Search Space Restriction of Neuro-evolution through Constrained Modularization of Neural Networks

Christian W. Rempis[1] and Frank Pasemann[1,2]

[1] Neurocybernetics Group, Institute of Cognitive Science
University of Osnabrück, 49069 Osnabrück, Germany
[2] Institute for Advanced Study, Wallotstr. 19, 14193 Berlin, Germany

Abstract. Evolving recurrent neural networks for behavior control of robots equipped with larger sets of sensors and actuators is difficult due to the large search spaces that come with the larger number of input and output neurons. We propose *constrained modularization* as a novel technique to reduce the search space for such evolutions. Appropriate neural networks are divided manually into logically and spatially related neuro-modules based on domain knowledge of the targeted problem. Then *constraint functions* are applied to these neuro-modules to force the compliance of user defined restrictions and relations. For neuro-modules this will facilitate complex symmetries and other spatial relations, local processing of related sensors and actuators, the reuse of functional neuro-modules, fine control of synaptic connections, and a non-destructive crossover operator. With an implementation of this so called ICONE method several behaviors for non-trivial robots have already been evolved successfully.

1 Introduction

The development of recurrent neural networks for behavior control of autonomous robots with evolutionary methods has a long and successful history [10], [4], [6]. Nevertheless, most experiments work with robots having only a small number of sensors and actuators, as in typical experiments described in [9], [8]. Although interesting non-trivial behaviors have to be expected to come up especially with complex robots having a larger number of sensors and actuators, only few experiments have been conducted in this domain. One main reason is that the search space for neuro-controllers gets inconveniently large if more and more sensor and motor neurons have to be used. This often makes it infeasible to evolve interesting solutions in reasonable time.

To cope with such large search spaces, strategies and heuristics have to be found that reduce the search space or that assist the experimenter to guide evolution towards effective network topologies. In this contribution, we propose that the manual segmentation of neural networks into smaller, *constrained* sub-networks, called *neuro-modules* [11][9], can significantly restrict the search space. This *constrained modularization* is based on domain knowledge about the behavior problem to be solved. The induced restrictions on the modules exclude large parts of the search space and focus the search on network topologies that have a higher chance to provide a desired solution. The kind of solution hereby can be biased by the experimenter to a large extend during the modularization. Furthermore resulting network topologies often are better to understand than unconstrained ones, allow an easier identification of relevant network parts, and make the reuse of already evolved networks structures possible. With this approach the evolutionary algorithm is not used as a universal problem solver that creates complex networks from scratch. Instead evolution is used merely as a tool to help the experimenter to confirm his specific solution approaches, that are usually still too complex to be constructed by hand.

In the next chapter we define the terms *constrained modularization*, *neuron group* and *neuro-module* as they are used here. Then, in chapter 3, we describe how modularization of large neural networks can reduce the search space and why resulting solutions of modular neuro-evolution often are easier to understand. First indications of the usability of this approach, based on the implementation of this method, are discussed in chapter 4 and 5, followed by a conclusion in the final chapter.

2 Constrained Modularization of Neural Networks

2.1 Constrained Modularization

The decomposition of a recurrent neural network into smaller, hierarchically and spatially organized sub-networks is here called *modularization*. A network hereby is, based on domain knowledge and user experience, manually split into connected neuron groups by the experimenter (Fig. 1 shows an example). To each neuron group *functional constraints* can be added, that force the compliance of user defined limitations or structural restrictions. These constraint functions can implement any restriction and manipulate the neural network directly, so that violations of constrains, e.g. originating from mutation operators, can be counteracted immediately.

With this *constrained modularization* the user tries to restrict the network development in such a way, that only a certain, promising type of network structures is possible. Hereby the user constructs a kind of constraint mask for the neural network, that specifically limits the network topology and thus leads to a smaller search space for the evolutionary algorithm.

Neurons can be grouped in two different ways: (1) by simple neuron groups, or (2) by more restrictive neuro-modules. Neuron groups and neuro-modules both allow a detailed topological, hierarchical and functional partition of the network to exclude unwanted areas of the search space. Both types of grouping are defined in the next two sections.

2.2 Neuron Groups

The simplest way to group neurons is the creation of *neuron groups*. These groups may contain any number of neurons sharing topological, functional or other properties. Hereby neuron groups can arbitrarily overlap. Thus each neuron can be part of many neuron groups at the same time. Neuron groups can be target of *constraint functions*. These functions force the compliance of certain, user defined constraints, rules and heuristics for their member neurons. This may include, for example, limiting the number of member neurons or synapses, forcing specific kinds of synaptic connection patterns, allowing synaptic plasticity for its members or resolving dependencies between neurons and synapses. The constraint functions therefore define the purpose of a group and contribute significantly to a search space restriction.

