Coevolution of Competing Agent Species in a Game-like Environment

Telmo Menezes[1] and Ernesto Costa[1]

Centre for Informatics and Systems of the University of Coimbra,
{telmo, ernesto}@dei.uc.pt

Abstract. Two species of agents coevolve in a 2D, physically simulated world. A simple fitness function rewards agents for shooting at agents of the other species. An evolutionary framework consisting of the gridbrain agent controller model and the SEEA steady-state evolutionary algorithm is used. We were able to observe a phenomenon of species specialization without the need for geographical separation. Species with equal initial conditions were shown to diverge to different specialization niches by way of the systems dynamics. This kind of research may lead to more interesting gaming environments, where the world keeps changing and evolving even in the absence of human interaction.

1 Introduction

The evolution of agent controllers has promising applications in the generation of computational intelligence for video games. The behaviors of computer controlled entities may adapt to the demands of the environment, instead of being explicitly programmed. This kind of approach fits particularly well with games genres that are bases on multi-agent environments. NERO [1], for example, is a recent experimental computer game that explores the evolution of agent controllers. NERO simulates a battle of agent teams, and the human player is required to train a team formed of neural network controlled agents under an evolutionary process. The human player must create training situations by placing agents and objects in the scenario, as well as tweaking a fitness function throughout the process.

We experiment with the evolution of game agents without human intervention. Our goal is to demonstrate that interesting behaviors can arise from the coevolution of competing species, using a predefined, simple fitness function.

The experimentation we performed uses gridbrains as agent controllers, and the simulation embedded genetic algorithm (SEEA) to provide the evolutionary process. Both these models have been developed to define an evolutionary framework of computational intelligence for multi-agent simulations. We encourage the reader to refer to previously published work on these models [2] [3].

In the next two section we will give an overview of the gridbrain model and the SEEA algorithm. Following, we will describe the experimental settings and then present results.

2 The Gridbrain

The gridbrain is a virtual machine designed to serve as a brain for an autonomous agent. It belongs to the family of genetic programming, with some similarities to Parallel Distributed Genetic Programming [4] [5] and Cartesian Genetic Programming [6][7]. It consists of a network of computational components placed on rectangular grids. There are two types of grid: alpha and beta. Alpha grids are associated with sensory channels and are responsible for processing perceptual information. The beta grid receives inputs from the alpha grids and outputs decisions. A gridbrain can have any number of alpha grids (one of each sensory channel), but only one beta grid.

Components are information processing units. They are the computational building blocks of gridbrains. Much like machine code instructions, components belong to classes of functionalities: input/output, arithmetic, boolean logic, information aggregation, synchronization and memory.

Components and connections may be active or inactive, depending on if they belong to an *active path*. An active path is a sequence of connections that links a component that generates information to a component that triggers and action.

The gridbrain model provides a set of genetic operators that can be used by an evolutionary algorithm to produce replication with change. There are three types of operators. Two of them are usual in evolutionary computation systems: mutation and recombination. The third, formating, deals with adapting the shape of the grids as evolution progresses and is related to the complexification of gridbrains.

Mutation operators are defined at connection level and component level. There are two pairs of connection level operators: *add/remove* and *split/join*. The operators in each pair are symmetrical, one performing the inverse operation of the other. The add operator inserts a new valid connection and the remove operator deletes an existing connection from the gridbrain. These mutations occur with respective probabilities of p_a and p_r per existing connections in the gridbrain. The split operator routes an existing connection through an intermediary component in the grid, and the joint operator reverts a previous splitting. Splits and joins occur with respective probabilities of p_s and p_j per existing connection.

There is one component level operators: *change inactive component*, which replaces an existing component with a new one. The new component is randomly selected from the grid component set. These mutations occur with a probability of p_c per component in the gridbrain.

The recombination operator for the gridbrain has to deal with the recombination of differently shaped grids, containing networks with different topologies. Furthermore, there are different types of components, so the network nodes are heterogeneous. It uses a connections tagging mechanism to use equivalent connection groups as the units of recombination. This recombination operator always produces valid gridbrains and is able to recombine functionalities in a meaningful way.

