Modeling a Neural Network Based Control for Autonomous Production Systems

Daniel Rippel [1] and **Florian Harjes** [1] and **Bernd Scholz-Reiter** [1]

Abstract. The increasing complexity of production logistic systems has lead to an emergence of new decentralized control concepts. The Collaborative Research Center 637 (CRC 637) investigates the advantages and limitations of autonomous control as one of these concepts. This research mainly focuses on control strategies consisting of precise descriptions of decision-making processes. In addition, it is developing a modeling framework for autonomously controlled logistic systems called ALEM.

As the derivation of precise behavioral descriptions from real world scenarios is difficult, artificial neural networks has come into focus. They provide a suitable method to imitate a system's behavior without a formal definition. The architecture for adaptive relocation control (ARC) uses neural networks for control purposes on shop floors. This article proposes a concept of modeling to include ARC's decentralised decision strategy into the ALEM modeling framework.

1 INTRODUCTION

Autonomous control aims to increase the flexibility and performance of logistic systems by allocating planning and control abilities to individual system elements. Therefore, different methods for decision making are of central interest. Most of these methods base on precise mathematical or procedural descriptions of the decision process. To model and simulate autonomously controlled logistic systems, the Autonomous Logistic Engineering Methodology (ALEM) has been developed. This methodology provides tools to model and simulate autonomous logistic systems for logistic experts. ALEM relies on a strict definition of the system elements' decision processes, represented by UML-Activity diagrams and UML-State Machines. In particular, creating models of already existing systems leads to difficulties with the derivation of diagrams which formally describe the system‘s decision strategies.

To cope with this difficulty, artificial neural networks provide a suitable solution to recreate the behavior of an existing logistic system, the decision processes of which cannot be formally described [16]. The networks can be trained to imitate the original system's behavior. Further, neural networks have been applied successfully to several production logistic scenarios, for example to predict delivery dates or manufacturing costs, for quality control or for sequence optimization [3, 6, 16, 21].

Although artificial neural networks provide a convenient technique for decentralized decision making [17], ALEM is incapable to include neural networks into its models at the moment. Therefore, this article investigates the requirements for the integration with ALEM. In addition, the article depicts the advantages of this Integration both in context of autonomous control and with regard to the application of neural networks to production logistic scenarios.

The analyses of the requirements on ALEM refers to an example shop floor scenario. The scenario applies a decentralized control strategy using artificial neural networks for decision making. ALEM is used to model the scenario; thereby identifying missing notational concepts. To close this gap, the article proposes concepts and notational elements to represent the neural networks within ALEM models.

The first section introduces autonomous control and neural networks in detail. The second section depicts the shop floor example and explains the neural control strategy applied in the scenario. The third section depicts the most important aspects of the corresponding ALEM model as well as concepts to represent neural networks within the model. The article closes with sketching the advantages of integrating neural networks into ALEM.

1.1 AUTONOMOUS CONTROL

The term autonomous control is used in various scientific disciplines like physics, biology, artificial intelligence, control theory, and engineering sciences. In the context of logistic systems, Hülsmann and Windt define autonomous control as the "processes of decentralized decision-making in heterarchical structures. It presumes interacting elements in non-deterministic systems, which possess the capability and possibility to render decisions independently. The objective of Autonomous Control is the achievement of increased robustness and positive emergence of the total system due to distributed and flexible coping with dynamics and complexity" [9].

The application of autonomous control delegates the planning capabilities to work pieces or workstations, and disconnects commodities from the customer's orders. The objects dynamically create their own production plans. For example, once an intelligent logistic object enters production, it autonomously requests manufacturing from suitable workstations on the shop floor. Several relocation strategies have been investigated. Examples are the "Queue Length Estimator" an ant colony algorithm or the Dynamic Logistic Routing Protocol. In addition, the objects apply individual objectives. In case of a malfunction or a changing production situation, the work pieces are capable of reacting flexible. Once they are aware of a disturbance, they request manufacturing from another workstation of the same type. With regard to their product structure, they can shift the order of manufacturing steps. This allows postponing the problematic production step and can help resolving bottle neck situations [19]. Due to the dynamic allocation of work pieces to orders, they may shift their target end product. Thus, they can flexibly react to changes in the production situation, if for example new orders arrive or active

[1] BIBA - Bremer Institut für Produktion und Logistik GmbH at the University of Bremen, Germany, email: {rip, haj, bsr}@biba.uni-bremen.de

ones are canceled.

