Ontological Modeling of Meta Learning Multi-Agent Systems in OWL-DL

Ondřej Kazík, Roman Neruda

***Abstract*—In this article we present an unified ontological description logic model of data-mining computational multi-agent systems. The model consists of organizational role-based description, description of data-mining methods and database of experiment results. Its main purpose is support for meta learning application – choice of suitable data-mining methods for unknown data based on previous experience.**

***Index Terms*—Data mining, meta learning, roles, description logic, ontology.**

I. Introduction

Discovering the patterns in data usually requires deeper understanding of both the data and the data mining methods, in order to be able to use them with satisfactory results. The long term goal of our approach is therefore to develop meta learning systems, which would learn from their previous experience, and which would be able to give advice on what methods to use in particular situations. The multi-agent-based approach brings in many advantages to the complex task of meta learning when compared to non-agent solutions (such as WekaMetal extension of Weka data mining tool [1]). The *computational multi-agent systems*, i.e. application of agent technologies in the field of hybrid intelligence, showed to be promising by its configuration flexibility and capability of parallel computation in construction of data-mining processes. The possibility of meta learning is studied, i.e. the searching and machine learning in the space of computational methods' parameters. This is the case of data mining MAS Pikater [2].

The goal of this paper is to propose an ontology model of data-mining multi-agent system, particularly its role-based description, model of data-mining methods and a database of experiments. The current state of a MAS, together with history of previous experiments and description of implemented computational methods, will be represented in central authority of ontological agent. Other system agents will change state of the model and use OA services (e.g. matchmaking of agents and meta learning). The recommendation algorithm based on previous experiments will be implemented by means of the proposed ontology model. This will include finding of the most similar previously tested data with respect to a metadata metric and choice of a method with the best result on the data.

In the next section we summarize relevant related work, the following section contains overall description of the multi-agent system for data mining, the meta learning part of which is implemented by means of ontologies further in the paper. The section IV introduces OWL-DL ontology model of data-mining MAS and its different components, i.e. its role-based model, model of data-mining methods, and model of metadata. In section V the meta learning scenario is realized by means querying of this OWL-DL model. Section VI concludes the paper.

Ondřej Kazík is with Charles University, Faculty of Mathematics and Physics, Malostranské náměstí 25, Prague, Czech Republic, kazik.ondrej@gmail.com.

Roman Neruda is with Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, Prague, Czech Republic, roman@cs.cas.cz.

II. Related Work

In our previous work, an ontology based on the concept of role was proposed. The scenarios of pre-processing, supplementary learning, search in parameter space of computational methods were taken into account (e.g. [3], [4]). We implemented the Ontology Agent (OA) as an infrastructure for an ontology model. OA is a central authority, representation of the MAS ontology model together with OWL-DL tools which manage it. The open-world reasoning allows inferring of new facts from model axioms and assertions about current state of a system. Closed-world reasoning handles axioms of integrity constraints and does not allow state which would violate them. SPARQL engine processes specialized queries concerning the current state and inferred facts of the model. Other agents register themselves with the OA and they can communicate and query it by means of standard FIPA messages. The set of actions changing the state of ontology, which can use other agents, is specified, as well as queries services over the model. The OA thus realizes representation of the current state of MAS, correctness verification and matchmaking of agents and more abstract concepts (e.g. roles and groups).

The need of enrichment of the model by description of computational models and data-mining processes in computational MAS was reflected in [5]. Other ontological description of data mining tasks exists, such as the KDDOnto model [6], which is, however, not focused on MAS solutions.

The XML description of experiments in order to exchange results of machine-learning experiments is realized in language ExpML [7]. The description of data characteristics, i.e. metadata, in SQL-based databases with respect to the meta learning is shown in [2]. However, the XML and database-based solutions seem insufficient and too fixed for complex problems.

Our previous work with meta learning in data-mining MAS Pikater can be found e.g. in [2] and [4], where the algorithms have been implemented without integrated ontology representation — which is subject of this paper. In [8], the authors present a meta learning approach which takes into account another set of data-characteristics and propose ranking method of machine-learning classification methods based on previous experiments.

