Case-Base Maintenance with Multi-Objective Evolutionary Algorithms

Eduardo Lupiani · Stewart Massie · Susan Craw · Jose M. Juarez · Jose Palma

Abstract Case-Base Reasoning is a problem-solving methodology that uses old solved problems, called cases, to solve new problems. The case-base is the knowledge source where the cases are stored, and the amount of stored cases is critical to the problem-solving ability of the Case-Base Reasoning system. However, when the case-base has many cases, then performance problems arise due to the time needed to find those similar cases to the input problem. At this point, Case-Base Maintenance algorithms can be used to reduce the number of cases and maintain the accuracy of the Case-Base Reasoning system at the same time. Whereas Case-Base Maintenance algorithms typically use a particular heuristic to remove (or select) cases from the case-base, the resulting maintained case-base relies on the proportion of redundant and noisy cases that are present in the case-base, among other factors. That is, a particular Case-Base Maintenance algorithm is suitable for certain types of case-bases that share some indicators, such as redundancy and noise levels.

In the present work, we consider Case-Base Maintenance as a multi-objective optimization problem, which is solved with a Multi-Objective Evolutionary Algorithm. To this end, a fitness function is introduced to measure three different objectives based on the Complexity Profile model. Our hypothesis is that the Multi-Objective Evolutionary Algorithm performing Case-Base Maintenance may be used in a wider set of case-bases, achieving a good balance between the reduction of cases and the problem-solving ability of the Case-Based Reasoning system. Finally, from a set of the experiments, our proposed Multi-Objective Evolutionary Algorithm performing Case-Base Maintenance shows regularly good results with different sets of case-bases with different proportion of redundant and noisy cases.

Eduardo Lupiani · Jose M. Juarez · Jose Palma
University of Murcia, Spain
E-mail: {elupiani,jmjuarez,jtpalma}@um.es

Susan Craw · Stewart Massie
Robert Gordon University, Scotland, UK
E-mail: {s.craw,s.massie}@rgu.ac.uk

Keywords Case-Based Reasoning · Case-Base Maintenance · Multi-Objective Evolutionary Algorithms

1 Introduction

Case-Base Reasoning (henceforth CBR) is a problem-solving methodology that solves new problems by adapting solutions that were used to solve previous problems (Riesbeck (1989); Watson (1998)). CBR uses independent pieces of knowledge, called cases, to represent past solved problems. As time goes by and new problems are solved, then new cases are created and stored in a knowledge source known as the case-base. In this way, CBR reuses the solution of solved problems in the solving of future problems. The CBR cycle is a framework that establishes the steps required to perform the complete case-base reasoning process (Aamodt and Plaza (1994)). This cycle consists of four sequential steps. First, the *retrieve* step returns the most similar cases to the new problem. Second, the *reuse* step proposes a new solution based on the retrieved cases. Third, the *revise* step verifies the returned solution if this is necessary. Finally, the forth step is the creation of a new case that may be *retained* in the case-base. This last step is one of the advantages of the CBR methodology, because retaining new cases as problems are solved, increases the case-base size and has the potential of improve the problem-solving ability of the CBR system.

However, retaining a large number of cases may cause performance problems, and may even damage the problem-solving ability of the CBR system (Pan et al (2007); Smyth and Keane (1995)). Given an input problem, the retrieval time to look for similar cases could be large enough to outweigh the benefits of applying this knowledge. In addition, a large number of cases may also complicate the adaptation process unnecessarily (Smyth and Keane (1995)).

Unfortunately, the use of brute force for searching for the best case-base with both low number of cases and error rate is not possible due to the huge amount of possible combinations of cases. Moreover, finding that case-base in this way from the existing data is at best representative of the type of problems that will be faced, but actual problems will be different and systems change over time further challenges the representative assumption of the problem domain. Consequently, the use of brute-force may well find over-fitted case-bases.

Instead of brute force, it is possible to use Case-Base Maintenance algorithms (henceforth CBM algorithms). The CBM algorithms have, among their objectives, finding a case-base smaller than the original, but with similar problem-solving capabilities (Leake and Wilson (1998)). Nonetheless, the maintenance domain is a weak theory domain and it is difficult to determine what is the ideal or optimum number of cases to achieve the best results (Leake and Wilson (2011)). For this reason, the CBM algorithms use heuristics to select those cases that will be part of the maintained case-base, where these heuristics are designed to remove (or select) either redundant or noisy cases (Massie et al (2005, 2006)). Therefore, whereas some more aggressive CBM algorithms try to find case-bases within or close to the knee bend area, other more conservative CBM algorithms search for a case-base within the stable error region.