The ALEM framework is developed to model and simulate autonomously controlled systems. The methodology is designed to enable logistic experts to create models. The methodology consists of three components. First, a procedure model (ALEM-P) which guides the user through the process of modeling. Second, the ALEM-Notation (ALEM-N) which provides model elements and diagrams to represent the system, and third, a software tool (ALEM-T), which implements the procedure as well as the notation. The notation uses diagrams and elements of the Unified Modeling Language (UML) and adds several domain specific diagrams [18].

As process and system models usually imply a high degree of complexity [13], ALEM applies a view concept (Figure 1) proposed by Scholz-Reiter et al. [18]. The views focus on single aspects of the overall model and enables editing of lesser complex segments of the model [14].

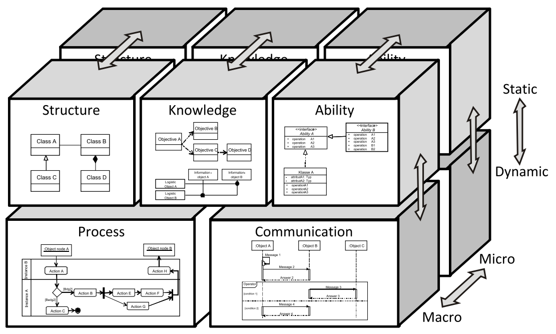

Figure 1. ALEM View Concept [18]

The views use default UML diagrams to capture different aspects of the overall model. For example, the structure view uses a UML-Class Diagram to depict the elements present in the logistic system. Those elements‘ abilities or knowledge are mapped using class diagrams, situated in the respective views of the model.

Symbolic modeling methods, like for example ALEM, require that knowledge, procedures or decision making strategies are modelled explicitly. In contrast, connectionist modeling methods, like artificial neural networks, capture knowledge within their structure. Usually knowledge is learned and not encoded directly. The next section introduces neural networks in more detail.

1.2 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are derived from the biological concept of neural systems in nature [20]. Their strengths are their ability to learn from experience gained during operation, their fast data processing and their small modelling effort. Further, they have the ability to approximate complex mathematical functions which are completely unknown or cannot be described exactly. The knowledge of human experts is a typical example. The decision making of human experts is based on individual experience which cannot be mapped to an exact mathematical description. In this case, neural networks approximate the mathematical coherences in a “black-box” manner.

An artificial neural network basically consists of neurons which are structured in layers and connected via weighted links. There are at least two layers, the input and the output layer. Between these layers, an unlimited number of hidden layers is possible [5].

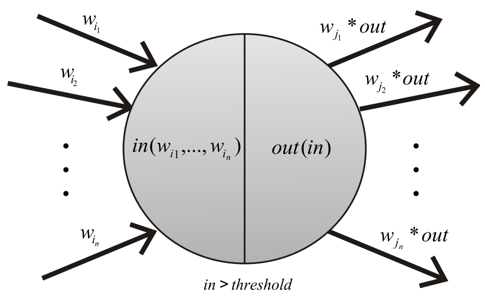

Figure 2. Artificial Neuron (own depiction)

Data processing within artificial neural networks is massively parallel. If a node receives an input signal, the value is transformed according to the node’s activation function and is checked against the node‘s threshold. If the value exceeds the threshold, the result either proceeds directly to succeeding nodes or an output-function calculates the output (Figure 2). By this means, data is processed parallel through the nodes of the different layers. The final result of the calculation is presented at the output nodes.

