Post-print of: Spychalski P., Arendt R.: Machine Learning in Multi-Agent Systems using Associative Arrays. PARALLEL COMPUTING. Vol. 75, (2018), p. 88-99. DOI: 10.1016/j.parco.2018.03.006.

Machine Learning in Multi-Agent Systems using Associative Arrays

Przemysław Spychalski[1], **Ryszard Arendt**[2]

Gdansk University of Technology, Faculty of Electrical and Control Engineering
1. tel.: +48 58 347 2139, e-mail: przemyslaw.spychalski@pg.edu.pl
2. tel.: +48 58 347 2157, e-mail: ryszard.arendt@pg.edu.pl

Abstract: In this paper a new machine learning algorithm for multi-agent systems is introduced. The algorithm is based on associative arrays, thus it becomes less complex and more efficient substitute of artificial neural networks and Bayesian networks, which is confirmed by performance measurements. Implementation of machine learning algorithm in multi-agent system for aided design of selected control systems allowed to improve the performance by reducing time of processing requests, that were previously acknowledged and stored in learning module. This article contains an insight into different machine learning algorithms and includes the classification of learning techniques regarding the criteria depicted by multi-agent systems. The publication is also an attempt to provide the answer for a question posted by Shoham, Powers and Grenager: „If multi-agent learning is the answer, what is the question?"

Keywords: Machine learning, Multi-agent system, Associative array, Artificial neural network, Bayesian network, Performance evaluation.

1. Introduction

Research from the past decades led that learning ability is no longer the domain only of living beings but also computer programs (such as software agents). This allowed extending the functionality of systems and devices, which nowadays appear to be intelligent and cooperate with each other. Machine learning (ML) is the subfield of computational intelligence that grew from computer science research related with pattern recognition [1]. Thanks to discussed discipline it became possible to build, among other, autonomous vehicles, voice-controlled interfaces and applications for image recognition [2]. Moreover, in certain applications learning capable software excel above the people, experts in particular fields. Proper example might be the defeat of multiple champion in "GO" game with artificial intelligence developed by Google [3]. Thus, the systems equipped with machine learning algorithms begin to manifest the creativity and even an ability to predict human decisions. There are three basic types of machine learning (which also applies to multi-agent systems): supervised, unsupervised and reinforcement learning.

Supervised learning is characterized by generalization of the problem and building more common hypothesis based on labeled training data [4]. Developed hypothesis is a subject to review by external source, which has necessary knowledge related with particular classification task. In this type of learning, a feedback regarding evaluation of the results is provided. Hence, learning objectives are known and their effects are measurable. The outcome of the learning process is correct classification of future input data based on previously learned patterns (supervised learning is widely used to effectively search the relationship, between inputs and corresponding outputs). Therefore, it improves the overall performance of multi-agent systems. The difference between supervised and reinforcement learning lies in the assumption, that in this case behavior of agent is not evaluated (supervised learning distinguishes the assessment of system outcome, not the agent's behavior). Latest applications of supervised learning in multi-agent systems concerns the generation of a strategy for service oriented architecture adaptation in changing conditions [5].

This paper introduces a new methodology to implement machine learning capability in multi-agent systems, which is based on associative arrays. The motivation of using associative arrays instead of artificial neural networks or Bayesian networks was dictated by the fact that this data structure operates on input values of any type. Artificial neural networks and Bayesian networks using floating point numbers by default, whereas agents communicates using string messages standardized by FIPA-ACL language. Thus, adjusting these methods to retrieve native values from multi-agent system requires advanced techniques (such as bag-of-words model).

However, even with that there is still a demand of preliminary training of both networks on specified data sets in order to classify patterns received from agents properly. Proposed machine learning algorithm operates on string values and allows online learning, in which new knowledge is being acquired in real-time during multi-agent system execution. The main contribution of implemented method is improvement of multi-agent systems performance and higher computational efficiency of data extraction comparing with artificial neural networks and Bayesian networks.

The next Section presents the classification of machine learning techniques in multi-agent systems from different perspectives, taking into account the criteria and directions of several research investigations. The Section 3 describes the architecture of multi-agent system that was used as a playground for performance evaluation of different machine learning algorithms. The Section 4 provides a statement of the problem, which is addressed by implementation of machine learning algorithm. Furthermore, there is also an explanation, why the already available ML methods are not suitable for direct application in such cases. The Section 5 introduces the developed machine learning algorithm based on associative arrays. The Section 6 contains performance evaluation of different ML algorithms, which is the comparison of proposed solution with state-of-the-art. The Section 6 refers to Shoham et al. [6] that advice to defend the theoretical evaluation with an experimental one, since many algorithms that meet formal criteria fail in practice, and vice versa. The Section 7 concludes the accomplished research.

2. State of the art

Machine learning in multi-agent systems is a subject of intensive research in academic environment and R&D industry. Researchers have considered this aspect in many categories. Noteworthy is the classification of multi-agent learning (MAL) taking into account the system dispersion. Learning could be carried out in centralized manner, but also could have distributed nature. Another important attribute is the number of agents that take advantage from the learning process. In some cases, only one agent can acquire benefits, but in the other several agents achieve the improvement at the same time [7].

Another classification criterion is the agents' behavior during the learning process. Agents might cooperate with each other or gaining knowledge independently [8]. The cooperative learning is divided into two major subdivisions: team learning and concurrent learning [9]. In the team learning there is single agent, which acts as a mentor to enhance the potential of an entire team. In the concurrent learning, every agent applies distinctive learning algorithms to improve only their individual capabilities. Nevertheless, both cooperative learning strategies will maximize the utility of agent groups.

Learning in multi-agent systems can be classified according to the criterion for determining, whether agents in the system are aware of their own and other agents' learning process. Crucial is also the homogeneity (or heterogeneity) of learning algorithms in multi-agent system [10]. Finally, discussed issue can be divided due to the fact of model utilization, during the learning process. Usage of the models could give the agent a priori knowledge about environment in which it will operate, as well as the information referring to the dynamics of other agents [11]. Recently the aspects of interactive and on-line multi-agent learning in noisy and constantly changing environment was taken into account [12, 13].

