myjournal manuscript No.
(will be inserted by the editor)

KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems*

J. Alcalá-Fdez[1], **L. Sánchez**[2], **S. García**[3], **M.J. del Jesus**[1], **S. Ventura**[4], **J.M. Garrell**[5], **J. Otero**[2], **C. Romero**[4], **J. Bacardit**[6], **V.M. Rivas**[1], **J.C. Fernández**[4], **F. Herrera**[3]

[1] University of Jaén, Department of Computer Science, 23071 Jaén, Spain
e-mail: {jalcala,mjjesus,vrivas}@ujaen.es
[2] University of Oviedo, Department of Computer Science, 33204 Gijón, Spain
e-mail: {luciano,jotero}@uniovi.es
[3] University of Granada, Department of Computer Science and Artificial Intelligence, 18071 Granada, Spain
e-mail: {salvagl,herrera}@decsai.ugr.es
[4] University of Córdoba, Department of Computer Sciences and Numerical Analysis, 14071 Córdoba, Spain
e-mail: {sventura,cromero,i82fecaj}@uco.es
[5] University Ramon Llull, Department of Computer Science, 08022 Barcelona, Spain
e-mail: josepmg@salle.url.edu
[6] University of Nottingham, Department of Computer Science and Information Technology, NG8 1BB Nottingham, UK
e-mail: jqb@cs.nott.ac.uk

Received: date / Revised version: date

Abstract This paper introduces a software tool named *KEEL* which provides a platform for an analysis of evolutionary learning to Data Mining problems: regression, classification, unsupervised learning, etc. It includes evolutionary learning algorithms based on the different approaches, pitt, michigan, IRL, and the integration of evolutionary learning algorithms with different pre-processing techniques, providing a complete analysis of any learning model in comparison with the existing ones.

Key words Computer-Based Education, Data Mining, Evolutionary Computation, Experimental Design, Graphical Programming, Java, Knowledge Extraction, Machine Learning.

1 Introduction

Evolutionary Algorithms (EAs) [11] are optimization algorithms based on natural evolution and genetic processes. Nowadays in Artificial Intelligence (AI), EAs are considered as one of the most successful search techniques for complex problems.

In recent years EAs, particularly Genetic Algorithms (GAs) [14,15], have proved to be an important technique for learning and knowledge extraction. This makes them also a promising tool in Data Mining (DM) [12, 16,52]. The idea of automatically discovering knowledge from databases is a very attractive and challenging task. Hence, there has been a growing interest in DM in several AI-related areas, including EAs. The main motivation for applying EAs to knowledge extraction tasks is that they are robust and adaptive search methods, which perform a global search in the space of candidate solutions (for instance, rules or another form of knowledge representation). The use of EAs in problem solving is a widespread practice. Problems as image retrieval [42], the learning of controllers in robotics [28] or the improvement of e-learning systems [39] show their suitability as problem solvers in a wide range of scientific fields.

Although EAs are powerful for solving a wide range of scientific problems, their use requires certain programming expertise along with considerable time and effort in order to write a computer program for implementing the often sophisticated algorithm according to user's needs. This work can be tedious and needs to be done before users can start on which they should be really working on. Given this situation, the aim of this paper is to introduce a non-commercial Java software tool, named *KEEL* (Knowledge Extraction based on Evolutionary Learning)[1], that allows the user to analyze the behaviour of evolutionary learning for different kinds of DM problems: regression, classification, unsupervised learning, etc.

This tool can offer several advantages. First of all, it reduces programming work. It includes a library with

* Supported by the Spanish Ministry of Science and Technology under Projects TIN-2005-08386-C05-(01, 02, 03, 04 and 05).

[1] http://www.keel.es

evolutionary learning algorithms based on the different approaches (pitt, michigan and IRL) and easies the integration of evolutionary learning algorithms with different pre-processing techniques. It can alleviate researchers from the mere "technical work" of programming and thus they are able to focus more on the analysis of their new learning models in comparison with the existing ones. Second, it extends the range of possible users applying evolutionary learning algorithms. A wide library of EAs and an easy to use software considerably reduces the level of knowledge and experience required by researchers in evolutionary computations. As a result, researchers with less knowledge in this framework would be also able to apply these algorithms on their problems. Third, using a strictly object-oriented approach for the library and the software tool these can be used on any machine with Java. As a result, any researcher can use KEEL on his machine, independently of the operating system.

The paper is arranged as follows. The next section introduces a study on some non-commercial DM softwares and the main benefits that the KEEL software tool offers with respect to the remaining software tools. Section 3 presents some aspects of KEEL, its main features and the modules that comprise it. In Section 4, two examples are given to illustrate how KEEL should be used. Finally, Section 5 points out some conclusions and future works.

2 A Study on some Non-Commercial Data Mining Software

A search on the Internet about DM software reveals that a lot of commercial and non-commercial DM tools and libraries exists in development on the whole scientific community. We recommend visiting the directories of software KDnuggets [2] and The-Data-Mine [3] for an overall look of the most of them. Although a lot of them are for commercial purposes, a few are available on an open source basis. Among the commercial software are leading mining suites such as SPSS Clementine [4], Oracle Data Mining [5] and KnowledgeSTUDIO [6].

We can distinguish between libraries whose purpose is to develop new EAs for specific applications and DM suites that incorporate learning algorithms (some of them may use EAs for this task) and provide a mechanism to establish comparisons among them with researching objectives. Over the Internet and in specialized literature, we can find a large number of libraries dedicated to evolutionary computation. As generic tools in which it is possible to develop different EAs for different problems, we must remark *ECJ* [23], *EO* [17], *Evolvica* [40], *JCLEC* [46] and *Open Beagle* [13]. We can also find some libraries designed for a concrete type of EA: genetic algorithms [8], genetic programming [32], memetic algorithms [19], learning classifier systems [25], evolutionary multiobjective optimization [43] and distributed EAs [44].

Nowadays, more researches base on DM tools [38] or they employ tools specifically designed for an area of DM, see for example [48]. Our interest is focused on free distributions of software dedicated to an overall point of view of DM, in which developers may choose to extend the functionality since the source code is available. Probably the most well-known open source DM package is Weka [51], a collection of Java implementations of ML algorithms. Others packages different from Weka are also available in open source.

The objective of this section is to present a survey of all of them, to summarize their main strong points and to introduce why we have designed KEEL and which are its main benefits.

2.1 Non-Commercial Suites

In this section, we list the open source DM software tools that deserves the mention due to the acknowledgement, qualities or acceptation they have.

