Optimizing Self-Organizing Control Architectures with Genetic Algorithms: The interaction between natural selection and ontogenesis

Nikolaus Almássy and Paul Verschure

Institute for Informatics - AI Laboratory, University Zürich-Irchel
Winterthurerstrasse 190, CH - 8057 Zürich
Technical Report Nr. 92.10
To appear in: *Proceedings of the PPSN '92, September 1992*

Abstract

To model the relationship between natural selection and ontogenesis, techniques from genetic algorithms and neural networks can be combined. We model this interaction as two complementary processes were the adaptation of the individual takes place to adjust to properties of the system environment interaction that cannot be predicted from the perspective of the genome. Adaptation of the individual during ontogenesis is modeled with a control architecture for an autonomous agent developed in the distributed adaptive control framework. Natural selection is modeled with a parallel genetic algorithm. The results indicate that "species" can evolve as an emergent property of the interaction of these two adaptational processes.

1. Introduction

In recent years Genetic Algorithms (GA) have gained much attention in AI, cognitive science and theoretical biology. Although GA's have been very successful in search and optimization, the full potential of natural selection is still not captured in these techniques. In nature there seems to be a strong interaction between selection at the phenotypic and ontogenetic level (the adaptation at the level of individuals during their lifetime). If we want to understand natural selection we need to combine these two levels of adaptation.

To model this adaptational process that takes place in two time frames, or the grand and small loops of natural and somatic selection during ontogenesis [4], we combined a parallel genetic algorithm with a control architecture for an autonomous agent. This control architecture, based on the design methodology of distributed adaptive control [13, 10] can be seen as a model of the development of categorization.

Others have applied genetic algorithms to neural networks, these attempts however were mostly concerned with issues of genetic algorithm design rather the modeling evolution as observed in nature [1, 14]. In another approach [11], it was attempted to model the interaction of individuals in a competitive environment. When we would take the above mentioned work on the combination of GA's and neural networks as a starting point in modeling the interaction between natural selection and adaptation of the individual, these attempts suffer from several problems. First, the neural networks that are considered cannot be taken as plausible models of learning and adaptation as observed in nature since they are based on supervised learning schemes. Second, in these cases the neural network is considered in isolation from realistic sensors and effectors with which an interaction with the environment can be established. Just as not every neural network can be seen as a plausible model of learning and adaptation just because it is a neural network, not each attempt to combine a neural network with a GA can be interpreted as a model of the interaction between natural selection and ontogenesis.

Adaptation at the level of the individual during ontogenesis is implemented in our case with a model based on the mechanics of classical conditioning [12]. This neural model formed the basis of a self-organizing control architecture for a mobile robot (the agent)[10, 13]. Contrary to the earlier mentioned neural models in our case ontogenesis is modeled as a self-organizing process. This is based on the assumption behind this work that these processes cannot be founded on supervised techniques (see [12] and [13] for a discussion). Adaptation must be taken as a process that develops without the aid of a supervisor (see also [2–4]). The architecture that controls the agent is defined according to a value scheme which specifies the morphology, the properties of the sensors and effectors, the initial reflexes, and local properties of the control architecture. The value scheme is determined by the genome of the agent. Through its value scheme an organism is already coarsely adapted to its environment. The value scheme, however, does not predefine a action dependent categorization of the environment. The fine tuned adaptation that is the result of forming this categorization depends on the way in which the sophisticated sensor of the agent is integrated into its actions. This integration process is under the control of the interaction between the agent and its environment and expressed in the interconnectivity of the neural fields representing the states of the sophisticated sensor and those representing states of the primitive sensors that can trigger specific actions (this will be elaborated in the next section).

Adaptation at the level of a population of individuals is implemented with a genetic algorithm [9] which modifies some local properties of the control architecture of the agent in order to produce new agents with different properties giving rise to other behavior. The GA operates only at the level of the genotype that encodes the local properties of the agent. The conversion of this genotype to the phenotype (the instantiation) is simply the decoding of a binary string into a set of five parameters that specify the local properties of the control architecture. Although the distributed adaptive control architecture is reasonably constrained from the levels of behavior and neurobiology [12] and has been shown to be also robust in applications in real robots the GA aspects of our modeling attempt do not have such a solid empirical base. From our perspective, however, this attempt to model the interaction between natural selection and ontogenesis is useful to generate intuitions on how this process actually can be modeled and what sort of constraints on the GA techniques involved are relevant.