In multi-agent systems, the learning aspect is sophisticated, due to the fact of system dispersion. Moreover, the multiplicity and behavior of individuals involved in the learning process introduce a challenging field for deployment of machine learning algorithms. There are certain, proven methods and techniques to implement machine learning algorithms. Some of them are based on nature observations (Artificial neural networks), others on probabilistic equations (Bayesian networks), yet another on theory of decision support (Decision trees). Depending on the requirements, different solutions are applied, nevertheless they often belong to these three major groups.

Artificial neural networks (ANNs) are computational tools used for resolving and modelling of large-scale, real-world problems [14]. This method follows the assumptions analogous to the biological neural network that appears in the brains of living beings. This approach uses similar mechanisms to those, occurring in the encephalon for solve complex problems, however does not imitate operations ongoing between neurons directly. For instance, analogical connections between artificial neurons (nodes) in particular layer are established. Furthermore, all of those connections have their own weights (comparable to the synapses). In addition, the activation threshold of artificial neuron represents biological cell transition to active state. Artificial neural networks are typically composed of several layers: input layer, output layer and hidden layers (that enables deep learning). ANNs has many advantages, among them can be distinguished computational parallelism, adaptability and nonlinearity that allows better estimation of real-world problems. Recent research is focused on the application of ANNs in multi-agent systems to solve the consensus dilemma, in which neural network learning is used to approximate the uncertain nonlinear dynamics [15, 16].

Bayesian networks (BNs) are probabilistic representations of random variable's set and its conditional probability tables depicted by the Directed Acyclic Graph [17]. Bayesian networks (aka Belief networks) legibly represent the probability distribution between corresponding random variables on each node. Variables on the nodes are conditionally independent from their successors. In opposite to the decision trees, it is not possible to find the path from one variable to another, because the nodes are not connected. Bayesian networks are associated with the Naive Bayes classifier (commonly and widely applied in machine learning applications). This classifier is feature-independence oriented special type of Bayesian network, in which particular variables have no parents. The main advantage of Bayesian networks is that, they are direct representations of the world, not of reasoning processes [18]. Distributed multi-agent learning based on Bayesian networks was taken into account in recent years [19].

Comprehensive and valuable surveys regarding machine learning in multi-agent systems were introduced over the past two decades [2, 9, 20]. In the last-mentioned reference, the authors accurately noticed that MAL is frequently discussed in the context of rather simple applications, usually in toy-world scenarios. Commonly, a significant MAL aspect is discussed on very simplified systems, consisting of only two agents [6]. There is the lack of practical applications for this mechanism in real-world scenarios. This motivated the implementation of machine learning algorithm in an agent-based system for aided design.

3. Multi-agent system architecture

The multi-agent system for aided design of selected control systems will be used to evaluate the performance of different ML algorithms. The system consists of independent software agents, which through realization of the unique algorithms and mutual communication solves design tasks regarding proper selection of the structure and various components of selected control systems [21].

Design process begins with gathering of design requirements from the user, relating to (inter alia): information about the control object, type of controllers, actuators, sensors, design restrictions, simulation parameters, etc. The next step involves loading the mathematical models of system components, creation of ready-made structure of the control system and simulation tests. Based on the control trajectory analysis and the model characteristics, as well as confrontation of these data with guidelines located in a knowledge base, the quality assessment of design solution is conducted [22].

After estimating the quality of the designed control system, a detailed report from the performed analysis is generated. The user can acknowledge the simulation results and eventually correct the provided design requirements and assumptions. The diagram of distributed multi-agent system for aided design of selected control systems is presented in the Figure 1.

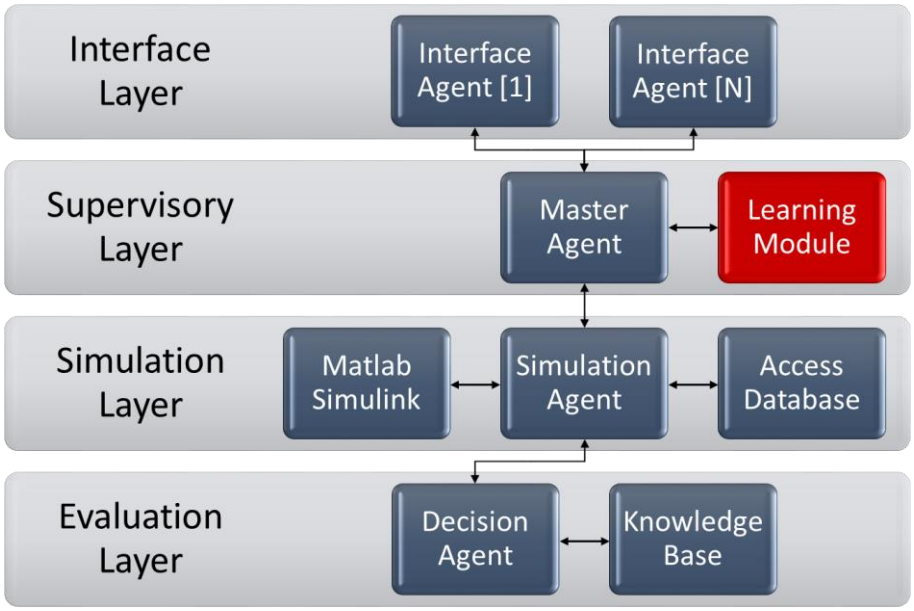

Fig. 1. Architecture of multi-agent system for aided design of selected control systems

The multi-agent system consists of four agent types: user interface agent (Interface Agent), supervisory agent (Master Agent), simulation tests agent (Simulation Agent), design evaluation agent (Decision Agent). Additionally, agents utilize the information from database, knowledge base and learning module. Invoked instances of particular agents and external tools used by these instances creates logical layers.

Interaction between agents inside each layer practically does not occurs (only the inter layers cooperation is applied). Further part of the article describes an attributes of each component of multi-agent system and its role in the design process. The inter layers cooperation during requests processing is described as well.

