NeatSqueak on Wheels: Neural Networks Applied to Movement Optimization

Hugo Peixoto[1,2], João Portela[1,2], Rui Teixeira[1,2], Filipe Castro[1,2], and Luís Paulo Reis[1,3]

[1] Faculdade de Engenharia da Universidade do Porto, Portugal
[2] NIFEUP - Núcleo de Informática da Faculdade de Engenharia da Universidade do Porto, Portugal
[3] LIACC - Laboratório de Inteligência Artificial e Ciência de Computadores da Universidade do Porto, Portugal
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
`{ei03047,hugo.peixoto,ei04034,ei03095,lpreis}@fe.up.pt`
`http://www.fe.up.pt`

Abstract. In *CiberRato* competitions, deliberative agent implementations commonly use path planning algorithms. This creates the need for a path following module, which would control the mouse motors and drive him through a set of calculated waypoints. A common approach is manually defining a decision tree, where the threshold values and motor outputs are empirically determined, obtaining a sub-optimal implementation. We've addressed this problem using optimization techniques based on unsupervised learning. A neural network based solution was developed through the use of genetic algorithms. Comparative data supporting that this approach surpasses manual decision tree implementations is presented for waypoint following behaviours. The results achieved enabled us to conclude that the development platform may be a valuable tool for *CiberRato* competition participants.

Keywords: *MicroRato*, *CiberRato*, Artificial neural network, Genetic algorithm, Movement, Optimization, NEAT, Intelligent Robotics

1 Introduction

CiberRato is a set of competitions, held by the University of Aveiro, composed by a simulation of a virtual environment, in which several agents are located. The environment consists of a 2D rectangle-like arena, which is populated by obstacles and one or more targets, each one signaled by a beacon. The virtual robots are placed in a predetermined starting grid. The competitors must provide software which leads the agents to accomplish a set of predefined goals[1].

The virtual robots' morphology is simple. They're composed of a circular base, equipped with sensors, actuators and command buttons. Information is sent and received through a connection with the simulator, which adds noise to both the sensor readings and the actuator values[1].

In mainstream competitions, the main goal consists of finding one or more targets, which are located in arbitrary positions of the map. The score is usually based on the number of collisions and on the time they take to complete their tasks.

This environment poses a series of challenges regarding the algorithmic nature of autonomous robot control[2]. World mapping, path planning, waypoint following, error control and exploration behaviours can be seen as examples of such challenges.

Most documented agent architectures implemented by the participants of these competitions include a waypoint following module. In reactive architectures, it can be seen that there is a behaviour which moves the agent in the direction of a beacon[3]. In more complex architectures, there are several implementations of various path planning algorithms, which need to be complemented with a behaviour that moves the agent to a defined point[4][5][6][7].

This shows us that the waypoint following behaviour is widely used throughout several architectures, and it is present in almost every agent implementation. This being true, the optimization of this module brings a general increase in most agents' performance.

The architectures described throughout several articles[3][4][5][6][7] explain that the behaviour is usually tuned manually, without the application of an optimization technique, suggesting that the results obtained are sub-optimal solutions.

The only actuators available to the agent are the powers supplied to both left and right engines, represented as real values in the range $[-0.15, 0.15]$. We propose that the optimal values can be calculated from a set of parameters related to the set of waypoints and to the current motors' power. To determine the relationship between these two, we suggest a technique based on artificial neural networks for the optimization of the agents' movement, when following a set of waypoints.

First, the algorithmic background of the explored approach will be shortly exposed. In section 3, the architecture beneath the *CiberRato* neuroevolution platform is described, along with some choices like the fitness function that distinguishes the individuals. Next, section 4 documents the data gathered through the study, along with its analysis. Finally, in section 5, we will discuss the obtained results, detailing both the manual and the machine learning approaches.

2 State of the art

Improving the waypoint following behaviour performance can be achieved using reinforcement learning techniques. The application of these methods is attractive, as there is no need to specify how the task is to be achieved[8].

According to [9], there are two main strategies for solving reinforcement-learning problems. The first one is based on the search in the space of behaviours, trying to find one that performs well. The second approach uses statistic techniques and dynamic programming methods to estimate the utility of each action. We opted for the first strategy, in which evolutionary algorithms are commonly used techniques.