The suitability of one particular CBM algorithm relies on how well the heuristic suits the characteristics of the domain, such as, the proportion of redundant and noisy cases within the case-base. In addition, the order in which the cases are explored may also affect the resulting maintained case-base (Pan et al (2007); Wilson and Martinez (2000, 1997)), even when the same CBM heuristic is applied. With this in mind, usually an evaluation is done to study the maintained case-bases for the considered CBM algorithms, and to chose the most suitable maintained case-base for the particular problem domains.

Our approach is to consider CBM as a multi-objective optimization problem, in such a way that the proposed algorithm may be used with any type of case-base. The purpose of this CBM is to generate a case-base with low number of redundant cases and few noisy cases, minimizing three objectives based on Complexity Profiling (Massie et al (2005, 2006)). In the last decades, Multi-Objective Evolutionary Algorithms (MOEAs) have been applied successfully in multi-objective optimization problems (Coello et al (2007); Eiben and Smith (2003); Yu and Gen (2010)). Therefore, our approach is to solve the optimization problem with MOEA in order to get an effective and well-maintained case-base irrespective of the redundancy and noise levels present in the original case-base.

The rest of the paper is organised as follows. The next section gives an overview of existing work. Section 3 describes how to represent the levels of redundant and noisy cases of a case-base. Section 4 explains how to perform CBM with a MOEA that optimises three different types of objectives. In section 5, we evaluate the MOEA with different case-bases, and other CBM algorithms. Lastly, sections 6 and 7 discuss the experimental results and highlight our conclusions, respectively.

2 Background

2.1 Case-Base Reasoning

Case-Based Reasoning (CBR) is a methodology that makes use of past experiences to solve new problems (Lopez de Mantaras et al (2005)). In a CBR system the atomic unit of knowledge is the *case*, the knowledge of previously experienced, specific problem situations (Aamodt and Plaza (1994)). The representation of a case is basically a tuple composed of two descriptions that characterise a problem and its solution. When multiple cases are created to solve problems in a given domain, they together form the knowledge base from which the CBR system can perform its problem-solving process (Aamodt and

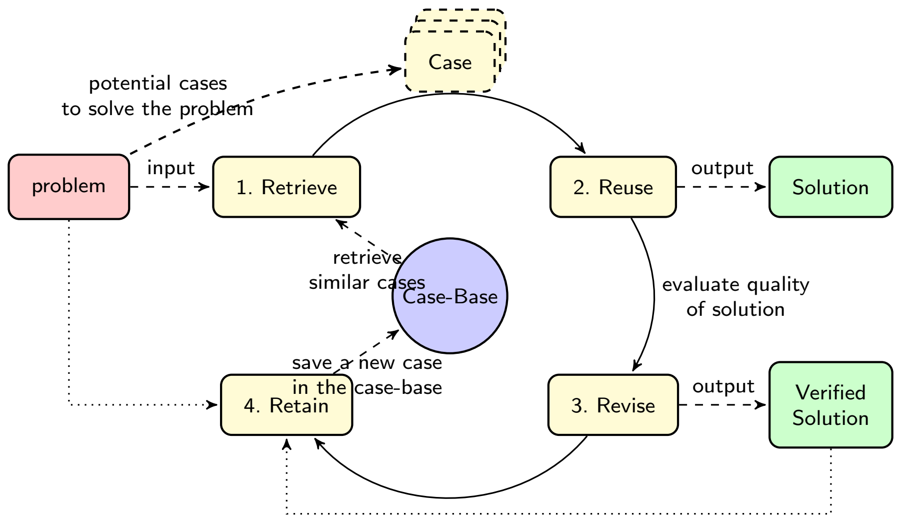

Fig. 1 The CBR cycle and its four steps (adopted from Aamodt and Plaza (1994)). The CBR system using analogy-based reasoning to either build or adapt a solution to solve the input problem.

Plaza (1994)). Within the CBR community, the knowledge base is known as the *case-base*.

Despite the fact that multiple alternative representations of a case are possible, the simplest case representation is based on a vector of attributes, which describes the problem, and a single attribute describing the solution. Other complex representations are possible, such as workflows (Gil (2012)), signals (Montani et al (2006); Olsson et al (2004)), time sequences (Juarez et al (2009)) or graphs (Bunke et al (2005)) among others.

