Sketching a methodology for efficient Supply Chain Management agents enhanced through Data Mining

ARTICLE *in* INTERNATIONAL JOURNAL OF INTELLIGENT INFORMATION AND DATABASE SYSTEMS · JANUARY 2008

DOI: 10.1504/IJIIDS.2008.017244 · Source: DBLP

3 AUTHORS, INCLUDING:

Andreas Symeonidis
Aristotle University of Thessaloniki
79 PUBLICATIONS **336** CITATIONS
SEE PROFILE

Pericles A. Mitkas
Aristotle University of Thessaloniki
250 PUBLICATIONS **1,372** CITATIONS
SEE PROFILE

Sketching a methodology for efficient Supply Chain Management agents enhanced through Data Mining

Andreas L. Symeonidis*, Vivia Nikolaidou and Pericles A. Mitkas

Electrical and Computer Engineering Department,
Aristotle University of Thessaloniki,
54 124 Thessaloniki, Greece
E-mail: asymeon@iti.gr
E-mail: vivia@ee.auth.gr
E-mail: mitkas@eng.auth.gr
*Corresponding author

Abstract: Supply Chain Management (SCM) environments demand intelligent solutions, which can perceive variations and achieve maximum revenue. This highlights the importance of a commonly accepted design methodology, since most current implementations are application-specific. In this work, we present a methodology for building an intelligent trading agent and evaluating its performance at the Trading Agent Competition (TAC) SCM game. We justify architectural choices made, ranging from the adoption of specific Data Mining (DM) techniques, to the selection of the appropriate metrics for agent performance evaluation. Results indicate that our agent has proven capable of providing advanced SCM solutions in demanding SCM environments.

Keywords: Data Mining; DM; intelligent agents; Supply Chain Management; SCM; agent-oriented methodology; bidding; forecasting.

Reference to this paper should be made as follows: Symeonidis, A.L., Nikolaidou, V. and Mitkas, P.A. (2008) 'Sketching a methodology for efficient Supply Chain Management agents enhanced through Data Mining', *Int. J. Intelligent Information and Database Systems*, Vol. 2, No. 1, pp.49–68.

Biographical notes: Andreas L. Symeonidis is a Lecturer with the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki and a Research Associate with the Informatics and Telematics Institute in Thessaloniki, Greece. He received his Diploma and PhD from the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki in 1999 and 2004, respectively, and has recently concluded his post-doctoral research on the evaluation of agent efficiency. His research interests include software agents, data mining and knowledge extraction, intelligent systems and evolutionary computing.

Vivia Nikolaidou is a PhD student at the Electrical and Computer Engineering Department of the Aristotle University of Thessaloniki, Greece. Her PhD thesis subject is "Self-evaluation and autonomous improvement of agent behaviour in multi-agent systems through data mining". She graduated in Electrical and Computer Engineering at the same university in 2004. Her research interests include intelligent software agents and multi-agent systems, data mining, databases and software engineering.

Pericles A. Mitkas is a Professor of Electrical and Computer Engineering at the Aristotle University of Thessaloniki, Greece. He is also the Associate Director of the Informatics and Telematics Institute of the Center for Research and Technology – Hellas (CERTH). His research interests include databases and knowledge bases, data mining, software agents, and bioinformatics. He received his Diploma in Electrical Engineering from Aristotle University of Thessaloniki in 1985 and an MSc and PhD in Computer Engineering from Syracuse University, USA, in 1987 and 1990, respectively. Between 1990 and 2000, he was a Faculty member with the Department of Electrical and Computer Engineering at Colorado State University in USA.

1 Introduction

The interaction between intelligent agents and DM is a new and interesting research field, which can be viewed from two different perspectives: either top-down, where agents are used as the discrete, yet interacting elements of the knowledge discovery process (data accumulation, cleaning, algorithm application etc.), or bottom-up, where DM is used for the extraction of useful knowledge models that can be incorporated into the agents, to improve their performance/efficiency.

In the second approach, issues related to modelling and measuring improvement usually arise, leading to the definition of ad hoc metrics that cannot be used in a more generic manner. Our work is focused towards this direction: to survey existing literature, identify common practices, and provide a methodology to find a set of valid metrics and use them to evaluate the efficiency of agents in a very popular (for DM experts) domain: the field of SCM.

SCM primitives comprise the management of materials, information and finance in a network consisting of suppliers, manufacturers, distributors and customers (Stanfield, 2002). Practically, all these activities are intended to deliver the optimal result to the end-user via procurement of raw materials, manufacturing, distribution, and customer service (Kim et al., 2004). Nowadays, increasing competition has emphasised the need for more flexible, robust and powerful SCM. Managing the supply chain requires coordination of distributed and heterogeneous information. Since Multi-Agent Systems (MAS) are designed for distributed problem-solving and negotiating, it is clear that they provide an ideal solution to this problem (Ferber, 1999). This is also supported by Wu et al. (2000), who present a multi-agent modelling framework based on explicit communication between constituent agents, such as manufacturers, suppliers, retailers and customers. Indeed, SCM is fundamentally concerned with coherence among multiple, globally distributed decision-makers.

On the other hand, issues related to SCM have been successfully dealt with in the past with the exploitation of DM techniques. Customer and supplier categorisation, market basket analysis, and inventory scheduling are typical problems where DM is applied, to provide efficient solutions.

In this context, we introduce a methodology, based on intelligent agent technology and data-mining techniques, which allows us to model, measure and improve the performance of SCM agents. We have crash-tested our approach in an extremely dynamic and multi-variate environment; the TAC SCM game, described in further detail by Collins et al. (2004). In TAC SCM, offering a bid at the optimal price is a crucial

point in the game, since it directly affects the agent's profit and ranking. We have thoroughly analysed the problem at hand and have come up with a methodology, which provided us with a number of TAC-related metrics, helping us to evaluate our agent's performance. We have carefully selected a combination of DM techniques to extract the appropriate knowledge models, which are used by the agent to predict the winning price of bids. The final ranking of our agent, agent *Mertacor*, with respect to others, indicates that data mining can indeed be used to considerably improve an agent's performance in such environments.

