2009 International Workshop on Agents and Data Mining Interaction

ADMI-09
(admi09.agentmining.org)

Joint with 2009 Eighth International Conference on
Autonomous Agents and Multiagent Systems
(AAMAS2009)

11 May 2009

Budapest
Hungary

Message from the Workshop Chairs

We are pleased to welcome you to attend the 2009 International Workshop on Agents and Data Mining Interaction (ADMI-09), joint with AAMAS2009. We hope you enjoy the program, and have intensive discussions on ADMI research through ADMI-09.

In recent years, Agents and Data Mining Interaction (ADMI) has emerged as a very promising research field. Following the success of ADMI-06 in Hongkong, ADMI-07 in San Jose, and ADMI-08 in Sydney, the ADMI-09 workshop in Budapest, provides a premier forum for sharing research and engineering results, as well as potential challenges and prospects encountered in the coupling between agents and data mining.

The ADMI-09 workshop encourages and promotes theoretical and applied research and development, which aims at: o exploiting agent-enriched data mining and demonstrating how intelligent agent technology can contribute to critical data mining problems in theory and practice; o improving data mining-driven agents and showing how data mining can strengthen agent intelligence in research and practical applications; o exploring the integration of agents and data mining towards a super-intelligent information processing and systems; and o identifying challenges and directions for future research on the synergy between agents and data mining.

The ten accepted papers by ADMI-09 are from seven countries. A majority of ADMI-09 submissions comes from the European countries, which forms the characteristic of ADMI-09 indicating the booming of ADMI research in Europe. The workshop also invited two speakers addressing Agents and Data Mining in Bioinformatics, and knowledge-based reinforcement learning.

Previous ADMI workshop proceedings have been published by IEEE Computer Society press. The ADMI-09 papers have been accepted for publication as an LNAI postproceedings by Springer. We appreciate Springer, in particular Mr. Alfred Hofmann, for the publication support.

ADMI-09 is sponsored by Agent-Mining Interaction and Integration Special Interest Group (AMII-SIG: www.agentmining.org). We appreciate the guideline by the Steering Committee.

More information about ADMI-09 can be found from the workshop website:
http://admi09.agentmining.org.

Finally, we appreciate the contributions made by all authors, program committee members, invited speakers, panelists, and AAMAS2009 workshop and local organizers.

March 2009

Philip S Yu
ADMI-09 General Chair
Longbing Cao, Vladimir Gorodetsky, Jiming Liu and Gerhard Weiss
ADMI-09 Workshop Co-Chairs

ADMI-09 Workshop Organizing Committee

Executive Committee

General Chair:	**Philip S. Yu** Dept. of Computer Science, University of Illinois at Chicago, USA
Workshop Co-Chairs:	**Longbing Cao** Faculty of Engineering & Information Technology, University of Technology, Sydney
	Vladimir Gorodetsky St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences
	Jiming Liu Department of Computer Science, Hong Kong Baptist University
	Gerhard Weiss Software Competence Center Hagenberg GmbH, Austria
Organizing Co-Chairs:	**Andreas Symeonidis** Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece
	Oleg Karsaev St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences
Webmaster:	**Peerapol Moemeng** Faculty of Engineering and Information Technology, University of Technology, Sydney

ADMI-09 Program Committee

Ahmed Hambaba	San Jose State University, USA)
Ajith Abraham	Norwegian University of Science and Technology, Norway
Andrzej Skowron	Institute of Decision Process Support, Poland
Boi Faltings	Artificial Intelligence Laboratory, EPFL, Switzerland
Chengqi Zhang	University of Technology, Sydney, Australia
Daniel Kudenko	University of York, UK
Daniel Zeng	Arizona University, USA
David Taniar	Monash University, Australia
Dionysis Kehagias	ITI-CERTH, Greece
Agnieszka Dardzinska	Bialystok Technical University, Poland
Eduardo Alonso	City University London, UK

Eugenio Oliveira	University of Porto, Portugal
Heikki Helin	TeliaSonera Finland Oyj, Finland
Hillol Kargupta	University of Maryland, USA
Joerg Mueller	Technische University Clausthal, Germany
John Debenham	University of Technology Sydney
Juan Carlos Cubero	University de Granada, Spain
Kazuhiro Kuwabara	Ritsumeikan University, Japan
Ken Kaneiwa	NICT, Japan
Leonid Perlovsky	AFRL/IFGA, USA
Matthias Klusch	DFKI, Germany
Mehmet Orgun	Macquarie University, Australia
Mengchu Zhou	New Jersey Institute of Technology
Michal Pechoucek	Czech Technical University, Czech Republic
Mirsad Hadzikadic	University of North Carolina, Charlotte, USA
Nathan Griffiths	University of Warwick, UK
Ngoc Thanh Nguyen	Wroclaw University of Technology, Poland
Pericles A. Mitkas	Aristotle University of Thessaloniki, Greece
Ran Wolff	Haifa University, Israel
Seunghyun Im	University of Pittsburgh at Johnstown, USA
Sung-Bae Cho	Yonsei University, Korea
Sviatoslav Braynov	University of Illinois at Springfield, USA
Valerie Camps	University Paul Sabatier, France
Vijay Raghavan	University of Louisiana, USA
Wee Keong Ng	Nanyang Technological University, Singapore
Wen-Ran Zhang	Georgia Southern University, USA
William Cheung	Hong Kong Baptist University, HK
Yanqing Zhang	Georgia State University, USA
Yasufumi TAKAMA	Tokyo Metropolitan University, Japan
Yiyu Yao	University of Regina, Canada
Yves Demazeau	CNRS, France
Zbigniew Ras	University of North Carolina, USA
Zili Zhang	Deakin University, Australia

Table of Contents

A Self-Organized Multiagent System for Intrusion Detection

E.J. Palomo, E. Domínguez, R.M. Luque, and J. Muñoz

Department of Computer Science
E.T.S.I.Informatica, University of Malaga
Campus Teatinos s/n, 29071 – Malaga, Spain
{ejpalomo,enriqued,rmluque,munozp}@lcc.uma.es

Abstract. This paper describes a multiagent system with capabilities to analyze and discover knowledge gathered from distributed agents. These enhanced capabilities are obtained through a dynamic self-organizing map and a multiagent communication system. The central administrator agent dynamically obtains information about the attacks or intrusions from the distributed agents and maintains a knowledge pool using a proposed growing self-organizing map. The approach integrates traditional mathematical and data mining techniques with a multiagent system. The proposed system is used to build an intrusion detection system (IDS) as a network security application. Finally, experimental results are presented to confirm the good performance of the proposed system.

Keywords: Multiagent system, data mining, growing self-organization, network security.

1 Introduction

Multiagent systems are a very active field to handle the dynamism in the environment due to its modularity and its flexibility. These characteristics are essentials in complex, large and distributed domains. Network security applications are provided with an enormous amount of data from which the systems must be capable to detect anomalies, intrusions or attacks. Therefore, modularity and flexibility offered by multiagent systems can be applied to network security to obtain better results.

The proposed system carries out a reasoning of the domain using local knowledge. Several multiagent systems for knowledge discovery have recently been proposed [2, 10, 12]. Multiple agents acting on different environments extract a local knowledge and communicate this information back to the central administrator agent to form a knowledge pool about the domain. In this sense, distributed agents are specialized at detecting intrusions or attacks while the central administrator agent maintains a knowledge base of the different anomalies together with the normal behavior of the system. The data in the knowledge base are clustered and analyzed to identify similarities and variations between the anomalies and normal behaviors. A proposed growing SOM model is used for clustering data in the knowledge base.

The self-organizing map (SOM) has been widely used as a unsupervised learning method for clustering high-dimensional input data and mapping these data into a two-dimensional representation space [4]. This mapping preserves the topology in terms of

distances in the representation space and provides an easy visualization of clusters on the map. One limitation of this model is its static network architecture, where topology and number of nodes have to be established in advance. This task can be difficult because a prior study of the problem domain, especially when we have vectors of high dimensionality, is needed.

The problem of determining the number of nodes of a SOM has been addressed by different SOM models. Some of them are the incremental grid growing, growing grid, growing SOM, and the hypercubical SOM. These models have in common the addition of new nodes in the neighborhood of others, which are considered as bad representatives of data mapped into them. However, the generated topology is not as flexible as possible. The growing grid adds rows or columns in a 2-D grid, so the topology is always a rectangular grid. There are wide spectrum application domains where a rectangular grid is not the most suitable topology to represent the input data. In order to make easy the identification of relations among input data and mirror its inherent structure, the map should capture the own topology of the data as faithfully as possible. The hypercubical SOM grows into more than two dimensions, but forsakes visual understanding of data. The rest of the models have a more adaptive topology but are limited to a growth in 4 directions at the boundary of the map. Moreover, the inherent hierarchical structure of data is not represented for these models.

In this paper, a growing SOM with a more flexible adaptation of its topology to represent input data is presented. This new model splits the nodes with highest representation error, taking into account their more dissimilar neighbors. Furthermore, hierarchical relations among data are provided by classifying the prototype vectors of clusters. Therefore, a two-level hierarchy of input data is mapped in a 2-D representation space.

The remainder of this paper is organized as follows. Section 2 contains a detailed description of the proposed system architecture for network security. In Section 3, the new growing SOM model for clustering data in the knowledge base is presented. Experimental results from implementing an IDS with the proposed multiagent system are provided in Section 4. Finally, Section 5 concludes this paper.

2 Overview of the Proposed Multiagent System

Network security application have complex and distributed domains with a huge amount of data from which the attacks or anomalies must be detected. Therefore, modularity and flexibility offered by multiagent systems are suitable characteristics to obtain good results.

An IDS monitors the IP packets flowing over the network to capture intrusions or anomalies, where anomalies are considered deviations from the normal behavior. Therefore, the proposed system detects anomalies or attacks by classifying IP connections into normal or anomalies connections. Moreover, the proposed system also identifies the type of known attacks. Some anomaly detection systems using data mining techniques such as clustering, support vector machines (SVM) and neural network systems have been proposed [5, 6, 9]. The artificial neural networks provide many advantages in the detection of network intrusions [1]. However, IDSs based on self-organization usually show high false positive rates [11].

The proposed multiagent architecture is composed by multiple distributed agents and a central administrator agent. Distributed agents detect anomalies or attacks. The central administrator keeps a global knowledge that is accessible by the other agents for further information about the domain. Similarly, the central administrator agent can transfer the knowledge base to the distributed agents to enhance the intrusion detection capability of individual agents.

Knowledge base maintenance using some similarity measure is essential for detecting common patterns or behaviors and for generalizing to new attacks. Therefore, the central administrator agent uses a clustering technique based on the proposed growing SOM to represent the global knowledge. The novelty of the proposed architecture (see Figure 1) is mainly with the hybridization of mathematical techniques and the proposed clustering technique in a multiagent system to implement an IDS.

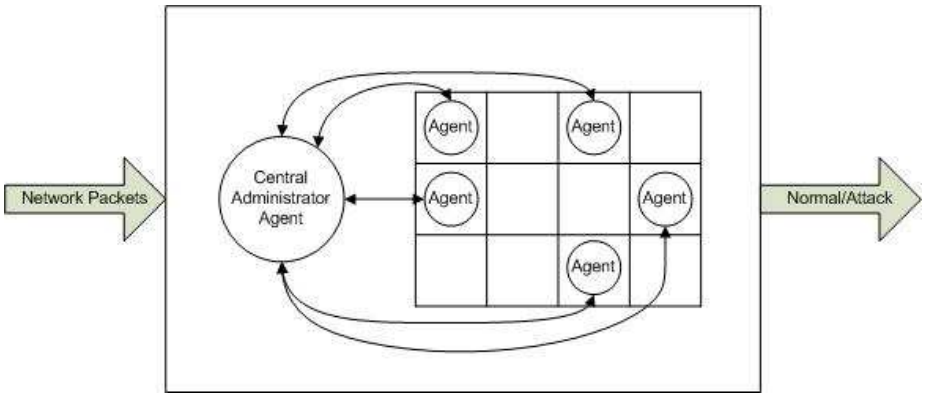


Fig. 1. IDS based on a self-organized multiagent system

3 New Growing SOM Model

The new growing SOM is a novel neural network for clustering high-dimensional input data and mapping these data into a two-dimensional representation space, where this representation space adapts its topology as faithfully as possible to input data and hierarchical relations among data are represented as groups of clusters.

This model is initialized to a SOM of 2x2 nodes, where each node represents an agent. Then, the neural network is trained through the usual feature map self-organizing process, adapting the weight vectors of nodes w_i according to input data x as the following expression

$$w_i(t+1) = w_i(t) + \alpha(t)h_i(t)[x(t) - w_i(t)] \quad (1)$$

where α is the learning rate decreasing in time, h_i is a Gaussian neighborhood function that reduces its neighborhood kernel at each iteration and t denotes the current training iteration.

Once training ends, the splitting process of the nodes of the map begins. Here, the representation quality of each node is checked. The representation quality of a node is also called quantization error (qe), which is a measure of the similarity of data mapped onto each node, where the higher is the qe, the higher is the heterogeneity of the data cluster. The quantization error of the node i is defined as follows

$$qe_i = \sum_{x_j \in C_i} \|w_i - x_j\| \tag{2}$$

where C_i is the set of patterns mapped onto the node i, x_j is the jth input pattern from C_i, and w_i is the weight vector of the node i.

The first step of the training algorithm is to compute the initial quantization error qe_0 of the map. This quantization error measures the dissimilarity of all input data and it is used for the growth process of the SOM together with the τ parameter, following the splitting condition given in (3).

$$qe_i < \tau \cdot qe_0 \tag{3}$$

This way, the quantization error of a node i (qe_i) must be smaller than a fraction τ of the initial quantization error (qe_0). Otherwise, the node is considered an error node (e), indicating that their mapped data are heterogeneous enough to be represented for more than one node. Then, these nodes are split into two nodes, so just one node is added for each error node. Thus, the parameter τ controls the growth process of the map, so the smaller the parameter the bigger the map and vice versa.

Before an error node is split, we have to decide the location where the two nodes are placed. This location is determined by their two more similar neighbors. Here, the neighbors of a node are the closest nodes in the same row, column or diagonal, so a node can have up to 8 possible neighbors depending on its position at the map (see Fig. 2(a)). This way, the indexes of the two more similar neighbors of the error node (s_1 and s_2) are chosen as follows:

$$\begin{aligned} s_1 &= \arg\min_i(\|w_e - w_i\|), \qquad w_i \in \Lambda_e \\ s_2 &= \arg\min_i(\|w_e - w_i\|), \qquad w_i \in \Lambda_e - \{w_{s_1}\} \end{aligned} \tag{4}$$

where w_e is the weight vector of the error node, w_i is the weight vector of the ith node, and Λ_e is the set of neighbor nodes of e. Note that in finding the second more similar neighbor, Λ_e is the set of neighbor nodes of e except the most similar neighbor s_1.

The new neurons are placed between the error node e and their most similar nodes s_1 and s_2. If between these nodes there is no space, a conflict is happened, and a row and/or a column is created to solve the conflict and place the new nodes. All the possible conflicts are shown in Figure 2.

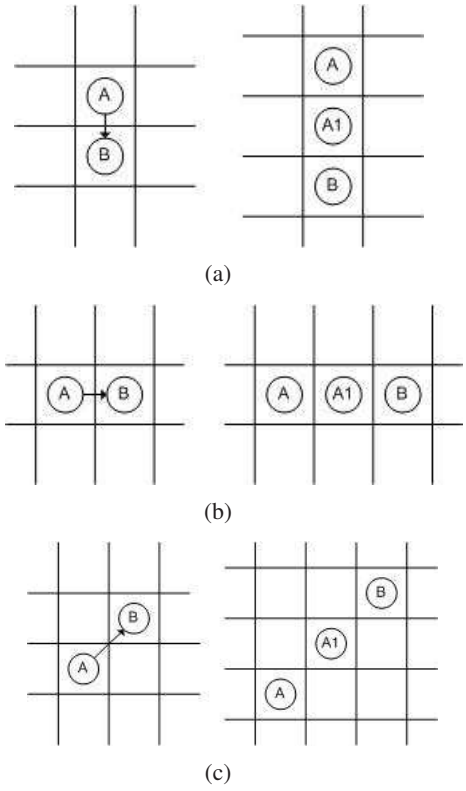


Fig. 2. Different splitting conflicts: (a) A row or (b) a column or (c) a row and a column are inserted between A and B, where A is the error node and B is one of its most similar neighbors

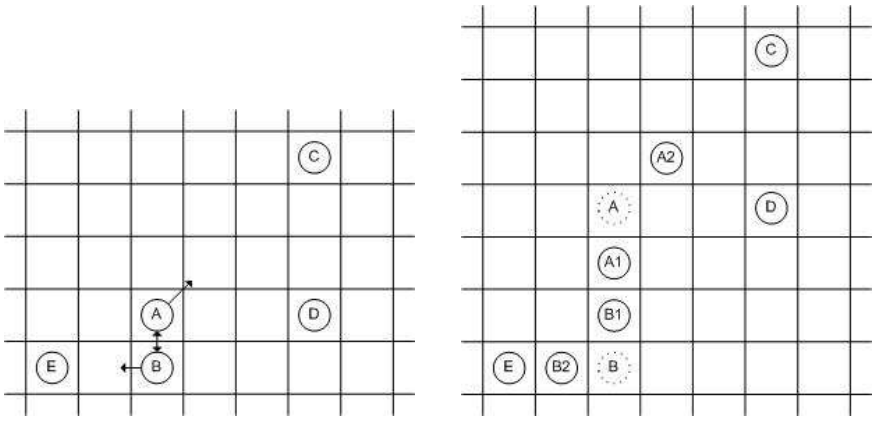


Fig. 3. An example of the splitting process with conflicts: (a) Shows a grid with 5 nodes, where A and B are the error nodes and the arrows indicate the direction of their two most similar neighbors (B and C for A, and A and E for B). Note that two row conflicts between A and B are present. (b) Shows the grid after the node splitting, where two rows has been added to solve the two row conflicts between A and B, and four new nodes have been created (A1, A2, B1 and B2). After splitting, A and B has been deleted.

Since more than one node can be divided after training, when conflicts are present we have to take into account the rest of dividing nodes in order to avoid add unnecessary rows or columns. An example of this splitting process is shown in Figure 3.

The weight vectors of the new nodes are initialized as the mean of the weight vectors of the error node and the chosen neighbor. If two error nodes have as most similar neighbor the other error node, as A and B nodes in Figure 3, the new weight vector is $2/3$ the weight vector of the error node and $1/3$ the weight vector of the neighbor. In both cases, the weight vector will be initialized as follows:

$$w_{new} = c_1 w_e + c_2 w_s, \qquad w_s \in \{s_1, s_2\} \tag{5}$$

where $c_1 = c_2 = 1/2$ if node s is not a error node, and $c_1 = 2/3$ and $c_2 = 1/3$ if node s is also a error node and one of their two most similar neighbors is e.

After the map has grown with the splitting process, the SOM is trained again and the entire process is repeated until the splitting condition (3) is satisfied for all nodes. Then, hierarchy relations must be represented in the representation space. For that reason, a clustering of clusters is performed at this stage. This clustering is performed as the usual self-organizing process, but considering weight vectors of nodes as input data. As consequence, groupings of clusters will be discovered providing information about a two-level hierarchy from input data. This is a bottom-up hierarchy discovering, where inherent hierarchical relations are represented at two levels.

4 Experimental Results

The proposed multiagent system has been proved by implementing an Intrusion Detection System (IDS). The KDD Cup 1999 benchmark data set has been used. It was created by MIT Lincoln Laboratory and contains a wide variety of instrusions simulated in a military network environment. To make easy the implementation, the 10% KDD Cup 1999 benchmark training and testing data set has been selected, which contain 494,021 and 311,029 connection records, respectively. Each connection record is composed of 41 pre-processed features from monitoring TCP packets. In the training data set exists 22 attack types in addition to normal records, which fall into four main categories [3]: DoS (denial of service), Probe, R2L (Remote-to-Local) and U2R (User-to-Root). In the testing data set 15 new attack types are found.

Both data sets contain three qualitative features in addition to quantitative features: protocol type, service and status of the connection flag. These qualitative features have to be mapped to quantitative values in order to compare two vectors with the Euclidean distance. However, qualitative values cannot be directly mapped into quantitative values since it assigns an order among qualitative values. Therefore, each qualitative feature has been replaced with new binary features (dummy features), which represent the possible values of each qualitative feature. Thus, the number of features has been increased from 41 to 118.

Training Set	Detected(%)	False Positive(%)	Identified(%)
DSS1	92.53	0	60.4
DSS2	95.05	0.69	95

Table 1. Training results for DSS1 and DSS2.

Training Set	Detected(%)	False Positive(%)	Identified(%)
DSS1	90.97	1.5	91.06
DSS2	91.07	3.53	90.52

Table 2. Testing results with 311,029 records for DSS1 and DSS2.

	Detected(%)	False Positive(%)	Nodes
GSOM MAS	90.97	1.5	19
K-Map	99.63	0.34	144
SOM	97.31	0.042	400
SOM (DoS)	99.81	0.1	28800

Table 3. Testing results for differents IDSs based on self-organization.

Two different multiagent system were organized with two data subsets selected from the 10% training data set: DSS1 and DSS2 with 500 and 169,000 connection records, respectively. Both data subsets contain the 22 attack types in addition to normal records. The distribution of the selected data mirrors the distribution of data in the 10% training data set, which has an irregular distribution. We chose 0.1 and 0.08 as values for parameters τ in both systems, generating architectures of 19 and 15 agents, respectively. Training results are shown in Table 1. Related works are usually interested in distinguish between attacks or normal records. However, we are also interested in classify an anomaly into its attack type. Thus, the detected rate are the attacks that were detected as attacks, the false positive rate are the normal connection records that were detected as attacks, and the identified rate are the connection records that were identified as their correct connection type.

After training, the two systems were tested with the 10% testing data set. Testing results are given in Table 2, where detection rates are similar but the false positive rate is lower for the first trained system. It can be owing to the high amount of training data for the second system, which yielded a more specialized system in those data, whereas the first system lernt more generally the structure patterns.

Most of the related works have used self-organizing models to implement an IDS. In [8], a hierarchical Kohonen net (K-Map) is composed of three pre-specified levels, where each level is a single K-Map or SOM. Their best result was 99.63% detection rate after testing, but taking into account several limitations. Three attack types were just used during the testing, while we used 38 attack types, where 15 attack types were unknown. Also, they used a combination of 20 features that had to be established in advance, and 48 neurons were used in each level. From [3], we chose the only one SOM trained on all the 41 features in order to compare results. This neural network achieved a detection rate of 97.31%, but using 400 neurons. An emergent SOM (ESOM) for the Intrusion Detection process was proposed in [7]. They achieved detection rates between 98.3% and 99.81% and false positives between 2.9% and 0.1%. However, the intrusion detection was just limited to DoS attacks, they used a pre-selected subset of 9 features and between 160x180 and 180x200 neurons were used. In addition, their best result (99.81% detection rate and 0.1% false positive rate) was just trained and tested with one DoS attack type, the smurf attack. A comparison between our experimental results (GSOM MultiAgent System) and the rest of related works results are summarized in Table 3.

5 Conclusions

A multiagent system based on a growing self-organizing map has been proposed in this paper. This system has capabilities to analyze and discover knowledge gathered from distribuited agents. The central administrator agent keeps a global knowledge that is accessible by the other agents for further information about the domain and controls the size of the architecture by means of clustering techniques based on the proposed growing SOM.

In order to faithfully represent input data, the proposed growing SOM provides a more flexible adaptation of its topology by adding or deleting nodes. Moreover, hierar-

chical relations among data are provided by clustering the generated nodes and showing a two-levels hierarchy of input data mapped in a 2-D representation space.

This novel multiagent system has been used to implement an Intrusion Detection System (IDS). It was trained and tested with the KDD Cup 1999 benchmark data set. After testing, we achieved a 90.97% detection rate and a false positive rate of 1.5%. These results are compared with the results from related works. Apparently these other results are better than ours, however they have several limitations in common. First, the number of features and attacks are reduced for training andor testing. Second, the model architecture has to be fixed in advance and is static. Third, hierarchical relations from data are not represented. Finally, the architecture complexity is higher due to the big amount of nodes used. All these observations indicate that a previous knowledge about problem domain is needed in these works, which is difficult in many problem domains, especially when high-dimensional data are present.

Acknowledgements

This work is partially supported by Spanish Ministry of Science and Innovation under contract TIN-07362, project name Self-Organizing Systems for Internet.

References

1. J. Cannady. Artificial neural networks for misuse detection. In anonymous, editor, *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98) October 5-8 1998. Arlington, VA.*, pages 443–456, 1998.
2. G. Czibula, A. Guran, G. Cojocar, and I. Czibula. Multiagent decision support systems based on supervised learning. *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, 3:353–358, 2008.
3. L. DeLooze and L. AF DeLooze. Attack characterization and intrusion detection using an ensemble of self-organizing maps. In *7th Annual IEEE Information Assurance Workshop*, pages 108–115, 2006.
4. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.
5. W. Lee, S. Stolfo, P. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang. Real time data mining-based intrusion detection. In *DARPA Information Survivability Conference & Exposition II*, pages 89–100 vol.1, 2001.
6. R. Maxion and K. Tan. Anomaly detection in embedded systems. *Computers, IEEE Transactions on*, 51(2):108–120, 2002.
7. A. Mitrokotsa and C. Douligeris. Detecting denial of service attacks using emergent self-organizing maps. In *5th IEEE International Symposium on Signal Processing and Information Technology*, pages 375–380, 2005.
8. S. Sarasamma, Q. Zhu, and J. Huff. Hierarchical kohonenen net for anomaly detection in network security. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 35(2):302–312, 2005.
9. K. Tan and R. Maxion. Determining the operational limits of an anomaly-based intrusion detector. *Selected Areas in Communications, IEEE Journal on*, 21(1):96–110, 2003.
10. L. Wickramasinghe and L. Alahakoon. Dynamic self organizing maps for discovery and sharing of knowledge in multi agent systems. *Web Intelli. and Agent Sys.*, 3(1):31–47, 2005.

11. H. Ying, T.-J. Feng, J.-K. Cao, X.-Q. Ding, and Y.-H. Zhou. Research on some problems in the kohonen som algorithm. In *International Conference on Machine Learning and Cybernetics*, pages 1279–1282 vol.3, 2002.
12. W.-R. Zhang and L. Zhang. A multiagent data warehousing (madwh) and multiagent data mining (madm) approach to brain modeling and neurofuzzy control. *Inf. Sci. Inf. Comput. Sci.*, 167(1-4):109–127, 2004.

Concept Learning For Achieving Personalized Ontologies: An Active Learning Approach ⋆

Murat Şensoy[1] and Pınar Yolum[2]

[1] Department of Computing Science, University of Aberdeen, AB24 3UE, Aberdeen, UK
m.sensoy@abdn.ac.uk
[2] Department of Computer Engineering, Boğaziçi University, Bebek, 34342, Istanbul,Turkey
pinar.yolum@boun.edu.tr

Abstract. In many multiagent approaches, it is usual to assume the existence of a common ontology among agents. However, in dynamic systems, the existence of such an ontology is unrealistic and its maintenance is cumbersome. Burden of maintaining a common ontology can be alleviated by enabling agents to evolve their ontologies personally. However, with different ontologies, agents are likely to run into communication problems since their vocabularies are different from each other. Therefore, to achieve personalized ontologies, agents must have a means to understand the concepts used by others. Consequently, this paper proposes an approach that enables agents to teach each other concepts from their ontologies using examples. Unlike other concept learning approaches, our approach enables the learner to elicit most informative examples interactively from the teacher. Hence, the learner participates to the learning process actively. We empirically compare the proposed approach with the previous concept learning approaches. Our experiments show that using the proposed approach, agents can learn new concepts successfully and with fewer examples.

1 Introduction

In concept learning approaches, an agent teaches another agent a concept from its ontology by providing positive and negative examples of the concept [1]. Positive examples of the concept are chosen among the instances of the concept, where as the negative examples are chosen among the non-instances. Using these examples, the learning agent tries to learn the concept. In this context, learning a concept means having the ability to correctly classify a new example as an instance or a non-instance of the concept. Once a concept is correctly learned, both the teacher and the learner have the same understanding of the concept. Previous approaches show that agents can successfully teach each other concepts from their ontologies using examples [1–3]. Although quality of learning depends on quality of the given examples, previous approaches do not specifically address the problem of how to select the most useful examples for the learner.

Selecting the most useful examples means finding a set of positive and negative examples with which the learner can successfully discriminate the concept from other

⋆ This research has been partially supported by Boğaziçi University Research Fund under grant BAP07A102 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073.

concepts. Since the focus is on discrimination from other concepts, selection of a concise set of negative examples is more important then selection of positive examples. Unlike positive examples, negative examples should be selected among a diverse set of instances that belong to various concepts. However, the number of these instances may not be bounded in many settings. For instance, while the teacher in Example 1 is teaching a *motorcycle* concept to the learner, any instance of any other concept (e.g, Sony TV, Leitz ink, and so on) are all negative examples for the *motorcycle* concept, If negative examples are not chosen intelligently, the number of negative examples required to teach a concept will be high, limiting the usability of the learning in real life.

Example 1 A seller, who has the upper ontology in Figure 1, wants to sell a motorcycle to a consumer. However, the consumer has the lower ontology in Figure 1 and does not know *Motorcycles* concept. Therefore, to convince the consumer for buying a motorcycle, the seller must firstly teach the consumer what a motorcycle is.

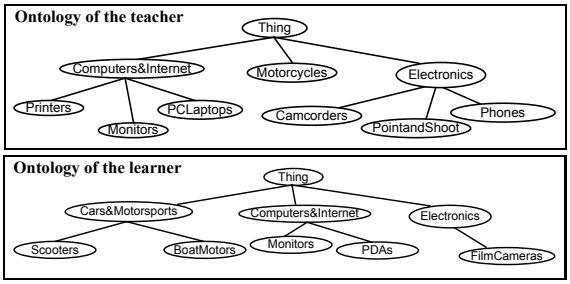


Fig. 1. Ontologies of the teacher (seller) and the learner (buyer).

In current approaches to concept learning, the learner is passive. That is, the training examples are solely chosen by the teacher. However, this assumes that the teacher has an accurate view of what the learner knows, which concepts are confusing for it, and so on. We propose to involve the learner in the learning process by enabling it to interact with the teacher to elicit the most useful examples for its understanding of the concept to be learned.The main contributions of this work are the following:

- Each agent represents its domain knowledge using an ontology and manages this ontology using a network of *experts*. In this way, as in the real life, expertise on a domain is distributed over different domain experts. An expert knows one concept in depth. New concepts are learned by the most related experts.
- Using its expertise and a *semi-supervised learning approach*, an expert first learns the new concept roughly without receiving any negative example from the teacher. Then, it elicits more informative examples from the teacher using *active learning*.
- The comparisons of the proposed approach with existing concept learning approaches (with passive learners) show that our approach can teach an agent a concept better than existing approaches with fewer examples.

2 Representing Knowledge

We consider ontologies that are also taxonomies, such that each node in the taxonomy is a specialization of its parent. Figure 2 shows a taxonomy derived from the product categories of the *Epinions* Website [10]. In machine learning, concepts are usually defined as collections of instances that share certain features [2]. Hence, we assume that there exists a set $F = \{x_1, \ldots, x_n\}$ of features. Then an instance can uniquely be characterized by its values for each of the features, $(x_1, \ldots, x_n)$, where the value of a feature x_i is either 0 or 1. This value denotes whether the instance has the feature or not. The set U denotes the set of all possible instances. Each concept C has a set of instances denoted as $I(C) \in U$, and a set of non-instances denoted as $NI(C) = U - I(C)$. In the problem of instance-based concept learning, a learner agent is given a set of positive examples $PE(C) \in I(C)$ and a set of negative examples $NE(C) \in NI(C)$. Using these examples, the learner trains a classifier to learn C. The concept is assumed to be learned if the probability of misclassification (P_{err}) is less than a threshold. Because the number of examples that can be given is limited, the total number of examples ($|PE(C)| + |NE(C)|$) should be as small as possible without compromising P_{err} much. In this paper, we assume that if two concepts C_A and C_B are sibling concepts in the ontology, then they cannot have a common instance ($I(C_A) \cap I(C_B) = \emptyset$).

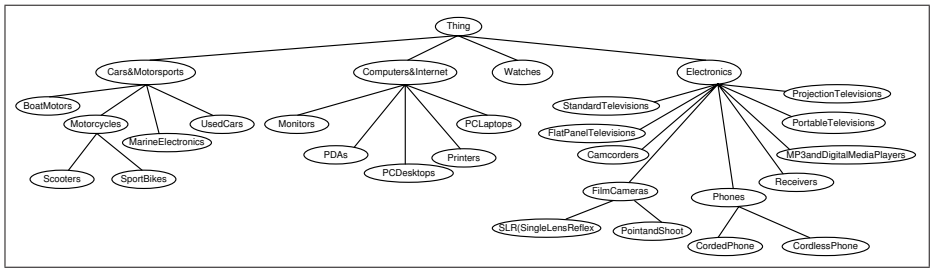


Fig. 2. An ontology from the product categories of Epinions Website

In the current instance-based concept learning approaches, one classifier is trained to learn each concept independently [1, 2]. Although the concepts are related through parent-child relationships, their classifiers are regarded as independent of one another. Therefore, if an agent has N different concepts in its ontology, there exist N independently trained classifiers for these concepts. Such approaches require each classifier to learn how to discriminate instances of one concept from those of every other concept in the ontology. Therefore, in order to learn a single concept, the agent uses the whole domain knowledge. In this paper, we envision that the domain knowledge related to an ontology is managed by a set of *experts*, each of which is knowledgeable in a certain concept. By knowledgeable in a concept, we mean that the expert can correctly report which of the concept's subclasses the object belongs to. For example, consider the ontology in Figure 2. An expert on motorcycles (denoted as $A_{Motorcycles}$) can tell us correctly that Burgman 400 is a scooter. Formally, let C be a concept with n child concepts in the concept taxonomy. Each child concept of C in the taxonomy is denoted

as C_i, where $i = \{1,\ldots,n\}$. Given that an instance $X \in I(C)$, the expert agent for C, denoted as A, can classify X as an instance of C_j with a confidence $P(C_j|C,X)$, where $j = \{0,\ldots,n\}$. C_0 denotes that the instance belong to C but not any of its child concepts in the taxonomy. Two experts are connected to each other with a communication link if there is a parent-child relationship between the concepts they represent (e.g., A_{Thing} and $A_{Electronics}$). Since each expert has different expertise, in order to train an expert, we do not have to use the whole knowledge of the domain, but we use only the instances of the concept that the agent represents. That is, an expert agent A is given examples from the C_j, where $j = \{0,\ldots,n\}$. A stores these labeled examples and uses them to train a classifier to learn child concepts of C. Given an instance, this expert can categorize the instance into one of $n+1$ classes, where each of the first n classes represents one subconcept of C and the last class represents C's instances that do not belong to any of the subconcepts.

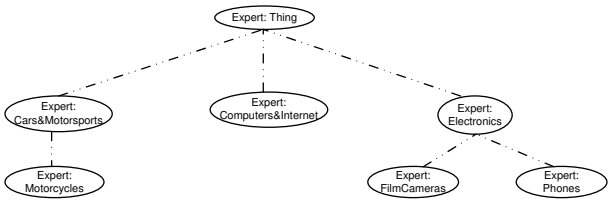


Fig. 3. Experts for the ontology in Figure 2.

Figure 3 demonstrates the experts for the ontology in Figure 2 and the communication links between them. In the figure, the expert of *Thing* concept can distinguish instances of *Car&Motorsports*, *Computers&Internet*, and *Electronics* concepts, while the expert of *Phones* concept can only distinguish instances of *CordedPhone* and *CordlessPhone* concepts. In our approach, experts must cooperate to identify whether an object is an instance of a specific concept or not, because none of these experts have complete expertise on the whole ontology and domain. Initially, the expert that represent the root concept decides whether a given object belongs to any of the subclasses of the concept it represents. If the object belongs to one of the subclasses, then the expert delegates the classification of this object to the expert that represents that subclass. This chain of delegation continues until the expert decides not to delegate the task to any other expert. Example 2 shows how the experts in Figure 3 cooperate.

Example 2 Deciding whether an object o is a CordedPhone requires the computation of *P(Corded Phone|o)*. To compute this, first A_{Thing} computes *P(Electronics|Thing,o)* using its classifier. This is the probability that o is an instance of *Electronics* given that it is an instance of *Thing*. Because every object is an instance of *Thing*, this probability is equal to $P(Electronics|o)$. Then, A_{Thing} passes o to $A_{Electronics}$ together with the computed $P(Electronics|o)$ value. $A_{Electronics}$ computes the probability $P(Phones|Electronics,o)$ and multiplies it with $P(Electronics|o)$ to compute $P(Phones|o)$. Note that $P(A,B|o)$ is equal to $P(B|o)$ if B is a child concept of A. Similarly, $A_{Electronics}$ passes o and the computed $P(Phones|o)$ to A_{Phones}. Lastly, A_{Phones} computes $P(CordedPhone|o)$ using $P(CordedPhone|Phones,o)$ and $P(Phones|o)$. In this way the object is classified in a cooperative manner.

3 Actively Learning A Concept

Consider Example 1 again, in order to learn *Motorcycles* concept from the teacher's ontology (O^t) and add it into its ontology (O^l), firstly the learner should determine the right place to put *Motorcycles* in O^l. In other words, the learner should determine the parent concept of *Motorcycles* concept in O^l. Parent concept of *Motorcycles* can be defined as the most specific concept subsuming it. For example, *Thing* concept subsumes *Phones* concept, that is $I(Phones) \subset I(Thing)$. However, *Thing* is not the parent of *Phones* concept, because there is a more specific concept *Electronics* that satisfies $I(Phones) \subset I(Electronics)$. Therefore, the parent concept is *Electronics*.

From a machine learning perspective, a concept C_B^l subsumes another concept C_A^t as much as the instances of C_A^t are classified as being an instance of C_B^l. This is actually an estimation of the probability that $I(C_A^t) \subseteq I(C_B^l)$ is true, denoted as $E[P(C_B^l|C_A^t)]$. As usually assumed in the literature, we can assume that the teacher can provide representative set of positive examples, namely $PE(Motorcycles)$. Therefore, using these examples, we can measure how much a concept C_X^l in the learner's ontology subsumes *Motorcycles*. If every motorcycle example $e \in PE(Motorcycles)$ is an instance of C_X^l, namely $P(C_X^l|e) \approx 1.0$, then we can assume that $Motorcycles \subseteq C_X^l$.

We compute $E[P(C_X^l|Motorcycles)]$ by averaging $P(C_X^l|e)$ values for motorcycle examples. Each $P(C_X^l|e)$ value is computed using a set of classifiers. Each classifier may have some uncertainty. This means that even though e is an instance of C_X^l, $P(C_X^l|e)$ may be computed smaller than 1.0. If classifiers are trained enough, this uncertainty should be very small. Therefore, we assume that classifier uncertainty does not exceed a maximum value. We call this the maximum tolerance to the classification uncertainty (ε). If C_X^l subsumes Motorcycles, then the computed $E[P(C_X^l|Motorcycles)]$ value should be greater than $1.0-\varepsilon$. Otherwise, this average should be significantly smaller than $1.0-\varepsilon$, because a portion of motorcycle examples should not be classified as an instance of C_X^l, namely $P(C_X^l|e) \approx 0$. This idea is summarized in Equation 1. In order to compute $P(C_X^l|e)$, the learner passes example e to A_{Thing}^l, which is the expert representing *Thing* concept in O^l. Then, e is classified in a cascaded manner by the experts in the hierarchy as explained in Section 2. This way, for each C_X^l and e, $P(C_X^l|e)$ is computed.

$$E\left[P(C_X|C_Y)\right] = \frac{\sum_{e \in PE(C_Y)}\left[P(C_X|e)\right]}{|PE(C_Y)|} \tag{1}$$

In our example, the learner determines that the most specific concept C_X^l satisfying $E[P(C_X^l|Motorcycles)] \geq 1.0-\varepsilon$ in its ontology is *Car&Motorsports*. As a result, the learner believes that $I(Motorcycles) \subseteq I(Car\&Motorsports)$. Confidence of this belief is the value of $E[P(Car\&Motorsports|Motorcycles)]$. The higher this value, the more confident the learner about this belief. Now, there are two possibilities, either *Car&Motorsports* is *semantically equal* or the parent of *Motorcycles* in O^l.

3.1 Determining Semantic Equivalence

Two concepts are semantically equal if their instances are exactly the same. Let C_A^t be a concept from the teacher's ontology and C_B^l be a concept from the learner's ontology.

