Available online at www.sciencedirect.com

ScienceDirect

Engineering Applications of Artificial Intelligence 20 (2007) 691–698

Engineering Applications of ARTIFICIAL INTELLIGENCE

www.elsevier.com/locate/engappai

AI techniques in modelling, assignment, problem solving and optimization[☆]

Zsolt János Viharos*, Zsolt Kemény

Computer and Automation Research Institute, Hungarian Academy of Sciences, H-1111, Kende u. 13–17, Budapest, Hungary

Received 6 November 2006; accepted 20 November 2006
Available online 25 January 2007

Abstract

This paper recapitulates the results of a long research on a family of artificial intelligence (AI) methods—relying on, e.g., artificial neural networks and search techniques—for handling systems with high complexity, high number of parameters whose input or output nature is partly unknown, high number of dependencies, as well as uncertainty and incomplete measurement data. Aside from classical modelling, basic problem solving and optimization techniques are presented. Finally, a novel submodel decomposition method is shown with an extended feature selection algorithm highlighted, along with possibilities of further development. Examples of practical application are shown to illustrate the viability of the methods.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Neural networks; Submodel decomposition; Model reuse; Production complexity; Extended feature selection; Modeling; Problem solving; Optimisation

1. Introduction

Nowadays, many areas of practical science and technology face systems with high complexity and notable uncertainty whose handling requires efficient methods for learning the system's properties and dependencies and depositing this knowledge in a flexible, reusable form which can be easily reconfigured for various interpretations. While these tasks are already demanding themselves, even the systems to be handled impose many challenges.

A high number of parameters: For complex systems consisting of many—more or less identifiable—components, such as production lines, biochemical processes etc., it is quite natural that a large number of parameters are required for their adequate description. While some of these systems or phenomena allow simple and uniform handling of some groups of parameters (it is enough to think of finite element methods), this cannot be easily done in a significant number of other problems (see Fig. 1).

A high number of dependencies: For several components integrated into one system, the number of dependencies to be modelled quickly increases, as numerous new relations are added to those inherent to the subsystems (see Fig. 1). However, not only the high number of actually existing dependencies would require vast resources, but also the even higher number of *potential dependencies*, i.e., assumptions which have to be either verified or rejected during modelling, calling for an efficient way of pruning superfluous relations.

Uncertainty of measurement data: The first step of gathering knowledge about physical systems is measurement which also introduces noise and data uncertainty. Aside from the measurement data themselves, it is thus important to know the expected tolerances for both measurement and reproduction in a subsequent simulation.

[☆]The research work presented in this paper is in part financed through the EU 6th Framework Project No. NMP2-CT-2004-507487 *VRL-KCiP—Virtual Research Lab for Knowledge Communities in Production—network of excellence* (with respect to knowledge handling) and *Multisens* (with respect to production feedback) under contract no. 512668, as well as the Hungarian National Research Fund under Grant no. OTKA T043547 (with respect to production control, submodel extraction).

*Corresponding author. Tel.: +36 1 279 6195; fax: +36 1 466 7503.
E-mail addresses: zsolt.viharos@sztaki.hu (Z.J. Viharos), kemeny@sztaki.hu (Z. Kemény).

0952-1976/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2006.11.007

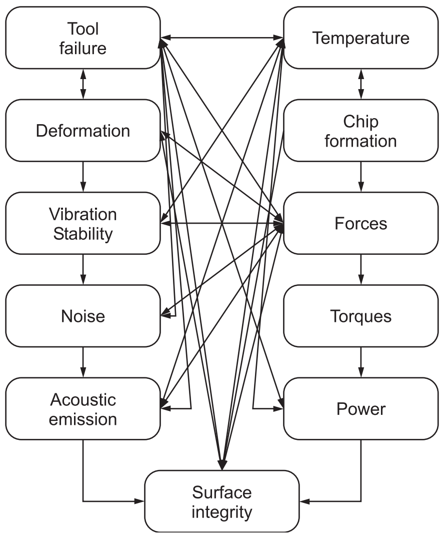


Fig. 1. A practical example for complexity: key interdependencies for a cutting process (Peklenik and Jerele, 1992). Each basic node may contain numerous parameters—the total number of variables can easily go into the hundreds for such industrial processes.