The rest of the paper is organised as follows: Section 2 reviews existing methodologies of agent-based SCM, focusing mainly on evaluation methods. Section 3 presents our proposed methodology, striving to find the correct set of metrics and evaluate the system. Section 4 provides an overview of the TAC SCM game and justifies *Mertacor's* architecture, while it identifies where the system characteristics evaluation is going to be performed on. Section 5 focuses on the DM evaluation of the system, describing in detail the DM experiments conducted, to decide on the best winning bid price model, and testing it against an ad hoc, nevertheless useful bidding mechanism. Section 6 summarises our work and concludes the paper.

2 Related work: existing methodological approaches

The problem of defining the most appropriate methodology for developing a *DM-enhanced* SCM MAS[1] is two-fold:

- decide on the most significant of the attributes of the agent at hand, to which the software developer must focus on
- decide on the way the evaluation of performance/efficiency of an SCM problem should be addressed.

Within this context, we have conducted a thorough literature review to identify the state-of-the-art approaches for intelligent (multi-)agent system development in general, and for the SCM domain in specific.

2.1 Intelligent MAS scope and attributes

According to Landauer and Bellman (2002), the most important characteristics of an intelligent computing (agent) system are:

- speed and scope of adaptability to unforeseen situations
- rate of effective learning
- accurate modelling and prediction of the relevant external environment
- speed and clarity of problem identification and formulation
- effective association and evaluation of disparate information
- identification of assumptions and pre-requisites
- creation and use of symbolic language.

On the other hand, Dumke et al. (2000) define a set of metrics for all agent and multi-agent system levels (design, description, working, etc., levels). At the agent working level, metrics are provided for the following agent system characteristics:

- communication
- interaction
- learning
- adaptation
- negotiation
- collaboration
- coordination
- cooperation
- self-reproduction
- performance
- specialisation.

May one decide to develop an MAS, he or she is required to identify the focus of the application and decide on a combination of practices and metrics, so as to better evaluate its performance. In the case of SCM, where focus is usually given on revenue and profit, some of these metrics are considered subsidiary (i.e., self-reproduction), while others are paid extreme attention to (i.e., performance, adaptability). The next section discusses the existing approaches.

2.2 Evaluation of Supply Chain Management (SCM) MAS

Smirnov et al. (2004) discuss resource allocation Genetic Algorithms (GAs). They consider two sub-problems:

- coalition formation
- product and resource allocation tasks in a multi-agent environment.

In their evaluation, they compare the two approaches, having them approximate positive ramp membership functions in terms of pay-offs (e.g., pay-off per car, gross pay-offs per body).

Moyaux et al. (2004) use game theory to analyse collaborative strategies in a forest supply chain. They use collaboration in an attempt to reduce the Bullwhip effect (phenomenon where the variance of orders amplifies upstream the supply chain). They run a simulation to evaluate each company's inventory holding and backorder costs and build a game in the normal form, which is then analysed using *Game Theory*. They identify two *Nash equilibria* incurring the minimum cost of the supply chain, both of which include collaboration between companies. They evaluate performance by minimising the cost in an attempt to reduce the Bullwhip effect.

Simek et al. (2004) use Reinforcement Learning (RL) algorithms for procurement agents. They introduce their SCM model and show that RL outperforms traditional tools for dynamic problem-solving in daily business, but identify situations where RL fails to perform efficiently. RL is compared against a primitive (but effective) algorithm and against a double exponential smoothing mechanism, and the evaluation is based on total reward.

Kwon et al. (2005) employ optimisation and Case-Based Reasoning (CBR) models on a web services-based platform. They argue that these methods fail to address more complicated problems such as revenue maximisation and stochastic dimension, despite the advancement of optimisation techniques. They compare the performance outcome of the prototype system, which uses linear programming and mixed integer programming, against their optimisation model, on a variety of scenarios. Evaluation is performed with respect to an ad hoc relative ratio, which is based on performance outcome (in terms of profit).

Johnston et al. (2005) employ agent technology in order to improve the performance of the traditional Kanban system, while maintaining its recognised usability. They present the Intelligent Kanban system and test it against three other different Kanban system implementations, as well as against the Traditional Kanban System. Evaluation is performed with respect to wait time, total stock and percentage of back order.

Based on the above, one may argue that there is currently no standard way of evaluating the different approaches employed in SCM-related MAS. Evaluation metrics are ad hoc, and most of the times performance is measured in terms of profit, since it is obviously the most immediate target of SCM. Thus, special care should be taken to distinguish a mere profitable system from a truly intelligent one. Additionally, for SCM MAS it is a common practice to test the proposed mechanism/algorithm against a heuristics-based approach that satisfies certain criteria.

Having reviewed the existing work, we present a methodology, which allows us to select an appropriate set of metrics for a DM-enhanced SCM MAS and correctly evaluate its performance.

3 A methodology to identify metrics in DM-enhanced SCM MAS

Going through the variety of SCM models conceived in bibliography, one may argue that focusing on the SCM part would force one to either develop his or her model keeping a particular chain in mind or come up with a possibly inadequate model. Our attempt was to reach the 'happy medium', by not focusing on the exact tasks of the supply chain but on the decomposition of their processes, after they have been defined by the expert user. Our methodology is aimed to be generic instead of constrained to each specific implementation. It attempts to identify the most important system characteristics and find the appropriate metrics, as well as the appropriate way to measure them, to have a correct evaluation. It consists of the following steps:

- identify process layers
- identify actions and interactions within each layer
- identify the system characteristics to be evaluated for each action/interaction
- select the respective set of metrics

- perform measurements
- compare against a heuristics-based approach.

The above-mentioned steps are applied to all software engineering facets involved in the design and implementation of the MAS. In the case of DM-enhanced MAS, these facets are:

- the modelling of the agent system itself, by specifying actions and interactions
- the identification and exploitation of the DM-extracted knowledge models.

In the remainder of this paper, an overview of the generic model is provided, while focus is given on the DM facet. Details on the agent-modelling facet can be found at Symeonidis and Mitkas (2005). An abstract model for SCM MAS that embeds knowledge extracted through DM is depicted in Figure 1.

The methodology steps for a DM-enhanced SCM MAS are as follows:

Step 1: Any SCM MAS with DM capabilities may be structured as a four-layer architecture. These layers are:

- the data pre-processing
- knowledge discovery
- knowledge organisation and decision–making
- the knowledge diffusion layer.

Steps 2–3: Table 1 denotes the action/interactions between the aforementioned layers and their corresponding metrics, mainly focused on the DM facet.