- *ADaM* [41]: This toolkit is packaged as a suite of independent components in order to employ them working in grid or cluster environments. It provides feature selection capabilities, image processing and data cleaning.
- *D2K (with E2K) [20]*: Data to knowledge toolkit provides a visual programming environment and a set of templates to connect with other standard packages. It incorporates external packages to perform image and text mining. D2K software tool offers an external set of evolutionary mechanisms used for developing basic GAs (E2K).
- *KNIME* [4]: This modular environment enables easy integration of new algorithms, data manipulation and visualization methods as models. It is compatible with Weka and it also supports statistics via integrated R [35].
- *MiningMart* [27]: It is developed with the purpose of re-using best-practice cases of pre-processing data stored in very large databases. MiningMart is focused on the pre-processing chain, not on the possible steps of DM.
- *Orange* [9]: It is a library of core objects and routines that includes a large variety of standard and not-so-standard ML and DM algorithms, in addition to routines for data input and manipulation. It is also a scriptable environment for prototyping new algorithms and testing schemes built in Python.

[2] http://www.kdnuggets.com/software
[3] http://the-data-mine.com/bin/view/Software
[4] http://www.spss.com/clementine
[5] http://www.oracle.com/technology/products/bi/odm
[6] http://www.angoss.com/products/studio/index.php

– *Tanagra* [36]: Tanagra is a DM software for educational and research purposes. It proposes several data mining methods from exploratory data analysis, statistical learning, machine learning and databases area.
– *Weka* [51]: Weka is the best well-known software tool to perform ML and DM tasks. The algorithms that it includes can either be applied directly to a data set or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Given that it is practically the standard in DM suites, it is also related with a complete set of extra packages for completing some aspects.
– *YALE* [26]: It is a free open-source environment for KDD and ML that provides a rich variety of methods which allow prototyping new applications and makes costly re-implementations unnecessary.

All these software tools are provided by several functionalities, but each one supports them in a different way. In the following subsection we analyze how these software tools tackle a defined set of basic and advanced functionalities.

2.2 Study based on Functionality

Once seen some of the available DM software tools, we propose to analyze them by following a set of functionality criteria. We do not want to establish a comparison among all software tools or emphasize the improvement of one with respect to another. Our intention is focused on pointing out the main strengths or weakness of each software, in order to compile a set of characteristics in which the existing software tools have not an advanced functionality.

In order to do that, we have established a set of basic and advanced characteristics that the suites could have or not. The idea implies to detect the mayor differences among the software tools and, principally, categorize KEEL as an alternative for the researches with others requirements. Table 1 shows a summary of the characteristics pointed out. All of them have been selected while evaluating all the software tools, tutorials and guidelines for their use. The only one that we have add is the *EAs* integration, given that the main motivation of KEEL is just it. We distinguish four levels of support in some characteristics: none (N), basic support (B), intermediate support (I) and advanced support (A). If the support can not be understood by levels, the notation used is Yes (Y) for supporting and No (N) for no supporting.

In the following, we briefly explain all the issues of the Table 1:

– *Language* is the programming language used in the development of the software. C++ language is less portable with respect to Java.
– *Graphical Interface* includes functionality criteria which can be managed through a handy interface by the user.
  – *Graph representation* indicates that the experiments or knowledge flows are represented via graphs with node-edge connections. This alternative is more interpretable and user-friendly than using a chain of processes or a tree representation of modules.
  – *Data visualization* includes tools for representing the data sets through charts, tables or similar mechanisms.
  – *Data management* comprises a set of toolkits that allow us to perform basic manual operations with the data, such as removing or modifying rows, columns, etc.
– *Input / Output* functionality criteria point out the different data formats supported, such as *ARFF* (standard of Weka), *others* (including C4.5 input .names standard [34], .xls, .csv, XML) and *database connection.* The support is given if the environment can load or save data in these formats or can transforms them into a standard one used by it.
– *Preprocessing Variety.* It comprises *discretization* [22], *feature selection* [29], *instance selection* [50] and *missing values imputation* [2]. The trend of most of the suites is to offer a good feature selection and discretization set of methods, but they overlook specialized methods of missing values imputation and instance selection. Usually, the contributions included are basic modules of replacing or generating null values and methods for sampling the data sets by randomly (stratified or not) or by value-dependently.
– *Learning Variety.* It is the support over main areas of DM, such as predictive tasks (*classification*, *regression*,
anomaly/deviation detection), and descriptive tasks (*clustering*, *association rule discovery* ,*sequential pattern discovery*) [45]. The existence of the classical models determines to achieve an intermediate level, and the existence of advanced models indicates the advanced level in the area.
– *Off/On-line run* of the experiment created. An On-line run implies that the environment and algorithms modules need to be in the same machine and the experiments are completely dependent of the platform. An off-line run entails the independency of the experiments created with respect to the environment, allowing the experiment to be executed in other machines.
– *Advanced Features* includes some of the non-common criteria incorporated for extending the functionality of the software tool.
  – *Postprocessing*, usually for tuning the learned model for the algorithm.

Table 1 Summary of the characteristics of each DM software tool

Software	Language	Graphical Interface			Input / Output			Preprocessing Variety				Learning Variety				Run Types		Advanced Features			
		Graph representation	Data visualization	Data management	ARFF data format	Other data formats	Data Base connection	Discretization	Feature Selection	Instance Selection	Missing values imputation	Classification	Regression	Clustering	Association Rules	On-line run	Off-line run	Postprocessing	Meta-Learning	Statistical tests	Evolutionary Algorithms
ADaM	C++	N	N	I	Y	N	N	N	A	B	N	I	N	A	B	Y	N	N	N	N	B
D2K	Java	Y	A	I	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	N	N	I
KNIME	Java	Y	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	N	I	B
MiningMart	Java	Y	B	A	N	N	Y	I	A	B	I	B	B	N	N	Y	N	N	N	N	B
Orange	C++	Y	A	A	N	Y	N	A	I	B	B	I	N	I	I	N	Y	N	N	N	N
Tanagra	C++	N	A	A	Y	Y	N	B	A	B	N	A	I	A	A	Y	N	N	I	A	N
Weka	Java	Y	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	N	N	B
YALE	Java	N	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	A	B	I

- *Meta-learning*, that includes more advanced learning schemes, such as bagging or boosting, or meta learning of the parameters of the algorithms.
- *Statistical tests* for establishing comparisons among the results obtained. An advanced support of this property requires a complete set of parametric and non-parametric statistical tests; a basic support implies the existence of the well-known statistical tests (such as t-test).
- *Evolutionary models* support points out the integration of EAs into the DM areas that the software tool offers. A basic support of this feature implies the use of genetic algorithms in some techniques (usually, genetic feature selection). To upgrade the level is necessary to incorporate EAs in learning or meta-learning models.