These two concepts are semantically equal if and only if $I(C_A^t) = I(C_B^l)$. In order to believe that C_A^t and C_B^l are equal, the learner must have evidences for $I(C_A^t) \subseteq I(C_B^l)$ and $I(C_B^l) \subseteq I(C_A^t)$. The learner can easily test $E[P(C_B^l|C_A^t)] \geq 1.0 - \varepsilon$ as explained before. This is the evidence for $I(C_A^t) \subseteq I(C_B^l)$. However, testing $E[P(C_A^t|C_B^l)] \geq 1.0 - \varepsilon$ is not possible for the learner, since it does not know C_A^t yet. Therefore, the learner cooperates with the teacher. It sends representative positive examples of C_B^l to the teacher. Then, the teacher computes $E[P(C_A^t|C_B^l)]$ using Equation 1 and sends it back to the learner. The learner decides that C_B^l and C_A^t are semantically equal if $E[P(C_A^t|C_B^l)] \geq 1.0 - \varepsilon$ and then it informs the teacher about this equivalence. In this way, the learner and the teacher map concepts from their ontologies. Confidence of the semantic mapping between C_B^l and C_A^t is $E[P(C_B^l|C_A^t)] \times E[P(C_A^t|C_B^l)]$.

In our example, the learner knows that *Car&Motorsports* subsumes *Motorcycles*. If *Motorcycles* has a semantic equivalent in the ontology of the learner, this must be *Car&Motorsports*, because it is the most specific concept subsuming *Motorcycles*. Therefore, the learner needs to decide whether *Motorcycles* and *Car&Motorsports* are semantically equivalent or not. For this purpose, the learner sends the representative positive examples of *Car&Motorsports* concept in its ontology to the teacher. Then, the teacher computes the degree of subsumption as explained before (see Equation 1) and informs the learner. The learner realizes that *Motorcycles* does not subsume *Car&Motorsports*. Therefore, *Motorcycles* is added to the learner's ontology as a new child concept of *Car&Motorsports*, as explained in the next sections.

3.2 Selection of Examples

In our example, the learner decided that *Motorcycles* should be added to its ontology as a new child of *Car&Motorsports*. If *Car&Motorsports* did not have any child concept before, the learner would create a new expert to represent *Car&Motorsports* concept. However, in our example, *Car&Motorsports* is already represented by an expert ($A^l_{Cars\&Motor}$), because it already has some child concepts. For $A^l_{Cars\&Motor}$, learning *Motorcycles* means discriminating its instances from the other instances of the *Car&Motorsports* concept. For this purpose, $A^l_{Cars\&Motor}$ needs positive and negative examples of *Motorcycles*. It already has a set of positive examples given by the teacher. However, the teacher has not given any negative examples yet. At this point, the main purpose of $A^l_{Cars\&Motor}$ is to elicit the most informative negative and positive examples of *Motorcycles* from the teacher.

We know that a portion of *Car&Motorsports*' instances are not motorcycles. The useful negative examples should come from this portion. The most informative negative examples are the ones that are not obvious. Because the teacher does not have much information about the learner's ontology, it may not provide useful or informative negative examples without interacting with the learner. In this setting, $A^l_{Cars\&Motor}$ interacts with the teacher on the behalf of the learner, because it is an expert on *Car&Motorsports* and its subconcepts.

In previous works, the learners are passive and the teachers define the negative and positive examples to be used during the training of the learners. Because the negative examples are too diverse, the teachers may select the negative examples from the instances of the concepts that are most likely to be confused with the concept to be

taught. Afsharchi *et. al.* propose that teachers may give negative examples from the concepts that are similar to the concept to be taught [2], because similar concepts may be confused easily. For the computation of similarity, teachers use the distance between the concepts in their ontologies. For example, a teacher selects the negative examples mostly among the instances of the sibling concepts in its ontology [2]. If the teacher and the learner have similar ontologies, this approach sounds reasonable. However, if their ontologies are highly different, then most of the provided negative examples may be useless for the learner.

In our approach, negative examples are not directly given by the teacher, because the teacher cannot estimate which examples are more useful or informative for the learner. Therefore, initially, $A^l_{Cars\&Motor}$ tries to learn $Motorcycles$ roughly without receiving any negative examples from the teacher. For this purpose, it uses a semi-supervised learning approach. Yu *et. al.* [5] and Fung *et. al.* [6] show that, only using positive examples, it is possible to extract some negative examples from the unlabeled examples. In this paper, we follow a similar approach. In our case, $A^l_{Cars\&Motor}$ determines the most obvious negative examples of $Motorcycles$ using the positive examples and the unlabeled examples (the known instances of $Car\&Motorsports$). Some of these unlabeled examples should be negative examples of $Motorcycles$, because $Car\&Motorsports$ is not semantically equivalent to $Motorcycles$. Motorcycle instances should have some features in common that make them separate from the other instances of $Car\&Motorsports$ concept. In order to determine which features are more important for the $Motorcycles$ concept, the differences of the feature distributions between the positive examples and the unlabeled examples can be used.

3.3 Significance of Instances

If a feature discriminates motorcycle instances from the other instances of *Car & Motorsports*, this feature should be owned more frequently by the positive examples than the unlabeled examples. For example, both the positive examples of $Motorcycles$ and the unlabeled examples have an engine as a feature, because all of these examples are instances of *Car&Motorsports* concept. Therefore, the feature of having an engine is not significant for $Motorcycles$ concept. However, although every positive example of $Motorcycles$ has a *saddle*, a considerable portion of unlabeled examples does not contain that feature. This implies that, the feature of having a *saddle* is more significant for characterizing $Motorcycles$ concept and may help discriminating motorcycle examples from the rest of the unlabeled examples.

Let f^p_i be the frequency of the feature x_i in positive examples and f^u_i be the frequency of x_i in the unlabeled examples. Then, f^p_i / f^u_i defines how significant x_i is for discriminating instances of $Motorcycles$ from the other instances of *Car & Motorsports*. Significance of x_i for the positive examples is denoted as $s(x_i)$. If x_i does not appear more frequently in the positive examples, it is assumed that x_i is not significant ($s(x_i) = 0$). Otherwise, x_i is assumed to be significant as much as its relative frequency in the positive examples ($s(x_i) = f^p_i / f^u_i$).

We can estimate how significant an instance I is as a motorcycle example, using the significance of its features. For this purpose, we use Equation 2. This equation

states that an instance is significant as a motorcycle example as much as the significance of its features. After computing the significance value for each known instance of *Car&Motorsports* using Equation 2, the obvious negative examples of *Motorcycles* are chosen among the instances that have the least significance values.

$$sig(I) = \sum_{x_i \in I} s(x_i),\ where\ s(x_i) = \begin{cases} 0 & \text{if } \frac{f_i^p}{f_i^u} \leq 1.0 \\ \frac{f_i^p}{f_i^u} & \text{otherwise} \end{cases} \quad (2)$$

3.4 Putting it together

Figure 4 summarizes how the learner learns a concept from the teacher. Using the positive examples of *Motorcycles* and the obvious negative examples (line 2), $A^l_{Cars\&Motor}$ trains a classifier (line 3). This classifier can roughly discriminate instances of *Motorcycles* from other instances of *Car&Motorsports*. However, the boundary between these classes is not learned precisely, because only the obvious negative examples are used.

Moreover, some of these negative examples can be wrongly chosen. These conditions may seriously affect the performance of the trained classifier. Therefore, the learner $A^l_{Cars\&Motor}$ iteratively elicits more useful negative examples from the teacher and learns this boundary more precisely and correctly (lines 4–15). At each iteration, $A^l_{Cars\&Motor}$ samples instances of *Car&Motorsports* (line 5) and then using the classifier, it labels these sampled instances as instance of *Motorcycles* or not (line 6). Some of these instances are close to the class boundary and so they are classified with low certainty. These labeled instances are chosen to be asked to the teacher (line 7). That is, $A^l_{Cars\&Motor}$ gives the teacher a set of instances and their estimated labels (line 8). Then, the teacher returns corrected labels of the asked instances (line 8). If the teacher does not return any labeled instances, learning phase is ended and the trained classifier is returned (line 9). Otherwise, the returned labeled instances are used by $A^l_{Cars\&Motor}$ to retrain the classifier and a new iteration starts (lines 12-14). In this way, during these iterations, $A^l_{Cars\&Motor}$ elicits the most confusing negative and positive examples of *Motorcycles* for the learner.

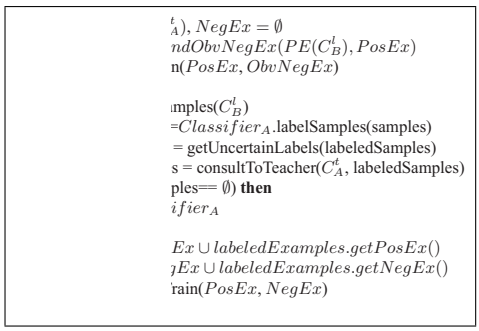


Fig. 4: Algorithm for A^l_B to learn concept C^t_A.

Using the procedure above, *Motorcycles* concept is learned correctly. Then, it is placed into the learner's ontology as a new subconcept of *Car&Motorsports*. Lastly, we test whether *Motorcycles* concept subsumes some subconcepts of *Car& Motorsports* or not. If this is the case, concept-subconcept relationships are rearranged.

4 Evaluation

In order to evaluate our approach, we conduct several experiments in online shopping domain. For this purpose, we derive domain knowledge from Epinions [10]. Epinions

is a website that contains millions of product reviews and thousands of product descriptions. Products are classified into categories. In order to represent the domain knowledge, we select a subset of these categories and construct the ontology shown in Figure 2. In our experiments, an instance refers to a product item such as *IBM ThinkPad T60*, which is an instance of *PCLaptops* concept. Each product item has a web page in Epinions website and this page contains specification of the product item in English. More than 23,000 different product items are used as instances in this work. We derive a core vocabulary from these specifications and each word in this vocabulary is used as a feature [5].

In our experiments, there is one teacher agent and one learner agent. In the implementation of the agents and the experts, we use JAVA and the C4.5 decision tree classifier of WEKA data mining project [7]. Maximum tolerance to the uncertainty in the classification (ε) is set as 0.05. In each experiment, the teacher and the learner have a different ontology that contains only a subset of the concepts in the Epinions ontology. These ontologies are constructed by randomly choosing 25% of the concepts in the Epinions ontology. Therefore, the teacher and the learner have different ontologies.

We evaluate our approach in three steps. First, we show how successful our approach is in mapping the semantically equivalent concepts in different ontologies. Second, we show how successful our approach is in incrementally learning a new concept from the teacher. Third, we compare our approach with the current approaches and show that our approach learns new concept successfully with fewer negative examples.

Mapping Equivalent Concepts. In our experiments, each concept in the learner's ontology has a semantically equivalent concept in the teacher's ontology. Therefore, the learner finds out the mappings between the concepts in its ontology and the concepts in the ontology of the teacher. Quality of concept mappings only depends on the quality of the knowledge that the teacher and the learner have about the concepts in their ontologies. This knowledge is represented by the related experts. If experts are not trained properly or they cannot combine their expertise correctly, either the learner cannot find the correct mappings or the mappings are made with low confidence. In Figure 5, we tabulate the average confidence of the mappings that are concluded by the learner in our experiments. The table shows that our approach can successfully map semantically equivalent concepts to each other with high confidence values. This implies that experts can successfully represent the domain knowledge related to their expertise and successfully combine their expertise.

Concept	Mapped Concept	Confidence
BoatMotors	BoatMotors	0.98
Camcorders	Camcorders	0.98
Cars&Motorsports	Cars&Motorsports	1.0
Computers&Internet	Computers&Internet	0.97
CordedPhone	CordedPhone	0.97
CordlessPhone	CordlessPhone	0.97
Electronics	Electronics	0.99
FilmCameras	FilmCameras	0.97
FlatPanelTelevisions	FlatPanelTelevisions	0.95
MarineElectronics	MarineElectronics	1.0
Monitors	Monitors	1.0
Motorcycles	Motorcycles	0.95
MP3andDigital...	MP3andDigital...	1.0
PCDesktops	PCDesktops	0.98
PCLaptops	PCLaptops	0.99
PDAs	PDAs	0.98
Phones	Phones	1.0
PointandShoot	PointandShoot	0.98
PortableTelevisions	PortableTelevisions	0.96
Printers	Printers	1.0
ProjectionTelevisions	ProjectionTelevisions	0.97
Receivers	Receivers	1.0
Scooters	Scooters	0.93
SLR(SingleLensReflex)	SLR(SingleLensReflex)	0.98
SportBikes	SportBikes	0.98
StandardTelevisions	StandardTelevisions	0.96
UsedCars	UsedCars	1.0
Watches	Watches	1.0

Fig. 5: Average confidence of the mappings.

Incrementally Learning New Concepts. In our approach, a new concept is learned by

an expert incrementally on behalf of the learner. At each iteration, the expert labels a set of examples using its existing knowledge about the new concept and consults the teacher. Then, the teacher instructs the expert about the correct labels of these examples. The feedback from the teacher is used to refine and improve the knowledge of the expert about the new concept. Figure 6 shows the performance of our approach at each iteration in terms of the probability of misclassification. In Figure 6, after the first iteration, the expert learns the new concepts roughly. It fails around %12 of its classifications related to this new concept. This error rate is not acceptable for the teacher, so the expert continues with the next iteration. The second iteration results in a considerable progress in the learning performance; the expert classifies more than %95 of sampled instances correctly after the second iteration. The classification error drops to zero at the fifth iteration, which means that the teacher and the learner have exactly the same understanding for this concept. This result shows that the learner has been successful in obtaining useful labeled examples from the teacher as feedback, so that the new concept is learned correctly.

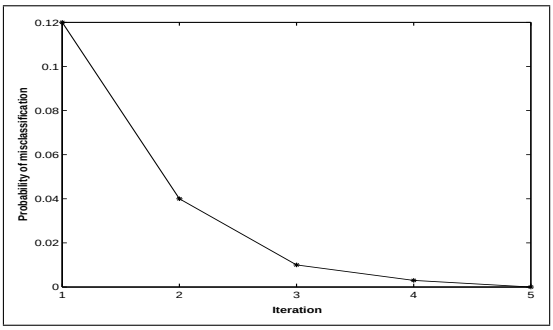


Fig. 6: Misclassification at different iterations.

Benchmark Comparisons. We compare our approach with a teacher-driven concept learning approach. This approach represents the current concept learning approaches in the literature. Contrary to the proposed approach, in those approaches, the learner is inactive during the selection of the negative examples [1, 3, 2]. The teacher selects the negative examples using its own ontology and viewpoint. Then, the learner is given positive examples and negative examples of the concept to be taught. Using these examples, the learner trains a classifier to learn the concept. As in the work of Afsharchi *et. al.* [2], in the teacher-driven approach, the teacher selects negative examples of a concept more frequently among the instances of the most similar concepts in its ontology. That is, the more similar a concept is to the concept to be taught, the more probably its instances are selected as negative examples. For the measurement of the similarity, the distance between the concepts within the teacher's ontology is used. One of the main contributions of our approach is learning new concepts with a small number of negative examples. In order to measure how successful our approach is in learning new concept for different number of negative examples, we set up experiments where the teacher is allowed to

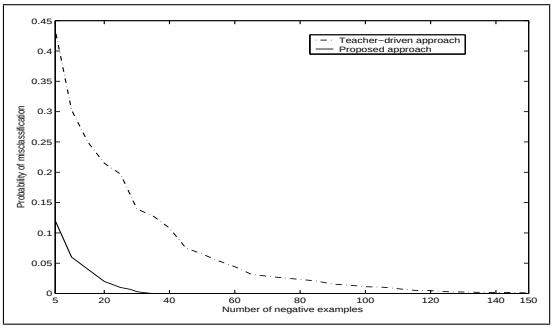


Fig. 7: Misclassification vs. number of negative examples.

give or label only a predefined number of negative examples. Then, these examples are given to the learner (as feedback in our approach). After training the learner with these examples, probability of misclassification is computed. Figure 7 compares the results for the teacher-driven approach and the proposed approach.

As seen in Figure 7, the teacher-driven approach requires more negative examples than the proposed approach in order to achieve an acceptable performance. With only five negative examples, the learner that uses the proposed approach fails only on the 12% of its classifications. However, in the same case, the learner using the teacher-driven approach misclassifies an instance with a probability of slightly higher than 0.4. Similarly, with only 35 negative examples, on the average, the proposed approach can learn a concept perfectly, while the teacher-driven approach requires approximately 150 negative examples for the same quality of learning. This result is not surprising, because our approach allows the learner to estimate some negative examples using the unlabeled data. Using these estimated negative examples, the learner can roughly draw a boundary between the instances and non-instances of a concept without receiving any negative examples from the teacher. Then, the learner iteratively elicits the most informative negative examples to specify this boundary better.

In the proposed approach, while learning a concept, the learner firstly defines the parent of the concept in its ontology. Then, learning the concept is reduced to discriminating the concept from other child concepts of its parent. This simplifies the original problem, since the concept need not to be differentiated from the other concepts in the ontology. Therefore, our approach requires smaller number of negative examples to learn the concept.

In our experiments, we notice that the learner using teacher-driven approach usually confuses the concepts like *Monitors* and *FlatPanelTelevisions*. These concepts are similar to each other in some way, but they are distant in the ontology of the teacher. Therefore, the teacher regards these concepts as dissimilar. As a result, the teacher gives fewer negative examples from *Monitors* while teaching *FlatPanelTelevisions* and vice versa. For example, in one of our experiments, while learning *FlatPanelTelevisions* concept, the learner using the teacher-driven approach confused *Monitor* and *FlatPanelTelevisions* instances, because only a small portion of the given negative examples was from *Monitors* and these examples were not enough. The learner learned to discriminate *FlatPanelTelevisions* and *Monitors* concepts only after 145 negative examples were given by the teacher and only five of these examples were from *Monitors*.

5 Discussion

This paper develops a framework for instance-based concept learning, where each agent (learner or teacher) represents its domain knowledge with a network of experts that are knowledgeable in different concepts. Using our proposed method, a learner can estimate some negative examples of the concept to be learned and obtain feedback about these negative examples from the teacher to learn the concept accurately. Our experiments show that our approach significantly outperform a teacher-driven approach that represents other instance-based concept learning approaches in the literature by enabling learners to learn a concept with few examples.

Sen and Kar [1] propose an approach for two agents to share a concept. In this approach, the teacher provides predefined positive and negative examples of the concept to be shared to the learner. The learner uses these examples to train a classifier to learn the concept. Sen and Kar show that, using the provided examples, the learner can learn the concept successfully. In Sen and Kar's approach, the learned concepts are not from ontologies and the learner is totally passive.

In order to learn new concepts in a peer-to-peer setting, Afsharchi *et. al.* [2] propose an instance-based approach. In that setting, when an agent confronts an unknown concept, it chooses teacher agents among the other agents in its neighborhood and these teachers teach the concept to the agent by providing positive and negative examples of the concept. Good negative examples are defined as the ones that are similar to the positive examples, because these negative examples are assumed to be easily misclassified. Therefore, a teacher chooses negative examples mostly among the instances of the concepts that are close to the concept to be taught in the concept hierarchy of the teacher (e.g., sibling concepts). After collecting negative and positive examples of the concept, the learner uses a machine learning approach to learn the new concept. In this approach, the learner is not incorporated in the process of choosing negative examples.

Active learning approaches enable learners to engage and involve in the learning process more [8]. In this paper, we propose an approach for concept learning in which learners elicit the most informative training examples from their teachers. To the best of our knowledge, active learning is not used for ontology evolution before. In this paper, we combine instance-based concept learning with the semi-supervised learning and active learning approaches in a novel way. Therefore, our work significantly distinguishes from the literature.

References

1. Sen, S., Kar, P.: Sharing a concept. In: Working Notes of the AAAI-02 Spring Symposium. (2002) 55–60
2. Afsharchi, M., Far, B., Denzinger, J.: Ontology guided learning to improve communication among groups of agents. In: Proceedings of AAMAS'06. (2006) 923–930
3. Doan, A., Madhaven, J., Dhamankar, R., Domingos, P., Helevy, A.: Learning to match ontologies on the semantic web. VLDB Journal (2003) 303–319
4. : Epinions web site: http://www.epinions.com (1999)
5. Fung, G.P.C., Yu, J.X., Lu, H., Yu, P.S.: Text classification without negative examples revisit. IEEE TKDE **18**(1) (2006) 6–20
6. Yu, H., Han, J., Chang, K.C.C.: PEBL: Web page classification without negative examples. IEEE TKDE **16**(1) (2004) 70–81
7. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
8. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. J. Mach. Learn. Res. **5** (2004) 255–291

The Complex Dynamics of Sponsored Search Markets*

Valentin Robu***, Han La Poutré, and Sander Bohte

CWI, Center for Mathematics and Computer Science
Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands
{robu, hlp, sbohte}@cwi.nl

Abstract. This paper provides a comprehensive study of the structure and dynamics of online advertising markets, mostly based on techniques from the emergent discipline of complex systems analysis. First, we look at how the display rank of a URL link influences its click frequency, for both sponsored search and organic search. Second, we study the market structure that emerges from these queries, especially the market share distribution of different advertisers. We show that the sponsored search market is highly concentrated, with less than 5% of all advertisers receiving over 2/3 of the clicks in the market. Furthermore, we show that both the number of ad impressions and the number of clicks follow power law distributions of approximately the same coefficient. However, we find this result does not hold when studying the same distribution of clicks per rank position, which shows considerable variance, most likely due to the way advertisers divide their budget on different keywords. Finally, we turn our attention to how such sponsored search data could be used to provide decision support tools for bidding for combinations of keywords. We provide a method to visualize keywords of interest in graphical form, as well as a method to partition these graphs to obtain desirable subsets of search terms.

1 Introduction

Sponsored search, the payment by advertisers for clicks on text-only ads displayed alongside search engine results, has become an important part of the Web. It represents the main source of revenue for large search engines, such as Google; Yahoo!; and Microsoft's Live.com, and sponsored search is receiving a rapidly increasing share of advertising budgets worldwide. But the problems that arise from sponsored search also present exciting research opportunities, for fields as diverse as economics, artificial intelligence and multi-agent systems.

In the field of multi-agent systems, researchers have been working for some time on topics such as designing automated auction bidding strategies in uncertain and competitive environments (e.g. [4, 15]). Another emergent field which studied such topic is agent-based computational economics (ACE) [12], where significant research effort has focused on the dynamics of electronic markets through agent-based simulations. One particular topic of research for the ACE community is how order and macro-level market structure can emerge from the micro-level actions of individual users. However,

*** Currently in the Intelligence, Agents, Multimedia Group, University of Southampton, UK.

* This work was performed based on a Microsoft Research "Beyond Search" award. The authors wish to thank Microsoft Research for their support.

most existing work has been based on simulations, as there are few sources of large-scale, empirical data from real-world automated markets. In this context, empirical data made available from sponsored search provides an excellent opportunity to test the assumptions made in such models in a real market.

In this paper, which is based on large-scale Microsoft sponsored search data, we provide a detailed empirical analysis of such data. To do this, we make use of several techniques derived from computational economics, and especially complex systems theory. Complex systems analysis (which we briefly review below) has been shown to be an excellent tool for analyzing large social, technological and economic systems, including web systems [13, 10, 6].

1.1 The data set

The study provided in this paper is based on a large dataset of sponsored search queries, obtained from the website Live.com[1]. The search data provided consists of two distinct data sets: a set of sponsored search dataset (URLs returned are allocated to advertisers, through an auction mechanism) and an organic search dataset (standard, unbiased web search). The sponsored search data consists of 101,171,081 distinct impressions (i.e. single displays of advertiser links, corresponding to one web query), which in total received 7,822,292 clicks. This sponsored dataset was collected for a roughly 3-month period in the autumn of 2007. The organic search data set consists of 12,251,068 queries, and was collected in a different 3-month interval in 2006 (therefore the two data sets are chronologically disjoint).

It is important to stress that in the results reported in this paper are based mostly on the sponsored search data set[2]. Furthermore, the sponsored search data we had available only provides partial information, in order to protect the privacy of Microsoft Live.com customers and business partners. For example, we have no information about financial issues, such the prices of different keywords, how much different advertisers bid for these keywords, the budgets they allocate etc. Furthermore, while the database provides an anonymized identifier for each user performing a query, this does not allow us to trace individual users for any length of time.

Nevertheless, one can extract a great deal of useful information from the data. For example, the identities of the bidders; for which keyword combinations their ads were shown (i.e. the impressions); for which of these combinations they received a click; the position their sponsored link was in when clicked etc... Insights gained from analyzing this information forms the main topic of this paper.

2 Complex systems analysis applied to the web and economics

Complex systems represents an emerging research discipline, at the intersection of diverse fields such as AI, economics, multi-agent simulations, but also physics and biology [2]. The general topic of studies in the field of complex systems is how macro-level

[1] This data was kindly provided to us by Microsoft research through "Beyond Search" award

[2] The only exception is a plot on the distribution number of clicks vs. display rank in Sect. 3, included for comparison reasons.

structure can emerge from individual, micro-level actions performed by a large number of individual agents (such as in an electronic market). For web phenomena, complex systems techniques have been successfully used before to study phenomena such as collaborative tagging [10] or the formation of online social groups [1].

One of the phenomena that are indicative to such complex dynamics is the emergence of scale-free distributions, such as power laws. The emergence of power laws in such a system usually indicates that some sort of complex feedback phenomena (e.g. such as a preferential attachement phenomena) is at work. This is usually one of the criteria used for describing the system as "complex" [2, 6]. Research in disciplines such as econophysics and computational economics discusses how such power laws can emerge in large-scale economic systems (see [6, 13] for a detailed discussion).

2.1 Power laws: definition

A ***power law*** is a relationship between two scalar quantities x and y of the form:

$$y = cx^{\alpha} \tag{1}$$

where α and c are constants characterizing the given power law. Eq. 1 can also be written as:

$$\log y = \alpha \log x + \log c \tag{2}$$

When written in this form, a fundamental property of power laws becomes apparent; when plotted in log-log space, power laws appear as straight lines. As shown by Newman [13] and others, the main parameter that characterizes a power law is its slope parameter α. (On a log-log scale, the constant parameter c only gives the "vertical shift" of the distribution with respect to the y-axis.). Vertical shift can vary significantly between different phenomena measured (in this case, click distributions), which otherwise follow the same dynamics. Furthermore, since the logarithm is applied to both sides of the equation, the size of the parameter α does not depend on the basis chosen for the logarithm (although the shifting constant c is affected). In the log-log plots shown in this paper, we have chosen the basis of the logarithm to be 2, since we found graphs with this low basis the more graphically intuitive. But, in principle, the same conclusions should hold if we choose the logarithm basis to be, e.g. e or 10.

3 Influence of display rank on clicking behavior

The first issue that we studied (for both sponsored and organic search data) is how the position that a URL link is displayed in influences its chances of receiving a click. Note that this particular issue has received much attention in existing literature [8]. To briefly explain, Microsoft's Live.com search interface (from which the data was collected), is structured as follows:

- For sponsored search there are up to 8 available slots (positions) in which sponsored URL links can be placed. Three of these positions (ranked as 1-3) appear at the top of the page, above the organic search results, but delimited from those by a different background. In addition, the page can display up to 5 additional links in a side bar at the right of the page.

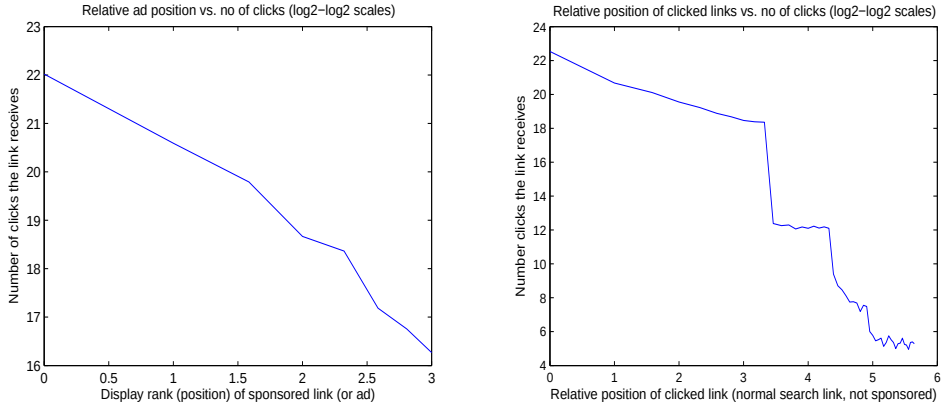


Fig. 1. Distribution of clicks received by a URL, relative to its position on the display, for sponsored and organic search. A(left-side, sponsored search): There are up to 8 sponsored advertiser links displayed: 3 on the top of the page, and 5 in a side bar. B(right, organic search): There are usually 10 positions displayed per page, with multiple result pages appearing as plateaus.

– The "organic" search results are usually returned as 10 URL links/page (a user can opt to change this setting, but very few actually do).

All the sponsored links are allocated based on an auction-like mechanism between the set of interested advertisers (such a display, in any position is called in "impression"). However, the advertisers only pay if their link actually gets clicked - i.e. "pay per click" model. The exact algorithm used by the engine to determine the winners and which advertiser gets which position is a complex mechanism design problem and not all details are made public. However, in general, it depends on such factors as the price the bidder is willing to pay per click, the relevance of the query to her set of terms, and her past performance in terms of "clickthrough rate" (i.e. how often links of that user were clicked in the past, for a given keyword). By contrast, in organic search, returned results are ranked simply based on relevance to the user's query.

3.1 Results on display position bias and interpretation

Results for the position bias on click distribution are plotted in Figure 1: part A (left side) for sponsored search and part B (right side) for the organic search. Note that both of these are cumulative distributions: they were obtained by adding the number of clicks for a link in each position, irrespective of the exact context of the queries or links that generated them. Furthermore, both are drawn in the log-log space.

There are two main conclusions to be drawn from these pictures. For the sponsored search results (Figure 1.A), the distribution across the 8 slots seems to resemble a straight line, with a slope parameter aprox. $\alpha = 2$. However, such a conclusion would be too simplistic: there is, in fact, a difference between the slope between the first 3

positions (up to $log_2 3$, on the horizontal axis), and the last 5 positions. The slope for the first 3 positions is around $\alpha_1 = 1.4$, while for the last 5 is around $\alpha_2 = 2.5$. The most likely reason for this drop comes from the way the Live.com search interface is designed. The first 3 slots for sponsored search links are shown on the top of the page, above the organic search results, while the last 5 are shown in a side bar on the right of the page.

Figure 1.B corresponds to the same plot for organic search results, the main effect one notices is the presence of several levels (thresholds), corresponding to clicks on different search pages. We stress that, since this is a log-log plot, the drop in attention between subsequent search pages is indeed very large - about two orders of magnitude (i.e. the top-ranked link on the second search page is, on average, about 65 times less likely to be clicked than the last-ranked link on the first page). The distribution of intra-page clicks, however, at least for the first page of results, could be roughly approximated by a power law of coefficient $\alpha = 1.25$.

All this raises of course the question: what do these distributions mean and what kind of user behaviour could account for the emergence of such distributions in sponsored search results? First, we should point out that the fact that we find power law distributions in this context is not completely surprising. Such distributions have been observed in many web and social phenomena (to give just one example, in collaborative tagging systems, in the work by one of the co-authors of this paper [10] and others). In fact, any model of "top to bottom" probabilistic attention behaviour, such as a user scanning the list of results from top to bottom and leaving the site with a certain probability by clicking one of them could give rise to such a distribution. Of course, more fine-grained models of user behavior are needed to explaining click behavior in this context (an example of such a model is [8]). But for now we leave this issue to further research, and we look at the main topic of this paper which is examining the structure of the sponsored search market itself.

4 Market structure at the advertiser level

In this Section, we look at how sponsored search markets are structured, from the perspective of the participants (i.e. advertisers that buy search slots for their URLs). More specifically, we study how relative market shares are distributed across link-based advertisers. We note that in many markets, an often cited rule, also informally attributed to Pareto, is that 20% of participants in a market (e.g. customers in a marketplace) drive 80% of the activity. Here, we call this effect the "market concentration".

In a sponsored search market, the main "commodity" which produces value for market participants (either advertisers and the search engine) is the number of clicks. Therefore, the first thing that we plotted (first, using normal, i.e. non-logarithmic axes) is the cumulative share of different advertisers (see Figure 2. A. - left side graph). From this graph, one can already see that just the top 500 advertisers get roughly 66% (or about two-thirds) of the total 7.8 million clicks in the available data set[3].

[3] Note that an advertiser was taken, following the available data, by the domain URL of the sponsored link. This is a reasonable assumption, in this case. For example, Ebay uses many

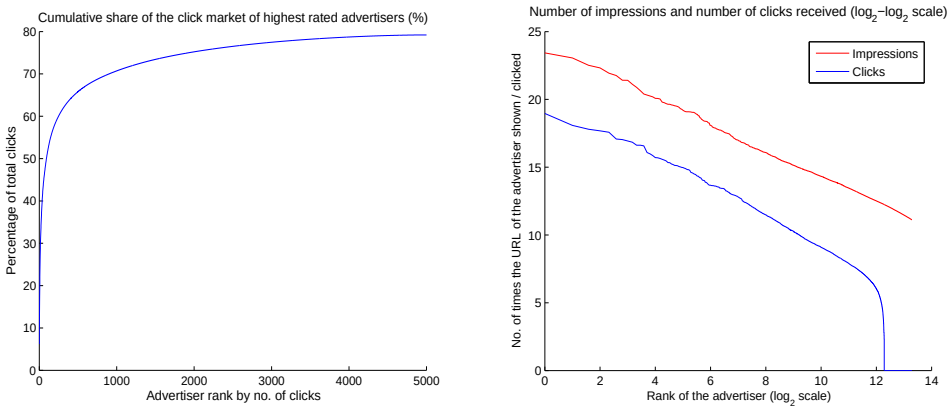


Fig. 2. A (left-side): Cumulative percentage distribution of the number of clicks advertisers in the market receive, wrt. to their rank position, considering the top 5000 advertisers in the market (normal scales). B (right): Log-log scale distributions of the number of impressions, respectively number of clicks, received by the top 10000 advertisers in the market. Note that both distributions follow approximately parallel power laws, but the click distributions levels off in a "long tail" after the first 4000 advertisers, while the impression distribution has a much longer tail (not all appearing in the Figure).

Since in our data, there are *at least* 10000 distinct advertisers (most likely, there are many more, but we only considered the top 10000), this means that a percentage of less than 5% of all advertisers have a two-thirds market share. This suggests that sponsored search markets are indeed very concentrated, perhaps even more so than "traditional" real-world markets.

4.1 Distribution of impressions vs. distribution of clicks for the top advertisers

Next, we studied the detailed distribution of the numbers of impressions (i.e. displayed URLs) and clicks on these impressions, for the top 10000 distinct advertisers. Results are shown in Figure 2.B. (right-hand side graph), using a log-log plot.

The main effect that one can see from Figure 2.B. is that the distribution of impressions and the distribution for clicks received by the advertisers form two approximately parallel, straight lines in the log-log space (i.e. they are two power laws of approximately the same slope coefficient α). There is one important difference, though, which is the size of the "long tail" of the distribution. The distribution of the number of clicks (lower line), levels off after about 4000-5000 positions. Basically, in data terms, this means that advertisers beyond the top 5000 each receive a negligible number of clicks, at least in the dataset we examined. The reason for this may be that their ads almost always appear in the lower display ranks, or simply that they bid on a set of rarely used

sponsored links to different products. However, using this technique, Ebay is taken as one advertiser, regardless of how many different items its URLs point to.

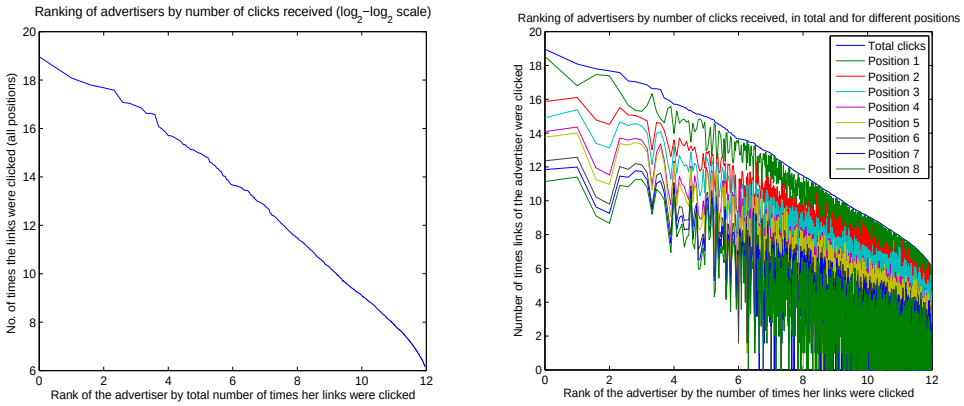


Fig. 3. Distribution of advertiser market share, based on their ordered rank vs. the number of clicks their links receive (log-log scales). The left-hand side plot (part A) gives the total number of clicks an advertiser received for all impressions of her links, regardless of the position they were in. The right-hand side (part B) gives the number of clicks received, both in total, but also when her ads were displayed on a specific position on the page (among the 8 ranked slots of the sponsored search interface).

(or highly specialised) search keywords. By contrast, the distribution of impressions still continues for many more positions (although we only represent the top 10000 distinct advertiser IDs here, as the rest do not play any significant role in the click market).

4.2 Distribution of market share per display rank position

The previous Section examined the power law distributions of the number of clicks each advertiser gets *in aggregate* (i.e. over all display ranks his/her links are shown in). Here, we look how an advertiser's market share distribution is affected when broken down per display rank (an issue we already touched on in Sect. 3).

However, we first make a slight restriction in the number of advertisers we consider. As shown in Sect 4.1 above, there is a power law distribution in the clicks received by the top 4000 advertisers, advertisers ranked beyond this position each receive a negligible number of clicks. Therefore, in this Section, we restrict our attention to the top 4000 advertisers. As these 4000 advertisers receive over 80% of all 7.8 million clicks in the data set (see Figure 2.A), we do not risk loosing much useful information.

Results are shown in Figure 3. First, in Figure 3.A. we show again, more clearly, the power law distribution of the number of clicks for the top 4000 advertisers. Note that this is a "wide" distribution, in the sense that it covers 4000 positions and several orders of magnitude. On the right-hand side graph (Figure 3.B), we show the same graph, but now, for each advertiser, we also break down the number of clicks received by the position his/her sponsored URL was in when it was clicked.

Surprisingly, perhaps, the smooth power law shape is not followed at the level of the display rank - in fact, for the lower levels the variance becomes so great that the dis-

tribution breaks down, at the display rank level. We hypothesize the most likely reason for this variance is the way each individual advertiser does the bidding for the preferred keywords at different points in time, or the way he specifies the way his keyword budget could be used in different periods. For example, some advertisers may have a short-running sale campaign, when they will bid aggresively for the preffered keyword, hence getting the top spot. By contrast, others may prefer to have longer-running ads, even if they don't get the top spot every time. Some anecdotal evidence from online marketing suggests that even just the repeated display of a link of a certain merchant on the screen may count: if a user sees an ad repeatedly in his/her attention space, that may establish the brand as more trustworthy.

In Figure 3.B, by loking the the top 4 advertisers in this dataset, one can already see that users ranked 2 and 3 utilize a rather different strategy than "the trend" represented by users 1 and 4. While their total numer of clicks does follow, approximately the power law, they seem to get, proportionally speaking, more clicks on the top-ranked slot on the page than the rest. While, in order to preserve the privacy of the data, we cannot mention who these companies are, it does seem that users 2 and 3 are actually "aggregators" of advertising demand. By this, we mean online advertising agencies or engines (or automated services offered by the platform itself) that aggregate demand from different advertisers and do the bidding on their behalf. Apparently, this allows them to capture, proportionally, more often the top slot for the required keyword. Unfortunately, however, we cannot investigate this aspect further, since the dataset provided does not contain any information about bidding, budgets or financial information in general.

In the following and last Section of this paper, we turn our attention to a somewhat different problem: how could we use insights gained from analyzing this query data to provide a bidding decision support for advertisers taking part in this market.

5 Using click data to derive search term recommendations

The previous Sections of this paper used complex systems analysis to provide a high-level examination of the dynamics of sponsored search markets. In this Section, we look at how such query log data could be used to output recommendations to individual advertisers. Such an approach should lead to answers to questions such as: What kind of keyword combinations look most promising to spend one's budget on, such as to attract a maximum number of relevant user clicks? While the previous analysis of power-law formation was done at a macro-level, in this Section we take a more local perspective. That is, we do not consider the set of all possible search terms, but rather a set that is specific to a domain. This is a reasonable model: in practice, most advertisers (which are typically online merchants), are only concerned with a restricted set of keywords which are related to what they are actually trying to sell.