3.1 Interface Agent

The main purpose of the Interface Agent is gathering of requirements, assumptions and restrictions related to the design of control system. The interface gathers those data by generating the questions, corresponding with user's expertise. The answer possibilities are diversified: starting with an indication of the preferred control system structure, by specifying the numerical parameters and simulation tests duration, ending with percentage range that determines the permissible deviations from the desired target value. The main menu of the interface including questions divided into several basic thematic groups, for instance:

- Object: Allows to specify the entity, for which the control system is being implemented;
- Components: Here the user could choose the elements included in the control system;
- Structure: This category allows to select the preferred structure of the control system;
- Simulation: Enables the user to define the parameters and execution time of simulation tests;
- Database: Launches an application that allows the database access via ODBC to adjust the parameters of system component models.

After collecting all necessary design requirements, occurs their insertion into the table that contains two columns (parameter and value). Then, the information is transferred to the supervisory agent (Master Agent).

3.2 Master Agent

Master Agent is responsible for managing the design process. Distributed multi-agent system could consists with more than one user interface (usually from a few to several instances). Master handles the information from all interfaces, performs an initial validation of the received data and prioritizes tasks using the implemented queuing mechanism.

The supervisory agent mediates between user interfaces and the agents responsible for the dynamic calculations (Simulation Agent), as well as static evaluation (Decision Agent). Master transmits the encapsulated data to computational agents after receiving the affirmation, concerning a readiness to the request processing (method based on the token passing).

If the Master Agent has large number of queued design requests for an analysis and responses from computational agents are negative (in case of excessive system load), then Master is obligated to invoke another instance of the Simulation Agent. This additional agent is activated in order to restore the desired system performance (invocation is done only if required hardware resources are available).

Therefore, it is not necessary to use an expropriation mechanism of system resources, since the requests are being processed in parallel (starvation-free scheduling). Master Agent has also an ability to reduce the number of computational agents, when particular Interface Agent's request did not come for an amount of time, longer than established timeout (system idle state).

3.3 Simulation Agent

Simulation Agent requires relatively large computational power comparing with other agents (quad-core CPU with >2.5GHz frequency and minimum 4GB of physical memory is recommended). Simulation Agent purpose is to carry out the dynamic calculations. After receiving design requirements from the Master Agent (which are obtained from the user of multi-agent system), Simulation Agent preliminarily analyzes this data to verify if all components necessary to build the control system are available. Thereafter, a method to run the simulation environment based on the Matlab Simulink software is invoked. Simulation Agent loads all control system components from Simulink model's base to the workspace. Afterwards, connections between blocks are automatically generated. When the complete control system model is built, an algorithm that verifies the correctness of established structure is activated. This algorithm checks whether all inputs and outputs are coupled and if there is no unattached nodes (to avoid errors during the Matlab Simulink runtime).

The models loaded into the workspace from Simulink model's base initially have default parameters. To set the target coefficients, it is necessary to read values saved by the user in database. Connection with external database is established via ODBC (Open DataBase Connectivity). ODBC is standard, which provides API (Application Programming Interface) to work with database, regardless of used application, programming language or the type of database. ODBC protocol is available natively in almost every operating system and requires no additional software installed. Therefore, waived from a strategy, to directly control the model's base from the user interface level (otherwise, it would be required to install the Matlab Simulink on every workstation, where the Interface Agent is being instantiated). In this approach, mathematical models are generics and agnostics to the type of control system. Only if the connection with database is established and parameters adjusted by the user are loaded, models' adaptation to particular task occurs.

The next step after loading all the parameters from database is to run simulation tests. User predefines the duration of these simulations in requirements. The longer the simulation is run, the results are more accurate and better reflect a real characteristics of the designed control system. However, setting too lengthy simulation timeout leads to a significant prolongation of the design process and causing delays in providing the test results (as well as unacceptable usage of system resources by single thread). Optimal simulation time depends on complexity of the designed control system and ranges from a few, up to several minutes. The final stage after completion of simulation tests is to save the results and send them to the decision-making agent in order to evaluate the simulation outcome.

3.4 Decision Agent

The main task of the Decision Agent is quality assessment of the modeled control system. Decision Agent performs static calculations that requires moderate computing capabilities and are less time-consuming compared to the dynamic simulations. Therefore, in the implemented multi-agent system, a ratio of evaluation agents to calculation agents is lower (generally, for one Decision Agent falls several Simulation Agents).

After receiving the test results from the Simulation Agent, their confrontation with the design constraints occurs. These restrictions concern, in example: the range of exceeded reference value, the maximum setting time etc. This allows to verify the usefulness of obtained design solution. Decision Agent during the assessment utilizing the rules from a knowledge base. Those rules have following syntax: IF...THEN...ELSE

Decision Agent generates the report from an analysis of the designed control system. This report contains, in addition to the quality assessment, the information regarding control system structure, components details, simulation results, and user guidelines (for instance the adjustments of controller settings or system structure modifications in order to improve the control trajectory and meet the designated quality criteria).

Generated report includes as well more general information, such as the valuation of potential costs of building the designed control system (based on vendor's catalog data) etc. At the end, a complete report is sent back to the Master Agent, which forwards it to an appropriate Interface Agent. After that, the user of multi-agent system can review a findings provided in the report (this solution strive for industry automation).

3.5 Multi-Agent Platform

Implemented multi-agent system is dispersed between multiple workstations. Computers are connected by wireless LAN (Local Area Network). To maintain the system coherence in heterogeneous hardware environment, it was necessary to select the multi-agent platform that ensures stable and efficient system management, as well as constant supervision of the interaction between agents. The system was based on the universal multi-agent platform UMAP [23]. The platform creates the containers, inside which the agents' processes are running in compliance with the specification of FIPA (Foundation for Intelligent Physical Agent). UMAP platform provides single, universal agent template, which is an abstract class. The definition of agent behavior is established by overriding four basic template methods as follows:

- `Run();`
- `OnActivate();`
- `OnDeactivate();`
- `HandleMessage(Message message);`

This solution allows to initialize the unlimited number of a given agent instances, which makes each system based on the UMAP platform very scalable. Another advantage is that the agent's data transfer process (responsible for transport of serialized messages in XML format) and execution of inner algorithms are performed asynchronously in at least two independent threads (this solution is highly efficient and capable of an immediate request processing). Agents within the UMAP platform communicates with each other using Agent Communication Language (ACL), which is standard defined by FIPA (the other common agents' language is Knowledge Query and Manipulation Language). It is noticeable that multi-agent system has higher predispositions to solve complex design challenges, comparing with centralized standalone applications. However, it has also some limitations.

