Autonomous Design of Modular Intelligent Systems

Pavel Nahodil, Jaroslav Vítků
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Technicka 2, 16627, Prague 6, Czech Rep.
Email: nahodil@fel.cvut.cz, vitkujar@fel.cvut.cz

KEYWORDS

Agent, Architecture, Artificial Life, Creature, Behaviour, Hybrid, Neural Networks, Evolution.

ABSTRACT

We propose our original system capable of autonomous design of general-purpose complex modular hybrid systems. The resulting hybrid systems will be able to employ various techniques of learning, decision-making, prediction etc. Presented topic is from Artificial Life domain, but contributes also to fields such as Artificial Intelligence, Biology, Computational Neuroscience, Ethology, Cybernetics and potentially into many other aspects of research. The autonomous design is implemented as an optimization of system topology with respect to given problem. The principle of design is based on modified neuro-evolution and can be compared to modular neural networks. One of the main requirements is standardization of communication between very different subsystems. Here, each subsystem - module implements arbitrary algorithm and is treated as a Multiple-Input Multiple-Output subsystem. First, the design of simulator used is described, then the basic principle of hybrid networks is explained with it benefits and drawbacks. Finally, simple example is mentioned.

INTRODUCTION

The main research focus of this paper lies in design of modular hybrid agent architectures. Until now, there is no superior architecture solving all possible tasks. Rather, agents are designed specifically for solving a particular problem in a given domain. Such agents are usually composed of some set of sub-systems. The selection of these subsystems and their interconnection is chosen by a human designer based on his knowledge and experience.

But the correct selection of particular modules and their connectivity is more and more complicated. Now, the research is composed from more and more isolated and specialized fields. It is very hard to track latest changes in more than one field and to maintain some general picture. Even though, it is obvious that there is some redundancy in particular approaches. For example there are the same problems solved by more abstract (e.g. Markov chains for sequence recognition (Fink 2003)) or more biologically-plausible (e.g. neural-based sequence recognition (Wang and Arbib 1990; Nguyen et al. 2012; Wang and Tsai 1998; Murre et al. 1989)) approaches. There is also redundancy how to solve the same problem between various research fields.

In many tasks, there is available some metric which says how good is a behaviour generated by the agent. Based on this metric (fitness), there is possible to design agent architectures by means of some optimization technique. There are many potential benefits of automatic design of modular systems.

THEORETICAL BACKGROUND

Design methods can be generally divided into the following three primary groups:

- **Knowledge based architecture (top down)**: Knowledge based architectures use knowledge to guide agent behaviour. Much of the debate regarding the role of knowledge within an agent aims on how it is represented within the context of the control system. Knowledge representations involve "physical structures which have correlations with aspects of the environment representation (relationship with the external world) and thus a predictive power (ability to predict from actual knowledge) for the system". Agent modules are functional blocks like planning, learning or perception blocks and the behaviours (avoid obstacles, identify object, explore the environment) emerge from the interaction of the modules.
- **Behaviour based architecture (bottom up)**: behaviour based architectures build up the system of behaviour producing modules instead of functional, as is the case in knowledge based architectures. Agent modules are behaviours such as avoid obstacles, identify object, and explore the environment. The functions of the system (planning, learning and others) emerge from the interaction of these modules and the environment. The most important phenomenon of behaviour based architectures is emergent behaviour. Emergent behaviour implies a holistic capability where the sum is considerably greater than its parts. Emergence is the appearance of novel properties in whole system. Intelligence emerges of the components of the system (Kadleček and Nahodil 2001).

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

• **Hybrid approach (combined)**: Hybrid architectures integrate a knowledge based component for planning and problem solving with behavioural components that produce robust performance in complex and dynamic domains. Strong evidence exists, that hybrid systems are found in biology, implying that they are compatible, symbiotic, and potentially suitable for use in agent control. The focus of research in this area lies in defining an interface between deliberation and reactivity, which is poorly understood so far. Hybrid models include hierarchical integration and coupled planning and reacting.