Step 4: The user has to specify the appropriate metrics for the system characteristics involved in each action/interaction, to succeed in defining a benchmark for evaluating the performance of the system. Complicated benchmarking processes may arise in case the user decides to measure disjoint actions and interactions. Nevertheless, in DM-enhanced SCM MAS, focus is mainly given on *Adaptation* and *Specialisation*, key characteristics of the Knowledge Discovery layer. Improvement of these characteristics usually has a positive effect on the agent's *Performance* as well.

Step 5: *Learning* can be measured by applying different DM algorithms on the same data set and evaluating their performance. Carefully selecting the appropriate model and tuning its parameters improves *Learning*. In a similar way, *Adaptation* is measured by using the derived models on different situations and can be improved by retraining our agents.

Step 6: It is a common practice for DM-enhanced SCM MAS to adopt a heuristics approach with acceptable performance, to have a fail-safe mechanism. Such a mechanism is often employed to benchmark DM models' efficiency and evaluate it with respect to anticipated profit. In case of undesired behaviour (i.e., exceeding a cut-off threshold), the DM process is overridden and the heuristic approach takes over.

Figure 1 An abstract model for DM-enhanced SCM MAS

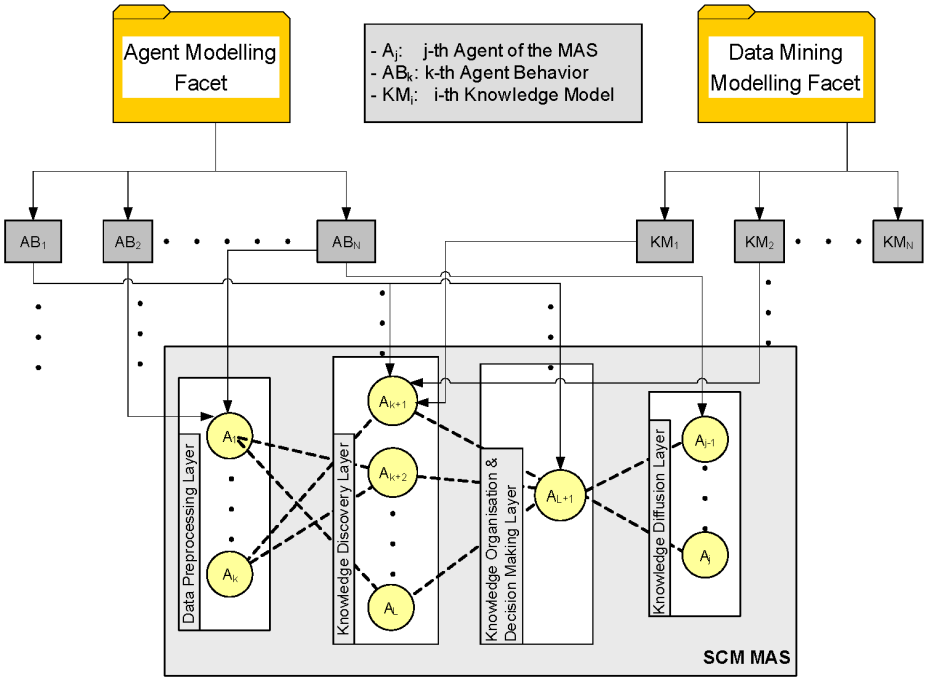


Table 1 Identification of the actions/interactions between SCM MAS layers and the corresponding characteristics to be evaluated

Process layer	*Action/interaction*	*System characteristics*
Preprocessing	Select appropriate data categories	Learning, specialisation
	Select chain parts	Learning, specialisation, communication, collaboration, cooperation, coordination
	Pre-process data	Learning, specialisation, performance, adaptation
Knowledge discovery	Select algorithms	Adaptation, performance, specialisation
	Evaluate algorithms	Learning, performance, specialisation
Knowledge organisation	Evaluate and validate knowledge models	Learning, adaptation, performance, specialisation
	Incorporate static business rules	Specialisation, interaction, learning, adaptation
	Organise and integrate knowledge	Learning, adaptation, specialisation
Diffusion	Locate interested parties	Learning, adaptation, negotiation, self-reproduction, specialisation
	Transmit information	Communication, interaction, adaptation, negotiation, collaboration, cooperation, coordination, self-reproduction, specialisation
	Receive feedback	Communication, interaction, adaptation, negotiation, collaboration, cooperation, coordination
	Process feedback	Learning, adaptation, negotiation, performance, specialisation

The next sections attempt to model – through the methodology presented in this section – a DM-enhanced agent competing at the TAC SCM. All layers identified in an SCM MAS (Step 1) have been incorporated into one agent and Section 4 provides a justification for this choice. In addition, Section 4 specifies the system characteristics that have to be evaluated in each layer, while Section 5 focuses on the Pre-processing and Knowledge Discovery layers, to help the reader better understand the evaluation of a DM-enhanced MAS within its appropriate context.

4 An intelligent approach: Mertacor

4.1 The SCM TAC game

Within the scenario of the TAC SCM game, as described by Collins et al. (2004), each participating agent represents a PC assembler with limited production capacity. Six such agents compete in selling 16 different types of PCs to potential customers. The agents' task is to negotiate on supply contracts, bid for customer offers, manage daily assembly activities and ship completed offers to customers. The environment constitutes a supply chain, where agents trade simultaneously with suppliers (for PC components procurement) and customers (to sell assembled units). The produced PCs are divided into three product ranges (*Low*, *Medium* and *High*) according to the compilation implemented.

Negotiations are performed through a Request-For-Quote (RFQ) mechanism, which proceeds in three steps:

- Buyer issues RFQs to one or more sellers.
- Sellers respond to RFQs with offers.
- Buyers accept or reject offers. An accepted offer becomes an order.

The customer requires a reserve price, which is between 75% and 125% of nominal price of components. To get paid, the agent must deliver on time; otherwise, it is charged with a penalty. At the end of the game, the agent with the greatest revenue is declared winner. Game length is 220 simulated days, each day lasting 15 s. Arunachalam and Sadeh (2004) provide more information on the game.

4.2 Mertacor architecture

May one examine the specifications of the TAC SCM game, he or she may identify four tasks each agent needs to deal with:

- procurement of hardware components and negotiation on cheap component contracts
- management of the inventory and stock requirements
- scheduling of the production and delivery process
- bidding and negotiation on PC sales to customers.

