Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study

José Ramón Cano, Francisco Herrera, and Manuel Lozano

***Abstract*—Evolutionary algorithms are adaptive methods based on natural evolution that may be used for search and optimization. As data reduction in knowledge discovery in databases (KDDs) can be viewed as a search problem, it could be solved using evolutionary algorithms (EAs).**

In this paper, we have carried out an empirical study of the performance of four representative EA models in which we have taken into account two different instance selection perspectives, the prototype selection and the training set selection for data reduction in KDD. This paper includes a comparison between these algorithms and other nonevolutionary instance selection algorithms. The results show that the evolutionary instance selection algorithms consistently outperform the nonevolutionary ones, the main advantages being: better instance reduction rates, higher classification accuracy, and models that are easier to interpret.

***Index Terms*—Data mining (DM), data reduction, evolutionary algorithms (EAs), instance selection, knowledge discovery.**

I. Introduction

Advances in digital and computer technology that have led to the huge expansion of the Internet means that massive amounts of information and collection of data have to be processed. Scientific research, ranging from astronomy to human natural genome, faces the same problem of how to deal with vast amounts of information. Raw data is rarely used directly and manual analysis simply cannot keep up with the fast growth of data. Knowledge discovery in databases (KDD) and data mining (DM) can help deal with this problem because they aim to turn raw data into nuggets and create special edges.

The KDD process includes problem comprehension, data comprehension, data preprocessing, DM, evaluation, and development [1], [8], [29], [34]. The first three processes (problem and data comprehension, and data preprocessing) play an essential role in successful DM.

Due to the enormous amounts of data, much of the current research is based on scaling up DM algorithms. Other research has also tackled scaling down data. The main problem of scaling down data is how to select the relevant data and then apply a DM algorithm [21]. This task is carried out in the data preprocessing phase in a KDD process.

Manuscript received September 15, 2002; revised May 8, 2003 and July 25, 2003. This work was supported in part by Project TIC-2002-04036-C05-01.

J. R. Cano is with the Department of Electronic Engineering, Computer Systems and Automatics, Escuela Superior de La Rábida, University of Huelva, Huelva 21819, Spain (e-mail: jose.cano@diesia.uhu.es).

F. Herrera and M. Lozano are with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain (e-mail: herrera@decsai.ugr.es; lozano@decsai.ugr.es).

Digital Object Identifier 10.1109/TEVC.2003.819265

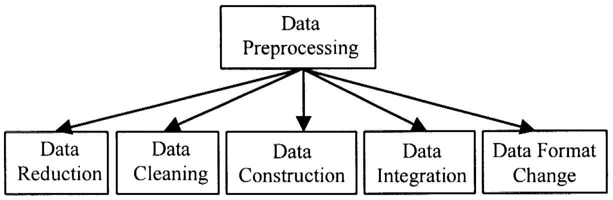


Fig. 1. Strategies in data preprocessing.

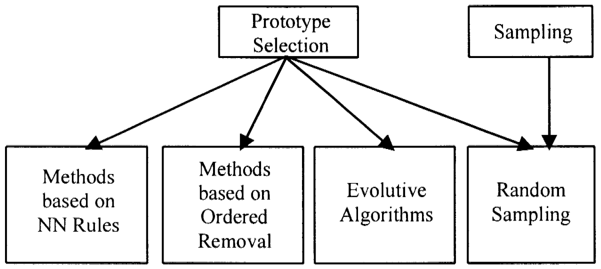


Fig. 2. PS algorithms.

Fig. 1 shows the different strategies in the data preprocessing phase.

Our attention is focused on data reduction, which can be achieved in many ways.

- By selecting features [20], we reduce the number of columns in a data set.
- By making the feature values discrete [12], we reduce the number of possible values of features.
- By selecting instances [5], [18], [22], we reduce the number of rows in a data set.

This study deals solely with instance selection (IS) [6], [21], [26], [33], [36] by means of evolutionary algorithms (EAs) for data reduction in KDD.

IS is a focusing task in the data preparation phase [8] of KDD. It is one of the effective means of data reduction. IS can follow different strategies: sampling, boosting, prototype selection (PS), and active learning. We are going to study the IS from the PS perspective. The PS algorithms that we have assessed can be classified as shown in Fig. 2. A complete review of these is presented in Section II-C.

EAs [3] are general-purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions to problems. The basic idea is to maintain a population of chromosomes, which represent plausible solutions to the problem and evolve over time through a process of competition and controlled variation. EAs have been used to solve the IS problem, with promising results [17], [25].

The aim of this paper is to study the application of some representative EA models for data reduction, and to compare them with nonevolutionary instance selection algorithms (hereafter referred to as classical ones). In order to do this, our study is carried out from a twofold perspective.

1) IS-PS: The analysis of the results obtained when selecting prototypes (instances) for a 1-NN (nearest neighbor) algorithm. This approach will be denoted as instance selection-prototype selection (IS-PS).
2) IS-TSS: The analysis of the behavior of EAs as instance selectors for data reduction, when selecting instances to compose the training set that will be used by C4.5 [24], a well-known decision-tree induction algorithm. In this approach, the selected instances are first used to build a decision tree, and then the tree is used to classify new examples. This approach will be denoted as (instance selection–training set selection (IS-TSS).

The analysis of the behavior of EAs for data reduction in KDD is, in fact, the most important and novel aspect of this paper.

As with any algorithm, the issue of scalability and the effect of increasing the size of data on algorithm behavior are always present. To address this, we have carried out a number of experiments on IS-PS and IS-TSS with increasing complexity and size of data.

In order to do this, this paper is set out as follows. In Section II, we introduce the main ideas about IS, describing the two processes which IS algorithms take part in, the IS-PS and the IS-TSS, and we also summarize the classical IS algorithms. In Section III, we introduce the foundations of EAs and summarize the main features of the models considered in this paper, giving details of how EAs can be applied to the IS problem. In Section IV, we explain the methodology used in the experiments. Sections V and VI deal with the results and the analysis of small and medium data sets, respectively. Section VII deals with the study of the main points of the evolutionary instance selection approach, including scalability to larger data sets. Finally, in Section VIII, we reach our conclusion.

II. Instance Selection

In this section, we describe the two strategies which IS takes part in as presented in this paper, as well as a summary of classical IS algorithms.

A. Instance Selection for Prototype Selection (IS-PS)

The 1-NN classifiers predict the class of a previously unseen instance by computing its similarity with a set of stored instances called prototypes. PS–storing a well-selected, proper subset of available training instances has been shown to increase classifier accuracy in many domains. At the same time, the use of prototypes dramatically decreases both storage and classification-time costs.

A PS algorithm is an IS algorithm that attempts to obtain a subset of the training set that allows the 1-NN classifier to achieve the maximum classification rate. Fig. 3 shows the way in which a PS algorithm acts.

Each PS algorithm is applied to an initial data set in order to obtain a subset of representative data items. We assess the accuracy of the subset selected using an 1-NN classifier.

B. Instance Selection for Training Set Selection (IS-TSS)

There may be situations in which there is too much data and this data in most cases is not equally useful in the training phase of a learning algorithm [25]. Instance selection mechanisms have been proposed to choose the most suitable points in the data set to become instances for the training data set used by a learning algorithm. For example, in [25], a genetic algorithm (GA) is used for training data selection in radial based function networks.

Fig. 4 shows a general framework for the application of an IS algorithm for TSS. Starting from the data set TR, the IS algorithm finds a suitable set S, then a learning or DM algorithm is applied to evaluate each subset selected (C4.5 in our case [24]) to obtain a model from the data set. This model is assessed using the test data set TS.

C. Overview of Instance Selection Algorithms

Historically, IS has been mainly aimed at improving the efficiency of the NN classifier. The NN algorithm is one of the most venerable algorithms in machine learning. This algorithm calculates the Euclidean distance (possibly weighted) between an instance to be classified and each training-neighboring instance. The new instance to be classified is assigned to the class of the nearest neighboring one. More generally, the k-NN are computed and the new instance is assigned to the most frequent class among these k neighbors. The k-NN classifier was also widely used and encouraged by early theoretical results related to its Bayes error generalization.

However, from a practical point of view, the k-NN algorithm is not suitable for dealing with very large sets of data due to the storage requirements it demands and the computational costs involved. In fact, this approach requires the storage of all the instances in memory. Early research in instance selection first tried to reduce storage size. The algorithms used in this study are described in the remainder of the subsection, with the exception of EAs, which are described in Section III.

1) Methods Based on NN Rules:

Cnn [15]—It tries to find a consistent subset, which correctly classifies all of the remaining points in the sample set. However, this algorithm will not find a minimal consistent subset [33].

Enn [31]—Edited NN edits out noisy instances, as well as close border cases, leaving smoother decision boundaries. It also retains all internal points, which keeps it from reducing the storage requirements as much as most other reduction algorithms.

Renn [31]—The repeated edited NN continues to widen the gap between classes and smooth the decision boundary, applying *Enn* repeatedly.

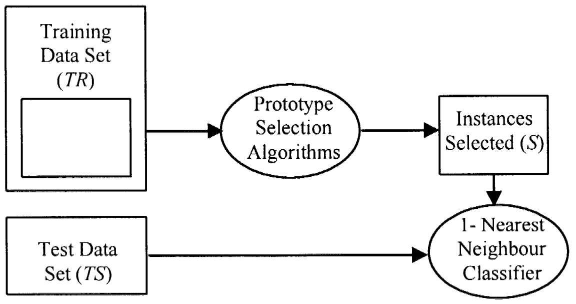


Fig. 3. IS-PS strategy.

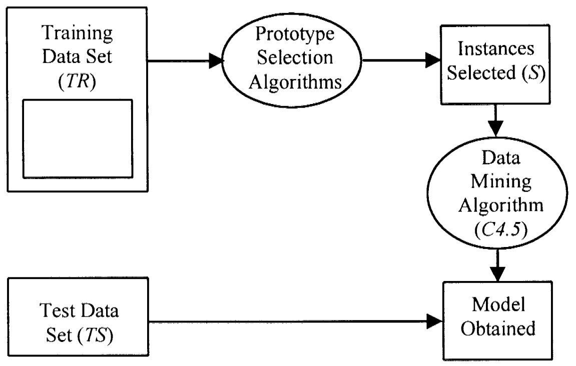


Fig. 4. IS-TSS strategy.