Formating is an operator that adapts the shape of the grids according to the network contained in the gridbrain. It is of a non-stochastic nature and can be seen as an adaptation mechanism. The changes it performs do not affect the phenotypical expression of the individual. The purpose of formating is to regulate the search space of the evolutionary process. It is part of the complexification process. We attempt to create systems that are initialized with empty brains, and that undergo an increase in complexity as higher quality solutions are found. Solutions are constructed through iterated tweaking, in a similar fashion to what happens in nature. Our goal is to have the size and complexity of the gridbrains to be determined by the demands of the environment.

The set of gridbrain operators were also designed with the goal of bloat control [8], and have been shown to be effective in this respect [2] [3].

3 Simulation Embedded Evolutionary Algorithm

The *Simulation Embedded Evolutionary Algorithm* (SEEA) is a steady-state evolutionary algorithm that we developed for the purpose of evolving agents in continuous simulations without generations. It allows the seamless integration of an evolutionary process with a multi-agent simulation. New individuals may be generated at any moment and on demand.

SEEA keeps a buffer of individuals for each species in the simulation. Each time an individual dies or is removed from the population, its fitness is compared to the fitness of a random individual in the buffer. If the fitness is equal or greater, the recently removed individual replaces the old one in the buffer, otherwise the fitness of the old individual is updated using the expression:

$$f_{new} = f_{old} \cdot (1 - a)$$

where a is the *fitness ageing factor*.

The purpose of fitness ageing is to maintain diversity in the buffer. Individuals that make it to the buffer have a chance of producing offspring. The higher their fitness, the more offspring they are likely to produce, but eventually they will be replaced, even if it is by a lower fitness individual. Fitness ageing takes place when an individual in the buffer is challenged by an individual removed from the population, so it adapts to the current rate of individual removal, which can be variable and unpredictable.

When the simulation requests a new individual, two elements in the buffer are selected at random and recombined if recombination is to be done, or one is selected at random and cloned for simple mutation. Mutation is applied, followed by formating, and finally the placement of the offspring in the simulation environment. A probability of recombination, p_{rec}, is defined. This probability is used to randomly decide in each individual request if recombination is to be applied.

In simulations with fixed populations, as the one presented in this paper, agent removal is always followed by the creation of a new agent.

The basic SEEA algorithm presented is capable of creating evolutionary pressure so that, as the simulation advances, agents tend to increase their fitness value. One limitation of it is that it only promotes the emergence of behaviors that benefit each agent directly. It may be insufficient to promote the emergence of collective behaviors, where some degree of altruism is needed.

Group fitness in SEEA is a mechanism to reflect the success of other members of a species that coexisted with an agent in the simulation on the agent's own fitness value. The principle is to allow an agent to increase its fitness by developing behaviors that benefit the fitness of other agents of its species. It is inspired on the biological theory of *group selection* [9].

The simulation environment and fitness evaluation do not have to be altered in any way for group fitness to be applied. Instead of directly using the individual fitness evaluation provided by the environment, f_i, a *composite fitness* is used. The composite fitness, f_c is calculated by the expression:

$$f_c = (1 - g).f_i + g.f_g$$

where f_g is the *group fitness component* and g is the *group factor*. The group factor is a real value in the $[0, 1]$ interval. The higher it is, the more important the group success is to the composite fitness of each agent.