As described by Kontogounis et al. (2006), our *Mertacor* employs a modular architecture, having one module for each of the aforementioned tasks. The agent's core provides a wrapper around the modules and undertakes communication with suppliers and customers. This architecture provides flexibility and extensibility, making it easy to apply *Mertacor*'s strategy to other real-life SCM environments as well.

Mertacor's four modules are:

- the Inventory Module (IM)
- the Procuring Module (PM)
- the Factory Module (FM)
- the Bidding Module (BM).

Two of the most significant strategies of inventory management are make to order and make to stock. *Mertacor*'s *IM* employs an Assemble-To-Order (ATO) strategy, a hybrid combination of the two aforementioned techniques, which defines the inventory levels that need to be satisfied and below which replenishment is needed. The ATO strategy was selected as the most appropriate one, since, according to Cheng et al. (2001), it is the most suitable strategy for environments where replenishment times are larger than assembly times, like in the case of the TAC SCM game.

PM uses heuristic techniques to predict future demands and order affordable components in advance. During the first two days, *Mertacor* attempts to build an initial inventory, to quickly start production. On the third day, *Mertacor* switches to a normal-state procurement strategy, dividing the RFQs it sends into three categories: Normal, to satisfy inventory reorder levels, Critical, to satisfy customer orders, and Early, to obtain low-priced components before they may be needed.

FM is responsible for producing accurate assembly schedules and for providing the bidder with information on the factory production capacity. It creates a projection of what the factory should expect in the near future by realising a simulation model, based on knowledge on future supplier deliveries, customer deliveries, customer orders, future production and delivery schedules. Demand is simulated for the 15 forthcoming days (look-ahead time) and then information is diffused to the rest of the modules.

Finally, BM attempts to predict the winning price of each order. It performs offline DM on logs of previously played games and uses the derived models to predict the winning price of each order. Although these models are highly reliable (see analysis in the following section), a fail-safe (non-intelligent) mechanism has also been incorporated, to ensure agent efficiency in the transitional phases of the game (Start/End phase).

The architecture adopted for *Mertacor* abides by the requirements discussed in Section 2. As far as the characteristics of intelligent systems are concerned (Section 2.1, part (a)), one may argue that apart from the creation of the symbolic language, which does not directly apply to *Mertacor*'s context, all other primitives (learning, modelling and association of disparate information) are taken into consideration, either by the TAC SCM game server (interfaces to suppliers and customers, game rules), or through the adoption of the knowledge extraction mechanism (algorithmic performance, modelling of the environment, etc.).

After having applied our proposed methodology in Section 3, we have come to the conclusion that *Learning*, as well as *Adaptation*, are the two characteristics that each TAC SCM participating agent should focus on. By the use of a well-justified DM

mechanism, *Learning* becomes an inherent property of our agent, while *Adaptation* is met by adjusting *Mertacor*'s bidding policy to the varying competition difficulty (for example, semifinal and final rounds were more competitive than qualifying rounds). The adopted architecture proved efficient in all competition levels (Collins et al., 2004).

Since the whole notion of the game is organised around agent revenue, *Learning* and *Adaptation* were measured with respect to the Bidding Module, since predicting the winning price of each bid proved crucial to agents' efficiency. Too high a price may result to failure of coming to an agreement with the customer, while too low a price may decrease profit. Apart from the evaluation of the DM mechanism itself, we also evaluated it against the fail-safe mechanism. The analysis of the bidding mechanism (Knowledge Discovery layer), as well as its evaluation, is discussed next.

5 Evaluation of Mertacor's bidding mechanism

Estimation of the winning price of the bids can be modelled as a regression problem, where the desired output is the agent's bidding price for clients' RFQs and the inputs are the parameters related to the bid that are known to the agent. The initial set of attributes considered is shown in Table 2.

Available data was split into three subsets, each one representing a different market range (LOW–MEDIUM–HIGH), both for the finals (placement positions 1–6) and second finals (placement positions 7–12) of the 2005 game, resulting to six different data sets (*finalsLOW–MEDIUM–HIGH* and *secondFinalsLOW–MEDIUM–HIGH*). The interested reader can download this data on the TAC website (http://www.sics.se/tac/). To experiment on the data with a variety of training techniques and algorithms, the Waikato Environment for Knowledge Analysis (WEKA) suite, described by Witten et al. (1999) was used, providing with a wide range of filters for pre-processing, model evaluation, visualisation and post-processing.

Table 2 Initial set of attributes for Mertacor's bidding mechanism

Attribute description	*Attribute name*
Demand (Total PCs requested the day the RFQ was issued)	`demandAll`
Demand in the product's market range	`demandRange`
Duedate	`dueDate`
Reserve price	`reservePrice`
Maximum price of PCs of same type sold in the last one day	`max1`
Maximum price of PCs of same type sold in the last two days	`max2`
Minimum price of PCs of same type sold in the last one day	`min1`
Minimum price of PCs of same type sold in the last two days	`min2`
Minimum price of PCs of same type sold in the last three days	`min3`
Minimum price of PCs of same type sold in the last four days	`min4`
Winning price of the bid	`price`

5.1 Pre-processing

The instances within the initial data sets ranged from 45,000 to 230,000 instances. Analysis was performed on all data sets. Nevertheless, owing to space limitation, we shall discuss only one case (*finalsLOW* data set), while analysis on the other cases was performed in an analogous manner. The initial data set contained 156,228 records of bids. To remove redundant information and enable quicker and more accurate training, a number of pre-processing filters were tested against the data set. In particular, we applied the *CfsSubsetEval*[2] (Hall, 1998), *WrapperSubsetEval*[3] (Kohavi and John, 1997), and *ReliefFAttributeEval*[4] (Sikonja and Kononenko, 1997) filters for attribute selection, using the *GreedyStepwise* and *RandomSearch* search methods. The trimmed data set contained the following attributes as input: the demand, the reserve price for components, the maximum price of same type PCs for the two previous days and the minimum price for the previous day, while price was the output attribute. To reduce the number of instances for training, and since the class attribute (price) is numeric, the *StratifiedRemoveFolds*[5] (Breiman et al., 1984) method was selected.