Rnn [13]—The reduced NN rule searches in Cnn's consistent set, the minimal subset which correctly classifies all the learning instances. However, this approach is efficient if and only if Cnn's consistent set contains the minimal consistent set of the learning set, which is not always the case.

Vsm [23]—It removes an instance if most of its k-nearest neighbors (>60%) classify it correctly or incorrectly.

Multiedit [9]—It is a modification over Enn algorithm that guarantees the statistical independence in the prototype selected.

Mcs [7]—Mcs system (model class selection system) tends to avoid noise.

Shrink [18]—It is similar to the Rnn. It retains border points but unlike Rnn, this algorithm is sensitive to noise.

Ib2 [18]—It is similar to Cnn but using a different selection strategy.

Ib3 [2]—It outperforms Ib2 introducing the *acceptable* instance concept to carry out the selection.

Icf [6]: It tries to select the instances which classify more prototypes correctly. Icf uses coverage and reachable concepts to carry out the selection.

2) *Methods Based on Ordered Removal:*

Drop1 [32]—Essentially, this rule tests to see if removing an instance would degrade leave-one-out cross-validation generalization accuracy, which is an estimate of the true generalization ability of the resulting classifier.

Drop2 [32]—Drop2 changes the order of removal of instances. It initially sorts the instances in TR by the distance to their nearest enemy (nearest instance belonging to another class). Instances are then checked for removal beginning at the instance furthest from its nearest enemy. This tends to remove instances furthest from the decision boundary first, which in turn increases the chance of retaining border points.

Drop3 [32]—Drop3 uses a noise filtering pass before sorting the instances in training date (TR) Figs. 3 and 4. This is done using the rule: Any instance incorrectly classified by its k-NN is removed.

3) *Methods Based on Random Sampling:*

Rmhc(s)[30]—First, it randomly selects a subset S from TR which contains a fixed number of instances s ($s = \%|\text{TR}|$). In each iteration, the algorithm interchanges an instance from S with another from TR-S. The change is maintained if it offers better accuracy.

Ennrs(s) [33]—Similar to Rmhc, it randomly selects a subset S from TR, which contains a fixed number of instances s ($s = \%|\text{TR}|$) but in each iteration the algorithm interchanges all instances from S with instances from TR. The change is maintained if it offers better accuracy.

III. Evolutionary Instance Selection Algorithms

Most of the success of EAs is due to their ability to exploit the information accumulated about an initially unknown search space. This is their key feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic, etc.) are inappropriate. In such cases, they offer a valid approach to problems requiring efficient and effective search techniques.

In this section, we first describe the EAs used in this study, and then present the key points of their application to our problem, as well as the representation and the fitness function.

A. *Evolutionary Algorithms (EAs)*

EAs [3] are stochastic search methods that mimic the metaphor of natural biological evolution. All EAs rely on the concept of a *population* of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as *mutation*, *selection*, and (sometimes) *recombination* to evolve toward increasingly better fitness values of the individuals. The *fitness* of an individual reflects its objective function value with respect to a particular objective function to be optimized. The mutation operator introduces innovation into the population by generating variations of individuals and the recombination operator typically performs an information exchange between different individuals from a population. The selection operator imposes a driving force on the process of evolution by preferring better individuals to survive and reproduce when the members of the next generation are selected.

Next, we describe the four models of EAs that will be used in this paper as evolutionary instance selection algorithms. The first two are the classical GA models; the generational one and the steady-state one. The third one, heterogenious recombination and cataclysmic mutation (CHC), is a classical model that introduces different features to obtain a tradeoff between exploration and exploitation, and the fourth one is a specific EA approach, designed for binary search spaces.

1) Generational Genetic Algorithm (GGA): The basic idea in GGA [14], [16] is to maintain a *population* of *chromosomes*, which represent plausible solutions to the particular problem that evolves over successive iterations (*generations*) through a process of competition and controlled variation. Each chromosome in the population has an associated fitness to determine which chromosomes are to be used to form new ones in the competition process. This is called *selection*. The new ones are created using genetic operators such as *crossover* and *mutation*. The classical model of GAs is the GGA, which consists of three operations:

1) evaluation of individual fitness;
2) formation of a gene pool (intermediate population) through selection mechanism;
3) recombination through crossover and mutation operators.

The selection mechanism produces a new population $P(t)$ with copies of chromosomes in $P(t-1)$. The number of copies received for each chromosome depends on its fitness; chromosomes with higher fitness usually have a greater chance of contributing copies to $P(t)$. Then, the crossover and mutation operators are applied to $P(t)$.

Crossover takes two individuals called parents and produces two new individuals called the offspring by swapping parts of the parents. In its simplest form, the operator works by exchanging substrings after a randomly selected crossover point. The crossover operator is not usually applied to all pairs of chromosomes in the new population. A random choice is made, where the likelihood of crossover being applied depends on probability defined by a *crossover rate*.

Mutation serves to prevent premature loss of population diversity by randomly sampling new points in the search space. Mutation rates are kept small, however, otherwise the process degenerates into a random search. In the case of bit strings, mutation is applied by flipping one or more random bits in a string with a probability equal to the *mutation rate*.

Termination may be triggered by reaching a maximum number of generations or by finding an acceptable solution by some criterion.

2) Steady-State Genetic Algorithm (SGA): In SGAs [35], usually only one or two offspring are produced in each generation. Parents are selected to produce offspring and then a replacement/deletion strategy defines which member of the population will be replaced by the new offspring. The basic algorithm steps of SGA are the following.

1) Select two parents from the population P.
2) Create an offspring using crossover and mutation.
3) Evaluate the offspring with the fitness function.
4) Select an individual in P, which may be replaced by the offspring.
5) Decide if this individual will be replaced.

In step 4), the replacement strategy can be chosen (e.g., replacement of the worst, the oldest, or a randomly chosen individual). In step 5), the replacement condition can be chosen (e.g., replacement if the new individual is better, or unconditional replacement). A widely used combination is to replace the worst individual only if the new individual is better (this is the one we have adopted in this paper).

3) CHC Adaptive Search Algorithm: During each generation the CHC [10] develops the following steps.

1) It uses a parent population of size N to generate an intermediate population of N individuals, which are randomly paired and used to generate N potential offspring.
2) Then, a survival competition is held where the best N chromosomes from the parent and offspring populations are selected to form the next generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator. HUX exchanges half of the bits that differ between parents, where the bit position to be exchanged is randomly determined. CHC also employs a method of incest prevention. Before applying HUX to two parents, the Hamming distance between them is measured. Only those parents who differ from each other by some number of bits (mating threshold) are mated. The initial threshold is set at $L/4$, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold is reduced by one.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any members of the parent population) the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to reseed the population. Reseeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other $N-1$ new chromosomes in the population. The search is then resumed.

4) Population-Based Incremental Learning (PBIL): PBIL [4] is a specific EA designed for binary search spaces. The PBIL algorithm attempts to explicitly maintain statistics about the search space to decide where to sample next.

The object of the algorithm is to create a real valued probability vector V_p, which, when sampled, reveals high quality solution vectors with high probability. For example, if a good solution can be encoded as a string of alternating 0s and 1s, a possible final V_p would be 0.01, 0.99, 0.01, 0.99, etc. Initially, the values of V_p are set at 0.5. Sampling from this vector yields random solution vectors because the probability of generating a 1 or 0 is equal. As the search progresses, the values of V_p gradually shift to represent high evaluation solution vectors through the following process.

1) A number of solution vectors (*Nsamples*) are generated based upon the probabilities specified in V_p.
2) V_p is pushed toward the generated solution vector with the highest evaluation *Sbest*

$$V_p[\mathrm{i}] = V_p[\mathrm{i}] \cdot (1 - LR) + Sbest[\mathrm{i}] \cdot LR$$

where *LR* is the *learning rate*, which specifies how close the steps are to the best solution.

3) V_p is pushed far away from the worst evaluation, *Sworse*, where *Sbest* and *Sworse* differ. This is accomplished as follows:

$$\text{If } Sbest[\text{i}] <> Sworse[\text{i}] \text{ then}$$
$$V_p[\text{i}] = V_p[\text{i}] \cdot (1 - Negat\_LR) + Sbest[\text{i}] \cdot Negat\_LR$$

where *Negat_LR* is the *negative learning rate*, which specifies how far away the steps are from the worst solution.

4) After the probability vector is updated, sampling the updated probability vector produces a new set of solution vectors, and the cycle is continued.

Furthermore, PBIL applies mutations to V_p, with an analogous aim as mutation in the case of GAs: to inhibit premature convergence. Mutations affect V_p with low probability P_{m} in a random direction *Mut_Shif*.

It must be pointed out that there are problems setting the *LR* and *Negat_LR* in PBIL, which affects the convergence and the quality of solutions.

B. Evolutionary IS

EAs may be applied to the IS problem, because it can be considered as a search problem. In this paper, these EAs have been called *evolutionary instance selection algorithms*. Examples of these algorithms may be found in [17], which deals with the application of the GAs to IS-PS, and [25] which deals with the application of the GAs to IS-TSS.

The objective of this paper is to study the performance of the four EAs described in the previous section as IS algorithms applied to IS-PS and to IS-TSS for data reduction in KDD, comparing their results with the ones obtained by the algorithms introduced in Section II-C.

The application of EAs to these two approaches is accomplished by tackling two important issues: the specification of the representation of the solutions and the definition of the fitness function.

1) Representation: Let us assume a data set denoted TR with m instances. The search space associated with the instance selection of TR is constituted by all the subsets of TR. Then, the chromosomes should represent subsets of TR. This is accomplished by using a binary representation. A chromosome consists of m genes (one for each instance in TR) with two possible states: 0 and 1. If the gene is 1, then its associated instance is included in the subset of TR represented by the chromosome. If it is 0, then this does not occur.