The group fitness component reflects the variation in the fitness of other agents, during the lifetime of the agent for which it is being calculated. The principle is to only reflect fitness variations that the agent may have helped cause in others. An effective way to compute the group fitness component is to maintain a group fitness sum, G for each agent. When an agent is sent into the environment, its G is initialized by applying the expression:

$$G = -\sum_{a \in S(t_0)} f_a(t_0),$$

where t_0 is the simulation time at which the agent was created, $S(t_0)$ is the set of agents in the world belonging to the same species as the agent we are calculating G for, at simulation time t_0, and $f_a(t_0)$ is the current individual fitness for agent $a \in S(t_0)$. Then, during the lifetime of the agent, each time another agent of the same species dies, we increment G by that agent's final individual fitness. When an agent dies, it's final group fitness sum is calculated by applying the expression:

$$G' = G + \sum_{a \in S(t)} f_a(t)$$

This way, in the end of the agent lifespan, G contains the summation of the variations of individual fitnesses in other agents of the same species, during that lifespan. Finally, the group fitness component is given by:

$$f_g = \frac{G}{pop - 1}$$

where *pop* is the population size of the agent's species in the environment. In case this population is variable, an average can be used. This way, f_g gives us the individual fitness variation per other agent in the environment.

4 Experimental Setup

Experimentation is done with LabLOVE [10], a tool that we developed for our research and that is available to the community as open source. LabLOVE includes an implementation of the Gridbrain, the SEEA algorithm and a physically simulated multi-agent environment, as well as real-time visualization and data recording modules.

We define a videogame inspired scenario where agents are rewarded for shooting at agents of the other species. Agents have the ability to move in a 2D physically simulated world, shoot, eat food items and emit sounds. They have two sensory channels: vision and audition. Vision provides them with information about objects in the environment, including other agents and the targets. Audition allows them to perceive sounds emitted by other agents and by shots. The effectiveness of a laser shot is related to the amount a time an agent spends aiming at a target. A simulation parameter named *laser_interval* (l_i) establishes the number of simulation cycles that an agent must spend aiming at a target for it to have full effect. The amount of damage that a target suffers from a shot is determined by the expression $d = l_i \cdot l_t \cdot d_{max}$, where d is the damage, l_t is the time the agent spent aiming at that object and d_{max} is the maximum damage that a shot can cause. Agents can increase their chances of destroying a target by cooperating, which means, shooting at the same target at the same time.

Two species are evolving at the same time, and the world is constantly populated with a fixed number of food items that the agents can consume to increase their energy level. The only difference between the two species is their color, species A being red and species B being blue. Food items always provide an increase in energy level of 1.

Gridbrains are configured to have three grids: one alpha grid to process vision sensory information, another alpha grid to process audition sensory information and the beta grid. Additionally to generic gridbrain components, input and output components of the types detailed in table 1 are included in the respective grids component sets.

The input value of an action component, i determines the intensity with which the action is triggered. The force applied in go and rotate actions, as well as the intensity of the laser shot is proportional to i. The energy costs of actions is proportional to i.

Every time an agent shoots at a target, a score is calculated by multiplying the damage caused by this shot with the number of simultaneous successful shots from other agents. We define the fitness function as the best score obtained during the agent's lifetime. This encourages the agents to evolve mechanisms to produce more effective shots, but also to synchronize their shooting with the other agents.

Vision Input	Description
Position	Position of the object relative to the agent's vision field. -1 is the farthest possible to the left and 1 is the farthest possible to the right.
Distance	Distance from the object, divided by its view range, resulting in a value in $[0, 1]$.
Target	1 is object is on target, 0 otherwise.
Eating target	1 is object is current eating target, 0 otherwise.
Line of Fire	1 is currently on the target of this object, 0 otherwise.
Color	Distance between agent's color and this object's color.
Sound Input	**Description**
Position	Position of sound origin relative to agent.
Distance	Distance of sound source, divided by sound range.
Action	**Description**
Go	Apply forward force to agent.
Rotate	Apply rotation torque to agent.
Fire	Fire laser.
Eat	Attempt to eat nearest object.
Speak	Emit a sound.

Table 1. Sensory inputs and actions used in the experiment.

Evolutionary parameters are configured as following: add/remove connection probability is set to $p_a = p_r = 0.01$; split/join connections probability is set to $p_s = p_j = 0.01$; the change inactive component operator is used, with $p_c = 0.2$; change parameter is set to $p_p = 0.01, \delta = 1.0$; recombination probability is set to $p_{rec} = 0.25$; SEEA buffer size is set to $s_{buf} = 100$ and the fitness ageing factor is set to $a = 0.5$. These values where found to be good choices by previous benchmarking.