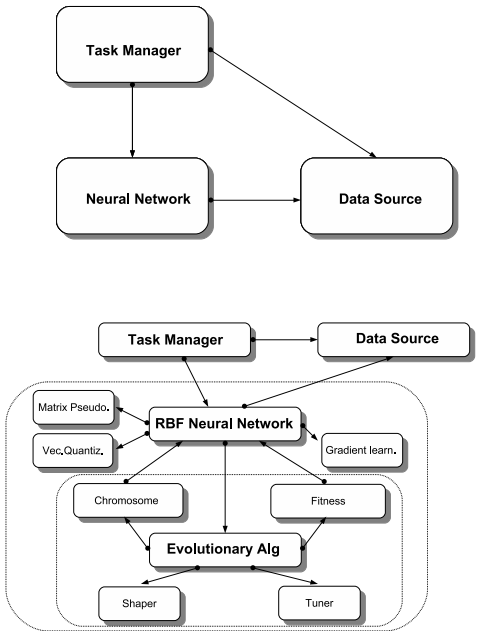

Fig. 1. Two examples of computational MAS — the simplest one (top), and the more complicated one (bottom) containing a neural network trained by an evolutionary algorithm.

III. Overall system architecture description

Hybrid models including combinations of artificial intelligence methods, such as neural networks, genetic algorithms, and fuzzy logic controllers, can be seen as complex systems with a large number of components and computational methods, and with potentially unpredictable interactions between these parts. These approaches have demonstrated better performance over individual methods in many real-world tasks [9]. The disadvantages are their bigger complexity and the need to manually set them up and tune various parameters. Also, there are not many software packages that provide a large collection of individual computational methods, as well as the possibility to connect them into hybrid schemes in various ways. Multi-agent systems seem to be a suitable solution to manage the complexity and dynamics of hybrid systems. In our approach, a computational MAS contains one or more computational agents, i.e. highly encapsulated objects embodying a particular computational intelligence method and collaborating with other autonomous agents to fulfill its goals. Several models of development of hybrid intelligent systems by means of MAS have been proposed, e.g. [10] and [11].

We will use the role-based analysis on the computational MAS scenario to create an interaction and organizational model. We are exploiting the conceptual framework of the AGR model [12]. Its organization-centered perspective allowing modular and variable construction of MAS is well suited especially to more complicated configurations of computational agents. Other formalisms, such as GAIA, would cope with dynamics of individual agents at the analysis level. We are leaving this dynamical aspect to the development of algorithms controlling individual instances of agents.

For two examples of computational MAS see Figure 1. These descriptions correspond to physical implementation of agents employing the JADE agent platform and Weka data mining library [13]. The system in our scenario consists of a Task Manager agent, Data Source agent, two computational agents (RBF neural network and Evolutionary algorithm agent) and supplementary agents. In the case of RBF network, there are unsupervised (vector quantization) and supervised (gradient, matrix inverse) learning agents. The evolutionary algorithm agent needs Fitness, Chromosome, Shaper and Tuner agents.

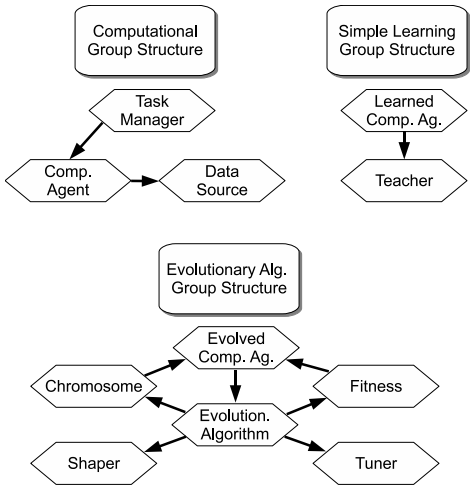

Fig. 2. The organizational structure diagram of the computational MAS

Such a computational MAS is represented by a role *organizational structure* shown at Figure 2. It consists of possible groups, their structures, described by means of admissible roles and interactions between them. This organizational structure contains the following group structures:

- *Computational Group Structure*. It contains three following roles: a Task Manager, Computational Agent implementing a computational method and Data Source which provides it with training and testing data.
- *Simple Learning Group Structure* consisting of two following roles: a Teacher and Learned Computational Agent. This structure is instantiated by three groups for each Teacher (Vector Quantization, Gradient and Matrix Inverse).
- *Evolutionary Algorithm Group Structure* contains an Evolutionary Algorithm Agent, Evolved Computational Agent, Chromosome which translates representation of an individual into the model parameters, Tuner with probabilities of the algorithm, and Shaper scaling the individual fitness.