The role of the ontogenetic adaptational process that we model is to make the system adjust to regularities of the system environment interaction that cannot be predicted from the perspective of the genome or that show such a high variability that a genetic predefinition would lead to unacceptable brittleness (e.g. [7]). Contrary to other suggestions [8], were learning and natural selection are seen as identical processes, in our proposal these are taken as complementary processes. The value scheme, which is a result of natural selection, specifies a specific interaction space of the agent with a related domain of possible sensory states. The adaptational process during ontogenesis makes the system focus in on subparts of these domains that express the actual properties of the way in which it interacts with the environment and allows a fine tuned adaptation to the environment.

2. Distributed Adaptive Control

The agent we will consider is developed according to the design methodology of distributed adaptive control[10, 13]. We will base our example on a simulated agent that learns to avoid obstacles and locates food sources in an environment.

Sensors: The agent has three sensors: a range finder (the sophisticated sensor), a collision detector, and a target sensor (the primitive sensors). The range finder spans over $-90°$ and $90°$ from the center of the robot. It consists of 37 elements that each cover a part of this receptive field. The angular resolution of these elements decreases when they get closer to the center of the agent. The system has 37 collision detectors that cover the same region as the range finder. The two target detectors can each detect their distance to the target. They are placed at $-90°$ and $90°$ degrees from the center of the robot. The difference of the distance detected by the two target sensors determines whether the target is to the left or right of the center of the agent.

Actions: The agent can perform five actions which are *retract-turn left*, *retract-turn right*, *advance-turn left*, *advance-turn right*, and *advance*. The latter action will be executed when none of the others are active. The system can only make turns over $9°$.

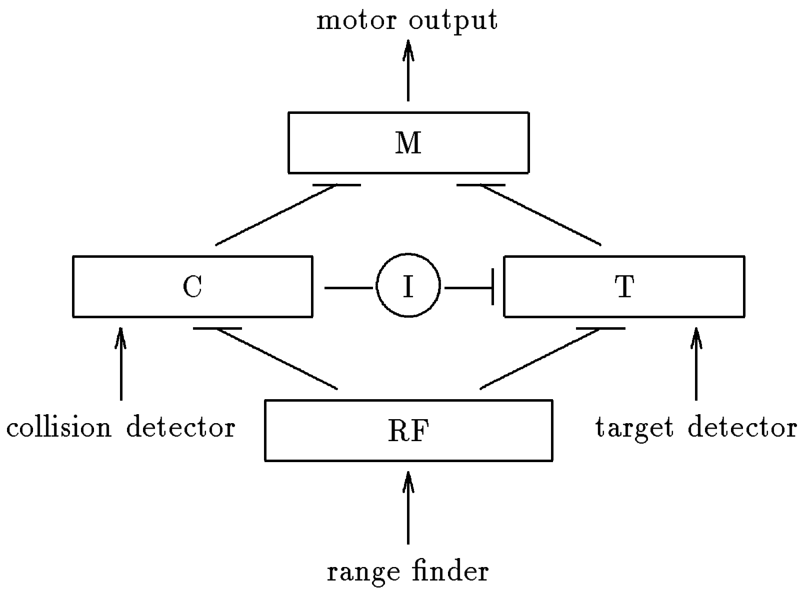

Figure 1. The basic architecture of the DAC structure (see text for explanation).

Control: The control architecture consists of four neural fields and one inhibitory unit (see figure 1). The first field (RF) consists of 37 units and receives input from the range finder. Activation a_i of each unit is determined by the inverse value of a corresponding unit r_i in the range finder. Herewith, these units code a measure for *time to contact.* The transduction function is given by:

$$s_j = f(r_j) = \frac{1}{e^{x(r_j - d^{max})}} \qquad (1)$$

where d^{max} determines at what distance from the sensor a maximal response will be given and x is a scaling factor. In these experiments d^{max} was 30 while x was set to 0.1. The second field (C), which consists of 37 units, receives its input from the collision detectors. Here too, there is a one to one mapping between units and elements of the sensor. If one element of the collision detector is triggered, the corresponding unit will get activation value 1. The third field (T), consisting of two units, is related to the difference between the distances calculated by the two target detectors. If the distance detected by the right target detector is lower then the one detected by the left one, the first unit will be switched on and vice versa. The units of C and T also receive input from RF:

$$h_i^\lambda = c_i^\lambda + \sum_{j=1}^{N} K_{ij}^\lambda s_j \qquad (2)$$

where λ indexes C and T, h_i^λ denotes the input to unit i in C or T, c_i^λ is the input from the related primitive sensor, s_j is the activation of the units in RF and K_{ij}^λ the weight of the connection between the units of RF and T and C. These weights are updated according to:

$$\Delta K_{ij}^\lambda = \frac{1}{N}[\eta^\lambda s_i^\lambda s_j - \epsilon^\lambda \overline{s}^\lambda K_{ij}^\lambda] \qquad (3)$$