Incomplete information: Somewhat related to measurement data uncertainty is the partial availability of information which practically means that some vectors of measured data are obtained with some of their elements missing (either due to sensor malfunction or because at the given point of the measuring process, the given data element would have no sensible meaning, (Zhang and Rong, 2005) or totally invalid. In some cases, discarding these incomplete vectors would be unpracticable as they make up a substantial part of all measured data. Therefore, a method is needed which can handle incomplete data sets.

Unknown input/output character of parameters: For data collected at various measurement points and labelled as relevant parameters, it is, in many cases, still not clear whether they are to be regarded as an input or an output in some relation. The input/output nature of a parameter may either be determined by the given problem to be solved or the given point of view of modelling (and may thus be different for another problem), or it may be entirely unknown if a priori knowledge about the system is sparse.

To overcome these difficulties, a flexible approach is needed which can reuse knowledge already acquired for a given input/output arrangement. *Also, since complex systems show notable similarity with respect to the above mentioned properties, one may be encouraged to reuse methods—elaborated for one type of complex systems—for various other cases after performing reworking or fine-tuning as needed.*

The range of problems stated above is a natural terrain for various artificial intelligence (AI) approaches. In this paper, a family of methods based on artificial neural networks (ANNs) is presented which was successfully applied in several industrial examples. First, classical modelling is addressed, including handling missing data, uncertainties and unknown input/output arrangement. Hereafter, basic problem solving and optimization techniques are presented, and finally, a novel submodel decomposition method is shown, along with possibilities of further development.

2. Classical modelling

If sufficient measurement information is available, the first step toward handling a system is the creation of an adequate model. This is essential for setting up planning, control or prediction methods, as well as for testing and validating them in a simulation environment before practical application. In numerous cases, one faces nonlinear relations which may contain significant uncertainty and may even change over a longer time. In these cases, the limits of conventional methods—as differential equations or simple function fitting and interpolation—are quickly reached while AI techniques, as demonstrated by many cases over the past decade, are still able to deal with these modelling tasks.

In a number of cases, ANNs are used for modelling complex nonlinear relations, as they can easily handle strong nonlinearities, a large number of parameters and missing data, furthermore, they can adapt to changes occurring in the modelled system or process (Cholewa, 2005). *In its classical way of application*, the layout of an ANN is largely predetermined by the fixed classification of variables as inputs or outputs, yet aside from this, no assumptions about the approximated function are taken into consideration. Having set up a network topology, learning and test patterns compiled from measurement data are presented to the network and finally, the knowledge stored in the ANN can be recalled using only predetermined inputs as inputs and obtaining the results at the predetermined outputs. There are, however, four requirements which appear in modelling large and complex systems and render the conventional use of ANNs inefficient:

1. Non-invertible functions are not guaranteed to be successfully modelled with ANNs in their conventional layout. Most often, this is due to the fact that the input/output assignment of variables was done in advance of training, with insufficient knowledge about the given relation's nature. This may become a trap, since ANNs are typically applied exactly when there is little known about the function and its general type as well.
2. If several ANN models are concatenated to obtain a given result, approximation errors may increase beyond acceptable bounds at the last output layer. Coupling conventional ANNs may only be safe if their accuracy is guaranteed to lie within specified limits. (Note that a useful approach was proposed by Ghiassi and Saidane (2005), where an initial neural network is augmented by further nonlinearities and additional layers if accuracy deteriorates beyond a given bound.)
3. Even for the same system, a wide scope of problems may exist. These are related to the same system—which

suggests that their solution can rely on the same knowledge—but in the classical approach, each of them would require separate ANNs, each having its own specific input/output configuration.

4. Since obtaining knowledge about a complex system is cumbersome and demands large computational efforts, therefore, such knowledge should be stored in a reusable form—this is, however, hardly possible with the rigid configuration of a classical ANN.