The most commonly used implementation of a CBR system was proposed in (Aamodt and Plaza (1994); Lopez de Mantaras et al (2005)). Figure 1 depicts this process, which consists of a four-step process: (1) *Retrieve*: the system searches for those cases that have a problem description with high similarity to the input problem description, for instance using a k-nearest neighbour algorithm (Cover and Hart (1967)); (2) *Reuse*: using the cases retrieved, the system builds a solution to solve the input problem; (3) *Revise*: the system checks whether the solution proposed in the previous step is correct and, finally, (4) *Retain*: a new case is created with the description of the input problem and the output solution, and this is stored in the case-base.

One of the most important characteristic of this CBR model is that its learning process is incremental and continuous, since new solved cases are added to the case-base, thus the learning step is integrated into the problem solving process itself.

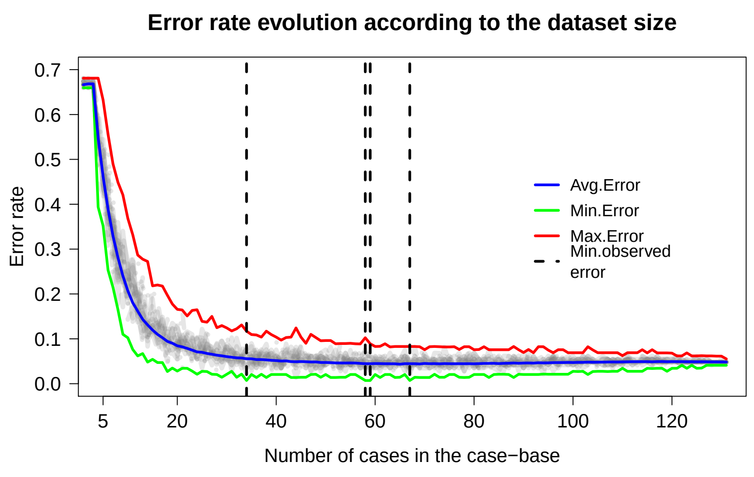

Fig. 2 Evolution of the error rate for a sample dataset.

2.2 Utility Problem

Retaining new cases in the case-base is one straightforward approach to improve the system's expertise while solving problems of the domain. However, if too many of them are retained, the retrieval time may be negatively affected when fetching the most similar cases to the input problem. This scenario exhibits aspects of the *utility problem* (Francis and Ram (1993)), which appears when the size of the case-base is detrimental to the scalability of the system. On top of that, when only redundant cases are retained in the case-base, the accuracy of the system is not improved.

Figure 2 shows the error rate on a sample CBR system as the number of cases in the case-base increases. To build the figure, we have created 1000 case-bases for each case-base size by selecting cases randomly from the set of all the known cases. Therefore, we have 1000 case-bases of 5 cases, 1000 cases-bases of 6 cases, and so on. The error rate of each case-base is measured, using 40% of the data as the test set. Finally, for each set of 1000 cases-bases the test set error rate is averaged. The axes represent the error rate and the number of cases in the case-base. The top and bottom lines represent the worst and best error rate for the particular number of cases, respectively, and the middle line is the average error rate. The shading part of the graph is the error rates given by the different combinations of cases for the corresponding case-base size. Finally, the vertical dashed lines represent each time that the minimum observed error rate from the sample is reached.

According to the figure, whereas increasing the number of cases results in a reduction of the error up to certain limit, it is not clear at what number of cases the error rate stops falling. In addition, the variance of the error rate, represented as the distance between the maximum and minimum observed error rate, reduces as the number of cases increases. Hence, even though it is possible to find case-bases with both low number of cases and error, those case-bases are in the region of high variance of the error rate and are likely to be over-fitted to the known set of cases. Actually, they may not be good choices to solve future unseen problems in this domain. In this way, the case-bases located within or close to the knee bend area in Figure 2 are less consistent and minor changes may have a big impact on error rates, hence the over-fitting to the existing case-base. In contrast, the more stable error rate region with more cases contains maintained case-bases that are representative of the full problem domain. On top of that, despite of the fact that the error rate of the system is lower with higher number of case within the case base, simultaneously with the larger case-bases the CBR system has higher retrieval time to fetch the most similar cases, thus retaining new cases may end in a utility problem.

