ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Evolutionary Computation based System Decomposition with Neural Networks

Robert Kaltenhäuser[1], Erik Schaffernicht[2], Frank-Florian Steege[1,3] and Horst-Michael Gross[1] *

1- Neuroinformatics and Cognitive Robotics Lab - Ilmenau University of Technology - Helmholtzplatz 5, 98693 Ilmenau - Germany
2- Center of Applied Autonomous Sensor Systems - Örebro University - Sweden
3- STEAG Powitec GmbH - 45219 Essen-Kettwig - Germany

Abstract. We present an evolutionary approach to divide a complex control system into smaller sub-systems with the help of neural networks. Thereto, measured channels are partitioned into several disjunct sets, representing possible sub-problems, while the networks are used to assess the quality of the resulting decomposition. We show that this approach is well suited to calculate correct decompositions of complex control systems. Furthermore, the obtained neural networks are used to predict important process factors with considerable better approximation quality than monolithic approaches that have to deal with all input channels in parallel.

1 Introduction

The identification of sub-systems in complex control processes is a very important part of design and optimization of control systems. Industrial plants, like bioreactors or waste incineration plants often have many measuring devices which provide information about process values like temperature, oxygen level, and many more. On the other hand, there are many actuators which can influence the process proceeding at the respective plant. If the task is to design a control system which optimizes the process, it is crucial to know which actuator influences which process parameter. The goal of our approach is to automatically decompose such a complex control system into smaller, possibly decoupled sub-systems, which are more manageable and, therefore, can be better optimized (see figure 1). To this end, a strategy has been developed, which makes use of concepts and methods from the field of evolutionary computing. We show that decomposed neural networks (NNs) produce better approximations in comparison to monolithic approaches.

In the following, in section 2 we first give an overview of existing approaches for system decomposition and present our evolutionary algorithm in section 3. Accordingly the testing environment, a simulation of a complex control process, is explained in section 4.1 while the results of the decomposition are shown in section 4.2. Finally, we draw conclusions in section 5.

*This work has received funding from the German Federal Ministry of Education and Research as part of the APD project under grant agreement no. 01LY1011B

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

2 Related Work

Different approaches for reducing the complexity of tasks have been explored in the literature. Efforts were made towards spatial and functional problem decomposition. For the former one, several NNs are trained to approximate a single target channel $y(t)$ with different input spaces. Depending on the current input situation, the corresponding neural network is chosen for controlling the task. This can be done e.g. with the *Mixture of Experts* approach combined with an evolutionary algorithm to evolve NNs for the respective input spaces [1]. In functional problem decomposition, the target channel $y(t)$ is composed of its functional components, e.g. $y(t) = g\left(f_1(x), ..., f_{N_P}(x)\right)$. Here $f_i(\cdot)$ are the functions of the partial channels which are merged by the combining function $g(\cdot)$. In [2, 3] the authors use modular NNs and, with the help of cooperative co-evolution, evolve the structure of the modules such that they approximate the partial channels.

The main drawback of both methods is that they are only able to decompose a single channel $y(t)$ into basic sub-channels, and, therefore are not suitable to decompose complex processes consisting of many target channels and inputs.

Another way to handle a system's complexity is feature selection [4]. The approach is to prune irrelevant inputs and use only those which actually influence the target. Approaches for input pruning include statistical methods [5], wrapper methods [6], embedded methods [7], or combinations of the aforementioned ones [8]. In theory, input pruning can be used for system decomposition as well. A network N_i, which approximates channel x_i using all other channels $(x_1, \ldots, x_j, \ldots, x_n)$ with $j \neq i$ as inputs is pruned. The remaining input channels capture the dependencies to the channel x_i. If training and pruning is repeated for all channels, the resulting dependency-graph can be analyzed for independent components, which represent sub-systems. Although satisfying in theory, our tests show that input pruning often overestimates dependencies which leads to closely meshed dependency graphs and makes it impossible to decompose a system into separated sub-system as it is the aim of this paper.

In the following section, we present a new approach to combine methods from evolutionary computing with the idea of input pruning to circumvent this problem. As a result, we obtain NNs which represent the independent subsystems of the overall system.

3 Evolution for Automatic System Decomposition

The goal of our approach is to decompose a complex system consisting of n channels $x_1, \ldots, x_n$ into m sub-systems $S_1, \ldots, S_m$ $(S_i \cap S_j = \emptyset)$ in a sense of splitting different measurement channels into disjoint partitions, so that no functional dependencies between channels of different partitions exist (see figure 1).