NEAT - Neuroevolution of augmenting topologies - is a neuroevolutionary technique which uses a genetic algorithm for the evolution of artificial neural networks[10], where both topology and connection weights are evolved.

With NEAT, one needs to design and implement a fitness function that evaluates the neural network according to the training goals. As previously stated[8], this technique requires no knowledge of expected network outputs, topology and connection weights.

Radi and Poli[11] showed that NEAT halts its network improvement after a smaller number of generations than other TWEANN[4] algorithms, which leads us to believe that there is no significant gain in evolving a population for a big number of generations, opting instead for recording several runs of the evolution process.

3 Approach

Through the application of the NEAT algorithm, we developed neural networks capable of driving the agent towards a specified waypoint. The agent's motors' power values are directly obtained from the neural network's outputs. This neural network will be refered to as *brain*.

The agent's path following architecture is presented in algorithm 1. The `inputs()` method used in line 7 will be discussed in section 3.2.

```
1  set_target(first_waypoint());
2  while simulation is running do
3      read_sensors();
4      if on top of target waypoint then
5          set_target(next_waypoint());
6      end
7      (leftMotor, rightMotor) = brain.activate(inputs());
8      drive_motors(leftMotor, rightMotor);
9      request_sensors();
10 end
```

Algorithm 1: Basic agent algorithm

[4] TWEANN: Topology and weight evolving artificial neural networks

The evaluation of each brain depended on the number of waypoints that the respective agent could reach within the simulation time limit. For each waypoint reached the agent was rewarded with 1 fitness point. To further distinguish between agents who have reached the same number of waypoints, they were additionally rewarded according to the distance covered towards the next waypoint. This is shown in the following formula.

$$f = N_w + 1 - \frac{D(P_a, P_{w+1})}{D(P_w, P_{w+1})}$$

- N_w - number of waypoints reached
- $D(x, y)$ - distance between point x and y
- P_a - final position of the agent
- P_w - position of the last visited waypoint
- P_{w+1} - position of the next waypoint

3.1 Evolution platform and server configuration

The evaluation of each brain is made through the use of *CiberRato* simulator. In order to reduce the evaluation time of all brains in a generation, we opted to develop a distributed platform specifically for the evolution and evaluation of *CiberRato* brains.

The platform requires the definition of a set of paremeters. First, one must specify the *CiberRato* server configuration. Second, the set of labyrinths in which the brains will be evaluated must also be specified.

Regarding the server configuration, we opted for the standard parameters, with one exception. The server usually adds gaussian noise to the sensors and actuators. In our experiments, this noise was disabled. This decision was made so that this first attempt could focus on the path following problem. An error correction module could be added in further experiments through the use of modular neural networks[12], for instance.

3.2 Test and control groups

There were several parameters which could serve as the neural network inputs. To explore the results that each set of parameters would yield, we designed a set of test groups comprising several input configurations. These groups' characteristics are illustrated in table 1.

Each waypoint is represented by two input nodes to the network, one for the distance and another for the angle. Motor information requires two additional input nodes, one for each previous motor power value. The number of inputs and outputs required by each group is also represented in table 1.

A control group was used, this group implemented a manually tuned waypoint following module — group Upsilon. This implementation is based solely on the angle between the agent's direction and the next waypoint.

	Waypoints	Motor	Input nodes	Output nodes
Alpha	One	No	2	2
Beta	One	Yes	4	2
Gamma	Two	No	4	2
Delta	Two	Yes	6	2

Table 1. Groups' characteristics

3.3 Tests Description

The agent should be capable of optimizing a series of different movements. It should be capable of following a straight path without veering, make sharp and shallow turns clock and counter-clockwise and also reverse it's orientation. A set of training maps were devised in order to evaluate each agents' efficiency performing these tasks.

To validate the generalization of the resulting networks, there was the need to create a map in which several patterns could be analysed. The map waypoints arrangement can be seen in figure 1.

Due to the deterministic nature of the neural network architecture and the absence of noise, the same simulation run more than once will always yield the same results.