Two data sets were finally produced, containing the one-third (1/3) and one-eighth (1/8) of the initial instances, respectively. On these resulting data sets, a number of classifiers and meta-classifier schemata were applied. These are discussed below.

5.2 Training

Four different classification (regression) schemas were applied, to decide on the one that optimally meets the problem of predicting the winning bid of an order:

- *Linear Regression*
- *Neural Networks*
- *SMOreg* (*Support Vector Machines*)
- the *M5*' algorithm.

A number of experiments were conducted applying *Linear Regression*, on different data sets and with different algorithm parameters. This algorithm had a rather mediocre efficiency, with a *correlation coefficient* (*cc*) of around 0.97, a *Relative Absolute Error* (*RAE*) within the range of 15–20% and a *Root Mean Square Error* (*RMSE*) within the range of 65–75% for the finals and 90–110% for the second finals. All values remained at the same level through variations on the initial data sets and the algorithm's parameters.

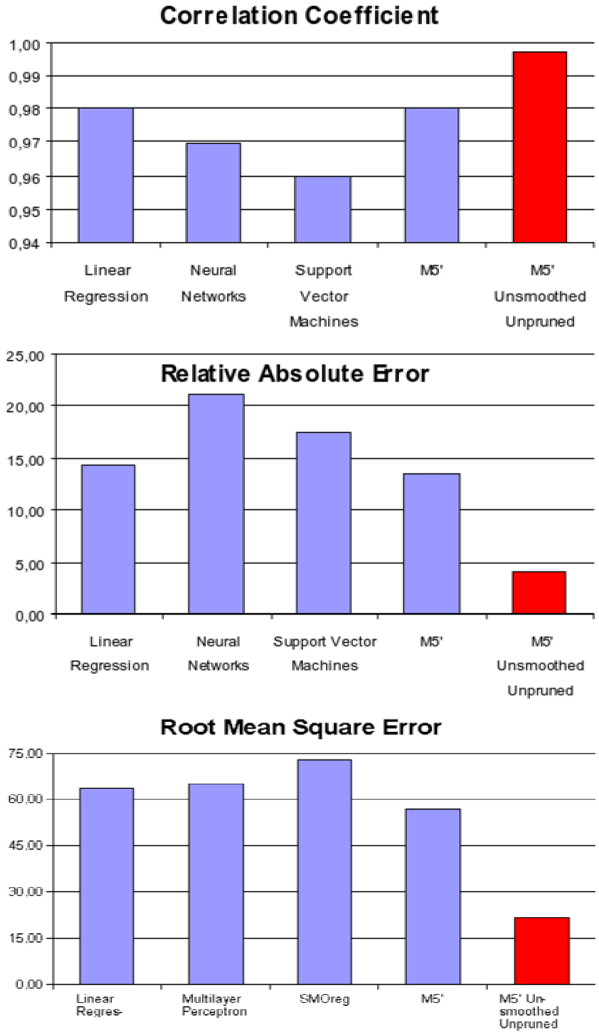
Neural Networks, though expected to perform better (Wang and Witten, 1997), yielded worse results in our case, with a *correlation coefficient* and *RMSE* approximately the same as in the case of *Linear Regression*, but with *RAE* always around 20%. It should be denoted, though, that increasing or decreasing the learning rate did not seem to significantly influence the resulting efficiency, while using different training techniques (either using a training set or *n*-fold cross-validation) nearly doubled the *Mean Absolute Error* (*MAE*), indicating overfitting.

In the case of *Support Vector Machines*, where *Lagrange* multipliers are employed to find a function that approximates the data, the performance was comparable with the aforementioned techniques, while its performance was significantly slower. *SMOreg* failed to give an *RAE* lower than 16%, while exceeding an *RAE* of 20%, when it came to the data sets of the second finals.

In the case of the *M5*', however, the derived models' efficiency was substantially better, having a *cc* around 0.99, an *RAE* of about 10% and *RMSE* of about 50. Furthermore, when performing DM on unsmoothed data and producing an unrpruned model, the *cc* reached 0.9967, while decreasing the *RAE* to about 4.0112%, *RMSE* to 25 and keeping a very small value for *MAE* (9.0825).

A comparison of all the aforementioned techniques is illustrated in Figure 2. All experiments were conducted using the *finalsLOW* data set of five attributes and 1/8 of the initial data set and having the same user-supplied test set.

Figure 2 Qualitative comparison of the classifiers applied



5.3 Meta-classification

Apart from the classification schemes applied, we also tested some meta-classifier schemes, striving for the optimum performance. *Additive Regression* and *Bagging*, described by Friedman (1999) and Breiman (1996), respectively, consecutively train learning schemes with different bootstrap samples of the initial data set. *Bagging* comes from the term 'Bootstrap Aggregating', while *Additive Regression* introduces some randomness to the procedure. *Additive Regression* and *Bagging* were applied to the same data sets and using the same training options, to ensure result correctness and to be able to correctly compare their results with the results of the previous classification algorithms.

The *Additive Regression* schema was applied using the following classifiers:

- *DecisionStamp*[6]
- *REPTree*[7]
- the *M5*' classifier.

DecisionStamp's results were disappointing, having a *cc* varying from 0.91 to 0.97, an *RMSE* from 67 to 131 and a *RAE* from 25% to 40%. *REPTree* yielded significantly better results, having a *cc* varying from 0.95 to 0.97, an *RMSE* that reached 40 at best and an *RAE* from 20% to 25%. However, the *M5*' classification schema once again performed optimally, with the *cc* reaching the maximum value of 0.9994, the *RMSE* from 8 to 25 and the *RAE* falling to 0.74%.

Bagging was applied using the *REPtree* and the *M5*' classifiers. As the performance of *DecisionStamp* was worse than the classifiers alone, we decided to exclude this algorithm from the *Bagging* schema. The first combination gave results comparable with those of *Additive Regression–REPtree* combination, with a *cc* of 0.99, an *RMSE* near 30 for the finals and 60 for the second finals and an *RAE* slightly above 5% for all experiments. When the *M5*' was used with Bagging, the results were better with respect to *REPTree* (*cc*: 0.99 – *RAE*: 9.64 – *RMSE*: 41.26), nevertheless did not reach the performance of the *Additive Regression – M5*' schema.

Figure 3 summarises the average performance of the *Additive Regression* and *Bagging* meta-classifier schemas, with the respective classifier used.