The simulation is defined to have 20 agents of each species and 20 food item. Agents have a random maximum lifespan of 9.5 to 10.5 Kcycles. They can die because this lifespan is reached, or by expending to much energy or being shot. The complete experiment definition can be found in the *battle.lua* file, that is in the *experiments* directory of the LabLOVE release.

5 Results

Four simulation runs were performed. We will present and analyze the evolutionary history of two of them.

In figure 1 we present the evolution of several metrics for the first simulation run. Each panel present the evolution of a metric for the two species. Species A is represented in solid line, while species B is represented in dotted line. In panel a) it can be observed that both species go through a period of fitness increase until about 1×10^5 Kcycles. From this moment on, a sharp increase in the fitness of

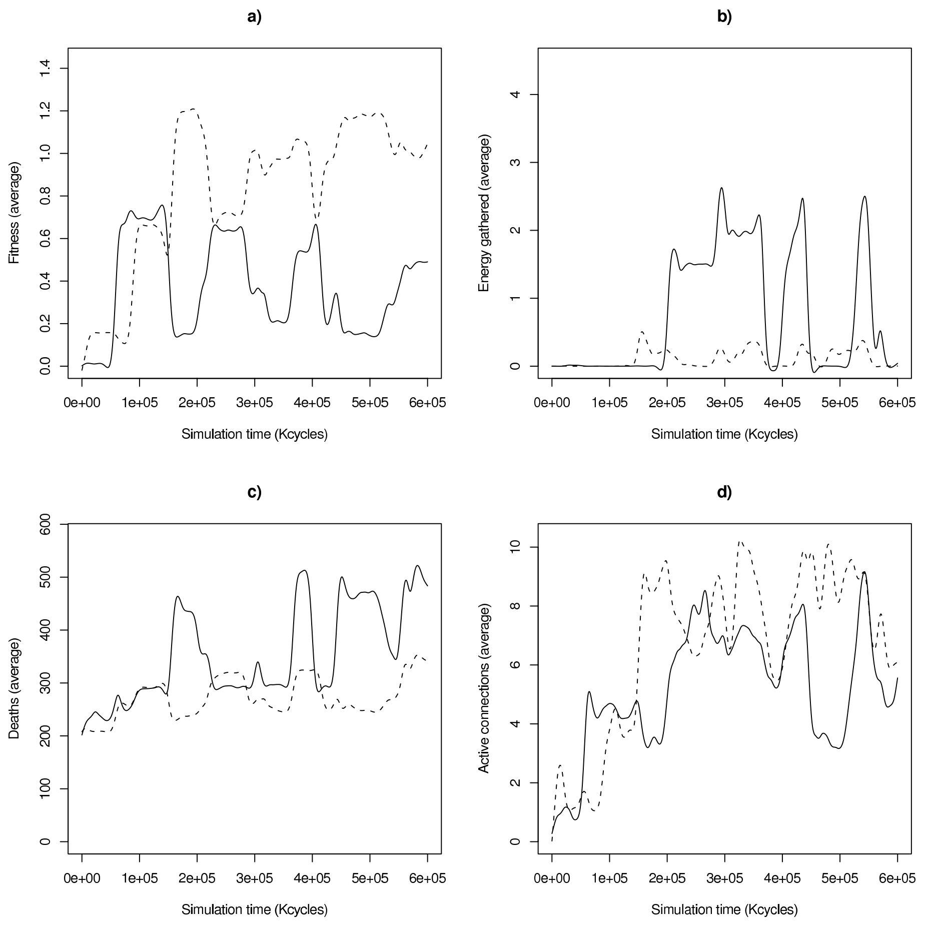

Fig. 1. Evolutionary history of a battle simulation run. Several metrics are presented for the two species. Species A is represented in solid line, species B in dotted line. a) Average fitness; b) Average energy gathered; c) Deaths; d) Average active connections.

species B affects the performance of species A, causing a decline in its fitness. As can be observed in panel c), this relates to species B gaining the ability to destroy species A agents, and thus causing an increase in the species A death rate. Until the end, the simulation run goes through periods of convergence and divergence in the fitness of both species. Almost symmetrical fitness curves are created, with the axis of symmetry being an horizontal line coinciding approximately with the 0.6 fitness level. At several moments the fitness of both species becomes almost the same, only to diverge again, with species B always taking the lead.