Every concrete organization of the MAS is built with respect to the rules of the organizational structure. Aims of the agents are fulfilled by assuming of roles or establishing of groups and interactions. The agents can play different roles in different groups and even a complicated MAS can be built from these structures.

IV. Data-mining MAS Ontology

Our integrated ontological model contains the following three modules: the role model, data-mining methods ontology, and on-line model of past experiments used by meta learning algorithms. Its representation and control by

Ontology Agent (OA) – proposed as a central authority of role-based MAS model in [5] – allows us to maintain computational MAS by means of services of the OA.

The importance of organizational aspect of MASes rises with their growing complexity. In many organizational MAS frameworks, the role is a central concept. In such approaches, the modular development of MASes is allowed by its decomposition to group structures which consist of permissible roles and communication protocols between them. An agent can enter into a group and utilize a communication protocol by playing some role of the corresponding structure.

In [3] we formalized the *role model* of computational MAS from the previous section in OWL-DL in order to support its run-time management by means of automated reasoning and model querying. The model is divided into open-world axioms – which derive necessary properties from given facts – and closed-world axioms, i.e. integrity constraints – which define admissible states of the system. The superior concepts and their relations in the role model is in the Figure 3: *Agent* representing superclass of all role-concepts; concept *Group* stands for groups of agents; and *Initiator* with *Responder* modeling communication protocols. In this role-based model, the structure of general data-mining MAS has been proposed. We have included various scenarios: simple computational processing, pre-processing, ensemble methods, and methods' parameter optimization. In this article we will take the role model as a basis for general ontology of meta learning in data-mining MAS.

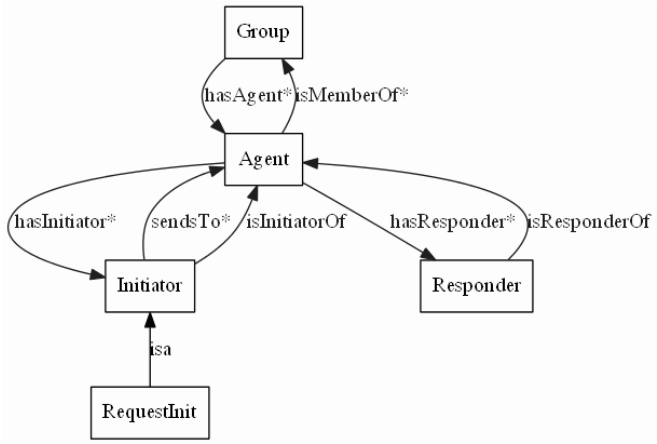

Fig. 3. Superior concepts and their relations in T-Box of the role model.

The ontology of *data-mining methods* has been proposed in [5] as a part of characterization of properties and functionalities of the computational agents in the system and particularly of the methods they implement. The computational methods (instances of general concept *CompMethod*) were described with respect to the allowed data format, task, and options in order to support computational MAS creation and hierarchy of data-mining models was constructed. This knowledge simplifies development of data-mining processes. User or automated data-mining processes construction algorithm is informed about allowed combinations of data, computational methods, and their options. All computational agents with organizational behavior determined by their role assignment are implementation of certain computational method and connected with relation *implementsMethod*. Thus, the matchmaking among agents present in the system can be constrained by properties of required computational methods. In result, the search-space of meta learning problem, as will be presented later, can be simplified.