Analyzing the characteristics presented in Table 1 we can highlight that most of software tools have a basic support for EAs. Moreover, the software tools studied usually integrate a representative set of algorithms for each type of learning and preprocessing task but the experiments are thinking for being run on the same environment, which is not useful when the algorithms require high computation times (such as happens with the EAs).

From our point of view users need a software tool where they can analyze the behaviour of evolutionary and non evolutionary algorithms for different kinds of DM problems and run their experiments in both modes (off-line and on-line). Based on these requirements we have developed the KEEL software tool. In the next section we will describe KEEL in detail.

3 KEEL

Essentially, KEEL is a software tool which allows the integration of evolutionary models in the different areas of learning and preprocessing tasks, making easy to the user the management of these techniques. The models correspond with the most well-known and performed models in each methodology, such as evolutionary feature and instance selection [5,21], evolutionary fuzzy rule learning and Mamdani rule tuning [1,31], genetic artificial neural networks [24,37], Learning Classifier Systems [3,49], etc.

The presently available version of KEEL consists of the following function blocks[7]:

- *Data Management*: This part is composed by a set of tools that can be used to build new data, export and import data in other formats to KEEL format, data edition and visualization, apply transformations and partitioning to data, etc...
- *Design of Experiments*: The aim of this part is the design of the desired experimentation over the selected data sets. It provides options for many choices: type of validation, type of learning (classification, regression, unsupervised learning), etc...
- *Educational Experiments*: With a similar structure to the previous part, allows us to design an experiment which can be step-by-step debugged in order to use this as a guideline to show the learning process of a certain model by using the platform with educational objectives.

[7] http://www.keel.es/software/prototypes/version1.0/ /ManualKeel.pdf

Taking into account each one of the function blocks, KEEL can be useful by different types of user, which expect to find determined features in a DM software.

In the following, we describe the user profiles who it is designed for (Subsection 3.1), its main features (Subsection 3.2) and the different ways of working integrated in the software tool. They include the data management functionality, which is explained in Subsection 3.3 and the details of the creation of experiments in off-line and on-line mode, described in Subsections 3.4 and 3.5, respectively.

3.1 User Profiles

Essentially, KEEL is an integration of an environment with a defined architecture and a development of knowledge extraction as expandable modules. It is mainly intended for two categories of users: researchers and students. Either group has a different set of needs:

- *KEEL as a research tool*: The most common use of this tool for a researcher will be the automated execution of experiments, and the statistical analysis of their results. Routinely, an experimental design includes a mix of evolutionary algorithms, statistical and AI-related techniques. Special care was taken to make possible that a researcher can use KEEL to assess the relevance of his own procedures. Since the actual standards in machine learning require heavy computational work, the research tool is not designed to offer a real-time view of the progress of the algorithms, it is designed to rather generate a script and be batch-executed in a cluster of computers. The tool allows the researcher to apply the same sequence of pre-processing, experiments and analysis to large batteries of problems and focus his attention in the summary of the results.
- *KEEL as an educational tool*: The needs of a student are quite different to those of a researcher. Generally speaking, the objective is no longer that of making statistically sound comparisons between algorithms. There is no need of repeating each experiment a large number of times. If the tool is to be used in class, the execution time must be short and a real-time view of the evolution of the algorithms is needed, since the student will use this information to learn how to adjust the parameters of the algorithms. In this sense, the educational tool is a simplified version of the research tool, where only the most relevant algorithms are available. The execution is made in real time. The user has a visual feedback of the progress of the algorithms, and can access the final results from the same interface used to design the experimentation.

Both types of user require an availability of a set of features in order to be interested in using KEEL. Then, this is when we describe the main features of the KEEL software tool.

3.2 Main Features

KEEL is a software tool developed to ensemble and use different DM models. We would like to remark that this is the first software toolkit of this type containing a library of evolutionary learning algorithms with open source code in Java. The main features of KEEL are:

- EAs are presented in predicting models, pre-processing (evolutionary feature and instance selection) and post-processing (evolutionary tuning of fuzzy rules).
- It includes data pre-processing algorithms proposed in specialized literature: data transformation, discretization, instance selection and feature selection.
- It has a statistical library to analyze algorithms' results. It comprises a set of statistical tests for analyzing the normality and heteroscedasticity of the results and performing parametric and non-parametric comparisons among the algorithms.
- Some algorithms have been developed by using a Java Class Library for Evolutionary Computation (JCLEC)[8] [46].
- It provides an user-friendly interface, oriented to the analysis of algorithms.
- The software is aimed to create experimentations containing multiple data sets and algorithms connected among themselves to obtain a result expected. Experiments are independently script-generated from the user interface for an off-line run in the same or other machines.
- KEEL also allows to create experiments in on-line mode, aiming an educational support in order to learn the operation of the algorithms included.
- It contains a Knowledge Extraction Algorithms Library[9], remarking the incorporation of multiple evolutionary learning algorithms, together with classical learning approaches. The main employment lines are:
  - *Different evolutionary rule learning models have been implemented.*
  - *Fuzzy systems.* Fuzzy rule learning models with a good trade-off between accuracy and interpretability.
  - *Evolutionary neural networks.* Evolution and pruning in neural networks, product unit neural networks, and radial base models.
  - *Genetic programming.* Evolutionary algorithms that use tree representations for extracting knowledge.
  - *Subgroup discovery.* Algorithms for extracting descriptive rules based on patterns subgroup discovery have been integrated.
  - *Data reduction (instance and feature selection and discretization).* EAs for data reduction have been included.

[8] http://jclec.sourceforge.net/

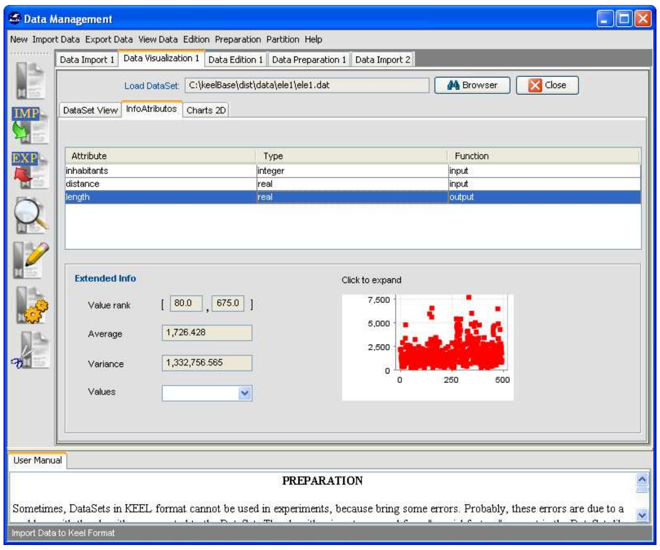
[9] http://www.keel.es/software/prototypes/version1.0/ /AlgorithmsList.pdf

KEEL integrates the library of algorithms in each one of the function blocks that it is composed. We have briefly presented its function blocks before, but in the following subsections, we will describe the possibilities that KEEL can offer us in relation with data management, off-line experiment design and on-line educational design.