Neural networks can learn from experience gained through operation. Hereto, sets of training data are repeatedly presented to the network until it computes the favored output. The learned knowledge is stored within the edges’ weights. Correspondingly, a continuous adjusting of the internal weights takes place during the training process. This leads to an approximation of the mathematical function, which maps the input values to correct output values. In order to verify the generalization of the represented approximation, further sets of validation data are used. In addition, this avoids a phenomenon called *Overfitting* [8] which denotes a simple memorising of the presented training data.

In general, three types of learning processes can be distinguished. *Supervised Learning* denotes a procedure, where the input data as well as the expected output is presented to the neural network [2]. In the case of *Reinforcement Learning*, only the input data is presented. Instead of the desired output, the neural network obtains feedback whether the calculated values are correct or not [12]. Finally, *Unsupervised or self-organised learning* encompasses the presentation of input data without any feedback. In this case, the neural network tries to recognize patterns within the input data autonomously [10].

2 Adaptive Relocation Control (ARC)

Scholz-Reiter et al. introduced an intelligent production control concept for a customer oriented shop floor production using artificial neural networks [15]. Neural network based controllers relocate work pieces between machines and workstations on different production levels.

In production environments as well as in general, neural networks running over long time periods face the risk of *Catastrophic Forgetting* [4]. That is, that useful knowledge is overwritten with new information during the learning process. Due to this behavior, neural networks are mostly trained for only one specific task or situation. If the situation changes, the networks are either retrained or replaced. To face this problem, Scholz-Reiter et al. propose an additional software architecture (ARC) to structure the learning processes [17]. The software architecture supervise the performance of the neural networks and simultaneously train alternative networks. In case of decreasing performance, the active network is exchanged with an alternative one. This section first introduces the shop floor environment. Afterwards, the setup of the additional software architecture and its top level functions are described.

2.1 ARC SETUP

The neural control concept focuses on a shop floor environment [15]. The production facility is divided into several specialized departments, like a sawmill or a turnery. Work pieces can pass workstations and machines in any order. Shop floor production is characterised by a high degree of customization, caused by the production of only single pieces, prototypes and small series. Furthermore, customization leads to high variances concerning the order of production steps for every work piece.

The control concept following Scholz-Reiter establishes an inventory based control by means of artificial neural networks to this environment. It consists of closed control loops located between the different production stages of the shop floor. The embedded neural networks are responsible for the relocation of work pieces. Each time, a work piece is finished on a machine or workstation, a neural controller determines the subsequent facility the work piece is relocated to (see Figure 3). For each type of work pieces an appropriate neural controller is established between the stages of the shop floor.

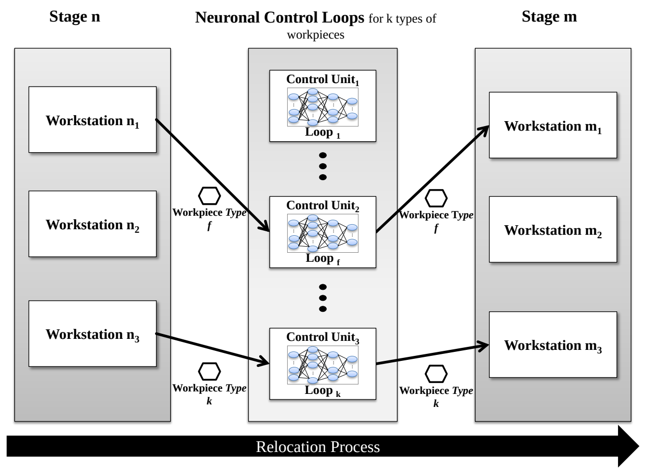

Figure 3. Strucure of the control concept (own depiction)

A neural controler consists of two neural networks, one for the relocation decisions and one for prediction proposes. The controller is completed by an optimizer. The decisional networks determine the optimal machine or workstation on the subsequent production stage. Therefore, setup and processing time of the regarded work piece, as well as the current inventory levels of possible successors are taken into account.