Figure 3 *cc*, *RAE* and *RMSE* of the applied meta-classifiers

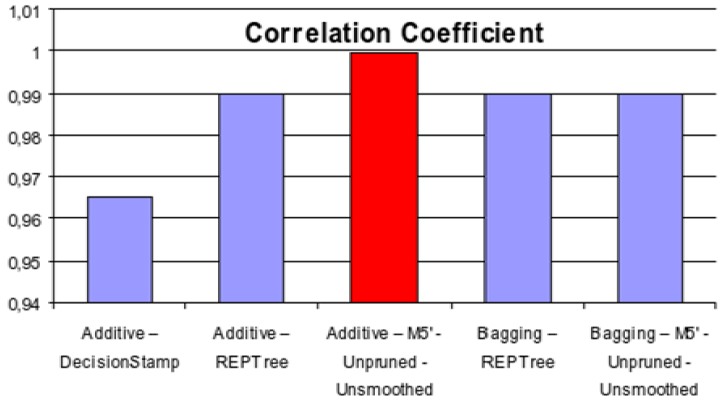
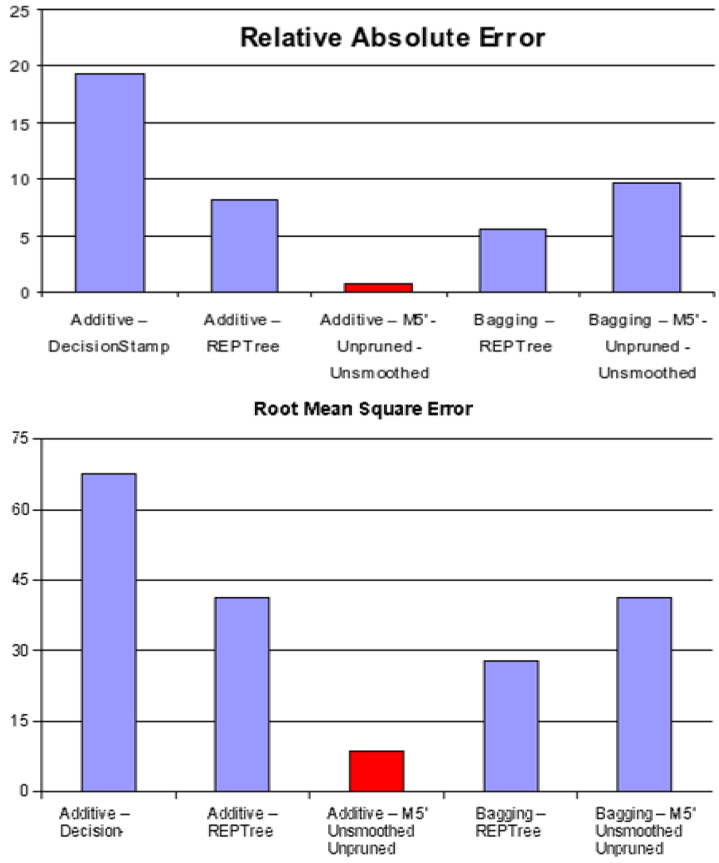


Figure 3 *cc*, *RAE* and *RMSE* of the applied meta-classifiers (continued)



5.4 Comparison with the fail-safe mechanism

To further evaluate our *model's* performance, we have compared it against *Mertacor*'s non-intelligent mechanism, the *Follower*. The *Follower*, named after its ability to follow bid prices online, evaluates the minimum and maximum prices for PCs ordered the previous day as provided by the daily price reports in order to approximate the winning price for bids. It assumes that the lowest price achieved on the previous day corresponds to the lowest customer RFQ reserve price and the highest price corresponds to the highest reserve price. It then calculates the current winning price by employing linear interpolation between the above values for the current reserve price.

The *Follower* was incorporated into *Mertacor* to help our agent perform better during the *Start*- and *End*- game effects. Since all six agents start the game with zero inventory, they attempt to procure components as soon as possible, in order to start assembly. Increased component demand leads to high component prices and, thus, increased PC prices. On the other hand, when reaching the end of the game, agents try to sell their remaining stock at any cost. Consequently, prices are significantly lower. In both phases, the model is not capable of capturing the trends of the market, since not enough data on these phases exist.

Table 3 Performance of our model for the three phases of the game

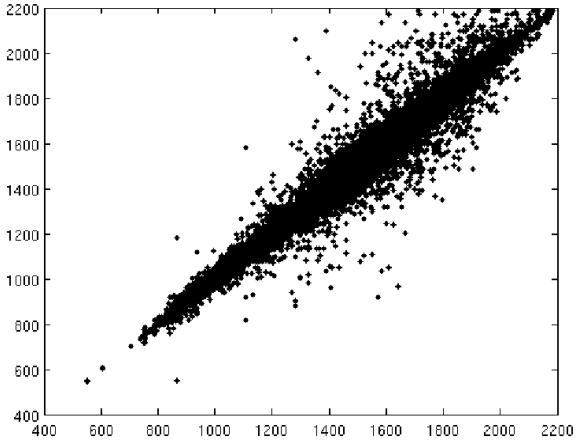
		Initial state	*Final state*	*Normal state*
Additive regression – M5' unsmoothed unpruned	CC	0.88	0.91	1.00
	RAE (%)	25.35	34.84	0.74
	RMSE	164.67	213.56	8.63
Fail-safe mechanism	CC	0.99	0.98	0.98
	RAE (%)	14.08	13.85	14.17
	RMSE	67.99	125.70	59.52

The fail-safe mechanism was tested against the *finalsLOW* data set in order to validate its usefulness. Indeed, the *Follower* resulted (*Normal State*) in a correlation coefficient close to 0.98, while the RAE was 14.17% and the RMSE was 59.52, performance comparable with some of the training algorithms considered. Figure 4 shows the predicted vs. actual price for the *finalsLOW* data set both for the selected model (Figure 4(a)) and the fail-safe mechanism (Figure 4(b)), while Table 3 illustrates the performance of the selected model against the fail-safe mechanism for the three game phases.