3.3 Data Management

The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed, or made more easily accessible [33]. Data preparation comprises those techniques concerned with analyzing raw data so as to yield quality data, mainly including data collecting, data integration, data transformation, data cleaning, data reduction and data discretization [53]. Data preparation can be more time consuming than data mining, and can present equal challenges to data mining. Its importance lies in that the real-world data is impure (incomplete, noisy and inconsistent data), high-performance mining systems require quality data (removing anomalies or duplications) and quality data yields high-quality patterns (to recover missing data, to purify data and to resolve data conflicts).

Data Management module integrated in KEEL allows us to perform the data preparation stage independently of the remaining of the DM process. This module is focused on the group of users denoted domain experts. They are familiar with their data, they know the processes that produce the data and they are interested in reviewing these processes for improving or understanding them. In short, domain users are those whose interest lies in applying processes to data of their own and they usually are not experts in DM.



Load DataSet: C:\keelBase\dist\data\ele1\ele1.dat

Attribute	Type	Function
inhabitants	integer	input
distance	real	input
length	real	output

Value rank [80.0 , 675.0]
Average 1,726.428
Variance 1,332,756.565
Values

PREPARATION

Sometimes, DataSets in KEEL format cannot be used in experiments, because bring some errors. Probably, these errors are due to a

Fig. 1 Data Management

Figure 1 shows an example window of the *Data Management* module in the section of *Data Visualization*. The module has seven sections, each one is accessible through the buttons on the left side of the window. In the following, we will briefly describe them:

- *Creation of a new data set*: This option allows us to generate a new data set compatible with the other KEEL modules.
- *Import data to KEEL format*: Since KEEL works with a specific data format (alike the ARFF format) in all its modules, this section allows us to convert various data format to KEEL format, such as CSV, XML, ARFF, connections with data bases, etc.
- *Export data from KEEL format*: It is the opposite option to the previous. It converts the data treated with KEEL procedures in other external formats to establish a compatibility with other software tools.
- *Visualization of data*: It is the option used for representing and visualizing the data. With it, we can see a graphical distribution of each attribute and comparisons between two attributes.
- *Edition of data*: This area is dedicated to manage the data manually. The data set, once loaded, can be edited in terms of modifying values, adding or removing rows and columns, etc.
- *Data Partition*: Given that the experiments modules of KEEL work with partitions of data in order to validate the results, this zone allows us to make them. It supports k-fold cross validation, 5x2 cross validation and hold-out validation with stratified partition.
- *Data Preparation*: It is the section that allows us to perform automatic data preparation for DM, including cleaning, transformation and reduction of data. All techniques integrated in this section are also available in the experiments modules of KEEL.

3.4 Design of Experiments: Off-Line Module

In the last few years, a large number of DM software tools have been developed with researching objectives. Some of them are libraries which allow to reduce programming work of new algorithms: ECJ [23], JCLEC [46], learning classifier systems [25], etc. Others are DM suites that incorporate learning algorithms (some of them may use EAs for this task) and provide a mechanism to establish comparisons among them. Some examples are Weka [51], D2K [20], etc.

This module is a Graphical User Interface (GUI) that allows to design experiments for solving different problems of regression, classification and unsupervised learning. Its main aim is to obtain the directory structure and files required for running the designed experiments in any local machine with Java (see Figure 2).

The experiments are graphically modelled based on data flow, where they are represented by graphs with node-edge connections. To design an experiment, first we

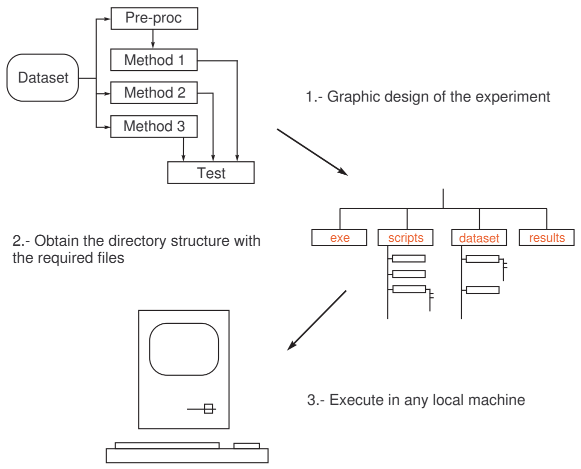
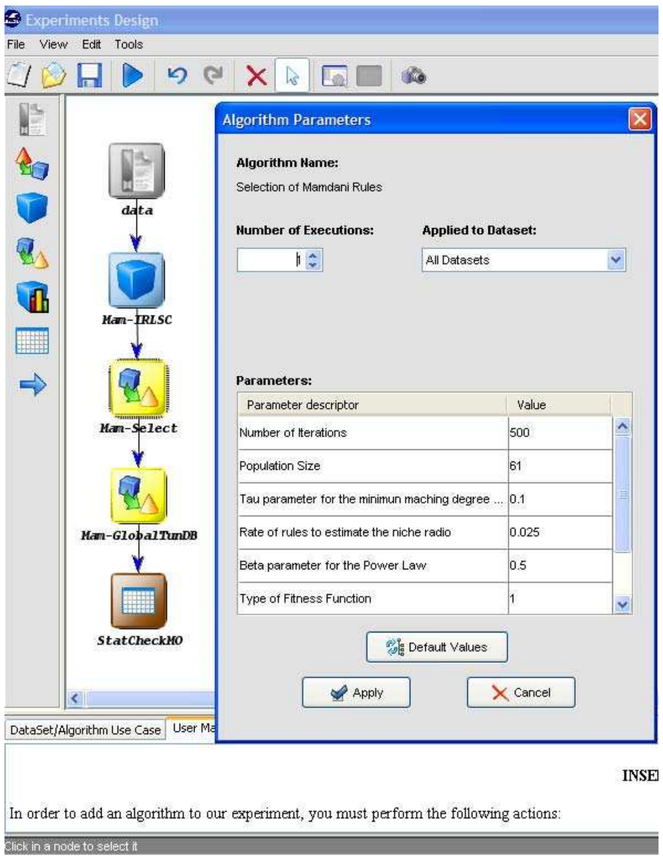