Artificial Neural Networks

Differences between Artificial Neural Networks and classical computation executed by computers are mentioned in various literature. We will only mention that von-Neumann type of computer implements from its principle deterministic computation based on exact data. The bottleneck of this architecture poses constraints to the maximum speed of computation. The fact that individual information have to be distinguished without errors causes consumption of considerable amount of energy. Also the second characteristic makes the architecture unfeasible for processing the real-world data. Compared to this, brain is computational system designed by evolution with the following requirements:

- highly power-efficient
- highly robust against errors in hardware
- works well with real world noisy data

These requirements probably determined that the brain implements highly parallel type of computation, that the computation as well as memory is decentralized. We can see that these features are often direct opposite to von-Neumann architecture, so as to many computation models based on it.

There are several reasons why the Artificial Neural Networks are not used more widely, one of them is that neural networks often work as a black-boxes and we do not have methods to design networks of appropriate size. This is addressed in our work too.

Hybrid Networks

Hybrid networks combine top-down and bottom-up approaches in one network with heterogeneous nodes. Various classification schemes of hybrid systems have been proposed so far. According to (Wermter and Sun 2000), there are three main types of hybrid systems. These are divided according to ways how the integration of symbolic and sub-symbolic systems are done. These are:

• **Unified neural architectures** - these systems rely solely on connectionist representations, but symbolic interpretations of nodes or links are possible. Often, a specific knowledge of the task is built into a unified neural architecture.

• **Hybrid transformation architectures** - transform symbolic representations into neural representations or vice versa. The main processing is performed by neural representations but there are automatic procedures for transferring neural representations to symbolic representations or vice versa.

• **Hybrid modular architectures** - contain both symbolic and neural modules appropriate to the task. Here, symbolic representations are not just initial or final representations as in a transformation architecture.

This paper proposes approach which designs naturally the third one - *Hybrid modular architectures*. Based on modules used, the resulting modular systems can be also of another type (e.g. network of only subsymbolic systems).

Evolutionary Design of Hybrid Networks

One of the most challenging goals of our research still remains to propose efficient Evolutionary Algorithm (EA) for design of networks composed of MIMO systems. In case of design networks with no constraints on topology, the space of possible solutions grows extremely fast. Nowadays, there are many solutions how to pose constraints on the topology, so the problem becomes feasible for EAs (Fekiac et al. 2011). As a good example can be mentioned Cartesian Genetic Programming (CGP) (Fišer et al. 2010) used for automatical design of logic circuits. Other useful method for design of hybrid networks is called Multi-Layered Iterative Algorithm of Group Method Data Handling (MIA-GMDH), a modification of original GMDH (Ivakhnenko 1970). In (Kordík 2006), the modification of MIA GMDG, called Group of Adaptive Models Evolution (GAME) is used for evolutionary design of feed-forward hybrid neural networks. However, all these methods employ only Multiple-Input Single-Output (MISO) neurons (subsystems).

OUR APPROACH

There are many benefits provided our approach to hybridization of Artificial Neural Networks. These range from comparison of performance between various subsystems, better general overview of available approaches and potentially autonomous research in the future. Also, there are many requirements posed by our approach, these main requirements will be described in the following sub-sections.

Reusable Modules

It is very time-consuming to re-implement outcomes of a particular research. It would be easier to test and publish resulting algorithms or systems in a reusable manner. In situation when researchers will share actual working prototypes of their work, the speed of research can be highly increased. Requirements for system which allows us to reuse work are mainly:

- Standardized communication
- Programming language independence
- Simple to use
- Decentralized - modular design.

In such a system, particular modules published by various researchers can be interconnected together in one working system. In the field of robotics, these

requirements are already addressed by system called Robotic Operating System (ROS) (Quigley et al. 2009). We do not see the reason why this system should not be used more broadly, at least also in AI or ALife research.