These difficulties can be overcome if a generic, reusable ANN-based model is compiled—exactly this strategy is followed by the approach of Viharos et al. (2002). Prior to applying this method, a sufficiently large set of training patterns must be given, with allowable estimation tolerances assigned to each component of the learning vectors. At this stage, it is not necessary to know the input/output assignment of the components, as the algorithm will find the best configuration automatically. Performing a complete search of all possible configurations would be too slow, therefore, a heuristic method, *sequential forward selection* is used (Viharos et al., 1999a). Here, a possible output candidate is selected and the learning performance (learning speed and accuracy) of the ANN is monitored, keeping allowable tolerances of the given parameter in mind. Should the training of the ANN for a given input/output arrangement succeed, the selected variable becomes eligible for being an output. Once a variable is validated as an output, further output candidates are selected and checked until the highest number of outputs is found. *This automatic search can also detect non-invertible relations*, as in their case, training the input/output arrangement corresponding to their inverse fails. An important characteristic of the method is the unchanging topology of the network where, as shown in Fig. 2, unused neurons and links are not deleted but only protected from being altered during learning.

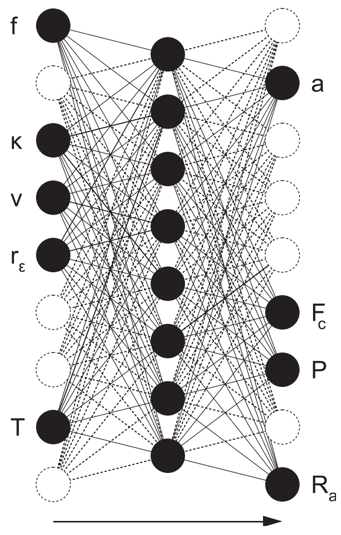


Fig. 2. Protected (dotted) and unprotected (solid) states of different neurons and corresponding weights.

A similar treatment is applied when *incomplete data sets* are encountered. While numerous methods paste up missing components in training and test data by interpolation, the concept of Viharos et al. does not make this necessary. *Here*, *weights corresponding to missing data are protected and remain omitted by the given learning step*. As a result (see Viharos et al., 2002), this is suitable for handling incomplete data. *Interestingly*, "*impaired*" *training vectors often bring better learning results if data vectors to be learned contain redundant information.*

3. Problem solving

Having once assembled the general, multi-purpose ANN-based model as described above, it can be used to solve a wide variety of problems, *independent of the input/output arrangement which may vary from task to task*. The key to the model's versatility arises, aside from the fact that it is among the best models attainable with an ANN of a given size and topology, from the possibility of both direct and indirect use. In any application of the general model, unknown parameters are estimated using known ones, three cases being possible (Fig. 3):

- *Classical approach*—Here, exactly those parameters are unknown which are outputs of the ANN containing the general model. In this case, the use is straightforward; known parameters are fed into the ANN whose outputs directly deliver the unknown ones.
- *Inverse approach*—Here, unknown parameters coincide with the input variables of the ANN. In this case, an iterative search finds a set of inputs approaching the desired (known) output to a prescribed degree. This method is also suitable for non-invertible relations.
- *General use*—A some of the unknown parameters are outputs of the ANN while others are inputs. In this case, known inputs are fed straight into the ANN while the rest of the task is again performed via iterative search.

Should an indirect, iterative search be needed, the solution is sought in compliance with the following three constraints (the fourth one being the relations of the model itself):

- *Condition regarding known output parameters*—complying with this constraint ensures that a valid solution estimates known output parameters by forward calculation within specified bounds of estimation error.
- *Condition regarding unknown input parameters*—this is determined by the valid domain of inputs of the ANN model which is assumed to be covered by the set of training data.
- *Condition regarding unknown output parameters*—determined by the valid range of the ANN's output, a prospective solution is only accepted if the unknown output remains in this acceptable range.