Pre-proc
Dataset
Method 1
Method 2
Method 3
Test
1.- Graphic design of the experiment
2.- Obtain the directory structure with the required files
exe
scripts
dataset
results
3.- Execute in any local machine

Fig. 2 Design of experiments

have to indicate the type of validation (k-fold cross validation [18] or 5x2 cross validation [10]) and the type of learning (regression, classification or unsupervised learning) we want to use. Then, we have to select the data sources, put the methods in the workspace and connect it, combining the evolutionary learning algorithms with different pre-processing and post-processing techniques if needed. Finally, we can add statistical tests to achieve a complete analysis of the studied methods and a box to obtain a summary of the results. Notice that each component of the experiment is configured in separate dialogues that can be opened by double-clicking the respective graph. Figure 3 shows an example of an experiment following the MOGUL methodology [6] and with a box to obtain a summary of the results. In this Figure is also shown the configuration window of one of the used post-processing methods.

When the experiment has been designed, the user can choose either to save the design in a XML file or to obtain a zip file. If the user chooses to obtain a zip file, then the system will generate a zip file with the directory structure and the required files for running the designed experiment in any local machine with Java. This directory structure contains the data sources, the jar files of the algorithms, the configuration files in XML format, a script file with all the indicated algorithms in XML format and a Java tool, named *RunKeel*, to run the experiment. *RunKeel* can be seen as a simple EA scripting environment which reads the script file in XML format, runs all the indicated algorithms and saves the results in one or several report files.

Obviously, this kind of interface is ideal for experts of specific areas who know the methodologies and methods used in their interest area and with the intention of proposing a new method and to compare it with the well-known methods available in KEEL.



Algorithm Parameters

Algorithm Name:
Selection of Mamdani Rules

Number of Executions: 1

Applied to Dataset: All Datasets

Parameters:

Parameter descriptor	Value
Number of Iterations	500
Population Size	61
Tau parameter for the minimun maching degree ...	0.1
Rate of rules to estimate the niche radio	0.025
Beta parameter for the Power Law	0.5
Type of Fitness Function	1

In order to add an algorithm to our experiment, you must perform the following actions:

Click in a node to select it

Fig. 3 Example of an experiment and the configuration window of a method.

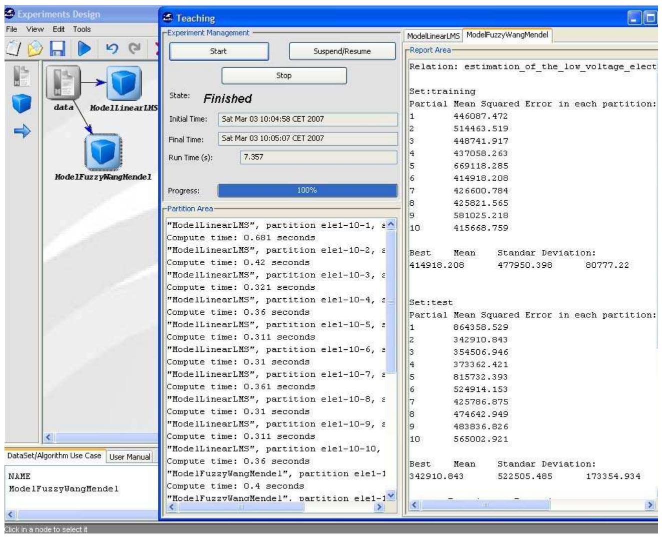
3.5 Computer-Based Education: On-Line Module

There is a variety of terms used to describe the educational use of computer [30]. Computer-assisted instruction (CAI), computer-based education (CBE) and computer-based instruction (CBI) are the broadest terms and can refer to virtually any kind of computer use in educational settings. These terms may refer either to stand-alone computer learning activities or to computer activities which reinforce material introduced and taught by teachers.

Most of the software developed in DM and evolutionary computation domain is designed for research purposes (libraries, algorithms, specific applications, etc.). But there are some free softwares that are designed not only for research purposes but also for educational purposes. These systems are easy to use due to the fact that they provide a GUI oriented to help in the user interaction with the system for doing all the tasks (to select data, to choose parameters, to run algorithms, to visualize the results, etc). Some examples of open source DM systems are Weka [51], Yale [26] and Tanagra [36].

This module is a GUI that allows the user to design an experiment (with one or more algorithms), to run it and to visualize the results on-line. The idea is to use it as a guideline to show the learning process of a certain model. This module has a similar structure as the previous one but only including the more adequate algorithms and options for academic objectives.

When an experiment is designed the user can choose either to save the experiment in a XML file or to run it. If the user chooses to run it, then the system will show an auxiliary window which will allow to manage and to visualize the running of each algorithm. When the run finishes, this window will show the results obtained for each algorithm in independent tags, showing for example the confusion matrices for classification or the mean square errors for regression problems (see Figure 4).



```
NAME
ModelFuzzyWangMendel
```

State: *Finished*

Initial Time: Sat Mar 03 10:04:58 CET 2007

Final Time: Sat Mar 03 10:05:07 CET 2007

Run Time (s): 7.357

```
"ModelLinearLMS", partition ele1-10-1, s
Compute time: 0.681 seconds
"ModelLinearLMS", partition ele1-10-2, s
Compute time: 0.42 seconds
"ModelLinearLMS", partition ele1-10-3, s
Compute time: 0.321 seconds
"ModelLinearLMS", partition ele1-10-4, s
Compute time: 0.36 seconds
"ModelLinearLMS", partition ele1-10-5, s
Compute time: 0.311 seconds
"ModelLinearLMS", partition ele1-10-6, s
Compute time: 0.31 seconds
"ModelLinearLMS", partition ele1-10-7, s
Compute time: 0.361 seconds
"ModelLinearLMS", partition ele1-10-8, s
Compute time: 0.31 seconds
"ModelLinearLMS", partition ele1-10-9, s
Compute time: 0.311 seconds
"ModelLinearLMS", partition ele1-10-10,
Compute time: 0.36 seconds
"ModelFuzzyWangMendel", partition ele1-1
Compute time: 0.4 seconds
"ModelFuzzyWangMendel", partition ele1-1
```

```
Relation: estimation_of_the_low_voltage_elect

Set:training
Partial Mean Squared Error in each partition:
1       446087.472
2       514463.519
3       448741.917
4       437058.263
5       669118.285
6       414918.208
7       426600.784
8       425821.565
9       581025.218
10      415668.759

Best    Mean    Standar Deviation:
414918.208      477950.398      80777.22


Set:test
Partial Mean Squared Error in each partition:
1       864358.529
2       342910.843
3       354506.946
4       373362.421
5       815732.393
6       524914.153
7       425786.875
8       474642.949
9       483836.826
10      565002.921

Best    Mean    Standar Deviation:
342910.843      522505.485      173354.934
```

Fig. 4 Auxiliary window of an experiment with two algorithms.