Unified Communication

Despite the unification of communication on some level, different modules of very different complexity typically use very different types of communications. However, our autonomous design of modular systems poses even more challenging requirement: to use one type of communication in the entire system.

Since it is known that an arbitrary function can be approximated by feed-forward neural network with only one hidden layer (Cybenko 1989; Hornik et al. 1989), it can be assumed that any behaviour of arbitrary complexity can be build by means of neural networks with more complex topologies and/or more complex models of neurons.

Therefore the neural networks can be seen as theoretical way how to build an arbitrary system. Based on this fact, we adopt neural-based communication in our design of systems. The resulting modular networks can be compared to modular neural networks (Auda and Kamel 1999), where each subsystem (module) is represented by Multiple-Input Multiple-Output (MIMO) system.

HYBRID MODULAR SYSTEMS

We define a hybrid modular system as a network of interconnected modules in a given topology. These modules communicate by means of defined connections. As mentioned earlier, there is no common communication language for all types of subsystems. Therefore the "language" used by ANNs was used in the entire system. Because of similarities with neurons, these subsystems are called *neural modules*. Typical neural module is implemented as a black-box with m inputs and m outputs, this can be seen in 1.

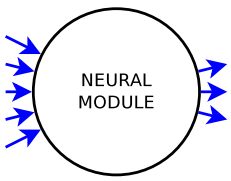

Fig. 1. Concept of neural module - black box with general number of independent inputs and outputs

In order to reuse many current implementations of systems, the ROS nodes are used here. The Fig.2 depicts how messages from ROS node are translated in modem into "language of ANNs". In current implementation, each connection represents one value of primitive type float, ideally in range $val = \langle -1; 1 \rangle$. Modem is used for modulating and demodulating this signal from/into ROS messages.

The drawback of this encoding is in the fact that encoding of more complex data types requires many

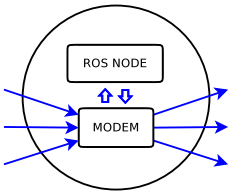

Fig. 2. Neural Module which encapsulates one ROS node. Blue lines represent neural communication (real-valued data) and blue arrows denote ROS messages. ROS node is launched as an external process by the simulator. Modem translates ROS messages into neural communication and back

neural connections. There are many techniques for autonomous design of ANN topologies. Many of them also try to deal with many connections in the networks. So this problem will be dealt with in the future by finding the appropriate encoding for these networks. The main requirements are:

- Ability to efficiently represent large numbers of connections
- Ability to represent connections of MIMO nodes.

The main advantage of such hybrid approach is in combining top-down and bottom-up approaches tightly together. It deals with one of the biggest drawbacks of ANNs: by adding (top-down designed) a subsystem of known structure, the resulting network becomes a grey-box. On the other hand, the top-down systems empowers with the main benefits of ANNs, such is robustness.

Moreover: the user can choose how much top-down and how much bottom-up architecture wants to design. This is done simply by providing more or less complex neural modules to the algorithm.

THE SIMULATOR NENGOROS

The simulator of complex modular hybrid systems has to be able to simulate systems of various complexities and levels of abstraction, ranging from Artificial Neural Networks (ANNs) to pure symbolic approaches. For these purposes we decided to combine two current software tools into one simulator, called NengoRos.

Nengo

The first component is large-scale neural networks simulator called Nengo (available at `nengo.ca`). This simulator supports various types of neurons, both of second and third generation (Maass 1996), even in one simulation. Also modularity in networks is well supported. Design of networks in this simulator is based on Neural Engineering Framework (NEF) (Eliasmith and Anderson 2003), but standard design of ANNs can be used too. The Fig.3 shows an example neural-circuit implemented in Nengo simulator.