The networks for decision-making are feed-forward networks with two hidden layers. The number of output and input nodes depends on the count of alternative workstations on the next production stage. The networks are trained using the resilient-propagation algorithm [1] in a supervised procedure. The initial training was conducted offline with training sets modelling simple priority rules. The following online training bases on manually chosen examples gained through operation.

The predictive neural networks deliver forecasting values of future inventory levels influenced by the current relocation decision. Due to the comparatively high training effort of recurrent neural networks, the predictive networks are also conducted as feed-forward networks with two hidden layers [7]. To achieve a comparable performance, the prediction horizon was set to 7 days. For their supervised training process, the backpropagation algorithm [12] was used both for offline and online training. The training data for the initial offline training was derived from the inventory data logged during simulation runs with a rule based control.

Finally, an optimizer computes the optimal inventory level using an adaption of the simulated annealing algorithm. In this way, the inventory levels of all machines and workstations are optimized in respect of a maximal adherence to due dates for the shop floor.

2.2 ARC RELOCATION CONTROLLER

To cope with the problem of Catastrophic Forgetting, the control concept described in the previous section is extended by a software architecture [17]. The architecture for ARC encloses the networks (see Figure 4), and evaluates the performance of the employed neural network.

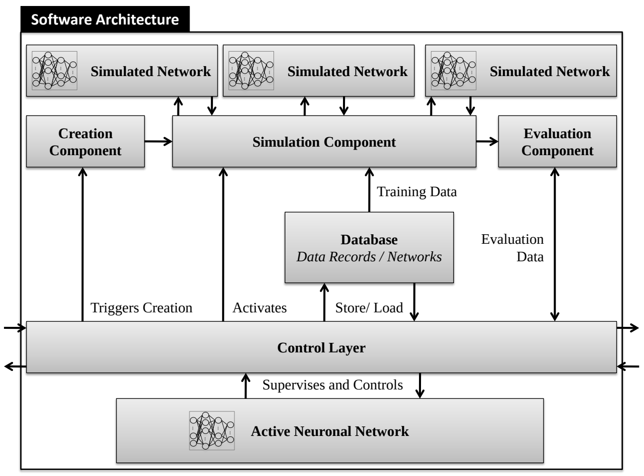

Figure 4. Strucure of the Adaptive Relocation Control (ARC) [17]

If the quality of decisions made by a neural network decreases, alternative networks are generated and trained. Employing internal simulations, the performance of the active neural network and the possible alternatives is compared. If one of the alternative networks achieves a better performance, an exchange takes place. The architecture further collects training data for the simulated alternative networks and supervises the current production situation.

The relocation controller can be interpreted as a three layered system containing the execution layer with the currently active network, the data storage as a mid-layer and a top layer containing control, evaluation and simulation. The functions covered by the controller are now described.

Evaluation The controller continuously measures the performance of every active and simulated neural network. In addition, it measures the global adherence to due dates. The continuous evaluation ensures that the most efficient neural networks are applied.

Creation each controller creates alternative networks, depending on the current production situation or on trends becoming apparent. Therefore, statistical a-priori knowledge is used to determine network architectures.

Simulation The training and simulation of alternative networks is carried out using an internal material flow simulation. The input data is collected during operation, which allows comparing the simulated and the running neural network's performances. The continuous simulation offers alternatives to the actual operating control.

Database The database associates replaced neural networks and production situations in which specific networks or network architectures performed well. In addition, it stores equally classified sets of training data. In cases where changes in the production situation become apparent, situation specific information can be retrieved.

Control The control component manages the interaction of single functions. The results of the evaluation are processed and, if necessary, the creation and simulation of alternative networks is triggered. In addition, the active network is exchanged if an alternative network performs better. Further, the component collects training data for the simulation as well as the required data to evaluate the controllers performance.

3 SHOP FLOOR SCENARIO IN ALEM

This section depicts the most important aspects of an ALEM model, corresponding to the ARC job shop example explained. Thus, the scenario's features are characterized first. Afterwards, those ALEM diagrams, related to the ARC controller are presented and a concept of integrating the neural networks into the model is proposed.