4 Case Studies

This section presents two case studies as examples for illustrating the functionality and the process of creating experiment in the KEEL software tool. The first case study is focused in the development of a comparison of some algorithms and in a later analysis of the results, by using the off-line module. The second example is a presentation of the educational on-line module.

4.1 Off-Line Case Study

Our purpose in this example is to establish a comparison of three methods that belong to different ML techniques and use EAs in the learning task. The experiment graph is represented in Figure 5. In this figure, we have used a k-Nearest Neighbour classifier with a previous pre-processing stage of prototype selection guided by a CHC model (EIS-CHC + ClasifKNN) [5]. We have also used a XCS classifier [49] and an Evolutionary Product Unit based Neural Networks (NNEP) [24].

By clicking the *Experiment* option in the main menu of the KEEL software tool, we define the experiment as

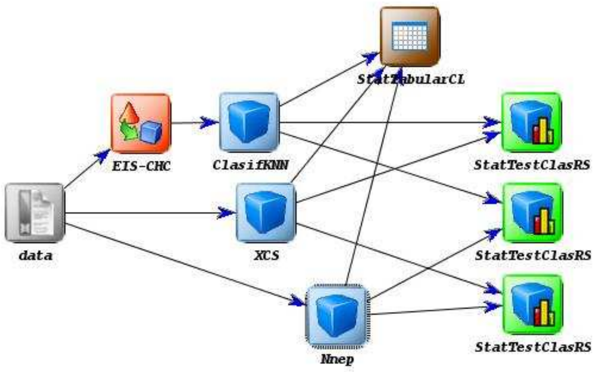


Fig. 5 Experiment graph of the off-line example.

Classification problem and we use a *10-fold cross validation* procedure to analyze the results. Then, we have to choose the data sets for beginning the setup of the experiment graph. This example uses the *Iris* data set, but more of one could be chosen at the same time.

The graph of the Figure 5 represents the flow of data and results among the algorithms and procedures. A node can represent an initial data flow (data set), a preprocess/post-process algorithm, learning methods and tests or visualization of results modules. They can be easily distinguished according to the color of the node. All their parameters can be adjusted by clicking twice on the node. Logically, directed edges connecting two nodes imply a relation between them (data or results interchange). When the data is interchanged, the flow includes pairs of train-test data sets in classification and regression problems. By taking account of this explanation, the graph describes a flow of data from the *Iris* data set to three nodes, two of them are learning methods. At the top of them, the flow of data is the input of a pre-process method, whose operation consist of reducing the training data by removing instances and the resulted subset is used as reference set of the later k-NN classifier. XCS and NNEP will use the complete training data for learning the model.

From now, all the models would be learned and the data would be classified according to the training and test data. These results are the inputs of the visualization and the tests modules. The module *StatTabularCL* gets as inputs these results and generates output files with the results obtained, such as confusion matrixes for each method, accuracy and error percentages for each method, fold and class, and a summary of final results. On the other hand, the graph represents other way of results flow, interconnecting each two methods with a test module. In this case, the test module used is the signed-rank Wilcoxon non-parametrical procedure *StatTestClasRS* for comparing two samples of results. The experiment establishes a pair-wise statistical comparison among the three methods.

```
----------------------------------------------------------
CONFUSION MATRIX. ALGORITHM: ClasifKNN
----------------------------------------------------------

TEST RESULTS
,iris-setosa,iris-versicolor,iris-virginica
iris-setosa,50,0,0
iris-versicolor,0,46,4
iris-virginica,0,3,47
TRAIN RESULTS
,iris-setosa,iris-versicolor,iris-virginica
iris-setosa,440,10,0
iris-versicolor,0,433,17
iris-virginica,0,11,439
```

Fig. 6 Confusion Matrix obtained for EIS-CHC + ClasifKNN

Once the graph is described, we can create the experiment associated and save it as zip file for an off-line run. Following the structure of directories shown in Figure 2, the experiment is created as a set of XML scripts and a jar program for running it. Within the *results* directory, there will be directories used for housing the results of each methods during the run. Whether the method is a learning method, its associated directory will house the model learned or if is a test/visualization procedure, its directory will house the results files. The experiment has been run and we can find the result file of the confusion matrix (see Figure 6 for the confusion matrix of the EIS-CHC + ClasifKNN classifier) or the one associated with a Wilcoxon comparison (Figure 7).

```
Wilcoxon signed rank test, Classification
Classification error in each fold:
Algorithm = 0
Fold 0 : 0.06666666666666667
Fold 1 : 0.06666666666666667
Fold 2 : 0.06666666666666667
Fold 3 : 0.06666666666666667
Fold 4 : 0.06666666666666667
Fold 5 : 0.0
Fold 6 : 0.0
Fold 7 : 0.13333333333333333
Fold 8 : 0.06666666666666667
Fold 9 : 0.13333333333333333
Algorithm = 1
Fold 0 : 0.0
Fold 1 : 0.0
Fold 2 : 0.0
Fold 3 : 0.06666666666666667
Fold 4 : 0.0
Fold 5 : 0.0
Fold 6 : 0.06666666666666667
Fold 7 : 0.13333333333333333
Fold 8 : 0.0
Fold 9 : 0.06666666666666667
Null hypothesis, true difference in means is equal to 0
Output=0: There is no evidence against H0
p-value: 0.09934224785065712
```

Fig. 7 Results for signed-rank Wilcoxon test comparing XCS with NNEP.

With a simple design of a logical flow of data by means of a graph representation, an user can create an experiment involving several data sets, interconnect pre-processing tasks with learning tasks, integrate and configure powerful learning models with classical ones, compile the results obtained establishing statistical comparisons among them and finally run all this process in an independent machine with the only requirement of having JAVA Virtual Machine installed in it.

4.2 On-Line Case Study

An example of an educational experience is shown in Figure 8. Our propose is to observe the learning process of a regression algorithm of fuzzy rule learning [47] over a *electrical energy distribution problem* [7] by using five labels per variable.