Robotic Operating System

In order to be able to integrate ANN simulation with more abstract subsystems contained in modules, the Robotic Operating System was selected (available at `ros.org`). ROS is a decentralized software tool to simulate modular systems, mainly for purposes of robotic

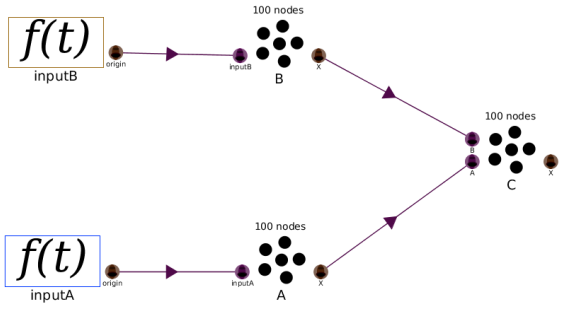

Fig. 3. Example of neural-based system in Nengo. The system contains three neural ensembles, 10, Example of neural-based system in Nengo. The system contains three neural ensembles, two represent input value and the third one computes the sum

research. Modules (nodes) run in own processes and communicate by means of messages over peer-to-peer network. Each module can publish/subscribe to arbitrary type of predefined messages, see Fig.4.

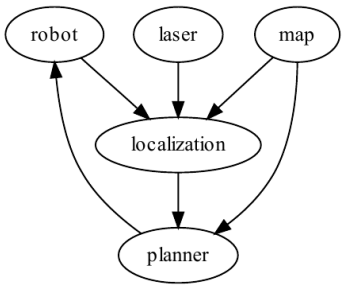

Fig. 4. Example of simple network of ROS nodes

ROS modules communicate by means of predefined messages. These messages are composed of arrays of primitive data types often.

MODIFICATION OF WEIGHT MATRIX ENCODING

For the rudimentary testing of new simulator of hybrid systems, it was decided to show one of the advantages of hybrid networks. It is a smaller number of connections between modules, compared to classical ANN.

Weight Matrix Encoding

The simplest encoding, based on a matrix containing the connection weights, was chosen. The form of matrix is depicted in the Fig.5.

The matrix contains weights between neurons represented by real-values in a given range. This matrix can be encoded in a linear chromosome by concatenating particular lines. Standard EA can be then used to modify this chromosome - to search through the space of possible connections.

Modified Encoding for a Hybrid System

We need to modify this encoding in order to be able to use it for systems with MIMO nodes. While neural

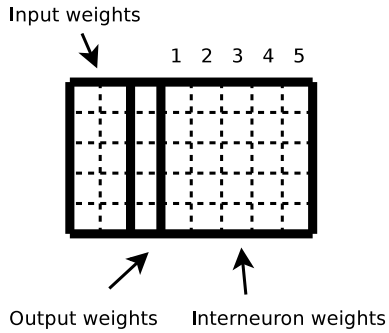

Fig. 5. Example of weight-matrix encoding ANN with two-dimensional input, one-dimensional output and 5 neurons

module have m inputs and n outputs in general. In the simplest case, it is possible to represent each input/output of module as one neuron and use standard encoding described earlier. As can be seen in the Fig.7, it is necessary to hold only upper triangle matrix. Decoding then has three parts:

- Make undirected connections according to matrix
- Change connection directions so they match inputs/outputs
- Discard invalid edges.

In this encoding, and invalid edge connects input with input or output with output, see Fig.7.