For the analysis in this paper, we have chosen as a domain 50 keywords related to the tourism industry (i.e. online bookings of tickets, travel packages and such). The reason for this is that much of this activity is already fast moving online (e.g. a very substantial proportion of, for example, flight tickets and hotel reservations are now carried out online). Furthermore - and perhaps more important - there are low barriers of entry and the field is not dominated by one major player. This contrasts, for example,

other domains, such as the sale of Ipods and accessories, where Apple Stores can be expected to have a dominant position on the clicks in the market.

5.1 Deriving distances from co-occurrence in sponsored click logs

Given a large-scale query log, one of the most useful pieces of information it provides is the co-occurence of words in different queries. Much previous work has observed that the fact that two search keywords frequently appear together in the same query gives rise to some implicit semantic distance between them [10].

In this paper, we take a slightly different perspective on this issue, since, in computing the distances, we only use those queries which received at least one sponsored search click for the text ads (i.e. URLs) displayed alongside the results. We argue this is a subtle but very important difference from simply using co-occurence in organic search logs. The fact that queries containing some combination of query words lead to a click on a sponsored URL implies not only a purely semantic distance between those keywords, more important for an advertiser, the fact that users searching on those combinations of keywords have the possible intention of buying things online.

Formally, let $N(T_i, T_j)$ denote the number of times two search terms T_i and T_j appear jointly in the same query, if that query received at least one sponsored search click. Let $N(T_i)$ and $N(T_j)$ denote the same number of queries leading to a click, in which terms T_i, respectively T_j appear in total (regardless of other terms they co-occur with). The cosine similarity distance between terms T_i and T_j is defined as:

$$Sim(T_i, T_j) = \frac{N(T_i, T_j)}{\sqrt{N(T_i) * N(T_j)}} \tag{3}$$

5.2 Constructing keyword correlation graphs

The most intuitive way to represent similarity distances is through a keyword correlation graph. The results from our subset of 50 travel-related terms are shown in Figure 4. In this graph, the size of each node (representing one query term) is proportional to the absolute frequency of the keyword in all queries in the log. The distances between the nodes are proportional to the similarity distance between each pair of terms, computed Eq. 3, where the whole graph is drawn according to a so called "spring embedder"-type algorithm. In this type of algorithm, edges can be conceived as "springs", whose strength is indirectly proportional to their similarity distance, leading to cluster of edges similar to each other to be shown in the same part of the graph.

There are several commercial and academic packages available to draw such complex networks. The one we think is most suitable - and which was used for graph Figure 4 - is Pajek (see [3] for a description). Note that not all edges are considered in the final graph. Even for 50 nodes, there are $\binom{50}{2} = 1225$ possible pairwise similarities (edges), one for each potential keyword pair. Most of these dependencies are, however, spurious (they represent just noise in the data), and our analysis benefits from using

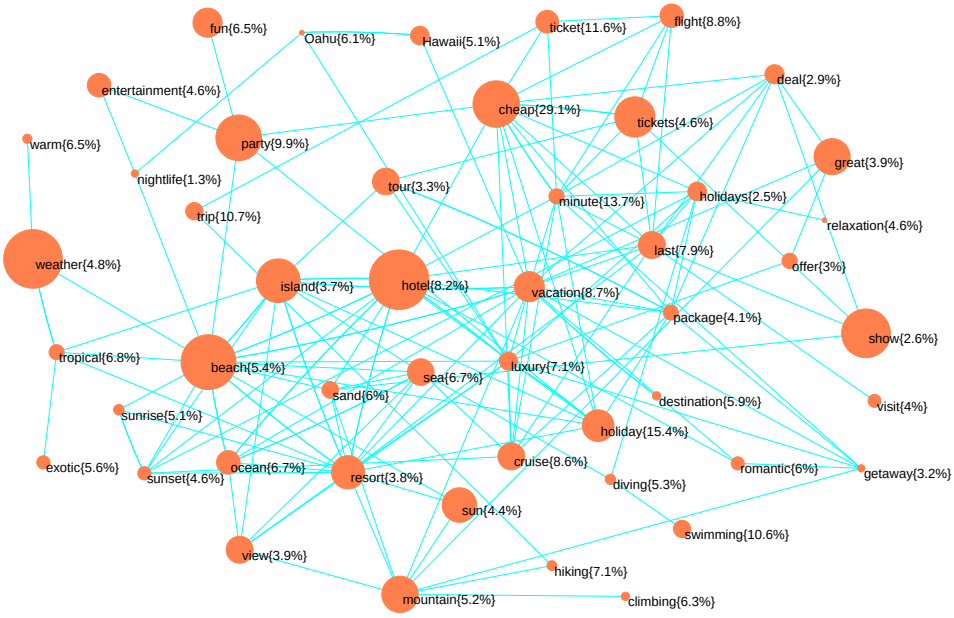


Fig. 4. Visualization of a search term correlation graph, for a set of search terms related to the tourism industry. Each search term is assigned one colored dot. The size of the dots gives its relative weight (in total number of clicks received), while the distances between the dots are obtained through a spring-embedder type algorithm and are proportional to the co-occurrence of the two search terms in a query. Each dot is marked with its success rate (percentage of the total number of impressions associated with that query word that received a click).

only the top fraction, corresponding to the strongest dependencies. In the graph shown in Figure 4, containing 50 nodes, only the top 150 strongest dependencies were considered in the visualization.

5.3 Graph correlation graphs: results

There are several conclusions that can be drawn from the visualization in Figure 4 constructed based on the Live.com sponsored search query logs. First, notice that each node was labelled not only with the term or keyword it corresponds to, but also with the aggregate click-through rate (CTR), specific for that keyword. Basicallly, this is the percentage of all the queries that used the term which generated at least one click to a sponsored search URL displayed with that query.

Note that these click-through rates may, at a first glance, seem on the low side: in general only a few percent of all queries actually lead to a click on an sponsored (i.e. advertiser) link. Nevertheless, as a search engine receives millions of queries in a

rather short period of time, even a 5%-10% click-through rate can be quite significant. Note that some keywords (such a "cheap") have a higher click-through rate than others. The reason for this may be that people searching for "cheap" things (e.g. cheap airline tickets, cheap holiday packages, hotel rooms etc.) may already have the intention to buy something online, and therefore are more likely to [also] click on sponsored links.

However, the most interesting effect to observe in Figure 4 are the term clusters that emerge in different parts of the graph, from the application of the spring-embedder visualization algorithm. For example, the leftmost part of the graph has 4 terms related to weather, such as "warm", "tropical" and "exotic". On the top left part of the graph, one can find terms such as "entertainment", "nightlife", "party" and "fun", while very bottom part includes related terms as such "climbing", "hiking" and "mountain". The top-right part includes commercial terms such as: "ticket", "tickets", "flight", "cheap", "last", "minute". The central part of the graph includes terms such a "beach", "sand", "sea", "resort", "ocean", "island" etc. Additionally, pairs of terms one would naturally associate do indeed appear close together, such as "romantic" and "getaway" and "sunset" and "sunrise" and "ocean".

In the following, we discuss an algorithm that can detect such clusters automatically. More precisely, we would like an algorithm that selects combinations of tags that look promising in attracting queries and clicks.

5.4 Automatic identification of sets of keywords

In this Section, we show how keyword graphs could be automatically partitioned into relevant keyword clusters. The technique we use for this purpose is the so called "community detection" algorithm [14], also inspired by complex systems theory. In network or graph-theoretic terms, a community is defined as a subset of nodes that are connected more strongly to each other than to the rest of the network (i.e. a disjoint cluster). If the network analyzed is a social network (i.e. vertexes are people), then "community" has an intuitive interpretation. However, the network-theoretic notion of community detection algorithm is broader, has been succesfully applied to domains such as networks of items on Ebay [11], publications on arXiv, food webs [14] etc.

Community detection: a formal discussion Let the network considered be represented a graph $G = (V,E)$, when $|V| = n$ and $|E| = m$. The community detection problem can be formalized as a partitioning problem, subject to a constraint. Each $v \in V$ must be a assigned to exactly one group (i.e. community or cluster) $C_1, C_2, ...C_{n_C}$, where all clusters are disjoint.

In order to compare which partition is "optimal", the metric used is *modularity*, henceforth denoted by Q. Intuitively, any edge that in a given partition, has both ends in the same cluster contributes to increasing modularity, while any edge that "cuts across" clusters has a negative effect on modularity. Formally, let $e_{ij}, i, j = 1..n_C$ be the fraction of all edge weights in the graph that connect clusters i and j and let $a_i = \frac{1}{2}\sum_j e_{ij}$ be the fraction of the ends of edges in the graph that fall within cluster i. The modularity Q of a graph $|G|$ with respect to a partition C is defined as:

$$Q(G,C) = \sum_i (e_{i,i} - a_i^2) \tag{4}$$

Algorithm 1 *GreedyQ Partitioning*: Given a graph $G = (V,E), |V| = n, |E| = m$ returns partition $< C_1, ...C_{n_C} >$

1. $C_i = \{v_i\}, \forall i = \overline{1,n}$
2. $n_C = n$
3. $\forall i, j$, e_{ij} initialized as in Eq. 5
4. repeat
5. $\quad < C_i, C_j >= \text{argmax}_{c_i,c_j}(e_{ij} + e_{ji} - 2a_i a_j)$
6. $\quad \Delta Q = \max_{c_i,c_j}(e_{ij} + e_{ji} - 2a_i a_j)$
7. $\quad C_i = C_i \bigcup C_j$, $C_j = \emptyset$ *//merge* C_i *and* C_j
8. $\quad n_C = n_C - 1$
9. until $\Delta Q \leq 0$
10. $maxQ = Q(C_1, ..C_{n_C})$

Informally, Q is defined as the fraction of edges in the network that fall within clusters, minus the expected value of the fraction of edges that would fall within the same cluster, if all edges would be assigned using a uniform, random distribution.

As shown in[14], if $Q = 0$, then the chosen partition c shows the same modularity as a random division. A value of Q closer to 1 is an indicator of stronger community structure - in real networks, however, the highest reported value is $Q = 0.75$. In practice, [14] found (based on a wide range of empirical studies) that values of Q above around 0.3 indicate a strong community structure for the given network. In our case, the edges that we considered in the graph (recall that only the strongest 150 edges are considered) have a weight, defined as shown in Eq. 3 above. For the purpose of the clustering algorithm, this weight has to be normalized by the sum of all weights in the system, thus we assign initial values to e_{ij} as:

$$e_{ij} = \frac{1}{\sum_{ij} sim_{ij}} sim_{ij} \tag{5}$$

5.5 The graph partitioning algorithm

The algorithm we use to determine the optimal partition is the "community identification" algorithm described in [14], formally specified as Alg. 1 above. Informally described, the algorithm runs as follows. Initially, each of the vertexes (in our case, each keyword) is assigned to its own individual cluster. Then, at each iteration of the algorithm, two clusters are selected which, if merged, lead to the highest increase in the modularity Q of the partition. As can be seen from lines 5-6 of Alg. 1, because exactly two clusters are merged at each step, it is easy to compute this increase in Q as: $\Delta Q = (e_{ij} + e_{ji} - 2a_i a_j)$ or $\Delta Q = 2 * (e_{ij} - a_i a_j)$ (the value of e_{ij} being symmetric). The algorithm stops when no further increase in Q is possible by further merging.

Note that it is possible to specify another stopping criteria in Alg. 1, line 9, e.g. it is possible to ask the algorithm to return a minimum number of clusters (subsets), by letting the algorithm run until n_C reaches this minimum value. Furthermore, this

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
beach	party	package	weather	getaway	diving	cruise	show	last
luxury	entertainment	vacation	exotic	romantic	swimming	sunrise	tickets	minute
hotel	nightlife	holidays	tropical			sunset	ticket	visit
island	fun	destination	warm				cheap	
resort	Hawaii	deal					flight	
sun	Oahu	tour						
mountain		offer						
ocean		great						
hiking								
climbing								
sea								
sand								
Keywords eliminated to increase modularity: holiday, holidays, relaxation, trip.								

Fig. 5. Optimal partition of the set of travel terms in semantic clusters, when the top 150 edges are considered. The partition was obtained by applying Newman's automated "community detection" algorithm to the graph from Figure 4. This partition has a clustering coefficient Q=0.59.

algorithm is computationally efficient, since it is basically linear in the size of the graph (number of keywords considered), hence it can be applied even to very large datasets.

5.6 Discussion of graph partitioning results

The results from the graph partitioning algorithm, showing the partition maximises the modularity Q for this setting, is shown in Figure 5. Note that this is not the only possible way to partition this graph - if one would consider a different number of strongest dependencies to begin with (in this case we selected the top 150 edges, for 50 keywords), or a different stopping criteria, one may get a somewhat different result. Furthermore, note that some keywords, which were very general and could fit in several clusters (shown below the figure), were pruned in order to improve modularity, through a separate algorithm not shown here.

Still, the partition results shown in Figure 5 match well what our intuition would describe as interesting combinations of search terms, for such a setting. There is one large central cluster, of terms that all have reasonably strong relations to each other, and a set of small, marginal clusters on the side. The large cluster in the middle could be further broken by the partition algorithm, but only if we force some other stop criteria than maximum modularity (such as a certain number of distrinct clusters).

The partition in Figure 5 fits well with what can be graphically observed in Figure 4: actually, most of the clusters obtained automatically after partition can be identified on different parts of the graph. This does not have to be a one-to-one mapping, however, because in a 2D drawing, the layout of the nodes after "spring embedding" may vary considerably and, furthermore, there are keywords which could fit well into 2 clusters, and were assigned to one as that had a slightly higher modularity.

6 Discussion

6.1 Contribution of the paper & related work

Our work can be seen as related to several other directions of research. Similar techniques to the ones used in this paper have been succesfully applied to analyze large-scale collaborative tagging systems [10] and preference networks for Ebay items [11].

The amount of work which is specifically geared to sponsored search auctions, especially empirical studies, has so far been rather limited (probably not least due to lack of extensive datasets in this field). Much of the previous work, e.g. [8] looks mostly at the bias introduced by a link's display rank on clicking behaviour (such as discussed in Sect. 3 of this paper). Another important direction of work uses existing intuitions about user clicking behaviour to design different allocation mechanisms for this problem - the work of [5] is a good example of this approach. By comparison to our work, the approach taken by [5] studies mostly at mechanism design issues arising from computational advertising, rather than perform an empirical examination of such markets.

One paper that is related in scope to ours, since it also provides an empirical study of search engine advertising markets is [9]. This work takes, however, a different perspective on this problem, also due to the different type of data the authors had available. By contrast to our work, the data that [9] use comes from a single, large-scale advertiser. This means they do get access to more detailed information (including financial one) and can say more about actual bidding behaviour. By comparison, the data available to us for this study does not contain any detailed financial information, but, unlike [9] it allows us to have a global level view of the whole market (from the perspective of the search engine, not just a single advertiser). This provides very important insights about the structure of sponsored search markets.

Finally, there exists previous work that has applied similar co-occurence-based techniques to organic search logs or tagging systems [7, 10]. However, our focus in this paper is different: we do not aim to to merely deduce what is the semantic distance between keywords in the general sense, but what kind of combinations of keywords are financially interesting for a sponsored search advertiser to bid on. This is the reason why the size of the nodes and distances computed in Figure 4 are built using only queries which lead to an actual click on a sponsored ad. Basically, this is equivalent to filtering only the "opinion" (expressed through queries) of the subset of users that are likely to buy something online, rather than all search engine users. To our knowledge, this is the first paper to use sponsored search click data in this way.

6.2 Future work

This work, being somewhat preliminary, leaves many aspects open to future research. On such aspect would be is the issue of *externalities*: how the presence of links by competing advertisers influences the clickthrough rates of other bidders. As the competition is basically on customers' attention space, externalities play an important role in the efficacity of sponsored search impressions.

Another very interesting topic would be to study the structure of sponsored search markets (in terms of advertiser market share etc.) not only at the global, macro-level,

but at the level of individual sets of keywords. In fact, sponsored search can be seen not only as one market, as a network of markets, since most advertisers are interested in (and bid on) a specific set of keywords related to what they are selling. For example, we could apply our "community detection" algorithm to partition not only sets of search keywords, but also sets of bidders (advertisers) interested in those keywords. This should allow us to derive more in-depth insights into the structure of sponsored search.

Acknowledgements
The authors thank Microsoft Research for their support, in the framework of a 'Beyond Search" award. We also wish to thank Nicole Immorlica and Renato Gomes (Nortwestern University) for many useful discussions in the preliminary stages of this work.

References

1. A. Baldassarri, A. Barrat, A. Cappocci, H. Halpin, U. Lehner, J. Ramasco, V. Robu, and D. Taraborelli. The berners-lee hypothesis: Power laws and group structure in flickr, 2008. Proc. of Dagstuhl Seminar on Social Web Communities, Dagstuhl DROPS 08391.
2. Y. Bar-Yam. The dynamics of complex systems. *Westview Press*, 2003.
3. V. Batagelj and A. Mrvar. Pajek - A program for large network analysis. *Connections*, 21:47–57, 1998.
4. S. M. Bohte, E. Gerding, and J. L. Poutré. Market-based recommendation: Agents that compete for consumer attention. *ACM Trans. Internet Technol.*, 4(4):420–448, 2004.
5. C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *WWW '07: Proc. 16th Int. Conf. World Wide Web*, pages 531–540. ACM Press, 2007.
6. T. Carter. A short trip through entropy to power laws, 2007. Complex Systems Summer School, Santa Fe Institute, NM.
7. R. L. Cilibrasi and P. M. B. Vitanyi. The google similarity distance. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):370–383, 2007.
8. N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proc of. WSDM '08*, pages 87–94. ACM Press, 2008.
9. A. Ghose and S. Yang. Analyzing search engine advertising: firm behavior and cross-selling in electronic markets. In *WWW '08: Proc. of the 17th Int. Conf. on World Wide Web*, pages 219–226. ACM Press, 2008.
10. H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *Proc. 16th Int. World Wide Web Conf. (WWW'07)*, pages 211–220. ACM, 2007.
11. R. K.-X. Jin, D. C. Parkes, and P. J. Wolfe. Analysis of bidding networks in eBay: Aggregate preference identification through community detection. In *Proc. AAAI Workshop on Plan, Activity and Intent Recognition*, 2007.
12. K. L. Judd and L. Tesfatsion. Handbook of computational economics II: Agent-Based computational economics. *Handbooks in Economics Series, North-Holland, the Netherlands*, 2005.
13. M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323–351, 2005.
14. M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev.*, E 69, 066133, 2004.
15. V. Robu and H. La Poutré. Designing bidding strategies in sequential auctions for risk averse agents. In *Agent-Mediated Electr. Commerce,*, pages 76–89. Springer LNBIP 13, 2007.

Towards Cooperative Predictive Data Mining in Competitive Environments

Viliam Lisý, Michal Jakob, Petr Benda, Štěpán Urban, and Michal Pěchouček

Agent Technology Center, Dept. of Cybernetics, FEE, Czech Technical University
Technická 2, 16627 Prague 6, Czech Republic
{lisy, jakob, benda, urban, pechoucek}@agents.felk.cvut.cz

Abstract. We study the problem of predictive data mining in the competitive multi-agent setting, in which each agent is assumed to have some partial knowledge needed for correctly classifying a set of unlabelled examples. The agents are self-interested and therefore need to reason about the trade-offs between increasing their classification accuracy by collaborating with other agents and disclosing their private classification knowledge to other agents through such collaboration. We analyze the problem and propose a set of components which can enable cooperation in this otherwise competitive problem. These components include measures for quantifying private knowledge disclosure, data-mining models suitable for multi-agent predictive data mining, and a set of strategies by which agents can improve their classification accuracy through collaboration. The overall framework and its individual components are validated on a synthetic experimental domain.

1 Introduction

We study the case of multiple self-interested parties (termed *agents* further on) working on a common but partitioned predictive data mining task[1] in a competitive environment. The knowledge used in solving the data mining task is considered a valuable asset of each party because accurately predicting data classifications is assumed to give the party a competitive advantage. One of the ways the classification accuracy can be improved is by exchanging knowledge with other parties. Given the competitive nature of the domain, such exchange cannot be done on an arbitrary basis but needs to follow sound strategies that consider knowledge lost and gained during each transaction. The ability to implement such strategies is thus a necessary precondition to enabling cooperation between the agents.

In this paper, we outline a framework and describe a set of specific methods and techniques that make such an implementation possible. First, we develop a set of private knowledge disclosure metrics that measure the classification knowledge lost in data and model exchanges between the agents. We identify a set of operations that need to be supported by the underlying classification models to make them applicable in the cooperative setting, in particular the possibility to effectively evaluate private

[1] Different agents have different datasets but all the datasets are sampled from a single underlying model

knowledge metrics and to incrementally accommodate prediction knowledge obtained from other agents and, vice versa, to effectively extract knowledge to be shared. Not all classification models fulfil these requirements – we identify the Naive Bayes classifier as a particularly suitable class of models for the semi-cooperative prediction, and we show how individual operations can effectively be implemented for this class. We then describe several cooperative strategies that the agents can use to improve their classification capability. For each strategy, we measure the resulting improvement of agent's classification capability and set it against the loss of privacy entailed.

The work presented should be viewed as an initial step towards enabling cooperation in predictive data mining in a community of self-interested and competitive agents.

2 Related Work

Over the last decade, privacy preserving data mining (PPDM) [1] has attracted considerable attention. One of the main objectives of the field is designing data transformations which allow publishing data without losing privacy. In most cases, the emphasis is on protecting the privacy of *individuals* described by the data records – not on the protection of knowledge contained in the data set as a whole. Consequently, losing exact information about a small number of records is considered unacceptable while complete models learned from the data can be freely shared unless they allow the derivation of individual records [2].

This is well acceptable for the preservation of private information about individuals, such as medical records, but it is not sufficient in reasoning about the preservation of private data, knowledge, and know-how of companies. Almost any data mining result, a cluster model, a frequent sequence, or an association rule derived from the private company data can have commercial potential and should not be blithely shared without further assessment.

The subfield of PPDM that deals with this kind of private information, i.e. the knowledge contained in collections of data rather than individual records, is termed *corporate privacy* in [3] or *knowledge hiding* e.g. in [4]. There is, however, no general framework to reason about private knowledge disclosure – each technique has its own methods and measures of quality of private knowledge hiding. The research closest to the focus of this paper is classification rules [5] and association rules [6] hiding. The objective of these methods is to transform the data by resampling, reduction and/or perturbation so that a specified set of rules cannot be mined from the data; the outcome is measured in terms of the number of rules successfully hidden, the number of original data items modified and the number of new rules introduced by the transformation.

In contrast to the above, in this paper, we do not aim to conceal any specific classification rules; instead, we aim to measure and minimize the overall knowledge disclosed about private classification models possessed by individual agents (and possibly obtained by generalizing their private data). As far as measures for quantifying privacy loss in PPDM are concerned, a summary is presented in [7]. Existing measures, often utilizing the notion of information entropy, are designed to measure the disclosure of a specific kind of knowledge. Moreover, except for a few exceptions (e.g. [8] for as-

sociation rules), existing measures deal with individual privacy and are therefore not applicable to our problem.

On the other hand, privacy loss measured in terms of changes to information entropy has been used in multi-agent system research outside the data mining field (e.g. in multi-agent meeting scheduling [9], or multi-agent planning [10].

3 Problem Description

The problem addressed is *semi-cooperative classification* in competitive multi-agent domains. Each agent aims to accurately classify a set of unlabelled examples drawn from the same data distribution. In order to improve its classification accuracy, the agent can either exploit its own data and models or request additional knowledge from other agents. More specifically, the problem is defined as follows. There are multiple agents in the system, each containing its:

- set of labelled training data D_i
- set of unlabeled data for classification C_i (termed *task data*)
- a data mining algorithm M_i that can construct a classification model $M_i(S)$ from any labelled set of data S

The task of each agent is to classify its unlabelled task data C_i. In doing so the agent employs a particular *strategy* which involves exploiting its training data D_i and communicating with other agents about their models, data and classifications. Each strategy results in a specific classification accuracy (measured by a designated accuracy metric – see Section 4.1) and disclosure of a specific amount of private knowledge sourced externally (measured by a designated private knowledge loss metric – Section 4.2).

4 Relevant Measures

In this section, we introduce measures for evaluating agent's knowledge sharing actions with respect to (1) the loss of private knowledge and (2) the increase in classification accuracy. We briefly introduce the measures here and discuss their properties and suitability in a more detail in Section 7.

4.1 Private Knowledge Loss

According to our literature survey, no metrics for measuring the loss of information about a private classification model has been so far reported. In the following, we describe and analyze how symmetrised Kullback-Leibler divergence and mutual information can be used for this purpose. The application of these measures to quantify private knowledge loss is novel.

Both proposed measures are based on the following idea. If an agent sends some information originating from its private classification model (e.g. classified examples or partial models) to another agent, this information can be used to (approximately) reconstruct the private classification model of the sending agent. For example, if the

agent discloses a set of feature vectors classified by the private model, these samples can be used to train a new model that approximates the original one. The proposed measures therefore evaluate the distance between the original private model and the (hypothetical) reconstructed model and use it as a quantification of the private classification knowledge loss.

The proposed measures assume that any classifier can be interpreted in two ways. Either as a representation of a joint probability distribution over the feature space and the classification labels, or as a realization of a function assigning a unique label to each feature vector. Although the first representation is more informative, it is often hard to extract from certain classes of classification models. For example, the k-nearest neighbour classifier approximates the distribution by the ratio of examples of individual classes in the neighbourhood of the feature vector; decision trees can approximate the distribution for a feature vector by the ratio of class frequencies of training examples corresponding to the leaf the feature vector belongs to. For the cases where the distribution can be extracted, we propose a measure based on the probabilistic similarity measure between the original distribution represented by the private classifier and the distribution realized by the reconstructed classifier.

On the other hand, extracting the distribution from a set of rules derived by ILP or similar method is not straightforward. Moreover, the internal structure of some classifiers is not always accessible to the agent (e.g. in the case of external libraries) and the classifier can be accessed only as a black-box producing a classification (class label) for any input feature vector. If classifications are the only available information about a classifier, we propose a metric based on information theory that measures the information present in the classifications produced by the reconstructed model about the classifications corresponding to the original private model.

Symmetrized Kullback-Leibler Divergence Kullback-Leibler (KL) divergence (also termed *information divergence* or *information gain*) is a standard measure used in probability theory to compute the similarity of two probability distributions. For two discrete probability distributions P and Q over the same random variable (the same feature space in our case), the KL divergence of Q from P is defined as

$$D_{kl}(P\|Q) = \sum_x P(x) log_2 \frac{P(x)}{Q(x)} \tag{1}$$

where x in the sum iterates over the range of the random variable. KL divergence in this form is not symmetric, i.e., $D_{kl}(P\|Q) \neq D_{kl}(Q\|P)$ in most cases. That is why a symmetrised form of Kullback-Leibler divergence is often used

$$D_{skl} = D_{kl}(P\|Q) + D_{kl}(Q\|P) \tag{2}$$

The iteration over the range of the random variable corresponds to iterating over the whole feature space in the case of a distribution realizing a classifier. This is generally computationally expensive for large distributions. For example in our case of 10 features with 10 possible values each, the size of the feature space is 10^{10}. However, for some specific classifiers, the value of KL-divergence can be approximated or even exactly computed in a significantly more effective way. (See Section 5.2).

Mutual Information Mutual information is a standard information-theoretic measure quantifying the mutual dependence of two random variables. It represents the amount of uncertainty about one random variable that is removed by knowing the value of the other random variable. This measure can be used to measure how much information a model reveals about a data set or about the classifications produced by a model regardless of the structure of the classifier.

For random variables X and Y, mutual information is defined as

$$I(X:Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) log_2 \frac{P(x,y)}{P(x)P(y)} \tag{3}$$

Mutual information can be used to compute both the amount of information a model reveals about a data set and the amount of information one model contains about another model. We focus on the second case here. Classifications produced by the private model and by its approximation correspond to the two random variables in the formula. The only step needed is to estimate the probability distribution $P(X,Y)$, i.e., a normalized matrix specifying how frequently one classifier classifies a feature vector to class x while the other classifies the same vector to class y, which may or may not be the same. In the case of a smaller feature space, this probability distribution can be computed precisely by iterating over the whole feature space, similarly to the case of the distribution-based metric. Such explicit computation is not possible for large feature spaces. Instead, we therefore sample the two classification models on a chosen subset of the feature space and estimate the probability distribution $P(X,Y)$ from the confusion matrix resulting from the two sets of produced classifications.

A big advantage of this measure is its complete independence on the classifier implementation. A disadvantage of the mutual information metric is the amount of samples needed for good approximation of the distribution. We have performed a simple synthetic experiment that demonstrates this problem and shows some basic properties of the metric. We have simulated classification into 10 classes as in the experiments presented later. We were measuring the information that an imprecise classifier contains about an ideal classifier that always assigns the correct class. The probabilities in formula 3 can be estimated from the confusion matrix of the classifier, so we were generating only that matrix in this experiment. Correct classification was provided with probability p, otherwise the classification was random to any of the remaining classes with uniform distribution. Figure 1 shows the dependence of the mutual information metric on the number of generated classifications in the confusion matrix in a single run of the experiment. The probability of correct classification were set to $p = 0$, $p = 0.5$ and $p = 1$, respectively, and the results shown are the mean of thousand runs. The variance was very high for less than 100 samples but decreased for more samples; the results obtained from more than 1000 samples were already almost identical across individual runs.

As we expected, when there is enough data samples, the mutual information between the correct classification and a classifier that always misclassifies into one of the nine wrong classes is almost zero. If the classifier classifies a half of the examples correctly, the mutual information is higher, and it is the highest when the classifier classifies all examples correctly. In the latter case of correct classification, it converges to

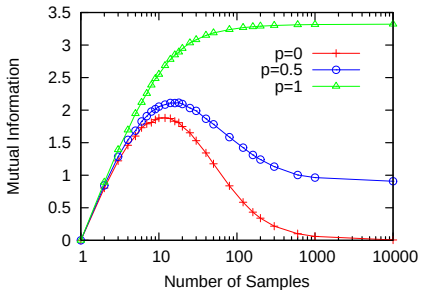


Fig. 1. Dependence of the mutual information metric on the number of samples used to construct the confusion matrix (from which the metric is calculated) for different probabilities of correct classification.

the value of 3.32 bits, which corresponds to distribution $P(X,Y)$ with 10 classes and the probability uniformly distributed along the main diagonal.

The inaccurate probability estimations caused by a smaller number of samples lead, besides high variance, to a misleading increase in the measured information disclosure e.g. incorrectly indicating that ten random classifications reveal more information about the private model than a thousand classifications that are correct in half of the cases. In order to obtain reliable assessment of information disclosure using the mutual information metric, at least a thousand samples are needed.

4.2 Classification Accuracy

There are many measures for assessing classification accuracy. Some of them are based on ratios of true positive and true negative classifications or consider different costs of misclassification to different classes – see e.g. [11] for an overview. The primary focus of this paper is on private knowledge preservation. We therefore use a ratio of correctly classified examples to all examples on a test data set as a basic classification accuracy measure.

5 Naïve Bayes for Multi-Agent Data Mining

A data mining model suitable for semi-cooperative multi-agent classification task should satisfy several criteria:

- *creating and joining sub-models* If agents exchange sub-models, their creation and joining should be possible and computationally efficient.
- *confidence* The model should be able to output the degree of confidence in the classification to allow the agents to decide when they need some extra information from other agents.
- *effective measures computation* The model should allow fast computation (or approximation) of relevant measures, in particular the private knowledge loss.

– *compact model size* The models should have a compact representation to allow sharing in case of limited communication bandwidth

One of the models that satisfies all the above properties is Naïve Bayes classifier. Joining two models is straightforward if they are represented as frequencies instead of probabilities (see Section 5.2). Creating a model requires only one iteration through the data set. The classifier output in the form of a posterior probability is suitable confidence measure and as we show below, the suggested privacy measures can be computed exactly in a reasonable time for this model.

5.1 Naïve Bayes Classifier

Let X is an n-dimensional feature space, X_i is a random variable representing the i-th component of the feature vector, and S is a set of all classes. Naive Bayes then classifies a given sample $(x_1,\ldots,x_n)$ into class

$$d = \arg\max_{s\in S}\left[P(S=s)\prod_{i=1}^{n}P(X_i=x_i|S=s)\right] \tag{4}$$

The product in the formula approximates the joint probability

$$P(X_1=x_1,X_2=x_2,\ldots,X_n=x_n|S=s) \tag{5}$$

under the assumption of the conditional independence of individual features.

5.2 Joining Naïve Bayes Models

The model sharing strategies require the ability to join classification models. In our framework, this is implemented in the following way. Naïve Bayes classifier is composed from a set of probability distributions

$$M = \bigcup_{s\in S}\{P(s),P_1(x_1|s),\ldots,P_n(x_n|s)\} \tag{6}$$

where $x_1\in X_1,\ldots,x_n\in X_n$. We have implemented the model as

$$\bigcup_{s\in S}\{f(s),f_1(x_1|s),\ldots,f_n(x_n|s)\} \tag{7}$$

where $f(s)$ is the frequency of class s in the training set and $f_i(x_i|s)$ is the frequency of i-th attribute in the subset of the training data corresponding to class s. The number of data samples in the training set used to train the classifier can be directly determined from the implemented model as $a=\Sigma f(s)$. Probability $P(s)$ is then given by $\frac{f(s)}{a}$ and probabilities $P_i(x_i|s)=\frac{f_i(x_i|s)}{f(s)}$

Two models M^1 and M^2 are then joined as:

$$\bigcup_{s\in S}\{f^1(s)+f^2(s),f_1^1(x_1|s)+f_1^2(x_1|s),\ldots,f_n^1(x_n|s)+f_1^2(x_n|s)\} \tag{8}$$

KL-Divergence for Naïve Bayes The Naïve Bayes classifier and its feature independence assumption allows to significantly simplify the computation of certain measures. A simplified calculation of the otherwise computationally expensive KL-Divergence for two probabilistic distributions P_1 and P_2 representing two Naïve Bayes classifiers is shown below[2]

$$D_{kl}(P_1 \| P_2) \quad = \quad \sum_{s \in S} P_1(s) \left(log_2 \frac{P_1(s)}{P_2(s)} + \sum_{i=1}^{n} \sum_{x_i \in X_i} P_1(x_i|s) \, log_2 \frac{P_1(x_i|s)}{P_2(x_i|s)} \right) \quad (9)$$

where n is the number of features, X_i are the domains of the features and S is it set of classes.

6 Cooperation Strategies

As described in Section 3, the agents can use a cooperation strategy in order to improve their classification accuracy. The strategy can either involve requesting information about other agents' private models or asking the other agents to classify a set of unlabelled data using their private models. In the following description of the strategies, we use the term *requestor* for the agent aiming to improve its classification using the private knowledge of the other agents (which are termed *providers*).

6.1 Model and Sub-model Sharing

In this strategy, the requestor asks the providers for a partial or degraded version of their private models. In our experiments, the degraded version is a model trained only on a fraction of the dataset; other means of degradation, such as adding noise to the model, or defining the model only on a subset of the feature-space, can be employed. The providers willing to share some information create partial models degraded to a level satisfying their private knowledge disclosure constraints and send them to the requestor. The requestor then merges the information provided in the received partial models into its own private model and uses the resulting improved model to classify its task data. In the case of the Naïve Bayes model, the merge algorithm has been described in Section 5.2.

6.2 Data Classification

In the case of the data classification strategy, the requestor first classifies its task data using its private classifier and sorts the results according to the confidence of the classification (the a posterior probability in the case of the Naive Bayes model). It then decides about the fraction of data with the lowest classification confidence and sends them to the providers for classification. The providers use their private classification models to label the data and send the resulting classifications back to the requestor. The requestor aggregates the received classifications with the results of its own classifier

[2] Detailed derivation is available on request from the authors.

and determines the final classification of the data. The aggregation mechanism used in our implementation is majority voting with a preference for own classification in case of ties. A certain disadvantage of the data classification strategy for the requestor is that by specifying the data to be classified the requestor discloses some information about its own data.

In order to measure the private knowledge loss using the KL measure, an additional step is needed when using the data classified strategy. The labelled data provided by the providers in response to the requestor's queries are first used to train a Bayes classifier; this classifier is then used to calculate the amount of private knowledge disclosed.

7 Experiments

In order to validate our approach and to obtain quantitative data on the behaviour of the proposed metrics and strategies, we have designed a set of experiments on a synthetic domain.

7.1 Experiment Domain and Setting

The feature space is composed of two features; each feature can assume a value from $\{0,1,\ldots,9\}$. The feature vectors belong to one of ten possible classes with equal probability. Data samples for each class are drawn from a Gaussian distribution (restricted to the feature space) defined by its mean and a fixed unit variance. The distribution means are selected randomly with a uniform probability distribution from the whole feature space. As a result, identical feature vectors with different classifications can appear in the domain. The situation is illustrated in Figure 2.

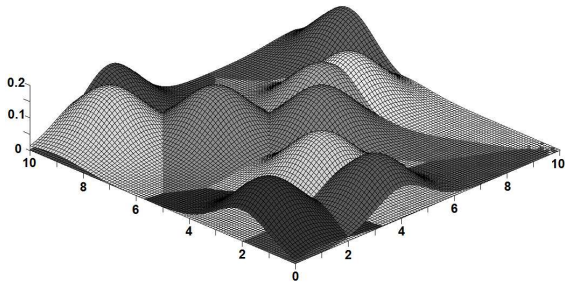


Fig. 2. The domain used in the experiments. Each Gaussian generates data samples from a different class.

The community of agents in the experiments comprises ten identical agents $A_0, A_1, \ldots A_9$. Each agent employs one of the two cooperation strategies (see Section 6) to classify its own task data with the help of the classification knowledge obtained from all other agents. Each agent has its private classification model trained on hundred data samples (D_i) and it has another thousand data samples to classify as its task data (T_i). All

presented results are averages over one hundred runs of the experiment with identical settings.

KL divergence as well as the naïve Bayes classification works well only with positive distributions, i.e. each combination of features and classification is considered possible (there are no zeros in the distribution). In order to achieve this property in our experiments, we initialize the frequencies representing the Naïve Bayes classifier as explained in Section 5.2 with ones. However, these extra ones are kept only once in the process of model merging.

7.2 Results for the Model Sharing Strategy

The main parameter of the model sharing strategy is the portion of the training data the provider uses to train a degraded classifier that it sends to the requestor. The dependency of individual measures on this parameter are depicted in the upper row of graphs in Figure 3.

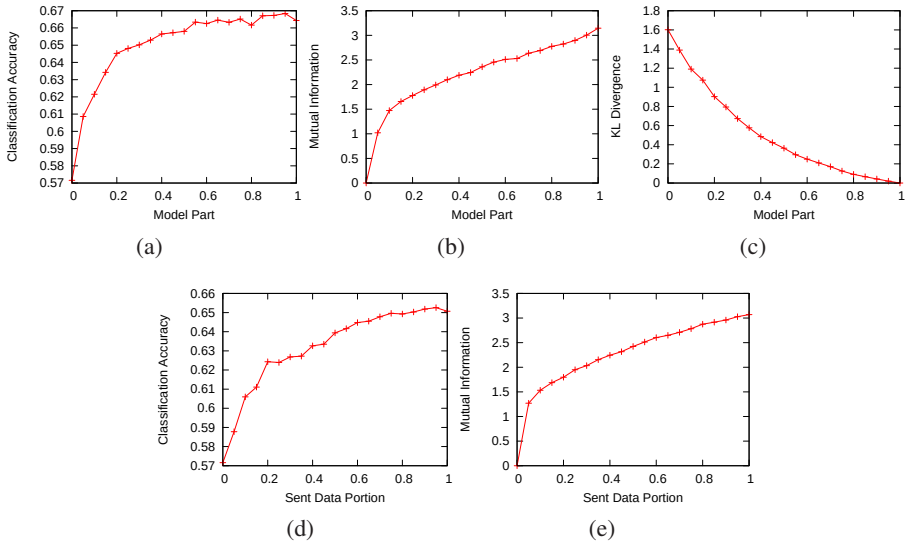


Fig. 3. Classification precision of requestor A_0 using the model sharing strategy and the private knowledge loss of agent A_1 responding to requestor's queries according to various measures for the model sharing (a,b,c) and the data classification (d,e) strategies.

Classification Accuracy Classification accuracy measure plotted in Figure 3(a) shows how model parts from other agents merged together with the original model of the requestor agent A_0 improve requestor's classification accuracy. The increase in prediction accuracy was expected – the bigger the part of provider agents' models merged into the

model of the requestor, the more information about the domain the resulting model captures. If the requestor A_0 receives models learned only from 5% of providers' data, the accuracy of its classification is approximately 61%. The accuracy increases quickly at the beginning but the improvement tails off closer to sharing full models; this is because the private classification model is already well approximated at this point.

Private Knowledge Loss In this strategy, the provider agents disclose their private knowledge in the form of partial classification models – see the results for the individual measures below. Note that the requestor agent does not disclose any information at all.