To get disjoint partitions, an evolutionary optimization is carried out. An individual in the evolution is represented by a decomposition of the system

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

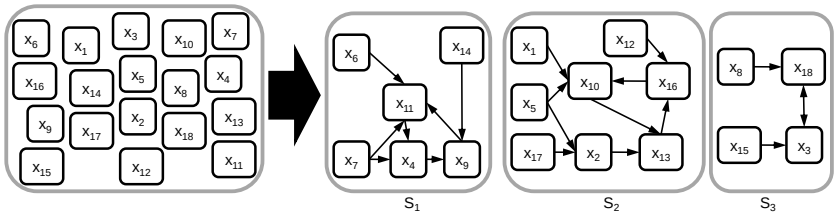

Fig. 1: Decomposition of a complex system with measurement channels x_i and unknown relations into three sub-systems/partitions $S_1, \ldots, S_3$ with known relations (gray boxes). A relation from e.g. x_5 to x_2, such that $x_2 = f(x_5)$ is represented as an arrow from x_5 to x_2.

consisting of an assignment of channels to partitions. At the start of the optimization, several initial decompositions are created by assigning them to a randomly chosen partition. After this, they are assessed for their usability as a system decomposition. For each partition a neural network is trained, trying to approximate the current values x_j^t of each channel $x_j \in S_j$ in the partition at time t and taking past values $x_j^{t-1}, \ldots, x_j^{t-l}$ of the channels as inputs. The resulting approximation error

$$E = \sum_{i=1}^{m} \sum_{j=1}^{o} \sum_{k=1}^{p} \left(y_{ij}(x_{ik}) - t_{ijk} \right)^2$$

is used as a fitness function for the evolutionary algorithm. Where p is the number of training samples, m the number of partitions and o the number of output-neurons for the corresponding partition. Moreover x_{ik} denotes the k-th input-vector of the i-th partition, y_{ij} the output of the j-th neuron in the i-th partition and t_{ijk} is the k-th target value for the corresponding neuron.

The basic idea of our approach can be described as follows: The more channels of an individual are assigned to an incorrect partition, the more of the dependencies are crossing the borders of the partitions. Since the NNs are only trained with values of channels from the same partition, individuals with a false partitioning receive a high approximation error and a poor fitness score, since required information for the approximation of the channels is missing. Only a division of the channels in partitions that correspond with the real sub-systems receives the best fitness and the minimal approximation error, since all the information to approximate and predict the current channel values is available.

Therefore, individuals with a high fitness and accordingly a good decomposition should be reproduced using mutation and recombination. The fitness of this newly created individuals can now again be evaluated by the approximation error of NNs trained on them. This is repeated until a stopping criterion is met. That way, the individuals coding particular system decompositions are evolved until they ideally reach the optimal decomposition.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Note that the optimal number of sub-systems can be automatically determined by the algorithm by permanently deleting empty partitions which arise during mutation and recombination. Since the number of partitions constantly decreases, we recommend to start the evolution with more partitions than truly exist in the system. We used about twice the (suspected) size of the number of real sub-systems.

Algorithm 1 Evolutionary System Decomposition with Neural Networks

Input: channels $X = \{x_1, \ldots, x_n\}$, size of population μ, number descendants λ
  for $l = 1 \rightarrow \mu$ **do**
    $I^{(l)} = \{S_1^{(l)}, \ldots, S_m^{(l)}\}$ {create individual by random channel partitioning}
    train neural network for every partition $S^{(l)} \in I^{(l)}$
    $F^{(l)} = E^{(l)} = \sum_{i=1}^{m} \sum_{j=1}^{o} \sum_{k=1}^{p} \left(y_{ij}(x_{ik}) - t_{ijk}\right)^2$ {calculate fitness}
  end for
  repeat
    for $l = (\mu + 1) \rightarrow (\mu + \lambda)$ **do**
      select individual $d_q \in I$ based on fitness F {e.g. ranking selection}
      $I^{(l)}$ = reproduce d_q {mutation & recombination}
      train neural network for every partition $S^{(l)} \in I^{(l)}$
      $F^{(l)} = E^{(l)} = \sum_{i=1}^{m} \sum_{j=1}^{o} \sum_{k=1}^{p} \left(y_{ij}(x_{ik}) - t_{ijk}\right)^2$ {calculate fitness}
    end for
    delete λ individuals with the worst fitness $min_{\lambda}(F(I))$
  until stopping criterion is met
  return decomposed system $\{S_i | i = 1, \ldots, m\}$ with $S_i \cap S_j = \emptyset\ , S_i, S_j \subset X$

4 Experiments

4.1 Implementation Details

We implemented the described methods using a (20+20) - evolutionary algorithm with Lamarckism and a linear ranking selection for determining the parents of the next generation. Moreover, we used uniform crossover with a crossover rate of $p_C = 0.5$ while the partitioning was mutated by randomly assigning a small number of channels to other partitions. Thereby, we adapted the mutation rate by a local variance adaption strategy. Afterwards, the evolution was executed for a total of 500 epochs. See [9] for more details about the selected evolutionary techniques.

For the training of the NNs the Levenberg-Marquardt algorithm with Bayesian Regularization and cross-validation was used, and after training the networks were pruned with the Optimal Brain Surgeon algorithm [10].