2) Fitness Function: Let S be a subset of instances of TR to evaluate and be coded by a chromosome. We define a fitness function that combines two values: the classification rate (*clas_rat*) associated with S and the percentage of reduction (*perc_red*) of instances of S with regards to TR

$$\text{Fitness}(S) = a \cdot \text{clas\_rat} + (1 - a) \cdot \text{perc\_red}.$$

The 1-NN classifier (Section II-C) is used for measuring the classification rate, *clas_rat*, associated with S. It denotes the percentage of correctly classified objects from TR using only S to find the nearest neighbor. For each object y in S, the nearest neighbor is searched for amongst those in the set $S \backslash \{y\}$. Whereas, *perc_red* is defined as

$$\text{perc\_red} = 100 \cdot \frac{(|\text{TR}| - |S|)}{|\text{TR}|}.$$

The objective of the EAs is to maximize the fitness function defined, i.e., maximize the classification rate and minimize the number of instances obtained. In the experiments presented in this paper, we have considered the value $\alpha = 0.5$ in the fitness function, as per a previous experiment in which we found the best tradeoff between precision and reduction with this value.

We use 1-NN in both strategies (IS-PS and IS-TSS) to evaluate the fitness of a chromosome, in order to test the effectiveness of performing instance selection independently of the ultimate classification algorithm to be applied to the test data set. Note that in the IS-TSS study the classification algorithm to be applied to the test set is C4.5, which is very different from the 1-NN algorithm. This mismatch doest not occur in the IS-PS study, where 1-NN is used for both fitness computation and classification of the test set.

TABLE I
SMALL SIZE DATA SETS

Data set	Num. Instances	Num. Features	Num. Classes
Cleveland	297	13	2
Glass	214	9	6
Iris	150	4	3
LED24Digit	200	24	10
LED7Digit	500	7	10
Lymphography	148	18	4
Monk	432	6	2
Pima	768	8	2
Wine	178	13	3
Wisconsin	683	9	2

IV. Experimental Methodology

In this section, we present the methodology used in the experiments carried out. The data sets used in our experiments are available from UCI Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html).

As previously mentioned, we have carried out our study of IS-PS and IS-TSS problems using two size problems: small and medium. We intend to study the behavior of the algorithms when the size of the problem increases. Section IV-A describes the data sets used (small and medium), Section IV-B introduces the partition of the data sets that were considered for applying the algorithms, Section IV-C introduces the parameters associated with the algorithms, and finally, in Section IV-D the results are presented in table form.

A. Data Sets

1) Small Size Data Sets: See Table I.
2) Medium Size Data Sets: See Table II.

B. Partitions and Executions

The data sets considered are partitioned using the *tenfold cross-validation* procedure. Each initial data set T, is randomly divided into ten disjoint sets of equal size $T_1, \ldots, T_{10}$. We

TABLE II
MEDIUM SIZE DATA SETS

Data set	Num. Instances	Num. Features	Num. Classes
Pen-Based Recognition	10992	16	10
Satimage	6435	36	6
Thyroid	7200	21	3

TABLE III
PARAMETERS

Algorithm	Parameters
Ib3	*Acept. Level*=0.9, *Drop Level*=0.7
Rmhc	*s*=90% , *Eval*=10000
Ennrs	*s*=90% , *Eval*=10000
GGA	P_m=0.001, P_c=0.6, *Pop*=50, *Eval*=10000
SGA	P_m=0.001, P_c=1, *Pop*=50, *Eval*=10000
CHC	*Pop*=50 , *Eval*=10000
PBIL	*LR*=0.1, *Mut_shift*=0.05, P_m= 0.02, *Pop*=50, *Negative_LR*=0.075, *Eval*=10000

maintain the original class distribution (before partitioning) within each set when carrying out the partition process. We then conduct ten pairs of training and test sets, $(\mathrm{TR}_i, \mathrm{TS}_i)$, $i = 1, \dots, 10$. For each pair i, the test set, TS_i, is T_i, and the training set, TR_i, is the union of all of the other $T_j, j \neq i$.

Ten trials were run for each data set and *IS* nonprobabilistic algorithm. During the ith-trial, the instance selection algorithm is applied to TR_i, selecting a subset of instances. The selected instances are used as a training set by a classification algorithm (1-NN or C4.5, depending on the experiment), and then the accuracy of the classification algorithm in the test set is measured. Using EAs and the classical random algorithms (Rmhc and Ennrs), we executed each one three times for each partition, taking the average of the results of these three executions per partition, therefore, we have 30 trials per data set for each one of them.

C. Parameters

Whether either small or medium data sets are evaluated, the parameters used are the same: see Table III.

D. Types of Table Results

In the following sections, we will present two types of tables, each one with the following structure.

1) Complete Results Table: This type of table shows the results obtained in IS-PS or IS-TSS by the classical and evolutionary instance selection algorithms, respectively. In order to observe the level of robustness achieved by all algorithms, the table presents the average of the results offered by each algorithm in the data sets evaluated (small or medium group of data sets). Each column shows (see for example, Table IV):

- The first column shows the name of the algorithm. As reduction is one of our main objectives, we observe two main groups in the algorithm results (one group with a data reduction higher than 70% and the rest with a lower reduction). We highlight, with the simbol "(*)" the algorithm group that offers the best reduction rates (>70%). The highlighted algorithms will be used in the following table results.

TABLE IV
AVERAGE RESULTS OF IS-PS FOR SMALL DATA SETS

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac. Trn	%Ac. Test
1-NN	0.07		68.44	65.73
Cnn (*)	0.01	71.98	49.11	52.18
Drop1(*)	0.23	86.97	68.10	62.74
Drop2	0.20	55.19	67.23	67.16
Drop3(*)	0.15	78.56	**86.62**	63.80
Enn	0.10	35.07	72.67	66.37
Ib2 (*)	0.03	77.80	47.75	51.36
Icf (*)	0.29	75.81	66.08	62.01
Mcs	0.09	16.90	74.56	67.08
Multied	0.24	54.70	61.33	61.06
Renn	0.25	37.43	72.30	65.97
Rnn (*)	1.89	90.38	68.17	65.51
Shrink	0.08	30.41	72.05	66.64
Vsm (*)	0.01	74.56	59.89	58.87
Ib3	0.06	65.71	60.42	64.67
Rmhc(*)	54.37	90.16	57.02	58.17
Ennrs (*)	69.95	90.16	69.52	64.13
GGA (*)	70.8	87.72	77.12	**67.54**
SGA (*)	68.6	90.50	**77.89**	**67.65**
CHC (*)	20.48	**96.05**	75.86	64.37
PBIL (*)	43.2	**93.79**	72.49	64.63

- The second column contains the average execution time associated to each algorithm. The algorithms have been run in a Pentium 4, 2.4 GHz, 256 RAM, 40 Gb HD.
- The third column shows the average reduction percentage from the initial training sets.
- The fourth column contains the accuracy when 1-NN or C4.5 is applied using the training set selected. This column presents two subcolumns: the first one shows the training accuracy when using the training set selected (S_i) from the initial set (TR_i); the second subcolumn shows the test accuracy of S_i over the test data set TS_i. Each subcolumn presents the accuracy mean. We study both subcolumns in an effort to evaluate the existence of overfitting in the instance selection phase.

In the third, fourth, and fifth columns, the two best results per column are shown in bold.

By studying only the first type of table as described before, two drawbacks can be appreciated.

- The evaluation of just the mean classification accuracy over all the data sets hides important information because it is possible that a particular algorithm performs exceptionally well on one data set but not as good as the others on the rest of the data sets. If the mean error of this algorithm over all the data sets is better than the other methods, it does not necessarily mean that this algorithm is better.
- The average classification rate, across many data sets, is not very meaningful, because each data set represents a different classification problem, and different data sets have many different degrees of difficulty. For instance, if in a data set where the "baseline accuracy" (i.e., the relative frequency of the majority class) is 60% and an algorithm obtains an accuracy rate of 80%, then a good result has been obtained, whereas in a data set where the baseline accuracy is 90%, an algorithm obtaining an accuracy of 80% is a bad result.

TABLE V
RANKING AND NUMBER OF BEST RESULTS OF ALGORITHMS WHICH HAVE A REDUCTION RATE GREATER THAN 70%

%Reduction			%Ac. test 1-NN			%Ac. 1-NN*0.5 +%Rd.*0.5		
Nam	**Rnk**	**Bst**	**Nam**	**Rnk**	**Bst**	**Nam**	**Rnk**	**Bst**
CHC	1.3	7	*SGA*	3.5	2	*PBIL*	2.7	4
PBIL	2.9	1	*GGA*	4.2	0	*SGA*	3.4	1
SGA	5.1	0	*PBIL*	4.3	5	*CHC*	3.5	4
Rnn	5.8	0	*Ennrs*	5.4	0	*Rnn*	4.6	0
Rmhc	5.8	1	*Rnn*	5.6	1	*Ennsr*	5.3	1
Drop1	6.6	1	*CHC*	6.6	1	*GGA*	5.5	0
Ennrs	6.8	0	*Drop1*	7.2	1	*Rmhc*	7.5	0
Ib2	7.3	0	*Drop3*	7.2	0	*Drop1*	7.6	0
GGA	7.8	0	*Icf*	8.4	0	*Drop3*	8.1	0
Cnn	9.7	0	*Rmhc*	8.6	0	*Icf*	10.1	0
Drop3	9.8	0	*Vsm*	9.4	0	*Ib2*	10.5	0
Icf	11	0	*Cnn*	10.1	0	*Vsm*	10.6	0
Vsm	11.1	0	*Ib2*	10.5	0	*Cnn*	11.6	0

To avoid these drawbacks, we have included a second type of table showing rankings and the number of data sets in which each algorithm obtained the best results. The best result in the second type of table eliminates the first drawback. The average ranking does not use the absolute value of the algorithms' accuracy, only its relative value among the competing algorithms that clarifies the results offered, thus also avoiding the second drawback.