Mutual Information The mutual information is measured between the original private model of the provider and the (degraded) model sent to the requestor. The data used to sample the distribution $P(X,Y)$ from formula 3 are the complete training set the provider used to obtain its private classifier. The dependency of the mutual information between the models and the portion of the data used to train the shared model is depicted in Figure 3(b). Even though it is measured on a different agent, the trend is similar to the classification improvement of the requestor. It means that the mutual information metric accurately describes the amount of useful information contained in the sent model; more data used for creating the degraded model implies higher private knowledge disclosure.

KL-Divergence Figure 3(c) depicts the trend of KL divergence, which measures the difference between the disclosed model and the private one. The trend is in agreement with expectations – initially, increasing the amount of data on which the degraded model is created contributes strongly to narrowing the difference between the provider's private model and the sent model; the contribution decreases when the degraded model approaches the provider's private model (i.e. when the portion of the training data used approaches one).

7.3 Results for the Data Classification Strategy

The main parameter of the data classification strategy is the percentage of the task data with the smallest classification confidence the requestor sends to the providers for classification. The experimental results concerning this strategy are summarized in the bottom row of graphs in Figure 3.

Classification Accuracy Figure 3(d) confirms the basic hypothesis that the classification accuracy can be improved using the data classification strategy – the accuracy grows clearly with the increased amount of shared testing data of the requestor. The irregularities in the trend are caused by fluctuations of the classification precision between individual runs – a smoother graph would be obtained for a higher number of runs or a large data set. The improvement through collaboration is largest when the data on which the requestor has a very low confidence are sent to the providers for classification. The increase in accuracy diminishes when the agent requests classifications for the majority of its testing data; this is because then the requestor already has a good chance of predicting the class correctly.

Private Knowledge Loss In contrast to the model sharing strategy, both the requestor and the providers lose private knowledge in this case. The requestor A_0 loses private knowledge about its task data while the provider agents lose information about their private models.

The amount of information lost by the requestor can be quantified by the number of examples it sends to the other agents for classification. The loss for the providers can be expressed by the proposed measures.

Mutual Information The plot in Figure 3(e) depicts the loss of knowledge about providers' private models caused by responding to requester's queries. The loss of the private knowledge of providers increases with the number of queries answered, fast at the beginning and slower towards the end when the requestor already has enough information to reconstruct the providers' models accurately.

KL-Divergence We do not present a plot of KL divergence metric for this strategy because the metric proved unsuitable for the data classification strategy. KL divergence computes the distance between two probability distributions realized by the corresponding classifiers. In order to use KL divergence, we first need to reconstruct the classifier from the classified samples sent. The problem arises from the fact that different distributions can correspond to identical classifications. If a feature vector can belong to multiple classes in the ground truth, the original classifier can learn this fact but it always outputs only the most probable class. A classifier trained only from sampling the original classifier will have the probability of classifying to the dominating class equal to one, and the probabilities of classifying to the other classes equal to zero. As a result, the KL divergence can grow even though the similarity of produced classifications increases.

7.4 Strategy Comparison

Both the evaluated strategies are usable for improving classification accuracy of the requestor agents. Starting from the same base level (57%), the model sharing strategy achieved up to 67% accuracy while the data sharing reaches only 65% even if all the thousand task data samples are shared. On the other hand, the model sharing strategy has to reveal the whole model of the provider agent (instead of classified samples) in order to achieve the peak accuracy. We compare loss of the private knowledge about the task data and the private knowledge contained in classifiers.

The information about the task data is communicated only in the case of data sharing strategy. The requestor agent sends a portion of its testing data given by the strategy parameter to all provider agents. The strategy parameter represents how much of the private testing data will be disclosed.

The knowledge about private classification model is disclosed in both strategies. The measure suitable for measuring the amount of private knowledge revealed in both of them is the mutual information metric. The graphs for both strategies are almost identical. According to that, sending model trained on k samples form an agent training set reveals as much private knowledge about the complete model as answering $10 * k$

queries for classification of other agents task data. The size of the task data sets is ten times bigger than the size of the training sets.

The inapplicability of KL divergence as a sound private knowledge loss measure for the data classification strategy indicates that the knowledge exchanged in the strategy does not allow to fully reconstruct a provider's model, in particular its classification confidence. This is not the case for the model sharing strategy where the confidence information is disclosed as part of the exchanged partial models.

8 Conclusion

We have analyzed the trade-offs between improving classification accuracy and losing private classification knowledge in the multi-agent setting. Reasoning about such trade-offs is of high relevance whenever multiple competitive parties want to cooperate in order to improve their own performance. We believe such situations are common in real-world environments, including various business sectors (e.g. banking, insurance etc) or areas of international cooperation.

Understanding that cooperation cannot arise unless individual parties can reason about losses and benefits entailed by such cooperation, we have designed a set of measures and a benchmark task on which the measures can be validated. Afterwards, we have presented two elementary strategies consisting of a set of operations which are likely to be the building blocks of any multi-agent strategy aimed at improving classification accuracy through agent cooperation. The first strategy is based on sharing partial classification models; the other strategy employs consulting other agents' opinions about the classification of specific examples. We have evaluated the strategies on a synthetic domain and compared their respective advantages and disadvantages. The results obtained are in agreement with our theoretical expectations and indicate that the framework developed could be used for building semi-cooperative data mining systems.

The work presented is, in any case, only a first step towards enabling more complex strategies e.g. involving reciprocity or payment for the private knowledge disclosed. In a longer term, we aim to design fully autonomous agents that can automatically reason about the benefits and costs of individual knowledge sharing operations in the context of semi-cooperative predictive data mining.

9 Acknowledgments

We gratefully acknowledge the support of the presented research by Office for Naval Research project N00014-09-1-0537 and the Research Programme No. MSM6840770038: Decision Making and Control for Manufacturing III by the Ministry of Education of the Czech Republic.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proc. of the ACM SIGMOD Conference on Management of Data, ACM Press (May 2000) 439–450

2. Kantarcioğlu, M., Jin, J., Clifton, C.: When do data mining results violate privacy? In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2004) 599–604
3. Clifton, C., Kantarcioglu, M., Vaidya, J.: Defining Privacy for Data Mining. In: National Science Foundation Workshop on Next Generation Data Mining. (2002) 126–133
4. Bonchi, F., Saygin, Y., Verykios, V., Atzori, M., Gkoulalas-Divanis, A., Kaya, S., Savas, E.: Privacy in Spatiotemporal Data Mining. Mobility, Data Mining and Privacy: Geographic Knowledge Discovery (2008)
5. Natwichai, J., Li, X., Orlowska, M.: Hiding Classification Rules for Data Sharing with Privacy Preservation. LECTURE NOTES IN COMPUTER SCIENCE **3589** (2005) 468
6. Verykios, V.S., Gkoulalas-Divanis, A.: A Survey of Association Rule Hiding Methods for Privacy. In: Privacy-Preserving Data Mining. Springer US (2008)
7. Bertino, E., Lin, D., Jiang, W.: A Survey of Quantification of Privacy Preserving Data Mining Algorithms. In: Privacy-Preserving Data Mining. Springer US (2008)
8. Bertino, E., Fovino, I., Provenza, L.: A Framework for Evaluating Privacy Preserving Data Mining Algorithms*. Data Mining and Knowledge Discovery **11**(2) (2005) 121–154
9. Franzin, M., Rossi, F., Freuder, E., Wallace, R.: Multi-Agent Constraint Systems with Preferences: Efficiency, Solution Quality, and Privacy Loss. Computational Intelligence **20**(2) (2004) 264–286
10. van der Krogt, R.: Privacy loss in classical multiagent planning. Intelligent Agent Technology, IEEE / WIC / ACM International Conference on **0** (2007) 168–174
11. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)

Enhancing Agent Intelligence through Data Mining: a Power Plant Case Study

Christina Athanasopoulou and Vasilis Chatziathanasiou

Department of Electrical and Computer Engineering,
Aristotle University of Thessaloniki, Thessaloniki, Greece
{athanasc,hatziath}@eng.auth.gr

Abstract. In this paper, the methodology for an intelligent assistant for power plants is presented. Multiagent systems technology and data mining techniques are combined to enhance the intelligence of the proposed application, mainly in two aspects: increase the reliability of input data (sensor validation and false measurement replacement) and generate new control monitoring rules. Various classification algorithms are compared. The performance of the application, as tested via simulation experiments, is discussed.

Key words: Intelligent agents, data mining, power plant

1 Introduction

The deregulation of the electric power and the continuously stricter environmental terms, imposed by international protocols, call for more efficient and more economic Thermal Power Plant (TPP) operation. In order to achieve optimum plant operation, electric companies embrace new, advanced control monitoring systems. The confrontation of the complexity of these new technologies and the profitable exploitation of the plethora of data produced by them necessitate the provision of intelligent tools that will assist the personnel to detect and to face the operational problems in real time.

In this direction companies as ABB and Metso renew their commercial applications mainly by adding tools that supervise the trends of operational parameters [1, 2]. At the same time, research groups investigate the possibility of applying artificial intelligence methods for the production of intelligent tools that will solve TPP operating problems [3–5].

However, the existing solutions in the literature usually are designated for specific plant technology (eg. boiler Benson or Sultzer, with/without low-NOx burners etc), rendering thus the adaptation expensive and time-consuming, if not impossible. Besides, most of the proposed methods are developed for the confrontation of particular problems and are not applicable to other cases [4, 5].

What is missing is a generalized framework for the design and development of intelligent applications for the support of the TPP operation control. Our research work introduces a framework for the formation of an adaptable and extendable system, that can correspond to all the TPP needs from the reduction of gas emissions to the timely breakdown avoidance. It combines latest advances in three research areas; knowledge

engineering, data mining (DM) and multi-agent systems (MASs). The strength of the proposed framework lies in the embedment of DM models into agents.

The rest of this paper is organized as follows. Section 2 discusses the main issues that the proposed application faces. The three research areas, which form the basis of our approach, are listed in section 3. Section 4 introduces the proposed methodology. The MAS architecture is outlined in section 5. Issues concerning the extraction of DM models are discussed in section 6, while the performance of the DM algorithms in section 7. Section 8 includes some evaluation points resulting from simulation experiments and section 9 the conclusions.

2 Main IPPAMAS issues

The proposed Intelligent Power Plant engineer Assistant Multi-Agent System (IPPAMAS) is an integrated tool that extends from the sensor data validation to the provision of information and indications to the power plant personnel. It initially evaluates and improves the reliability of the plant measurements in order to reassure that data used by the next stages are correct. Then it exploits the historical data by applying data mining techniques and combines the derived DM models with the current information, so as to produce knowledge and provide the adequate indications to the plant personnel. Finally, it diffuses the produced information to the appropriate users, in the suitable form, in the right time and place. Selective literature, along with the proposed approach, for these three main issues are given in the following paragraphs.

Sensor Validation and Estimation The evaluation of information on which the decision-making for an action is based, plays a particularly important role in the industry. Erroneous data can lead to critical situations because of wrong handlings. The values of the sensors, as thermometers, are considered as information in the case of power plants.

Typical approaches for the detection of incorrect readings of a sensor include the use of hardware redundancy and majority voting, analytical redundancy, and temporal redundancy [6]. Artificial Intelligence techniques have also been proposed for sensor validation and sensor values estimation [3, 7, 8]. Nevertheless, these various methods have serious drawbacks in practice (increased demands in human expertise and computer resources, time consuming etc).

At the current project it was decided to base the validation on rules that derive from the measuring equipment requirements, the plant operation specifications and the personnel experience [9]. As far as sensor values estimation is concerned, DM algorithms are applied for deriving models that estimate the value of one variable based on others used as input parameters. The estimated values can then be used instead of the recorded ones by a measuring instrument out of order. Software agents apply both the validation rules and the DM models during the on-line monitoring procedure.

Decision Support The potential contribution of intelligent MAS to decision support systems (DSSs) is appreciated as significant by various researchers [10, 11]. The MAS technology has been adopted in many research projects for supporting the users at the decision-making procedure [11, 12].

The main part of the presented MAS evaluates the current conditions on-line and provides the users with the appropriate indications. Also, it locates the cases that the installed commercial control monitoring system fails to produce the proper alarm signals and undertakes their activation (eg. when the measurement of a variable associated with an alarm is erroneous).

Information Flow The right information management and distribution plays a key role for an enterprise. MAS technology has been used for this aim in many applications in sectors as the power industry and logistics [13, 14]. A MAS for information management is able to incorporate and handle the heterogeneous, distributed sources of information in a TPP and to model the various users. Main advantages of the proposed system, as far as information fusion is concerned, are the ubiquity and the context-awareness, which are introduced [15].

3 Research Areas

Knowledge engineering Domain knowledge is considered to be one of the determinants in a large-scale KDD project [16]. It is important to mention that knowledge on the plant operation is not explicitly expressed by the persons who possess it, neither it is easily explained and standardized in books or handbooks. Knowledge engineering provided the scientific way of transforming the implicit knowledge of personnel into explicit, so as to exploit it for the development of the proposed application. In addition, it supplied the tools for modeling the problem and producing specifications comprehensible, hence easily applicable.

Data Mining The Knowledge Discovery in Databases (KDD) procedure was applied for the correlation of operational parameters for which there was no known relation to connect them (i.e. from thermodynamics, plant specifications etc.). Its advantages comparatively to conventional statistical methods are that it offers techniques for confronting the vast volume of historical plant operational data and that it does not prerequisite the determination of specific questions.

Multiagent System The control system of a TPP must meet increasingly demanding requirements stemming from the need to cope with significant degrees of uncertainty. MASs have recently emerged as a powerful technology to face the complexity of a power plant. Indeed, several experiences already testify to the advantages of using agents in control monitoring [17, 18]. The autonomy of MAS components reflects the intrinsically decentralised nature of modern distributed control monitoring systems. The flexible way in which agents operate and interact (both with each other and with the environment) is apt to the dynamic and unpredictable TPP operation. In addition, agents are most appropriate for developing context-aware and ubiquitous applications [15, 19], that will offer their services all over the plant premises and will be able to adapt to different situations, from normal operation to a crisis resolution. Finally, agents are most suitable for embedding DM models [20]. This was significantly important for the presented application, as its intelligence is mainly based on these models.

4 Methodology

For the design and development of the system, KE was chosen for modelling the problem, KDD for handling the enormous volume of operation data and facing the lack of known equation connecting them, and MAS for the confrontation of the complexity and the distributed nature of the system.

It was argued that DM techniques were adopted for enhancing the MAS intelligence by extracting knowledge from the vast amounts of operation data. However, a critical sector, as the power industry, cannot unquestioningly rely on automatically extracted statistical models, as there is the risk to come up with good statistical results that have no physical meaning. Such models would not reflect the actual operation, thus could lead to dangerous choices, if applied. In order to eliminate this risk, a KE method was utilized to capture domain knowledge for evaluating the appropriateness of the DM models. Furthermore, through KE the bounds of using DM models were set: prerequisites of a certain action consist one boundary and the conditions that necessitates it, the other one (Figure 1). The DM models were used to suggest on cases that lie in between. The MAS was designed appropriately to apply the rules and the models on-line.

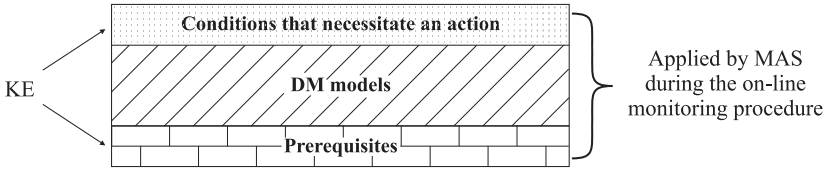


Fig. 1. Data mining application area

Our proposed methodology consists of the following five steps:

1. Concentration and formulation of information and implicit knowledge possessed by the plant personnel. Application of a KE method for modelling the plant and specifying the IPPAMAS thoroughly.
2. Application of the KDD on the plant operation historical data for extracting the most suitable models. The models are used for estimating the values of variables and for modelling actions.
3. Embedment of derived models in agents for the replacement of erroneous sensor measurements and for the calculation of optimum value of operation parameters.
4. Development of the MAS for the management of data, the application of models and rules and the configuration of appropriate indications to the TPP personnel.
5. Deployment of a Wireless Local Area Network (WLAN) combined with a (pre-existing or not) wired Ethernet LAN for the diffusion of the services allover the plant premises.

A KE methodology, CommonKADS, was chosen for the case of a TPP [21]. It offers a predefined set of models that together provide a comprehensive view of the project

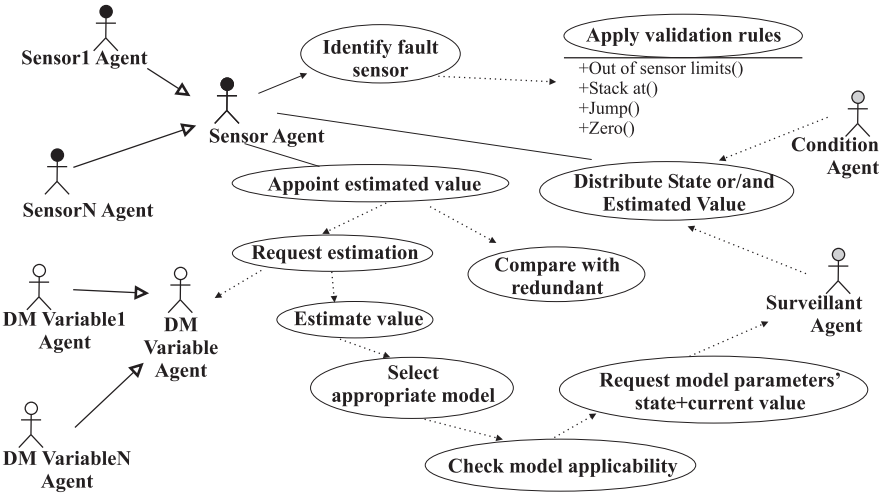


Fig. 2. Produce recommendations on control actions or parameters' regulation

under development. The decision to base the development of the system on MAS predicated the adoption of its extension MASCommonKADS, which adds a model capturing particular aspects of the agents [22].

For the application of the data preprocessing techniques and the classification algorithms, the widely used Waikato Environment for Knowledge Analysis (WEKA) was chosen [13, 24].

The Java Agent Development framework was used for the implementation of the MAS [25]. Particularly, for deploying the Data-Mining Agents the Agent Academy platform (version "Reloaded") was used [26]. A unique functionality offered by Agent Academy is the embedding of models that derive from the application of most of the classification algorithms available in WEKA. As it is implemented upon the JADE infrastructure, it is self-evident that the deployed agents are compatible with the remaining ones created with JADE.

5 MAS Architecture

The MAS architecture is structured in three layers: Sensors Layer, Condition Monitoring Layer and Engineer Assistant Layer. At each layers agents act in groups that correspond to different plant subsystems. There are also central agents that assure the consistency of solutions and the resolution of conflicts.

The Sensor Layer (1st) is responsible for the identification and reconstruction of sensor faults. It ensures that data entered to the application are valid (Figure 2).

The Condition Monitoring Layer (2nd) is responsible for the safe operation of the TPP and its optimization. In this phase, meaning is assigned to data to produce the appropriate information, as alarm signals and suggestions on handlings. Its main func-

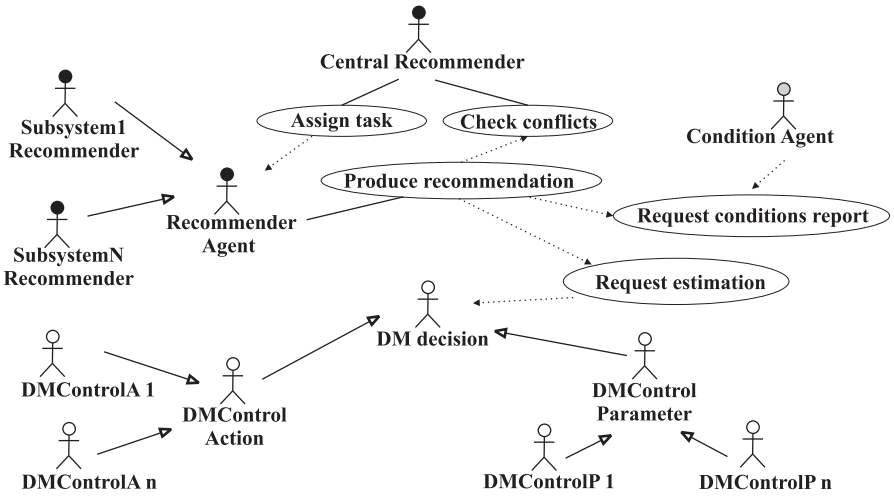


Fig. 3. Sensor validation and fault values replacement with estimated ones

tionalities are depicted in Figure 3. The activities that follow the 'Request Estimation' are as shown in Figure 2.

The Engineer Assistant Layer (3rd) distributes the information to the appropriate users. Information pieces are handled differently depending on the current operating conditions and the context of the users.

At each layer several agents cooperate for confronting the respective tasks. These agents extend one of the six basic agent types:

1. Variable Agent: identification of faulty sensors
2. Data-Mining Agent: DM models application
3. Recommender Agent: recommendations on actions
4. Condition Agent: alarms triggering
5. Distribution Agent: information distribution
6. User-Interaction Agent: information personalization

There are also four auxiliary agents that provide their services to all levels: Supervisor Agent, Sensor Surveillant Agent, Synchronization Agent and DBApplication Agent. In addition, Monitoring Agent is responsible for the MAS coordination and Trainer Agent for the MAS retraining.

6 DM models

DM models are embedded into DM agents that participate at the first two layers, as these concern the plant operation data. The extraction of models concerning the user profiles for the 3rd layer was also investigated, but rejected, as there is not significant space for improvement. The latter is mainly due to two reasons: 1) it is known from the beginning who the users will be (i.e. categories, particularities, skills, preferences

etc) and 2) the security specifications of a TPP restrict the degree of personalization allowed.

For each discrete case more than one DM model is extracted and stored at the application's repository. The respective DM agent chooses the most appropriate of them when requested on-line. These models derive from:

1. Applying different classification algorithms to the same dataset. This is done in order to have models selected with different criteria, as statistical metrics and existence of missing data or outliers. Depending on the current circumstances one of them might be more appropriate than the others.
2. Applying the same classification algorithm (the one presenting the best performance) to different datasets. These datasets are selected from the initial dataset by applying attribute selection methods. Having models with different input parameters offer the flexibility to apply each time the one whose parameters are all valid at the current moment.

At the Sensor Layer, one agent that extends the Sensor Agent type is created for each sensor. Obviously, this does not concern all the plant sensors; a part of them was chosen based on their significance for the plant operation, linkage to alarm signals and usage by the application itself. In the event of a false reading, an estimated value is used to replace the recorded value of the measurement instrument. The task of estimating a value is undertaken by an agent that extends the basic Data-Mining Agent. As depicted in Figure 2, the agent 'Selects the appropriate model' among the available ones and 'Checks the model applicability'

At the Condition Monitoring Layer, agents that extend the basic types DMControlAction and DMControlParameter apply DM models for advising on actions and for calculating the optimum value of controllable parameters (in order to tune a subsystem appropriately). Likewise at the 1st layer, two or more models for each examined case are stored to the application's repository.

The DM models used in the Condition Monitoring Layer were not derived from the whole data. The dataset comprised of only a part of the historical data that reflected the best system operation judged according to selected criteria. These criteria varies depending on the goals set for each subsystem. For instance, the criterion for the Flue-gas subsystem is the reduction of one or more of the flue-gas emissions (NOx, CO2 etc.).

7 DM Results

The KDD was followed for deriving the DM models. A detailed description of the stages proceeding DM is beyond the scope of this paper (see [9]).

Data were taken from the TPP Meliti of the Public Power Corporation, Greece. They concerned three plant subsystems. For each subsystem 80-100 variables were selected. The records covered one year of almost continuous operation. The datasets were split according to the operation mode (eg. full/low load). The one concerning full plant operation represented 70% of the data, approximately. It contained, after the cleaning,

Table 1. Statistical results of various classification algorithms applied for the modeling of the variable 'Flue gas temperature after the Economiser'. Mean value= 294.5 C

	Algorithm	Correlation Coefficient	Mean Absolute Error(C)	Relative Absolute Error(%)
Functions	Linear Regression	0.812	1.845	60.19
	Least Med Sq	0.560	2.068	67.47
	MultilayerPerceptron	0.994	0.363	11.86
	Simple Linear Regression	0.630	2.330	75.83
	RBF Network	0.786	1.65	53.95
	SMOreg	0.772	1.649	53.79
	Pace regression	0.772	1.642	53.65
Lazy	IB1	0.987	0.167	5.45
	IBk (k=2)	0.988	0.166	5.42
	Kstar	0.998	0.017	0.04
	LWL	0.732	2.111	68.86
Rule	Conjuctive Rule	0.784	2.016	65.77
	Decision Table	0.993	0.173	5.64
	M5Rules	0.992	0.226	7.38
Tree	Decision Stump	0.698	2.225	72.57
	M5P	0.997	0.171	5.59
	REPTree	0.992	0.170	5.55
Meta	Additive Regression (Decision Stump)	0.882	1.441	47.00
	Bagging (Decision Stump)	0.699	2.221	72.43
	Regression By Discretization (J48)	0.993	0.229	7.48
	CVParameterSelection (REPTree)	0.994	0.165	5.54

more than 360000 instances. These were unequally distributed to 15 files of different length.

The example presented in this paper concerns the Flue-gas subsystem of the boiler. The original dataset for this specific subsystem comprised of 88 variables (temperatures, pressures, flows, etc.), which were reduced to 44 by applying aggregation techniques. In an effort to further reduce the input parameters, attribute selection filters were tested [9].

Initially, we experimented with a range of classifiers, diversifying their parameters and the combinations with pre-processing techniques. Table 1 includes three statistic metrics for the performance of 21 classifiers (implementation was provided by WEKA), when applied to a dataset comprised of 10252 rows of data for modeling the variable 'Flue-gas temperature after the Economiser' (i.e. at the exit of the boiler).

For comparison reasons the statistics listed were the best ones that resulted from the application of each algorithm after modifying the initial parameters values set in

WEKA. Some of them performed significantly better (or worse) with modified parameters, while the performance of others was not affected considerably by this. It should be noted that the performance of the meta-algorithms was judged in comparison with the performance of the base algorithm on top of which they were applied.

Five the algorithms were selected for the remaining of the experiments based on their statistics and the response time. The REPTree gave satisfactory results in the majority of the cases (Figure 4a). It builds a regression tree using information gain [24]. It had minimum execution times, so it was used for evaluating the selected datasets and the appropriateness of the various preprocessing techniques. Next in line came M5Rules and MultilayerPerceptron that had good performance, but were time consuming. M5Rules generates a decision list for regression problems using separate-and-conquer (it is based on the M5 algorithm) [29]. MultilayerPerceptron is neural network which uses backpropagation to train [30]. Finally, the algorithms IBk (k=2) and Kstar (K*) usually demonstrated the best performance (Figure 4c). They both belong to the category of instance-based classifiers. IBk is a k-nearest neighbours classifier that normalizes attributes by default [27]. K* differs from other instance-based learners in that it uses an entropy-based distance function [28].

However the requirements in computer resources of the last four ones discouraged or even stopped their application, depending on the dataset dimensions. For these reasons it was decided to apply the REPTree at the beginning for discarding the datasets and then apply all five of them for extracting the final models. As aforementioned, we decided to use different algorithms for extracting the DM models because each of them handles effectively different cases, i.e. missing data, outliers etc. Consequently, depending on the case, there is always a proper model available and the whole system responds effectively. The above-mentioned are equally important for the retraining procedure. Indicatively, the performance of some algorithms, including IBk, is affected by the presence of missing data, while REPTree deals quite efficiently with them (Figure 4).

Apart from the performance metrics, the results were also evaluated based on their physical meaning (correlation of variables). This also involved knowledge interpretation by domain experts. For this purpose, adequate knowledge representation formalism was used (eg. rules defined by a decision tree).

8 Evaluation

The performance of IPPAMAS was tested through simulation of several cases. Several possible problems were simultaneously introduced in the simulation, as the non-scheduled termination of an agent, presence of false sensor measurements, etc. The MAS deployed for the evaluation comprised of 78 agents and concerned three boiler subsystems. The simulation studies revealed several issues with respect to the service prototype. Three are the points that are directly related with the embedment of DM models in agents:

1. Response time. In all cases the MAS corresponded successfully within the predefined time period of 1 minute.

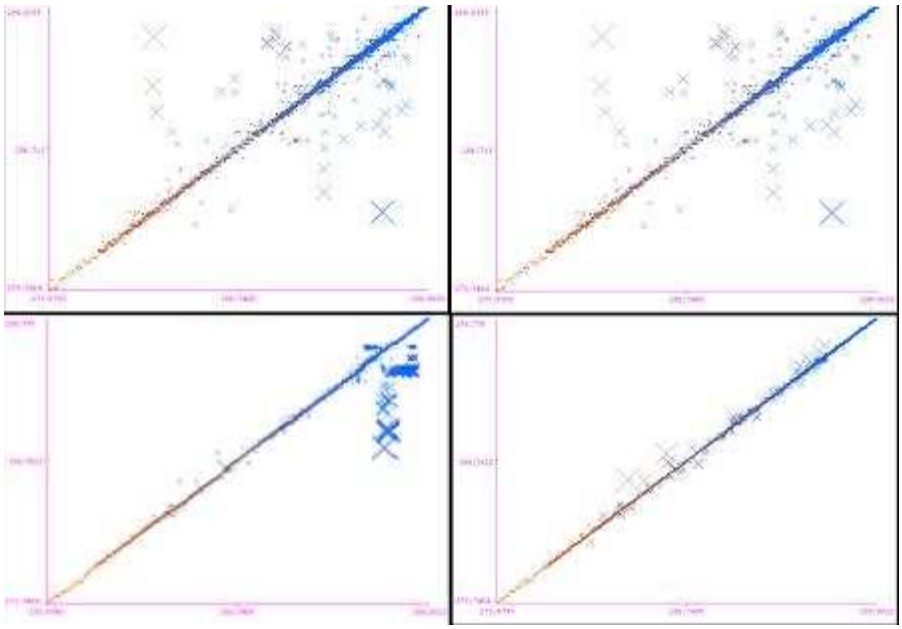


Fig. 4. The performance of two classification algorithms applied for modelling the variable Flue gas Temperature after the Economiser: a-REPTree, b-REPTree (missing values), c-IBk, d-IBk (missing values)

2. Accuracy. The estimated values were within the acceptable fault tolerance as predefined by the experts (eg. 1C for the Flue gas Temperature after the Economiser).
3. Reliability. The MAS proved that it can handle many simultaneous cases and that it produces the same output for the same input.

With respect to the content, the indications for actions were found appropriate in the majority of cases (70%). The remaining were considered unnecessary or overdue by the plant personnel, leading to the conclusion that either more data are needed for the training of the system or the specifications (predefined rules etc) are not enough. However, it was pointed out that the percentage of unsuccessful indications might be reduced when more subsystems will be monitored by the application, as it will have an overall picture of the operation (most of the subsystems are interconnected). The appropriateness of the DM models, which was already implied by the statistics metrics, was proved through the simulation. The agents succeeded in choosing the most suitable model depending on the circumstances. They also accomplish to offer solutions and to overcome problems by following alternative scenarios.

9 Conclusion

In this paper, a MAS and DM techniques are combined in order to reproduce effectively the complex power plant operation, which is difficult to model otherwise. This combi-

nation increases the adaptability and the extensibility of the system; the extraction of new DM models based on new operation data is sufficient for capturing changes that concern the TPP operation (caused by wear, maintenance, replacement, or even addition of new mechanical equipment). The DM models enhance the MAS intelligence so as to successfully replace false sensor measurements and to produce the appropriate indications for actions to the plant personnel. Through the MAS the DM models are exploited so as to function not solely as a tool for calculations, but as a support tool for the personnel.

The short run plans include the design and configuration of more simulation experiments that may reveal more points that call for attention and on the same time give rise to innovative ideas.

Finally, as future work is concerned, an interesting topic for research would be the application of DM techniques to the application data of the MAS, in order to improve the performance of the agents.

References

1. ABB Group, Products & Services, `http://www.abb.com/ProductGuide`
2. Metso, `www.metsoautomation.com`
3. D. Flynn, ed.: Thermal Power Plant Simulation and Control. IEE, London (2003)
4. Hadjiski, M., Boshnakov, K., Christova, N., Terziev, A.: Multi Agent Simulation in Inference Evaluation of Steam Boiler Emission. In: 19th European Conference on Modeling and Simulation, pp. 552-557. Riga, Latvia (2005)
5. Ma, Z., Iman, F., Lu, P., Sears, R., Kong, L., Rokanuzzaman, A.S., McCollor, D.P., Benson, S.A.: A Comprehensive Slagging and Fouling Prediction Tool for Coal-Fired Boilers and its Validation/Application. Fuel Process. Technol. 88, 1035-1043 (2007)
6. Frank, P.: Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge Based Redundancy- a Survey and Some New Results. Automatica 26, 459-470 (1990)
7. Eryurek, E., Upadhyaya, B.R.: Sensor Validation for Power Plants Using Adaptive Backpropagation Neural Network. IEEE Trans. Nucl.Science 37, 1040-1047 (1990)
8. Ibarguengoytia, P.H., Vadera, S., Sucar, L.E.: A Probabilistic Model for Information and Sensor Validation. The Computer Journal 49(1), 113-126 (2006)
9. Athanasopoulou, C., Chatziathanasiou, V.: Intelligent System for Identification and Replacement of Faulty Sensor Measurements in Thermal Power Plants (IPPAMAS: Part 1). Expert Systems With Applications 36, 8750-8757 (2009)
10. Shim, J.: Past, Present, and Future of Decision Support Technology. Decision Support Systems 33(2), 111-126 (2002)
11. Vahidov, R.: Intermediating User-DSS Interaction with Autonomous Agents. IEEE Trans. on Systems, Man, and Cybernetics 35(6), 964-970 (2005)
12. Gao, S., Xu, D.: Conceptual Modeling and Development of an Intelligent Agent-Assisted Decision Support System for Anti-money Laundering. Expert Systems with Applications, 36, 1493-1504 (2009)
13. Lucas, C., Zia, M.A., Shirazi, M.R.A., Alishahi, A.: Development of a Multi-agent Information Management System for Iran Power Industry-A Case Study. In: Power Tech 2001 Proceedings vol.3. IEEE, Porto (2001)
14. Pechoucek, M., Marik, V.: Industrial Deployment of Multi-agent Technologies: Review and Selected Case Studies. Auton Agent Multi-Agent Syst 17, 397-431 (2008)

15. Athanasopoulou, C., Chatziathanasiou, V.: Prototype For Optimizing Power Plant Operation. In: Mangina, E., Carbo, J., Molina, J. (eds.) Agent-based Ubiquitous Computing. Atlantis press (2009)
16. Kopanas, I., Avouris, N.M., Daskalaki, S.: The Role of Domain Knowledge in a Large Scale Data Mining Project. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) SETN 2002. LNAI vol.2308, pp.288-299. Springer, Berlin (2002)
17. Arranz, A., Cruz, A., Sanz-Bobi, M.A., Ruiz, P., Coutino, J.: DADICC: Intelligent System for Anomaly Detection in a Combined Cycle Gas Turbine Plant. Expert Systems with Applications 34, 2267-2277 (2008).
18. Mangina, E.: Application of Intelligent Agents in Power Industry: Promises and Complex Issues. In: Marik, V., Muller, J., Pechoucek, M.(eds.) Multi-Agent Systems and Applications III. LNAI, vol.2691, pp.564-574 Springer, Berlin (2003)
19. Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L., Crowley, J.: Agent Based Middleware Infrastructure for Autonomous Context-Aware Ubiquitous Computing Services. Computer Communic. 30, 577-591 (2007)
20. Symeonidis, A., Mitkas, P.A.: Agent Intelligence Through Data Mining. Springer, New York (2005)
21. Shreiber, G.,Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., Wielinga, B.: Knowledge Engineering and Management: the CommonKADS methodology. MIT Press, Cambridge, Massachusetts (2000)
22. Iglesias, C.A., Garijo, M.: The Agent-Oriented Methodology MASCommonKADS. In: B. Henderson-Sellers, P. Giorgini (eds.) Agent-Oriented Methodologies, pp. 46.78. IDEA Group Publishing (2005)
23. WEKA, `http://www.cs.waikato.ac.nz/~ml/weka/index.html`
24. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques (2nd ed.). Morgan Kaufmann, San Francisco (2005)
25. JADE, `http://jade.tilab.com`
26. Agent Academy, `https://sourceforge.net/projects/agentacademy`
27. Aha, D.W., Kibler, D., Albert, M.: Instance-Based Learning Algorithms. Machine Learning 6, 37-66 (1991)
28. Cleary, J., Trigg, L.: K*: An Instance-Based Learner Using an Entropic Distance Measure. In: 12th Inter.Confer. on Machine learning, pp. 108-114 (1995)
29. Hall, M., Holmes, G., Frank E.: Generating Rule Sets from Model Trees. In: 12th Australian Conference on Artificial Intelligence, pp. 1-12. Springer-Verlag (1999)
30. Bishop, C.M.: Neural Networks for pattern recognition. Oxford University Press, New York (1995)

A sequence mining method to predict the bidding strategy of trading agents

Vivia Nikolaidou and Pericles A. Mitkas

Aristotle University of Thessaloniki,
Department of Electrical and Computer Engineering,
Thessaloniki, Greece
+30 2310 996349 , +30 2310 996390
vivia@ee.auth.gr , mitkas@eng.auth.gr

Abstract. In this work, we describe the process used in order to predict the bidding strategy of trading agents. This was done in the context of the Reverse TAC, or CAT, game of the Trading Agent Competition. In this game, a set of trading agents, buyers or sellers, are provided by the server and they trade their goods in one of the markets operated by the competing agents. Better knowledge of the strategy of the trading agents will allow a market maker to adapt its incentives and attract more agents to its own market. Our prediction was based on the time series of the traders' past bids, taking into account the variation of each bid compared to its history. The results proved to be of satisfactory accuracy, both in the game's context and when compared to other existing approaches.

1 INTRODUCTION

On-line auctions, especially the ones operated by autonomous agents, are a relatively new area of research, while still being extensively used in the real world. The subject is both wide and complex enough to guarantee an ample research potential, which is combined with the possibility to immediately deploy the obtained results to real applications. On the other hand, the information explosion during the last decade has yielded vast amounts of data, out of which useful results have to be extracted. Combined with the ever increasing processing power available, it has given a boost to the research of Data Mining techniques.

One such technique is Sequence Mining, which has generated considerable interest mainly due to its potential for knowledge extraction in large sets of sequential data. Sequence mining tries to identify frequent patterns in datasets where data items appear in a somewhat predictable fashion, such as text, aminoacid chains, or a sequence of process calls in a computer. Traditional data mining techniques, such as classification and clustering, cannot be directly applied but can be modified for sequence mining. Alternatively, sequential data can be transformed to a form that is suitable for processing by one of the existing algorithms.

The wide adoption of intelligent agents in research and practice is showing more and more examples where they are called to extract knowledge out of data, sequential or not. Agent developers can also exploit large data repositories by extracting knowledge models that can be embedded into intelligent agents [11]. Electronic auction environments

represent the biggest application domain of software agents. It is in this domain where the successful blending of Intelligent Agents and Data Mining proves fruitful for both of these technologies.

In this work, we try to predict the bidding strategy of a large set of trading agents in the CAT game of the Trading Agent Competition [4]. Since the prediction is mostly based on analyzing the bids' history, we employ a novel Sequence Mining technique, which watches the variation of the bids according to their history, in order to classify the bid sequences. The architecture of the agent, for which we designed our technique, is also presented in this context.

The rest of this paper is organized as follows: Section 2 analyzes the bidding strategies that we were called to distinguish, while Section 3 presents the CAT game that was used as a framework. Section 4 explains the architecture of our agent, Mertacor, which incorporated the resulting model. Section 5 focuses on the challenging issues of Sequence Mining and subsequently analyzes our approach to this issue, eventually arriving to the results' presentation. Section 6 reviews existing literature approaches to similar issues. The work concludes with Section 7, which discusses the results and gives clues for future works.

2 BIDDING STRATEGIES

Intelligent agents, due to their autonomy and their reasoning capabilities, have long been used in auctions as bidding entities with great success. They can easily be designed to implement various bidding strategies and to adapt them, if needed. Agents can also make timely decisions that require complex computations. Although several more bidding strategies exist, the following ones are adopted in the CAT game.

2.1 Zero-Intelligence Constrained (ZI-C)

One of the most well-known strategies is the Zero-Intelligence trading strategy, first developed by Gode and Sunder [6]. This strategy randomly selects a bid based on a uniform distribution, without taking into account any market conditions or seeking any profit, hence the term Zero-Intelligence. In order to avoid possible loss, they define the Zero-Intelligence Constrained (ZI-C) strategy, used in the CAT game. This strategy sets the item's cost value as a minimum boundary to the bidding price, whereas the maximum value remains the same as in ZI-U.