Where N defines the number of units in RF, η^λ the learning rate, ϵ^λ the decay rate, and $\overline{s}^\lambda$ the average activation in field λ. Due to this integration process expressed in (3) the control of the actions of the agent shifts from the primitive sensors to the sophisticated sensor. Where the control of C and T was predefined the way in which RF triggers actions is dependent on the specifics of the system environment interaction. The last field (M) consists of 5 command neurons which code the basic actions the system can perform. The relations between T, C, and M are pre-wired and not modifiable: a collision to the left will automatically trigger a retract and turn to the right action. A collision to the right will trigger the reverse action. If the target is detected to the left of the system a turn left and move forward action will be executed. If the target is detected to the right the system will advance and turn to the right. One unit in C or T can trigger a unit in M. Last, there is the inhibitory unit I that regulates the relation between C and T. Activation in C will increase its activation. When the activation of I is above a certain threshold, τ^I the output from T to M will be inhibited. The system will then suppress the approach to the target.

3. Fitness and Adaptation

The local properties of the control architecture define a large space of possible behaviors in a given environment. In order to evaluate the performance of the agent, we define the

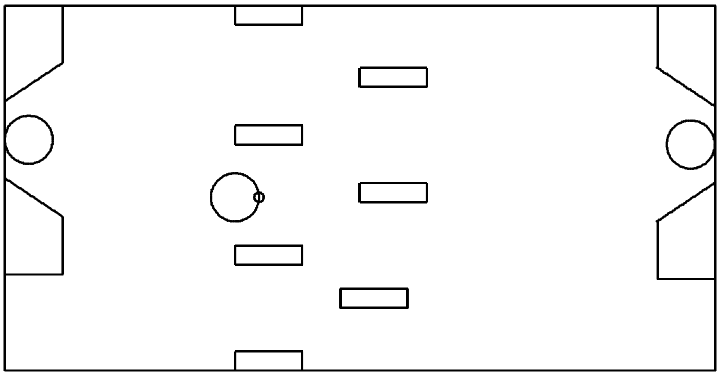

Figure 2. The simulated environment in which the agent was tested. The two circles on both sides represent targets, and the circle in the center represents the agent which faces the target on the right side.

quality (or fitness) of the behavior by the number of collisions, and the number of targets found by the agent. The fitness of one agent (A) is calculated with the following function:

$$f(A) := 3 * N\_targets - N\_collisions \tag{4}$$

This function puts a stronger weight on the number of targets that were found, to force the agent to explore the environment. It is important to note that this function expresses also the properties of the environment as well as of the agent.

The environment to which the agent had to adapt, was filled with several obstacles and contained two targets (see figure 2). The agent is supposed to avoid obstacles and approach targets. To find the targets the agent had to alternate between the left and the right target and to cross the "wall" of obstacles in between. The agent could only detect a target from within half the length of the environment. A target is found by the agent when it is touched, after which it is turned off until the other target is found. The following five parameters of the control architecture of the agent were modified by the GA: the learning rate and decay for the connections between RF and both C and T and the threshold for the inhibitory unit I. The standard parameter values which were based on previous simulation experiments are:

$$\begin{array}{lll} \eta^C := 0.1 & \eta^T := 0.1 & \tau^I := 0.15 \\ \epsilon^C := 0.9 & \epsilon^T := 0.9 & \end{array} \tag{5}$$

These settings lead to an acceptable behavior of the agent in various tasks and environments. When these parameters are set to maladapted values various things can go wrong: If the threshold of the inhibitory unit is set too high, approach will seldomly be overruled by avoidance, which will lead to many collisions when targets are surrounded by many obstacles. If it is set too low, the agent will not reach the target if it is near an obstacle since minimum activation in C will lead to avoidance and the inhibition of approach. If the learning rate η^C is too high the agent will start avoiding all objects even if they are still relatively far away and thus not explore the environment since minimal activation in

RF will trigger activation in C. If it is set too low, too many collisions will occur since the integration of RF in the actions will only very slowly take place. Moreover, the decay must be set correctly in order to avoid a bias towards an exceptional situation that can occur during the early phases of the adaptional process. There is a strong interdependence between these parameters of the control architecture. When certain parameters are set to suboptimal values, different settings of other parameters can (to some extent) adjust this handicap.