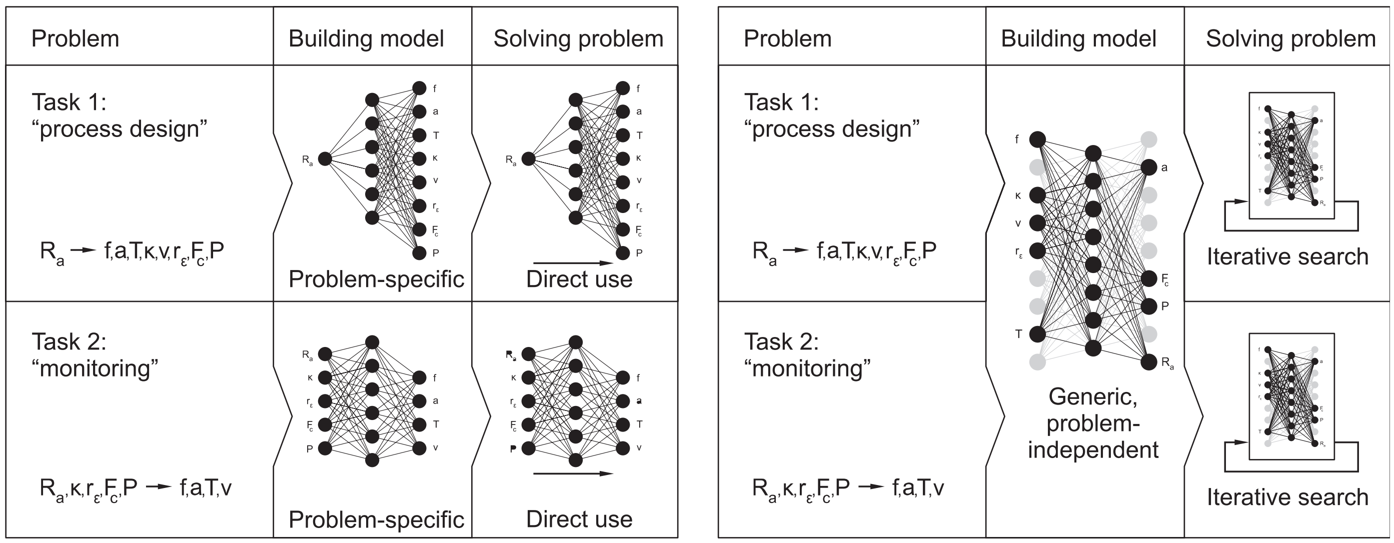


Fig. 3. Steps of problem statement, modelling and problem solving for the classical use of ANNs (left) and the task-independent generic ANN model (right).

In Monostori and Viharos (2000), examples from industrial production show the role of a proper input–output search in picking out non-invertible relations to learn them the correct way. Should a non-invertible relation be encountered, learning it imposes no hindrance to a correctly configured ANN; however, the non-invertible nature does show in the high number of solutions found in an indirect problem solution process.

4. Optimization

If the solution to the problem, e.g., due to non-invertible dependencies, is not unique, an additional criterion can be used to pick an optimal solution from the set of possible ones. In this case, it would be straightforward to let the aforementioned requirements still act as constraints and the secondary preference act as a criterion to be optimized during search. Experience, however, has shown that it is more advantageous to loosen the constraints and handle them as criteria with weights varying during subsequent optimization steps. The solution found this way is then expected to optimize the secondary preferences to the best possible degree while it is still in keeping with the constraints within an agreeable distance.

An application example of iterative optimization with such constraints is shown by Viharos et al. (1999b) where *simulated annealing* is used to obtain a set of valid solutions to manufacturing problems. Also, the technique initially applied to only one production step is extended in Viharos et al. (1999b) to a higher level of production: the block-oriented *ProcessManager* framework presented in Viharos et al. (1999b) can deal with an entire process chain where the result of an earlier step may influence all subsequent steps.

5. Submodel decomposition

Highly complex systems imply models which—due to the high number of parameters and the dense network of interrelations—can be handled as a whole only at staggering computational costs. It is thus advantageous to decompose these complex models to several smaller interconnected submodels which can be easily handled one by one (moreover, we can always select the set of submodels relevant to the given problem, so that submodel decomposition results in subtask decomposition as well). For this purpose, a submodel finding approach combining feature selection and artificial neural networks—a culmination of the ANN-based techniques presented before—was developed by Viharos et al. (2003) and Viharos (2005). The application of the algorithm has two main prerequisites:

- A data set of sufficient size has to be supplied, e.g., in form of a database table where columns represent the variables to describe the system and each row stands for these variables recorded at a given time.
- Since in subsequent parts of the algorithm, an ANN is employed to test whether a given variable can be estimated using other parameters, a maximal tolerable error has to be assigned to each variable when estimating it with an ANN model.