2.2 Zero-Intelligence Plus (ZIP)

Since the ZI-C strategy, described in 2.1, draws bids randomly, the efficiency achieved is not enough to reach the efficiency of markets with human trading agents. As a result, Cliff in [2] introduced the ZIP strategy, in which the trading agents increase or reduce their profit margin by watching the market's conditions. More specifically, they monitor the following values: all bid and ask prices in the market, irrespective of whether the corresponding shouts led to a transaction or not, as well as the transaction price itself. The agents then adjust their profit margin accordingly.

At the beginning of a trading day, all ZIP agents have an arbitrarily low profit margin. When a transaction occurs that indicates that they could acquire a unit at a more convenient price, their profit margin is increased. However, ZIP agents have a back-up strategy, which prevents them from raising their profit margin too high. When a buyer's shout gets rejected, the shout price is increased and, when a seller's shout gets rejected, the shout price is decreased. Similarly, they watch transactions made by competing sellers and lower their profit margin if needed, so as to not be undercut by competing sellers or buyers.

2.3 Gjerstad-Dickhaut (GD)

Gjerstad and Dickhaut in [5] defined a bidding strategy based on belief functions, which indicate how likely it is that a particular shout will be accepted. This is achieved by watching the history of observed market data - namely, the frequencies of submitted bids and asks, as well as the frequencies of bids and asks that lead to a transaction. Since more recent bids and asks are of higher importance than older ones, the authors introduce a sliding window function in the agents' memory, that only takes into account the latest L shouts in the market's history.

Their belief functions are based on the assumptions that, if an ask is accepted, all asks at a lower price will also be accepted and, if an ask has been rejected, all asks at a higher price will also be rejected. Similarly, if a bid is accepted, all bids at a higher price will also be accepted and, if a bid is rejected, all bids at a lower price will also be rejected.

2.4 Roth-Erev (RE)

Roth and Erev's purpose was to create a strategy that would mimic the behavior of human players in games with mixed strategy equilibria [3]. Therefore, they used reinforcement learning algorithms on the agent's profit margins, in order to adjust them to the market's conditions.

This strategy only depends on the agent's direct feedback with the market mechanism and is therefore independent of the auction mechanism itself. More specifically, both ZIP and GD require the trading agents to have access to the history of bids and asks, as well as all accepted transactions and their prices. However, RE does not require any such data, but only relies on the same agent's interaction with the market mechanism. Therefore, it is generic enough to be used in any auction environments.

3 REVERSE TAC ("CAT") GAME

The CAT game is being held in the context of the Trading Agent Competition or TAC [4]. Since the markets in this game are not fixed, but instead created by each competing agent, it is called Reverse TAC, or CAT. The name CAT also refers to Catallactics, the science of economic exchange.

In the CAT game, a set of trading agents is generated by the game itself, while the contestants' purpose is to design specialist agents. Each specialist agent will operate a

single market and set the rules for it. Trading agents are free to choose only one market in each operating day, and they can only buy and sell in the market that they choose.

All trading agents are either buyers or sellers and remain so for the entire game. They all buy or sell the same item in single-unit auctions. However, they are allowed to trade several items per day, placing a new bid or ask after a completed transaction.

The trading agents, buyers or sellers, incorporate one of the four trading strategies described in Section 2. Their private values, or estimations of the value of the goods traded, are drawn from a random distribution. They also incorporate a market selection strategy, which helps them choose the market that they judge to be most profitable to them. The traders' private values, bidding strategies, market selection strategies, as well as trade entitlement (the number of items that they are allowed to trade each day) are unknown to the specialists.

The specialists' goal is to design the rules and conditions of their markets. More specifically, they have to define:

a) The agent's charging policy. Agents announce their fees at the beginning of each day. They may include fees for one or more of the following:
   - ⋆ Registration fee: paid by each trading agent who registers on this particular market on this day
   - ⋆ Information fee: paid by each trading agent and each other specialist who requests information on the market's shouts and transaction
   - ⋆ Shout fee: paid for each shout placed
   - ⋆ Transaction fee: a standard sum paid for each transaction by each agent involved
   - ⋆ Profit fee: a percentage on the transaction's price, paid by both the buyer and the seller

b) The market's accepting policy. This policy defines which shouts, bids or asks made by the traders are accepted and which are rejected.

c) The market's closing condition. Although a trading day consists of a number of trading rounds, announced at the beginning of the game, the market does not have to close at the end of each round, but instead is free to close at any time during the trading day.

d) The matching policy for the shouts – more particularly, which bid is matched to which ask, in order to form a successful transaction.

e) The pricing policy for each transaction. Each transaction closes at a different price, same for both the buyer and the seller.

The game consists of an unknown number of days, given in the server's configuration file but not announced. Each day consists of a number of trading rounds, with each round having a certain duration. At the beginning and at the end of each day, as well as between each day, there is some free time in order for agents to complete any possible calculations. All these durations, apart from the game's duration, are announced at the beginning of the game.

Scoring is only made during assessment days, which start and end at some random point in the game's duration. The randomness of these days, as well as the game's duration being unknown, have as a purpose to avoid exploiting initial and final conditions by

the specialists, but instead focus on constructing stable markets that are able to function properly at any duration.

Scoring on each trading day and for each specialist is determined by the sum of the following three factors:

i. The agent's profit for each day, divided by the total profit of all agents for the day, in order to be normalized on a scale from 0 to 1.
ii. The agent's market share, as in, the ratio of traders subscribed to this particular market on this day, again normalized in a scale from 0 to 1.
iii. The agent's transaction rate, as in, the ratio of shouts accepted that led to a successful transaction. This is again a number from 0 to 1. Rejected shouts are not calculated.

The interested reader can find more information on the CAT game in [4].

4 AGENT MERTACOR

Agent Mertacor employs a modular architecture, as shown in Figure 1. Using a combination of microeconomics theory and heuristic strategies, it tries to maximize its profit without losing a significant portion of market share. The main parts of the agent are briefly discussed below.

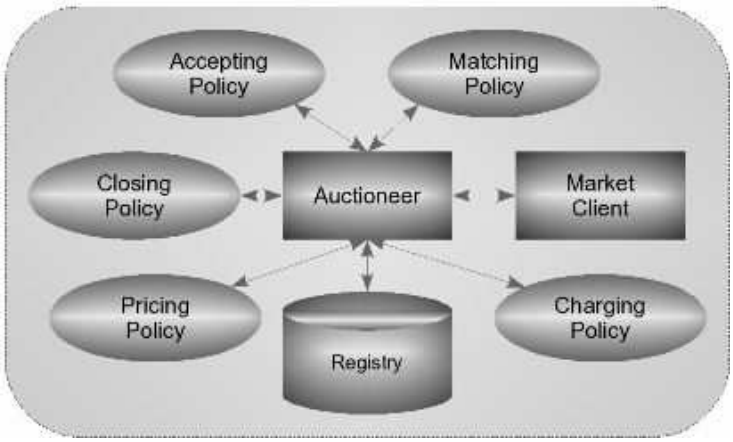


Fig. 1. Mertacor's architecture

4.1 Auctioneer

The Auctioneer is the central part in Mertacor's architecture. It is in charge of coordinating the other parts of the agent, especially when it comes to communication between

the Market Client, the Registry and the policy modules of the agent. It is also in charge of computing the global equilibrium point, based on bid history. The equilibrium point is the price where the curve of offer distribution, sorted from lowest to highest, meets the curve of demand distribution. The term "global" refers to the fact that all agents are taken into account for this computation, whether they belong to the agent's particular market or not.

4.2 Market Client

The Market Client is the element that is responsible for the communication with the server. It transforms all information from the internal representation structure of the agent to the form that is understandable by the server and back. It is also responsible for subscribing to the markets of competing trading agents, in order to receive information on shouts and transactions made in their market. The reason behind this is that, normally, a specialist only has information on other specialists' profit and market share at the end of each day. Further information can only be acquired by subscribing to the other specialist's market, paying the appropriate information fee, if set.

4.3 Accepting Policy

The accepting policy is a tricky part in the agents' design. It must be low enough to keep a high transaction rate, but still high enough to guarantee many transactions. Mertacor's accepting policy is divided into two parts.

The first part is activated at the beginning of the game, before the global equilibrium point is calculated. At this point, Mertacor implements the same rule used by NYSE: namely, in order for a shout to get accepted, it must be at a price better than the day's best shout, or the day's "quote".

As soon as the global equilibrium point is calculated, Mertacor switches to an equilibrium-beating accepting policy. This means that, in order for a shout to get accepted, it must be better than the global equilibrium computed. This works as an attraction to intra-marginal traders (buyers and sellers whose private values are, respectively, higher and lower than the global equilibrium), while keeping extra-marginal traders (buyers and sellers whose private values are, respectively, lower and higher than the global equilibrium) from trading goods.

4.4 Matching Policy

Mertacor uses the matching policy described by Wurman et al in [14]. Incoming unmatched bids and asks are sorted to two separate heaps. When a bid is matched with a shout, they are automatically moved to two separate "matched" heaps. When the market is closed, the highest bids are matched with the lowest asks. This method maximizes both social welfare and Mertacor's profit margin.

4.5 Clearing Condition

Mertacor's condition for clearing the market is again twofold. In the first phase of the game, before the global equilibrium point is computed, Mertacor behaves like NYSE, clearing the market at a given probability after each shout. This method, called "continuous clearing", has been proven to be adequately efficient, while keeping a high transaction throughput [6].

After the global equilibrium point calculation is completed, Mertacor switches to a modified round-clearing condition. This means that the market is closed at the end of each round. The variation used is that Mertacor switches again to a continuous clearing policy towards the end of the game, in order to maximize throughput.

4.6 Pricing policy

Like the accepting policy and the clearing condition, Mertacor's pricing policy is also divided into two phases: before and after the global equilibrium point is computed.

During the first phase, Mertacor uses a variation of a discriminative k-pricing policy [10]. The k parameter is computed as the ratio of sellers in this game, resulting in a k of 0.5 for a balanced market. Our policy slightly favors sellers when there are more buyers and favors buyers when there are more sellers, giving them an incentive to balance the market.

In the second phase of the game, Mertacor switches to a global equilibrium pricing policy. This means that the price for all transactions is the same price as the global equilibrium. This way, Mertacor gives each trader the same profit that they would gain in an efficient global partitioning.

In both cases, though, Mertacor favors intra-marginal traders at the expense of extra-marginal ones. This means that, if a transaction is executed between an intra-marginal trader and an extra-marginal one, Mertacor will clear it at the price given by the intra-marginal trader.

4.7 Charging Policy

Choosing the right charging policy is the most challenging task in the design of a CAT agent. Setting the fees charged too low may not yield enough revenue, while values that are too high may discourage traders from joining the specialist's market.

As explained in Section 3, each specialist may impose five different fees: a) registration fee, b) information fee, c) shout fee, d) transaction fee, and e) profit fee.

Mertacor only imposes the profit fee, keeping the other ones down to zero. This is decided keeping in mind that only successful transactions must lead to a payment – in other words, an agent who does not earn anything should not pay anything either.

Heuristic experiments showed that the optimal value (i.e., the value which maximizes the specialist's score) is 0.2. This can go as low as 0.1, whenever the specialist's market share is too low, but as high as 0.3, when the specialist estimates that the market conditions are suitable.

5 PREDICTING THE BIDDING STRATEGY

In order to classify each trading agent as intra-marginal or extra-marginal, we needed to know its private value. The most crucial step in determining it was to find the agent's bidding strategy, which had to be predicted by observing its bidding history.

5.1 Modeling the Problem

The problem of predicting the bidding strategy of an agent can be seen as a classification problem, where each agent has to be assigned one of four labels (ZI-C, ZIP, GD or RE). However, it differs from conventional classification problems, in that the most characteristic input data is the past history of bids. With each bid having a specific timestamp and being correlated to its past bids, the problem is identified as a sequence mining one, where various time series have to be matched into labels.

Sequence mining is of particular interest because of various reasons. The first is the high dimensionality of the problem. More particularly, in our case, the history of past bids can comprise several hundreds or even thousands of samples, depending on the game's duration. One can easily understand that it is necessary to reduce the problem's dimensionality, allowing for better performance of the classification algorithms.

However, high dimensionality is but the tip of the iceberg. In many cases, including ours, the number of dimensions is not known a priori, but instead continuously grows according to the bids history. For example, during the first trading days of the game, we often have fewer than ten past samples. However, as the game advances and bids are constantly added to the history logs, the time series grows longer.

The third and most important issue in sequence mining is the notion of "sequence". This means that it is not merely a set of incoming value-timestamp pairs, but each value and each timestamp is related to its history. In fact, most information can be deduced by looking at the bid history and not at a bid itself, since, for example, ZI-C could give practically any bid value observed separately. On the other hand, conventional classification or clustering algorithms treat each feature as independent of the other ones and do not look for relationships between features. This means that it is not easy to examine each bid's history.

The most common approach in bibliography so far, for example by Martinez-Alvarez et al [9], is to use a moving window to split the time series and feed the latest N samples to the classifier. The authors take this approach one step further by normalizing input data. However, this still does not provide a satisfactory solution to the third issue described above, since each sample is still treated independently in the algorithms.

5.2 Our Approach

In our approach, we take this philosophy one step further by introducing deltas to the classifier. More specifically, let us consider a set of prices $p_0 ... p_n$, with p_0 representing the most recent one, and their corresponding set of timestamps, $t_0 ... t_n$. Notice that increasing subscript values in the sequence denote earlier points in the timeline. We define a sliding window of size 6, but only the first value p_0 is fed as-is to the classifier's input. For each of the rest of the values p_1 to p_5, we subtract the previous one in the

series (more recent in the bid history). The results, d_0 to d_4, give the classifier the notion of sequence, therefore facing the corresponding issue.

Furthermore, we have four additional features as the classifier's inputs. These are calculated by subtracting, for each one of d_1 to d_5, the value of the second previous sample, d_0 to d_4 accordingly.

Since the difference in time is more independent statistically than the difference in bid value, we only define a simple sliding window of size 3 when it comes to time differences. This means that we have three time-related attributes: t_0, t_0 - t_1, and t_1 - t_2.

The input dataset is completed by adding two more attributes independent from the time series, namely the type of the trader (buyer or seller), as well as whether this particular bid led to a successful transaction.

Since agents have limited time available for calculations while the game is running, the model was built offline. Classification was continuous, with the traders being assigned a label multiple times during the course of the game. In the case of a wrong prediction, traders' private value would be estimated wrongly, which might eventually lead to their misclassification as intra-marginal or extra-marginal.

Training data was initially taken from our own experiments with the CAT platform. This was used in the first version, designed for the qualifying rounds of the game. However, for the final games, we retrained the model using values from the qualifying games only. Only qualifying games 2 and 4 were used. The reason behind this is that games 1 and 3 were not run at full-length, but instead lasted only 100 trading days, which is only barely enough for the traders to finish exploring the markets and settle to their preferred one. While the number of trading days is unknown due to the game specifications, it was hinted that they would last much longer. In any case, since our model's inputs only depended on the data so far, the algorithm was designed to perform well in full-length as well as slightly shorter games.

Table 1. Summarized classification results

Algorithm	Correctly Classified	Incorrectly Classified	Kappa Siatistic	Mean abs. error	RMS error
Bagging - J48	65.55%	34.45%	0.54	0.24	0.34
J48	56.47%	43.53%	0.42	0.23	0.44
SMO	29.22%	70.78%	0.03	0.36	0.46
Perceptron	45.45%	54.55%	0.26	0.33	0.41

Algorithm Selection. The next step was the selection of the most suitable algorithm. This is done in a manner similar to [12], but skipping the data preprocessing part. We compared the following three algorithms: Neural Networks, J48 and Support Vector Machines. We used the WEKA platform [13] to train the model, using 10-fold cross validation.

We eventually decided to discard Support Vector Machines because of its slow response and inadequate performance for the specific input data. Time was an important issue, since the trading rounds are of fixed duration. Additionally, the classification accuracy did not exceed 30%. Meta-classification might have slightly improved it, but it might have been at the expense of complexity.

Of the remaining two algorithms, J48 outperformed Neural Networks, giving a classification accuracy of 56.5 percent. Confirming our findings in [12], meta classification, in the form of Bagging, further enhanced the model's performance, boosting it to 65.5 percent. Final results are presented in Table 1.

Tables 2 amd 3 presents the detailed accuracy by class, as well as the confusion matrix, for the winning algorithm, the combination of Bagging and J48.

Table 2. Detailed accuracy by class for winning algorithm

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.65	10.3%	68.0%	0.66	0.67	0.87	GD
0.72	13.7%	65.7%	0.73	0.69	0.89	ZIC
0.61	16.2%	58.8%	0.61	0.60	0.83	ZIP
0.61	6.2%	72.8%	0.62	0.67	0.88	RE

Table 3. Confusion matrix for the winning algorithm

GD	ZIC	ZIP	RE	$<--$ classified as
4936	536	1296	739	GD
326	**5787**	1533	107	ZIC
1210	1562	**5034**	417	ZIP
782	926	700	**3909**	RE

The Effect of Difference Order. In our dataset, we used differences of first and second order, which are a rough representation of the first and second derivative. In order to further illustrate the effect of difference order in the results, we constructed three datasets: price1, price2 and price3, which contain the differences of first, first and second, and first to third order, respectively. The chosen algorithms were run on all three datasets and the comparative results are depicted in Table 4.

We observe that the dataset which contains the second-order difference has the highest average classification accuracy on this dataset. For the highest-performing algorithm, namely the combination of Bagging and J48, we observe a slightly better performance for the price3 dataset. However, since all differences are computed on the fly

Table 4. Classification accuracy by algorithm and difference order

	price1	price2	price3
Perceptron	43.37	45.45	45.00
SMO	29.26	29.22	29.25
J48	56.66	56.47	55.76
Bagging – J48	65.21	65.55	65.67
Average	48.63	49.17	48.92

during the course of the game in order to obtain a correct classification, we decided to use the price2 dataset, which also gives the highest average performance and needs less computational resources than price3.

6 RELATED WORK

The most relevant work in bibliography is the one of Gruman and Narayana in [7]. They compared the performance of Support Vector Machines (SVMs) and Hidden Markov Models (HMMs). Their experiments were made in benchmark-like tests, using traders and specialists of controlled variations. They obtained a classification accuracy ranging from 52 to 62 percent using HMMs, according to fine-tuning of the HMM's parameters. Our approach not only outperforms it, but it is also implemented in "real-game" instead of controlled conditions. Furthermore, in contrast to HMMs, the decision-tree-based J48 hardly requires any fine-tuning, making it more robust when tested in the real world.

Bapna et al in [1] attempt to predict trading agents' Willingness-To-Pay in auction environments. When it comes to classifying agents' bidding strategies, they assume a single bidding strategy for all traders, namely the MBR (Myopic Best Response) strategy, and classify traders as Evaluators, Participators and Opportunists, as well as MBR and non-MBR traders. Details on the classification method, the input data, as well as classification accuracy, are not given, since the work's focus is mainly on Willingness-To-Pay and trader classification is merely used as an intermediate step.

When it comes to sequence classification, one example is the work by Lesh et al in [8]. They present an algorithm which examines all features of the given sequence and selects the ones to be used as input to traditional classification algorithms, such as Na?ve Bayes or Winnow. Zaki enhances this work in [15] by taking into account issues, such as length or width limitations, gap constraints and window size. This approach has increased complexity, but still does not take into account the continuity of the time series.

The sliding window approach prevails in more recent works, such as Martinez-Alvarez et al in [9]. They have a sequence clustering problem, in which they normalize the input data and subsequently determine the optimal window size. Taking into account the similarities between classification and clustering, this approach further supports the novelty of our work taking into account deltas instead of a sliding window, even normalized.

7 CONCLUSIONS AND FUTURE WORK

We have used sequence mining, a data mining technique, to improve the decision mechanism of an agent. Using this technique, a specialist agent can successfully predict the bidding strategy of a set of trading agents. Our model was designed as part of a widely accepted, general-purpose trading agent competition game. Nevertheless, it remains generic enough to allow its adaptation to a multitude of generic auction environments with little or no modification.

The classification accuracy proved to be satisfactory, since our agent was able to correctly classify a large majority of the traders. Using J48, an algorithm based on decision trees, also had the advantage of quick performance, allowing Mertacor more time for other calculations.

Mertacor proved competitive by ranking fifth in the final games out of a total of 14 participants. Games were often augmented by the eventual presence of one or more test agents.

Future work in this direction will most likely be headed towards the direction of further introducing the notion of continuity into the models. The main current problem is that, while there is only one correct class for each time series, our model makes several predictions, one for each bid placed. Results could probably be enhanced by a meta-layer, which would take into account previous predictions of the same model for each agent and decide accordingly.

8 ACKNOWLEDGEMENTS

This work is part of the 03ED735 research project, implemented within the framework of the Reinforcement Programme of Human Research Manpower (PENED) and cofinanced by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.E.-European Social Funding).

References

[1] Bapna, R., Goes, P., Gupta, A. and Karuga, G. Predicting Bidders' Willingness to Pay in Online Multiunit Ascending Auctions: Analytical and Empirical Insights. Informs Journal on Computing, 20(3):345–355, INFORMs (2008)

[2] Cliff, D. 1997. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HP-97-91. Hewlett-Packard Research Laboratories, Bristol, England (1997)

[3] Erev, I. and Roth, A. E. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. American Economic Review, 88(4):848–881 (1998)

[4] Gerding, E., McBurney, P., Niu, J., Parsons, S., and Phelps, S. Overview of CAT: A market design competition. Technical Report ULCS-07-006. Department of Computer Science, University of Liverpool, Liverpool, UK. Version 1.1 (2007)

[5] Gjerstad, S. and Dickhaut, J. Price formation in double auctions. Games and Economic Behaviour, 22:1–29 (1998)

[6] Gode, D. K., and Sunder, S. Allocative efficiency of markets with zero-intelligence traders: Market as a partial sustitute for individual rationality. Journal of Political Economy, 101(1):119–137 (1993)

[7] Gruman, M. L. and Narayana, M. Applications of classifying bidding strategies for the CAT Tournament. In Proceedings of the International Trading Agent Design and Analysis Workshop. (Chicago, IL, USA, July 14, 2008). TADA 2008. AAAI, 11–18 (2008)

[8] Lesh, N., Zaki, M. J., and Ogihara, M. Mining features for sequence classification. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge Discovery and Data Mining (San Diego, CA, United States, August 15–18, 1999). ACM, 342–346 (1999)

[9] Martinez – Alvarez, F., Troncoso, A., Riquelme, J. C., and Aguilar – Ruiz, J. S. LBF: A Labeled-Based Forecasting Algorithm and Its Application to Electricity Price Time Series. In Proceedings of the Eighth IEEE International Conference on Data Mining (Pisa, Italy, December 15 – 19, 2008). ICDM'08. IEEE, 453 – 461 (2008)

[10] Sattherthwaite, M.A. and Williams, S.R. The Bayesian Theory of the k-Double Auction. The Double Auction Market – Institutions, Theories, and Evidence. Addison-Wesley, D. Friedman and J. Rust, Chapter 4, pp. 99–123 (1993)

[11] Symeonidis, A.L. and Mitkas P.A. Agent Intelligence through Data Mining, Springer Science and Business Media (2005)

[12] Symeonidis, A.L., Nikolaidou, V. and Mitkas, P.A. Sketching a methodology for efficient Supply Chain Management agents enhanced through Data Mining. International Journal of Intelligent Information and Database Systems 2(1): 49–68, Elsevier (2008)

[13] Witten, I.H. and Eibe, F. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, New Zealand (1999)

[14] Wurman, P. R., Walsh, W. E., and Wellman, M. P. Flexible double auctions for electronic commerce: theory and implementation. Decision Support Systems, 24(1):17-27, Elsevier (1998)

[15] Zaki, M. J. Sequence mining in categorical domains: incorporating constraints. In Proceedings of the ninth international conference on Information and knowledge management (McLean, Virginia, United States, November 06–11, 2000). ACM, 422–239 (2000)

Auto-Clustering using Particle Swarm Optimization and Bacterial Foraging

Jorge Cordero H., Yifeng Zeng, and Jakob Rutkowski O.

Department of Computer Science, Aalborg University, Selma Lagerlfs Vej 300 Aalborg 9220, Denmark,
jpcordero@exatec.itesm.mx, yfzeng@cs.aau.dk, jro@cs.aau.dk,
WWW home page: http://www.cs.aau.dk/ yfzeng/

Abstract. This paper presents a hybrid approach for clustering based on particle swarm optimization (PSO) and bacteria foraging algorithms (BFA). The new method *AutoCPB* (Auto-Clustering based on particle bacterial foraging) makes use of autonomous agents whose primary objective is to cluster chunks of data by using simplistic collaboration. Inspired by the advances in clustering using particle swarm optimization, we suggest further improvements. Moreover, we gathered standard benchmark datasets and compared our new approach against the standard *K-means* algorithm, obtaining promising results. Our hybrid mechanism outperforms earlier PSO-based approaches by using simplistic communication between agents.

1 Introduction

The aim of Data clustering is to recognize patterns in Data and form groups (clusters C) of interdependent objects. Exhaustive clustering associates every possible object to a given cluster. Clustering algorithms can produce disjoint or overlapping clusters. In this work we focus on Data clustering (exhaustive and disjoint) utilizing pairwise Euclidean distance between points in a m dimensional vectorial space. Specifically, the objective is to partition a dataset $D = \{d_i | i = 1, \cdots, n\}$ with n records into a set of $k = |C|$ clusters according the following constraints: Every $d \in D$ has to be assigned to a cluster $C_i \in C$ such that $\forall_{C_i \in C}\ C_i \neq \emptyset$ and $\forall_{C_i, C_j \in C}\ C_i \cap C_j = \emptyset$. In this paper, we compare *AutoCPB* with the *K-means* [1] algorithm due to its popularity and robustness.

Several machine learning techniques have been applied to solve the problem of clustering. For example in [2], a neural network clusters Data using entropy estimates. In [3], a genetic algorithm is combined with Nelder-Mead simplex search in order to produce a hybrid clustering algorithm. On the other hand, self organizing maps [4] have also been used for partitioning a dataset without specifying the number of clusters. However, much effort has not been dedicated to study evolutionary collaborative scheme for clustering. We propose an extension of the elemental particle swarm clustering algorithm for clustering.

Multiagent systems are used for simulating complex environments. In this paper, we propose a swarm intelligence clustering algorithm which takes advantage of simplistic communication and evolutionary methods in agents. Particle swarm optimization [5] is

a class of evolutionary algorithms which aims to find a solution to a given optimization problem. On the other hand, bacterial foraging algorithms [6] are a new paradigm in searching based on behavior of biological systems. In this paper we, we extend the advantages of social influence in PSO with the influence of bacterial foraging behavior. The rest of this paper is organized as follows: Section 2 introduces background materials related to PSO and BFA. Section 3 describes our novel approach in detail. Section 4 depicts experimental results and some implementation details. Finally, Section 5 concludes our discussion and provides interesting remarks for future work.

2 Background

The fundamental idea of swarm intelligence algorithms [7] is that a set of individuals can cooperate in a decentralized manner increasing their productivity. Thus, the aim is to find mechanisms that can model complex systems, and represent them in a formal way [8].

2.1 Particle Swarm Optimization Exposed

Particle swarm optimization is a form of stochastic optimization based on swarms of simplistic, social agents [5]. Primary algorithms of particle swarm optimization perform search over a m dimensional space U by using a set of agents. In this lattice, an agent (particle) i occupy a position $x_i(t) = \{x_{i,j}(t)|j = 1,\cdots,m\}$ and has a velocity $v_i(t) = \{v_{i,j}(t)|j = 1,\cdots,m\}$ in an instant t, with a 1:1 correspondence (both $x_i(t)$ and $v_i(t)$ contain a set of components $\{j = 1,\cdots,m\}$ mapped to coordinates in U). A simple PSO algorithm [7] works as follows: In the initialization phase, every agent takes positions around $x(0) = x_{min} + r(x_{max} - x_{min})$ and the velocities are set to 0 (x_{min} and x_{max} are the minimal and maximal magnitudes in U and r is a real number between 0 and 1). Secondly, the algorithm enters in the search phase. The search phase consists in the following steps: Best cognitive/global position updating and velocity/position updating. In the cognitive updating step every agent sets a value for the current (local) best position $y_i(t)$ that it has directly observed. The agent's local optima $y_i(t+1)$ is updated according to equation 1:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{otherwise,} \end{cases} \tag{1}$$

whereas $f(x_i(t))$ is a fitness function which evaluates the goodness of a solution based on position $x_i(t)$. In contrast, the global updating step sets in each iteration the best possible position Y_i observed by any agent. Equation 2 depicts the selection of the best global position.

$$Y_i(t+1) = \begin{cases} Y_i(t) & \text{if } f(y_i(t)) \geq f(Y_i(t)), \forall y_i(t) \\ y_i(t) & \text{if } \exists y_i(t)|f(y_i(t)) < f(Y_i(t)), \end{cases} \tag{2}$$

The velocity/position updating step selects new values for the position $x_i(t+1)$ and velocity $v_i(t+1)$ using equations 3 and 4 respectively:

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{3}$$

$$v_i(t+1) = v_i(t) + \underbrace{c_1 r_1 (y_i(t) - x_i(t))}_{\textbf{Cognitive component}} + \underbrace{c_2 r_2 (Y_i(t) - x_i(t))}_{\textbf{Global component}}, \tag{4}$$

where the c_1 and c_2 parameters are used to guide the search between local (cognitive component) and social (global component) observations. $r_1, r_2 \in [0,1]$ are random parameters which introduce a stochastic weight in the search. Finally, the algorithm stops until some convergence point is reached. A suggested convergence test is proposed in [9]; it consists in testing whether $(f(Y_i(t)) - f(Y_i(t-1)))/f(Y_i(t))$ is smaller than a small constant ε for a given number of iterations. It is important to note that each particle in this multiagent system shares its knowledge (global best position) with all other particles by means of a neighborhood topology. Several topologies have been proposed [7] (i.e. star, ring, clusters, Von Neumann, etc.). The difference between neighborhoods lies in how fast or slow (depending on connectivity) knowledge propagates through the swarm. Further improvements have been proposed for the basic PSO algorithm [7], [9]: Velocity clamping, inertia weight and constriction coefficient.

2.2 Bacterial Foraging Algorithms

Bacterial foraging algorithms are a new class of stochastic global search techniques [6]. Such algorithms emulate the foraging behavior of bacteria while situated in some nutrient substance. During foraging, a bacterium can exhibit two different actions: Tumbling or swimming. The tumble action modifies the orientation of the bacterium. During swimming (chemotactic step) the bacterium will move in its current direction. After tumbling, the bacterium checks if it can find nutrients in its current direction, and if it can, then it will swim for a finite number of steps in that direction. After the bacterium has collected a given amount of nutrient, it will divide in two. The environment can also act in the bacteria population by eliminating or dispersing them.
A bacterial foraging algorithm can be defined as follows. Given a m dimensional search space U, each bacterium i has a position $x_i(t) = (x_{i,j}(t) | j = 1, \cdots, m)$ with m components at time t. It also has a chemotactic step size $C(i) > 0$ which influences the bacterium's step length. Initially the Bacteria is situated in several points in U. Then, each bacterium generates a random tumbling vector b_i consisting of m components. Following, the agent will enter into the swimming phase. Therefore, it will update its position one step at a time according to equation 5.

$$x_i(t+1) = x_i(t) + C(i) * b_i \tag{5}$$

swimming will continue for a number N_a of iterations iff $f(x_i(t+1)) < f(x_i(t))$ holds. Once that all agents finished tumbling and swimming, they reproduce. New generations are created only with the half of the *healthiest* agents. *Healthy* agents can be interpreted as the ones which performed the smallest number of chemotactic steps. Finally, an elimination/dispersal step deletes and reallocates a percentage of agents at random.

The algorithm will run for a fixed number of iterations. A simple improvement for tumbling/swimming is to make bacteria attract each other in some area; and then repeal each other as they consume nearby nutrients. The idea is to calculate the cell to cell fitness, and add it to the fitness position for each bacterium.

3 PSO/BFA Multiagent Clustering

PSO algorithms have a tendency to fall into local optima, because of the lack of diversity in the swarm. An improvement of the original PSO algorithm [11] is to hybridize it with another swarm intelligence method [12].

3.1 PSO Clustering

We implemented a simple clustering algorithm denominated *ClusterP*, which was proposed in [11]. Such method combines PSO and *K-means* for grouping data. The dataset $D = \{d_{1,m}, d_{2,m}, \cdots, d_{n,m}\}$ defines the number of components m and instances n in the search space U. Thus, each datum $d_{i,j} \in D$ can be seen as a point in U. Since the aim is to find a set of k desired clusters C containing every element in D; it is easy to visualize that the main problem is to find the set of $O = \{o_1, o_2, \cdots, o_k\}$ centroids that minimize the fitness function (Euclidean distance) with respect to each point in D. In *clusterP*, every agent $p_l \in P$ represent a solution with the set of position $O_l = \{o_{l,j} | j = 1, \cdots, k\}$. Algorithm 3.1 shows the *ClusterP* technique. *ClusterP* works as follows: It receives

Algorithm 3.1: The *ClusterP* Algorithm.

Input: Data $D = \{d_1, d_2, \cdots, d_n\}$, k.
Output: Clusters $C = \{C_1, C_2, \cdots, C_k\}$.

1: Initialize swarm P (distribute O).
2: **REPEAT**:
3: **FOR** $i = 1$ to n:
4: **FOR** $j = 1$ to k:
5: Calculate distances $dist(d_i, o_j)$.
6: **END FOR**
7: $C_h \Leftarrow d_i$ **iff** $argmin_{o_h \in O}(dist(d_i, o_h))$.
8: **END FOR**
9: **FOR** $l = 1$ to $|P|$:
10: **Update** cognitive positions for p_l using Eq. 1.
11: **Update** global position for P using Eq. 2.
12: **END FOR**
13: **FOR** $l = 1$ to $|P|$:
14: **Update** velocities for p_l using Eq. 3.
15: **Update** positions for p_l using Eq. 4.
16: **END FOR**
17: **UNTIL** stopping condition holds.

the Data D and an integer k. First, it collocates every agent p in the environment, each containing a set of centroids O (line 1). Then, it assigns every record d_i to a cluster C_h iff the distance with its centroid $dist(d_i, o_h)$ is minimal (lines 3-8). Following, it obtains the cognitive and global position (lines 9-12), and updates every of its centroids velocities and positions (lines 13-16) as explained in Section 2.1. The algorithm stops after a fixed number of iterations or if no significant progress is made according to the fitness function.

3.2 Automatic PSO Clustering

ClusterP was extended in [13] in order to find the optimal number of clusters automatically. The new method is called *AutoCP*. It starts with an initial ḧighn̈umber k' of clusters ($k' \leq n$), and it deletes the *inconsistent* clusters. We describe the form to detect inconsistent clusters as follows: First, for each cluster C_j we calculate its weight $W_j = \sum_{q=1}^{|C_j|} dist(o_j, d_q)$ as the sum of the distances from the center o_j to its points q_j where $q_j \neq o_j$. Immediately, all weights W from the clusters are sorted. Then, we normalize all weights W_j with respect to the cluster with lowest value W_s, such that the set of local thresholds become $th = \{th_j | j = 1, \cdots, k', th_j = W_s/W_j, W_s = argmin_{w \in W}(w)\}$. Finally, a cluster C_j is declared as *inconsistent* iff its threshold t_j is lower than the global threshold T, whereas T is in the range of [0,1]. Algorithm 3.2 introduces *AutoCP*. As

Algorithm 3.2: The *AutoCP* Algorithm.

Input: Data $D = \{d_1, d_2, \cdots, d_n\}$.
Output: Clusters $C = \{C_1, C_2, \cdots, C_{k'}\}$.

1: Initialize swarm P (distribute O).
2: **REPEAT**:
3: Assign D to clusters as in algorithm 3.1.
4: **Update** cognitive/global positions as in algorithm 3.1.
5: **Update** velocities/positions as in algorithm 3.1.
6: **FOR** $j = 1$ to k':
7: Calculate W_j for each cluster C_j.
8: **END FOR**
9: Obtain $W_s = argmin_{W_j \in W}(W_j)$.
10: **FOR** $j = 1$ to k':
11: $th_j = W_s/W_j$.
12: **IF** $(th_j < T)$
13: Remove C_j and $k' = k' - 1$.
14: **END IF**
15: **END FOR**
16: **UNTIL** stopping condition holds.

previously mentioned, *AutoCP* simply adds a mechanism to *ClusterP* for automatic de-

tection of clusters with no further modification (lines 6-15). At the end of each iteration, we select and discard inconsistent clusters.

3.3 AutoCB Clustering

In this section we present the method for automatic clustering using bacterial foraging algorithm *AutoCB*. It is an adapted version of the bacterial foraging algorithm for automatic clustering. Basically, it uses the *K-means* principle to add Data to the closest centroids combined with a bacterial foraging search. In algorithm 3.3, every agent observates a clustering solution represented by the positions of bacteria as centroids (notice that times are references to current $x_i(t)$ or posterior $x_i(t+1)$ positions in space. x_i in fact represent the tentative position for centroid o_i). In order to keep this algorithm as simple as possible, we decided not to add any complex initialization method. We simply exchanged the swimming mechanism to be executed firstly, followed by tumbling. In this way we use the fitness function to set up bacteria in good spots since the beginning. Initially, the algorithm distributes the positions for all centroids of every

Algorithm 3.3: The *AutoCB* Algorithm.

Input: Data $D = \{d_1, d_2, \cdots, d_n\}$.
Output: Clusters $C = \{C_1, C_2, \cdots, C_{k'}\}$.

1: Initialize B (distribute O randomly).
2: FOR Number of Elimination/Dispersal Steps
3: **FOR** Number of Reproduction Steps
4: **FOR** Number of Chemotactic Steps
5: **FOR** Each Bacterium $i \in B$
6: **WHILE** $m < MaxSwimLength$
7: Delete inc. clusters as in alg. 3.2.
8: Assign D to C as in alg. 3.1.
9: **IF** $f(x_i(t+1), o_i) < f(x_i(t), o_i)$
10: $o_i = x_i(t+1)$.
11: $m = m+1$.
12: **END IF**
13: **ELSE**
14: $m = MaxSwimLength$.
15: **END ELSE**
16: **END WHILE**
17: $tumble(i)$, generate new $x_i(t+1)$.
18: **END FOR**
19: **END FOR**
20: $reproduce(B)$.
21: **END FOR**
22: $eliminate(B), disperse(B)$.
23: END FOR

agent in B at random (line 1). For every bacterium i, the chemotactic part of the algorithm starts (lines 4-19): The agents will update their centroids positions for a maximal number of swims $MaxSwimLength$ (lines 6-16). Firstly, the algorithm deletes inconsistent clusters as mentioned in lines 6-15 of algorithm 3.2 (line 7). Then, we assign each datapoint $d \in D$ to a cluster $C_j \in C$ according to the lines 3-8 of algorithm 3.1 (line 8). Every agent observates the cells $x_i(t+1)$ in front of its centroids o_i. Then, they swim and update their centroids positions from o_i to $x_i(t+1)$ iff the Euclidean distance $f(x_i(t+1), o_i)$ to the new point is smaller than the current one $f(x_i(t), o_i)$ (lines 9-12). Swimming might immediately stop according to the control variable m in line 14; if the fitness of the tentative cell $f(x_i(t+1), o_i)$ is greater or equal to the one of the current position $f(x_i(t), o_i)$.
Once agent i has finished swimming, it will $tumble$ (line 17). Thus, i will choose a tentative cell in a new direction $x_i(t+1)$ for each of its centroids $o_i = x_i(t)$. $x_i(t+1)$ is selected at random. Finally, the previous chemotactic process is encapsulated in a reproductive phase (line 3), and in a combined elimination/dispersion phase(line 2). The reproduction process deletes half of the agents having the smallest number of chemotactic steps. The elimination method destroys again a percentage α of agents at random. Dispersion reallocates a small percentage β of the remaining agents in U at random.