In order to support meta learning algorithms and store the results of previous computations, the *database of experiments and their results* is necessary. We take the general XML model of experiments as was described in [2]. The fixed structure of XML document is transformed in more flexible conceptual structure in T-Box of the general ontology model of data-mining MAS. The description logic solution offers different level of description generality, possibility of next development, as well as exploitation of standard OWL-DL tools, such as automated reasoning and query engines. The model contains description of experiments with computational methods, which they were performed on; their options; description of data properties, so called metadata; and the experiments' results. The main concepts together with their relations of the meta learn ontology together with part of the data-mining ontology are shown in the Figure 4. The concept of *Experiment* contains a tested method (instance of *CompMethod* concept from the data-mining ontology), its parameters (concept *Parameters*), and a data description (i.e. metadata) which the experiment was performed on (instance of the concept *Data*). The results according to various criteria (e.g. error rate, root mean squared error etc.) are stored as instances of concept *Evaluation* together with its name and value and added to the experiment individual.

V. Meta Learning

The choice of method for previously unknown data represents a hard problem. The meta learning is a method of solving of this problem by means of previous experience. In [2], the meta learning algorithm consists of two phases. In the first step, the most similar dataset in result database is found according to a distance from the unknown data. The distance is computed as follows:

$$d(m_1, m_2) = \sum_{i=1}^{n} w_i d_i(m_1[i], m_2[i])$$

where m_1 and m_2 are the two compared metadata, n is the number of items in the metadata or data characteristics, w_i is the weight for the single metadata items and d_i distance of two values of the ith item. Metadata consist of the following items [4]: *number of attributes* that specify data characteristics, *number of instances* — number of records in a dataset, *data type* — refers to all values of all attributes in the dataset. Possible values are *integer*, *real*, *categorical*, or *multivariate*, *default task* — type of a task that is connected with the data, currently the system can solve classification and regression types of tasks, and finally *missing values*. In the second step, the method that had the lowest error rate on the selected data will be found. We are employing the above ontology model to store the results of performed experiments. With the facts stored in this way, we are performing the recommendation of a method which fits the unknown data. We find the nearest metadata according to the above mentioned distance definition and return the method with best performance on the corresponding data.