In the following, the contents of this second type of table are described in depth.

2) Table With Ranking of Algorithms With Data Reduction Rates Greater Than 70%: In this type of table, we have used the results offered by each algorithm in the data sets evaluated (small or medium) to carry out a classification. We have presented the results using the following criterion: The algorithms, with a reduction rate greater than 70%, are placed in descending order according to the results obtained. The best algorithm is ranked first, etc. The average ranking of the algorithm is presented as a measure varying between 1 and the number of algorithms compared. An example of this type of table is Table V.

The columns of this type of table are described as follows.

- The first column shows classification using the reduction rate as the sole objective. This column is divided into three subcolumns: the first one (Nam) contains the name of the algorithm ranked in order, the second one (Rnk) presents its average ranking, and the third subcolumn (Bst) shows the number of data sets that have ranked that algorithm as the best.
- The second column offers the same information as the first one but with test accuracy as the sole objective.
- The third column presents the same information as the previous columns, but in this case the objective is a 50–50 combination of reduction and test accuracy.

V. SMALL DATA SETS: RESULTS AND ANALYSIS

A. Results and Analysis in IS-PS

Tables IV and V show the results obtained in IS-PS by the classical and evolutionary instance selection algorithms, respectively.

TABLE VI
AVERAGE RESULTS OF IS-TSS FOR SMALL DATA SETS

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	**%Ac. Test**
C4.5	0.01		89.53	71.79
Cnn (*)	0.01	71.98	85.45	49.10
Drop1(*)	0.23	86.97	87.25	53.41
Drop2	0.2	55.19	83.60	62.33
Drop3(*)	0.15	78.56	86.79	55.52
Enn	0.1	35.07	95.66	68.29
Ib2 (*)	0.03	77.80	84.71	51.22
Icf (*)	0.29	75.81	94.59	64.02
Mcs	0.09	16.90	90.70	**71.82**
Multied	0.24	54.70	**99.83**	60.25
Renn	0.25	37.43	97.27	68.40
Rnn (*)	1.89	90.38	92.34	62.03
Shrink	0.08	30.41	**96.12**	**69.26**
Vsm (*)	0.01	74.56	90.31	63.25
Ib3	0.06	65.71	79.18	65.22
Rmhc(*)	54.37	90.27	89.13	64.70
Ennrs(*)	69.95	90.27	91.31	62.04
GGA (*)	70.8	87.72	90.29	62.87
SGA (*)	68.6	90.50	88.94	63.36
CHC (*)	20.48	**96.05**	80.49	53.61
PBIL (*)	43.2	**93.79**	93.25	67.44

TABLE VII
RANKING AND NUMBER OF BEST RESULTS OF ALGORITHMS WHICH HAVE A REDUCTION RATE GREATER THAN 70%

%Reduction			%Ac. test C4.5			%Ac. C4.5*0.5 +%Rd.*0.5		
Nam	**Rnk**	**Bst**	**Nam**	**Rnk**	**Bst**	**Nam**	**Rnk**	**Bst**
CHC	1.3	7	*SGA*	4.7	1	*PBIL*	2.4	3
PBIL	2.9	1	*PBIL*	4.9	2	*SGA*	4.3	1
SGA	5.1	0	*Icf*	5	1	*Ennrs*	4.7	1
Rnn	5.8	0	*Ennrs*	5.3	0	*Rnn*	4.8	0
Rmhc	5.8	1	*Rmhc*	5.5	2	*Rmhc*	4.8	3
Drop1	6.6	1	*GGA*	5.5	2	*GGA*	5.9	1
Ennrs	6.8	0	*Rnn*	6.1	1	*CHC*	6.1	1
Ib2	7.3	0	*Vsm*	7.2	0	*Drop1*	8.5	0
GGA	7.8	0	*Drop1*	8.3	1	*Icf*	8.5	0
Cnn	9.7	0	*Drop3*	8.7	0	*Vsm*	9.3	0
Drop3	9.8	0	*Ib2*	9.9	0	*Drop3*	9.9	0
Icf	11	0	*CHC*	9.9	0	*Ib2*	10.5	0
Vsm	11.1	0	*Cnn*	10	0	*Cnn*	11.3	0

The following conclusions about the evolutionary instance selection algorithms for PS studying Tables IV and V can be made.

- The EAs present the best reduction rates, maintaining the highest test accuracy rates in Table IV.
- In Table V, we can see that the EAs offer the best results in reduction, being the best in eight out of ten data sets. In test accuracy, EAs behave in the same way, obtaining the best results in eight data sets. Evaluating the balance between test accuracy and reduction percentages, EAs obtain the best result.

This indicates that EAs offer the best results in IS-PS.

B. Results and Analysis in IS-TSS

The following conclusions about the evolutionary instance selection algorithms for TSS studying Tables VI and VII can be made.

- In Table VI, we can see that EAs only have been outperformed in their test accuracy by methods which offer small

TABLE VIII
AVERAGE RESULTS OF IS-PS FOR MEDIUM DATA SETS

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac Trn	%Ac Test
1-NN	46		94.19	94.18
Cnn (*)	4	97.32	79.33	80.17
Drop1 (*)	254	96.72	78.00	76.85
Drop2 (*)	215	89.57	88.62	88.62
Drop3 (*)	338	96.25	89.03	85.44
Enn	139	5.82	95.43	94.39
Ib2 (*)	2	**97.78**	75.24	75.84
Icf (*)	386	90.04	76.77	76.68
Mcs	101	2.66	**95.96**	**94.38**
Multied	1778	13.99	92.43	92.10
Renn	489	6.47	95.37	94.30
Rnn (*)	13017	96.88	81.27	81.74
Shrink	206	4.89	95.19	**94.38**
Vsm	94	1.04	94.16	94.17
Ib3 (*)	42	71.67	91.49	92.61
Rmhc (*)	34525	90.02	91.59	91.15
Ennsr (*)	37802	90.02	92.79	92.75
GGA	66157	62.53	94.74	93.85
SGA	54656	62.91	95.00	93.67
CHC (*)	8072	**99.29**	93.31	93.53
PBIL (*)	32942	73.13	**96.23**	94.13

reduction rates. In general, all the algorithms obtained a training accuracy much larger than the test accuracy, indicating a large amount of overfitting.

- Table VII shows that EAs are the best algorithms with respect to classification accuracy in the test set, offering the best results in five data sets. EAs are again the best algorithm when we consider both reduction and test accuracy rates as the objective, producing the best results in six data sets.
- Table VI points out the presence of overfitting in all algorithms evaluated. Unfortunately, the overfitting appears frequently in the learning phase [27]. In particular, the overfitting is present in our case due to the selection is done using 1-NN in the fitness function in IS-TSS strategy. Hence, the set of selected instances was specifically adjusted to 1-NN, whereas the algorithm used to classify the test set was C4.5 as mentioned earlier.

The main conclusion that can be drawn when using small data sets is that EAs are very good algorithms for data reduction having both high reduction rates and accuracy.

VI. MEDIUM DATA SETS: RESULTS AND ANALYSIS

In this section, we present a similar study for the three medium size data sets shown in Table II. We aim to study the behavior of the algorithms when increasing the size of the problem by trying to find a scaling up algorithm for scaling down data. The tables for each data set in IS-PS and IS-TSS can be found in the Appendix at the end of this paper (Tables XVI–XXI).

A. Results and Analysis in IS-PS

Tables VIII and IX show the results in a similar way to the previous ones for small data sets.

TABLE IX
RANKING AND NUMBER OF BEST RESULTS OF ALGORITHMS WHICH HAVE A REDUCTION RATE GREATER THAN 70%

% Reduction			%Ac. test *1-NN*			%Ac. *1-NN**0.5 +%Rd.*0.5		
Nam	Rnk	Bst	Nam	Rnk	Bst	Nam	Rnk	Bst
CHC	1	3	*PBIL*	2	2	*CHC*	1	3
Ib2	2	0	*Ib3*	3	1	*Ennrs*	4.3	0
Drop1	3.6	0	*CHC*	3.3	0	*Drop3*	4.6	0
Cnn	4	0	*Ennrs*	4.3	0	*Rmhc*	5.6	0
Drop3	5.3	0	*Rmhc*	6.3	0	*Drop2*	6.3	0
Rnn	5.3	0	*Drop2*	6.6	0	*Ib3*	6.6	0
Rmhc	8.3	0	*Cnn*	8	0	*Drop1*	7	0
Icf	8.6	0	*Drop3*	8.3	0	*Cnn*	7	0
Drop2	9	0	*Ib2*	8.3	0	*Ib2*	7.6	0
Ennrs	9.3	0	*Rnn*	8.6	0	*Rnn*	7.6	0
Ib3	9.6	0	*Drop1*	9.3	0	*PBIL*	9.6	0
PBIL	11.6	0	*Icf*	9.6	0	*Icf*	10.3	0

TABLE X
AVERAGE RESULTS OF IS-TSS FOR MEDIUM DATA SETS

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	%Ac. Test
C4.5	1		98.82	**94.07**
Cnn (*)	4	97.32	93.68	72.03
Drop1 (*)	254	96.72	95.19	78.40
Drop2 (*)	215	89.57	95.42	79.04
Drop3 (*)	338	96.25	95.14	75.33
Enn	139	5.82	99.26	93.83
Ib2 (*)	2	**97.78**	93.06	68.13
Icf (*)	386	90.04	97.85	82.39
Mcs	101	2.66	98.97	93.91
Multied	1778	13.99	**99.77**	90.61
Renn	489	6.47	**99.31**	93.83
Rnn (*)	13017	96.88	99.16	69.50
Shrink	206	4.89	99.23	93.85
Vsm	94	1.04	98.84	**94.07**
Ib3 (*)	42	71.67	94.94	84.67
Rmhc (*)	34525	90.02	98.11	91.80
Ennsr (*)	37802	90.02	97.99	89.73
GGA	66157	62.53	98.72	92.60
SGA	54656	62.91	98.38	92.55
CHC (*)	8072	**99.29**	98.29	91.82
PBIL (*)	32942	73.13	98.71	92.72

From the analysis of Tables VIII and IX, we would point out the following.