The agent adapts a strategy for completing its task only through constant interaction with the environment. Therefore adaptation is very dependent on a given environment and the properties evolved by the GA in a specific environment may not necessarily be well suited for another one.

This interdependence of the parameters makes the fitness landscape hard to analyze. It must be assumed that this fitness landscape is very rugged containing many local minima.

Since changes of the parameters of the control architecture (in our case learning rate, decay and threshold) as well as the initial conditions (like heading, orientation of obstacles) can have a discontinuous effect on the behavior of the agent, the fitness landscape (a vector valued function of the five parameters we chose to optimize) can not be differentiated, and is therefore especially suited for genetic algorithms.

4. A parallel GA

On the basis of standard GA software [6, 9] we developed a GA with a parallel master-slave architecture. One central host was generating new parameter sets by applying genetic operators to the current population of agents. The evaluations were then distributed over a local area network of several SUN workstations.

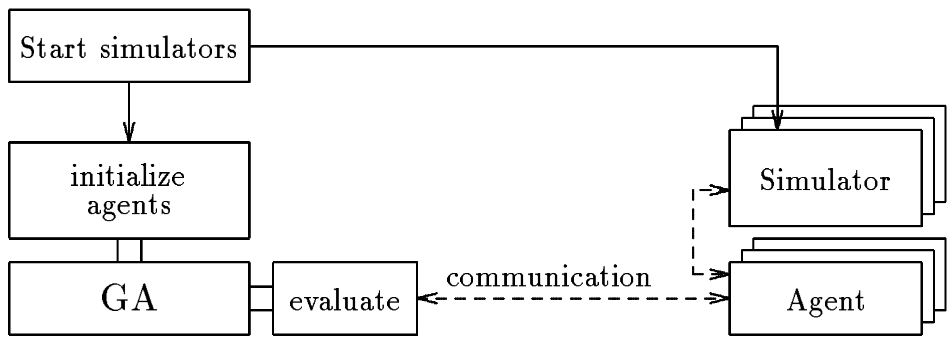

Figure 3. The architecture of the parallel GA including the setup of the simulation environments (see text for explanation).

In figure 3 the architecture and implementation of the GA is drawn: After starting the simulation environments, N agents are initialized and started by the evaluation part of the GA. The agent reports its performance via a communication link and communicates with the simulation process via ethernet.

Each agent was represented by a genotype of 15 bits that encoded the parameters. The evaluation of one agent took one hour (CPU time on a SPARCstation 1) in the Bugworld simulator [5].

5. Results

In the first experiment we used a population of ten agents that each were given 2000 steps for collecting targets in the environment shown in figure 2. The original agent with the default parameters (5) found six targets, but had 102 collisions. Particularly at the end of this evaluation, the agent performed worse than in the first 1000 steps. This was due to a trend where the weights of the connections between RF and C and T fields converged into a local minimum that caused a maladapted behavior (E.g. The agent was constantly approaching a target and then retracting again, before it could touch it).

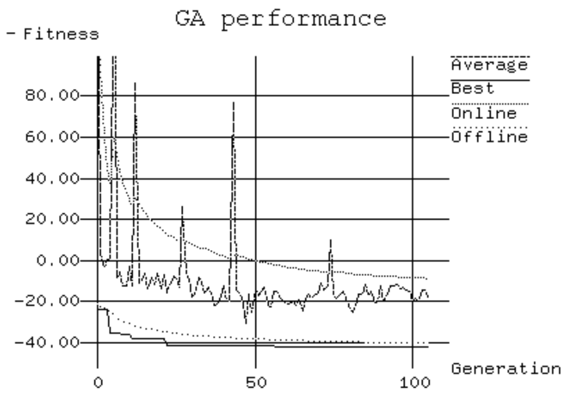

Figure 4. Evolution of the population of agents.