Having fulfilled these requirements, an algorithm can be run which uses ANNs to validate proposed submodels. In the most "conventional" case, the assignment of potential inputs and outputs as well as the isolation of proposed submodel structures is done in a separate block, prior to any ANN training, as proposed by earlier approaches (e.g., Caelli et al., 1999). Departing from this rigid setup, one can allow the structure of the interconnected submodels to be determined dynamically during learning.

The novel method presented in Viharos (2005) allows the flexible configuration of submodels, as well as free assignment of a given variable for input or output. As shown before, the highest number of outputs is selected in an input/output search based on ANN-learning performance. However, attempting to learn a potential output by an ANN can only signalize that there is a dependency "somewhere within the set of selected variables" but cannot weed out parameters totally independent of the given subsystem. This would result in a single ANN struggling to learn the entire structure in question, therefore, the reduction to smaller, easy-to-handle submodels must be cared for by other means. While the vast majority of such approaches determines the submodel structures before any ANN training takes place, this new method identifies the submodel structures dynamically, leaning on the results of earlier ANN training periods.

This is accomplished by an extended feature selection algorithm—developed by Viharos et al.—running on the complete set of variables and setting up a decision tree for submodel selection.

The extended feature selection algorithm applied here assumes a pure classification task, onto which even continuous parameters can be mapped with an appropriate heuristics. In the first step, a given parameter is selected and its values encountered in the training data set are grouped into clusters (i.e., intervals of equal length), so that at least one element is contained in each interval (this, in itself, being the first heuristic decision). Next, the algorithm checks how "distinct" these clusters are, i.e., how far apart the weight centers of the clusters are and how large the distances of the cluster weight center and the cluster's discrete points are. This test is performed for all parameters that can come in question, then, the one exhibiting the most "distinct" clustering of values is chosen.

Having selected the first parameter of interest, all remaining variables are tested again, each of them *together with the already highlighted parameter*, for the same measure of class distinctness, using Euclidean distances. Again, the parameter chosen to form a potential submodel together with the first preferred variable will be the one exhibiting the best class separability *together with the parameter already selected in the first run*. In every subsequent step, yet another unselected variable is tested the same way, and in every case, the one corresponding to best class separation is chosen (note that this *incremental selection*, as opposed to a combinatorically exhaustive test, is the second heuristic decision in the algorithm's layout).

Adding new parameters to the ones already selected, a deterioration of class separability can be observed which is guaranteed to be worst when all variables are taken for classification. However, since our goal is the creation of submodels, each containing only a relevant part of the model's entire parameter set, a suitable heuristics (the third such case in the algorithm) should be used to decide when adding new parameters should be stopped. By selecting only a part of the model's entire parameter set as the best performing variable group for one given clustering, we have created a candidate for a submodel.

Since three heuristic decision steps were taken to obtain the candidate submodel, this can be considered only an *assumption* which is to be either *verified* or *rejected* by the ANN algorithm. The latter begins validating a given part of the submodel structure—at a given point in the decision tree—and delivers first training results. Examining these and removing the successfully learned submodel from the "pool" of unclassified variables, feature selection is run again on the remaining data set and the decision tree is reconfigured if needed. Hereafter, ANN training takes place again. Thus, the method does not separate preselection and ANN training into disjoint tasks—in fact, *feature selection and training complement each other with their alternate execution* until all submodels are identified and learned.

Having completed the decomposition, the following results are obtained (see also Fig. 4):

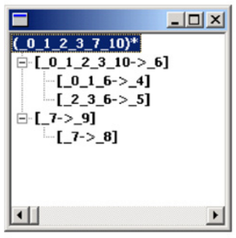
- A set of valid submodels, each containing a minimal set of the system's parameters with as many of them labelled as output as the ANN algorithm could find.
- A set of rejected submodels. These were originally proposed as submodels by the feature selection procedure but were judged invalid by the ANN algorithm. Storing these discarded patterns is useful for an early pruning of submodel candidates bound to fail.
- Since the valid submodels were spotted while ANNs were learning their parameter dependencies, this knowledge is readily accessible and applicable for problem solving as a network separate neural nets, each of them representing one submodel.

