.

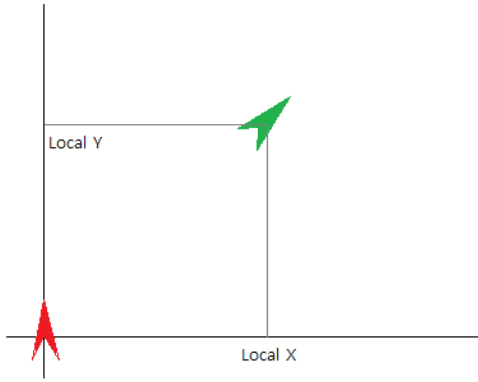


Figure 2: Each agent has a sensor which is defined in its local frame. For simplicity, only the closest 'it' is reported to a sensor of 'non-it', and vice versa.

Table 1: A list of predicates

Function	Description
0.123	a random constant between 0.0 1.0
local x	local x position of the closest enemy
local y	local y position of the closest enemy
(+ a b)	a plus b
(- a b)	a minus b
(* a b)	a times b
(/ a b)	if b = 0 then 1 else a divided b
(ifte a b c d)	if a ; b then c else d

program and replace the subtree with a new randomly generated predicate. By change the desired size of the 'victim' subtree, we can easily control the size of the entire program.

3.3 Reproduction Rules

There are three cases of reproduction in our simulation. The first rule is reproduction by capturing. If a predator get closed to a prey, a predator 'kills' a prey and reproduces itself. To maintain the ratio between the number of predators and preys, we choose lottery-based reproduction method. If an agent A needs to reproduce itself, a victim agent B is chosen from a group of A and replaced by A. On the other hand, if an agent B needs to be killed then a winner agent B is chosen from the same group and B replaces A. Here is a simple example in capturing event. If X kills Y, X' and Y' are selected from a predator group and a prey group respectively, and X replaces Y' and X' replaces X. (X, X', Y, T' are agents)

The second rule is removing a predator who fails to hunt for a long time, which is important cause this is only rule for giving a penalty to bad predators. The third rule is reproducing a prey who successfully evades, and this rule is an award for smart preys. Thus, all good/bad predators/preys get valid feedback by their pursuit and evasion record.

4 Results

One way of observing evolution is measuring the average of explicit fitness functions with respect to time. Even though we do not use any explicit fitness function in our simulation, we can approximate fitness functions since there is a simple optimal solution in pursuit and evasion behavior (Figure 3). By calculating a dif-

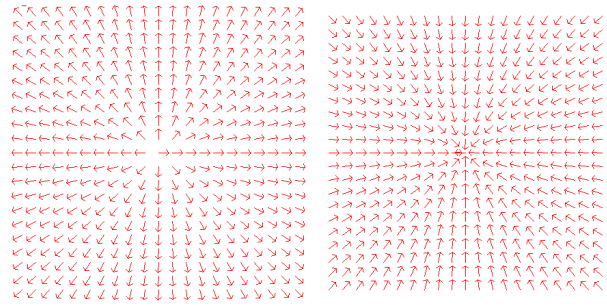


Figure 3: Optimal solutions. X and Y represent local coordinates, and direction is the next steering angle. The left is an optimal algorithm for predators (pursuit) and the right is for preys (escape)

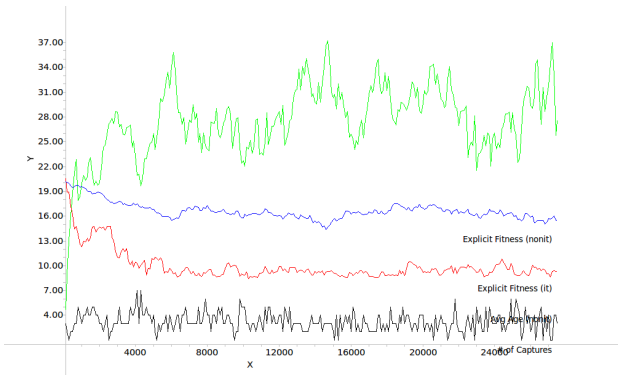


Figure 4: Red and blue lines represent explicit fitness for predators and preys respectively. Green is an average age of preys, and Black is a number of captures within certain time window.

ference between optimal solution, we can measure explicit fitness values. Thus, a lower fitness value is expect to be good in general. The result is shown in Figure 4. Evolution of predators ('it') seems more significant than that of preys ('non-it'). But preys are also well evolved when we observe their behaviors individually, so we conclude that the optimal solution for evasion in Figure 3 is not the only solution and good preys doesn't need to be similar to the optimal solution. On the other hand, most of successful predators resembles the 'source' like optimal solution.

For instance, the best-of-population predator in Figure 5 is somewhat similar to the optimal solution especially when the target prey is located in front of the predator. On the other hand, the best prey agent acts in totally different manner when it detects a predator in its direction, but it also shows very smooth turn and is able to run away successfully. Another virtue of this best prey is that this a.i. pattern can be easily represented in lisp predicates so it occurs frequently (Equation. 1).

$$(-(*(\text{local}_x) 0.12)(*(* - 0.77 0.12)(* 0.19 (\text{local}_x)))) \quad (1)$$

Another phenomenon of our simulation is that predators show faster evolution than preys in most of cases. This is due to the influence of capturing events which are the most frequent (and important) events in this tiny world. Capturing events directly duplicate the successful predators, but there is a possibility of a good prey being hunted since speed of predators is faster than that of preys. In another set of experiments without duplicating successful preys (Rules described in Section 3.3), preys show even slower evolution than before.



Figure 5: *The best-of-population agents. The left is one of the most successful predator, and the right is one of the best prey.*

5 Conclusions

We observe successful artificial evolution of agents in the game of tag with few reproduction rule and simple representation of program. Our contribution is that we minimize human intervention for more natural evolution. For this purpose, we exclude any explicit function or explicit parameter which might affect a direction of evolution. Instead, we use continuous simulation with simple reproduction rules and a set of lisp predicates which have no physical interpretation except raw sensor data predicates. From this simple configuration, we are able to observe a lot of interesting pursuit and evade behaviors after evolution.

Although our simple competition show successful evolution, there are limitations of our work should be improved in future work. Currently we always select a random victim in reproductin to maintain the ratio between predators and preys, but it is different from real reproduction in nature. Thus, we might need to allow some variation of populaion with larger simulation and carefully tuned parameters. Also, random parameters in mutation such as probability distribution over predicates can affect a direction of evolution. So, adopting bisexual reproduction and crossover operation would be a good solution to avoid explicit parameters.

References

REYNOLDS, C. 1994. Competition, coevolution and the game of tag. *MIT Press*.

SIMS, K. 1994. Evolving virtual creatures. In *SIGGRAPH*.
The game of tag, [http://en.wikipedia.org/wiki/Tag_\(game\)](http://en.wikipedia.org/wiki/Tag_(game)).

TURK, G. 2010. Sticky feet: Evolution in a multi-creature physical simulation. *Artificial Life XII*.

VENTRELLA, J. 1996. Sexual swimmers. *MIT Press*.