2.3 Case-Base Maintenance

According to Leake & Wilsons definition of maintenance: "Case-Base Maintenance implements policies for revising the organization or contents (e.g. representations, domain content or accounting information) of the case-base in order to simplify the future problem-solving process subjected to a particular set of performance objectives" (Leake and Wilson (1998)). In particular, CBM may be used to reduce the case-base size or to improve the quality of the solutions provided by the CBR system (Pan et al (2007); Wilson and Martinez (2000)), and the maintenance may be done either by hand, where an expert of the domain edits the case-base, or automatically by using CBM algorithms.

The importance of CBM is reflected by its inclusion into some CBR-cycles, like the proposed in (Göker and Roth-Berghofer (1999)), where the traditional cycle is divided into two different cycles, the *Application and Maintenance cycles*. Whereas the Application cycle contains the traditional steps Retrieve, Reuse and Revise, the Maintenance Cycle substitutes the Retain step with the maintenance tasks as well as the retain step. Usually the Application cycle is executed every time that an input problem is solved, and the Maintenance cycle is executed when the case-base need to be improved.

CBM algorithms has been studied in depth in Machine Learning disciplines such as Instance-based Learning, Exemplar-based Learning, as well as in Case-Based Reasoning. There are a wealth of approaches to perform CBM, such as those published in (Brighton and Mellish (1999); Gates (1972); Massie et al (2006); Pan et al (2007); Smyth and Keane (1995); Wilson (1972); Wilson and Martinez (2000)). These algorithms usually try to classify the cases within the case-base as redundant or noisy, and delete them according to a specific deletion policy. A case could be considered as noisy when most of its neighbours have different solutions, otherwise it would be considered as redundant (Massie et al (2005)). The task of identifying a redundant case is easier than determining whether a case is noisy. A case surrounded by cases with a different solution could be due to an error in the description, or it could simply be an exception (Rissland (2009)). For example, in a case-base that contains cases describing birds, we can describe a swan as a bird with white feathers, although there are infrequent black swans as well. In this way, it is incorrect to delete those cases that describe a swan as a bird with black feathers.

Some authors have proposed different taxonomies of CBM algorithms in an attempt to enhance our understanding of them. For example, (Pan et al (2007)) classify the different CBM algorithms according to the following features: (i) *Case Search Direction:* to build a maintained case-base, a CBM algorithm may start with an empty case-base and continues adding cases to it from the original case-base until a termination condition is reached. This is known as *Incremental* CBM; otherwise, the CBM algorithm is known as *Decremental.* (ii) *Order sensitive:* when the case-base is sorted in some way and this order determines the sequence in which cases are processed then the CBM is *Order Sensitive*; otherwise the CBM algorithm is known as *Order Insensitive.* (iii) *Selection Criteria:* when the decision to include a case in the case-base or not relies only on the local case parameters. For example, considering the solution given to its neighbours, then the criteria is *local*, however when all the cases are involved then it is *global.* Wilson and Martinez (Wilson and Martinez (2000)) propose an additional feature: (iv) *Type of cases to retain:* here, there are two approaches: retain those cases that have at least one case within their nearest neighbourhood with a different solution (border cases), or retain cases that have all their nearest neighbours with the same solution as themselves (central cases). In the first scenario, the maintained case-base retains cases to mark out borders, while in the second, the CBM algorithm is aimed at deleting the redundant cases.

Other classifications can be made according to whether a CBM algorithm is deterministic or non-deterministic. Whereas a deterministic CBM always generates the same maintained case-base from a given case-base, a non-deterministic CBM builds different maintained case-bases each time that the algorithm is executed. Most of the CBM algorithms in the literature are deterministic because their deletion policy is fixed and constrained by the order of cases in the original case-base, and because they always apply the same action (removal) when they classify a case as noisy or redundant.

2.4 The CBM Algorithms

The simplest CBM algorithm consists of a random deletion of cases until the case-base reaches a certain number of cases (Markovitch and Scott (1988)). However, CBM algorithms generally try to select cases in such a way that the CBR system using the maintained case-base has better or equal problem-solving capabilities than when the original case-base is used. To summarize the CBM algorithms proposed in the literature, we propose a classification based on four families: NN algorithms, Instance-based algorithms, DROP family, Competence and Complexity models.

Some algorithms are actually composite methods, whereby the maintenance task is divided into two steps. The first step is usually aimed at reducing the amount of noisy cases, and the second step aims to remove unnecessary or redundant cases.