Fig. 4 shows a screenshot of an actual industrial application in a rather low level of manufacturing where a production line is modelled using more than 60 parameters. In Viharos et al. (2003), another industrial application is shown for an intermediate level of production.

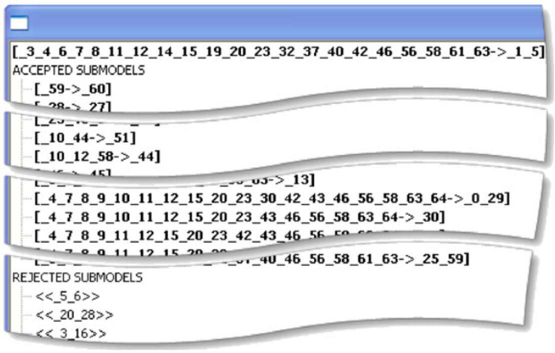
6. Further research

Currently ongoing research activities are aimed at extending submodel decomposition toward the framework of a multi-agent system (MAS) where knowledge specific to an agent is mapped onto a given submodel. The feasibility of this generalization is assumed because of remarkable analogies between ANN-based submodels and agents:

1. *Autonomy*—(a) *Decomposition* is present in a network of submodels, as well as in MAS. (b) Both submodels and agents represent *localized knowledge* with direct information exchange links only leading to its connective neighborhood.
2. *Architecture*—Composite models and MAS both have a *network architecture*.



```
(_0_1_2_3_7_10)*
[_0_1_2_3_10->_6]
    [_0_1_6->_4]
    [_2_3_6->_5]
[_7->_9]
    [_7->_8]
```



```
[_3_4_6_7_8_11_12_14_15_19_20_23_32_37_40_42_46_56_58_61_63->_1_5]
ACCEPTED SUBMODELS
    [_59->_60]
    [_28->_27]
    [_10_44->_51]
    [_10_12_58->_44]
    _63->_13]
    [_4_7_8_9_10_11_12_15_20_23_30_42_43_46_56_58_63_64->_0_29]
    [_4_7_8_9_10_11_12_15_20_23_43_46_56_58_63_64->_30]
    [_4_7_8_9_11_12_15_20_23_42_43_46
    _40_46_56_58_61_63->_25_59]
REJECTED SUBMODELS
    <<_5_6>>
    <<_20_28>>
    <<_3_16>>
```

Fig. 4. Two cases of submodel decomposition. In the example to the left, the net of accepted submodels consists of five main relations (in brackets), partitioning a system containing 11 description parameters. The fourth row in the window, e.g., shows that the algorithm identified a submodel with parameters 2, 3 and 6 as inputs for the estimation of output 5. The four identified submodels have common parameters, e.g., parameter 6 is estimated by the submodel shown in the second row, but it is to be found among the input variables of the next two submodels, too. Thus, a structure of interconnected submodels can be recognized additionally to the identification of its individual parts. To the right, the result of submodel decomposition in an industrial example with a large number of system parameters is shown.

3. *Adaptivity and proactivity*—(a) An elementary property of ANNs is their *learning ability*, also required for autonomous agents in a time-varying environment. (b) *Estimation* using locally available information, an inherent feature of submodel ANNs, is analogous to agents judging the *expected outcome of their own actions*, as well as the *anticipated changes in the near future*.

It can be thus assumed that submodel decomposition can be a basis for automatic agent formation in a MAS, submodel groups being the specific knowledge of the agents. *An important goal of such an initial agent formation is to break—at least partly—with the traditional practice of a rigid agent structure and allow free agent formation according to a given criterion.*

Prior to specific research activities, following crucial questions must be examined:

1. What criteria should lead one when taking groups of relations for an agent's specific knowledge? The key concerns are maximal *foreseeing* and *learning abilities* of agents. The requirement of acceptable global operating costs will later need further criteria, though this was so far only treated as a side effect in MAS (e.g., in Pěchouček et al., 2000).
2. How should learning data be grouped to express time-dependency correctly? Experiments may provide hints, but specific technical knowledge (changeover time or timing for one given workpiece) is expected to provide the first starting point.