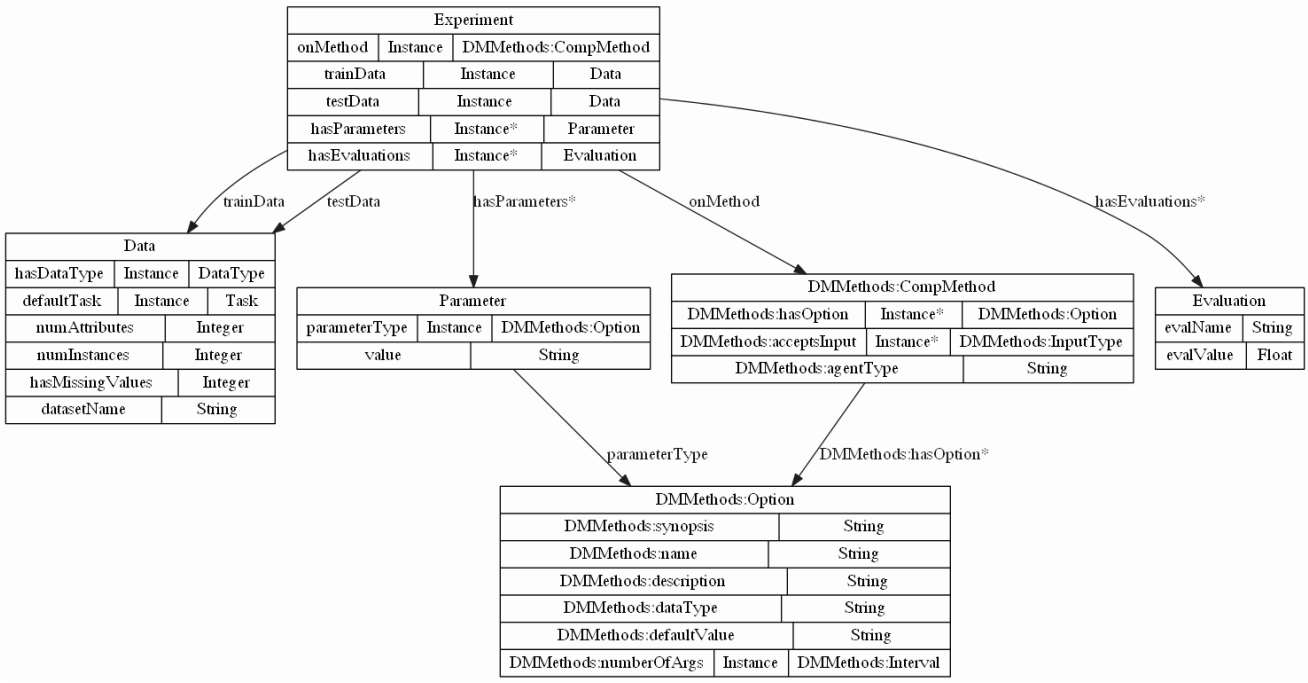

Fig. 4. The scheme of superior concepts of the experiment ontology and their relations.

Let us assume we have performed an experiment with multilayer perceptron computing agent on iris data-set [14] with a parameter set. The agent sent the results to the Ontology Agent to store it in the A-Box. The following A-Box facts describe metadata of the iris data-set:

$$TestedData(Iris)$$
$$datasetName(Iris, \texttt{"iris.arff"})$$
$$defaultTask(Iris, Classification)$$
$$hasDataType(Iris, Real)$$
$$hasMissigValues(Iris, 0)$$
$$numAttributes(Iris, 5)$$
$$numInstances(Iris, 150)$$

New instance of class Experiment together with machine-learning model parameters are created.

$$Experiment(ExpIrisMLP)$$
$$trainData(ExpIrisMLP, Iris)$$
$$testData(ExpIrisMLP, Iris)$$
$$onMethod(ExpIrisMLP, MultilayerPerceptron)$$
$$hasParameters(ExpIrisMLP, ParL)$$
$$hasParameters(ExpIrisMLP, ParM)$$
$$hasParameters(ExpIrisMLP, ParN)$$
$$hasParameters(ExpIrisMLP, ParV)$$
$$hasParameters(ExpIrisMLP, ParS)$$
$$hasParameters(ExpIrisMLP, ParE)$$
$$...$$

$$Parameter(ParL)$$
$$parameterType(ParL, OptMLPL)$$
$$value(ParL, \texttt{"0.4"})$$
$$...$$

The results of this experiment — which the computing agent sends to the Ontology Agent — contain various performance measures (e.g. error-rate, mean square error etc.). They are transformed in the following facts:

$$hasEvaluations(ExpIrisMLP, ER1)$$
$$hasEvaluations(ExpIrisMLP, KS1)$$
$$hasEvaluations(ExpIrisMLP, MAE1)$$
$$hasEvaluations(ExpIrisMLP, RMSE1)$$
$$hasEvaluations(ExpIrisMLP, RAE1)$$
$$hasEvaluations(ExpIrisMLP, RRSE1)$$

$$Evaluation(ER1)$$
$$evalName(ER1, \texttt{"error rate"})$$
$$value(ER1, 0.01999)$$
$$...$$

Previously stored results in the A-Box of the ontology can be investigated by a SPARQL language query. In order to compute the distance and sort query results according to its values, we will employ an extension contained in ARQ query engine which supports arithmetic. Equivalent query could be expressed in the SPARQL 1.1 language. The following query computes distance between the new dataset (instance of *NewData*) and all stored dataset (instances of *TestedData*).

```
SELECT  ?newdata ?mindata ?mindelta WHERE{
 ?newdata a ml:NewData.
 ?newdata ml:numAttributes ?attr1.
 ?newdata ml:numInstances ?insts1.
 ?newdata ml:hasMissingValues ?miss1.

 ?mindata a ml:TestedData.
 ?mindata ml:numAttributes ?minattr.
 ?mindata ml:numInstances ?mininsts.
 ?mindata ml:hasMissingValues ?minmiss.

 LET( ?minda := IF(
   ?attr1 >= ?minattr,
```