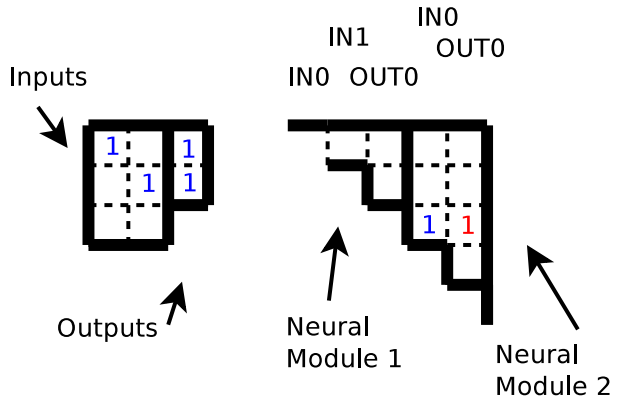

Fig. 6. Modified Weight Matrix Encoding for Network of MIMO nodes

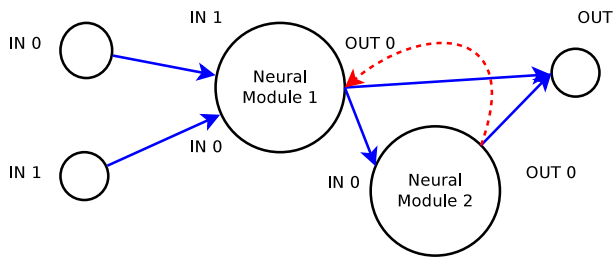

Fig. 7. Topology of a hybrid network encoded by a modified weight-matrix encoding (depicted in the previous Figure)

The obvious benefit is that we do not need so big matrix when connecting MIMO systems. The length of chromosome which encodes the topology with N modules is:

$$l = \frac{\alpha^2 - \alpha}{2} + IN \sum_k in_k + OUT \sum_k out_k, \quad (1)$$

and:

$$\alpha = \sum_k in_k + \sum_k out_k, \quad (2)$$

where k is number of nodes in the network, IN is number of inputs, OUT is number of outputs and $\sum_k OUT_k$ is sum of all outputs for all neural modules.

The drawback is that (in order to get unambiguous encoding) each single neuron has to be represented by its separate input and output in the matrix. This means that in case of encoding only simple ANN (where $m = 1$ and $n = 1$ for all modules), the chromosome will be twice long as in normal, matrix based encoding.

EXPERIMENTS

In order to gain some proof of our concept, a simple experiment was concluded. In this experiment, we compare performance of evolutionary algorithm designing standard (homogenous) ANN with EA designing hybrid modular network. Complexity of both networks (number of connections between nodes) will be similar, but the hybrid network will include a priory knowledge inside one neural module. This experiment should show that complexity of a modular system can be partially encapsulated by neural modules, therefore a design of these hybrid systems is simpler, compared to ANN.

Here, as depicted in the Fig.8, the EA optimizes parameters of a network in order to approximate behavior of a plant. Fitness function of network is computed as inverse of Means Squared Error (MSE) from one simulation. Topology of both networks was set to fully recurrent, so the EA searches just for optimal weights between nodes in the network (inside the NN-based model - Fig.8). Presented results are concluded from 10 runs of EA. In this case, the plant implements simple Fuzzy OR function, which is described by the equation:

$$y = max(\mu(a), \mu(b)). \quad (3)$$

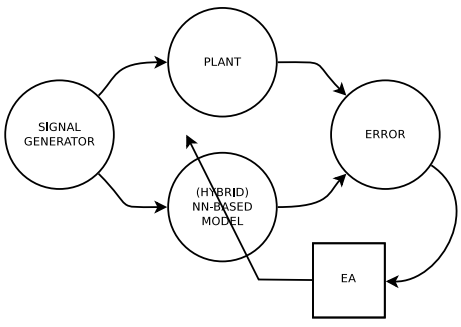

Fig. 8. Scheme of optimization of (hybrid) neural network topology. The goal is to build the model of plant (Fuzzy OR module here)

Evolution of ANN-based Model

The first part is in finding the weights in classical ANN. This model of network of 3rd generation uses Leaky Integrate-and-Fire (LIF) models of neurons (which is often used in emerging new HW platforms (Galluppi et al. 2010)). Neurons of third generation communicate with series of spikes between them. Input signal to the network is converted into spikes and output of the network is converted from spikes to real-valued variable. ANN is composed of $N = 4$ hidden neurons, two input neurons and one output neuron. Since the EA uses matrix-based encoding, this results of vector of real-valued numbers of length $l = 28$.