3.1 SCENARIO'S CHARACTERISTICS

ARC delegates the decision, to which workstation a work piece is relocated, to decentralized neural controllers. These controllers are set in between the production stages. This section investigates the scenario in terms of an autonomous control architecture.

Mainly, the system is characterized by three components: The work pieces, the workstations and the relocation controllers.

The workstations offer their ability to perform various manufacturing steps to work pieces. Each machine or workstation is associated to an input and to an output buffer storing work pieces. The workstation is aware of its current buffer status and can communicate the status on request.

The work pieces are manufactured in accordance to their type. Each product type is defined by a predefined sequence of production steps. In contrast to classic autonomous scenarios, work pieces do not select the next workstation/machine on their own, but request a decision from the corresponding relocation controller. Finally, the work piece executes the relocation controller's decision and proceeds to the selected workstation.

The relocation controllers The controller assumes the relocation of one certain work piece between two fixed production stages. If a work piece finishes manufacturing at a workstation, the relocation controller decides where on the next production stage the work piece shall processed.

Furthermore, the relocation controllers perform additional activities, like data acquisition, network evaluation and replacement as well as they collect training data. These activities implement another dimension of a decision strategy: The controller monitors itself, and adjusts its own decisions in accordance to global objectives.

The ARC control processes can be interpreted as autonomous control processes. Thus, the scenario is representable using ALEM. As ALEM focuses on procedural descriptions, model elements are introduced to capture the neural control strategy into ALEM.

Despite of the relocation controllers' decisional strategy, the scenario conforms to classical shop floor scenarios. Thus, following the reference models proposed by Scholz-Reiter [19] and Kolditz [11], an ALEM model can be built. The relocation controller consists of several top level functions, as well as of a set of neural networks. The next sections first examines which aspects of the controllers can be expressed using the actual ALEM notation, and which aspects require future work. Requirements for the integration of external neural control are stated and a possible interface is proposed.

3.2 MODEL OF THE CONTROLLER

The relocation controller is integrated into the structure view's class diagram [19] as specialization of the logistic resource (on the right of Figure 5). This allows the assignment of abilities, knowledge and behaviors to the controllers. The structure view's class diagram diagram models all elements present in the logistic scenario as well as the relations among them. As the relocation controllers are not part of the work pieces or workstations, they act as intelligent logistic objects and have to be included into this diagram.

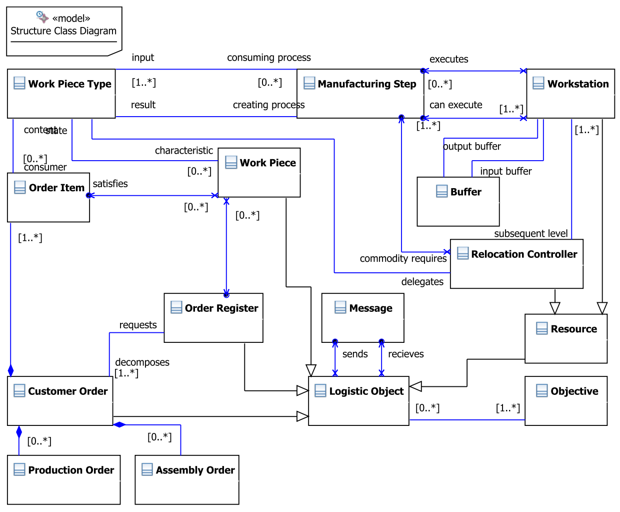

Figure 5. Adopted structure view's class diagram

The relocation controllers possess a set of basic abilities, which are incorporated into the ability view's class diagram. These abilities are essential to perform the top level functions provided. The corresponding interfaces are depicted in Figure 6.