5.5 Results: evaluation

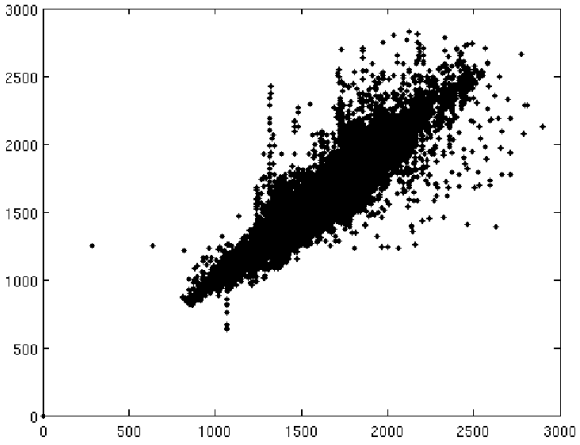
One can easily identify that the combination of the *Additive Regression* meta-classification schema with the *M5*' regression algorithm, applied on unsmoothed data and producing unpruned knowledge models *significantly outperforms* all other learning methods applied. This hypothesis was also validated on the rest available data sets (*finalsHIGH*, *finalsMEDIUM*, *secondFinals-HIGH*, *secondFinalsMEDIUM* and *secondFinalsLOW*), applying the same pre-processing processes. Table 4((a) for the second finals, (b) for the finals) summarises the results, showing that *Additive Regression* with *M5'* are the best choice for predicting the winning bid of an order.

Figure 4 (a) Predictive accuracy of the winning learning model and (b) the fail-safe mechanism



(a)

Figure 4 (a) Predictive accuracy of the winning learning model and (b) the fail-safe mechanism (continued)



(b)

Table 4(a) Comparing knowledge models' efficiency against all available second finals data sets

	Algorithm	*CC*	*RAE* (%)	*RMSE*
SecondFinalsLOW	Linear regression	0.93	28.99	90.17
	Neural networks	0.93	32.91	94.69
	Support vector machines	0.93	26.47	89.08
	M5'	0.95	22.77	61.09
	M5' Unpr. Unsm.	0.99	9.58	74.80
	Additive Regr. – DecisionStump	0.94	28.22	86.96
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*3.21*	*22.12*
	Bagging – REPTree	0.98	14.89	52.02
	Bagging – M5' Unpr. Unsm.	0.99	9.33	35.08
SecondFinalsMEDIUM	Linear regression	0.94	26.50	112.71
	Neural networks	0.95	24.85	105.69
	Support vector machines	0.93	28.66	109.61
	M5'	0.97	19.30	86.90
	M5' Unpr. Unsm.	0.99	8.63	49.22
	Additive Regr. – DecisionStump	0.94	26.74	111.85
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*3.01*	*23.53*
	Bagging – REPTree	0.98	13.26	65.52
	Bagging – M5' Unpr. Unsm.	0.99	9.72	41.48
SecondFinalsHIGH	Linear regression	0.94	26.78	105.51
	Neural networks	0.95	27.83	105.21
	Support vector machines	0.94	25.82	103.32
	M5'	0.96	21.29	87.14
	M5' Unpr. Unsm.	0.99	9.92	49.18
	Additive Regr. – DecisionStump	0.91	37.80	131.45
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*2.98*	*24.70*
	Bagging – REPTree	0.98	15.13	65.69
	Bagging – M5' Unpr. Unsm.	0.99	9.22	41.99

Table 4(b) Comparing knowledge models' efficiency against all available finals data sets

	Algorithm	*CC*	*RAE* (%)	*RMSE*
FinalsLOW	Linear regression	0.98	14.34	63.40
	Neural networks	0.97	21.26	64.82
	Support vector machines	0.96	17.48	72.84
	M5'	0.98	13.49	56.79
	M5' Unpr. Unsm.	1.00	4.00	21.62
	Additive Regr. – DecisionStump	0.97	19.24	67.51
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*0.74*	*8.63*
	Bagging – REPTree	0.99	5.62	27.76
	Bagging – M5' Unpr. Unsm.	0.99	9.64	41.26
FinalsMEDIUM	Linear regression	0.97	16.95	74.33
	Neural networks	0.97	20.21	75.20
	Support vector machines	0.97	17.54	73.81
	M5'	0.99	9.91	46.93
	M5' Unpr. Unsm.	1.00	3.93	24.02
	Additive Regr. – DecisionStump	0.96	25.98	92.30
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*0.90*	*16.14*
	Bagging – REPTree	1.00	4.84	31.38
	Bagging – M5' Unpr. Unsm.	1.00	3.45	20.83
FinalsHIGH	Linear regression	0.97	16.55	68.14
	Neural networks	0.98	18.94	71.91
	Support vector MACHINES	0.97	16.27	72.31
	M5'	0.99	10.03	45.35
	M5' Unpr. Unsm.	1.00	3.87	22.06
	Additive Regr. – DecisionStump	0.95	28.26	94.68
	Additive Regr. – M5' Unpr. Unsm.	*1.00*	*0.98*	*15.42*
	Bagging – REPTree	0.99	5.70	34.90
	Bagging – M5' Unpr. Unsm.	1.00	3.22	17.44

Attempting an intuitive explanation for the improved efficiency of the *M5*' algorithm in our problem, one could consider the type of data the learning schemes were applied on. The performance of *Linear Regression* and *Support Vector Machines* was obviously lower, since they both impose a linear relationship on the data. And, though *Support Vector Machines* are said to achieve non-linearity by mapping the training data (Smola and Scholkopf, 1998), still a priori knowledge of the mapping function is essential. The advantage of *M5*' over *Neural Networks* is that, since the former is based on decision trees, it produces a simple and compact model, while *Neural Networks* generally tend to produce extremely complicated models and need excessive parameter tweaking. As far as the meta-classifier schemes are concerned, *Bagging* is designed to improve the accuracy of predictors (Breiman, 1996), while *Additive Regression* is an enhancement over *Bagging* by introducing a stochastic factor (Friedman, 1999).

Consecutively, it is normal for the *Additive Regression – M5*' combination to perform optimally.

Another point that should be denoted is that, since the *Follower* performs in an analogous manner with some of the learning schemes applied, one should be very careful in selecting the training algorithm for accurate winning bid price prediction.