2.3 Neuro-modules

A stronger grouping of neurons is represented by so called *neuro-modules*. Neuro-modules are similar to neuron groups, but do not intersect partially; i.e., neurons can only be part of a single neuro-module at the same time. However, neuro-modules can be members of other neuro-modules and therefore can serve as *sub-modules*.

Neurons in a neuro-module are encapsulated by the module. Thus these neurons are only visible to the neurons of the same neuro-module. To connect the module with external neurons, it can provide a neural interface. This can be achieved by marking selected neurons of the module as *input* or *output* neurons (compare Fig. 1 where these neurons are marked with I and O). During evolution synaptic connections are inserted only between members of the same neuro-module and to interface neurons of sub-modules. Neuro-modules thus can be regarded as encapsulated, independent neural building-blocks with well defined interfaces. Their special purpose is to group strongly related neurons together (e.g. the sensors and motors of a joint or the neurons of a functional structure) and to control the way these neurons can connect to neurons outside of the module.

3 How Constrained Modularization Fosters Successful Evolutions

Modularizing and constraining a neural network according to domain knowledge of a behavioral problem can restrict the search space for neuro-evolution significantly. For comparison, an unconstrained minimal neuro-controller for walking of a humanoid robot with 42 motor and 37 sensor neurons already allows over 3300 synapses, while the same, but constrained modular network in Fig. 1 only allows 180 independent synapses. In this figure it can also be seen, that the modularized network is much more structured than an unconstrained one. It shows, that the constrained network already biases the possible network structures, here to get a symmetric network for walking based on internal oscillators or on an acceleration sensor (at the top module). This also shows that the initial networks for a neuro-evolution have to be specifically modularized regarding the given problem to be solved. Therefore, even for the same problem, different kinds of modularization promote different approaches to solutions. Experimenters can use this to bias the networks towards desired solution approaches.

Constrained modularization reduces the search space by constraining the structure, function and evolution process of the networks, as is described in the next sections.

3.1 Structure Constraints

Neuro-modules, with their ability to hierarchically structure a network and to shield their members from disruptive connections from arbitrary sources, bias the network

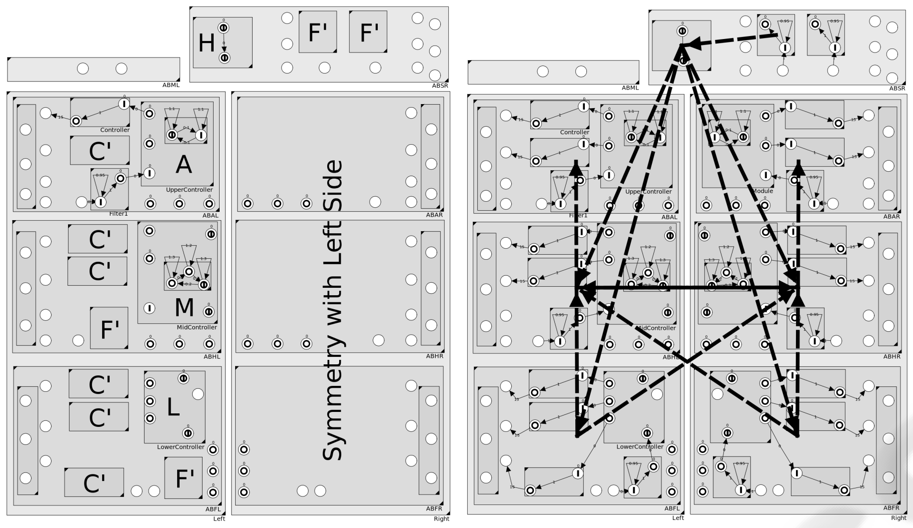

Fig. 1. Left: Constrained modularization of the control network of a humanoid robot. The 37 sensor and 42 motor neurons are separated into modules according to their locations on the robot (head, arms, middle body, legs). A symmetry constraint handles the neural structure of the right side. The upper left module was extended by an evolvable controller module and a filter module. Clones of these modules (*C' and F'*) have been added to each used motor neuron and acceleration sensor. Additional functional modules have been added (*H, A, M, L*) that can be exchanged during crossover by modules of the same type. These modules are expected to implement the actual behavior controller. During evolution neurons are only added to these functional modules. In (A) and (M) oscillator modules have been added that might be modified and incorporated into the control network. **Right:** This network is the result of executing the constraints for the left network. To additionally restrict the search space, synaptic pathways (*Black Dotted Lines*) have been added.