3.4 The AutoCPB Algorithm

In this section we describe in detail the hybrid algorithm for clustering based on PSO and BFA denominated *AutoCPB*. The algorithm follows the same structure of the AutoCB algorithm. Nevertheless, each bacterium agent i has assigned a position and a velocity to its centroids as previously seen in algorithm 3.1. The tumble is now guided by the swarm local and social beliefs. A tentative cell $x_i(t+1)$ is deterministically chosen according to the best neighboring cell $y_i(t+1)$ and the global best position $Y_i(t+1)$. Swimming is still performed for a given number of iterations but now the agent's steps vary in dimension according to its velocity. *AutoCPB* contains a bacterial foraging skeleton and a particle swarm optimizator. In this algorithm, we simply replace tumbling in line 17 from algorithm 3.3 by a more elaborated PSO-based foraging mechanism (lines 18-24). Logically, swimming in lines 9-12 of algorithm 3.3 is also modified so we take advantage of the guided tumbling.
Specifically, the *AutoCPB* clustering algorithm works as follows: It collocates every centroid o_i for agent i in U at random ($x_i(t) = o_i$). Then, the modified tumbling/swimming version is executed for all agents for a number of swims $MaxSwimLength$ (lines 4-25): Inconsistent clusters are removed (line 7) as seen in algorithm 3.2. All datapoints $d \in D$ are assigned to the remaining clusters $C = \{C_j | j = 1, \cdots, k^{'}\}$ (line 8). Then, all centroid positions $o_i = x_i(t)$ for agent i will be updated with the new position $x_i(t+1)$ iff $f(x_i(t+1)) < f(x_i(t))$ holds (lines 9-10). Otherwise, swimming stops in line 16. In this part of the algorithm, we swim by updating the value of $x_i(t+1)$ according equation 3 (line 11). We also record the best cognitive/local position $y_i(t+1)$ according to equation 1 for further tumbling computation (line 12). The final step of swimming is the update of the best global/social position $Y_i(t+1)$ (line 18).
In lines 21-24, PSO-based tumbling is executed. At this point, every centroid (with a velocity $v_i(t)$ and a position $x_i(t)$) starts observing its neighboring cells. It finally updates

Algorithm 3.4: The *AutoCPB* Algorithm.

Input: Data $D = \{d_1, d_2, \cdots, d_n\}$.
Output: Clusters $C = \{C_1, C_2, \cdots, C_{k'}\}$.

1: Initialize B (distribute O randomly).
2: FOR Number of Elimination/Dispersal Steps
3: **FOR** Number of Reproduction Steps
4: **FOR** Number of Chemotactic Steps
5: **FOR** Each agent $i \in B$
6: **WHILE** $m < MaxSwimLength$
7: Delete inc. clusters as in alg. 3.2.
8: Assign D to C as in alg. 3.1.
9: **IF** $f(x_i(t+1)) < f(x_i(t))$
10: $o_i = x_i(t+1)$.
11: **Update** $x_i(t+1)$ using Eq. 3.
12: **Update** $y_i(t+1)$ using Eq. 1.
13: $m = m+1$.
14: **END IF**
15: **ELSE**
16: $m = MaxSwimLength$.
17: **END ELSE**
18: **Update** $Y_i(t+1)$ using Eq. 2.
19: **END WHILE**
20: **END FOR**
21: **FOR** Each agent $i \in B$
22: **Update** $v_i(t+1)$ using Eq. 3.
23: **Update** $x_i(t+1)$ using Eq. 4.
24: **END FOR**
25: **END FOR**
26: $reproduce(B)$.
27: **END FOR**
28: $eliminate(B), disperse(B)$.
29: END FOR

its own velocity to $v_i(t+1)$ and tentative position $x_i(t+1)$ by using equations 3 and 4 respectively. Notice that agents do not modify its current position during tumbling but only during the swimming phase (lines 9-14). Finally, once that we have clustered U into C, *AutoCPB* will reproduce (line 26), eliminate and disperse (line 28) agents in the same manner *AutoCB* does.
In the next part of this paper we comprehensively tested every approach obtaining promising results.

4 Experimental Results

The algorithms were tested on nine datasets (D), two of them were artificially made and the rest were taken from the UCI repository [14]. The artificial datasets generated

at random are: Artificial1 (A1), having two classes and two dimensions and Artificial 2 (A2), having three classes and three dimensions. The UCI datasets are: Iris (Ir), having three classes and four dimensions, Wine (Wi), which has three classes and thirteen dimensions, Pima (Pi), containing two classes and eight dimensions, Haberman (Ha), which has two classes and three dimensions, BreastCancer (BC), consists of two classes and thirty dimensions, Glass (Gl), which has six cases and nine dimensions and Yeast (Ye), having ten classes and eight dimensions.

4.1 Environmental Settings

Each experiment is based on fifty consecutive runs of the same algorithm. For the *K-means* algorithm we set a maximal of 500 iterations (it terminates if no improvement is made). In the other algorithms, the assignation of points to clusters consists of a single iteration. For *ClusterP* and *AutoCP* the stopping condition is the one described in Section 2.1. For *AutoCB* and *AutoCPB* we used 10 elimination/dispersal iterations, 20 reproduction steps and 20 chemotactic steps. In every method we used 30 agents. Every dataset has a predefined class. Therefore, we set k in *ClusterP* as the number of classes (in fact, we found that K-means obtains the best results in this fashion). For the multiagent-based automatic clustering algorithms we used 30 agents and an upper limit $k^{'}$ of 10 initial clusters.

4.2 Cluster Validation

We used two validation measures [15], namely the inter cluster distance measure (ID) and the quantization error function (QEF). Both metrics have previously utilized to test cluster reliability [11]. The inter cluster distance calculates the average distances between the centroids and all their points in the cluster. It is calculated according to equation 6.

$$ID_C = \sum_{\forall_{o_i,o_j} \in CEN, i \neq j} dist(o_i, o_j)/|C|, \tag{6}$$

where o_i and o_j are centroids, CEN is the set of all centroids, $|C|$ is the total number of clusters C and $dist()$ is the Euclidean distance. In essence, we calculate the average Euclidean distance between all pairs of centroids. The QEF is a global distance measure that evaluates the average distance from all datapoints to centroids in every cluster. Equation 7 present the QEF metric.

$$QEF_C = \big(\sum_{j=1,k} (\sum_{\forall_{d_i \in C_j}} dist(d_i, o_j)/|C_j|) \big)/k, \tag{7}$$

whereas k is the number of clusters, d_i is a datapoint contained in cluster C_j. All other variables are defined as in equation 6. The previous measures can be used to express the quality of a solution. Thus, we proceed to establish a discussion of the performance of the algorithms.

4.3 Benchmark Testing

Table 1 includes the average results for the inter cluster distance, quantization error function, number of clusters and elapsed times (in milliseconds). Numbers in bold represent the best result for each dataset. We believe that the ID and QEF tests express an acceptable comparison between clusterings for this investigation. Thus, every algorithm makes use of the *K-means* oriented mechanism for assigning points to clusters.

For all cases, we appreciate that *K-means* is the worst performing algorithm in terms of quality and the best in running time. However, in many cases we are more concerned in the quality of a solution.

We can conclude that with respect to the QEF metric; *AutoCP* and *AutoCPB* are the best algorithms, finding minimal values. *AutoCP* having a minimal QEF in A1, A2, Ir, Gl and Ye. *AutoCPB* is the most competitive in Ha, Pi, Wi, BC and Ye. However, both methods produce similar results. In every dataset any of them can be the first and the second most competitive method. This behavior is logical, and it is a common example of how the random element introduces some noise to a search technique. *AutoCPB* performs random steps (reproduction, elimination and dispersion).

For the ID metric we observe an almost identical trend. *AutoCB* outperforms all other methods in some cases (A1, A2). Indeed, *AutoCB* is the most randomized algorithm, it performs a nearest neighbor search with poor guidance but with a dynamical evolutionary mechanism. In every case, we can see that *ClusterP* is suboptimal and falls into local optima. The later is one of the reasons why the evolutionary method in the form of automatic clustering and bacterial foraging show a promising enhancement.

We can observe in Table 1 that a higher number of clusters (A1, A2, Ha, Pi, Ir, Wi and BC) tends to give a lower result in both ID and QEF. This might be because our tests do not express what we are really looking for, or perhaps, that we have discovered a new feature in the dataset. Thus, we can modify our assumptions about the optimal number of clusters in a dataset. Actually, since we use the same cluster assignment method in all techniques, we can conclude that we have also found the optimal number of clusters k given our objective function.

5 Final Discussion

In this work we described two swarm intelligence techniques, namely particle swarm optimization and bacterial foraging. We described in detail several algorithms based on these paradigms. The essence of our work is that we managed to combine two major paradigms PSO and BFA in order to create a robust clustering algorithm. The resulting hybrid method *AutoCPB* showed improvements during experimentation. The original *AutoCP* algorithm was unable to reach the best results on its own. Actually, *AutoCB* is also suboptimal in non-random datasets because of its swarm model, which does not perform global search.

The testing was possible due to our implemented framework, which enabled us to test all algorithm in well known datasets. In general, we believe that the application of hybrid PSO/BF mechanisms is promising. We expect to improve AutoCPB to a major extent. Future work consists in identifying a rule to minimize local optima in AutoCPB. This

Table 1. Full clustering results. Swarm-based enhancements improve the performance of *K-means*.

D	Method	QEF	ID	$\|C\|$	E. time
A1	*K-means*	11.64 +/-0.10	28.4 +/-0	2	**14.06 +/-5.75**
A1	*ClusterP*	11.99 +/-0.42	29.95 +/-2.03	2	1394.68 +/-18.7
A1	*AutoCP*	**4.91 +/-0.57**	4.13 +/-2.46	5.6 +1.01	2105.08 +/-427.96
A1	*AutoCB*	7.75 +/-1.18	**3.36 +/-2.31**	4.48 +/-1.27	8439.36 +/-423.15
A1	*AutoCPB*	6.2 +/-0.7	3.73 +/-2.51	4.8 +/-0.76	835.3 +/-74.6
A2	*K-means*	44.95 +/-33.56	28.89 +/-11.26	3	**95.3 +/-7.36**
A2	*ClusterP*	29.92 +/-2.27	35.62 +/-7.2	3	6914.04 +/-59.66
A2	*AutoCP*	**24.36 +/-1.13**	8.09 +/-3.46	4.9 +/-1.06	9446.92 +/-1376.23
A2	*AutoCB*	30.40 +/-3.07	**7.14 +/-3.87**	4.36 +/-1.03	11914.98 +/-269.82
A2	*AutoCPB*	24.45 +/-2.11	7.73 +/-3.94	4.22 +/-0.93	3658.74 +/-298.91
Ha	*K-means*	14.59 +/-6.69	8.85 +/-0.13	2	**135.94 +/-7.25**
Ha	*ClusterP*	10.23 +/-0.32	6.83 +/-1.96	2	13655.02 +/-171.18
Ha	*AutoCP*	9.18 +/-0.65	1.39 +/-0.84	3.26 +/-0.83	18916.88 +/-3962.37
Ha	*AutoCB*	12.41 +/-1.73	1.57 +/-0.82	2.98 +/-0.98	12055.92 +/-272.07
Ha	*AutoCPB*	**8.87 +/-0.76**	**1.2 +/-0.7**	3.14 +/-0.64	6930 +/-366.89
Pi	*K-means*	132.86 +/-46.72	111.77 +0	2	**742.2 +/-15.54**
Pi	*ClusterP*	73.85 +/-3.363	35.25 +/-11.43	2	72757.18 +/-1564.94
Pi	*AutoCP*	72.35 +/-3.29	11.43 +/-12.17	3.02 +/-1.12	107948.8 +/-20853.2
Pi	*AutoCB*	96.33 +/-12.68	12.06 +/-11.92	3.16 +/-0.89	33195.9 +/-1179.08
Pi	*AutoCPB*	**65.43 +/-6.73**	**6.65 +/-5.6**	2.96 +/-0.53	37445.02 +/-4117.43
Ir	*K-means*	1.58 +/-0.58	1.13 +/-0.49	3	**117.5 +/-7.89**
Ir	*ClusterP*	0.98 +/-0.14	1.19 +/-0.33	3	8390.32 +/-91.91
Ir	*AutoCP*	**0.78 +/-0.06**	0.26 +/-0.15	3.8 +/-0.81	11591.68 +/-1577.85
Ir	*AutoCB*	1.13 +/-0.17	0.25 +/-0.16	3.46 +/-1.16	14862.18 +/-297.83
Ir	*AutoCPB*	0.84 +/-0.09	**0.24 +/-0.13**	3.62 +/-0.81	4303.76 +/-324.01
Wi	*K-means*	216.67 +/-52.69	162.28 +/-63.28	3	**403.76 +/-13.2**
Wi	*ClusterP*	131.5 +/-8.60	68.74 +/-43.93	3	26301.88 +/-546.39
Wi	*AutoCP*	99.49 +/-7.02	24.79 +/-18.72	5.48 +/-1.05	37724.1 +/-3461.18
Wi	*AutoCB*	141.87 +/-17.68	32.1 +/-22.11	4.54 +/-1.11	42410.36 +/-1111.42
Wi	*AutoCPB*	**97.87 +/-10.51**	**23.29 +/-21.49**	4.88 +/-1.1	13421.86 +/-799.42
BC	*K-means*	563.32 +/-142.04	665.67 +/-0	2	**1949.08 +/-215.4**
BC	*ClusterP*	314.87 +/-11.53	141.46 +/-51.5	2	174520.6 +/-3574.57
BC	*AutoCP*	196.4 +/-14.18	44.3 +/-41.07	5.3 +/-1.04	259376.08 +/-34956.1
BC	*AutoCB*	292.89 +/-53.18	59.86 +/-50.86	4.32 +/-1.28	108388.1 +/-2871.43
BC	*AutoCPB*	**195.01 +/-22.4**	**34.44 +/-35.2**	4.62 +/-0.83	92403.2 +/-8281.34
Gl	*K-means*	9.23 +/-7.21	0.63 +/-0.29	6	**698.74 +/-41.74**
Gl	*ClusterP*	3.31 +/-0.42	0.39 +/-0.22	6	25520.92 +/-280.95
Gl	*AutoCP*	**1.23 +/-0.25**	0.29 +/-0.24	2.28 +/-0.5	45504.3 +/-24056.07
Gl	*AutoCB*	2.31 +/-0.4	0.24 +/-0.16	2.94 +/-1.27	29877.22 +/-651.5
Gl	*AutoCPB*	1.62 +/-0.31	**0.21 +/-0.2**	2.2 +/-0.45	11431.86 +/-1466.4
Ye	*K-means*	1.46 +/-1.09	0.04 +/-0.02	10	**6815.02 +/-62.67**
Ye	*ClusterP*	0.8 +/-0.12	**0.03 +/-0.01**	10	175838.7 +/-1397.52
Ye	*AutoCP*	**0.26 +/-0.02**	**0.03 +/-0.01**	2.12 +/-0.33	211558.7 +/-46305.2
Ye	*AutoCB*	0.34 +/-0.04	0.04 +/-0.02	2.42 +/-0.61	35482.18 +/-558.49
Ye	*AutoCPB*	**0.26 +/-0.02**	**0.03 +/-0.02**	2.22 +/-0.42	69598.58 +/-3044,74

algorithm can be applied to other domains such as attribute clustering. This stochastic/evolutionary method can also show features not only in Data but between variables in datasets. A more specific analysis of parameter setting is also considered as a tentative improvement.

References

1. Han, F., Kamber, K.: Data mining: Concepts and techniques. Academic press. San Francisco, 334–395 (2001)
2. Tazutov, A., Kurenkov, N.: Neural network data clustering on the basis of scale invariant entropy. International Joint Conference on Neural Networks, 4912-4918 (2006)
3. Chandra, S., Murthy, J.: An Efficient Hybrid Algorithm for Data Clustering Using Improved Genetic Algorithm and Nelder Mead Simplex Search. Proc. of the Int. Conf. on Comp. Int. and Mult. Appl. 1, 498–510 (2007)
4. Goncalves, M., Andrade, M.: Data Clustering using Self-Organizing Maps segmented by Mathematic Morphology and Simplified Cluster Validity Indexes: an application in remotely sensed images. International Joint Conference on Neural Networks, 4421–4428 (2006)
5. Kennedy, J., Ebenhart, R.: Particle Swarm Optimization. Proceedings of the 1995 IEEE Int. Conf. on Neural Networks, 1942–1948 (1995)
6. Passino, K.: Biomimicry of bacterial foraging for distributed optimization and control. Control Systems Magazine, 52–67 (2002)
7. Engelbrecht, A.: Fundamentals of Computational Swarm Intelligence. Wiley (2005)
8. Heylighen, F., Joslyn, C.: Cybernetics and second-order cybernetics. Encyclopedia of Physical Science and Technology, Academic Press, New York (2001)
9. Bergh, F.: An analysis of particle swarm optimizers. PhD Thesis, University of Pretoria, South Africa (2002)
10. Clerc, M., Kennedy, J.: The particle swarm: Explosion, stability and convergence. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
11. van der Merwe, D., Engelbrecht, A.: Data Clustering using particle swarm optimization. The 2003 congress on Evolutionary Computation 1, 215–220 (2003)
12. Biswas, A., Dasgupta, S.: Synergy of PSO and bacterial foraging optimization: A comprehensive study on. Advances in Soft Computing Series, Springer Verlag, 255–263 (2007)
13. Abraham, A., Roy, S.: Swarm intelligence algorithms for data clustering. Chp. 12, 279–313 (2008)
14. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html (2007)
15. Theodoris, S., Koutroumbas, K.: Pattern Recognition. Academic Press (2006)

Agent-Enriched Data Mining Using an Extendable Framework

Kamal Ali Albashiri and Frans Coenen

Department of Computer Science, The University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom,
ali@csc.liv.ac.uk, frans@csc.liv.ac.uk

Abstract. This paper commences with a discussion of the advantages that Multi-Agent Systems (MAS) can bring to the domain of Knowledge Discovery in Data (KDD), and presents a rational for Agent-Enriched Data Mining (AEDM). A particular challenge of any generic, general purpose, AEDM system is the extensive scope of KDD. To address this challenge the authors suggest that any truly generic AEDM must be readily extendable and propose EMADS, The Extendable Multi-Agent Data mining System. A complete overview of the architecture and agent interaction models of EMADS is presented. The system's operation is described and illustrated in terms of two KDD scenarios: meta association rule mining and classifier generation. In conclusion the authors suggest that EMADS provides a sound foundation for both KDD research and application based AEDM.

1 Motivation and Goals

Agent-Enriched Data Mining (AEDM), also known as multi-agent data mining, seeks to harness the general advantageous of MAS in the application domain of Data Mining (DM). MAS technology has much to offer DM, particularly in the context of various forms of distributed and cooperative DM. Distributed (and parallel) DM is directed at reducing the time complexity of computation associated with the increasing sophistication, size and availability of the data sets we wish to mine. Cooperative DM encompasses ensemble mechanisms and techniques such as bagging and boosting. MAS have a clear role in both these areas. MAS technology also offers some further advantageous for AEDM, namely:

- Extendibility of DM frameworks,
- Resource and experience sharing,
- Greater end-user accessibility,
- Information hiding, and
- The addressing of privacy and security issues.

The last of the above advantageous merits some further comment. By its nature DM is often applied to sensitive data. The MAS approach would allow data to be mined remotely. Similarly, with respect to DM algorithms, MAS can make use of algorithms without necessitating their transfer to users, thus contributing to the preservation of intellectual property rights. MAS make it possible for software services to be provided

through the cooperative efforts of distributed collections of autonomous agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. Thus, it is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request.

The challenge of generic AEDM is the disparate nature and variety of modern DM, and the necessary communication mechanism required to cope with this disparate nature. One approach is to make use of the established Agent Communication Languages (ACLs) and mechanisms; well known examples include the Knowledge Query and Manipulation Language (KQML), the Knowledge Interchange Format (KIF), and the Foundation for Intelligent Physical Agents (FIPA) ACL [14]. All these ACLs have their advantageous and disadvantageous and tend to address particular forms of intra-agent communication; for example FIPA ACL is directed at agent negotiation. Each can be employed in the context of AEDM communication but on its own will not facilitate the shared agent understanding required to achieve generic AEDM. This would require recourse to the use of ontologies and/or some agreed meta-language. It is suggested in this work that a method of addressing the communication requirements of AEDM is to use a system of mediators and wrappers coupled with an ACL such as FIPA ACL, and that this can more readily address the issues concerned with the variety and range of contexts to which AEDM can be applicable.

To investigate and evaluate the expected advantageous of wrappers and mediators, in the context of generic AEDM, the authors have developed and implemented (in JADE) a multi-agent platform, EMADS (the Extendable Multi-Agent Data mining System). Extendibility is seen as an essential feature of the framework primarily because it allows its functionality to grow in an incremental manner. The vision is of an "anarchic" collection of agents, contributed to by a community of EMADS users, that exist across an "internet space"; that can negotiate with each other to attempt to perform a variety of DM tasks (or not if no suitable collection of agents come together) as proposed by other (or the same) EMADS users. An EMADS demonstrator is currently in operation.

The primary goal of the EMADS framework is to provide a means for integrating new DM algorithms and data sources in a distributed infrastructure and collaborative environment. However, EMADS also seeks to address some of the issues of DM that would benefit from the rich and complex interactions of communicating agents. The broad advantages offered by the framework are:

- Flexibility in assembling communities of autonomous service providers, including the incorporation of existing applications.
- Minimization of the effort required to create new agents, and to wrap existing applications.
- Support for end users to express DM requests without having detailed knowledge of the individual agents.

The rest of this paper is organised as follows. A brief review of some related work on Agent-enriched Data Mining (AEDM) is presented in Section 2. The conceptual framework together with an overview of the wrapper principle is presented in Section 3. The framework operation is illustrated in Section 4 using two DM scenarios: Meta Associ-

ation Rule Mining (MARM) and single label classification. Finally some conclusions are presented in Section 5.

2 Related Work

There are a number of reports in the literature of the application of Agent techniques to DM. The contribution of this section is a broad review of prominent AEDM approaches in the literature and discussion of the benefits that agent-driven DM architectures provide in coping with such problems. This section is not concerned with particular DM techniques; it is however concerned with work on the design of distributed and multi-agent system directed at DM.

The most fundamental approach to distributed DM is to move all of the data to a central data warehouse and then to analyze this with a single DM system, even though this approach intuitively guarantees accurate DM results, it might be infeasible in many cases.

An alternative approach is high level learning with meta-learning strategies in which all the data can be locally analyzed (local data model), and the local results at their local sites combined at the central site to obtain the final result (global data model). Meta-learning methods have been widely used within DM [30, 10], particularly in the area of classification and regression. These approaches are less expensive but may produce ambiguous and incorrect global results. In addition, Distributed DM approaches require centralised control that causes a communication bottleneck that sometimes leads, in turn, to inefficient performance and system failure.

To make up for such a weakness, many researchers have investigated more advanced approaches of combining local models built at different sites. Most of these approaches are agent-based high level learning strategies.

One of the earliest references to AEDM can be found in Kargupta et al. [19] who describe a parallel DM system (PADMA) that uses software agents for local data accessing and analysis, and a web based interface for interactive data visualization. PADMA has been used in medical applications. They describe a distributed DM architecture and a set of protocols for a multi-agent software tool. Peng et al. [25] presented an interesting comparison between single-agent and multi-agent text classification in terms of a number of criteria including response time, quality of classification, and economic/privacy considerations. Their results indicate, not unexpectedly, in favour of a multi-agent approach.

A popular AEDM approach is described in the METAL project [24] whose emphasis is on helping the user to obtain a ranking of suitable DM algorithms through an online advisory system. Gorodetsky et al. [16] correctly consider that the core problem in AEDM is not the DM algorithms themselves (in many case these are well understood), but the most appropriate mechanisms to allow agents to collaborate. Gorodetsky et al. present an AEDM system to achieve distributed DM and, specifically, classification. A more recent system, proposed in [23], uses the MAGE middleware [28] to build an execution engine that uses a directed acyclic graph to formalize the representation of KDD process. In [11] a multi-agent system F-Trade has been proposed. It is a web-based DM infrastructure for trading and surveillance support in capital markets.

The meta-learning strategy offers a way to mine classifiers from homogeneously distributed data. It follows three main steps. The first is to generate base classifiers at each site using a classifier learning algorithms. The second step is to collect the base classifiers at a central site, and produce meta-level data from a separate validation set and predictions generated by the base classifier on it. The third step is to generate the final classifier (meta-classifier) from meta-level data via a combiner or an arbiter. Copies of the classifier agent will exist, or be deployed, on nodes in the network being used (see for example [26]). Perhaps the most mature agent-based meta-learning systems are: JAM [29], BODHI [18], and Papyrus [6]. Papyrus is designed to support both learning strategies; meat-learning and central learning. A hybrid learning strategy is a technique that combines local and remote learning for model building [17]. In contrast to JAM and BODHI, Papyrus can not only move models from site to site, but can also move data when such a strategy is desirable. Papyrus is a specialized system which is designed for clusters while JAM and BODHI are designed for data classification. These are reviewed in details in [20].

Most of the previously proposed AEDM systems are used to improve the performance of one specific DM task. To the best knowledge of the authors, there have been only few AEDM systems that define a generic framework for the AEDM approach. An early attempt was IDM [9], a multiple agent architecture that attempts to do direct DM that helps businesses gather intelligence about their internal commerce agent heuristics and architectures for KDD. In [5] a generic task framework was introduced but was designed to work only with spatial data. The most recent system is introduced in [15] where the authors proposed a multi-agent system to provide a general framework for distributed DM applications. The effort to embed the logic of a specific domain has been minimized and is limited to the customization of the user. However, although its customizable feature is of a considerable benefit, it still requires users to have very good DM knowledge.

3 EMADS Overview

A high level view of the framework conceptualization showing the various categories of agents and their contributors is given in Figure 1. The housekeeping (DF and AMS) agents are specialized server agents that are responsible for helping agents to locate one another. They do not participate in problem-solving; they only play a role of a facilitator in the system. Note that any system configuration is not limited to single MAS. Larger systems can be assembled from multiple MASs, each having the sort of structure shown in Figure 2.

3.1 System Structure

The EMADS framework has several different modes of operation according to the nature of the participant. Each mode of operation has a corresponding category of *User Agent*. Broadly, the supported categories are:

– Developers: Developers are participants, who have full access and may contribute DM algorithms in the form of *Data Mining Agents* (DM Agents).

– Data Miners: These are participants, with restricted access to the system, who may pose DM requests through User Agents and *Task Agents* (see below for further details).
– Data Contributors: These are participants, again with restricted access, who are prepared to make data available, by launching *Data Agents*, to be used by DM agents.

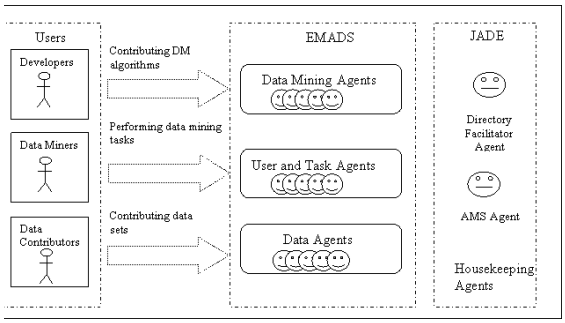


Fig. 1. High level view of EMADS conceptual framework.

The various categories of agents are illustrated in Figure 1: DM agents, Task Agents, User Agents and Data Agents. DM agents are usually specialist agents that posses an algorithm for a particular DM task or sub-task. DM agents may be based on legacy applications, in which case the agent may be little more than a wrapper that calls a pre-existing API (see subsection 3.3 for further detail).

Note that, before interaction with EMADS can commence, appropriate software needs to be downloaded and launched by the participant. Note also that any individual participant may be as a user, contributor and developer at the same time.

Conceptually the nature of the requests that may be posted by users is extensive. In the current demonstration implementation, the following types of generic request are supported:

– Find the "best" classifier (to be used by the requester at some later date in off line mode) for a data set provided by the user.
– Find the "best" classifier for the indicated data set (i.e. provided by some other participant).
– Find a set of Association Rules (ARs) contained within the data set(s) provided by the user.
– Find a set of Association Rules (ARs) contained within the indicated type of data set(s) (i.e. provided by other participants).

In the above a "best" classifier is defined as a classifier that will produce the highest accuracy on a given test set (identified by the mining agent) according to the detail of the request. To obtain the "best" classifier EMADS will attempt to access and communicate

with as many classifier generator DM agents as possible and select the best result. The classification style of user request will be discussed further in subsection 4.2 to illustrate the operation of EMADS in more detail.

The Association Rule Mining (ARM) style of request is discussed further in subsection 4.1. The scenario investigated here is one where an agent framework is used to implement a form of Meta-ARM where the results of the parallel application of ARM to a collection of data sets, with not necessarily the same schema but conforming to a global schema, are combined. Some further details of this process can be found in Albashiri et al. [3, 4].

3.2 Agent Interactions

Conceptually the EMADS system is a hybrid peer to peer agent based system comprising a collection of collaborating agents that exist in a set of containers. Agents may be created and contributed to the system by any user/contributor. One of these containers, the main container, holds a number of housekeeping agents that provide various facilities to maintain the operation of the framework. In particular the main container holds an Agent Management System (AMS) agent and a Directory Facilitator (DF) agent. The terminology used is taken from the JADE (Java Agent Development) [7] framework in which the framework is implemented. Briefly the AMS agent is used to control the life cycles of other agents in the platform, and the DF agent provides an agent lookup service. Both the main container and the remaining containers can hold various DM agents. Note that the EMADS main container is located on the EMADS host organisation site, while the other containers may be held at any other sites worldwide.

Figure 2 presents the EMADS architecture as implemented in JADE. It shows a sample collection of several application agents and housekeeping agents, organized as a community of peers by a common relationship to each other.

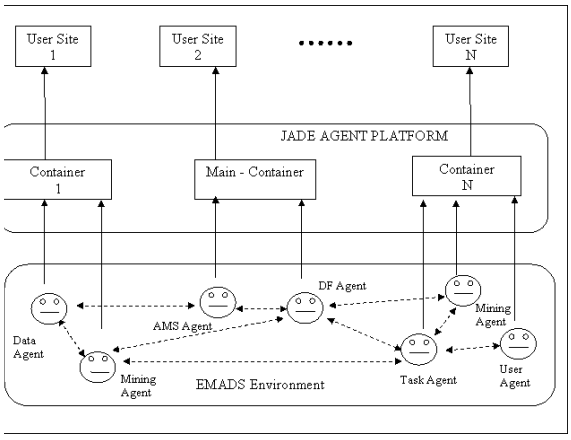


Fig. 2. EMADS Architecture as Implemented in Jade

With reference to Figure 2, a user agent runs on the user's local host and is responsible for: (i) accepting user input (request), (ii) launching the appropriate Task Agent to processes the user request, and (iii) displaying the results of the (distributed) computation. The user expresses a task to be executed using standard interface dialogue mechanisms by clicking on active areas in the interface and, in some cases, by entering thresholds values; note that the user does not need to specify which agent or agents should be employed to perform the desired task. For instance, if the question "What is the best classifier for my data?" is posed in the user interface, this request will trigger a Task Agent. The Task Agent requests the facilitator to match the action part of the request to capabilities published by other agents. The request is then routed by the Task Agent to the appropriate agents to execute the request, this will typically involve communication among various relevant agents within the system. On completion the results are sent back to the user agent for display.

The key elements of the operation of EMADS that should be noted are:

i. The mechanism whereby a collection of agents can be harnessed to identify a "best solution".
ii. The process whereby new agents connect to the facilitator and registering their capability specifications.
iii. That the interpretation and execution of a task is a distributed process, with no one agent defining the set of possible inputs to the system.
iv. That a single request can produce cooperation and flexible communication among many agents spread across multiple machines.

Agent Cooperation Cooperation among the various EMADS agents is achieved via messages expressed in FIPA ACL and is normally structured around a three-stage process:

i. **Service Registration** where providers (agents who wish to provide *services*) register their capability specifications with a facilitator.
ii. **Request Posting** where User Agents (*requesters* of services) construct requests and relay them to a Task Agent,
iii. **Processing** where the Task Agent coordinates the efforts of the appropriate service providers (Data Agents and DM Agents) to satisfy the request.

Note that Stage 1 (service registration) is not necessarily immediately followed by stage 2 and 3, it is possible that a provider services may never be used. Note also that the facilitator (the DF and AMS agents) maintains a knowledge base that records the capabilities of the various EMADS agents, and uses this knowledge to assist requesters and providers of services in making contact. When a service provider (i.e. Data Agent or DM Agent) is created, it makes a connection to a facilitator, which is known as its *parent facilitator*. Upon connection, the new agent informs its parent facilitator of the services it can provide. When the agent is needed, the facilitator sends its address to the requester agent. An important element of the desired EMADS agent cooperation model is the function of the Task Agent; this is therefore described in more detail in the following subsection.

The Task Agent A Task Agent is designed to handle a user request. This involves a three step process:

i. Determination of whom (which specific agents) will execute a request;
ii. Optimization of the complete task, including parallelization where appropriate; and
iii. Interpretation of the optimized task.

Thus determination (step 1) involves the selection of one or more agents to handle each sub-task given a particular request. In doing this, the Task agent uses the facilitator's knowledge of the capabilities of the available EMADS agents (and possibly of other facilitators, in a multi-facilitator system). The facilitator may also use information specified by the user (such as threshold values). In processing a request, an agent can also make use of a variety of capabilities provided by other agents. For example, an agent can request data from Data Agents that maintain shared data. The optimization step results in a request whose interpretation will require as few communication exchanges as possible, between the Task Agent and the satisfying agents (typically DM Agents and Data Agents), and can exploit the parallel processing capabilities of the satisfying agents. Thus, in summary, the interpretation of a task by a Task Agent involves: (i) the coordination of requests directed at the satisfying agents, and (ii) assembling the responses into a coherent whole, for return to the user agent.

3.3 System Extendibility

One of the principal objectives of the EMADS framework is to provide an easily extendable framework that can accept new data sources and new DM techniques. In general, extendibility can be defined as the ease with which software can be modified to adapt to new requirements or changes in existing requirements. Adding a new data source or DM techniques should be as easy as adding new agents to the system. The desired extendibility is achieved by a system of wrappers. EMADS wrappers are used to "wrap" up DM artifacts so that they become EMADS agents and can communicate with other EMADS agents. Such EMADS wrappers can be viewed as agents in their own right that are subsumed once they have been integrated with data or tools to become EMADS agents. The wrappers essentially provide an application interface to EMADS that has to be implemented by the end user, although this has been designed to be a fairly trivial operation.

In the current demonstration EMADS system two broad categories of wrapper have been defined: (i) data wrappers and (ii) tool wrappers; the first is used to create data agents and the second to create DM agents. Each is described in further detail in the following two subsections.

Data Wrappers Data wrappers are used to "wrap" a data source and consequently create a data agent. Broadly a data wrapper holds the location (file path) of a data source, so that it can be accessed by other agents; and meta information about the data. To assist end users in the application of data wrappers a data wrapper GUI is available. Once created, the data agent announces itself to the DF agent as a consequence of which it becomes available to all EMADS users.

Tool Wrappers Tool wrappers are used to "wrap" up DM software systems and thus create a mining agent. Generally the software systems will be DM tools of various kinds (classifiers, clusters, association rule miners, etc.) although they could also be (say) data normalization/discretization or visualization tools. It is intended that the framework will incorporate a substantial number of different tool wrappers each defined by the nature of the desired I/O which in turn will be informed by the nature of the generic DM tasks that it is desirable for EMADS to be able to perform. Currently the research team has implemented two tool wrappers:

i. The binary valued data, single label, classifier generator.
ii. The data normalization/discretization wrapper.

Many more categories of tool wrapper can be envisaged. Mining tool wrappers are more complex than data wrappers because of the different kinds of information that needs to be exchanged.

In the case of a "binary valued, single label, classifier generator" wrapper the input is a binary valued data set together with meta information about the number of classes and a number slots to allow for the (optional) inclusion of threshold values. The output is then a classifier expressed as a set of Classification Rules (CRs). As with data agents, once created, the DM agent announce themselves to the DF agent after which they will becomes available for use to EMADS users.

For example, in the case of the data normalization/discretization wrapper, the LUCS-KDD (Liverpool University Computer Science - Knowledge Discovery in Data) ARM DN (Discretization/ Normalization) software [1] is used to convert data files, such as those available in the UCI data repository [8], into a binary format suitable for use with Association Rule Mining (ARM) applications. This tool has been "wrapped" using the data normalization/discretization wrapper.

4 System Demonstration

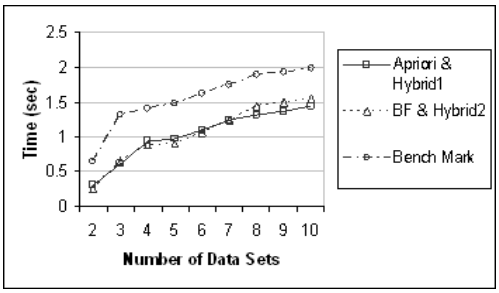
Perhaps the best way to obtain an intuitive sense of how the framework typically functions is to briefly look at an example of how it has been applied to real world scenarios. The following subsections describe two demonstration applications (Scenarios) implemented within the EMADS framework.

The first (discussed further in subsection 4.1) is a distributed meta mining scenario where EMADS agents are used to merge the results of a number of ARM operations, a process referred to as meta-ARM, to produce a global set of Association Rules (ARs). The challenge here is to minimise the communication overhead, a significant issue in distributed and parallel DM (regardless of whether it is implemented in an agent framework or not).

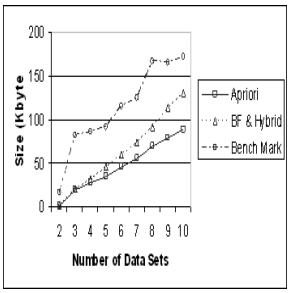
The second scenario (subsection 4.2) is a classification scenario where the objective is to generate a classifier (predictor) fitted to EMADS user's specified data set. It has been well established within the DM research community, for reasons that remain unclear but are concerned with the nature of the input data, that there is no single

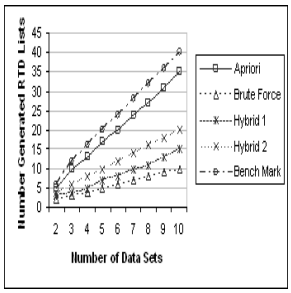
[1] $http://www.csc.liv.ac.uk/\tilde{f}rans/KDD/Software/$

"best" classification algorithm. The aim of this second scenario is therefore to identify a "best" classifier given a particular data set. Best in this context is measured in terms of classification accuracy. This experiment not only serves to illustrate the advantageous of EMADS but also provides an interesting comparison of a variety of classification techniques and algorithms.



(a) Processing Time





(b) Total size of RTD lists (c) Number of RTD lists

Fig. 3. Effect of number of data sources

4.1 Meta ARM (Association Rule Mining) scenario

The term *meta mining* is defined, in the context of EMADS, as the process of combining the individually obtained results of N applications of a DM activity. The motivation behind the scenario is that data relevant to a particular DM application may be owned and maintained by different, geographically dispersed, organizations. There is therefore a "privacy and security" issue, privacy preserving issues [1] are of major concerns in inter enterprise DM when dealing with private databases located at different sites. One approach to addressing the meta mining problem is to adopt a distributed approach. The meta mining scenario considered here is a meta Association Rule Mining (meta ARM) scenario where the results of N ARM operations, by N agents, are brought together.

Dynamic Behaviour of System for Meta ARM operations The meta ARM EMADS illustration described here was used to identify the most efficient Meta ARM agent process given a number of alternatives. The first algorithm was a bench mark algorithm, against which other Meta ARM algorithms were compared. Four comparison meta ARM algorithms were constructed (Apriori, Brute Force, Hybrid 1 and Hybrid 2). Full details of the algorithms can be found in [3]. In each case it was assumed that each data source would produce a set of frequent sets, using some ARM algorithm, with the results stored in a common data structure. These data structures would then be merged in some manner through a process of agent collaboration. Each of the Meta ARM algorithms made use of a Return To Data (RTD) lists, one per data set, to contain lists of itemsets whose support was not included in the original ARM operation and for which the count was to be obtained by a return to the raw data held at a data agent. The RTD lists comprised zero, one or more tuples of the form $< I, sup >$, where I is an item set for which a count is required and sup is the desired count. RTD lists are constructed as a meta ARM algorithm progresses. During RTD list construction the sup value will be 0, it is not until the RTD list is processed that actual values are assigned to sup. The processing of RTD lists may occur during, and/or at the end of, the meta ARM process depending on the nature of the meta ARM algorithm used.

The meta ARM scenario comprises a set of N data agents and $N+1$ DM agents: N ARM agents and one meta ARM agent. Note that each ARM agent could have a different ARM algorithm associated with it, however a common data structure was assumed to facilitate data interchange. The common data structure used was a T-tree [12], a set enumeration tree structure for storing item sets. Once generated the N local T-trees were passed to the Meta ARM agent which created a global T-tree. During the global T-tree generation process the Meta ARM agent interacted with the various ARM agents in the form of the exchange of RTD lists.

