ARTIFICIAL INTELLIGENCE SYSTEMS FOR ELECTRON BEAM PARAMETERS OPTIMISATION AT THE AUSTRALIAN SYNCHROTRON LINAC*

E. Meier†, M.J. Morgan,School of Physics, Monash University, Melbourne 3168, Australia
S.G. Biedron, Argonne National Laboratory , IL 60439, USA
and Sincrotrone Trieste, Italy
G. LeBlanc, Australian Synchrotron, Melbourne, Australia

Abstract

We report the development of an artificial intelligent (AI) system for the optimisation of electron beam parameters at the Australian Synchrotron Linac. The system is based on state of the art developments in Artificial Intelligence for video games and is adapted here to beam parameters optimisation problems. It consists of a genetically evolved neural network that mimics an operator's decisions to perform an optimisation task when no prior knowledge, other than constraints on the actuators, is available. The system's decisions are based on the actuators positions, the past performance of close points in the search space and the probability of reaching a better performance in the local region of the search space.

INTRODUCTION

With a view to control the energy and bunch length at the FERMI@Elettra Free Electron Laser (FEL) [1], the present study considers novel techniques for adaptive control. This project aims to adapt state of the art developments in Artificial Intelligence for video games to machine optimisation and control applications. Parallels are made between a "game agent" navigating in a battle field and an "optimisation agent" navigating in a search space. The idea is to evolve intelligent systems that can learn and adapt from their interaction with their environment through time.

The optimisation tool that has been developed for optimisation of the Australian Light Source is presently built for a 2D search space (i.e. two actuators) and can be further extended to N-dimensional problems. This tool is the first step towards adapting state of the art developments in video games, in order to control a linear accelerator. We aim to evolve a system that can make decisions in a similar way to what an operator would do. This is of particular interest for FEL accelerators as the parameters need to be readjusted in order to meet users' requirements regarding the light produced. In new FEL facilities, this is currently done by an operator loading the configuration that produces the closest beam parameters to those desired, and making adjustments until meeting the requirements. An A.I. system that can perform this task in an efficient way could save considerable time.

The basic idea is to develop a system that can learn through time and from its interaction with the machine. This is important, for example when jitter conditions change or when an element (either a controllable, magnet, etc) is faulty. When another controllable is to be used instead, the adaptive controller could quickly learn the response of the new controllable. The use of an AI system instead of could save considerable time in optimising the accelerator parameters.

BACKGROUND ON NEURAL NETWORKS

A neural network (NNET) consists of an interconnected group of artificial neurons as shown in Fig. 2. Each neuron receives stimuli from other nodes in the network; each of these inputs to a node has a "weight" w associated with it as well as an activation function, which tells a node when to fire. A neuron may also add a "bias" value θ, to the weighted inputs and any bias is passed through the activation function; the resulting value is available as the node output. Commonly used activation functions are linear, hyperbolic tangent, sigmoid, or gaussian. Gaussian networks are also known as "Radial Basis Function Networks" (RBFN) due to the radial nature of the activation function [2, 3].

During the training phase, the network is presented with an input vector and the resulting output vector is compared to the desired output vector; the network weights are then adjusted by a learning algorithm.

* Work supported by Monash University, Sincrotrone Trieste and the Australian Synchrotron.
† evelyne.meier@synchrotron.org.au

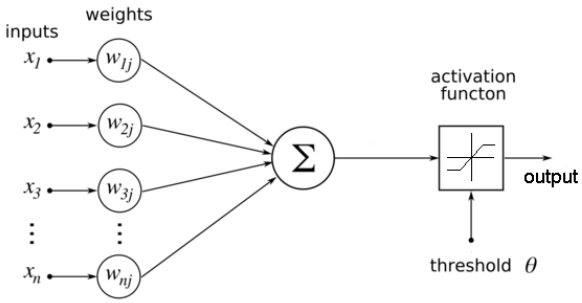

Figure 1: Schematic diagram of an artificial neuron. The node receives inputs from other nodes, which are multiplied by their respective weights and fed into the activation function.

THE NEURO-EVOLUTION THROUGH AUGMENTING TOPOLOGY (NEAT) TECHNIQUE

This technique was introduced by K. Stanley at the University of Texas, Austin (see Stanley et al. 2002). The motivation for the development of such system comes from the need of more adaptive behavior of game agents. In current video games, the actions of an agent are pre-programmed and there is little or no room for learning. This makes an agent's behavior predictable and the games less entertaining. To overcome this, Stanley et al. developed agents that can learn through their interactions with the player during the game. In that way, the game becomes more interesting for the player as the actions of the agent are not predictable. Their approach consists of genetically encoded neural networks, which are evolved over time, according to their performance. During the process, the network is told how well it performs each time it makes a decision, and evolution will select the fittest networks to generate a new generation. This training technique is an adaptation of a training method known as reinforcement learning [5].

Unlike conventional training techniques, NEAT acts on both the weights and the structures of networks to evolve individuals. A population of individuals is evolved by crossing over individuals though generations. Individuals are selected for cross over according to a fitness function. NEAT evolves networks from a simple initial structure and only complexify structures when necessary. The evolution process only stops when an individual has reached a desired level or fitness or when the maximum number of generations has been reached.

With NEAT, the whole population is evolved until an individual is considered successful in the task. This takes up to a few hundred generations and must be performed off-line. This would be impractical for video game applications, where the network would have to learn from its environment in real time [6, 7]. For this, an adaptation of NEAT to real time problems known as real time NEAT (rtNEAT) was developed. With this approach, a neural network takes control of a game agent for a limited time only. The structure of an agent is therefore evolving as the game is played, making it possible to develop more sophisticated behavior in real time and according to situations encountered during a specific game. This approach is of particular interest for applications in a real accelerator environment. Because the idea is to build a controller that eventually learns from its interaction with the machine in real time, it is necessary to have a system that does not rely on an off-line training.