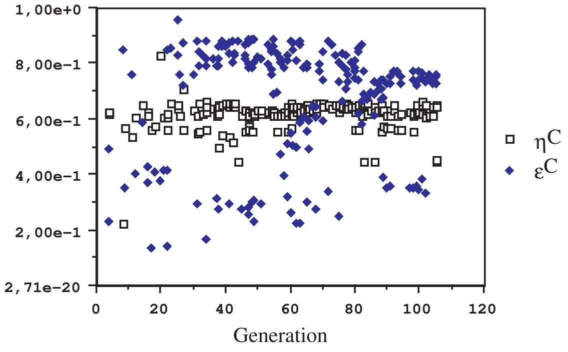

Figure 5. Evolution of η^C and ϵ^C.

In figure 4 the development of the fitness of the population of is shown. The dashed line represents the average performance of the whole population. Some peaks in this graph were caused by agents that left the environment and because of that endured many collisions. After 50 generations the average fitness of the population seemed to have converged into a more or less stable region around $f = -20$. Nevertheless the properties that were modified by the GA were still changing. This can be seen in the figures 5-7 where the change of the properties of the best 200 agents over 105 generations is displayed.

Figure 5 indicates a significant change in the optimal value for the decay rate ϵ^C after Generation 55. This drift in the phenotypic structure of the population is also visible in figure 6 even if there is a higher variability in ϵ^T.

The best agent found by the GA was created in generation 57 and found 15 targets with only 3 collisions (the trajectory of this agent is similar to the one seen in figure 8 but had less collisions). The parameters of this agent were set to the following values:

$$\begin{array}{lll} \eta^C := 0.63049 & \eta^T := 0.65323 & \tau^I := 0.36567 \\ \epsilon^C := 0.7796 & \epsilon^T := 0.45470 & \end{array} \tag{6}$$

When we compare this to the default parameters we see a much higher learning rate for both the collision and the target connections and a moderate value for the decay. The threshold was also set to a higher value.

In figure 9 the trajectory of the best individual of the experiment is evaluated in a different environment. This agent had 19 collisions and found 6 targets. This shows the robustness of the adaptation method, but also that optimal parameter values are highly dependent on the environment.

In figure 10 the variability and interdependency of several parameters of the control architecture is shown. It shows that several sub-species evolved that performed equally well. These sub-species are defined by different values of ϵ^λ

6. Discussion

Our experiment was a step towards modeling the interaction between natural selection and ontogenesis. To become a serious model of this phenomenon some features need to be added. For instance, in our case the genome is immediately translated into a structure that drives the behavior of the agent. In nature however, the translation of a genotype into a phenotype is the result of a developmental process where at any stage there is already a strong system environment interaction which already leads to a high variability and robustness[3]. Although our system shows that with this control architecture, species already evolved an inherent robustness against environmental changes showing relatively well adapted behavior for a large parameter domain, the full potential of natural systems is not fully explored yet. Therefore we believe that GA's have to be combined with developmental processes.

Another property that is not included in our approach is the flexibility of natural selection of not only finding the optimal solution in the fitness landscape but also to explore and change the fitness landscape. So we need to find ways to change and enlarge the fitness landscape by adding dimensions (generating new parameters). This should be done however without violating the self-organizing nature of these processes.

References

1 Richard K. Belew, John McInterney, and Nicol N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. Technical Report CS90-174, Cognitive Computer Science Research Group, Dept C-014, University of California, San Diego, 1990.

2 J. P. Changeux. *Neuronal Man: The biology of man.* Oxford University Press, Oxford, 1985.

3 G. M. Edelman. *Neural Darwinism: The theory of neuronal group selection.* Basic Books, New York, 1987.

4 G. M. Edelman. *The Remembered Present: A biological theory of consciousness.* Basic Books, New York, 1989.

5 Louis Goldstein and Kyle Smith. Bugworld, a distributed environment for the study of multi-agent learning algorithms. Technical report, University of California at Santa Cruz, Dept. of Computer Science, Santa Cruz, 1991.

6 John J. Grefenstette and Nicol N. Schraudolph. A user's guide to genesis 1.2ucsd. Technical report, Naval Research Laboratory, Washington, D.C. 20375-5000, 1987.

7 Calvin B. Harley. Learning the evolutionary stable strategy. *Journal of Theoretical Biology*, 89:611–633, 1981.

8 Geoffrey E. Hinton and Steven J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

9 John H. Holland. *Adaptation in natural and artificial systems.* Ann Arbor : University of Michigan Press, MI, 1975.

10 Rolf Pfeifer and Paul Verschure. Distributed adaptive control: a paradigm for designing autonomous agents. In *Proceedings of the First European Artificial Life Conference.* MIT, Cambridge Mass., 1992.