TABLE I
DATASETS USED IN THE EXPERIMENT AND THEIR COMPUTED DISTANCES. THE *task* TYPE IS EITHER CLASSIFICATION OR REGRESSION, *data* DENOTES THE TYPE OF DATA IN THE DATASET (CATEGORICAL, REAL, INTEGER OR MULTIPLE TYPES), *inst*, AND *attr* STANDS FOR NUMBER OF ROWS AND COLUMNS, RESPECTIVELY, *miss* SHOWS IF THE DATA CONTAINS MISSING VALUES. FINALLY, *dist* IS A DISTANCE FROM THE CONTACT-LENSES DATA.

file name	task	data	inst	attr	miss	dist
car	C	Cat	1728	7	F	0.101
magic	C	Real	19020	11	F	1.999
iris	C	Real	150	5	F	1.006
letter-recog	C	Int	20000	17	F	2.096
tic-tac-toe	C	Cat	958	10	F	0.087
weather	C	Mult	14	5	F	1.001
machine	R	Mult	209	10	F	2.050
haberman	C	Int	306	4	F	1.022
communities	R	Real	1994	128	T	4.091
lung-cancer	C	Int	32	57	T	2.420
contact-lenses	C	Cat	24	5	F	

```
   ?attr1 - ?minattr,
   ?minattr - ?attr1) )
 LET( ?mindi := IF(
   ?insts1 >= ?mininsts,
   ?insts1 - ?mininsts,
   ?mininsts - ?insts1) )
 LET( ?mindm := IF(
   ?miss1 >= ?minmiss,
   ?miss1 - ?minmiss,
   ?minmiss - ?miss1) )

 OPTIONAL{
  ?newdata ml:defaultTask ?mintask.
  ?mindata ml:defaultTask ?mintask.
 }
 LET( ?mindtask := IF(BOUND(?mintask), 0, 1))

 OPTIONAL{
  ?newdata ml:hasDataType ?mintype.
  ?mindata ml:hasDataType ?mintype.
 }
 LET( ?mindtype := IF(BOUND(?mintype), 0, 1))

 LET( ?mindelta :=
   COEFA*?minda + COEFI*?mindi + COEFM*?mindm
   + COEFTYPE*?mindtype + COEFTASK*?mindtask)
} ORDER BY ?mindelta
```

The result is a table with the unknown data individual, a previously tested data and a computed distance between them. The table is ordered according to the distance. The constants `COEFA`, `COEFI`, etc. are coefficients which are precomputed. They are a multiplication of the normalization constant (according to the range of items) and the relative weight of each item. Thus, an individual name in first row of the retrieved table specifies the closest metadata. For example, we have training set of metadata and the unknown contact-lenses dataset. The metadata items and distances to the new dataset are shown in the Table I, where we can see that the most similar dataset is the tic-tac-toe dataset.

In order to recommend a method for the unknown data we search for an experiment, where the best error rate or another performance measure was observed so far on the most similar data from the previous step. The search algorithm can be again expressed as a `SELECT` query executed by the SPARQL query engine. The retrieved best dataset is inserted as `NEAREST_DATA` and all experiments and methods performed on the data are sorted according to the error rate. The query is as follows:

```
SELECT ?experiment ?bestmethod ?errorrate
WHERE{
 ?experiment a ml:Experiment.
 ?experiment ml:testData NEAREST_DATA.
 ?experiment ml:hasEvaluations ?evaluationER.
 ?experiment ml:onMethod ?bestmethod
 ?evaluationER ml:evalName "error rate".
 ?evaluationER ml:value ?errorrate.
} ORDER BY ?errorrate
```

The recommended method, together with corresponding experiment and error rate, is again specified by the first row of the retrieved results. For the retrieved experiment, the method's parameters with which the experiment was performed can be easily obtained by $hasParameters$ relation. Thus along with the best method on the most similar data, its best setting is also recommended which can serve as a starting point for subsequent parameter space search process. Constraints regarding the method and its required properties can also be checked in the query, for example the allowed computational method's input data (e.g. some methods can work only with categorical data).