4.2 Results

The suggested method was tested on three simulated scenarios which are distinguished by different problem sizes, several levels of noise and by their complexity.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

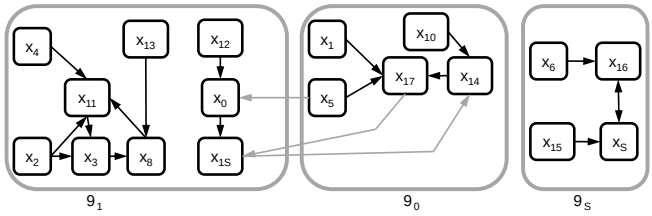

Fig. 2: Example for an improper decomposition of scenario 3. Dependencies which cross the border of the sub-systems are marked by gray arrows.

Scenario 1 was the easiest one with five channels, two sub-systems and a low level of noise, while scenario three, which is shown in figure 1, was that with the highest complexity with eighteen channels, three sub-systems, and a high noise-level. The test of our algorithm was performed ten times for every scenario, and the results were averaged.

For comparison purposes we calculated the approximation error for all tested scenarios for single monolithic NNs trained on all target channels at once and the approximation error of manually decomposed NNs, where the channels were partitioned with the help of background knowledge. The same training scheme, including Bayesian Regularization and Optimal Brain Surgeon, was used for all networks. The resulting error values were averaged over 100 trials.

As a result for scenario 1 and 2 our algorithm derived system decompositions which correspond perfectly to the real sub-systems. Even for the most complex scenario tested here, scenario 3, 80% of the decompositions had been found out correctly. Only 20% of the trials led to a suboptimal system decomposition. One of them is shown in figure 2. Here, while sub-systems S_1 and S_3 had been assigned to the right partitions, sub-system S_2 had been split into two partitions which indicates, that the evolutionary algorithm got stuck in a local optimum.

	Scenario 1		Scenario 2		Scenario 3	
$\bar{E}_{mon}$	$4.152 \cdot 10^{-5}$		0.3888		3.5178	
$\bar{E}_{mdec}$	$1.392 \cdot 10^{-5}$		0.3345		3.3771	
$\bar{E}_{adec\_cor}$	$6.345 \cdot 10^{-9}$	100%	0.1913	100%	3.1551	80%
$\bar{E}_{adec\_inc}$		0%		0%	4.1223	20%

Table 1: Test results on all data sets. Given is the mean approximation error for monolithic NNs $\bar{E}_{mon}$, for manually decomposed NNs $\bar{E}_{mdec}$ and for automatically decomposed NNs which resulted in a correct $\bar{E}_{adec\_cor}$ or incorrect decomposition $\bar{E}_{adec\_inc}$, with their respective proportion.

Additionally, the obtained error rates are shown in table 1. As expected, due to the decreased number of weights and the reduced complexity, the error rates of decomposed NNs are significantly lower than that of a monolithic approach. Moreover, our algorithm found NNs with approximation errors, which were even

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

lower than the mean errors of the manually decomposed NNs. This can be explained by the random factor in the training of NNs, e.g. as a consequence of random weight initialization. While bad training results effected the averaged results of our tests, they have no impact on the evolutionary algorithm, since they receive a bad fitness and are rejected accordingly.

5 Contribution of this Paper

In this paper, we introduced a new algorithm to automatically decompose a complex control problem into smaller, better manageable sub-problems.

The algorithm aims at decomposing a complex system comprised of several channels into independent sets, such that no dependencies among channels of different sets exist. This was achieved by an evolutionary algorithm which partitioned the channels into sets. The fitness of the resulting sub-systems can be evaluated by approximating the channels of each partition by NNs. The partitioning may then be evolved until the optimal decomposition is found.

We showed that our algorithm is able to correctly decompose even complex systems into independent components and can reliably estimate the current number of sub-systems. Moreover, we showed that such a system decomposition is able to create NNs with a considerable lower approximation error than monolithic NNs. The decomposed NNs can be used e.g. for model predictive control and enable human experts to handle sub-systems with much less complexity than the whole system.

References

[1] M. H. Nguyen, H. A. Abbass, and R. I. Mckay. A novel mixture of experts model based on cooperative coevolution. *Neurocomputing*, 70:155–163, 2006.

[2] V. Khare, X. Yao, B. Sendhoff, Y. Jin, and H. Wersing. Co-evolutionary modular neural networks for automatic problem decomposition. In *Congress on Evolutionary Computation (CEC)*, pages 2691–2698. IEEE Press, 2005.

[3] M. A. Potter and K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[4] E. Schaffernicht, V. Stephan, K. Debes, and H.M. Gross. Machine learning techniques for selforganizing combustion control. *KI 2009*, pages 395–402, 2009.

[5] K. Torkkola. *Feature Extraction: Foundations and Applications*, volume 207 of *Studies in fuzziness and soft computing*, pages 167–186. Springer Verlag, 2006.

[6] P. Somol, P. Pudil, J. Novovičová, and P. Paclík. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20:1157 – 1163, 1999.

[7] L. Y. Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

[8] E. Schaffernicht, C. Moeller, K. Debes, and H.-M. Gross. Forward feature selection using residual mutual information. In *ESANN 2009*, volume 1, pages 583–588, 2009.

[9] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford Univ. Press, 1996.

[10] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, 1:293 – 299, 1993.