Algorithms from the NN family select cases according to a nearest neighbour policy. One of the first attempts to reduce case-base size was CNN (Hart (1968)). Starting with a random case of each existing solution as initial case-base, the algorithm uses the rest of them as a test set and classifies them using the selected cases with a k-NN classifier. If a case is misclassified then it is added to the final case-base. The process stops when all the original cases are correctly classified. Although size reduction is possible, this algorithm does not check for noisy cases. RNN (Gates (1972)) was introduced as an extension of CNN to remove noisy instances from the resulting case-base after the use of CNN. Each case of the final case-base is removed, and if no case from the original case-base is misclassified then the candidate is finally removed, otherwise the case is added again. ENN removes misclassified cases according to the solutions of their three nearest neighbours (Wilson (1972)). When ENN is executed multiple times, taking each output as input of the next execution, then the method is called RENN. All-KNN consists of executing ENN k times, where each execution uses from 1 to k neighbours respectively to flag a case for no selection (Tomek (1976)). Some authors claim ENN and its variations are actually noise removal techniques (Wilson and Martinez (2000)).

The Instance-based family consists of algorithms that represent cases as instances (a data structure with a vector of features and an attribute class), and could be understood as a simplified representation of a case. IB2 and IB3 algorithms modify the IB1 classifier to perform CBM (Aha et al (1991); Aha (1992)). In the IB2 algorithm, if a case is misclassified by its nearest neighbours then this case is added to the final case-base. The IB3 algorithm includes a more restrictive condition to keep a selected case inside the final case-base by reducing noise. Shrink algorithm executes CNN and, finally, it removes from the resulting case-base those cases misclassified by their nearest neighbours (Kibler and Aha (1987)).

The CBM algorithms DROP1, DROP2 and DROP3 belong to the DROP family (Wilson and Martinez (2000)). All these methods introduce the concept of associate case, which is a case within the set of nearest neighbour cases with the same solution. In DROP1, a case is removed if most of its associates already in the maintained case-base are solved correctly by the CBR system with the case-base without it. In DROP2, a case is removed if most of its associates in the original case-base are solved correctly without it. DROP3 uses ENN to remove the noisy cases before executing DROP1.

The algorithms from the Competence Model and the Complexity Profiling Family are distinguished from the aforementioned families because they explore the implicit information of the case-base to build a maintenance model. The *Competence Model* (Smyth and Keane (1995)) defines concepts as Coverage, Reachability and Relative Coverage (Smyth and Mckenna (1999)). For each concept, a new CBM algorithm is proposed. COV and RFD use Coverage

Set and Reachability cardinality, respectively, to sort the case-base, and RC uses the Relative Coverage model. Finally, the sorted case-base is the input to CNN, which performs the maintenance. Another approach based on the competence model is the ICF algorithm, which uses the concepts coverage and reachability among cases to decide whether to remove a case from the case-base or not (Brighton and Mellish (1999)). *Complexity Profiling* estimates the proportion of redundant and noisy cases, as well as the existing error rate in the case-base (Massie et al (2005, 2006)). The basis of this approach is a local complexity measure that provides the probability of finding another case in the nearest neighbourhood of a case with the same solution. Although the local complexity measure is useful for case discovery (Massie et al (2005)), it does not provide enough information to evaluate the case-base competence.

2.5 Evolutionary Approach to perform CBM

The main objective of the CBM algorithms is to find the smallest subset of cases that provides the CBR system with a good problem-solving capability, usually through the removal of irrelevant or redundant cases. This problem can be modelled as a multi-objective optimization problem: minimising the number of cases within the case-base and maximising the CBR system accuracy. In this sense, Evolutionary Algorithms (EA) have been recognised as appropriate techniques for multi-objective optimisation because they perform a search for multiple solutions in parallel (Coello et al (2007)). Current evolutionary approaches for multi-objective optimisation include multi-objective EAs based on the Pareto optimality notion, in which all objectives are optimised simultaneously to find multiple non-dominated solutions in a single run of the EA. For example, in the works (Ahn et al (2007); Ishibuchi et al (2001); Kim and Han (2001)), the authors use Evolutionary Algorithms to perform CBM. Such evolutionary approaches are non-deterministic CBM algorithms, because each execution generates a different maintained case-base from the original case-base.