4. Problem statement

The multi-agent system for aided design of selected control system has been developed with the aim of effective work of many different users simultaneously [22]. It was desirable to assure and maintain an instant request processing in the system (close to real-time). Long waiting period for the request handling from particular user is not acceptable. While the multi-agent platform and computer network does not introduce a significant latency to the design flow, there might be situation (under certain circumstances) in which the user experience will be very disadvantageous.

The total time needed to generate solution by multi-agent system depends on the duration of particular design steps. The most time-consuming are simulation tests after building the model of control system. Performing this type of calculations cause the inaccessibility of resources within particular workstation for other agents. Interval period ranging from a few to several minutes. If there is no workstation, which has the available bandwidth, then all incoming requests from the Interface Agents must be queued and wait for the release of resources to be handled.

Although it is possible to add an additional Simulation Agent in the system. However, as previously stated, this agent type due to substantial hardware requirements could be run only on high performance workstations. Moreover, even if design requirements from multiple users are identical or the same request has been processed in the past, an entire process of building the control system model and simulation tests will be conducted from the beginning. This limitation introduces a bottleneck in multi-agent system. Mentioned problem could be overcome by the use of an appropriate learning algorithm. Methods based on artificial neural networks and Bayesian networks have certain drawbacks, which makes them a sub-optimal solution in this situation.

ANNs and BNs requires an initial training on significant amounts of input data to properly classify a given patterns [14]. Hence, these learning algorithms would be inactive for certain period from the activation of multi-agent system (until the sufficient amount of training data will be gathered). In addition, there is no simple method to carry out the extraction of data (foreground) from the previously trained artificial neural network [24]. Some research indicates also that Bayesian networks performs poorly when the network was trained in standard manner [17]. Moreover, the computational cost of Bayesian network learning is relatively high. Taking into account the disadvantages of available solutions, a novel mechanism to enable machine learning was developed.

5. Machine learning using associative arrays

Essential requirements regarding implementation of machine learning algorithm in distributed multi-agent system are efficiency, simplicity and reliability. Furthermore, the learning algorithm should be autonomous and shall not require an involvement of multi-agent system users in the learning process, since the Interface Agent and learning module could operate on separate computers dispersed geographically (user may even have no access to computer where machine learning process takes place).

New machine learning algorithm is proposed, in which learning is based on historical data. An algorithm aggregates design requirements jointly with solutions generated by multi-agent system (associative learning). The purpose of the learning process is to load the previously completed report containing the control system analysis after recognition of identical design requirements, which were processed by the system in the past (without exigency to build the model and conduct simulation tests from the beginning).

Historical data is stored as associative arrays in *.dat file, created during initialization of the Master Agent instance. Associative array is an abstract data structure, which preserves the pair of variables: specific key and determined value that can be accessed after providing this key. Associative arrays are designed to collect the large amount of selected data, to which an immediate access is required. Therefore, this data structure has been applied also in database servers and Kademlia protocol [25].

Form of the key in associative array can be arbitrary: starting with ordinal numbers, through strings, ending on tuple (the most important is that shall be the unique value). In the implemented algorithm, a hash was used (item obtained after application of hash functions). This solution was chosen due to the large size of input data (string containing input data comprises of more than 3,000 characters, whereas the hash of exactly 32). To calculate the hash, a popular cryptographic algorithm MD5 was applied.

Message-Digest algorithm 5 (MD5) allows generating 128-bit hash function from data sequence of any length. This algorithm was developed in 1992 by Ron Rivest (co-founder of RSA). Hashes with 128-bit length are slightly less complex compared with, i.e. SHA-1 (which consists of 160 bits), thus their calculation, serialization and saving in the file is more straightforward. The Figure 2 presents design requirements transmitted by the Interface Agent to Master Agent in form of string table and calculated MD5 hash value for this sequence (key):

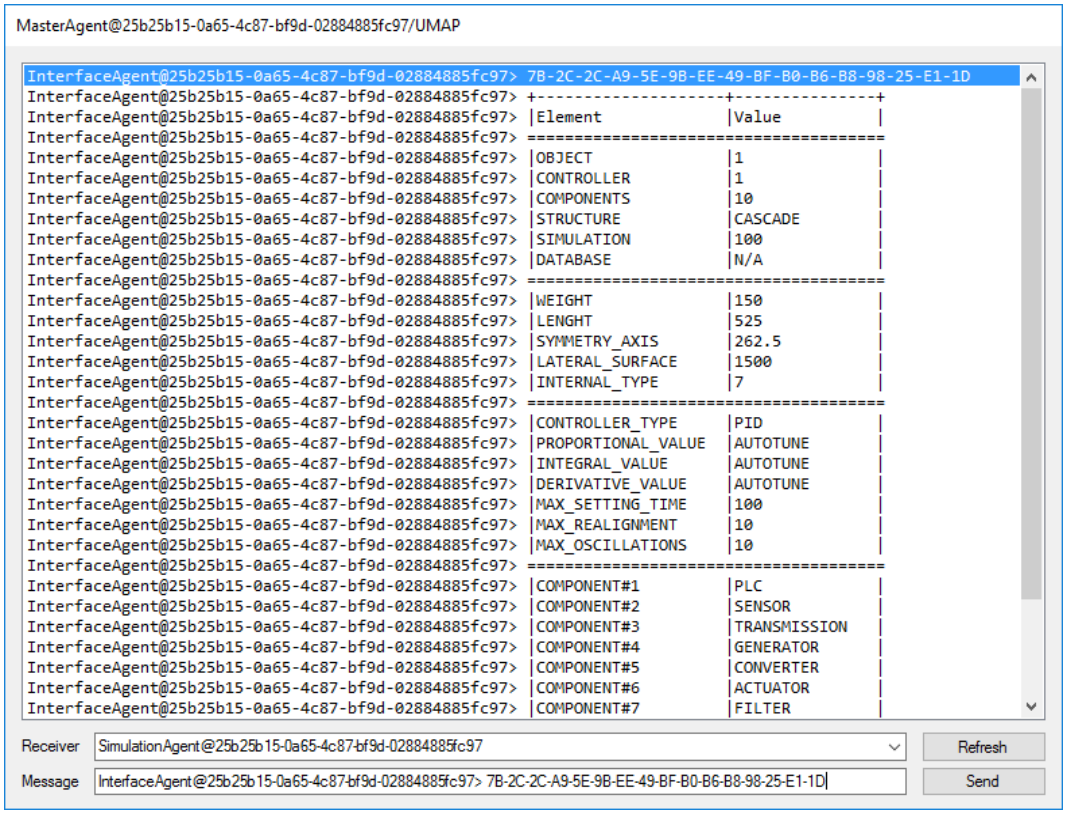

Fig. 2. Design requirements in string table form and calculated MD5 hash value for this sequence