APPROACH

Here we consider parallels between the evolution of a game agent in a battle field and an optimisation agent in a 2D search space.

To win a battle, a game agent has to make decisions based on the information it receives on its environment. Let's consider the situation in Fig. 3. In this figure, the game agents are the small dots, whose aim is to destroy the two square turrets. They also need to learn to navigate through the maze in order not to get trapped against a corner and getting shot. An agent's actions include shooting, moving forward, turning or jumping. Information on its environment can take into account the position of an enemy and whether this enemy is firing in its direction. Proximity of wall and other obstacles are also part of the inputs. Each time an agent takes an action, il will be rewarded or penalised depending on the outcomes. For example, if the agent has shot a turret it will receive a reward point. On the other hand, negative points are attributed when the agent is informed that the enemy is firing in its direction and the agent did not do anything about it, or took a wrong action [8].

A direct similarity with the game agent is that the optimisation agent must remain within the specified boundaries of the search space. This is very important because for an on-line optimisation actuators have physical limits. For instance, the voltage of an RF section or the current of a focusing magnet must remain below the safety limit. In a video game, the agent develops strategies to shoot the turrets. It is therefore attracted to the positions occupied by these latter objects in order to destroy them. In a similar way, the optimisation must be attracted to local maxima, by working its way towards the direction that increases the value of the objective function. In the video game, when a turret is destroyed, the agent would look for the existence of further enemy targets. Similarly, the optimisation agent is looking to find other maxima, if the value of the current local maxima is not satisfactory.

Like the game agent, the optimisation agent only has a limited range of action. It can only take little steps in the search space to try to improve the machine performance, and is only provided with information of past trials close to its current location. This is very important because it considerably reduces the amount of data that needs to be processed by the neural network.

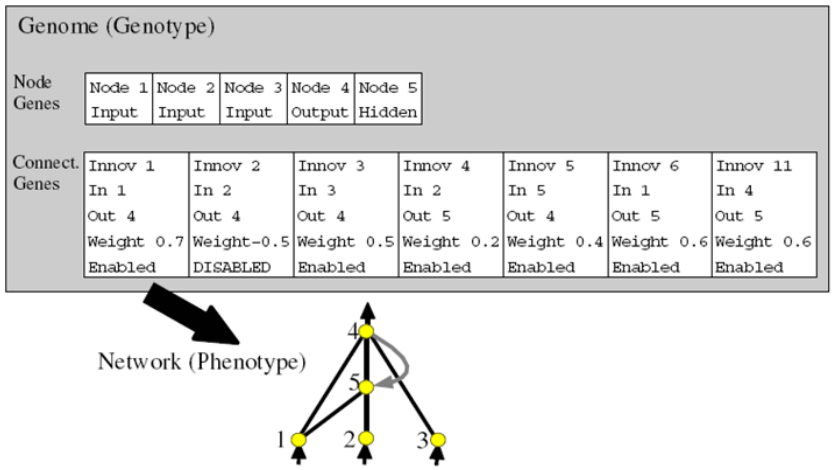

Figure 2: Encoding of NEAT. The neural network is genetically encoded with two sets of genes; the nodes genes and the connections genes. The node genes identify a nodes ID number and its type: input, output or hidden. The connection genes encode the start and end nodes of the connection as well as the weight and whether the connection is active or not. Figure taken from [4].

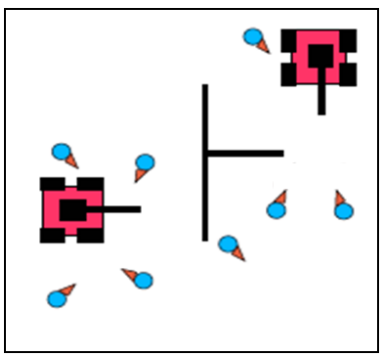

Figure 3: In a video game the agent has to navigate through a maze and aim to shoot the enemy turrets. Figure taken from [6].

FUTURE RESEARCH

The next step consists of the generalisation of the technique to N-dimension search spaces from the 2-D structure. This will complete the structure of the system. Simulations and experiments in an operational accelerator environment should validate the functionality of the optimisation agent.

A longer term project consists of adapting the technique to optimisation problems. This can be realised with two different approaches. The first consists of using the optimisation agent to optimise the parameters of an existing control algorithm. For example, it could be used to search for the most appropriate gains of a PID algorithm. In a second approach, the agent could perform the control directly. In the optimisation experiments the system had as its aim to maximise the value of an objective function. In a similar way, in a control problem the agent would have to minimise the deviation of the electron beam parameters from the desired settings.

REFERENCES

[1] Sincrotrone Trieste. *FERMI@Elettra Comissionning Design Report*, January 2007.

[2] G.W. NG. *Application of neural networks to adaptive control of nonlinear systems*. Research Studies Press, 1997.

[3] M.J.L. Orr. Introduction to radial basis function networks. April 1996.

[4] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10:99–127, 2002.

[5] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Morgan Kaufmann, 2002.

[6] K. O. Stanley, B. Bryant, and R. Miikkuulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9:653–668, 2005.

[7] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9:653–668, 2005.

[8] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong. Computational intelligence in games. In *Computational Intelligence: Principles and Practice. Piscataway, NJ: IEEE Computational Intelligence Society. chapter*, pages 155–191, 2006.