2.6 Evolutionary Algorithms

The Evolutionary Algorithms (EAs) are inspired in biological evolution (Holland (1975)), since they simulate biological processes to search for a solution to an optimization problem. According to (Yu and Gen (2010)), the main features of this type of algorithms are:

(i) Population-based: every EA has a population, which is a set of individuals that represent solutions to the given optimization problem, and these individuals contain genes to represent one particular solution. The population is important because it allows the EA to work with many solutions of the problem at the same time.

(ii) Fitness-oriented: every individual in the population have assigned a fitness value given by a fitness function, where this value is computed according to the genes of the individual. The fitness value is critical because of the fact that the individuals with better fitness values are better solutions to the optimization problem. Once the EA finishes, the final solution is typically one of the individuals of the population, although the solution may be proposed using all the final individuals.

(iii) Variation-driven: through the simulation of biological processes, such as crossing between individuals and mutation of genes, the EA creates a new generation of individuals that could solve the optimization problem in a better way. These two processes are important because they create new solutions to the problem.

Whereas there are many ways for representing the solution within an individual, a basic approach is representing the problem with a string of binary values (Eiben and Smith (2003); Yu and Gen (2010)). Hence, the string is known as an individual, and each of its binary values are the genes. For each individual the EA applies a function known as the fitness function, which indicates the suitability of the individual to resolve the optimization problem. The search for the best individual is an iterative process. Starting with a set of individuals known as the population, an EA uses three operations on it to create the next generation of individuals: reproduction, crossover and mutation. The reproduction operation aims to select the better individuals according to their fitness values. Crossover is applied only to selected individuals to create new individuals, usually exchanging their genes. Mutation flips randomly the genes of the individual to increase the diversity of individuals. At the end of the iteration process, the individuals within the final population are potential solutions to the optimization problem. Hence, a strategy is needed to choose the final solution as well.

Multi-Objective Evolutionary Algorithm (MOEA) is an EA that searches for a solution to a problem according to two or more optimization objectives (Coello et al (2007)). Unlike an EA, a MOEA's fitness function returns a value per each objective (Zitzler and Thiele (1999)). Expression 1 defines formally the optimization problem to minimize n objectives:

$$minimize(\mathit{\Phi}(x)) = minimize(\phi_1(x), \phi_2(x), \dots, \phi_n(x)), \tag{1}$$

where x is an individual, $\mathit{\Phi}$ is the fitness function, and each ϕ_n is the fitness function associated with an objective. Given the fitness values of two individuals, it is possible to define a relation of dominance between them (Deb et al (2002)). This dominance determines which individual is closer to the optimization objectives. Expression 2 defines formally the relation:

$$\begin{aligned} x \prec y \iff \\ \forall \phi_i(x), \phi_i(y) \in \mathit{\Phi}(x) : \phi_i(x) \leq \phi_i(y) \wedge \\ \exists \phi_j(x), \phi_j(y) \in \mathit{\Phi}(x) :\ \phi_j(x) < \phi_j(y), \end{aligned} \tag{2}$$

where x and y are the individuals, $x \prec y$ expresses that x dominates y, and n is the number of objectives. MOEA generates generations of individuals, where non dominated individuals have higher odds of survival.

3 Representing the Redundancy and Noise Levels of a Case-Base

Massie et al (2006) introduced Complexity Profiling to estimate the proportion of redundant and noisy cases, as well as the existing error rate in the case-base. The foundation of this approach is a local complexity, which is an approximation to find the proportion of cases with the same solution in the nearest neighbour set of the case. Expression 3 describes the complexity function for a case:

$$complexity(c, k) = 1 - \frac{1}{k} \sum_{i=1}^{k} p(c, i), \tag{3}$$

where k is the number of nearest neighbours to consider and $p(c, i)$ is the proportion of cases within the case's i-nearest neighbours that belong to the same solution as c. The co-domain for *complexity* function is $[0, 1]$. The more the *complexity* of a case is, the more likely the case would be noisy.

Complexity Profiling is a global measure of the case-base, and it is composed by three different indicators:

1. the *error rate* is the average of all the local complexities measures;
2. the *noise* is the proportion of all the complexity measures with values greater than ϵ; and
3. the *redundancy* is the proportion of all the complexity measures with values equal to ρ.