On top of these basic abilities, more complex actions, as well as the controllers life cycle are established. The controller only adapts

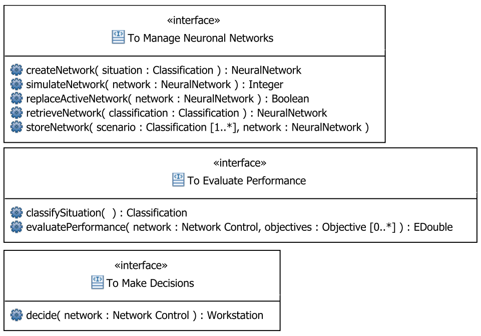
«interface»
To Manage Neuronal Networks
createNetwork(situation : Classification) : NeuralNetwork
simulateNetwork(network : NeuralNetwork) : Integer
replaceActiveNetwork(network : NeuralNetwork) : Boolean
retrieveNetwork(classification : Classification) : NeuralNetwork
storeNetwork(scenario : Classification [1..*], network : NeuralNetwork)
«interface»
To Evaluate Performance
classifySituation( ) : Classification
evaluatePerformance(network : Network Control, objectives : Objective [0..*]) : EDouble
«interface»
To Make Decisions
decide(network : Network Control) : Workstation

Figure 6. Abilities of the relocation controller

to two states: “Idle” and “Relocating”. While relocating the *Make Decision* activity is applied. While the controller is idle, it constantly executes the *Maintain Performance* activity. The corresponding diagrams are depicted in Figure 7. These activities make use of the abilities defined for the controller.

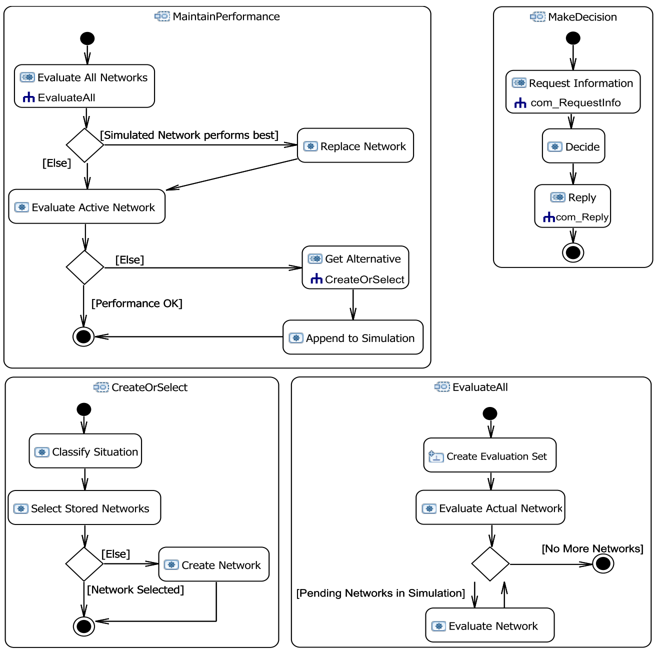

Figure 7. Relocation controller’s UML-activity diagrams (simplified)

3.3 REQUIREMENTS TOWARDS THE INTEGRATION

The ALEM-Tool is designed to model and simulate ALEM models. Thus, the networks have to be integrated in a way, that they can be used within the model’s simulation. Therefore, the networks are represented as a part of the model. Consequently, the corresponding model elements have to include all information which is necessary to address the networks within the simulation.

The representation and execution of the neural networks are the only aspects of the relocation controller, which cannot be captured by ALEM in a straight forward way. This section investigates these aspects.

The execution of these neural networks depends on the software used to implement them. The networks used in this example are implemented using the SNNS environment [2]. This environment provides TCP/IP interfaces to interact with the networks.

Representation: The external information source is interpreted as knowledge. Thus, a visual representative is integrated into the ALEM knowledge view. In this case, it includes the network’s TCP/IP address and port, as well as the parameters which are required to execute the network. The graphical elements, representing the neural networks are depicted in Figure 8.a.

Execution: Two additional abilities must be created to execute the networks. First, an ability to create the connection and to execute the network and second, an ability to interpret the results. In this case, the networks return a number corresponding to one of the machines on the next level. This value must be mapped to the concrete machine used in the ALEM model. The abilities are modelled within ALEM’s ability view (Figure 8.b)

Based on the knowledge element which represents the external information source, the executional ability can be derived. Furthermore, the knowledge object references both, the network’s return type, as well the ALEM type it corresponds to. This allows the derivation of the interpretation ability’s signature.