- The EAs, and particularly the CHC, appear to be the best proposal when the size of the problem increases. It offers the best reduction rate and one of the best accuracy rates. The vast majority of the algorithms (with the exception of Drop3) avoid overfitting when the size of the problem increases.
- Table IX indicates that CHC is the best algorithm to be applied to IS-PS. It offers the best ranking in reduction rates, being the best for all data sets. It presents the best ranking on balanced objectives, with the best results in all data sets.

In these kinds of problems, evolutionary and classical algorithms have difficulties maintaining high reduction rates together with high accuracy rates. Classical algorithms with high reduction rates are penalized in their accuracy rate.

TABLE XI
RANKING AND NUMBER OF BEST RESULTS OF ALGORITHMS WHICH HAVE A REDUCTION RATE GREATER THAN 70%

%Reduction			%Ac. test *C4.5*			%Ac. *C4.5**0.5 +%Rd.*0.5		
Nam	Rnk	Bst	Nam	Rnk	Bst	Nam	Rnk	Bst
CHC	1	3	*PBIL*	1.3	2	*CHC*	1	3
Ib2	2	0	*CHC*	2.6	0	*Rmhc*	4.3	0
Drop1	3.6	0	*Rmhc*	3	1	*Ennrs*	4.6	0
Cnn	4	0	*Ennrs*	4	0	*Drop1*	5.6	0
Drop3	5.3	0	*Ib3*	4	0	*Drop3*	6	0
Rnn	5.3	0	*Icf*	6.3	0	*Icd*	6.3	0
Rmhc	8.3	0	*Drop2*	7	0	*Ib3*	7.3	0
Icf	8.6	0	*Drop1*	8.6	0	*Cnn*	7.6	0
Drop2	9	0	*Drop3*	9.3	0	*PBIL*	8	0
Ennrs	9.3	0	*Cnn*	9.6	0	*Drop2*	8.6	0
Ib3	9.6	0	*Rnn*	10.6	0	*Rnn*	9	0
PBIL	11.6	0	*Ib2*	11.3	0	*Ib2*	9.3	0

B. Results and Analysis of IS-TSS

We would make the following conclusions about the evolutionary instance selection algorithms for IS-TSS studying Tables X and XI.

- CHC offers the best behavior in IS-TSS. It presents the best reduction rates and one of the best accuracy rates. The third column in Table X shows that CHC is the best algorithm with respect to reduction rate, as per our study. CHC receives the best result in Table XI covering all the objectives, and is the best in all data sets in respect of reduction and when both objectives are taken into account.
- In IS-TSS, the classical algorithms do not select subsets that generalize correctly (see Ib2 and Rnn in Table X).
- The two classical GAs do not work correctly for medium size data sets. We will discuss why this happens in the next section.

Clearly, when we mine medium-size data sets, the CHC algorithm improves its behavior giving the best results for scaling down data.

VII. KEY POINTS ON THE USE OF EVOLUTIONARY ALGORITHMS FOR DATA REDUCTION

In this section, we discuss some of the key points on the evolutionary instance selection: the execution time (Section VII-A), the scaling up of algorithms (Section VII-B), an analysis of why algorithms behave as they do (Section VII-C), and a final analysis of the simplicity obtained when training a decision tree algorithm with a small instance selection set (Section VII-D).

A. Execution Time

The execution time associated to EAs represents a greater cost than the classical ones, due to the evolution process involved.

However, their application is interesting because non-EAs that have a short execution time offer small reduction rates or, if the reduction rates increase, the algorithm shows overfitting.

The best algorithm behavior corresponds to the CHC, whose time is lower than that of the rest of the EAs, probabilistic ones and some of the classical IS ones.

CHC presents the lowest time amongst EAs due to its efficiency and reduction capabilities. Its chromosomes select a small number of instances from the beginning of the evolution, so the fitness function based on 1-NN has to perform a smaller number of operations.

B. Scaling Up

To test the effect of increasing the data set size, we have evaluated a larger data set than the previous ones. We have taken the Adult data set from the UCI repository, which contains 30 132 instances, 14 attributes, and 2 classes.

The algorithms we have studied, both classical and evolutionary ones, are affected when the size of the data set increases. The main difficulties they have to face are as follows.

- *Efficiency*: The efficiency of IS algorithms is at least $O(n^2)$, where n is the size (number of instances) of the data set. Most of them present an efficiency order greater than $O(n^2)$.

  When the size increases, the time needed by each algorithm also increases.
- *Resources*: Most of the algorithms assessed need to have the complete data set stored in memory to carry out their execution. If the size of the data were too big, the computer would need to use the disk as swap memory. This loss of resources has an adverse effect on efficiency due to the increased access to the disk.
- *Representation*: EAs are also affected by representation, due to the size of their chromosomes. When the size of these chromosomes is too big, the algorithms experience convergence difficulties. An example of this can be seen in Tables X and XI. GAs show convergence difficulties, as well as costly computational time.

The majority of classic algorithms cannot deal with the size of the Adult data set. Algorithms that were able to deal with this are Cnn, Ib2, and Ib3.

EAs were not able to converge to a good solution using chromosomes of that size (in a tenfold cross validation chromosomes would give a size of 27 118 genes).

To be able to apply EAs to large data sets such as the Adult one, we have designed a stratified strategy using the CHC. The CHC has been selected due to its effectiveness as shown in Sections V and VI. The stratified strategy consists of dividing the Adult data set into different strata. We decided to use three strata so that the size of each one would be the same as that used in the pen-based recognition data set.

The Adult data set is divided into three disjoint sets of equal size, T_1, T_2, and T_3. We maintain class distribution within each set in the partitioning process. CHC is applied to each T_i obtaining a subset selected SS_i. We then conduct three pairs of training and test sets, $(\mathrm{TR}_i, \mathrm{TS}_i)$, $i = 1,2,3$. For each pair i, the test set, TS_i, is T_i, and the training set, TR_i, is the union of all of the other $SS_j, j \neq i$. We evaluate each pair $(\mathrm{TR}_i, \mathrm{TS}_i)$ using 1-NN for IS-PS and C4.5 for IS-TSS.

Classical algorithms (Cnn, Ib2, and Ib3) are applied to the complete Adult data set in a tenfold cross validation procedure.

We have included the evaluation of classical algorithms following the stratified strategy (we add to their names "Strat"

TABLE XII
IS-PS FOR ADULT DATA SET

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac Trn	%Ac Test
1-NN	104		79.34	79.24
Cnn	6	99.21	26.40	26.56
Cnn Strat	2	98.56	43.53	28.92
Drop1 Strat	209	95.00	100.00	24.89
Drop2 Strat	182	71.31	26.56	63.70
Drop3 Strat	145	95.69	54.40	66.88
Ib2	5	99.94	25.20	25.14
Ib2 Strat	1	99.88	58.24	25.76
Ib3	21	79.42	72.61	74.09
Ib3 Strat	28	78.22	34.25	72.55
Icf Strat	163	87.16	77.68	66.42
CHC Strat	40391	99.82	85.94	81.95

TABLE XIII
IS-TSS FOR ADULT DATA SET

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	%Ac. Test
C4.5	2		88.72	85.40
Cnn	6	99.21	86.04	24.91
Cnn Strat	2	98.56	84.61	27.30
Drop1 Strat	209	95.00	100.00	24.90
Drop2 Strat	182	71.31	68.59	85.00
Drop3 Strat	145	95.69	75.18	76.27
Ib2	5	99.94	87.39	24.93
Ib2 Strat	1	99.88	100.00	26.23
Ib3	210	79.42	73.09	83.80
Ib3 Strat	28	78.22	73.21	84.43
Icf Strat	163	87.16	87.53	82.00
CHC Strat	40391	99.82	97.88	81.93

word with this menaning). This strategy permits the execution using a smaller data set so we can execute a higher number of classical algorithms.

The results for IS-PS and IS-TSS are presented in Tables XII and XIII.

If we analyze Tables XII and XIII, we would point out the following.

- Classical algorithms Cnn and Ib2 produce elevated reduction rates, but they lose classification capability. Their accuracy rate is reduced significantly (approximately 60% worse than the reduction rate offered by 1-NN or C4.5).
- Ib3 offers an intermediate reduction rate (80%), while its classification rate experiments a small reduction.
- Stratified strategy reduces the execution time, due to dataset reduction, but the algorithms maintain their behavior. The combination of stratified strategy and CHC offers the best result in this difficult problem.
- CHC Stratified overcomes the Ib3 reduction rates (by 20%), maintaining the highest accuracy rate in IS-PS, with a similar accuracy in IS-TSS for the test sets.

CHC with the stratified strategy is the best algorithm evaluated for large size data sets. It offers the best balance between reduction and accuracy rates. Our CHC with stratified strategy produces the smallest subsets maintaining their classification capabilities. However, this algorithm is by far the slowest one, indicating that even with the stratified strategy, processing time remains a problem when mining large data sets.

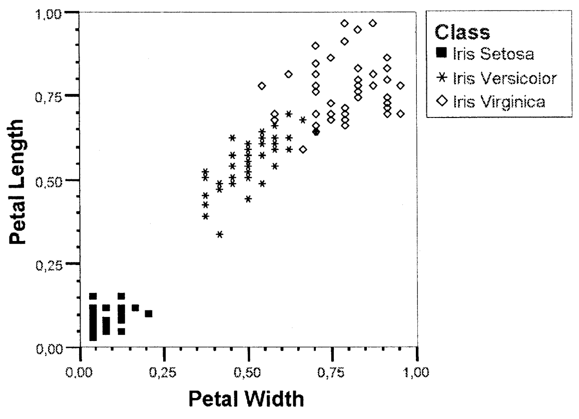


Fig. 5. Original Iris data set.