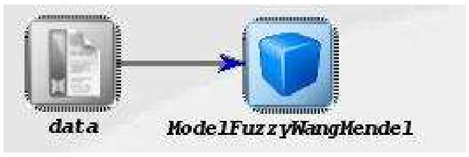
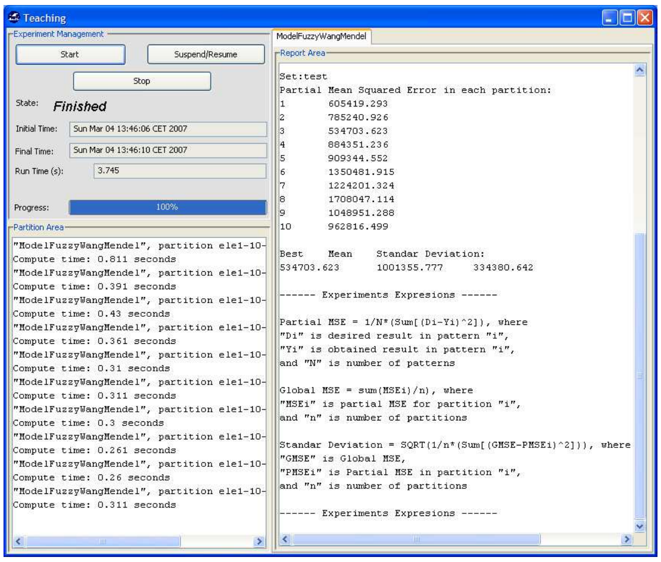


Fig. 8 Experiment graph of the on-line example.

The run of the experiment is performed in a new window (see Figure 9). The user/teacher can start, suspend/pause and stop the execution of an experiment at any moment in order to be able to see step by step partial reports of the execution. The final information that provides after the run finishes is the run-time, accuracy in each fold and average accuracy and a brief description of the used function to evaluate the accuracy of the solutions.



State: Finished
Initial Time: Sun Mar 04 13:46:06 CET 2007
Final Time: Sun Mar 04 13:46:10 CET 2007
Run Time (s): 3.745
Progress: 100%

```
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.811 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.391 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.43 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.361 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.31 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.311 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.3 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.261 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.26 seconds
"ModelFuzzyWangMendel", partition ele1-10-
Compute time: 0.311 seconds
```

```
Set:test
Partial Mean Squared Error in each partition:
1       605419.293
2       785240.926
3       534703.623
4       884351.236
5       909344.552
6       1350481.915
7       1224201.324
8       1708047.114
9       1048951.288
10      962816.499

Best     Mean     Standar Deviation:
534703.623      1001355.777     334380.642

------ Experiments Expresions ------

Partial MSE = 1/N*(Sum[(Di-Yi)^2]), where
"Di" is desired result in pattern "i",
"Yi" is obtained result in pattern "i",
and "N" is number of patterns

Global MSE = sum(MSEi)/n), where
"MSEi" is partial MSE for partition "i",
and "n" is number of partitions

Standar Deviation = SQRT(1/n*(Sum[(GMSE-PMSEi)^2])), where
"GMSE" is Global MSE,
"PMSEi" is Partial MSE in partition "i",
and "n" is number of partitions

------ Experiments Expresions ------
```

Fig. 9 Window of results obtained in the experiment.

This GUI is ideal for students who can interpret the results and know how to change the parameters in order to improve the results. In this example, they can prove how the Wand and Mendel's ad-hoc method [47] is very fast but the results are not very accuracies.

5 Conclusion and Future Work

In this paper, we have described a non-commercial Java software tool, named KEEL, which provides a platform for an analysis of evolutionary learning to Data Mining problems. This tool relieves researchers from the technical work and favours them to focus on the analysis of their new learning models in comparison with the existing ones. Moreover, this makes is easier for researchers with lower level of knowledge in evolutionary to apply evolutionary learning algorithms on their problems.

We have shown the main features of this software tool and we have distinguished three main parts: a module for data management, a module for designing experiments with evolutionary learning algorithms, and a module with educational objectives. We have also shown two case studies as examples for illustrating the functionality and the process of the creation of an experiment in the KEEL software tool.

The KEEL software tool is continuously updating and improving. At the moment, we are developing a new set of evolutionary learning algorithms and a test tool which will allow us to apply parametric and non-parametric tests on any set of data. We are also developing data visualization tools in the on-line and off-line modules and a graphical tool to run in a distributed way the experiments designed with the off-line module. Finally, we are also working on the development of a data set repository including the data set partitions and the algorithm results on these data sets, *KEEL-dataset*[10].

[10] http://www.keel.es/datasets.php

References

1. R. Alcalá, J. Alcala-Fdez, J. Casillas, O. Cordón and F. Herrera, Hybrid learning models to get the interpretabilityaccuracy trade-off in fuzzy modeling, Soft Computing **10:9,** (2006) 717-734.
2. G.E. Batista and M.C. Monard, An analysis of four missing data treatment methods for supervised learning, Applied Artificial Intelligence **17,** (2003) 519-533.
3. E. Bernadó-Mansilla and T.K. Ho. Domain of Competence of XCS Classifier System in Complexity Measurement Space, IEEE Transactions on Evolutionary Computation **9:1,** (2005) 82–104.
4. M.R. Berthold, N. Cebron, F. Dill, G. Di Fatta, T.R. Gabriel, F. Georg, T. Meinl and P. Ohl, KNIME: The Konstanz Information Miner, In: Proc. of the 4th Annual Industrial Simulation Conference, Workshop on Multi-Agent Systems and Simulations, Palermo, (2006).
5. J.R. Cano, F. Herrera and M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study, IEEE Transactions on Evolutionary Computation **7:6,** (2003) 561–575.
6. O. Cordón, M.J. del Jesus, F. Herrera and M. Lozano, MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach, International Journal of Intelligent Systems **14:9,** (1999) 1123–1153.
7. O. Cordón, F. Herrera, and L. Sánchez, Solving electrical distribution problems using hybrid evolutionary data analysis techniques, Applied Intelligence **10,** (1999) 5-24.
8. A. S. Chuang, An extendible genetic algorithm framework for problem solving in a common environment, IEEE Transactions on Power Systems **15:1**, (2000) 269–275.
9. J. Demšar and B. Zupan, Orange: From experimental machine learning to interactive data mining, White Paper (http://www.ailab.si/orange), Faculty of Computer and Information Science, University of Ljubljana.
10. T.G.Dietterich, Approximate Statistica Tests for Comparing Supervised Classification Learning Algorithms, Neural Computation **10:7,** (1998) 1895–1924.
11. A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing* (Springer-Verlag, Berlin 2003) 299.
12. A.A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms* (Springer-Verlag, Berlin 2002) 264.
13. C. Gagné and M. Parizeau, Genericity in evolutionary computation sofyware tools: Principles and case-study, International Journal on Artificial Intelligence Tools **15:2**, (2006) 173–194.
14. D.E. Goldberg, *Genetic algorithms in search, optimization, and machine learning* (Addison-Wesley, New York 1989) 372.
15. J.H. Holland, *Adaptation in natural and artificial systems* (The University of Michigan Press, London 1975) 228.
16. L.C. Jain and A. Ghosh, *Evolutionary Computation in Data Mining* (Springer-Verlag, New York 2005) 264.
17. M. Keijzer, J. J. Merelo, G. Romero and M. Schoenauer, Evolving objects: A general purpose evolutionary computation library, In: P. Collet, C. Fonlupt, J. K. Hao, E. Lutton, M. Schoenauer (eds), *Artificial Evolution: Selected Papers from the 5th European Conference on Artificial Evolution* (London, UK, 2001) 231–244.
18. R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proc. of the 14th International Joint Conference on Artificial Intelligence **2:12,** (1995) 1137–1143.
19. N. Krasnogor and J. Smith, MAFRA: A Java memetic algorithms framework, In Proc. of the Genetic and Evolutionary Computation Workshops, Las Vegas, Nevada, USA (2000) 125–131.
20. X. Llorà, E2K: Evolution to Knowledge, SIGEVOlution **1:3,** (2006) 10–16.
21. X. Llorà and J.M. Garrell, Prototype induction and attribute selection via evolutionary algorithms, Intelligent Data Analysis **7:3,** (2003) 193–208.
22. H. Liu, F. Hussain, C. Lim and M. Dash, Discretization: An Enabling Technique, Data Mining and Knowledge Discovery **6:4,** (2002) 393-423.
23. S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley and A. Chircop, ECJ: A Java based evolutionary computation research system. *http://cs.gmu.edu/ eclab/projects/ecj* (2007).
24. A. Martínez-Estudillo, F. Martínez-Estudillo, C. Hervás-Martínez and N. García-Pedrajas, Evolutionary product