6 Conclusions

Within the context of this paper, we have introduced a successful paradigm of the coupling of Intelligent Agent technology with DM. Having taken state-of-the-art MAS development and SCM evaluation practices into account, we have proposed a methodology to help researchers identify the appropriate metrics for a DM-enhanced MAS for SCM and use these metrics to evaluate its performance. Using this methodology, we have identified the parameters that best fit our model. On these parameters, namely agent *Learning* and *Adaptation*, we have provided an extensive analysis on how DM can be employed to improve the intelligence of our agent, agent *Mertacor*. A number of metrics were applied to help evaluate our results before incorporating the selected model to our agent. These metrics included comparison of the algorithms, specifically their correlation coefficients and their RAEs and RMS errors, as well as a fundamental comparison of the best performing algorithm against *Mertacor*'s own fail-safe mechanism, used to predict bidding prices during the transitional phases. The proposed mechanism proved our agent capable of increasing its revenue by adjusting its bidding strategy.

Out in the real world, *Mertacor* was crash-tested in the TAC SCM 2005 competition and succeeded in placing *third* among 32 agents from universities and research centres from all over the world.

Future directions on the coupling of agents and DM include the application of our methodology to improve the models used to predict the behaviour of both suppliers and customers, as well as to other real-life SCM scenarios.

References

Arunachalam, R. and Sadeh, N.M. (2004) 'The supply chain trading agent competition', *Electronic Commerce Research and Applications*, Vol. 4, pp.63–81.

Breiman, L. (1996) 'Bagging predictors', *Machine Learning*, Vol. 24, No. 2, pp.124–140.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classification and Regression Trees*, Chapman and Hall, New York.

Cheng, F., Ettl, M. and Lin, G. (2001) *Inventory-Service Optimization in Configure-to-Order Systems*, Technical Report RC 21781, IBM.

Collins, J., Arunachalam, R., Sadeh, N., Ericsson, J., Finne, N. and Janson, S. (2004) *The Supply Chain Management Game for the 2005 Trading Agent Competition*, Technical Report CMU-ISRI-04-139, CMU.

Dumke, R.R., Koeppe, R. and Wille, C. (2000) *Software Agent Measurement and Self-Measuring Agent-Based Systems*, Fakultaet fuer Informatik, Otto-von-Guericke-Universitaet Magdeburg, Preprint No. 11.

Ferber, J. (1999) *Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, London.

Friedman, J.H. (1999) *Stochastic Gradient Boosting*, Technical Report Stanford University, Stanford University, Palo Alto, California.

Hall, M.A. (1998) *Correlation-based Feature Subset Selection for Machine Learning*, Thesis submitted in partial fulfilment of the requirements of the degree of Doctor of Philosophy at the University of Waikato, Hamilton, New Zealand.

Johnston, J., Zhang, P. and Wallis, R.B. (2005) 'Intelligent Kanban: evaluation of a supply chain MAS application using benchmarking', *International Conference on e-Technology, e-Commerce and e-Service*, IEEE, Hong Kong, pp.396–399.

Kim, C-S., Tannock, J., Byrne, M., Farr, R., Cao, B. and Er, M. (2004) *State-of-the-art Review: Techniques to Model the Supply Chain in an Extended Enterprise (VIVACE WP2.5)*, University of Nottingham, Operations Management Division, Nottingham, England.

Kohavi, R. and John, G. (1997) 'Wrappers for feature subset selection', *Artificial Intelligence Journal, Special Issue on Relevance*, Vol. 97, Nos. 1–2, pp.273–324.

Kontogounis, I., Chatzidimitriou, K., Symeonidis, A. and Mitkas, P.A. (2006) 'A robust agent design for dynamic SCM environments', *LNAI*, Springer-Verlag, Vol. 3955, pp.127–136.

Kwon, O., Im, G. and Lee, K.C. (2005) 'MACE-SCM: an effective supply chain decision making approach based on multi-agent and case-based reasoning', *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, IEEE Computer Society, Big Island, Hawaii, pp.82–91.

Landauer, C. and Bellman, K.L. (2002) 'Refactored characteristics of intelligent computing systems', *Proc. Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, NIST, Gaithersburg, MD, pp.359–366.

Moyaux, T., Chaib-Draa, B. and D'Amours, S. (2004) 'Multi-agent simulation of collaborative strategies in a supply chain', *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, IEEE Computer Society, New York, pp.52–59.

Sikonja, M.R. and Kononenko, I. (1997) 'An adaptation of relief for attribute estimation on regression, machine learning', in Fished, D. (Ed.): *Proceedings of 14th International Conference on Machine Learning, ICML'97*, Nashville, TN, pp.296–304.

Simek, B., Albayrak, S. and Korth, A. (2004) 'Reinforcement learning for procurement agents of the factory of the future', *2004 Congress on Evolutionary Computation*, IEEE Press, Portland Marriott Downtown, Portland, Oregon, USA, pp.1331–1337.

Smirnov, A.V., Sheremetov, L.B., Chilov, N. and Cortes, J.R. (2004) 'Soft-computing technologies for configuration of cooperative supply chain', *Applied Soft Computing*, Vol. 4, No. 1, pp.87–107.

Smola, A.J. and Scholkopf, B.A (1998) *Tutorial on Supoprt Vector Regression*, NeuroCOLT2 Technical Report Series – NC2=TR=1998-030, NeuroCOLT group.

Stanfield, J. (2002) 'Agents in supply chain management', *AgentLink News*, Vol. 9, pp.11, 12.

Symeonidis, A.L. and Mitkas, P. (2005) *Agent Intelligence through Data Mining*, Springer Science & Business Media, Inc., New York.

Wang, Y. and Witten, I.H. (1997) 'Induction of model trees for predicting continuous classes', *Proceedings of the Poster Papers of the European Conference on Machine Learning*, Faculty of Informatics and Statistics, University of Economics, Prague, pp.128–137.

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

Wu, J., Ulieru, M., Cobzaru, M. and Norrie, D. (2000) 'Supply chain management systems: state of the art and vision', *9th International Conference on Management of Innovation and Technology*, IEEE Press, Singapore, pp.759–764.

Notes

[1]A MAS that comprises some agents with knowledge models derived from DM, used to improve the agents' intelligence and MAS performance. The prerequisites for such an integration is further explained by Symeonidis and Mitkas (2005).

[2]Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

[3]Evaluates attribute sets by using a learning scheme, e.g., ZeroR.

[4]Allows various search and evaluation methods to be combined in order to select attribute and attribute subsets.

[5]Takes a dataset and outputs a specified fold for cross validation.

[6]Building and using a decision stump.

[7]Fast decision tree learner.