topologies towards local processing units, rather than towards networks with high connectivity. This excludes many – in principle also potentially successful – topologies. But as a heuristic, large, highly connected networks tend to be unable to evolve complex local processing sub-networks, because synapses from arbitrary sources influence most neuron clusters in a disturbing way. This problem increases as the number of neurons in the network gets larger, because the probability for a synapse to be unrelated, and therefore potentially disruptive, increases with every neuron. Therefore we expect highly connected networks to have a lower probability to provide interesting, non-trivial solutions [1]. Therefore neuro-modules can be used to promote plausible connections based on domain knowledge, such as grouping motors and sensors of the same joint together. Neuro-modules also allow the definition of *synaptic pathways*, i.e. to prevent or permit connections between modules explicitly.

A powerful constraint on the structure of neuron groups is the definition of *symmetries* and *repetitive structures*. Depending, of course, on the targeted behavior problem, many evolutions can be greatly restricted when the desired network is assumed to be symmetric. Examples are walking or squatting of a humanoid robot or the repetitive structure of a multi-legged walking machine. Symmetries remove large parts of the search space, because the parameters of all symmetrized neurons and synapses are not part of the search space any more (e.g. the entire right side in Fig. 1).

An additional positive effect on the structure using modularization is the better readability of the resulting networks (see Fig. 1). Functional elements can be isolated more easily and signal paths can be better traced, because most synapses are locally connected and have less dependencies to other parts of the network.

3.2 Functional Constraints

Neuro-modules bias evolution to evolve local processing units, that are often related to local functions. Although, admittedly, it can not be guaranteed that the evolved structure of a module implements a single, well defined function, the tendency still is towards functions distributed over only a few, local modules. This still simplifies the isolation of such functions when an evolved network is analyzed.

Neuro-modules can also be used to represent predefined functional units, that may origin from previous evolutions or analytic reasoning. Once a functional processing unit is found by evolution, it can be reused in future evolutions as neural building block. Forcing evolution to reinvent already known processing units in each evolution from scratch only blows up the search space without any gain from the scientific perspective. With an additional *neuro-module insertion* mutation operator that can insert predefined neural building blocks from a library, larger, functionally more complex networks can evolve in shorter time.

Neural building-blocks can also be constrained with very *specific* constraint functions. Because building blocks can be constructed by hand – although often based on evolved structures – specialized constraint functions can be added. Such functions can be used to ensure, for instance, that the function or complex structure of a module is preserved independently of the mutations taking place. They can also be used to design complex modules, such as neural fields [3], memory units [15], oscillators [12], structures with adaptive synapses and the like.

Constraints can also be used to *clone* a mutable neuro-module and to reuse the same network structure in multiple places of the network. In this way the function of this module can still evolve, while it is used with all modifications in several places, profiting from enhancements immediately. A common usage of this is the definition of sensor filters or motor controllers (as in Fig. 1), where the same structure is required for any sensor or motor of the same type. If the sensor or motor works similar in all places, then the controller has not to be optimized multiple times.

3.3 Evolutionary Constraints

Neuro-modules are a suitable target for modification operators during evolution. Because neuro-modules are well structured, providing a specific interface to their surrounding network parts, they can be exchanged and replaced with only little impact on the rest of the network. This enables the usage of *modular crossover*. Crossover in most neural network implementations is highly destructive due to the potentially large structural differences between parents. Crossover between such unrelated networks most probably produces networks that are less fit than both of their parents, so that most of these networks usually do not survive. Modular crossover is less affected by this problem, because crossover takes place only at well defined network parts, namely at the module level. Modules are only replaced by compatible modules, which means that their interfaces match and the module types are similar.

In addition to module exchange between parents, modules may also be exchanged by compatible modules from a library of predefined building blocks or by neuro-modules co-evolving with the behavior controllers in their own populations.

A particular benefit of constrained modularization for evolution is that the approach to solve a given behavior problem can be biased to a large extend in advance. This way the experimenter does not only specify the problem to be solved, but also influences to a high degree, how the problem is going to be solved. Also, the iteration of evolutions becomes much easier: The behavior problem may be solved first by applying sharp restrictions on the evolving networks. Then, iteratively, the network can be opened for new solution approaches to stepwise enhance the behavior.