Software module that contains machine learning algorithm based on associative arrays has been placed in the supervisory layer of multi-agent system. The main reason for deployment of this approach is the fact that through supervisory layer all requests are passed – either from interface layer that contains design requirements from multi-agent system users, as well as from computational layers (both simulation and evaluation layer) that returns generated design solutions. As a result, it is possible to supervise learning module by the Master Agent directly, relieving the system user from this obligation. Learning module is created always on the same computer, where the Master Agent is invoked. Obviously, there is a possibility to place learning module in network location, however when an access to the module occur, then *.dat file will be buffered to local machine anyway. This solution would cause unnecessary delays during learning module access. Description of individual steps in implemented machine learning algorithm and its details are provided below, whereas on the Figure 3 a flowchart of the algorithm is presented:

- Awaiting to receive the request with user assumptions from the interface layer
- Download project requirements in string table form that contains the elements and its values
- Calculation of the unique MD5 hash value (key) from downloaded string table
- Searching *.dat file that consists of the historical data to find an identical hash
- Verification, whether *.dat file contains the suitable hash key

❖ The key was found
  - Extraction of the value associated with the key, which is previously calculated design solution for a given input
  - Sending of the complete report to an appropriate Interface Agent without necessity to conduct simulation tests from the beginning

❖ The key was not found
  - Passing the design assumptions to the Simulation Agent and Decision Agent, model creation, simulation tests
  - Generation and sending the report from an analysis of the control system back to the supervisory layer

- Association of the MD5 hash calculated from project requirements with value, that contains the report concerning generated design solution
- Saving of the created associative array in *.dat file as new line
- Dispatching of the report to destination Interface Agent

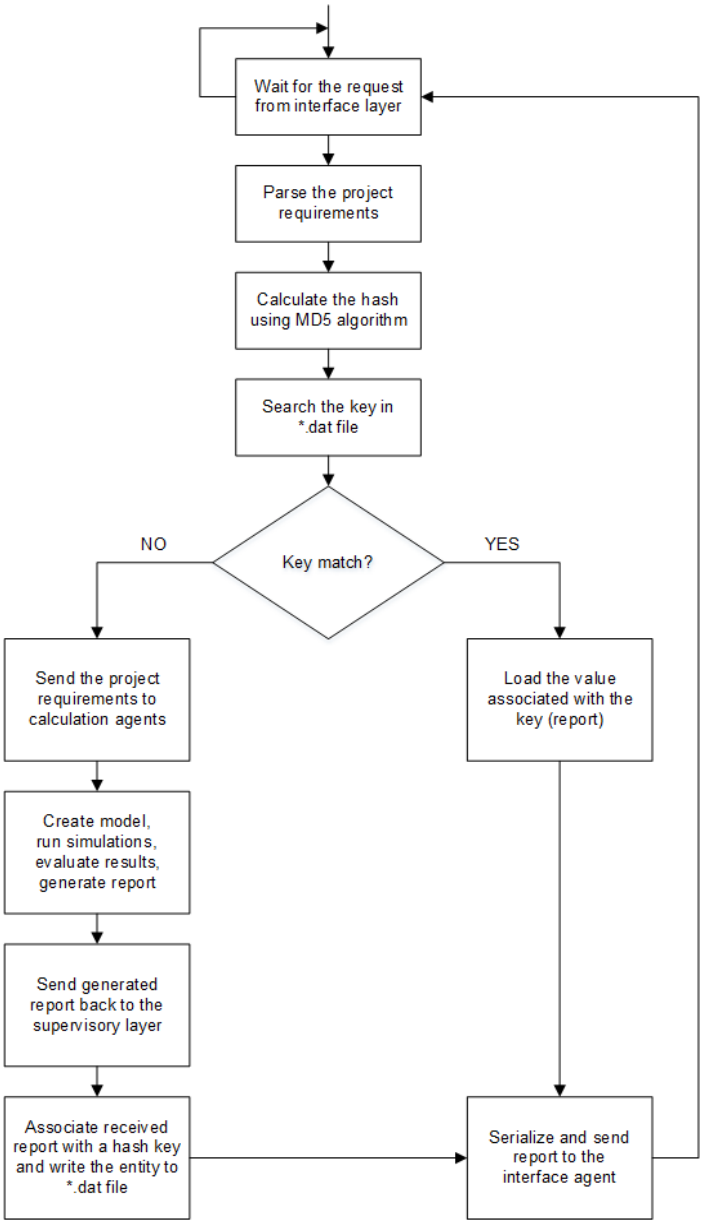

Fig. 3. Flowchart of implemented machine learning algorithm based on associative arrays

On the workstation's mass storage device, with the activation of every Master Agent instance, a new *.dat file is created (due to the obligation to catalog the associative arrays, generated during the learning process). File with *.dat extension is established in OnActivate() method, invoked during the agent initialization process. Whereas, on the agent deinitialization, OnDeactivate() method is called and the code responsible for deleting the file (linked with a given instance of the Master Agent) is executed. Lifetime of every *.dat file is convergent with the activity period of the Master Agent instance.

More than one Master Agent could be active on single computer, thus to avoid errors caused by concurrency access to the resource (when one instance read *.dat file and another wants to perform write), each Master Agent creates the separate file. All files are stored in the same location, hence in order to distinguish them, every file incorporates the different identifier, which is automatically generated GUID (Globally Unique Identifier), for instance: `7527cd69-07ff-4557-aa07-af54523245d7.dat`

Performance of machine learning algorithm is determined mostly by the search time of matching string in *.dat file, which contains stored associative arrays. For searching purposes, StreamReader() class from mscorlib.dll assembly was applied. This class allows to reading characters from the bit stream in particular encoding (the default is UTF-8). During the declaration of new StreamReader() object, the machine learning module file (hashFile) is being specified as target stream.