To enable an execution of these abilities in an arbitrary simulation platform, both functions must be integrated into ALEM’s reference libraries. Thus, templates can be created to transform the abilities’ notational elements into an executable form.

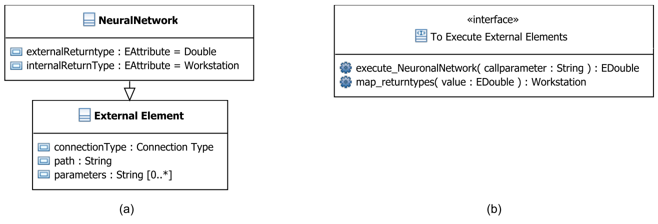
NeuralNetwork
externalReturntype : EAttribute = Double
internalReturnType : EAttribute = Workstation
External Element
connectionType : Connection Type
path : String
parameters : String [0..*]
(a)
«interface»
To Execute External Elements
execute_NeuronalNetwork(callparameter : String) : EDouble
map_returntypes(value : EDouble) : Workstation
(b)

Figure 8. (a) Required knowledge elements; (b) Additional abilities

Another important requirement towards the simulation of models involving external information sources is the selection of the simulation environment. In accordance to the type of external source, the environment must be able to establish a connection, for example via TCP/IP or via call operation for external files. It must provide means to send commands and receive responses from the data source. Therefore, the inclusion of such black -box elements requires the selection of specific simulation platforms.

[2] Stuttgart Neural Network Simulator, http://www.ra.cs.uni-tuebingen.de/SNNS/

4 CONCLUSION AND OUTLOOK

The article provides a concept of integrating neural networks into the ALEM modeling framework. Therefore, a shop floor example employing the ARC control approach is introduced. The scenario's neural controllers are interpreted as resources, to enable modelling in ALEM. Further, notational elements to depict external, neural decision units are proposed.

As a major drawback, the presented way of integrating neural networks includes the addition of external software components. The interface to these components cannot be modelled in ALEM directly. Therefore, it has to be implemented within the simulation environment. This requires the user to possess at least basic experience in simulation programming. Furthermore, the inclusion of external decision elements, like in this case of neural networks, reduces the completeness of the ALEM models. Thus, it is rendered impossible to trace the decisional behavior made in the modelled system only by examining the model itself.

Integrating neural controllers into the ALEM framework provides the opportunity to recreate an existing system's behavior, without formally describing the decision processes. Therefore, the real-world system's decisions can be used as training data for the networks. This eases the derivation of a model of the original system. The proposed way of representing neural networks in ALEM enables an exchange of those networks by other decision strategies. This supports experimenting with autonomous control strategies. Furthermore, the simulation of the original system's model provides data, suitable to compare the performance of different control strategies within the same scenario.

In addition, exchangeable decision strategies simplify the development of scenarios applying neural networks. Data gained through simulation runs using default control strategies can be used as a base for the initial training of neural networks. Further adaptions to a desired decision strategy can be obtained by online training. The notational elements proposed for ALEM are kept general. Thus, it is possible to create templates to include other non-procedural information sources. It is necessary to investigate extensions of these elements and to define clear structures and processes to generate and maintain connections to other external information sources. With focus on a possible simulation, the ALEM reference libraries must be adopted to provide simulation specific implementations of the related operations and data structures. Therefore, suitable simulation environments must be investigated.

ACKNOWLEDGEMENTS

This research is funded by the German Research Foundation (DFG) as part of the Collaborative Research Center (CRC) 637 "Autonomous Cooperating Logistic Processes - A Paradigm Shift and its Limitations", subproject B2, as well as part of the project "Automation of continuous learning and examination of the long-run behavior of artificial neural networks for production control", index SCHO 540/16-1.

REFERENCES

[1] Aristoklis D. Anastasiadis, George D. Magoulas, and Michael N. Vrahatis, 'New globally convergent training scheme based on the resilient propagation algorithm', *Neurocomput.*, **64**, 253–270, (2005).