4 Application

The modularization approach with the described features has been implemented in the ICONE (Interactively Constrained Neuro-Evolution) method. Currently the implementation supports structure evolution based on neuron, synapse and neuro-module insertions. Explicit specifications of neural pathways between neuro-modules are considered, as well as connection restrictions induced by the hierarchical interfaces of neuro-modules. Neuron groups and neuro-modules can be restricted with arbitrary, user defined constraint functions, such as symmetry, cloning and restrictions of neuron and synapse structures. A library of neuro-modules as basic building blocks is under continuous construction, including neuro-modules for different kinds of oscillations, memory, joint controllers, sensor filters, event detections, context switches and behavior interpolation. During evolution all aspects of the evolutionary algorithm can be modified on-line to guide evolution through the search space.

Manually modularizing large networks is not trivial. Therefore a graphical neural network editor was implemented that supports the visual manipulation of all mentioned aspects of the neural networks. Without such an editor, modularization is difficult to apply.

5 Examples

The modularization technique has been applied to develop neuro-controllers for several complex robots. These robots include for instance the six legged walking machine *Octavio* with 24 sensor and 18 motor neurons, and the *A-Series* humanoid robot with 37 sensor and 42 motor neurons. The developed behaviors include – among others – different kinds of walking, squatting and stabilized standing.

Some examples are shown in Fig. 2 and Fig. 3. Due to space limitations, details on the evolved behaviors will be presented in upcoming publications. But it can be said, that with pure structure evolution solutions for these kind of problems could not be found at all.

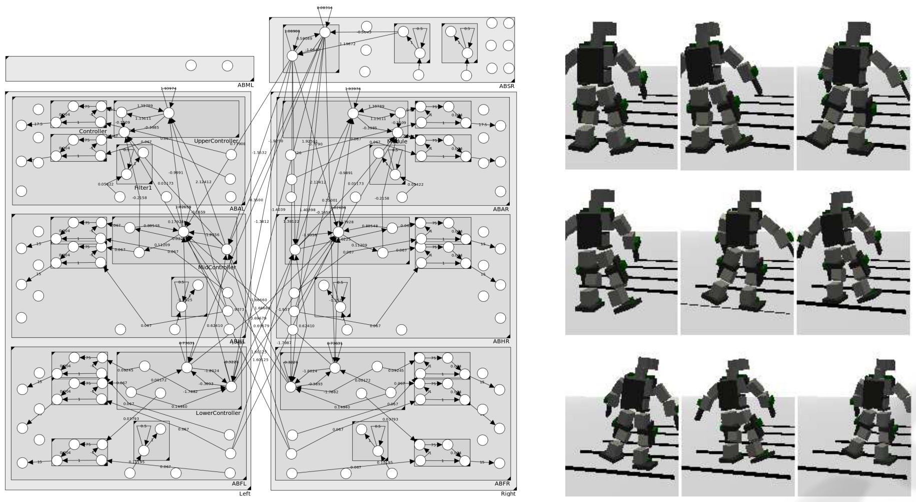

Fig. 2. Network and time-series of an evolved neuro-controller for walking with the *A-Series* humanoid, based on a constrained initial network similar to Fig. 1. The initial network was constrained to search for solutions based on the acceleration sensors of the shoulder (*Mid of Upper Right Module*). Substantially different networks can be produced starting with internal pattern generators as in Fig. 1. [Evolution: ≈ 400 generations with 150 individuals].