Afterwards, the file is scanned line by line to find the desired key (hashKey). If a given line contains the key, an entire sequence will be loaded and divided into two fragments (the key and the value). As a separator sign, the hard space was used. Selected separator sign in Unicode is marked as `\u00A0`. In the next step, after deconcatenation, a feedback message for the Interface Agent is formulated. Content of reply message includes only the second part of deconcatenated string (report). After this step, the file search is completed and the stream is being closed. The implementation of described mechanism is presented in the Algorithm 1:

```
System.IO.StreamReader sr = new System.IO.StreamReader(hashFile);

while ((line = sr.ReadLine()) != null)
{
    if (line.Contains(hashKey))
    {
        string[] report = line.Split('\u00A0');

        Message reply = message.CreateResponse();
        reply.performative = Performative.inform;
        reply.content = report[1].ToString();
        this.SendMessage(reply);

        sr.Close();
        return;
    }
    counter++;
}
sr.Close();
```

Alg. 1. Algorithm of searching the key and response creation from associated value

It is hard to estimate the percentage of times when new calls may reuse previous solutions. Depending on different course of design processes, the results are non-deterministic. At the beginning of multi-agent system activity, the number of repetitions is equal zero. However, with the increase of system usage, an accumulation of new data sets occurs. Associative arrays are persistent, thus it enhance the reusability of previously collected information with the duration of system activity up to 100%. The Section 6 contains performance comparison of proposed machine learning algorithm with ANNs and BNs.

6. Learning algorithm performance analysis

In order to verify the quality of proposed machine learning algorithm, its performance evaluation has been conducted. In the first stage of an analysis, the time required to generate design solution by multi-agent system (when learning algorithm was inactive) has been measured. In the next step assessed, how the enabling of learning algorithm will improve the overall performance of multi-agent system. Three directly consecutive requests, which contain identical design requirements (the same as presented in Figure 2) were sent. After that, the time needed to receive a response from the supervisory layer of multi-agent system has been measured.

Subsequently, the machine learning algorithm was enabled and the same measurement procedure has been carried out. To perform the tests 10 computers were involved (3x Interface Agent, 1x Master Agent, 3x Simulation Agent, 3x Decision Agent). Measurements were executed on the platforms with following hardware specification: Intel Core i7-6820HQ, DDRAM 2xDIMM 8GB, Intel SSDSCKJF180A5. Software requirements necessary to launch multi-agent system are Microsoft Windows and UMAP platform installed on each computer, as well as Matlab Simulink available on the units where Simulation Agent is invoked.

The multi-agent system was instantiated by load an assembly that including the agent classes and contain reference to an abstract class provided together with UMAP platform. This enables the identification of agents and their execution under control of UMAP container [23]. The results of multi-agent system performance evaluation (comprising the time in seconds, needed to develop design solution by the system), when an algorithm was inactive (first column), compared with active learning algorithm (second column) are presented in the Figure 4.

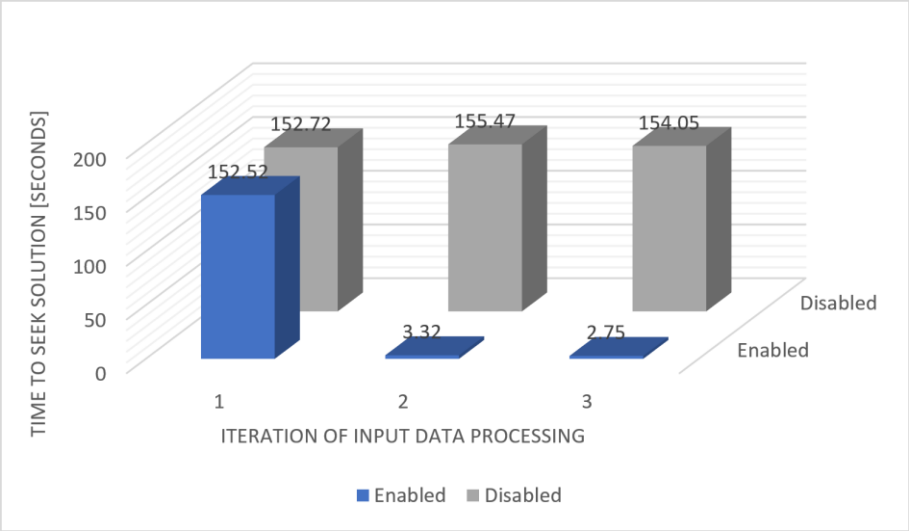

Fig. 4. Performance of multi-agent system with machine learning algorithm enabled vs disabled

Considering that execution times are non-deterministic, values from three iterations are provided. Time required to process the first pair of requests is similar to the circumstance, when machine learning algorithm was active, as well as with inactive learning algorithm. This arises from the fact that beforehand an algorithm did not acquire historical data, on which the learning process could base (the data was provided exactly in the first iteration). For the second and each subsequent requests, rapid increase of multi-agent system performance was observed. The time indispensable to generate the system report was reduced to less than a few seconds, needed to serialize the messages to XML format and transfer them between computers (on which agents were installed). Evidently, implementation of machine learning algorithm reflects on performance improvement at the level of respectively: 4682.83%; 5564.72%;

The next stage of this research involved the evaluation of implemented learning algorithm efficiency during searching a specific amount of associative arrays stored in learning module file. Measurements included time required to extract from *.dat file the value associated with obtained key and submit it to the Interface Agent. For testing purposes to assure the same initial conditions in every experiment, the data sets of historical workloads were simulated. The worst-case scenario, where searched entity is placed at the end of file, was assumed (the order of cataloged data is relevant during *.dat file searching process). Hardware parameters had a significant impact on performance results as well. The results of performance evaluation including time required to extract solution (measured in milliseconds) from various amount of associative arrays stored in *.dat file are presented in the Figure 5.