Experimentation and Analysis To evaluate the five Meta ARM algorithms (including the bench mark algorithm), in the context of the EMADS vision, a number of experiments were conducted. These are described and analyzed in this subsection. The experiments were designed to analyze the effect of the following:

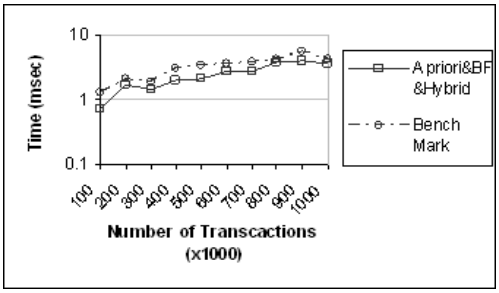
i. The number of data sources (data agents).
ii. The size of the data sets (held at data agents) in terms of number of records.
iii. The size of the data sets (held at data agents) in terms of number of attributes.

Experiments were run using two Intel Core 2 Duo E6400 CPU (2.13GHz) computers with 3GB of main memory (DDR2 800MHz), Fedora Core 6, Kernel version 2.6.18 running under Linux except for the first experiment where two further computers running under Windows XP were added. For each of the experiments we measured:

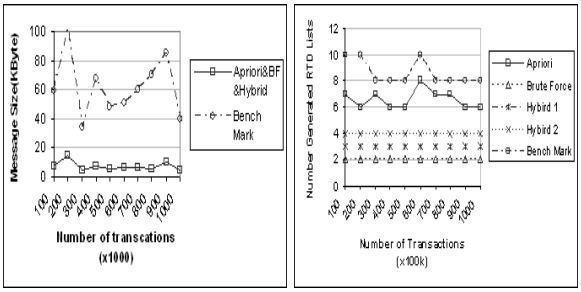
- Processing time (seconds/mseconds),
- The size of the RTD lists (Kbytes), and
- The number of RTD lists generated.

Note that the authors did not use the well known IBM QUEST generator [2] because many different data sets (with the same input parameters) were required and it was

found that the QUEST generator always generated the same data given the same input parameters. Instead the authors used the LUCS KDD data generator[2]. Figure 3 shows the effect of adding additional data sources using the four Meta ARM algorithms and the bench mark algorithm. For this experiment thirteen different artificial data sets were generated and distributed among four machines using $T = 4$ (average number of items per transactions), $N = 20$ (Number of attributes), $D = 100k$ (Number of transactions. Note that the slight oscillations in the graphs result simply from a vagary of the random nature of the test data generation.



(a) Processing Time



(b) Total size of RTD lists (c) Number of RTD lists

Fig. 4. Effect of number of data sources

Figure 4 demonstrates the effect of increasing the number of records. The input data for this experiment was generated by producing a sequence of ten pairs of data sets (with T =4, N =20) representing two sources on two differentmachines. From graph 4(a) it can be seen that the Brute Force and Hybrid 1 algorithms work best because the size of the return to data lists are limited as no unnecessary candidate sets are generated. This is illustrated in graph 4(b). Graph 4(b) also shows that the increase in processing time

[2] $http://www.csc.liv.ac.uk/\tilde{f}rans/KDD/Software//LUCS-KDD-DataGen/$

in all cases is due to the increase in the number of records only; the size of the RTD lists remains constant throughout as does the number of RTD lists generated (graph 4(c)).

Figure 3 and 4 also indicate, at least with respect to meta ARM, that EMADS offers positive advantages in that all the Meta ARM algorithms were more computationally efficient than the bench mark algorithm. The results of the analysis also indicated that the Apriori Meta ARM approach coped best with a large number of data sources, while the Brute Force and Hybrid 1 approaches coped best with increased data sizes (in terms of column/rows).

4.2 Classifier Generation scenario

In this subsection the operation of EMADS is illustrated in the context of a classifier generation task, however much of the discussion is equally applicable to other generic DM tasks. The scenario is that of an end user who wishes to obtain a "best" classifier founded on a given, pre-labelled, data set; which can then be applied to further unlabelled data. The assumption is that the given data set is binary valued and that the user requires a single-label, as opposed to a multi-labelled, classifier. The request is made using the individual's user agent which in turn will spawn an appropriate task agent.

For this scenario the task agent interacts with mining agents that hold single labelled classifier generators that take binary valued data as input. Each of these mining agents is then accessed and a classifier, together with an accuracy estimate, requested. Once received the task agent selects the classifier with the best accuracy and returns this to the user agent.

The DM agent wrapper in this case provides the interface that allows input of: (i) the identifier for the data set to be classified, and (ii) the number of class attributes (a value that the mining agent cannot currently deduce for itself); while the user agent interface allows input for threshold values (such as support and confidence values).

The output is a classifier together with an accuracy measure. To obtain the accuracy measures the classifier generators (DM agents) build their individual classifier using the first half of the input data as the "training" set and the second half of the data as the "test" set. An alternative approach might have been to use Ten Cross Validation (TCV) to identify the best accuracy. It should be noted that the objective here is to return a classifier, using TCV ten classifiers will be built and thus one of them would have to be selected.

From the literature there are many reported techniques available for generating classifiers. For the scenario reported here the authors' used implementations of eight different algorithms[3]:

i. FOIL (First Order Inductive Learner) [27]: The well established inductive learning algorithm for the generation of Classification Association Rules (CARs).
ii. TFPC (Total From Partial Classification): A CAR generator [13] founded on the P and T-tree set enumeration tree data structures.
iii. PRM (Predictive Rule Mining) [31]: An extension of FOIL.
iv. CPAR (Classification based on Predictive Association Rules) [31]: A further development from FOIL and PRM.

[3] Taken from the LUCS-KDD repository at $http://www.csc.liv.ac.uk/\tilde{f}rans/KDD/Software/$

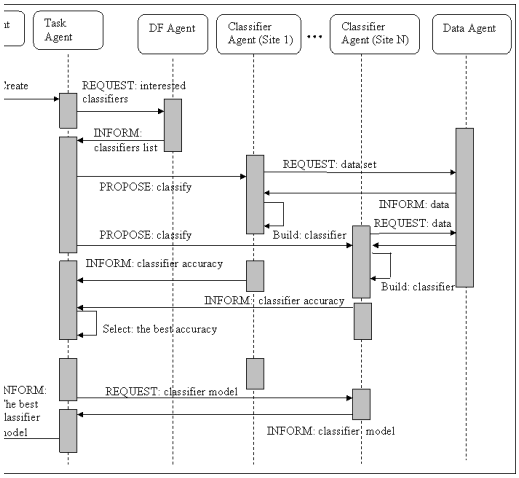


Fig. 5. Classification Task Sequence Diagram.

v. IGDT (Information Gain Decision Tree) classifier: An implementation of the well established C4.5 style of decision tree based classifier using information gain as the "splitting criteria".
vi. RDT (Random Decision Tree) classifier: A decision tree based classifier that uses most frequent current attribute as the "splitting criteria".
vii. CMAR (Classification based on Multiple Association Rules): A well established Classification Association Rule Mining (CARM) algorithm [22].
viii. CBA (Classification Based on Associations): Another well established CARM algorithm [21].

These were placed within an appropriately defined tool wrapper to produce eight (single label binary data classifier generator) DM agents. This was found to be a trivial operation indicating the versatility of the wrapper concept.

Thus each mining agent's basic function was to generate a classification model using its own classifier and provide this to the task agent. The task agent then evaluates all the classifier models and chooses the most accurate model to be returned to the user agent. The negotiation process amongst the agents is represented by the sequence diagram given in Figure 5 (the figure assumes that appropriate data agents exist). The figure includes *N* classification agents. The sequence of processing events commences with a user agent which spawns a (classification) task agent, which in turn announces itself to the DF agent. The DF agent returns a list of classifier DM agents that can potentially be used to generate the desired classifier.

The task agent then interacts with these DM agents who each generate a classifier and return statistical information regarding the accuracy of their classifier. The task agent selects the DM agent that has produced the best accuracy and requests the associated classifier; this is then passed back to the user agent.

Table 1. Classification Results

Data Set	Classifier	Accuracy	Time(sec)
connect4.D129.N67557.C3	RDT	79.76	502.65
adult.D97.N48842.C2	IGDT	86.05	86.17
letRecog.D106.N20000.C26	RDT	91.79	31.52
anneal.D73.N898.C6	FOIL	98.44	5.82
breast.D20.N699.C2	IGDT	93.98	1.28
congres.D34.N435.C2	RDT	100	3.69
cylBands.D124.N540.C2	RDT	97.78	41.9
dematology.D49.N366.C6	RDT	96.17	11.28
heart.D52.N303.C5	RDT	96.02	3.04
auto.D137.N205.C7	IGDT	76.47	12.17
penDigits.D89.N10992.C10	RDT	99.18	13.77
soybean.D118.N683.C19	RDT	98.83	13.22
waveform.D101.N5000.C3	RDT	96.81	11.97

Experimentation and Analysis To evaluate the EMADS classification scenario, as described above, a sub-set of the data sets available at the UCI machine learning data repository [8] were used (preprocessed by data agents so that they were discretized/normalized into a binary form). The results are presented in Table 1. Each row in the table represents a particular request and gives the name of the data set, the selected best algorithm as identified from the interaction between the EMADS agents, the resulting best accuracy and the total EMADS execution time from creation of the initial Task Agent to the final "best" classifier being returned to the user agent. The naming convention used in the Table is that: D equals the number of attributes (after discretization/normalization), N the number of records and C the number of classes (although EMADS has no requirement for the adoption of this convention).

The results demonstrate firstly, that EMADS can usefully be adopted to produce a best classifier from a selection of classifiers. Secondly, that the operation of EMADS is not significantly hindered by agent communication overheads, although this has some effect. Generation time, in most cases does not seem to be an issue, so further classifier generator mining agents could easily be added. The results also reinforce the often observed phenomena that there is no single best classifier generator suited to all kinds of data.

5 Conclusions

This paper described EMADS, a multi-agent framework for DM. The architecture provides a framework for the construction and operation of distributed software agents. The principal advantages offered by the system are that of experience and resource sharing, flexibility and extendibility, and (to an extent) protection of privacy and intellectual property rights. The use of a single facilitator offers both advantages and weaknesses with respect to scalability and fault tolerance. On the plus side, the grouping of a facilitator with a collection of agents provides a faster look-up service. However, even

though the intention is that the facilitator assists agents in finding one another and then to "step aside" while other agents communicate over a direct, dedicated channel so as to prevent a communication bottleneck; there is still the potential for a facilitator to become a critical point of system failure. Further work in this area is therefore required, one solution is to use more than one facilitator deployed on multiple machines for a better fault-tolerant platform.

The framework's operation is illustrated using both meta ARM and classification scenarios. Extendibility is presented by showing how wrappers are used to incorporate existing software into EMADS. Experience to date indicates that, given an appropriate wrapper, existing DM software can very easily be packaged to become a DM agent. Flexibility is illustrated using a classification scenario. Information hiding is illustrated in that users need have no knowledge of how any particular piece of DM software works or the location of the data to be used.

A good foundation has been established for both DM research and genuine application based DM. It is acknowledged that the current functionality of the framework is limited to classification and meta ARM. The research team is at present working towards increasing the diversity of mining tasks that can be addressed. There are many directions in which the work can (and is being) taken forward. One interesting direction is to build on the wealth of distributed DM research that is currently available and progress this in an MAS context. The research team is also enhancing the system's robustness so as to make it publicly available. It is hoped that once the system is live other interested DM practitioners will be prepared to contribute algorithms and data.

References

1. Aggarwal, C. and Yu, P., "A Condensation Approach to Privacy Preserving DataMining". Lecture Notes in Computer Science, Vol. 2992, pp. 183-199, (2004).
2. Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A. and Bollinger, T., "The Quest Data Mining System". Proceedings 2nd Int. Conf. Knowledge Discovery and Data Mining, KDD, (1996).
3. Albashiri, K., Coenen, F., Sanderson, R. and Leng. P., "Frequent Set Meta Mining: Towards Multi-Agent Data Mining". In Bramer, M., Coenen, F.P. and Petridis, M. (Eds.), Research and Development in Intelligent Systems XXIII., Springer, London, pp. 139-151, (2007).
4. Albashiri, K., Coenen, F., and Leng. P., "Agent Based Frequent Set Meta Mining: Introducing EMADS". Artificial Intelligence in Theory and Practice II, Proceedings IFIP, Springer, pp23-32, (2007).
5. Baazaoui H., Faiz S., Ben Hamed R., Ben Ghezala H., "A Framework for data mining based multi-agent: an application to spatial data". 3rd World Enformatika Conference WEC'05, Avril, Istanbul, (2005).
6. Bailey, S., Grossman, R., Sivakumar, H. and Turinsky, A., "Papyrus: a system for data mining over local and wide area clusters and super-clusters". In Proceedings Conference on Supercomputing, page 63. ACM Press, (1999).
7. Bellifemine, F. Poggi, A. and Rimassi, G., "JADE: A FIPA-Compliant agent framework". Proceedings Practical Applications of Intelligent Agents and Multi-Agents, pp 97-108, (1999). (See http://sharon.cselt.it/projects/jade).
8. Blake, C. and Merz, C. , "UCI Repository of machine learning databases". Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/mlearn/MLRepository.html, (1998).

9. Bose, R. and Sugumaran, V., “IDM: An Intelligent Software Agent Based DataMining Environment”. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2888-2893 San Diego, CA: IEEE Press, (1998).
10. Bota, J., Gmez-Skarmeta, A., Valds, M., and Padilla, A.,“Metala: A meta-learning architecture”. Fuzzy Days, pages 688-698, (2001).
11. Cao, L., and Zhang, C., “F-Trade: Agent-mining symbiont for financial services”. AAMAS, pp.1363-1364, (2007).
12. Coenen, F., Leng, P., and Goulbourne, G., “Tree Structures for Mining Association Rules”. Journal of Data Mining and Knowledge Discovery, Vol 8, No 1, pp25-51, (2004).
13. Coenen, F., Leng, P. and Zhang, L. “Threshold Tuning for Improved Classification Association Rule Mining”. Proceeding PAKDD, LNAI3158, Springer, pp216-225, (2005).
14. Foundation for Intelligent Physical Agents, FIPA 2002 Specification. Geneva, Switzerland. (See http://www.fipa.org/specifications/index.html), (2002).
15. Giuseppe, D., Giancarlo, F., “A customizable multi-agent system for distributed data mining”. Proceedings of the 2007 ACM symposium on applied computing, Pages: 42 47, (2007).
16. Gorodetsky, V., Karsaeyv, O., Samoilov, V., “Multi-agent technology for distributed data mining and classification”. Proceedings Int. Conf. on Intelligent Agent Technology (IAT 2003), IEEE/WIC, pp438-441, (2003).
17. Grossman, R. and Turinsky, A., “A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies”, In KDD Workshop on Distributed Data Mining, (2000).
18. Kargupta, H., Byung-Hoon, et al., “Collective Data Mining: A New Perspective Toward Distributed Data Mining”. Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, (1999).
19. Kargupta, H., Hamzaoglu, I. and Stafford B., “Scalable, Distributed Data Mining Using an Agent Based Architecture”. Proceedings of Knowledge Discovery and Data Mining, AAAI Press, 211-214, (1997).
20. Klusch, M., Lodi, G., “Agent-based Distributed Data Mining: The KDEC Scheme. Intelligent Information Agents”. The AgentLink Perspective. Lecture Notes in Computer Science 2586, Springer, (2003).
21. Liu, B. Hsu, W. and Ma, Y., “Integrating Classification and Assocoiation Rule Mining”. Proceedings KDD-98, New York, 27-31 August. AAAI. pp80-86, (1998).
22. Li W., Han, J. and Pei, J., “CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules”. Proceedings ICDM, pp369-376, (2001).
23. Luo, P., Huang, R., He, Q., Lin, F., and Shi, Z., “Execution engine of meta-learning system for kdd in multi-agent environment”. Technical report, Institute of Computing Technology, Chinese Academy of Sciences, (2005).
24. METAL Project. Esprit Project METAL, (2002). http://www.metal-kdd.org
25. Peng, S., Mukhopadhyay, S., Raje, R., Palakal, M. and Mostafa, J., “A Comparison Between Single-agent and Multi-agent Classification of Documents”. Proceedings 15th International Parallel and Distributed Processing Symposium, pp935-944, (2001).
26. Prodromides, A., Chan, P. and Stolfo, S., “Meta-Learning in Distributed Data Mining Systems: Issues and Approaches”. In Kargupta, H. and Chan, P. (Eds), AAAI Press/The MIT Press, pp81-114, (2000).
27. Quinlan, J. R. and Cameron-Jones, R. M., “FOIL: A Midterm Report”. Proc. ECML, Vienna, Austria, pp. 3-20, (1993).
28. Shi, Z., Zhang, H., Cheng, Y., Jiang, Y., Sheng, Q., and Zhao, Z., “Mage: An agent-oriented programming environment”. In Proceedings of the IEEE International Conference on Cognitive Informatics, pp 250-257, (2004).

29. Stolfo, S., Prodromidis, A. L., Tselepis, S. and Lee, W., “JAM: Java Agents for Meta-Learning over Distributed Databases”. Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 74-81, (1997).
30. Vilalta, R., Christophe, G., Giraud-Carrier, Brazdil, P., Soares, C., “Using Meta-Learning to Support Data Mining”. IJCSA 1(1): 31-45, (2004).
31. Yin, X. and Han, J., “CPAR: Classification based on Predictive Association Rules”. Proc. SIAM Int. Conf. on Data Mining (SDM’03), SF, CA, pp. 331-335,(2003).

Agent Assignment for Process Management: Pattern based Agent Performance Evaluation

Stefan Jablonski and Ramzan Talib

University of Bayreuth, Applied Informatics IV
D-95440 Bayreuth, Germany
Stefan.Jablonski@uni-bayreuth.de, Ramzan.Talib@uni-bayreuth.de

Abstract. In almost all workflow management system the role concept is determined once at the introduction of workflow application and is not reevaluated to observe how successfully certain processes are performed by the authorized agents. This paper describes an approach which evaluates how agents are working successfully and feed this information back for future agent assignment to achieve maximum business benefit for the enterprise. The approach is called Pattern based Agent Performance Evaluation (PAPE) and is based on machine learning technique combined with post processing technique. We report on the result of our experiments and discuss issues and improvement of our approach.

Keywords: Agent Assignment, Workflow, Process, Machine Learning, Business Benefit, Pattern based Agent Performance Evaluation.

1 Introduction

Many applications can be described by processes. For example, in a garment factory a garment production process starts with the assignment of designers to design tasks (e.g. to design a new T-shirt), continues with the design of a new garment, continues with its production, and finally ends up with selling the new products.

Of course, from a business (benefit) perspective there is a close relationship between the multiple steps of such a process. We focus in this article the dependency of good sales figures from the assignment of designers to design the tasks. Substantiated by our experience with garment production it is obvious that certain designers are delivering excellent designs for specific goods, while other designers only deliver moderate designs for these goods which lead to only moderate sales figures. The main two questions in the context of this observation are: .

- How to get aware of dependencies like the above one?
- How to respond to such an observation?

We want to discuss these two questions in different applications contexts: the first one is the conventional one, where designer assignment is done by responsible managers; the second one is characterized by the deployment of workflow management systems which automatically assign designers to design task.

In the conventional case it depends on the experience and the analytical knowledge of responsible managers to assign the right staff to the right tasks: due to intensive

observations managers find out which staff (agent) assignment results in best business benefit. When managers recognize that specific persons are guaranteeing best business benefit they consider this in future assignments. However, often these observations are not substantiated by analyzing production and sales data but are heavily determined by personal experiences and observations of responsible managers.

When automatic staff assignment like in workflow management systems takes place, personal experience is normally not incorporated into staff assignment. People are assigned to process steps if they are eligible to perform such a step. Whether these eligible persons are producing good or moderate sales figures can normally not be considered at the assignment [12]. In Section 2.2 we will detail staff assignment in workflow management; this will back up our observation.

In order to improve staff assignment two things have to be accomplished. First, business intelligence has to be introduced to detect dependencies between staff assignment and business benefit. In the conventional case, responsible managers could exploit this knowledge to perform more adequate staff assignment. In the case of automatic task assignment the results of business intelligence must be fed back into automatic task assignment policies.

In this paper we want to focus automatic task assignment of workflow management systems since it requires a kind of superset of solution strategies as manual task assignment needs. Concretely, we firstly want to identify effective and efficient forms of business intelligence for staff assignment in workflow management applications. Secondly, we want to define a method how the findings of business intelligence can be incorporated into staff assignment policies to produce more productive staff assignments which leads to better business performance.

Next, a short example will be depicted which will illustrate the dependency between agent assignment and the success of process execution. Again, we want to refer to a scenario from garment industry. Agent assignment is about the assignment of designers to special design tasks. The success of performance of a business process as a whole and the designer assignment in detail is measured by sales figures. Hereby, it has to be mentioned that business benefit here is in the form of good sales figures.

How does agent assignment work? Before agents can be assigned to processes (in order to perform them) they are characterized and classified. For instance, some employees of a garment company are classified as designers; other employees are classified as sale assistants or purchasers. For each step of a process eligible agents are determined. For example, designers are assigned to the process step ***Create Design***; sales assistants are assigned to the process step ***Define Selling Strategy***.

Consider a process ***Create Design*** of a garment enterprise process. Agents who selected this process step are responsible to create a specific design (e.g. shirt, pants). Furthermore, designs might be classified into different types. For example, shirt might be specialized into the sub-classes T-shirt and Blouse. The type of design that is needed is determined by some input data for the ***Create Design*** process step such that the respective designers knows what kind of design has to be created.

Each design created by an agent in the ***Create Design*** process step has different sales values (e.g. GOOD, MODERATE). Good sales value of a design depends on the

Table 1. Data pattern and agent performance

	Shirt Patterns			
	T-Shirt		Blouse	
Business benefit	**GOOD**	**MODERATE**	**GOOD**	**MODERATE**
Gabi	84	6	13	72
Clara	19	78	38	5

quality of design and therefore directly on the performance and expertise of the assigned designer, e.g. the agent assigned to the process ***Create Design***.

We now want to correlate agent assignment and products that had to be designed and do achieve business benefits. Table 1 depicts this correlation. Statistics in Table 1 shows that Gabi has good expertise of creating T-shirt designs (84 designs for T-shirt out of 90 designs result in good business benefits, i.e. good sales values) while Clara has expertise of creating Blouse designs. Besides, Gabi's expertise in designing Blouse is rather moderate (most of his designs result in moderate sales values); almost the same holds for Clara's design for T-shirt. Relating the results of Table 1 to our two contributions the following two tasks can be identified:

- It is necessary to detect relationships between type of designs (design patterns), agent assignment and achieved business benefits. We apply machine learning techniques to accomplish this task.
- It is necessary to provide a feedback cycle between encountered design patterns and future agent assignments.

Design patterns are general design groups which predict business benefit more accurately. They can be expressed as an expression of attributes-values of design data. Machine learning technique can be used to learn data patterns from any dataset. For the example from Table 1 this means: Identify relationships between different design patterns, agent assignments and achieved business benefits, first. Then take this knowledge and provide it for further selections of agents for specific design tasks.

Currently in WFMS, agents assignments are usually defines once and are not reevaluated whether agents do their job good or bad. From a business point of view our approach leads to two major benefits: First, relationships between data patterns and agent performance can be detected. This result is valuable for both conventional agent assignment and for agent assignments performed by WFMS. Second, in the latter case our approach guarantees automatic agent assignments promising best business benefit. Again, it is completely application dependent how business benefit is defined. This has to be done by domain experts before our approach is applied.

We call our approach Pattern Based Agent Performance Evaluation (PAPE). It consists of five major phases: In the first phase, domain knowledge is incorporated to find how business benefits are defined in the application. Second phase generates integrated data structure combining the machine learning technique and post-processing technique. This integrated data structure is used to evaluate the dependency between

agent assignment and success of process executions in the third phase. If a dependency is observed agent assignment rules are learned in the fourth phase and are presented to domain expert for further refinement. Rules finally refined by domain expert are updated in the WFMS (fifth phase).

As a prerequisite we need a sort of an event log (audit trail) which stores both data about agent assignments and achieved sales figures. Most information systems like WFMS, ERP, CRM, and B2B systems provide such event logs [8]. Especially, we will concentrate in this paper on the application of our method to WFMS which typically maintain such a log.

The remainder of the paper is organized as follows. We provide a general overview on workflow management systems and staff assignment strategies in Section 2. Section 3 explains our methodology. In Section 4 we are presenting the result of applying our method on real data sets. We discuss related work in Section 5. Section 6 concludes this paper and provides an outlook on future work.

2 Workflow Management

2.1 Overview

Workflow management is a technology which aims at the execution of workflow steps. It is concerned with providing the right information to support each workflow step. Also programs and tools, needed to perform the work steps, are provided by agents who are determined by a workflow management system as well. Finally data are routed between work steps that are either consumed and/or produced by them.

In this article we use the terms “process” and “workflow” synonymously though we principally see a difference between them [6]. However, in this article we always concentrate on the execution aspect and therefore want to take them as synonyms.

According to [6], a process consists of five major perspectives which altogether define how processes are executed. The functional perspective defines the skeleton of a process: it identifies the process and comprises general information about the process. Also, the functional perspective determines whether the process is elementary or composite. The behavioral perspective determines the control flow, i.e. the order of process execution. The operational perspective describes tools (programs, systems etc.) that are available for the execution of a process. The data perspective defines input data and output data of a process. This perspective consequently then determines data flow within a process. Last but not least the organizational perspective determines agents who are eligible to perform a certain process. This task is usually called agent assignment [4].

For this article the organizational perspective is the most interesting one. Nevertheless, our research could also be applied to other perspectives of a workflow. We already have investigated this issue; it will be presented in future articles.

The main task of a workflow management system is to coordinate all mentioned perspectives when a process has to be performed. In Figure 1 a small example is depicted. It shows two out of many processes that are executed during the design process. In the first step shown the garment collection (e.g. for the next season) has to be determined. The agent “Marketing” (i.e. the corresponding department) is responsible for

that. It uses an ERP application in order to define the next collection. The output of this process is "Design" which is a compound data item that describes all designs that have to be done for the next garment collection. This data is passed as input to the next step "Create Design". Here designers (agents) have to create designs utilizing a so called application "DesignApp".

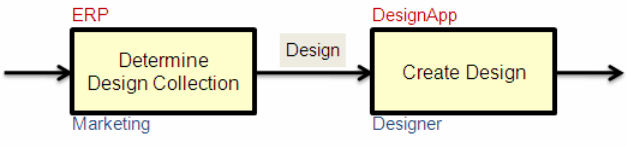


Fig. 1. Create design process in garment factory

A process management system has to take care that all process steps are executed in the right order, consuming the right input data, producing the right output data, applying the right applications, and last but not least select the right agents who have to perform a step. This last task will be considered in detail in the next sub-section.

2.2 Overview of Agent Assignment Strategies

In order to determine agents who have to perform a certain process the following steps are executed. First, agents are classified according their capabilities. Normally, they are associated to certain roles (e.g. designer, clerk). Second, those roles are used to describe a process step (organizational perspective). When a process step is executed its organizational perspective, i.e. the associated role is taken and all agents that are corresponding to such a role are informed about work to do. One of the eligible agents (mostly the fastest one) then claims that piece of work and has to perform it.

Here we present a short example of agent assignment rules for the ***Create Design*** process we are using throughout this article. The first rule shows which agents belong to the ***Designer*** role; the second rule shows that agents with the Designer role can execute the ***Create Design*** process.

- $RoleDesigner[] = [\text{“}Gabi\text{”}, \text{“}Clara\text{”}, \text{“}Jan\text{”}, \text{“}Eva\text{”}, \text{“}Bali\text{”}, \text{“}Emil\text{”}]$;
- $Performer.Role() = Designer$;

In almost all workflow management systems the role concept is coarse grained (i), is determined once at the introduction of the process application (ii), and is not integrated in a feedback loop (iii). The first argument means that roles are provided as "designer" or - and this is preferable - "Shirt Designer", "Pants Designer", etc. The second argument means that these role associations are done once and are not updated continuously. The third and most important argument means that it is not evaluated, i.e. it is not observed how successful certain tasks are performed in order to improve role assignment. For example, if a "Shirt Designer" would always produce designs which are very bad,

(s)he should be deprived of this role. All in all, agent assignment in conventional process management systems could be characterized as static.

Now, it is appropriate to re-state the contribution of this paper. First, we contribute an approach to determine how agents are working successfully, i.e. whether they are contributing to the business benefit. Second, we feed this information back into upcoming agent assignments, i.e. we associate those agents with process step who (most probably) guarantee a successful outcome of the whole process.

3 PAPE System Architecture

The proposed PAPE method evaluates dependency between data patterns, agent assignment and business benefit to learn agents working successfully and feeds back this knowledge for future agent assignments to achieve maximum business benefits.

How to evaluate the dependency of agent assignment and business success? Classical data mining techniques are data centric and solve business issues in an isolated way [5]. Thus, often impact of other domain resources or resource interdependency is lost. For the PAPE method, machine learning classifier can easily be employed to determine how design patterns predict business benefit (GOOD or MODERATE) but cannot directly be employed to evaluate the consequence of agent's expertise for GOOD or MODERATE business benefit. Moreover, for the respective design pattern, these classifiers cannot predict best agent promising best business benefit.

Fig. 2. PAPE method

So, to subjugate this problem and to learn dependency of agent assignment and business benefit the PAPE method uses machine learning technique (decision tree) combined with post-processing technique to generate an integrated data structure. Our method uses the workflow audit trail for a particular time period and generates integrated data structure which is used to evaluate the dependency between data patterns, agent assignments and business benefit. If this dependency is observed then performance based agent assignment rules are learned. These rules are more productive and are provided as a feedback for automatic agent assignments in WFMS which leads to successful process execution and better business performance.

The PAPE method is shown in Figure 2. It consists of five phases: Preprocessing Phase (Section 3.1), Integrated Data Structure Generation Phase (Section 3.2), Pattern Based Agent Performance Evaluation Phase (Section 3.3), Agent Assignment Learning Phase (Section 3.4) and Agent Assignment Updating Phase (Section 3.5).

3.1 Preprocessing Phase

In the preprocessing phase the workflow log is processed by domain expert and domain knowledge is incorporated to prepare the dataset to be used by PAPE method. Since it is completely application dependent how business benefits are defined, domain knowledge is required. Domain knowledge is used to decide different issues which are discussed in the following sub-sections as a guide line for preparation of dataset suitable for the PAPE method:

- Which processes should be evaluated by PAPE?
- Which data elements (e.g. sale) in the audit trail define a business benefit? If data values (sale) are not nominal then which range of values belongs to different business benefit classes (e.g. GOOD, MODERATE)?
- Which business benefit class is interesting (e.g. GOOD)?

Select Target Process: In order to find out which process is suitable for the PAPE approach, the following questions have to be answered:

- Does a process have available input data to get different data patterns? $->$ Only if data patterns can be identified such a process qualifies for PAPE.
- Do agents have different expertise or performance impact on business benefits for a process? $->$ Only if different expertise profiles are predominant agent selection qualifies for PAPE.

Domain expert can answer these questions. The ***Create Design*** process has input data (kind of design) that form data patterns and the business benefit is depending on the performance of executing agents as statistics from Table 1 reveals. Domain expert decide that ***Create Design*** is a target process suitable for PAPE.

Select Process Relevant Data: Data required for PAPE consist of process structure data, process execution data and business benefit data. Process structure data consist of process name, agent role and executing agent information. Process execution data consist of different data elements (e.g. design features) and is input to process before execution starts. Business benefit data is either part of execution data or is available

somewhere in the audit trail and is used for the evaluation of process success as a whole and agent performance in detail.

For the selected process domain experts select the process structure data, process execution data and business benefit data. It is application dependent how business benefits are defined. Business benefit data may be the part of application data or process execution data. This business benefit data should be integrated and consistent with process execution data. Domain experts extract and integrate business benefit data with process execution data and process structure data. For instance in ***Create Design*** process, business benefit data (e.g. sale) is the part of marketing department application data source. The relation between a process and its business benefit cannot be inferred automatically. Therefore use of domain knowledge is necessary. Domain experts prepare this dataset consisting of process structure data, process execution data and business benefit data.

Prepare Dataset: The last step of the Preprocessing Phase is to prepare the dataset which is acceptable for next phase. This phase applies machine learning technique to the input dataset to generate an integrated data structure. Data suitable for machine learning technique needs to be in nominal or numeric form. Process structure data is already in nominal form (e.g. Agent, Role, Process Name) but some data elements of the process execution data may not fulfill this requirement so need to be transformed into nominal form or ignored (e.g. design name). In general business benefit data is already in the nominal form.

If the business benefit data is not in discrete or nominal form then domain experts transform this business benefit data into discrete classes like GOOD, MODERATE. Discrete classes are defined by the domain expert as required in application context like EXCELLENT, GOOD, AVERAGE and POOR. When the dataset is preprocessed, the last step required is to define interesting business benefit class. Domain experts define when dataset is loaded into PAPE method and used in phases Pattern Based Agent Performance Evaluation and Agent Assignment Learning.

3.2 Integrated Data Structure Generation Phase

The PAPE method generates integrated data structure consisting of a decision tree and agent intelligence matrices using the dataset prepared in the preprocessing phase. It is used to identify the relationship between data patterns, agent assignments and business benefits. To generate an integrated data structure, first a decision tree is generated from process execution data and business benefit data using machine learning technique. Then post-processed is performed to extend the decision tree with agent intelligence matrices (one matrix for each branch of the tree) using process execution data, agent data and business benefit data.

Generate Decision Tree: A decision tree is generated using the J48 classification algorithm selected from Weka library [13]. It is a slight modification of the C4.5 decision tree [9]. It considers all the possible tests that can split the dataset and select a test set that gives the best information gain. It generates a decision tree for the given dataset by recursive partitioning and can handle nominal or numeric values.

The J48 algorithm is applied on process execution data (e.g. design features) and business benefit data (as classifying attribute). The algorithm generates a decision tree that consists of branches and leaves (Figure 3(A)). Each branch represents a data pattern

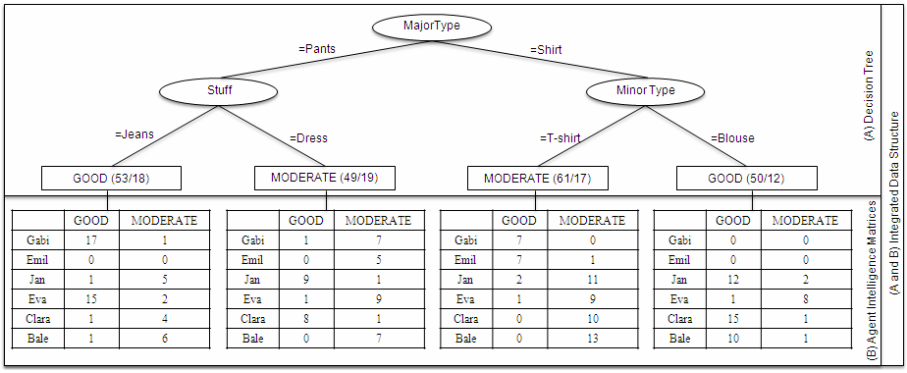
described by process execution data (e.g. MajorType="Shirt" and MinorType="Blouse"); the leaves represent the predicted business benefit class (e.g. GOOD) and accuracy of prediction (e.g. 50/12: which means out of 50 design instances 38 are GOOD and 12 are not GOOD).

Figure 3(A) displays the decision tree generated from the dataset of the ***Create Design*** process which is actually of large size; however, to demonstrate our method more precisely we have selected only a partial tree. Now, it typically constitutes 4 data patterns learned by 213 designs instances created by 6 agents of the ***Create Design*** process.

It needs to be said that J48 algorithm may generate a decision tree consists of single node (no branches) then PAPE method is not applicable for such processes because if no data patterns are found then relationship between data pattern, agent assignment and business benefit cannot be determined.

Post-processing: If the decision tree consists of different branches (data patterns) then post-processing is performed to extend each branch of the decision tree with an agent intelligence matrix. An agent intelligence matrix (Figure 3(B)) is a two dimension array whose number of rows and columns are determine by number of agents and business benefit classes in the dataset respectively and each particular row-column entry can be identified by agent and business benefit data.

To extend each branch of the decision tree with agent intelligence matrix the post-processing scenario is described: For each instance of the dataset, process execution data is used to parse the decision tree by traversing a particular branch and find its agent intelligence matrix. If the agent intelligence matrix already exists then only its row-column entry is incremented otherwise an intelligence matrix is extended with the branch and then the row-column entry is incremented. Which row-column entry of the agent intelligence matrix needs to be incremented? It is determined by the agent and business benefit data value of the instance being processed. When all instances of the dataset are processed an integrated data structure is generated as shown in Figure 3.



	GOOD	MODERATE
Gabi	17	1
Emil	0	0
Jan	1	5
Eva	15	2
Clara	1	4
Bale	1	6

	GOOD	MODERATE
Gabi	1	7
Emil	0	5
Jan	9	1
Eva	1	9
Clara	8	1
Bale	0	7

	GOOD	MODERATE
Gabi	7	0
Emil	7	1
Jan	2	11
Eva	1	9
Clara	0	10
Bale	0	13

	GOOD	MODERATE
Gabi	0	0
Emil	0	0
Jan	12	2
Eva	1	8
Clara	15	1
Bale	10	1

Fig. 3. Integrated data structure

3.3 Pattern Based Agent Performance Evaluation

Pattern based agent performance evaluation is performed using the integrated data structure. The decision tree can be used to evaluate dependencies between data pattern and business benefit (Figure 3(A)); the decision tree extended with agent intelligence matrices (Figure 3(A and B)) can be used to evaluate dependency between data pattern, agent assignment and business benefit. Hence, from the integrated data structure two types of rules can be learned: the first one relates the data pattern with a business benefit class (from root to leaf of the tree Figure 3(A)); the second one relates data pattern, agent (assignment) with a business benefit class (from root to particular row-column entry in agent intelligence matrix Figure 3(A and B)). For instance consider the two rules from rightmost branch of the integrated data structure:

- (Major Type= "Shirt" and Minor Type="Blouse") $->$ Sale=GOOD [Accuracy 76%]
- (Major Type= "Shirt" and Minor Type="Blouse" and agent="Clara") $->$ Sale=GOOD [Accuracy 94%]

The improved accuracy of the second rule in comparison to the first rule reveals that the agent (e.g. Clara) has expertise for the data pattern: there is a dependency between data pattern, agent assignment and business benefit. This agent expertise can be utilized for future agent assignment.

If on the other hand accuracy of the first rule is greater than that for the second rule then agent has no significant expertise for this data pattern. So there is no dependency between data pattern, agent assignment and business benefit. This is shown in the following two rules:

- Major Type= "Shirt" and Minor Type="Blouse") $->$ Sale=GOOD [Accuracy 76%]
- (Major Type= "Shirt" and Minor Type="Blouse" and agent="Eva") $->$ Sale=GOOD [Accuracy 11%]

This agent expertise cannot be utilized for future agent assignment for achieving business benefit. It needs to be mentioned that only an interesting business benefit class (e.g. GOOD) is used for rules learning and evaluation because we are interested in utilizing agent expertise to promote business for GOOD sale. This interesting business benefit class is mentioned by domain expert when dataset is loaded into the PAPE method (Section 3.1).

This phase finds those data patterns where agents have significant expertise in their attached agent intelligence matrix and forward this list of data patterns to next phase for learning agent assignment rules.

3.4 Agent Assignment Learning Phase

For each pattern where a relationship between data pattern, agent assignment and business benefit is observed this phase generates agent assignment rules. A list of agent assignment rules whose accuracy is significantly (at least 5%) higher than pattern accuracy are learned from the respective agent intelligence matrix. For the integrated data

structure in Figure 3 the following patterns and respective agent assignment rules can be learned:

(MajorType="Shirt"and MinorType="Blouse")[Accuracy=76%]

- IF (Agent="Clara") ? (Sale=""GOOD") [Accuracy = 94%]
- IF (Agent="Bale") ? (Sale="GOOD") [Accuracy = 91%]
- IF (Agent="Jan") ? (Sale="GOOD") [Accuracy = 86%]

(MajorType="Shirt" and MinorType="T-shirt")[Accuracy=28%]

- IF (Agent="Gabi") ? (Sale="GOOD") [Accuracy = 100%]
- IF (Agent="Emil") ? (Sale="GOOD") [Accuracy = 88%]

(MajorType="Pants" and Stuff = "Dress")[Accuracy=39%]

- IF (Agent="Jan") ? (Sale="GOOD") [Accuracy = 90%]
- IF (Agent="Clara") ? (Sale="GOOD") [Accuracy = 89%]

(MajorType="Pants" and Stuff = "Jeans") [Accuracy=66%]

- IF (Agent="Gabi") ? (Sale="GOOD") [Accuracy = 94%]
- IF (Agent="Eva") ? (Sale="GOOD") [Accuracy = 88%]

Agent Assignment rules are learned and are presented to domain experts for refinement and are finally forwarded to the Agent Assignment Updating Phase.