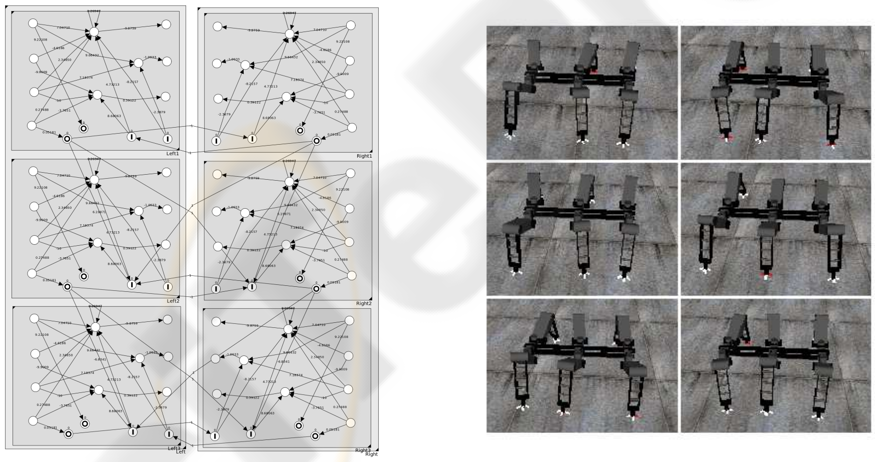

Fig. 3. Network and time-series of an evolved neuro-controller for walking with the 6-legged walking machine *Octavio*. Here only one leg controller (*Upper Left*) is evolved, all other legs clone this prototype controller. In addition the network has a right / left symmetry. The focus of this experiment is to find a universal leg controller for a multi-legged walking machine and appropriate interconnection patterns. With minor adjustments of the constraints the focus of the experiment can be changed, e.g. to find specialized leg controllers for front, mid and hind legs. [Evolution: ≈ 300 generations with 100 individuals].

6 Discussion

Modularizing a network by applying domain knowledge and user experience obviously restricts the search space for neuro-evolution algorithms. Attempts to restrict the search space have been conducted by many authors, because without restrictions the evolution of large network topologies of non-trivial robots becomes infeasible.

A common approach is the use of specific genome representations, that imply for instance a fully symmetric network [7], [13]. This approach can be compared to modularization with symmetry constraints. But because the symmetry is embedded directly in the genome representation or in the genotype-phenotype mapping, a new genome type has to be created for each new experimental scenario. Also complex, stacked symmetries are difficult to set up. During evolution such genomes are rigid and can not be extended if needed. Using, as proposed here, constraint functions to influence the relations of network parts, symmetry can be implemented as a simple extension of the network genome and can easily be removed or changed without changing the genome type. Furthermore constraints are not restricted to symmetry, but can enforce any kind of structural dependency, like complex spatial connections as used in neural fields [3].

Another approach to reduce the search space is to focus only on specific parts of the target robot. For instance, walking may be evolved with only the legs of a humanoid robot, replacing the entire upper body by a simple block of comparable mass [13]. This, indeed, reduces the search space, because all motors and sensors of the simplified body parts have been removed. Though, extending such a controller to the full robot becomes difficult, because the evolved controllers will ignore the influence of the other moving body parts. Also, for each new approach, a new simulated robot has to be designed, that focuses on the desired motor and sensor aspects, and therefore also iterative evolution in small steps becomes more difficult. With the proposed modularization technique, the complex target robot can be used right from the beginning. Unwanted sensors and motors can be excluded in the beginning by synaptic pathway restrictions and can be re-enabled at any time during the evolution. Therefore the evolution can start with a minimal subset of the robot's actuators and sensors, but can still include the non-essential robot parts to further optimize the controllers.

A third popular search space restriction approach is structure reuse. In most cases structure reuse is implemented within developmental evolution systems, like Cellular Encoding [5], where the genotype is mapped to a phenotype by applying a sequence of construction rules. These algorithms have been shown to reuse structures in multiple places while evolving the *structure blue-prints* only once. A disadvantage of such developmental approaches is, that the resulting modular structures are difficult to isolate for later usage. Furthermore, it is difficult to translate a complex starting network with this kind of modularity into its genotype representation, and to monitor and manipulate these modular structures during evolution. Neuro-modules as building blocks on the other hand do not require a complex mapping from genotype to phenotype and thus can be reused as entire structure with little effort. Other techniques [2], such as Modular NEAT [14], try to automatically define neuro-modules as building-blocks without a complex genotype-phenotype mapping. But also here, functional modules have to be reinvented in every evolution, because predefined modules can not be used. Furthermore, the reused sub-networks are arbitrarily aligned to the input and output neurons without domain knowledge, so that – especially in large networks – proper use of the modules becomes unlikely again.

Constrained modularization allows the utilization of all mentioned search space restriction methods in a uniform, extensible framework. With appropriate libraries of functional neuro-modules new types of larger control networks can be developed, that might give deeper and even new insights into neural organization of behavior. Constrained modularization as a general principle does not really restrict experimenters in their approaches, because new approaches can be simply implemented by introducing new constraint functions.