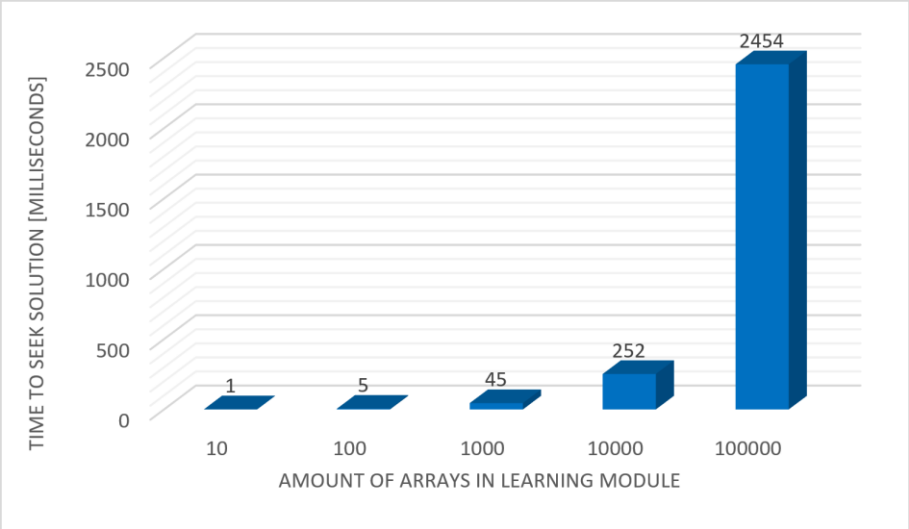

Fig. 5. Performance of machine learning algorithm during searching the various amount of associative arrays

In the final experiment, performance comparison of proposed machine learning algorithm based on associative arrays with artificial neural networks and Bayesian networks was carried out. To add support for those algorithms in learning module, the ready-made solutions were used. In case of artificial neural networks, the Encog3 was applied [26] and for Bayesian networks, the Naive Bayes test example was redesigned to solve defined classification problem [27]. By default, the training data was randomly created in runtime for both examples. The code was modified in such a way that the data sets are read from *.dat file. Moreover, the Encog3 neural network has operated on numerical values. It was necessary to translate strings using the bag-of-words model to provide this data on neural network input. The results of performance comparison including time required to extract solution (measured in milliseconds) from various amount of training data stored in *.dat file are presented in the Figure 6.

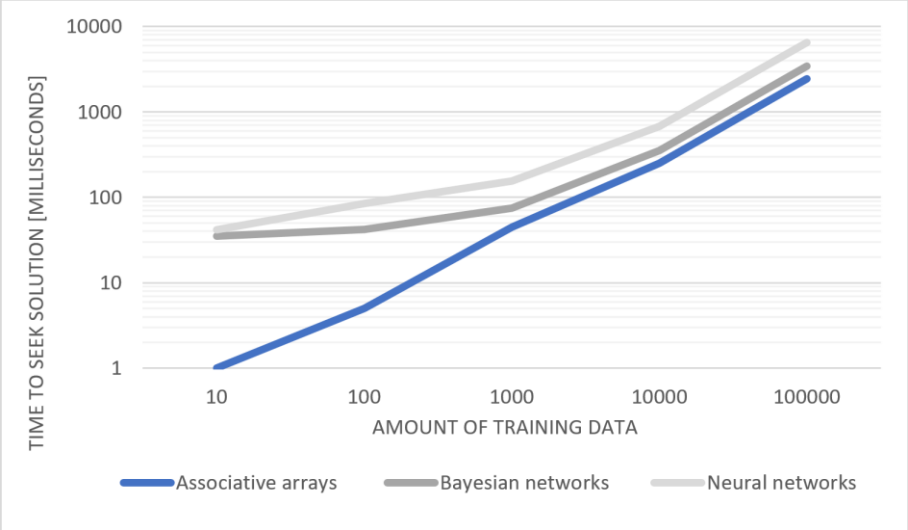

Fig. 6. Comparison of data extraction performance using different machine learning algorithms

With the increasing amount of training data, growing disparities in efficiency of machine learning algorithms can be observed. Artificial neural networks appear to be the least performant algorithm in conducted comparison. Bayesian networks are more efficient, however due to the prior training on the data sets this algorithm achieved worse performance than proposed solution based on associative arrays. For large amounts of training data, the differences are more noticeable. Nevertheless, even for smallest tested amount of the data sets, machine learning algorithm based on associative arrays was the most effective. During searching for solution within 10 entries, the time of data extraction with AAs, BNs, and ANNs was respectively: 1ms, 35ms, 42ms. The results of multi-agent system performance evaluation (comprising the time in milliseconds, needed to develop design solution by the system), when different fully trained ML algorithms were used by learning module are presented in the Figure 7.

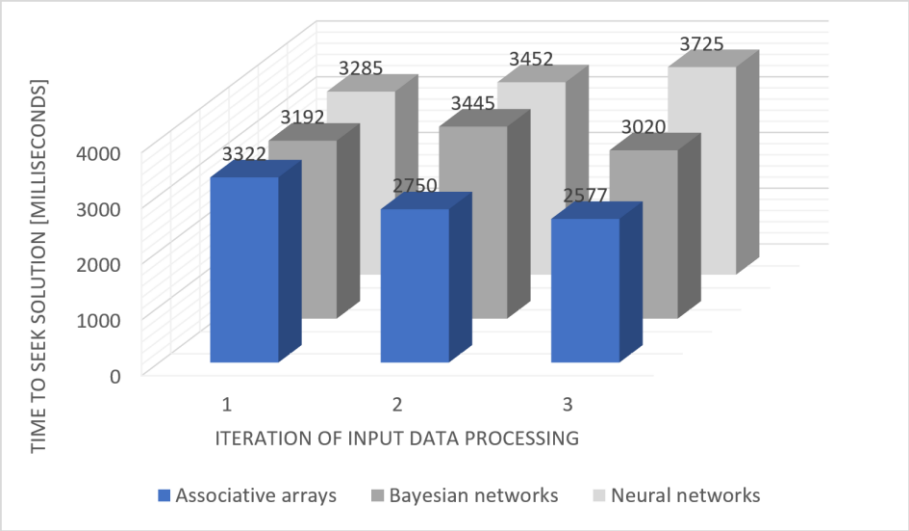

Fig. 7. Comparison of multi-agent system performance using different machine learning algorithms

7. Conclusions and further research

Proposed machine learning algorithm was based on associative arrays. An algorithm is less complex and more efficient substitute of artificial neural networks and Bayesian networks. In addition, it requires no prior learning on training data. Implementation of machine learning algorithm in multi-agent system for aided design of selected control systems allows to achieve the performance improvement in the range from 4682.83% to 5564.72% by reducing time of requests processing at the level of 150 seconds. Furthermore, the process of solution extraction from learning module does not introduce relevant overhead in multi-agent system, even for relatively large number of associative arrays.