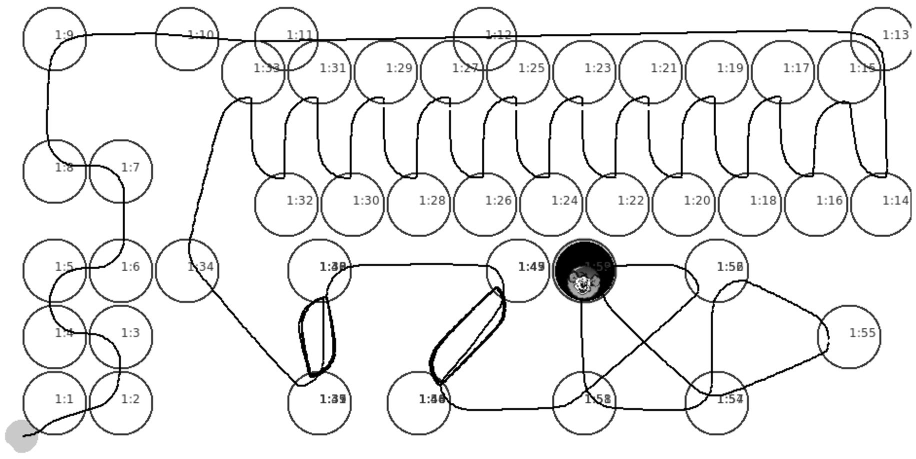

Fig. 1. Validation (Testing Grounds)

3.4 Evolution Methodology

The test groups were all evolved using the same parameters. Their population size was 104 individuals and the evolution process went on for 100 generations. The other genetic algorithm parameters were left untouched, and the ANJI platform default values were used. There are some settings worthy of notice. ANJI's capability to generate recurrent connections in its ANN was disabled, so that the network complexity wouldn't be heightened. We propose that the chosen input values are enough to determine the next simulation state. This being true, the neural network should need no extra information to provide an optimal mapping function between the inputs and the outputs.

The initial topology was set as a fully connected network, which is the simplest network in which all the outputs are dependent on every input.

4 Results

When the neuroevolution runs finished, all the champions were placed on the validation course. For each group, the group champion was defined as the run champion which had the best performance on this map. Their fitness was recorded on table 2. As described by the fitness function, these values represent, approximately, the medium number of waypoints that each brain reached per training map.

	Fitness
Alpha	7.891
Beta	8.197
Gamma	7.973
Delta	7.895

Table 2. Group champions' fitness

Group champions were then evaluated on all training maps and their performance recorded on table 3. The control group, Upsilon, was also evaluated.

	Testing grounds	Map 1	Map 2	Map 3	Map 4
Upsilon	2433	275	273	283	468
Alpha	**1653**	197	200	200	**320**
Beta	1714	**183**	196	197	324
Gamma	1674	187	**189**	**195**	**320**
Delta	1676	189	194	197	325

Table 3. Number of cycles that each group champion took to finish several maps

The fitness evolution was also recorded, as shown in figures 2, 3, 4 and 5. There are three lines in each graph, representing the minimum, average and maximum fitness per generation.

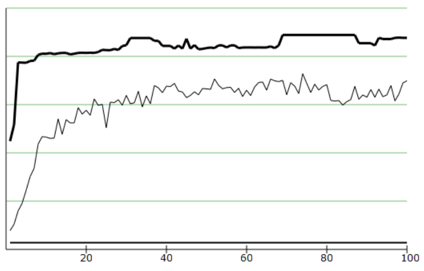

Fig. 2. Alpha fitness evolution graph

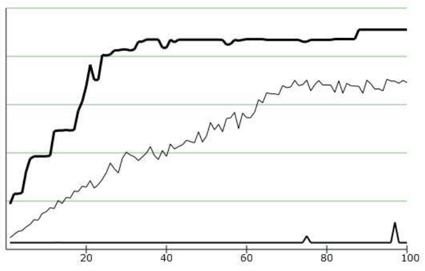

Fig. 3. Beta fitness evolution graph

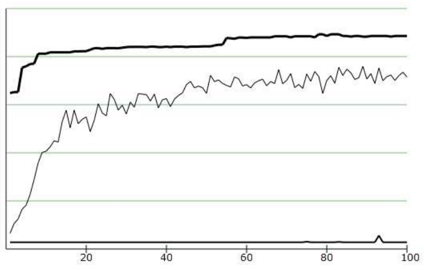

Fig. 4. Gamma fitness evolution graph

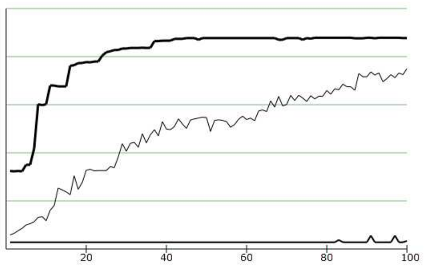

Fig. 5. Delta fitness evolution graph

4.1 Analysis

Observing table 3, it is possible to calculate the average improvement of the test groups upon the control group. The average improvement is $\mu = 30.47\%$, with a variance of $\sigma^2 = 2.5\%^2$ Although the number of samples is considerably small, the improvement is significative.