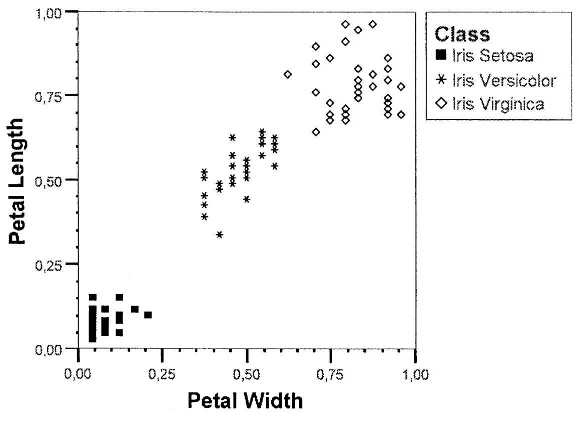


Fig. 6. Data subset selected by Multiedit.

C. An Analysis as to Why Algorithms Behave as They Do in IS-PS

In order to study the behavior of the classical algorithms and EAs, we project the data selected in the Iris data set by each one of the two most discriminative axes—petal length and petal width [19].

We have chosen this data set because it is one of the most studied benchmark data sets, it is well known, and it can be used in a two-dimensional graphical representation.

The following algorithms have been selected as representative in carrying out this study.

- Multiedit: because it gives higher accuracy and small reduction rates among classical algorithms.
- Cnn, Ib2, and Drop2: these are classical algorithms offering the best reduction rates.
- Ib3: due to its balance between reduction and accuracy rates.
- CHC: this is the algorithm offering the best behavior in reduction and accuracy rates.

Figs. 5–11 show graphically the selection strategy followed by each algorithm.

An analysis of these graphs shows the following behavior.

- Multiedit, which is the algorithm with lower reduction rates, tries to maintain the instances situated in the centre of the class, removing those in the boundaries.

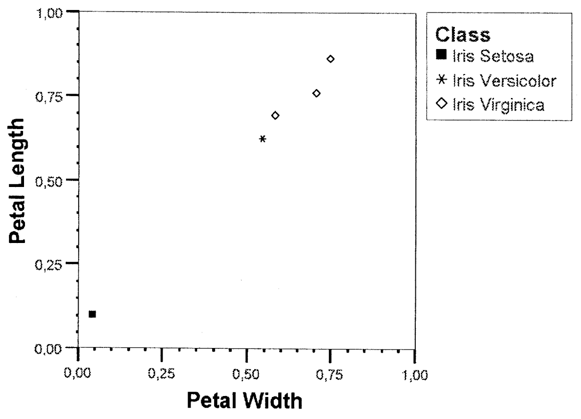


Fig. 7. Data subset selected by Cnn.

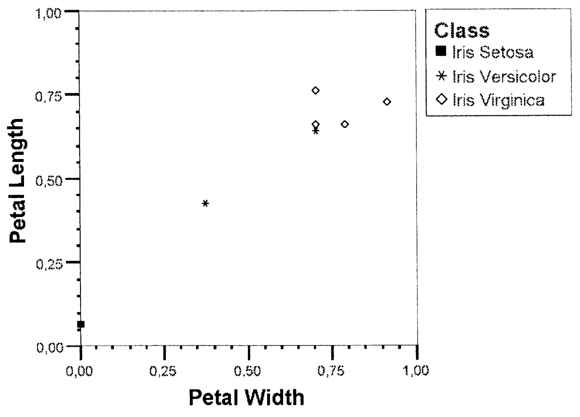


Fig. 8. Data subset selected by Ib2.

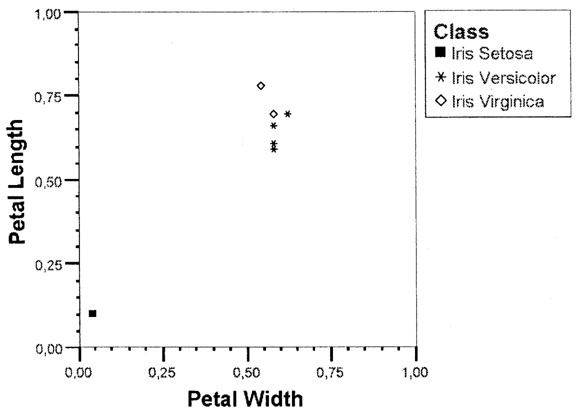


Fig. 9. Data subset selected by Drop2.

- Ib2, Drop2, and Cnn maintain the instances near the boundaries. They reduce the subset selected because the inner instances in the class can be classified using the boundary ones. The problem is that difficulties appear when correctly selecting and classifying the instances near the boundary.
- Ib3 selects more instances from the boundary because of its balance in reduction-accuracy, aiming to maintain enough information to increase classification rates in that zone. Ib3 tries to maintain boundary instances, as can be seen in Fig. 10.

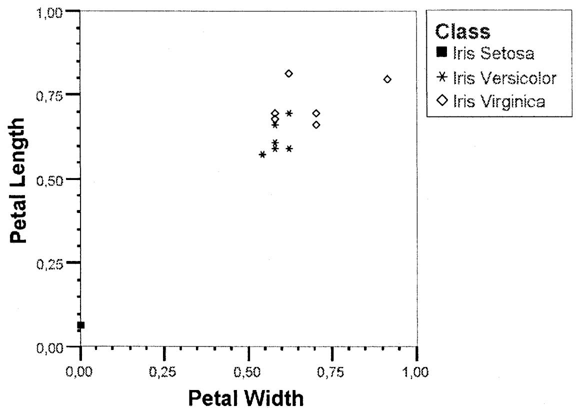


Fig. 10. Data subset selected by Ib3.

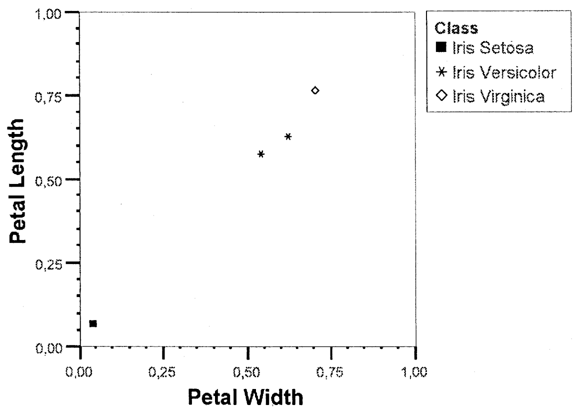


Fig. 11. Data subset selected by CHC.

- CHC presents a strategy that combines inner and boundary points. It does not tend to select instances depending on their a priori position in the search space (inner class or limit ones). CHC selects the instances that increase the accuracy rates, which means that the most representative instances for the whole data set are selected, independently of their a priori position.

This is the reason why CHC produces the best balance between reduction and accuracy rates. CHC is not limited in its selection by a particular strategy, so it can direct its efforts in reducing the subset selected without jeopardizing its accuracy rate.

D. On the Simplicity of Decision Trees Trained With a Small Instance Set Selected by EAs in IS-TSS

The presence of noise in data sets is a widely extended pathology in decision-tree induction [28]. Sources of noise can be error in measurements, error in data encoding, error in examples, missing values, etc. Another potential problem that potentially occurs in the induction of decision trees is the presence of clashes between examples (same attribute vector but different class). Both problems produce complex decision tree, poor comprehensibility, overfitting (decision tree overfits the data), and low classification accuracy of new data.

Larger decision trees introduce a loss of human interpretability of the models.

TABLE XIV
COMPARISON OF THE NUMBER OF RULES IN THE DECISION TREE FOR MEDIUM SIZE DATA SETS

	Pen-Based Recognition	**Satimage**	**Thyroid**
C4.5	192.1	277.5	27.9
CHC	24.4	13.8	7.1

TABLE XV
COMPARISON OF THE NUMBER OF RULES IN THE DECISION TREE FOR ADULT DATA SET

	Adult
C4.5	327.4
CHC Estrat	16.3

Pruning is the common approach to avoid the problem of overfitting noisy data. The basic idea is to incorporate a bias toward more general and simpler theories in order to avoid overly specific theories that try to find explanations for noisy examples. The two different ways [11] to deal with this are the following.

- Prepruning: Stop growing the tree when there is not enough data to make reliable decisions, or when the examples are acceptably homogeneous.
- Postpruning: Grow the full decision tree and then remove nodes for which there is not sufficient evidence.

The drawback of prepruning is that it is very difficult to know when to stop growing the tree.

Classical strategies based on post-pruning tend to underprune or overprune the decision tree.

We have applied EAs in the IS-TSS problem as seen in Sections V-B and VI-B, obtaining greater reduction and accuracy rates. In Table XIV, we show the number of rules offered by the decision tree obtained after CHC execution over the medium size data sets. This is compared with the decision tree obtained after applying C4.5 to the complete medium data sets. The number of rules extracted from a decision tree is the number of leaves in the tree, since each path ffom the root to a leaf correspond to a classification rule.

In the first column of Table XIV, the name of the algorithm is stated and the remaining ones give the average number of rules after the evaluation of both algorithms in a tenfold cross validation process over each one of the medium data sets.

In the Appendix (Tables XXII and XXIII), we can find both decision trees generated using the complete Thyroid data set and the subset selected by CHC.

We have also assessed the effect that the CHC with stratified strategy has on the number of rules discovered in the Adult data set. This effect is shown in the next table:

An analysis of Tables XIV and XV give the following conclusions.

- Table XIV shows the reduction that CHC gives in the model of a decision tree generated by the subset selected. As we can see, the number of rules associated to decision trees are significantly reduced, with a more interpretable model.
- Table XV corroborates the results that we have studied in Table XIV. The application of CHC in its classical or stratified version significantly reduces the size of the decision tree.

Briefly summarizing this subsection, we conclude that when the CHC is applied to IS-TSS problems it produces a small model with a high accuracy rate. This small model size increases its speed in classification, reducing its storage necessities, and increasing its human interpretability.