In panel b) it can be observed that, for several times, species A shows the ability to gather energy by consuming food items. This is an interesting result because the consumption of food items is not directly rewarded by the fitness function. It helps indirectly, because by sustaining higher energy levels, agents are harder to destroy, and by living longer they have a greater chance of producing good shots.

In this simulation run, the two species develop different strategies. Species B relies more on its ability to destroy opponents, while species A relies more on increasing its survival chances by consuming food items. This is interesting because both species are defined with the same initial conditions. It is an indication of the possibility of generating diversity through species coevolution in this type of simulation. It is likely caused by initial, random changes, sending each species in a different evolutionary path. This is consistent with non-linear, complex system behavior. Also, it is a niching phenomenon caused, not by physical separation, but by the dynamics of the system.

In panel d) we can observe that average gridbrain complexity is more unstable than in previous scenarios. Species go through stronger fluctuations in the average number of active connections. We believe this is cause by the *arms race* [11] between the two species leading to a more dynamic environment.

In figure 2 we present the evolutionary history of another run for the same scenario. The reason we do it is to illustrate that this scenario leads to diverse evolutionary histories. In this second run, it can be observed that the species attain a similar capability of destroying each other, leading to similar fitness curves oscillating around the 0.6 fitness value. The death count average for both similar also increases in a similar fashion and overall there is equilibrium between the species. As can be seen in panel b), species B goes through a period of greater ability in consuming food items, but to the end both species stabilize their consumption level around similar values.

6 Final Remarks

We were able to observe a phenomenon of species specialization without the need for geographical separation. Species with equal initial conditions were shown to diverge to different specialization niches by way of the systems dynamics. Another observation we made in this scenario is the unpredictable nature of evolutionary runs.