In the Fig.9 we can see the final setup of the experiment in the NengoRos simulator: signal generator feeds random, two-dimensional signal into the plant and a sub-network. In the sub-network (on the bottom), the particular dimensions are separated, converted into spikes and connected to the population of spiking neurons. Weighted signal from hidden neurons is fed into output neuron, where is converted into real-valued output signal.

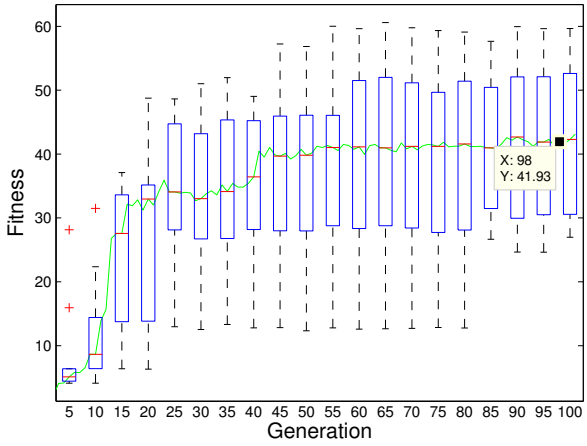

Fig. 10. Fitness of the best artificial neural network during the neuro-evolution

The graph in the Fig.10 shows a course of fitness value of the best network in the population.

Evolution of Hybrid ANN Model

Here, the model is composed according to our framework of hybrid modular networks. These networks can include selected MIMO modules. Obviously it is important which modules we use in order to solve some task. This model should prove that correct choice of modules can speed up the design of architecture and improve its results.

We are introducing neural module which implements "Fuzzy OR", this means that our hybrid network is able to use model of the plant directly. This hybrid network uses our modified weight-based encoding. In order to acquire similar complexity of model parameters, the number of neurons is selected to $N = 2$. According to