VIII. CONCLUSION

This paper addressed the analysis of the evolutionary instance selection algorithms and their use in data reduction in KDD.

An experimental study was carried out to compare the results of four EA models together with the classical IS ones from a twofold perspective, the PS and the TSS, over different kinds of data sets. The main conclusions reached are as follows.

- EAs outperform the classical algorithms, simultaneously offering two main advantages: better data reduction percentages and higher classification accuracy.
- The CHC is the most appropriate EA, according to the algorithms that we have compared. It offers the best behavior in IS-PS and IS-TSS when we increase the size of the data set. In addition, it is the fastest EA algorithm according to the study presented in this paper.
- The CHC algorithm significantly reduces the size of the decision tree trained from the selected instances, by comparison with a decision tree trained from all instances. This characteristic produces decision trees that are easier to interpret.
- In medium and larger size data sets, classical algorithms do not present balanced behavior. If the algorithm reduces the size then its accuracy rate is poor. When accuracy increases there is no reduction. The stratified version of CHC offers the best results when the data set size increases too much.

Therefore, as a final concluding remark, we consider EAs to be a good mechanism for data reduction in KDD, and in particular the CHC algorithm. It has become a powerful tool to obtain small selected training sets and therefore, scaling down data. CHC can select the most representative instances independently of their position in the search space, satisfying both the objectives of high accuracy and reduction rates. The main limitation of CHC is its long processing time, as mentioned earlier, which makes it difficult to apply this algorithm to very large data sets.

Finally, we would say that future research could be directed toward the study of the behavior of the evolutionary instance selection algorithms on databases with a large number of instances analyzing the scaling up to bigger data sets. One direction of study that we suggest is the stratified evolutionary model in order to carry out the evaluation of the largest data sets, searching for hybrid strategies between classical and evolutionary instance selection algorithms.

APPENDIX
TABLES FOR EACH DATA SET IN IS-PS AND IS-TSS (TABLES XVI–XXI)

TABLE XVI
IS-PS AVERAGE RESULTS FOR PEN-BASED RECOGNITION

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac Trn	%Ac Test
1-NN	66		99.36	99.39
Cnn (*)	4	98.04	84.85	85.69
Drop1 (*)	374	98.45	86.23	86.02
Drop2 (*)	318	97.69	91.03	91.03
Drop3 (*)	391	98.07	90.33	90.05
Enn	269	0.64	99.40	99.31
Ib2 (*)	2	98.49	74.20	75.04
Icf (*)	537	92.42	89.79	89.51
Mcs	141	0.36	99.54	99.40
Multied	4274	6.47	97.78	97.64
Renn	579	0.66	99.40	99.31
Rnn (*)	33168	97.52	77.97	79.03
Shrink	277	0.64	99.39	99.31
Vsm	107	0.00	99.36	99.39
Ib3 (*)	9	96.42	96.73	98.00
Rmhc (*)	69802	90.02	97.43	97.19
Ennsr (*)	75988	90.02	98.50	98.01
GGA	149281	61.47	99.22	98.82
SGA	103102	61.77	99.35	99.00
CHC (*)	18845	98.99	96.29	98.94
PBIL (*)	83923	73.59	99.77	99.05

TABLE XVII
IS-PS AVERAGE RESULTS FOR SATIMAGE

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac Trn	%Ac Test
1-NN	36		90.33	90.41
Cnn (*)	5	95.93	60.63	61.96
Drop1 (*)	206	93.66	84.29	81.68
Drop2 (*)	183	83.49	83.45	83.45
Drop3 (*)	301	93.25	87.93	81.03
Enn	79	9.67	92.11	90.10
Ib2 (*)	3	96.75	59.00	59.59
Icf (*)	378	84.53	70.18	70.18
Mcs	92	4.31	93.29	90.32
Multied	836	24.82	86.83	86.08
Renn	490	11.05	92.10	89.87
Rnn (*)	4042	95.15	73.26	73.68
Shrink	184	8.86	91.59	90.07
Vsm	99	0.00	90.33	90.41
Ib3 (*)	22	84.66	84.51	86.45
Rmhc (*)	15879	90.02	86.13	85.29
Ennsr (*)	16075	90.02	88.08	88.29
GGA	20245	63.07	91.22	89.94
SGA	21512	63.35	91.76	89.32
CHC (*)	2479	99.06	89.45	89.67
PBIL (*)	6917	72.19	93.75	90.48

TABLE XVIII
IS-PS AVERAGE RESULTS FOR THYROID

	Execution Time(sec)	% Reduction	*1-NN*	
			%Ac Trn	%Ac Test
1-NN	36		92.87	92.74
Cnn (*)	3	98.00	92.50	92.86
Drop1 (*)	182	98.06	63.47	62.86
Drop2 (*)	143	87.54	91.37	91.37
Drop3 (*)	322	97.44	88.82	85.24
Enn	68	7.15	94.78	93.76
Ib2 (*)	2	98.11	92.53	92.89
Icf (*)	244	93.17	70.34	70.35
Mcs	71	3.30	95.05	93.42
Multied	224	10.69	92.67	92.59
Renn	399	7.69	94.62	93.71
Rnn (*)	1841	97.98	92.58	92.51
Shrink	156	5.18	94.58	93.76
Vsm	76	3.12	92.80	92.71
Ib3	94	33.93	93.22	93.38
Rmhc (*)	17895	90.03	91.22	90.98
Ennsr (*)	21342	90.03	91.78	91.95
GGA	28945	63.05	93.78	92.79
SGA	39354	63.60	93.90	92.69
CHC (*)	2891	99.83	94.20	91.98
PBIL (*)	7985	73.61	95.17	92.86

TABLE XIX
IS-TSS AVERAGE RESULTS FOR PEN-BASED RECOGNITION

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	%Ac. Test
C4.5	1		99.27	96.46
Cnn (*)	4	98.04	90.54	64.32
Drop1 (*)	374	98.45	92.31	65.52
Drop2 (*)	318	97.69	93.28	67.89
Drop3 (*)	391	98.07	91.18	61.00
Enn	269	0.64	99.40	96.41
Ib2 (*)	2	98.49	88.18	56.32
Icf (*)	537	92.42	97.45	73.97
Mcs	141	0.36	99.28	96.26
Multied	4274	6.47	99.58	94.80
Renn	579	0.66	99.40	96.37
Rnn (*)	33168	97.52	98.42	67.09
Shrink	277	0.64	99.39	96.39
Vsm	107	0.00	99.27	96.43
Ib3 (*)	9	96.42	91.74	79.63
Rmhc (*)	69802	90.02	98.19	91.40
Ennrs (*)	75988	90.02	97.71	90.40
GGA	149281	61.47	98.91	94.78
SGA	103102	61.77	98.62	94.56
CHC (*)	18845	98.99	98.51	92.33
PBIL (*)	83923	73.59	99.06	94.85

TABLE XX
IS-TSS AVERAGE RESULTS FOR SATIMAGE

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	%Ac. Test
C4.5	1		97.58	86.71
Cnn (*)	5	95.93	91.69	54.83
Drop1(*)	206	93.66	94.39	73.08
Drop2 (*)	183	83.49	94.17	70.76
Drop3 (*)	301	93.25	95.26	67.46
Enn	79	9.67	98.45	86.40
Ib2 (*)	3	96.75	92.26	51.63
Icf (*)	378	84.53	96.96	75.23
Mcs	92	4.31	97.87	86.51
Multied	836	24.82	99.72	84.38
Renn	490	11.05	98.58	86.77
Rnn (*)	4042	95.15	99.07	48.90
Shrink	184	8.86	98.39	86.58
Vsm	99	0.00	97.58	86.71
Ib3 (*)	22	84.66	93.31	75.41
Rmhc (*)	15879	90.02	96.63	85.30
Ennrs (*)	16075	90.02	96.66	80.00
GGA	20245	63.07	97.40	84.16
SGA	21512	63.35	97.48	85.01
CHC (*)	2479	99.06	96.75	84.28
PBIL (*)	6917	72.19	97.28	84.29

TABLE XXI
IS-TSS AVERAGE RESULTS FOR THYROID

	Execution Time(sec)	% Reduction	*C4.5*	
			%Ac. Trn	%Ac. Test
C4.5	1		99.61	99.03
Cnn (*)	3	98.00	98.80	96.95
Drop1 (*)	182	98.06	98.88	96.60
Drop2 (*)	143	87.54	98.81	98.48
Drop3 (*)	322	97.44	98.97	97.52
Enn	68	7.15	99.92	98.67
Ib2 (*)	2	98.11	98.73	96.45
Icf (*)	244	93.17	99.14	97.97
Mcs	71	3.30	99.76	98.97
Multied	224	10.69	100.00	92.64
Renn	399	7.69	99.94	98.36
Rnn (*)	1841	97.98	100.00	92.50
Shrink	156	5.18	99.92	98.57
Vsm	76	3.12	99.66	99.06
Ib3	94	33.93	99.78	98.96
Rmhc (*)	17895	90.03	99.52	98.70
Ennrs (*)	21342	90.03	99.59	98.80
GGA	28945	63.05	99.85	98.85
SGA	39354	63.60	99.05	98.09
CHC (*)	2891	99.83	99.61	98.85
PBIL (*)	7985	73.61	99.78	99.02