While this initial agent formation may bring interesting results, the most important advantage expected is the run-time adaptation of the system's composition, the agents being reconfigured by the reorganization of the underlying submodels. To this end, *each agent is constantly monitoring its own performance* (the ratio of right and wrong decisions, the gain of bids won, etc.). Difficulties can be handled by the following steps:

1. First, the agent attempts to re-learn the deficient task, using recent training data, while inter-agent activities remain the same as in a conventional setup (Fig. 5).
2. Upon failure of re-training, the agent shares the unsolved problem with its neighbors and monitors them until it shows best (or agreeable) performance again (Fig. 6).
3. Should a neighboring agent constantly perform better, the problem submodel in question can be relocated to it, along with the associated decision rights. This may increase communication, yet the quality of decisions improves (possibly due to the new host of the task having better access to vital information). Relocation cost metrics can show if reassigning a given task improves total system performance (Fig. 7).

Two AI paradigms can be recognized here: reinforcement learning and distributed AI, several local blackboards being pivot points for reconfiguration. Experience in these domains provides hints, yet only actual tests can show how much—if any—supervision is needed to keep certain bounds, e.g., human understandability or stability. Dynamic reconfiguration introduces a new facet of agent communication: in addition to communicating decisions, bids, possibilities or parameters, agents would now *transfer decision rights and the corresponding knowledge*, *as well as protocols and knowledge topics*, requiring a new mode of communication, new protocols and new contents.

7. Conclusion

The first part of this paper highlighted fundamental phenomena encountered in highly complex physical systems (high number of parameters, high number of dependencies, uncertainty of measured data, incomplete

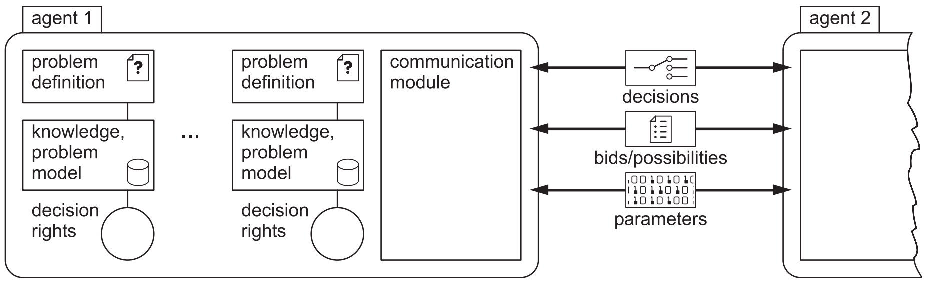


Fig. 5. Elements of conventional agent communication.

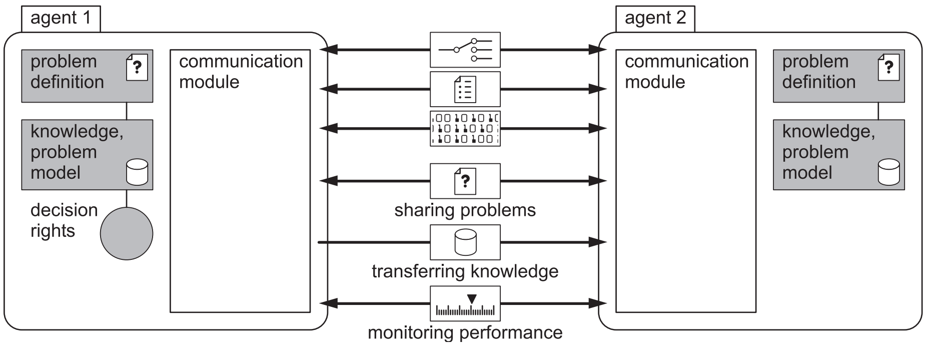


Fig. 6. Sharing an unsolved problem with other agents of the neighborhood.