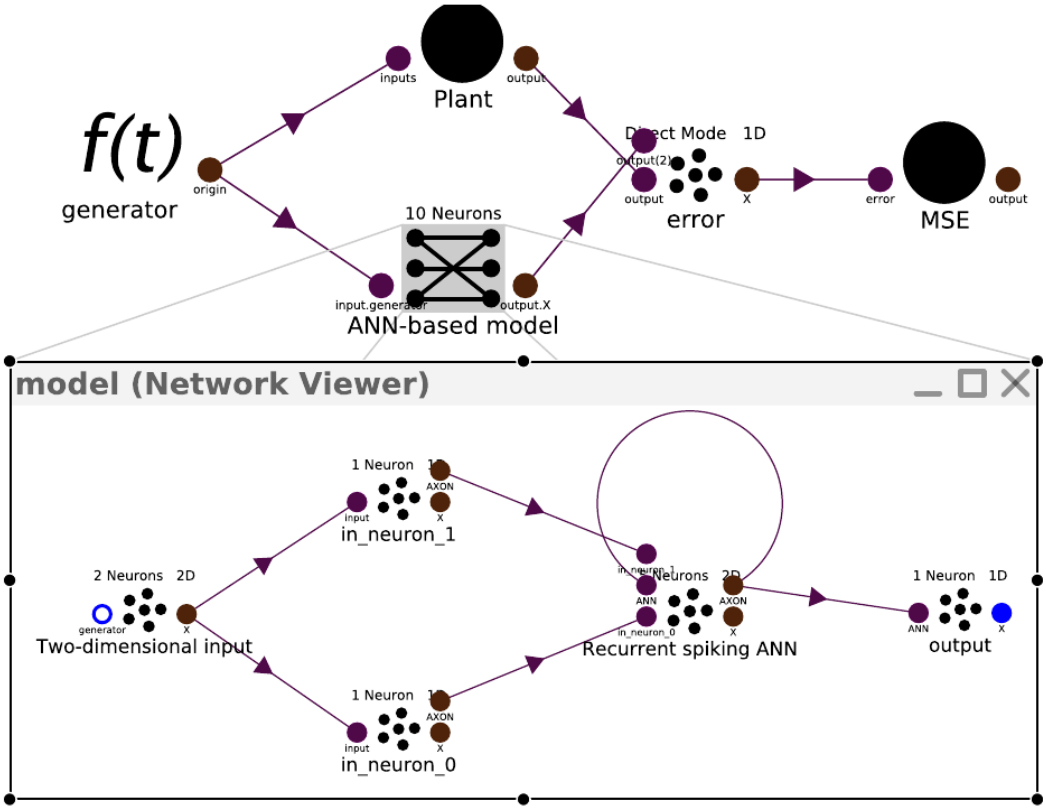

Fig. 9. Example of hybrid modular system - recurrent ANN of third generation tries to approximate plant

the eq.1, the length of genome is slightly higher than in case of classical ANN: $l = 32$. The rest of experiment setup is identical to the previous simulation, our neural module can be also connected in this fully recurrent architecture.

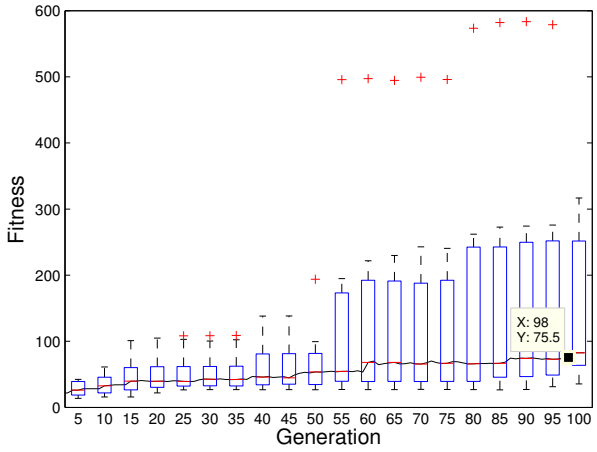

Fig. 11. Fitness of the best hybrid modular network during the neuro-evolution

The graph in the Fig.11 shows that evolution of the hybrid model was able to produce significantly better approximation of plant than classical ANN-based model.

CONCLUSIONS

In this paper we presented our approach to designing agent architectures based on hybrid modular systems. First, we defined requirements for such a system. Then we built simulator of these systems, so we are now able to test new methods of neuro-evolution. These methods will have to be able to compactly represent topologies with general MIMO subsystems. First version of such encoding was tested on small example network. We hope that our approach will enable us to automatically design modular architectures specially for on a given task. Also, such designer could combine current subsystems (neural modules) into new, so far unexplored combinations which exhibit interesting and potentially useful behavior.

The main challenge for the future work stays in finding the suitable representation of network topology. The main benefits of our approach are in combining top-down and bottom-up approaches while designing modular architectures. Now, the user can choose how robust and parallel, and how synoptical and efficient system to design. This can be done by providing specific neural modules.

ACKNOWLEDGEMENT

This research has been funded by the Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague under SGS Project SGS12/146/OHK3/2T/13.

REFERENCES

Auda, G. and Kamel, M. (1999). Modular neural networks: a survey. *Int J Neural Syst*, 9(2):129–151.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2:303–314.

Eliasmith, C. and Anderson, C. H. (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. The MIT press, Cambridge, ISBN: 0-262-05071-4.

Fekiac, J., Zelinka, I., and Burguillo, J. C. (2011). A review of methods for encoding neural network topologies in evolutionary computation. In *Proceedings 25th European Conference on Modelling and Simulation ECMS*, pages 410–416. ISBN: 978-0-9564944-2-9.

Fink, G. A. (2003). *Markov Models for Pattern Recognition*. Springer. ISBN 978-3-540-71766-9.