The nearest dataset from the previous step – tic-tac-toe – in our experiments performed with the best results with Weka PART method [14], i.e. C4.5 decision tree. The experiments, where the best result was obtained had parameters "U" (unpruned tree) and "M 1" (minimum number of instances per rule is one). The method after verification on the contact-lenses dataset correctly classified all instances of the dataset.

VI. CONCLUSIONS

In complex data-mining tasks, the problem of configuration of computational methods in data-mining processes arises. The multi-agent-based solution is a flexible approach to build such a distributed processes. The automated recommending of methods which are applicable to the unknown data – meta learning algorithm – improves the data-mining problem and assists users with expert information. Based on our previous work, we are proposing a general ontology model of data-mining MAS. The model contains an organization-centered role-based description of current MAS state, the description of computational methods, and a database of experiments and results. The model is represented in an ontology agent. The recommendation algorithm as a SPARQL query over the OWL-DL model is constructed and included as a meta-learning service in OA. The results of algorithm are shown in sample meta learning scenario.

The future research will be focused on the metadata metric, which can be further improved by tuning its parameters in order to get more reliable recommendations. The time requirements of the proposed queries will be studied. Improvements of the algorithms will be tested on large sets of metadata. We can utilize another methods of computational intelligence and recommendation, such as clustering methods, or decision trees, k-NN, instead of nearest neighbor (NN) which is currently used. The qualitative characteristics of methods which would restrict the search space will be taken into account. The meta learning would also handle with combinations of computational methods as a data-mining processes (e.g. pre- and post-processing, ensemble methods), instead of single machine-learning method.

Acknowledgment

Ondřej Kazík has been supported by the Charles University Grant Agency project no. 629612 and by the SVV project no. 265314. Roman Neruda has been supported by the Ministry of Education of the Czech Republic project no. ME10023.

REFERENCES

[1] M. Hall *et al.*, "The WEKA data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[2] O. Kazík, K. Pešková, M. Pilát, and R. Neruda, "Meta learning in multi-agent systems for data mining," in *Web Intelligence and Intelligent Agent Technology (WI/IAT) 2011*, 2011, pp. 433–434.

[3] O. Kazík and R. Neruda, "Role-based management and matchmaking in data-mining multi-agent systems," in *ADMI*, ser. Lecture Notes in Computer Science. Springer, 2012, pp. 95–106.

[4] O. Kazík, K. Pešková, M. Pilát, and R. Neruda, "Implementation of parameter space search for meta learning in a data-mining multi-agent system," in *ICMLA*, vol. 2. IEEE Computer Society, 2011, pp. 366–369.

[5] R. Neruda and O. Kazík, "Modeling data mining processes in computational multi-agent systems," in *MEDES '11*. ACM, 2011, pp. 61–67.

[6] C. Diamantini, D. Potena, and E. Storti, "KDDONTO: An ontology for discovery and composition of KDD algorithms," in *ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery*, 2009, pp. 13–24.

[7] J. Vanschoren, "ExpML description," http://expdb.cs.kuleuven.be/expdb/expml.php, 2008.

[8] P. B. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, pp. 251–277, 2003.

[9] P. Bonissone, "Soft computing: the convergence of emerging reasoning technologies," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pp. 6–18, 1997.

[10] Z. Zhang and C. Zhang, *Agent-Based Hybrid Intelligent Systems*. Springer, 2004.

[11] R. Neruda and G. Beuster, "Emerging hybrid computational models," in *Proc. of the ICIC 2006*, no. LNCS 4113, 2006, pp. 379–389.

[12] J. Ferber, O. Gutknecht, and M. Fabien, "From agents to organizations: An organizational view of multi-agent systems," in *AOSE 2003*, P. Giorgini *et al.*, Eds., no. LNCS 3950, 2004, pp. 214–230.

[13] R. Neruda and G. Beuster, "Toward dynamic generation of computational agents by means of logical descriptions," *International Transactions on Systems Science and Applications*, pp. 139–144, 2008.

[14] "Weka homepage," http://www.cs.waikato.ac.nz/ml/weka/.