3.5 Agent Assignment Updating Phase

Agent assignment rules finally refined by domain experts need to be deployed in the organizational model of the WFMS. This phase transforms assignment rules into XML form which can be deployed in a WFMS. For each data pattern and its expert agents, a role is created and agents are defined for each role. For instance consider the following notions for agent assignment rules learned in the previous sub-section:

- Role JeansPantsDesigner [] = ["Gabi", "Eva"];
- Role DressPantsDesigner [] = ["Jan", "Clara"];
- Role T-shirtDesigner [] = ["Gabi", "Emil"];
- Role BlouseDesigner [] = ["Clara", "Bale", "Jan"];

Now many roles are defined to execute a process. How can a particular role be selected for a particular data pattern? In a WFMS input data is made available before process execution starts. Based on the features of input data (design features) an appropriate role is selected using the following notions:

- IF(Design.MajorType="Pants" and Design.Stuff="Jeans") THEN Performer.Role() = JeansPantsDesigner;
- IF(Design.MajorType="Pants" and Design.Stuff="Dress") THEN Performer.Role() = DressPantsDesigner;
- IF(Design.MajorType="Shirt" and Design.MinorType="T-shirt") THEN Performer.Role() = T-shirtDesigner;
- IF(Design.MajorType="Shirt" and Design.MinorType="Blouse") THEN Performer.Role() = BlouseDesigner;

4 Experimental Results

To investigate the usefulness of our PAPE method experiments we performed on the datasets of two different garment factories; we call them Factory-A and Factory-B. The companies must be kept anonymous. In a collaborative effort of domain experts and workflow designers 26 processes were selected from factory-A and 33 processes from factory-B. Processes which have input data to form data patterns were initially selected and datasets were prepared by integrating process execution data, processes structure data and business benefit data. Due to non-disclosure we cannot provide all details of the dataset in this paper but we report the evaluation of our results.

We are using two graphs to represent these prediction accuracies: the BB-Without-PAPE graph represents prediction accuracy without pattern based agent performance evaluation (data pattern $->$ business benefit) and the BB-With-PAPE graph represents prediction accuracy with pattern based agent performance evaluation (data pattern, agent $->$ business benefit).

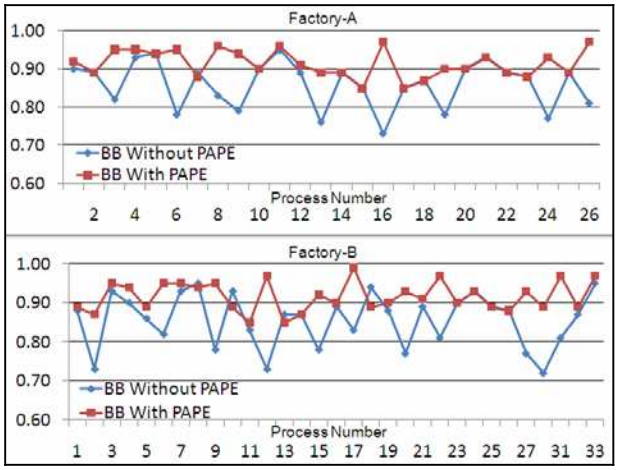


Fig. 4. Measuring the usefulness of the PAPE method

Figure 4 shows the results of our evaluation using the BB-Without-PAPE and BB-With-PAPE graphs. As shown in Figure 4 (upper part: Factory-A) there is a significant difference between prediction accuracies of BB-Without-PAPE and BB-With-PAPE graphs for 9 processes; this means that there is a dependency between agent assignment and business benefit. Whereas the prediction accuracies are almost the same for 17 processes; this means that these processes were executed equally well by all agents and no dependency between agent assignment and business benefit were observed. Similarly for factory-B such dependency is observed in 12 out of 33 processes; 21 processes were executed almost equally well (Figure 4 lower part).

Although pattern based agent performance evaluation is applicable to a comparatively small number of processes it is very valuable for achieving the business benefit

objective. We are currently working on the enactment of the PAPE method in these two factories. This means that we aim at the substitution of the current agent assignment rules with the rules learned by our method.

5 Related Work

Process mining techniques are used to support redesign and diagnosis the processes and are proven to be a valuable tool to gain insight into how business processes are being handled within an enterprise [11]. An implicit assumption of process improvement is to benefits the business [3]. But most process mining techniques re-construct a process model from the data recorded in an event log to reflect the behavior of process execution (discovery) [2] [8]. Then this observed behavior is compared with the original process model to detect possible deviations (conformance checking) [8],[1]. Most of the process mining research focuses on the functional and control flow perspective. Relatively small portion of research is found on the organizational perspective.

In [12],[8] van der Aalst et al. constructed the organizational models. These models represent the structure of the organization and the relationships between organizational structures and organizational population: who performs what and how performers are related. In [8], they developed the methods for mining social networks from process logs to analyze relationships between agents involved in the processes. They presented different matrices like handover of work matrices, in-between matrices and working together matrices to express potential relationships between agents. Also, in [12] they developed methods for mining organizational models and analyze relationships among organizational entities and units.

The work in [14], [7], [10] is related to agent assignment. In [7], Ly et al. focused on mining the agent assignments rules from process log using machine learning technique. He combined the organizational model with the audit trail data and learned agent assignment rules. These rules define the profiles of agents capable or eligible of performing an activity but not the profile of agents who execute the process in a way that direct to organizational business benefit.

Similarly in [14], Liu Yingbo et al. applied machine learning classifier to workflow event log to learn various kinds of activities each agent undertakes. When a new process is initiated this machine learning classifier suggests a suitable agent who can execute the particular process. But his objective was to reduce the burden of staff assigner for conventional agent assignment rather to automatic agent assignment in WFMS. Moreover the learned classifier predicts only one agent who might be on leave or busy with some other process.

Similarly in [10] S. Rinderle-Ma et al. proposed agent assignment rules mining technique. They compared the mined assignment rules with the pre-defined agent assignment rules in order to detect possible deviations. These deviations detect the security loopholes like offering the process to non authorized agent. The main contribution of this work is to first identify agents working successfully and feedback this information into WFMS to automate future agent assignments.

6 Conclusion and Future Work

In this paper we discussed a method for mining agent assignment rules based on agent performance. Agent performance is evaluated to determine how agents are working successfully for achieving business benefit and this information is feed backed for future agent assignment. Our method evaluate agent performance more precisely using machine learning technique and post-processing technique and suggests agent assignment rules which leads to the success of the performance of a business process as a whole and the agent assignment in detail.

We believe that our method shows some promise for improving the current state of workflow agent assignment strategies. Our future plans include further experiments with substituted agent assignment rules. An empirical study will be performed as an evaluation of adequacy of our PAPE method with the enactment of modified agent assignment rules in these factories. We also investigate to learn agent assignment rules for processes which have input data without having adequate data patterns.

References

1. W. M. P. v. d. Aalst. Business alignment: using process mining as a tool for delta analysis and conformance testing. *Requir. Eng.*, 10(3):198–211, 2005.
2. W. M. P. v. d. Aalst and B. F. v. Dongen. Discovering workflow performance models from timed logs. In *EDCIS '02: Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems*, pages 45–63, London, UK, 2002. Springer-Verlag.
3. P. L. Bannerman. Capturing business benefits from process improvement: four fallacies and what to do about them. In *BiPi '08: Proceedings of the 1st international workshop on Business impact of process improvements*, pages 1–8, New York, NY, USA, 2008. ACM.
4. C. Bussler. Organisationsverwaltung in workflow-management-systemen. 1997.
5. L. Cao and C. Zhang. Domain-driven, actionable knowledge discovery. *Intelligent Systems, IEEE*, 22(4):78–88, c3, July-Aug. 2007.
6. Jablonski. *Workflow Management: Modeling Concepts, Architecture and Implementation*. 1996.
7. L. T. Ly, S. Rinderle, P. Dadam, and M. Reichert. *Business Process Management Workshops*, chapter Mining Staff Assignment Rules from Event-Based Data, pages 177–190. Springer-Verlag, London, UK, 2006.
8. A. K. Medeiros, A. J. Weijters, and W. M. Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, 2007.
9. J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
10. S. Rinderle-Ma and W. M. P. v. d. Aalst. Life-cycle support for staff assignment rules in process-aware information systems. Beta working paper series, Eindhoven University of Technology, 2007.
11. A. Rozinat and W. M. P. v. d. Aalst. Decision mining in business processes. Beta working paper series, Eindhoven University of Technology, 2006.
12. M. Song and W. M. P. van der Aalst. Towards comprehensive support for organizational mining. *Decis. Support Syst.*, 46(1):300–317, 2008.
13. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

14. L. Yingbo, W. Jianmin, and S. Jiaguang. A machine learning approach to semi-automating workflow staff assignment. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 340–345, New York, NY, USA, 2007. ACM.

Improving agent bidding in Power Stock Markets through a data mining enhanced agent platform

Anthony C. Chrysopoulos, Andreas L. Symeonidis, and Pericles A. Mitkas

Department of Electrical & Computer Engineering
Aristotle University of Thessaloniki,
GR-54 124, Thessaloniki, Greece
anchrys@ee.auth.gr, asymeon@eng.auth.gr, mitkas@eng.auth.gr

Abstract. Like in any other auctioning environment, entities participating in Power Stock Markets have to compete against other in order to maximize own revenue. Towards the satisfaction of their goal, these entities (agents - human or software ones) may adopt different types of strategies - from na?ve to extremely complex ones - in order to identify the most profitable goods compilation, the appropriate price to buy or sell etc, always under time pressure and auction environment constraints. Decisions become even more difficult to make in case one takes the vast volumes of historical data available into account: goods' prices, market fluctuations, bidding habits and buying opportunities. Within the context of this paper we present Cassandra, a multi-agent platform that exploits data mining, in order to extract efficient models for predicting Power Settlement prices and Power Load values in typical Day-ahead Power markets. The functionality of Cassandra is discussed, while focus is given on the bidding mechanism of Cassandra's agents, and the way data mining analysis is performed in order to generate the optimal forecasting models. Cassandra has been tested in a real-world scenario, with data derived from the Greek Energy Stock market.

Keywords: Software Agents, Data Mining, Energy Stock Markets, Regression Methods.

1 Introduction

Agent technology has already proven its potential in various aspects of real-world trading and electronic markets. In complex environments, such as (Power) Stock Markets, where bipartite relationships between involved actors demand negotiation in order to come to an agreement, the utilization of autonomous and intelligent agents has become imminent [7] [16]. Numerous approaches have been employed in order to develop the optimal agent strategy with respect to the challenges the agents face. Among all issues rising when designing and developing systems for such highly dynamic markets, the extreme rate that data is generated at is considered of high importance. Additionally, like in all auction environments, decisions have to be made under extreme time pressure, making it difficult to apply simple algorithms and/or analytic strategies. These factors indicate that Data Mining (DM) could be a suitable technology for achieving an "intelligent" and efficient software solution.

Within the context of our work we provide a flexible, robust and powerful tool for dealing with all the issues related to the hyperactive and continuously changing Power Stock Market. We have built a multi-agent system (MAS) capable of efficiently handling the deluge of available data and of practicing various DM methodologies, in order to predict the prices of goods of interest. Our platform was benchmarked on data provided by the Greek Power Stock Market, which is a dynamic, partially observable environment. This environment allows for different strategic approaches to be followed, while it is highly challenging, due to the fact that each decision made affects instantly the future moves or decisions of the platform and the Stock Market itself.

Looking at the bigger picture, one may argue that an agent developed can employ DM, in order to extract useful nuggets of knowledge that could give him/her a predictive advantage over other competitors. In this context, we have applied DM in order to: a) analyze the historical data from the past auctions in order to predict the future values in goods of interest and, b) induce market behavior models and incorporate them into or agents, in order to provide them with a predictive edge over the competition.

The rest of the paper is organized as follows: Section 2 provides an overview of the Power Stock Market mechanisms (Auctions and Energy Market) available, as well as a state-of-the-art analysis. Section 3 presents the platform developed for monitoring and participating in the Power Stock market, and briefly discusses its architecture. Section 4 describes in detail the DM methodology applied, in order to decide on the optimal forecasting bid model, while Section 5 provides a pilot case scenario, aiming to illustrate the functionality of our platform. Finally, Section 6 summarizes work conducted and concludes the paper.

2 Power Markets

Up until recently, in most EU countries (Greece included), power supply was a physical monopoly, thus the establishment of a state or state-controlled administration department that would be responsible for producing, transferring and distributing, was justified. However, the advancement to more loose economic competition models has signified the cease of this physical monopoly, as far as the production and the distribution of energy are concerned. In turn, the liberation of the Power Market has given room for the development of Open Markets, where participants are able to choose between different energy products in different periods of time and may negotiate on their "folder of products". These folders can be negotiated under three different schemes:

- **The Long-term Market**, where participants come to direct agreements in form of long-term contracts.
- **The Day-ahead Market**, where buyers place their bids in 24 hourly auctions, in order to establish a contract for the next day.
- **The Real-time Market**, where buyers place their bids in order to establish a contract for the next hour.

Due to the physical model of the energy production-distribution-consumption network, long-term and day-ahead markets are the most dominant ones. Nevertheless, the

inability to store Power for long periods of time, dictates the development of a mechanism that can efficiently balance the supply and demand of Power and can be easily and constantly controlled. The administrator of each Power system is responsible for ensuring this balance between production and consumption, through the utilization of a Real-Time Market. The Real-Time Market model bears the highest risk for the participants, since the malfunction of an Electric node or the shutting down of a Power line can bring exaggerated rising or falling of the prices. Nevertheless, its existence is decisive for the coverage of Demand, in case the other two Markets do not provide enough Power. Additionally, it should be stated that, though the Real-time Market entails higher risk, it also provides greater profit opportunities.

2.1 Power Market Auctions

An *auction* is defined as a strict set of rules for the specification of the exchange conditions of goods [23]. In each auction, a number of transactions are performed between the participants, where each transaction comprises two elements: a) a protocol that defines the rules of the transaction mechanism as well as the actions allowed to an (human or software) agent participating in an auction and, b) a strategy, i.e. the methodology followed by an agent in order to fulfill its goal. The protocol of an auction is determined during its design and is announced to all the participants from the beginning. Agents' strategy is designed by each participant and is unknown to the rest.

In *Power Market Auctions*, two are the most important entities:

1. The Market participants (or Players)
2. The Independent System Administrator (ISA)

A *Player* is defined as any economical entity that accesses the Power Market [18]. In general, this entity may possess a group of Production Units or/and a group of Consumers. Each Player participating in the Power Market as a *Producer* should submit his/her power supply offers in pre-specified time intervals. Each offer contains the amount of supplying Power, as well as the minimum price he/she is willing to accept. One the other hand, each Player that participates in the Power Market as a *Consumer* should submit his/her power supply demands - within the same time intervals - along with the maximum price he/she is willing to pay for it.

The *ISA* is the administrator of the Power Transfer System, and also the Administrator of the Power Stock Market. Thus, ISA is responsible for the Settlement of the Power Market, taking into consideration the transferring limitations of the system. ISA collects bids for each hourly auction and has to calculate two curves: the aggregate ascending Supply Curve and the aggregate descending Demand Curve. In the simplified case where no transfer limitations are presented, the Settlement of the Market is the intersections of the two curves (Figure 1). This point determines the *Settlement Price of the Market* (SPM) - this is the price that the Producers are paid by the Consumers, the load production of each Production Unit and the consumption of each Consumer.

In case transfer limitations exist, ISA must follow a more complicated process. He/she then has to solve an optimization problem targeting to the maximization of the social prosperity (target function). Then the Power price, the Load production of each Production Unit and the consumption of each Consumer are calculated.

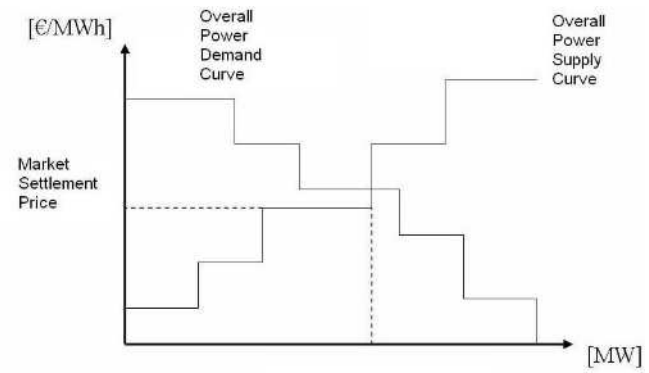


Fig. 1. The Aggregate Power Supply And Demand Curves

2.2 State-of-the-art

Various approaches have been employed for analyzing the behavior of Power Markets, some of which have adopted Agent Technology (AT) and DM primitives. In fact, results reported seem quite promising.

In the early stages, the implementations designed for Energy Markets were merely simulation environments for market regulations and did not exploit the advantages provided by the utilization of DM methodology. One of the first systems developed was MASCEM (Multi-Agent Simulator for Competitive Electricity Markets) [12], [13] [14], aiming to validate regulations and behaviors within the Electricity Markets, using only na?ve reinforcement learning strategies. Another agent platform was later on developed, to test the reliability of the proposals FERC of USA (Federal Energy Regulatory Commission) applied in the Standard Market Design (SDM) [10], [9]. In this implementation, the Producers (Enterprises or Persons), the Consumers and the ISA were modeled by the use of agents.

In the early 2000s , during the boost of MAS utilization, the Electric Power research Institute (ERPI) developed SEPIA (Simulator for Electrical Power Industry Agents), a multi-agent platform capable of running a plethora of computing experiments for many different market scenarios [2], [1]. SEPIA employs two different options for agent learning: a variation of the Q-Learning Algorithm [21], which corresponds each noticeable state to a suitable action, and an LCS (Learning Classifier System) morph [8], which utilizes a rule-based model and agents learn through amplified learning and genetic algorithms.

The Argonne National Laboratory, on the other hand, developed EMCAS (Electricity Market Complex Adaptive System) [5], an efficient implementation for handling the Electric Energy Market. Through EMCAS, one may study the complex interactions between the physical entities of the market, in order to analyze the participants and their strategies. Players' learning is based on genetic algorithms, while EMCAS supports stock market transactions, as well as bipartite contracts.

As far as the real-market analysis is concerned, Bagnall [3] has presented a simplified simulation model of Great Britain's Electricity Market, were the Producers are

agents that participate in a series of auctions-games. In any phase of the process, a Producer is faced with two options: a) try an already tested and applicable strategy that will reassure he/she will not lose money or, b) find a new strategy rule that will maximize his/her profits. To this end, each agent is modeled with LCS learning abilities and its behavior is monitored. Changes in the behavior are caused by the transition from a uniform pricing system to a system where each producer is paid at his/her supply price. Additionally, the possibility of cooperation between two agents when the rest Producers make offers in their cost limit is also observed.

Finally, Petrov and Sheble [11] introduced Genetic Programming in their simulation and tried to model the bipartite Auctions for the Electric Energy Market, by the use of agents. One of the players incorporates knowledge represented as a Decision-Making Tree, which is developed by Genetic Programming. The rest of the agent-players incorporate ruled-defined behaviors.

At this point, the work of Rosenschein and Zlotkin should be pointed out. During the early 90s, they laid the foundations of the Multi-Agent Negotiation Systems and set the relatively important attributes that someone should take into consideration [15]. These attributes are: a) Efficiency (with respect to Pareto and Global Optimality), b) Stability, c) Simplicity (as low computational demands as possible, thus increasing stability and efficiency), d) Distribution and e) Symmetry. All the abovementioned implementations are solutions to specific problems and most of the times they utilize a pre-specified DM technique. Within the context of our work, the developed system is greatly versatile. New types of prediction models can be easily created and substitute the previously used ones, while even the type of problem can be modified (time windows used for prediction can change - day, week, month, year).

3 Developed System

`Cassandra` is a multi-agent platform designed and developed to function in an automatic and semi-autonomous manner in Energy Markets. `Cassandra` employs DM techniques in order to forecast the Settlement Prices, as well as the power load of the Day-Ahead Market. The implemented system may even function in a fully autonomous manner if granted permission, and may proceed with the necessary actions for the establishing a contract. The efficiency of the system is highly dependent on the models generated from historical data, as well as a set of the fail-safe rule base specified by the Power Market expert. Through `Cassandra`'s interface, DM models are re-built dynamically, in order to study, evaluate and compare the results and choose the one that optimally projects running market trends.

3.1 `Cassandra` Architecture

`Cassandra` follows the IRF Architecture Model (Intelligent Recommendation Framework) [17], which defines a 4-layer functional model for the agent system. IRF is usually employed in enterprises for the optimization of the administration mechanism, since it can automate and integrate all the data producing or data demanding facets of a company.

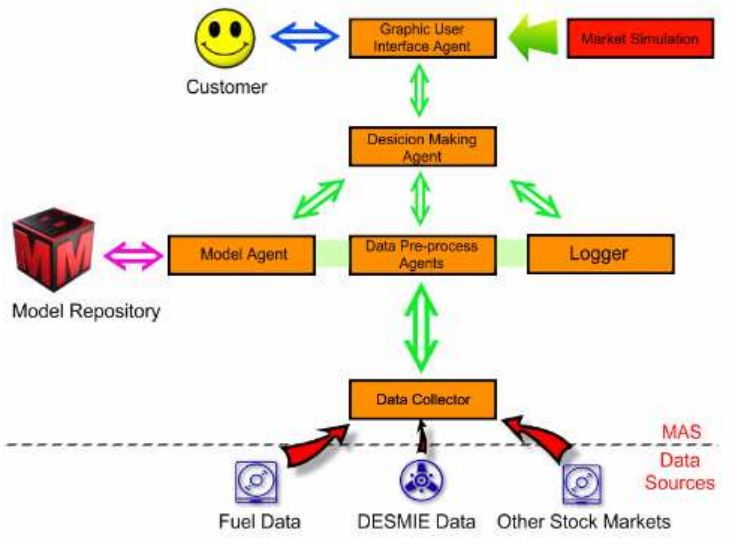


Fig. 2. The `Cassandra`'s 4-layer Architecture

Taking a closer look of the Power Market through the IRF prism, one may identify several tasks that have to be tackled:

- Collection of the historical data from previous auctions and processing.
- Application of the suitable DM algorithms in order to build the necessary forecasting models.
- Integration of generated models in the Business Intelligence of the System and evaluation of the results.
- Continuous monitoring Stock Market.

As expected, `Cassandra` employs a modular architecture (Figure 2), where each module is responsible for one of the aforementioned tasks. The platform also provides a wrapper around all modules and ensures communication with the system users. The modules comprising `Cassandra` are:

i. **Data Collection Module (DCM)**: It is responsible for the collection of historical data, either from files provided by the user, or directly from the Internet.
ii. **Data Processing and Mining Module (DPMM)**: One of the core modules of `Cassandra`, which is responsible for the processing of the data, preparing the training sets and applying the DM algorithms.
iii. **Decision Making Module (DMM)**: It aggregates all information in order to make the optimal decision in any given occasion.
iv. **Graphic User Interface Module (GUIM)**: It interacts with the users of the System. It must be user-friendly, and easily comprehensive.

3.2 Cassandra users

Cassandra identifies three types of users:

– System Analyst
The System Analyst (SA) is an expert user. He/she is the creator of the system, so he/she has absolute knowledge over it. SA is responsible for the smooth operation (unimpeded operation of the system, satisfaction of the software requirements of the users), as well as its efficiency. SA also provides domain knowledge, while he/she is responsible for generating the DM models, in order to decide on the best-performing one(s). Finally, SA checks the decision routines, in order to trace in time any errors that may occur.

Fig. 3. An overview of the Cassandra MU (a) and SM (b) View

– System Manager
The System Manager (SM) is also an expert user, not on agents and DM, but in the Power Stock Market. SM cannot interfere directly with the system, but using his/her experience in the Market SM can easily evaluate the strategic moves of the system and decide whether Cassandra operates efficiently enough or not. SM has the authority to

Table 1. Cassandra Functionality

User Type	Views	Privileges	Functionality
MU	– Monitoring	None	– Account Checking – Market Monitoring – www/RSS Browsing – Chart Viewing
SM	– Monitoring – Manager	Managing only	All the above plus: – Manual Bidding – Logger Handling
SA	– Monitoring – Manager – Analyst	All	All the above plus: – Agent Overview – Model Retraining – Model Configuration – Model Statistics – Cost Evaluator

manually override the prediction system (for example change the bids on the Settlement Price), but he/she cannot change the models used by the system for prediction (that is under the Analyst's jurisdiction).

– Monitoring User
The Monitoring User (MU) is the na?ve user of the system. MU only monitors the Power Market moves and the logs the decisions - Cassandra's placed bids. In case MU notices something 'out of the ordinary' (actions that do not have the expected results), MU notifies the SM. SM must then double check MU's observations and, in case of error, notify the SA, who will react accordingly in order to optimize system operation.

Each user group is accommodated through different views of the system, enabled upon user authentication. Table 1 summarizes the functionality provided to each user group.

3.3 Implementation

Cassandra provides a multi-functional user interface to facilitate its usage. It has been implemented in Java 1.6 and all agents are developed over the Java Agent Development Framework (JADE) [4], which conforms to the FIPA specifications [19]. The system provides authorized users (SAs) full control over the agents, from their creation till their termination. All DM models are built and evaluated on the WEKA (Waikato Environment for Knowledge Analysis) API [22].

As already denoted, Cassandra supports MUs, SMs, and SAs through respective views. When initialized, Cassandra projects the MU View (Figure 3a). The user has

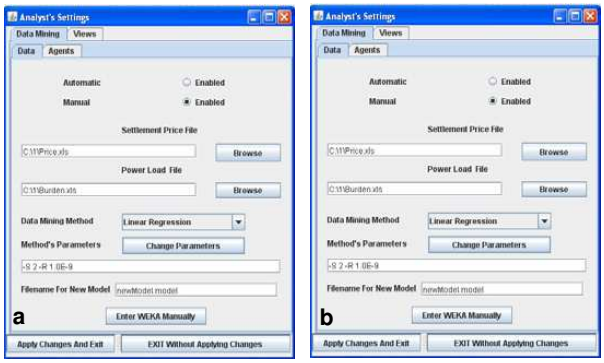


Fig. 4. The Agent and Data Mining Configuration panes

then the option to log in as an SM or an SA (through the 'Settings' menu) and enjoy the privileges of each user category. The SM View (Figure 33b) provides many additional features in comparison to the MU View. It realizes the Logger, a mechanism that provides a detailed overview of the bidding agent's predictions and actions, in order to monitor and evaluate the efficiency of the system. It also provides an override mechanism for manually bypassing the automated bidding process, in case SM considers the prediction to be faulty. On the other hand, the SA View comprises several useful design and development tools. Such tools are the Cost Evaluator, which calculates the cost of Production given the right Parameters, as well as a Agent and a Data Mining Configuration pane (Figure 4a and 4b, respectively), that provide full control on the MAS architecture (change the types and number of agents residing in the Processing and DM Layer, select different preprocessing agent behaviors) and the creation and evaluation of DM models (select new training and testing datasets, new algorithms/algorithm parameters etc), respectively.

4 Preliminary Results

Before building the `Cassandra` system, we performed a thorough analysis on available data, in order to build DM models that would efficiently predict the settlement price and the power load in a Day-ahead market. Both problems were modeled as regression problems, where the desired output would be the Decision Manager's prediction of the Settlement Price or Power Load with respect to past auction data. Various experiments and models were built, taking hourly/weekly/monthly periodicity into account. For the shake of simplicity, we provide the results on the models created based on a daily time window (the prior 23 hours-auctions are considered as input, requesting to predict the Settlement Prices and Power Loads for the 24th hour). Nevertheless, several other types of problems can be faced through `Cassandra`. Different time windows (taking into consideration the last week's, month's or year's values), as well as combinations of time windows and daily prediction models. Additional sources of data can be considered (fuel prices, other Stock Markets etc).

The WEKA suite was employed for the conduction of the experiments with a plethora of algorithms over the available datasets.

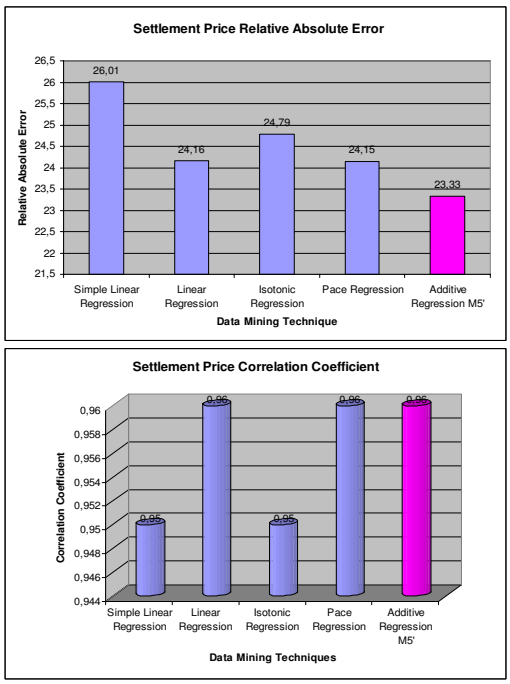


Fig. 5. A comparison of the regression schemes applied on the Settlement price

4.1 Training

Five different classification (regression) schemes were applied, in order to decide on the one that optimally meets the problem of predicting the Settlement Price and the Power Load for the 24th hour: a) Simple Linear Regression, b) Linear Regression, c) Isotonic Regression, d) Pace Regression, and e) Additive Regression [6]. The Simple Linear Regression models applied, as expected, gave poor results on every dataset it was applied on, since it is known to a very provincial technique. The correlation coefficiency (cc) of the model extracted was around 0.95 and 0.98, while the Relative Absolute Error (RAE) was around 26% and 22% for the Settlement Price and the Power Load, respectively. A large number of experiments were conducted applying Linear Regression, on different datasets (varying size) and with different algorithm parameters. The cc was a slightly better (0.96 and 0.98-1), as well as the RAE for the Settlement Price (24%). The RAE for the Power Load (10%), though, improved significantly. With the application of Isotonic Regression techniques, the results were as disappointing as in the case of Simple Linear Regression, only with a slight improvement in RAE for the Settlement

Price (24%). The Pace Regression algorithm (numerous parameters for algorithm fine-tuning), came up with very good results both for cc (0.96, 1) and for RAE (24%, 10%) for both goods. It's should be mentioned that increasing or decreasing the Estimator's parameters didn't significantly influence the resulting efficiency, while in some cases the Mean Squared Error was much higher that the simple techniques used, indicating overfitting.

4.2 Meta Classification

Apart from the regression schemas applied, we also tested some meta-classifier schemas, striving for optimal performance. Various Additive Regression schemes were tested against the same datasets, in order to ensure equally compared test results.

Three schemes applied to the dataset: a) *DecisionStamp* (building and using decision stamp), *REPTree* (fast decision tree learner) and *M5'* classifiers. The first performed worse than any other model extracted (*cc* 0.93 and 0.96, *RAE* 33% and 31%, respectively). Nevertheless, the *REPTree* schema came up with much better results (*cc* was 0.95 and 0.99, while *RAE* improved to 23.21% and 14.93%, respectively). Finally, *M5'* outperformed all schemes and algorithms, with cc reaching the maximum value of 0.96 and 1, while in the same time *RAE* was around 23% and 9%, respectively.

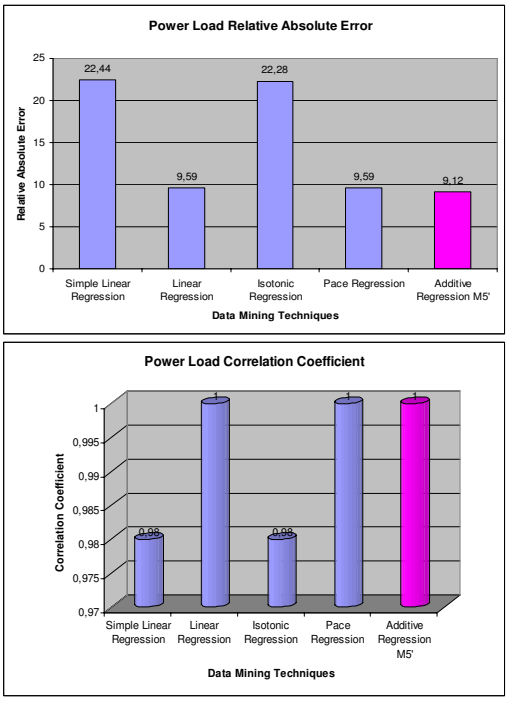


Fig. 6. A comparison of the regression schemes applied on the Settlement price

4.3 Data Mining Experiment Results

It's obvious that the combination of the Additive Regression meta-classification schema with the M5' regression algorithm, applied on the given datasets significantly outperforms all the other learning methods applied. The main advantage of M5' with respect to the other methods applied is that it produced a simple and compact tree model, in contrast with Simple and plain Linear Regression that attempted to impose a linear relationship on the data, Isotonic Regression that picks the attribute that results the lowest square error, and Pace Regression that is optimal when the number of coefficients of the linear model tends to infinity. Combined with Additive Regression, which introduces a stochastic factor, *M5'* succeeded in improving the accuracy of the predictors and achieved optimal performance. Figures 5 and 6 provide a qualitative comparison between the regression schemes applied, cross-validated on the same dataset.

5 Pilot Case Scenario

In order to demonstrate the proper functioning of `Cassandra`, we have set up a real-life scenario, based on historical data for the Greek Energy Stock Market (avail: http://www.desmie.gr/).

Within this Day-ahead market, Producers place each day 24-bid bundles for the forthcoming day's 24-hourly auctions, while Consumers declare their needs in Power Supply, along with the maximum price willing to pay. The ISA takes all this information in consideration and calculates the aggregate Supply and Demand Curves, thus defining the Settlement Price and Power Load for each one of the forthcoming day's auctions. All Producers that have made bids, less or equal than the specified Settlement Price are included in the next day's distribution network, and are paid at the Settlement Price for each MWh sold. The rest of the Producers are not included in the transaction.

The scenario `Cassandra` is tested against is the following: First, we simulate a power stock market by randomly select 5 days from the historical data on previous auctions. For each day, `Cassandra`'s agents try to predict the 24th-hour auction Settlement Price and Power Load, based on the values of the other 23-hour auctions. Then the predicted values are compared to the actual ones residing in the dataset. In case the predicted price is equal or less than the actual price, we consider bidding to be successful, (Within Market ranges) for that hourly auction. If not, we consider it unsuccessful. Three experiments were conducted, where `Cassandra` agents employed three different DM models:

i. The first DM model was extracted by the use of Simple Linear regression on raw data (no pre-processing was performed). Figure 7a depicts the results, where one may notice that most predictions were 'Unsuccessful'.
ii. The second DM model was again extracted by the use of Simple Linear regression, this time on a filtered (preprocessed) dataset. Figure 7b depicts the results, where improvement can be seen.
iii. Finally, the third DM model was extracted by the use of the Additive Regression with the M5' regression scheme. Figure 7c depicts the results, where improvement is obvious. 3.

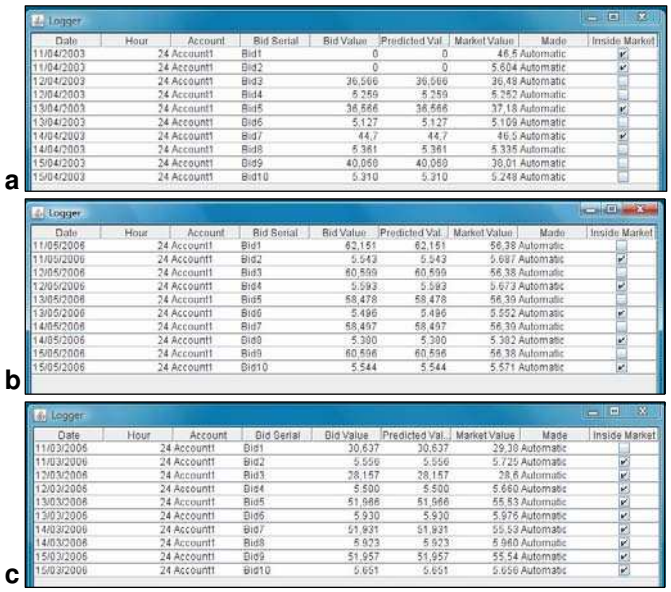


Fig. 7. Illustrating the accuracy of the three applied models

6 Conclusions and Future Work

Within the context of this paper we have presented `Cassandra`, a multi-agent system that employs DM primitives in order to automate the process of participating in the Power Stock Market. `Cassandra` succeeds in predicting forthcoming Settlement Prices and Power Load values, allowing its 'master' to bid within market ranges and maximize profit. The tool provides the ability to design the MAS and decide on the technique and algorithm on which to build the DM model on, while it provides a number of utilities for monitoring the market and analyzing trends.

`Cassandra`'s efficiency has been designed based on the Nash Equilibrium [20]. `Cassandra` offers the analyst with the ability to change system architecture dynamically, so as to adapt to changes of the market in terms of other players strategies.

Future work is focused on two directions: a) extensively study periodicity (same day each month, same weekend each year and so on), in order to identify an even more efficient model for staying 'within market ranges' and b) identify the maximum price to bid so as to maximize revenue. Additionally, one may work towards improving the GUI (Graphical User Interface) of the platform. The addition of more graphs, tables and diagrams would help in extracting useful information produced during the system's operation.

References

1. M. Amin. *Market Analysis and Resource Management*, chapter Restructuring the Electric Enterprise, pages 2–16. Kluwer Publishers, 2002.

2. M. Amin and D. Ballard. Defining new markets for intelligent agents. *IT Professional*, 2(4):29–35, 2000.
3. A. J. Bagnall and G. D. Smith. Game playing with autonomous adaptive agents in a simplified economic model of the uk market in electricity generation. In *In IEEE-PES / CSEE International Conference on Power System Technology POWERCON 2000*, pages 891–896, 2000.
4. F. Bellifemine, A. Poggi, and R. Rimassa. Developing multi-agent systems with JADE. *Lecture Notes in Computer Science*, 1986:89–101, 2001.
5. G. Conzelmann, G. Boyd, V. Koritarov, and T. Veselka. Multi-agent power market simulation using emcas. pages 2829–2834 Vol. 3, June 2005.
6. D. Freedman. *Statistical Models : Theory and Practice*. Cambridge University Press, August 2005.
7. M. He, N. R. Jennings, and H. fung Leung. On agent-mediated electronic commerce. *IEEE Trans. Knowl. Data Eng.*, 15(4):985–1003, 2003.
8. J. H. Holland. Genetic Algorithms and Classifier Systems: Foundations and Future Directions. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA87)*, pages 82–89, Cambridge, MA, 1987. Lawrence Erlbaum Associates.
9. D. P. Koesrindartoto and J. Sun. An agent-based computational laboratory for testing the economic reliability of wholesale power market designs. Computing in Economics and Finance 2005 50, Society for Computational Economics, Nov. 2005.
10. D. P. Koesrindartoto and L. S. Tesfatsion. Testing the reliability of ferc's wholesale power market platform: An agent-based computational economics approach. Staff General Research Papers 12326, Iowa State University, Department of Economics, May 2005.
11. V. Petrov and G. Sheble. Power auctions bid generation with adaptive agents using genetic programming. In I. of Electrical and E. Engineers, editors, *Proceedings of the 2000 North American Power Symposium*, 2000.
12. I. Praca, C. Ramos, Z. Vale, and M. Cordeiro. Mascem: a multiagent system that simulates competitive electricity markets. *Intelligent Systems, IEEE*, 18(6):54–60, Nov-Dec 2003.
13. I. Praca, C. Ramos, Z. Vale, and M. Cordeiro. Intelligent agents for negotiation and game-based decision support in electricity markets. *Engineering intelligent systems for electrical engineering and communications*, 13(2):147–154, 2005.
14. I. Praca, C. Ramos, Z. Vale, and M. Cordeiro. Testing the scenario analysis algorithm of an agent-based simulator for competitive electricity markets. In ECMS, editor, *Proceedings 19th European Conference on Modeling and Simulation*, 2005.
15. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, Massachusetts, 1994.
16. P. Stone. Learning and multiagent reasoning for autonomous agents. In *of the 20th International Joint Conference on Artificial Intelligence*, pages 13–30, 2007.
17. A. L. Symeonidis and P. A. Mitkas. *Agent Intelligence Through Data Mining*. Springer Science and Business Media, 2005.
18. A. Tellidou and A. Bakirtzis. Multi-agent reinforcement learning for strategic bidding in power markets. pages 408–413, Sept. 2006.
19. The FIPA Foundations. Foundation for intelligent physical agents specifications. Technical report, The FIPA Consortium, 2003.
20. W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.
21. C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.
22. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman, 2000.

23. P. R. Wurman, M. P. Wellman, and W. E. Walsh. The michigan internet auctionbot: a configurable auction server for human and software agents. In *In Second International Conference on Autonomous Agents*, pages 301–308. ACM Press, 1998.

Author Index