Fišer, P., Schmidt, J., Vašíček, Z., and Sekanina, L. (2010). On logic synthesis of conventionally hard to synthesize circuits using genetic programming. In *IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 346–351.

Galluppi, F., Jin, X., and Furber, S. (2010). The leaky integrate-and-fire neuron: A platform for synaptic model exploration on the spinnaker chip. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Hornik, K., Stinchcombe, M., and White, H. (1989). Approximation capabilities of multilayer feedforward networks. *Journal Neural Networks*, 2:359–366.

Ivakhnenko, A., G. (1970). Heuristic self-organization in problems of engineering cybernetics. *Automatica*, 6:207–219.

Kadleček, D. and Nahodil, P. (2001). *New Hybrid Architecture in Artificial Life Simulation*, volume 2159 of *Advances in Artificial Life*. Springer Verlag, LNCS edition. ISBN: 978-3-540-42567-0.

Kordík, P. (2006). *Fully automated knowledge extraction using group of adaptive models evolution*. PhD thesis, Czech Technical University in Prague, FEE, Dep. of Comp. Sci. and Computers.

Maass, W. (1996). Networks of spiking neurons: The third generation of neural network models. *In Journal Neural Networks*, 10:1659–1671.

Murre, J. M. J., Phaf, R. H., and Wolters, G. (1989). Calm networks: a modular approach to supervised and unsupervised learning. In *Proc. Int Neural Networks IJCNN. Joint Conf*, pages 649–656.

Nguyen, V., Starzyk, J., Goh, W.-B., and Jachyra, D. (2012). Neural network structure for spatio-temporal long-term memory. In *IEEE Transactions on Neural Networks and Learning Systems*, volume 23, pages 971–983.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Wang, D. and Arbib, M. A. (1990). Complex temporal sequence learning based on short-term memory. 78(9):1536–1543.

Wang, J.-H. and Tsai, M.-C. (1998). Learning recognition of temporal sequences by coding temporal distance in neural networks. In *In Proceedings of IEEE International Joint Conference on Neural Networks*, volume 2, pages 1422–1427.

Wermter, S. and Sun, R. (2000). *Hybrid Neural Systems*. Springer, Heidelberg, New York.

AUTHOR BIOGRAPHIES

PAVEL NAHODIL was born in Prague, Czech Republic. He obtained EE degree (Dipl. Ing) in Technical Cybernetics at the Faculty of Electrical Engineering at the CTU in Prague in 1970 and Ph.D. degree in the same branch in 1980. He has been working at the Department of Cybernetics at the Faculty of Electrical Engineering, Czech Technical University in Prague, where he was also appointed the Professor in Technical Cybernetics. He has led and consulted more than 110 diploma thesis here so far. He was also supervisor of about 30 PhD students till now. His present professional interest includes artificial intelligence, multi-agent systems, on behavior based intelligence robotics (control systems of artificial creatures) and the artificial life design in general. He is (co-)author of several books, university lecture notes, hundreds of scientific papers and large collection of scientific studies. He is also the organizer of some international conferences + reviewer (IPC Member) and a member of many Editorial Boards. His e-mail address is: `nahodil@fel.cvut.cz`.

JAROSLAV VÍTKŮ was born in Prague, Czech Republic. He graduated in 2011 in Czech Technical University in Prague, Faculty of Electrical Engineering in Artificial Intelligence. His diploma thesis "An Artificial Creature Capable of Learning from Experience in Order to Fulfill More Complex Tasks" was awarded by Price of Dean. Currently, he is a PhD student at the Department of Cybernetics, CTU in Prague, still under the guidance of his supervisor Pavel Nahodil. Here elaborates the results of his thesis as an automated design of complex modular systems inspired by Nature. His research interest includes hybrid neural networks, cognitive science, biologically inspired algorithms, behavioral robotics and Artificial Life in common. His e-mail address is: `vitkujar@fel.cvut.cz`.