[2] D.K. Chaturvedi, 'Artificial neural network and supervised learning', in *Soft Computing Techniques and its Applications in Electrical Engineering*, 23 – 50, Springer Verlag, Berlin Heidelberg, (2008).

[3] S.F. Crone, 'Forecasting in inventory management using artificial neuronal networks - a novel approach through asymmetric cost functions', in *Einsatz von Fuzzy-Sets, Neuronalen Netzen und Künstlicher Intelligenz in industrieller Produktion und Umweltforschung*, 59 – 69, VDI - Verlag, Düsseldorf, (2003).

[4] Akira Date and Koji Kurata, 'A property of neural networks of associative memory with replacing units', *Artificial Life and Robotics*, **12**(1-2), 291 – 294, (2008).

[5] G. Dreyfus, *Neural Networks Methodology and Applications*, Springer-Verlag Berlin Heidelberg, Berlin Heidelberg, 2005.

[6] Marco Garetti and Marco Taisch, 'Neuronal networks in production planning and control', *Production Planning and Control*, **10**, 324 – 339, (1999).

[7] T. Hamann, *Lernfähige intelligente Produktionsregelung*, Gito Verlag, Berlin, 2008.

[8] S. Haykin, *Neural Networks and Learning Machines (3rd Edition)*, Prentice Hall, New Jersey, USA, 2008.

[9] *Understanding of Autonomous Cooperation and Control in Logistics The Impact of Autonomy on Management, Information, Communication and Material Flow*, eds., M. Hülsmann and K. Windt, Springer Verlag, Berlin, 2007.

[10] *Self-Organizing Maps*, eds., T. Kohonen, M. R. Schroeder, and T. S. Huang, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[11] Jan Kolditz, *Fachkonzeption für selbststeuernde logistische Prozesse*, Ph.D. dissertation, Universität Bremen, Berlin, 2009.

[12] Wolfram-Manfred Lippe, *Soft-Computing: mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen (eXamen.press)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[13] A.-W. Scheer, *Business Process Engineering - Reference Models for Industrial Enterprises*, Springer, Telos, 1994.

[14] A.-W. Scheer, *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen*, Springer Verlag, Berlin, 2001.

[15] B. Scholz-Reiter and T. Hamann, 'The behaviour of learning production control', *CIRP Annals - Manufacturing Technology*, **57**(1), 459 – 462, (2008).

[16] B. Scholz-Reiter, T. Hamann, H. Höhns, and G. Middelberg, 'Decentral closed loop control of production systems by means of artificial neural networks', in *Proceedings of the 37 th CIRP-International Seminar on Manufacturing Systems*, pp. 199 – 203, (2004).

[17] B. Scholz-Reiter, F. Harjes, and T. Hamann, 'Automatisierung des Lernens neuronaler Netze in der Produktionssteuerung', *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, **1-2**, 101–104, (2010).

[18] B. Scholz-Reiter, J. Kolditz, and T. Hildebrandt, 'Engineering autonomously controlled logistic systems', *International Journal of Production Research*, **47**(6), 1449–1468, (2009).

[19] B. Scholz-Reiter, S. Sowade, T. Hildebrandt, and D. Rippel, 'Modelling of autonomous control explicitly considering orders as a kind of immaterial logistic objects in manufacturing processes', in *Proceedings of the 3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2009)*, München, Deutschland, (2009).

[20] W.-H. Steeb, *The Nonlinear Workbook: Chaos, Fractals, Neural Networks, Genetic Algorithms, Gene Expression Programming, Support Vector Machine, Wavelets, Hidden Markov Models, Fuzzy Logic with C++, Java and SymbolicC++ Programs*, World Scientific Publishing Co. Pte. Ltd, Singapore, 2008.

[21] N. Wang and J. Yu, 'Neuron based nonlinear pid control', *PRICAI 2006: Trends in Artificial Intelligence*, **4099**, 1089–1093, (2006).