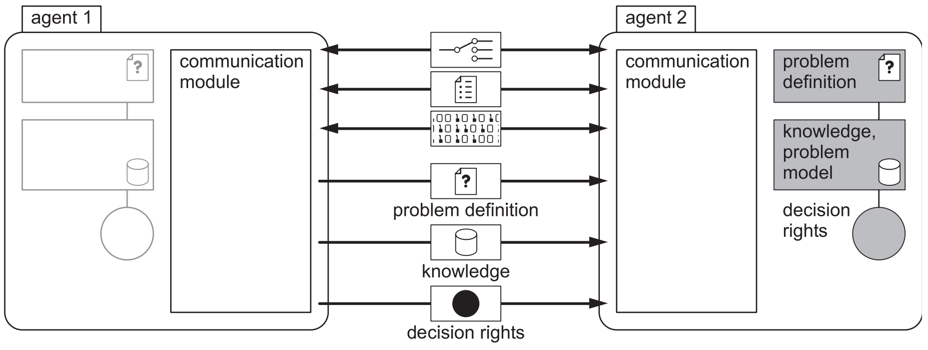


Fig. 7. Transferring knowledge and decision rights.

information and unknown input/output arrangement of parameters). To handle these in various types of problems, a family of ANN-based methods was presented (classical modelling, problem solving, optimization and submodel decomposition) which can be equally applied in lower and higher hierarchical levels of production. Their versatile applicability was demonstrated by examples of practical use in manufacturing systems. Finally, plans for future research were proposed where submodel decomposition will be used as a starting point for a flexible, reconfigurable multi-agent system.

References

Caelli, T., Guan, L., Wen, W., 1999. Modularity of Neural Computing. Proc. IEEE 87 (9), 1497–1518.

Cholewa, W., 2005. Identification of relevant signal features. Proceedings of the 10th IMEKO TC10 International Conference, pp. 1–4.

Ghiassi, M., Saidane, H., 2005. A dynamic architecture for artificial neural networks. Neurocomputing 63, 397–413.

Monostori, L., Viharos, Zs.J., 2000. Quality-oriented modelling and optimisation of production processes and process chains. Proceedings of the 16th IMEKO World Congress, pp. 439–444.

Pěchouček, M., Vokřínek, J., Bečvář, P., 2000. ExPlanTech: multiagent support for manufacturing decision making. Journal of Intelligent Manufacturing 20 (1), 67–74.

Peklenik, J., Jerele, A., 1992. Some basic relationships for identification of the machining processes. Annals of the CIRP 41 (1), 155–159.

Viharos, Zs.J., 2005. Automatic generation of a net of models for high and low levels of production. Proceedings of the 16th IFAC World Congress, Reg. No. 05127.

Viharos, Zs.J., Monostori, L., Markos, S., 1999a. Selection of input and output variables for ANN-based modeling of cutting processes. Proceedings of the 10th Workshop on Supervising and Diagnostics of Machining Systems, pp. 121–131.

Viharos, Zs.J., Monostori, L., Markos, S., 1999b. A framework for modelling, monitoring and optimisation of manufacturing processes and process chains by using machine learning and search algorithms. Proceedings of the Ninth IMEKO TC-10 International Conference on Technical Diagnostics, pp. 249–254.

Viharos, Zs.J., Monostori, L., Vincze, T., 2002. Training and application of artificial neural networks with incomplete data. Proceedings of the 15th International Conference on Industrial and Engineering Application of Artificial Intelligence, Lecture Nores on Artificial Intelligence, Springer Computer Books, Springer, Heidelberg, pp. 649–659.

Viharos, Zs.J., Monostori, L., Novák, K., Tóth, G., Csongrádi, Z., Kenderesy, T., Sólymosi, T., Lőrincz, Á., Kóródi, T., 2003. Monitoring of complex production systems in view of digital factories. Proceedings of the 17th IMEKO World Congress—Metrology in the 3rd Millennium, pp. 1463–1468.

Zhang, Q., Rong, G., 2005. Industrial application of data reconciliation for hybrid systems. Proceedings of the 16th IFAC World Congress, Reg. No. 01903.