7 Conclusions

In this contribution it was discussed how *constrained modularization* of large neural networks for robot control can significantly reduce the search space for neuro-evolution processes. Large neural networks can be spatially and logically partitioned by *neuron groups* and *neuro-modules*. Both types of grouping can be the target of *constraint functions*, that force the compliance of – partly very specific – constraints, such as network symmetries, dependencies, module cloning and connectivity structures between or within modules. The modularization is done manually to apply domain knowledge and to bias the search towards desired solution approaches. In this way the search space is restricted by the user to a well defined potential solution space, which increases the chance to find appropriate solutions. For modular neural networks new types of evolution operators are defined: *modular crossover* and *neuro-module insertions*. Modular crossover allows the exchange of sub-networks in a minimally destructive way. Insertions of functional neuro-modules as mutation allow the extension of a network with already working functional sub-networks, which eases the transfer of findings from previous evolutions and relieves evolution from reinventing already known structures. The described approach has been used to develop different behaviors for several robots with many sensors and actuators, including a multi-legged walking machine and humanoid robots. Detailed results are described in upcoming publications.

Acknowledgements

The authors thank Arndt von Twickel and Manfred Hild for many inspiring discussions. Thanks also to Verena Thomas and Ferry Bachmann for their contributions to the simulation environment. This work was partly funded by EU-Project Number ICT 214856 (ALEAR Artificial Language Evolution on Autonomous Robots. http://www.alear.eu).

References

1. Beer, R.D. (2010). Fitness Space Structure of a Neuromechanical System. To appear in Adaptive Behaviour.
2. Calabretta, R., Nolfi, S., Parisi, D., and Wagner, G. P. (2000). Duplication of Modules Facilitates the Evolution of Functional Specialization. Artificial Life, 6(1), pp. 69–84.
3. Coombes, S. (2005). Waves, Bumps, and Patterns in Neural Field Theories. Biological Cybernetics, 93(2), pp. 91–1008.
4. Floreano, D., Husbands, P., and Nolfi, S. (2008). Evolutionary Robotics. In Siciliano, B. and Khatib, O., editors, Springer Handbook of Robotics, pp. 1423–1451. Springer.
5. Gruau, F. (1995). Automatic Definition of Modular Neural Networks. Adaptive Behaviour, 3(2), pp. 151–183.
6. Harvey, I., Paolo, E., Wood, R., Quinn, M., and Tuci, E. (2005). Evolutionary Robotics: A New Scientific Tool for Studying Cognition. Artificial Life, 11(1-2), pp. 79–98.
7. Hein, D., Hild, M., and Berger, R. (2007). Evolution of Biped Walking Using Neural Oscillators and Physical Simulation. Proceedings of the RoboCup 2007: Robot Soccer World Cup XI.
8. Hülse, M., Wischmann, S., Manoonpong, P., von Twickel, A., and Pasemann, F. (2007). Dynamical Systems in the Sensorimotor Loop: On the Interrelation Between Internal and External Mechanisms of Evolved Robot Behavior. In Lungarella, M., Iida, F., Bongard, J. C., and Pfeifer, R., editors, 50 Years of Artificial Intelligence, volume 4850 of Lecture Notes in Computer Science, pp. 186–195. Springer.
9. Hülse, M., Wischmann, S., and Pasemann, F. (2004). Structure and function of evolved neuro-controllers for autonomous robots. Connection Science, 16(4), pp. 249–266.
10. Nolfi, S. and Floreano, D. (2004). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, ISBN 978-0262640565, Bradford Book.
11. Pasemann, F. (1995). Neuromodules: A Dynamical Systems Approach to Brain Modelling. In Herrmann, H. J., Wolf, D. E., and Poppel, E., editors, *Workshop on Supercomputing in Brain Research: from tomography to neural networks, KFA Julich, Germany*, World Scientific Publishing Co.
12. Pasemann, F., Hild, M., and Zahedi, K. (2003). SO(2)-Networks as Neural Oscillators. Computational Methods in Neural Modeling, 2686/2003, pp. 144–151.
13. Reil, T. and Husbands, P. (2002). Evolution of Central Pattern Generators for Bipedal Walking in a Real-Time Physics Environment. *IEEE Transactions on Evolutionary Computation*, Vol.6(2), pp. 159–168.
14. Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2004). Evolving Reusable Neural Modules. In Deb, K., et al., editors, Genetic and Evolutionary Computation – GECCO-2004, Part II, volume 3103 of *Lecture Notes in Computer Science*, pp 69–81, Springer.
15. Rempis, C. W. (2007). Short-Term Memory Structures in Additive Recurrent Neural Networks. Master's thesis, University of Applied Sciences Bonn-Rhein-Sieg, Germany.