The error, noise and redundancy are defined formally as follow:

$$error(M, k) = \frac{1}{|M|} \sum_{c \in M} complexity(c, k). \tag{4}$$

$$noise(M, k) = \frac{|\{c \in M | complexity(c, k) \geq \epsilon\}|}{|M|}. \tag{5}$$

$$redundancy(M, k) = \frac{|\{c \in M | complexity(c, k) = \rho\}|}{|M|}, \tag{6}$$

where M is a case-base, $c \in M$ is a case within M, and k is the number of neighbours of c. Experiments with $\epsilon = 0.5$ and $\rho = 0$ confirm that Complexity Profiling is correlated with the accuracy of the CBR system (Massie et al (2006)).

From the local complexities it is possible to create graphically the complexity profile of a given case-base. The cases are ranked in ascending order of complexity along the horizontal axis and the local complexity of each is plotted

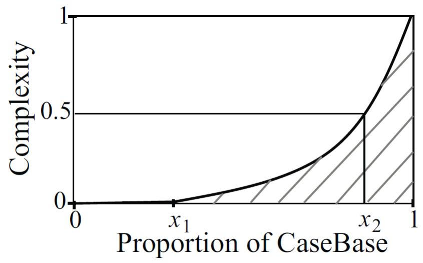

Fig. 3 Complexity Profile of Case-Base

on the vertical axis. Figure 3 shows a typical profile. With $\epsilon = 0.5$ and $\rho = 0$, the distance between 0 and where the curve breaks from the axis (x_1) is the proportion of redundant cases. The shaded area under the curve corresponds to average case complexity and estimates expected error rate. The proportion of potentially noisy cases is the distance between x_2 and 1.

4 Perfoming CBM with a MOEA

So to perform CBM with a MOEA, we need to set up two important objects. On the one hand, we need to represent a case-base as an individual of the population. On the other hand, we need to define a fitness function to evaluate the suitability of the individual for being an adequate maintained case-base.

4.1 Case-base Representation

Every individual in the population is a string of genes of binary type (so each gene may have either the values true of false), and the length of the string is equal to the number of cases in the original case-base. In this way, all the cases in the original case-base are assigned with an index value i, and the i-th gene of the individual represents whether the case with index i in the original case-base is retained in the maintained case-base that is represented by the individual.

Formally, let M be the original case-base, denoted by $M = \{c_1, c_2, \ldots, c_n\}$, where c_i is the i-th case of M ($|M| = n$), and $\wp(M)$ be the set of all the possible sub-case-bases that may be built from M. An individual $x = x_1x_2 \ldots x_{n-1}x_n$ is a string of genes with the same length as M, where each gene x_i, with values of $x_i \in \{true, false\}$, represent whether the case c_i is retained or not, respectively, in the maintained case-base. Let X be the set of all the possible individuals, so $x \in X$.

In order to map the cases from the original case-base (M) to the elements of the individual, we introduce the following function:

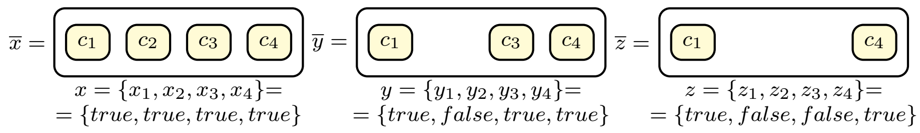
$\overline{x} =$
c_1
c_2
c_3
c_4
$x = \{x_1, x_2, x_3, x_4\} =$
$= \{true, true, true, true\}$
$\overline{y} =$
c_1
c_3
c_4
$y = \{y_1, y_2, y_3, y_4\} =$
$= \{true, false, true, true\}$
$\overline{z} =$
c_1
c_4
$z = \{z_1, z_2, z_3, z_4\} =$
$= \{true, false, false, true\}$

Fig. 4 Three case-bases $\overline{x}, \overline{y}$ and $\overline{z}$ and their corresponding individuals x, y and z.

$$\begin{aligned} \mathcal{M} : \quad & X \to \wp(M) \\ \mathcal{M}(x) = \overline{x} = & \{c_i \in M | x_i = true\}. \end{aligned} \tag{7}$$

For example, given the individual x with all elements set to true, $\mathcal{M}(x) = M$, otherwise if all elements are set to false then $\mathcal{M}(x) = \emptyset$. For the sake of clarity, we use the notation $\overline{x}, \overline{y}, \overline{z}$ as the case-base equivalent to the individuals x, y, z, respectively. Figure 4 depicts graphically how three different cases-bases $\overline{x}, \overline{y}, \overline{z}$ and their respective individuals x, y, z that represent them, where $\overline{x}$ is the original case-base M, and the case-bases $\overline{y}$ and $\overline{z}$ are sub-case-bases of $\overline{x}$.