From figures 2 and 4, one can observe that both alpha and gamma groups have a high fitness improvement around the first 50 generations, and little gain in the following 50. This could be an indication that there is little room for further optimization. It should also be noted that the gamma group, which uses motor speed information, has slightly higher average fitness values. This improvement is possible due to the server's inertia simulation.

Groups with two waypoints, whose fitness evolution is represented in figures 3 and 5, have a significantly slower progress. While beta group seems to stabilize around generation 80, the delta group doesn't appear to achieve a stable position within the 100 generations.

Improvement gained from the addition of the second waypoint information is not evident when alpha and beta groups' average values are compared. Nevertheless, it can be seen that beta groups yielded higher champion values.

As expected, as the neural network base complexity grows, the neuroevolution development time increases accordingly.

5 Conclusions and further study

Optimization in *CiberRato* waypoint following behaviours using artificial neural networks offers significant improvement over manually tuned approaches. Our results support that there is room for approximately 30% gain in the time spent following waypoints.

Considering how little the difference between the first and the second place can be (regarding the *CiberRato* competition), the improvement provided by our proposed module may prove decisive, specially when the agent's architecture is heavily based on waypoint following.

Although some useful observations can already be extracted from this experiment, further statistical data gathering would allow us to develop a correlation between the groups' performance and their network inputs. The number of generations for which the test groups ran didn't prove to be sufficient in the delta (two waypoints with motor information) group. Further study with extra generations could help determine the stabilization point.

The *CiberRato* neuroevolution platform, which integrates the *CiberRato* simulation system and the ANJI platform, can prove to be a valuable asset in further neural network optimization tasks. This will ease the development of further projects, allowing developers to focus on the design of fitness formulas, training courses and other neuroevolution parameters.

The presented study can be extended to integrate obstacle detection and avoidance. This would prove valuable in *CiberRato* competitions, where mapping is usually probabilistic — adding extra flexibility to the proposed behaviour.

Acknowledgements

This work was partially supported by FCT Project PTDC/EIA/70695/2006 "ACORD - Adaptative Coordination of Robotic Teams".

References

1. Nuno Lau and Artur Pereira. Manual do ciberrato. Technical report, Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, 2008.
2. Nuno Lau, Artur Pereira, Andreia Melo, António Neves, and João Figueiredo. Ciber-rato: Um ambiente de simulação de robots móveis e autónomos. *Revista do DETUA*, 3(7):647–650, 2002.
3. Emmanuel Lomba. R_zero: Um agente virtual "reactivo ma non troppo". *Revista do DETUA*, 3(7):666–669, 2002.
4. Pedro Ribeiro. Yam (yet another mouse) - um robot virtual com planeamento de caminho a longo prazo. *Revista do DETUA*, 3(7):672–674, 2002.
5. Luís Paulo Reis. Ciber-feup - um agente para utilizar o simulador ciber-rato no ensino da inteligência artificial e robótica inteligente. *Revista do DETUA*, 3(7):655–658, 2002.
6. Robin Lilja, Johan Hägg, Batu Akan, Fredrik Ekstrand, Hüseyin Aysan, Jörgen Lidholm, and Moris Benham. Technical description of the (l)ost agent. *CiberMouse@RTSS 2008 - The 29th IEEE Real-Time Systems Symposium*, pages 17–20, 2008.
7. Álvaro Monteiro, Fábio Aguiar, and Sara Carvalho. Speedygonzalez: A path planning and plan execution agent. *CiberMouse@RTSS 2008 - The 29th IEEE Real-Time Systems Symposium*, pages 21–23, 2008.
8. Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.
9. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
10. Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(1063-6560):99–127, 2002.
11. Amr Radi and Riccardo Poli. Discovering efficient learning rules for feedforward neural networks using genetic programming. pages 133–159, 2003.
12. Farooq Azam and Farooq Azam. Biologically inspired modular neural networks. Technical report, 2000.