11 Thomas S. Ray. Is it alive or is it ga? In *Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.

12 Paul Verschure and A.C.C Coolen. Adaptive fields: distributed representations of classically conditioned associations. *Network*, 2:189–206, 1991.

13 Paul Verschure, Ben Krose, and Rolf Pfeifer. Distributed adaptive control, the self organization of structured behavior. In *Robotics and Autonomous Systems.* Elsevier, 1992.

14 Darrell Whitley and Thomas Hanson. Optimizing neural networks using faster, more accurate genetic search. In *Third International Conference on Genetic Algorithms.* Morgan Kaufmann, June 1989.

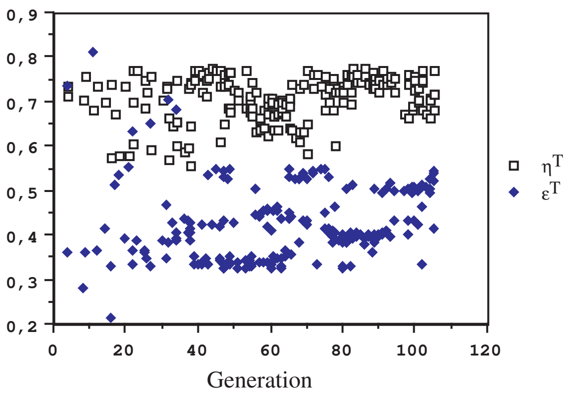

Figure 6. Evolution of η^T and ϵ^T.

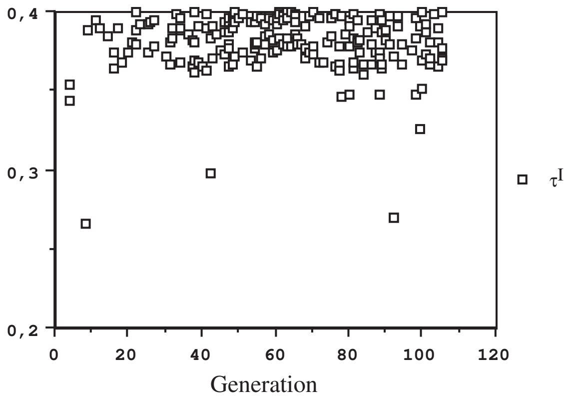

Figure 7. Evolution of τ^I.

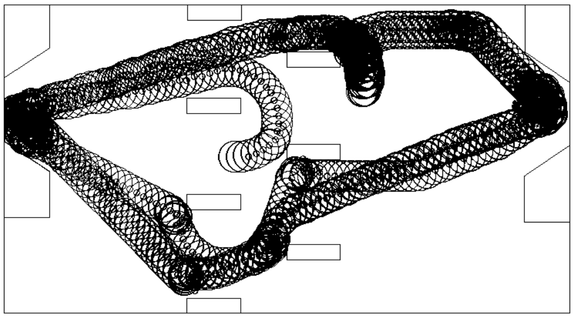

Figure 8. Path of the "default" agent in the environment.

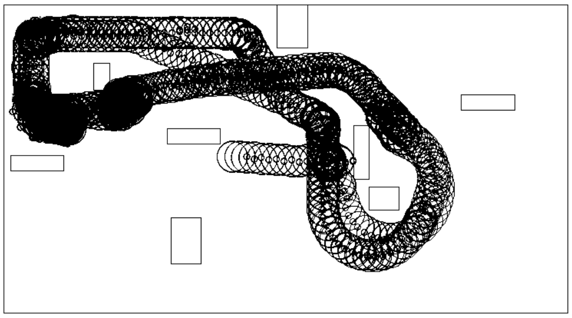

Figure 9. Path of the best agent in a different environment.

Gen	Coll	Trgts	Fit	η^C	ϵ^C	η^T	ϵ^T	τ^I
61	3	15	-42	0.62055	0.84664	0.63729	0.45552	0.39976
22	4	15	-43	0.60756	0.41396	0.60548	0.63324	0.39977
30	4	15	-43	0.62838	0.75606	0.70185	0.38654	0.37230
58	4	15	-43	0.62441	0.39562	0.68096	0.34014	0.39568
59	4	15	-43	0.61134	0.32046	0.67265	0.45542	0.37611

Figure 10. A selection of the parameter of the best agents developed by the GA.