TABLE XXII
DECISION TREE FOR THE THYROID DATA SET

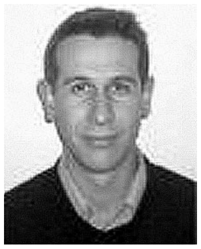
```
attribute#17 <= 0.005 : 3 (5782.0/1.4)
attribute#17 > 0.005 :
| attribute#21 <= 0.064 :
| | attribute#8 <= 0 :
| | | attribute#17 <= 0.017 :
| | | | attribute#20 <= 0.095 : 1 (9.0/1.3)
| | | | attribute#20 > 0.095 :
| | | | | attribute#18 > 0.023 : 3 (6.0/1.2)
| | | | | attribute#18 <= 0.023 :
| | | | | | attribute#17 > 0.009 : 1 (6.0/2.3)
| | | | | | attribute#17 <= 0.009 :
| | | | | | | attribute#18 <= 0.01 : 1 (3.0/2.1)
| | | | | | | attribute#18 > 0.01 : 3 (3.0/1.1)
| | | attribute#17 > 0.017 :
| | | | attribute#9 <= 0 :
| | | | | attribute#18 <= 0.024 : 1 (122.0/1.4)
| | | | | attribute#18 > 0.024 :
| | | | | | attribute#20 <= 0.095 : 3 (2.0/1.0)
| | | | | | attribute#20 > 0.095 : 1 (3.0/1.1)
| | | | attribute#9 > 0 :
| | | | | attribute#1 <= 0.65 : 1 (4.0/1.2)
| | | | | attribute#1 > 0.65 : 3 (2.0/1.0)
| | attribute#8 > 0 :
| | | attribute#20 <= 0.131 : 1 (5.0/2.3)
| | | attribute#20 > 0.131 : 3 (3.0/1.1)
| attribute#21 > 0.064 :
| | attribute#3 > 0 : 3 (107.0/1.4)
| | attribute#3 <= 0 :
| | | attribute#19 > 0.15 : 3 (25.0/1.3)
| | | attribute#19 <= 0.15 :
| | | | attribute#17 <= 0.006 :
| | | | | attribute#2 <= 0 : 3 (56.0/28.1)
| | | | | attribute#2 > 0 :
| | | | | | attribute#19 > 0.088 : 3 (12.0/2.5)
| | | | | | attribute#19 <= 0.088 :
| | | | | | | attribute#1 <= 0.59 : 3 (4.0/2.2)
| | | | | | | attribute#1 > 0.59 : 2 (4.0/1.2)
| | | | attribute#17 > 0.006 :
| | | | | attribute#19 <= 0.061 :
| | | | | | attribute#18 <= 0.019 : 2 (7.0/1.3)
| | | | | | attribute#18 > 0.019 : 3 (6.0/1.2)
| | | | | attribute#19 > 0.061 :
| | | | | | attribute#18 <= 0.03 :
| | | | | | | attribute#8 <= 0 : 2 (288.0/2.6)
| | | | | | | attribute#8 > 0 : 3 (9.0/1.3)
| | | | | | attribute#18 > 0.03 :
| | | | | | | attribute#20 <= 0.113 : 3 (5.0/1.2)
| | | | | | | attribute#20 > 0.113 : 2 (6.0/1.2)
```

TABLE XXIII
DECISION TREE FOR THE SUBSET OBTAINED USING CHC OVER THYROID DATA SET

```
attribute#21 <= 0.062 :
| attribute#17 <= 0.006 : 3 (13.0/1.3)
| attribute#17 > 0.006 : 1 (40.0/3.8)
attribute#21 > 0.062 :
| attribute#17 <= 0.005 : 3 (982.0/1.4)
| attribute#17 > 0.005 :
| | attribute#3 > 0 : 3 (21.0/1.3)
| | attribute#3 <= 0 :
| | | attribute#19 > 0.144 : 3 (5.0/1.2)
| | | attribute#19 <= 0.144 :
| | | | attribute#19 <= 0.06 : 3 (3.0/1.1)
| | | | attribute#19 > 0.06 :
| | | | | attribute#17 > 0.007 : 2 (40.0/2.6)
| | | | | attribute#17 <= 0.007 :
| | | | | | attribute#20 <= 0.099 : 2 (3.0/1.1)
| | | | | | attribute#20 > 0.099 : 3 (4.0/1.2)
```

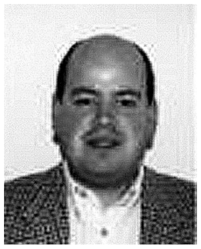
REFERENCES

[1] P. Adriaans and D. Zantinge, *Data Mining*. Reading, MA: Addison-Wesley, 1996.
[2] D. W. Aha, D. Kibbler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, pp. 37–66, 1991.
[3] T. Back, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. London, U.K.: Oxford Univ. Press, 1997.
[4] S. Baluja, "Population-based incremental learning," Carnegie Mellon Univ., Pittsburgh, PA, CMU-CS-94–163, 1994.
[5] H. Brighton and C. Mellish, "Identifying competence-critical instances for instance-based learners," in *Instance Selection and Construction for Data Mining*, H. Liu and H. Motoda, Eds. Norwell, MA: Kluwer, 2001, pp. 77–94.
[6] ——, "Advances in instance selection for instance-based learning algorithms," *Data Mining and Knowl. Dis.*, vol. 6, pp. 153–172, 2002.
[7] C. E. Broadley, "Addressing the selective superiority problem: Automatic algorithm/model class selection," in *Procc. 10th Int. Machine Learning Conf.*, Amherst, MA, 1993, pp. 17–24.
[8] P. Chapman, J. Clinton, T. Khabaza, T. Reinart, and R. Wirth. (1999) The CRISP-DM Process Model. [Online]. Available: www.crisp-dm.org/pub-paper.pdf
[9] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[10] L. J. Eshelman, "The adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetic Algorithms-1*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kauffman, 1991, pp. 265–283.
[11] F. Esposito, D. Malerba, and G. Semeraro, "A comparative analysis of methods for pruning decision trees," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 5, pp. 476–491, 1997.
[12] E. Frank and I. H. Witten, "Making better use of global discretization," in *Proc. 16th Int. Conf. Machine Learning*, I. Bratko and S. Dzeroski, Eds., 1999, pp. 115–123.
[13] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Trans. Inform. Theory*, vol. IT–14, pp. 431–433, May 1972.
[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[15] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inform. Theory*, vol. IT–14, pp. 515–516, May 1968.
[16] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[17] L. Kuncheva, "Editing for the k-Nearest neighbors rule by a genetic algorithm," *Pattern Recognit. Lett.*, vol. 16, pp. 809–814, 1995.
[18] D. Kibbler and D. W. Aha, "Learning representative exemplars of concepts: An initial case of study," in *Proc. 4th Int. Workshop Machine Learning*, 1987, pp. 24–30.
[19] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proc. 7th IEEE Int. Conf. Tools Artificial Intelligence*, 1995, pp. 388–391.
[20] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Norwell, MA: Kluwer, 1998.
[21] ——, "Data reduction via instance selection," in *Instance Selection and Construction for Data Mining*, H. Liu and H. Motoda, Eds. Norwell, MA: Kluwer, 2001, pp. 3–20.
[22] ——, "On issues of instance selection," *Data Mining and Knowl. Dis.*, vol. 6, pp. 115–130, 2002.
[23] D. G. Lowe, "Similarity metric learning for a variable-kernel classifier," *Neural Comput.*, vol. 7, no. 1, pp. 72–85, 1995.
[24] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[25] C. R. Reeves and D. R. Bush, "Using genetic algorithms for training data selection in RBF networks," in *Instance Selection and Construction for Data Mining*, H. Liu and H. Motoda, Eds. Norwell, MA: Kluwer, 2001, pp. 339–356.
[26] T. Reinartz, "A unifying view on instance selection," *Data Mining and Knowl. Dis.*, vol. 6, pp. 191–210, 2002.
[27] J. Reunanen, "Overfitting in making comparisons between variable selection methods," *J. Mach. Learn. Res.*, vol. 3, pp. 1371–1382, 2003.
[28] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Systems, Man, Cybern.*, vol. 21, pp. 660–674, May–June 1991.
[29] J. G. Shanahan, *Soft Computing for Knowledge Discovery*. Norwell, MA: Kluwer, 2000.
[30] D. B. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," in *Proc. 11th Int. Conf. Machine Learning*, 1994, pp. 293–301.
[31] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Systems Man, Cybern.*, vol. SMC–2, pp. 408–421, 1972.
[32] D. R. Wilson and T. R. Martinez, "Instance pruning techniques," in *Proc. 14th Int. Conf. Machine Learning*, 1997, pp. 403–411.
[33] ——, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, pp. 257–268, 2000.
[34] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Mateo, CA: Morgan Kaufmann, 2000.
[35] D. Whitley, "The GENITOR algorithm and selective preasure: Why rank based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. GAs*, Schaffer, Ed., 1989, pp. 116–121.
[36] J. Zhang, "Selecting typical instances in instance-based learning," in *Proc. 9th Int. Conf. Machine Learning*, D. Sleeman and P. Edwards, Eds., 1992, pp. 470–479.



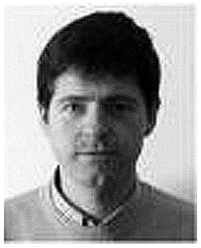
José Ramón Cano received the M.Sc. degree in computer science from the University of Granada, Granada, Spain, in 1999.

He is currently an Associate Professor with the Department of Electronic Engineering, Computer Systems and Automatics, University of Huelva, Huelva, Spain. His research interests include data mining, data reduction, evolutionary algorithms, and a combination of data reduction and evolutionary algorithms.



Francisco Herrera was born in 1966. He received the M.S. and Ph.D. degrees in mathematics from the University of Granada, Granada, Spain, in 1988 and 1991, respectively.

He is an Associate Professor in the Department of Computer Science and Artificial Intelligence, University of Granada. He is coeditor of three books and 13 journal special issues. He is coauthor of *Genetic Fuzzy Systems, Evolutionary Tuning and Learning of Fuzzy Knowledge Bases* (Singapore: World Scientific, 2001). His research interests include decision making problems in fuzzy environment, fuzzy rule-based systems, machine learning, data mining, genetic algorithms, and genetic fuzzy systems.



Manuel Lozano received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 1992 and 1996, respectively.

He is currently an Associate Professor with the Department of Computer Science and Artificial Intelligence, University of Granada. His research interests include fuzzy rule-based systems, machine learning, data mining, genetic algorithms, genetic fuzzy systems, and combination of fuzzy logic and genetic algorithms.