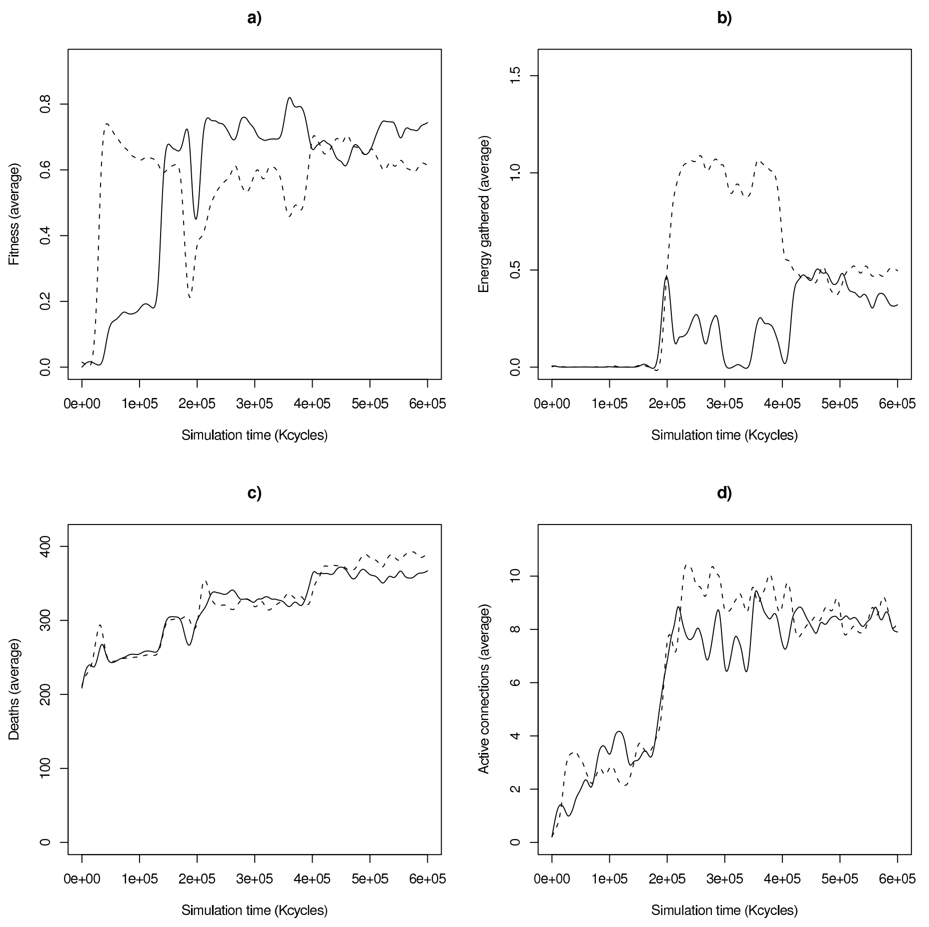

Fig. 2. Evolutionary history of another battle simulation run. Several metrics are presented for the two species. Species A is represented in solid line, species B in dotted line. a) Average fitness; b) Average energy gathered; c) Deaths; d) Average active connections.

Coevolution under our framework shows the potential for generating diverse and surprising agent behaviors, by exploring the interaction between the fitness function, the environmental basic rules and the dynamics of coevolution of species.

This kind of research may lead to more interesting gaming environments, where the world keeps changing and evolving even in the absence of human interaction.

Acknowledgments

The first author would like to acknowledge grant SFRH/BD/19863/2004 from *Fundação para a Ciência e Tecnologia (FCT)*, Portugal.

References

1. Stanley, K. O. and Bryant, B. D., and Miikkulainen, R., *Evolving Neural Network Agents in the Nero Video Game*. In Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games, 2005.
2. Menezes, T. and Costa, E., 2008. Artificial Brains as Networks of Computational Building Blocks. In *Proc. of the 5th European Conference on Complex Systems*. Jerusalem, Israel. http://www.jeruccs2008.org/node/606.
3. Menezes, T., 2008. *Evolutionary Computational Intelligence for Multi-Agent Simulations*. Departamento de Engenharia Informatica, Faculdade de Ciencias e Tecnologia, Universidade de Coimbra.
4. Poli, R. 1996. *Parallel Distributed Genetic Programming*. technical report CSRP-96-15. The University of Birmingham, UK.
5. Poli, R. 1999. Parallel Distributed Genetic Programming. *Chap. 27, pages 403–431 of:* D. Corne, et al. (ed), *Optimization, Advanced Topics in Computer Science*. Maidenhead, Berkshire, England: McGraw-Hill.
6. Miller, J. F. 1999. An Empirical Study of the Efficiency of Learning Boolean Functions using a Cartesian Genetic Programming Approach. *Pages 1135–1142 of: GECCO 1999: Proceedings of the Genetic and Evolutionary Computation Conference*. Orlando, Florida: Morgan Kaufmann, San Francisco.
7. Miller, J. F., & Thomson, P. 2000. Cartesian Genetic Programming. *Pages 121–132 of: Proceedings of the 3rd European Conference on Genetic Programming*. Edinburgh: Springer Verlag, Berlin.
8. Langdon, W. B., *The Evolution of Size in Variable Length Representations*. In 1998 IEEE International Conference on Evolutionary Computation, pages 633–638, Anchorage, Alaska, USA, IEEE Press, 1998
9. Smith, M. J., 1964. Group Selection and Kin Selection. *Nature*, **201**:1145–1147.
10. Menezes, T., *Lablove - Laboratory of Life on a Virtual Environment*. http://sourceforge.net/projects/lablove, 2007.
11. Dawkins, R. and Krebs, J., 1979. Arms Races Between and Within Species. *Proceedings of the Royal Society of London*, **205**(1161):489–511.