unit based neural networks for regression, Neural Networks **19,** (2006) 477–486.

25. M. Meyer and K. Hufschlag, A generic approach to an object-oriented learning classifier system library, Journal of Artificial Societies and Social Simulation, **9:3,** (2006). *http://jasss.soc.surrey.ac.uk/9/3/9.html.*
26. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz and T. Euler, YALE: Rapid Prototyping for Complex Data Mining Tasks. In Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2006) 1–6.
27. K. Morik and M. Scholz, The MiningMart Approach to Knowledge Discovery in Databases. In: Intelligent Technologies for Information Analysis, N. Zhong and J. Liu (eds), Springer, (2004) 47–65.
28. M. Mucientes, D.L. Moreno, A. Bugarín and S. Barro, Evolutionary learning of a fuzzy controller for wallfollowing behavior in mobile robotics, Soft Computing **10:10,** (2006) 881–889.
29. I.S. Oh, J.S. Lee and B.R. Moon, Hybrid Genetic Algorithms for Feature Selection, IEEE Transactions on Pattern Analysis and Machine Intelligence **26:11,** (2004) 1424-1437.
30. M. Ortega and J. Bravo, *Computers and Education in the 21st Century* (Kluwer Academic Publishers, Norwell 2000) 266.
31. J. Otero and L. Sánchez, Induction of descriptive fuzzy classifiers with the Logitboost algorithm, Soft Computing **10:9,** (2006) 825–835.
32. B. Punch and D. Zongker, lib-gp 1.1 beta. *http://garage.cse.msu.edu/software/lil-gp* (1998).
33. D. Pyle, *Data Preparation for Data Mining.* (Morgan Kaufmann, 1999).
34. J.R. Quinlan, *C4.5: programs for machine learning* (Morgan Kaufmann, San Mateo 1993) 316.
35. R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2005. *http://www.R-project.org*
36. R. Rakotomalala, TANAGRA: un logiciel gratuit pour l'enseignement et la recherche. In Proc. of the 5th Journées d'Extraction et Gestion des Connaissances 2, (2005) 697–702.
37. A.J. Rivera, I. Rojas, J. Ortega and M.J. del Jesus, A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks, Soft Computing **11:7,** (2007) 655–668.
38. J.J. Rodríguez, L.I. Kuncheva and C.J. Alonso, Rotation Forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence **28:10,** (2006) 1619–1630.
39. C. Romero, S. Ventura and P. de Bra, Knowledge Discovery with Genetic Programming for Providing Feedback to Courseware Author, User Modeling and User-Adapted Interaction. The Journal of Personalization Research **14:5,** (2004) 425–465.
40. A. Rummler, Evolvica: a Java framework for evolutionary algorithms. *http://www.evolvica.org* (2007)
41. J. Rushing, R. Ramachandran, U. Nair, S. Graves, R. Welch and H. Lin, ADaM: a data mining toolkit for scientists and engineers, Computers & Geosciences **31:5,** (2005) 607–618.
42. Z. Stejić, Y. Takama and K. Hirota, Variants of evolutionary learning for interactive image retrieval, Soft Computing **11:7,** (2007) 669–678.
43. J.C. Tan, T.H. Lee, D. Khoo and E.F. Khor, A multiobjective evolutionary algorithm toolbox for computer-aided multiobjective optimization, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics **31:4,** (2001) 537–556.
44. J.C. Tan, A. Tay and J. Cai, Design and implementation of a distributed evolutionary computing software, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics **33:3,** (2003) 325–338.
45. P.N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining* (Addison-Wesley, 2006) 769.
46. S. Ventura, C. Romero, A. Zafra, J.A. Delgado and C. Hervás, JCLEC: A Java framework for Evolutionary Computation, Soft Computing (2007) In press.
47. L.X. Wang and J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions Systems, Man, Cybernetics **22:6,** (1992) 1414-1427.
48. X. Wang, D.D. Nauck, M. Spott and R. Kruse, Intelligent data analysis with fuzzy decision trees, Soft Computing **11:5,** (2007) 439–457.
49. S.W. Wilson, Classifier Fitness Based on Accuracy, Evolutionary Computation **3:2,** (1995) 149–175.
50. D.R. Wilson and T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning **38,** (2000) 257–268.
51. I.H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques. Second Edition* (Morgan Kaufmann, San Francisco 2005) 525. *http://www.cs.waikato.ac.nz/ml/weka/index.html*
52. M.L. Wong and K.S. Leung, *Data mining using grammar based genetic programming and applications* (Kluwer Academic Publishers, Norwell 2000) 232.
53. S. Zhang, C. Zhang and Q. Yang, Data preparation for data mining, Applied Artificial Intelligence **17,** (2003) 375–381.