Subsequent research will focus on further grow of algorithm performance (which directly contribute to increase the overall multi-agent system productivity). This might be achieved through improvements in algorithm flow, for instance the promotion of frequently used solutions (transferring the most desired associative arrays at the beginning of *.dat file). Additionally, splitting of the key search between multiple threads will definitely reduce time of data extraction. Another method is to synchronize associative arrays in case when the system includes more than one Master Agent (synchronization could be conducted during system idle state). That will allow rapid collection of new training data and increasing the coverage of design solutions stored in associative arrays. Improvements might also be applied for the model of supervised learning, for instance by introducing new teacher in the learning process, which will provide an additional feedback regarding the quality of design solutions (it may be the administrator, who will intervene in the form of associative arrays, i.e. deleting incorrect entities).

In conclusion authors would like to refer to the question raised by the Shoham, Powers and Grenager: „If multi-agent learning is the answer, what is the question?". An appropriate answer in this case seems to be: "How to improve the performance of multi-agent system?". However, many questions in the context of machine learning in multi-agent systems still remain without the answer, many aspects are unexplored, many mechanisms and theories requires improvement, thus at the end authors wish good luck to all researchers in their studies and deployment of multi-agent learning.

References

[1] Smola A., Vishwanathan S.V.N.: "Introduction to Machine Learning", Cambridge University Press (2008)

[2] Tuyls K., Weiss G.: "Multiagent Learning: Basics, Challenges, and Prospects", AI Magazine 33(3) (2012) 43-52

[3] Silver D., Huang A., Maddison C.J., Guez A., Sifre L. et al.: "Mastering the Game of Go with Deep Neural Networks and Tree Search", Nature 529 (2016) 484-489

[4] Kotsiantis S.B.: "Supervised Machine Learning: A Review of Classification Techniques", Informatica 31 (2007) 249-268

[5] Sniezynski B.: “Agent-based Adaptation System for Service-oriented Architectures Using Supervised Learning”, Procedia Computer Science 29 (2014) 1057-1067

[6] Shoham Y., Powers R., Grenager T.: “If Multi-agent Learning is the Answer, What is the Question?”, Artificial Intelligence 171(7) (2007) 365-377

[7] Weiss G.: “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”, MIT Press (1999)

[8] Tan M.: “Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents”, Proceedings of the 5th International Conference on Machine Learning (1993) 330-337

[9] Panait L., Luke S.: “Cooperative Multi-Agent Learning: The State of the Art”, Autonomous Agents and Multi-agent Systems 11(3) (2005) 387-434

[10] Busoniu L., Babuška R., De Schutter B.: “A Comprehensive Survey of Multi-agent Reinforcement Learning”, IEEE Transactions on Systems, Man, and Cybernetics 38(2) (2008) 156-172

[11] Yang E., Gu D.: “Multi-robot Systems with Agent-based Reinforcement Learning: Evolution, Opportunities and Challenges”, International Journal on Modelling, Identification and Control 6(4) (2009) 271-286

[12] Khalil K.M., Abdel-Aziz M., Nazmy T., Salem A-B.: "MLIMAS: A Framework for Machine Learning in Interactive Multi-agent Systems", Procedia Computer Science 65 (2015) 827-835

[13] Xu J., Tekin C., Zhang S., Shaar M.: "Distributed Multi-agent Online Learning Based on Global Feedback", IEEE Transactions on Signal Processing 63(9) (2015) 2225-2238

[14] Basheer I.A., Hajmeer M.: “Artificial Neural Networks: Fundamentals, Computing, Design, and Application”, Journal of Microbiological Methods 43(1) (2000) 3-31

[15] Zhao L., Jia Y.: "Neural Network-based Adaptive Consensus Tracking Control for Multi-agent Systems under Actuator Faults", International Journal of Systems Science 47(8) (2014) 1931-1942

[16] Wang D., Ma H., Liu D.: "Distributed Control Algorithm for Bipartite Consensus of the Nonlinear Time-delayed Multi-agent Systems with Neural Networks", Neurocomputing 174 (2016) 928-936

[17] Grossman D., Domingos P.: “Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood”, Proceedings of the 21st International Conference on Machine Learning (2004) 46-54

[18] Pearl J., Stuart R.: “Bayesian Networks”, UCLA Cognitive Systems Laboratory (2000)

[19] Djuric P.M., Wang Y.: “Distributed Bayesian Learning in Multi-agent Systems”, IEEE Signal Processing Magazine 29(2) (2012) 65-76

[20] Stone P., Veloso M.: “Multi-agent Systems: A Survey from a Machine Learning Perspective”, Autonomous Robotics 8(3) (2000) 345-383

[21] Arendt R., Spychalski P.: “An Application of Multi-Agent System for Ship’s Power Systems Design”, Proceedings of the 20th International Conference Transport Means (2016) 380-384

[22] Arendt R., Kopczynski A., Spychalski P.: “Centralized and Distributed Structures of Intelligent Systems for Aided Design of Ship Automation”, Proceedings of the 38th International Conference on Information Systems Architecture and Technology (2017)

[23] Mrozek, D., Malysiak-Mrozek B., Waligora I.: “UMAP - A Universal Multi-Agent Platform for .NET Developers”, Beyond Databases, Architectures, and Structures: Communications in Computer and Information Science 424 (2014) 300-311

[24] Andrews R., Diederich J., Tickle A.B.: “Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks”, Knowledge-Based Systems 8(6) (1995) 373-389

[25] Maymounkov P., Mazieres D.: “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric” Proceedings of the 1st International Workshop on Peer-to-Peer Systems (2002) 53-65

[26] McCaffrey J.: “Test Run - Naive Bayes Classification with C#”, MSDN Magazine 28(6) (2013)

[27] Heaton J.: “Programming Neural Networks with Encog3 in C#”, Heaton Research (2011)