4.2 Fitness Function to Perform CBM

We propose a fitness function based on Complexity Profiling to solve an optimization problem with three objectives:

– $\mathbf{O_{size}}$: to minimize the difference between the current number of cases in the solution and the estimated number of non-redundant cases;
– $\mathbf{O_{reduncancy}}$: to minimize the number of redundant cases; and
– $\mathbf{O_{error}}$: to minimize the error rate level.

First, the objective $\mathbf{O_{size}}$ aims to estimate the minimum number of cases. Second, the objective $\mathbf{O_{redundancy}}$ is focused on avoiding case-bases with redundant cases. Finally the third objective $\mathbf{O_{error}}$ leads the search to find a case-base with the minimum error rate. According to these objectives, the resulting case-base is expected to have lower proportion of redundant and noisy cases.

At this point, given the set X of all the feasible individuals, and a number of neighbours $k \in \mathbb{N}$, the formal description of the fitness function Φ is shown as follows:

$$\begin{aligned} \Phi : \quad & X, \mathbb{N} \to \mathbb{R}^3 \\ \Phi(x,k) = \quad & (O^x_{size}, O^x_{redundancy}, O^x_{error}) = \\ = & \begin{cases} (O^x_{size} = f_{size}(x,k), \\ O^x_{redundancy} = redundancy(\overline{x},k), \\ O^x_{error} = error(\overline{x},k)) \end{cases} \end{aligned} \tag{8}$$

Note, that we use the notation O^x to refer to the fitness values of the individual x. For instance, O^x_{error} denotes the fitness value of the objective O_{error} for the individual x.

The functions that assign the fitness values to each objective needs to be defined too. Therefore, the function f_{size} is defined as follows:

$$\begin{aligned} f_{size}: &\quad X, \mathbb{N} \to \mathbb{R} \\ f_{size}(x,k) = &\ (threshold(M) - length(x))^2, \\ threshold(M) = &\ (|M| * (1 - redundancy(M,k)). \end{aligned} \tag{9}$$

where $length(x)$ is the number of genes of x set to true and $threshold(M)$ is the estimation of number of non-redundant cases in the original case-base M, which we call *redundancy threshold* and it is always constant for every maintained case-base from the original case-base. Therefore, the function f_{size} is the distance between the current number of cases in the case-base $\overline{x}$ and the redundancy threshold. This objective is squared to penalize those individuals with a greater number of cases. The values returned by functions $redundancy(\overline{x}, k)$ and $error(\overline{x}, k)$ in the fitness function (expression 8) oppose each other since a lower error rate often results in a higher redundancy and vice versa.

4.3 Dominance between Individuals.

Because we are introducing CBM as a multi-objective optimization problem in which the objectives conflict with each other, that is, improving one of the objectives may cause the worsening of the rest, then it is complicated to determine which individuals is the best for solving the problem. With the dominance operator $\prec$ between individuals, the MOEA has a mechanism to discern the closer individuals to the optimal solution. Given two individuals x and y representing two case-bases, and the fitness function Φ, the dominance relation for NSGA-II is defined as:

$$\Phi(x) \prec \Phi(y) \iff \tag{10}$$

$$\iff \begin{cases} \left(O^x_{size} \leq O^y_{size} \land O^x_{redundancy} \leq O^y_{redundancy} \land O^x_{error} \leq O^y_{error}\right) \land \\ \land \left(O^x_{size} < O^y_{size} \lor O^x_{redundancy} < O^y_{redundancy} \lor O^x_{error} < O^y_{error}\right) \end{cases}$$

For the sake of clarity we have omitted the parameter k from the function Φ .

4.4 NSGA-II

In this work we use the MOEA NSGA-II (Deb et al (2002)) to perform CBM. The main contributions of NSGA-II are a fast non-dominated sorting function and two operators to sort the individuals: a density estimation of the individuals in the population covering the same solution, and a crowded comparison operator.

The *fast-nondominated-sort* algorithm, details shown in Algorithm 1, given a population P returns a list of the non-dominated fronts $\mathcal{F}$, where the individuals in front $\mathcal{F}_i$ dominate those individuals in front $\mathcal{F}_{i+1}$. That is, the first front contains the non-dominated individuals, the second front has those individuals dominated only once, the third contains individuals dominated up to twice, and so on. The individuals in the same front could have similar case-bases; to avoid this situation NSGA-II uses the crowded comparison operator $\geq_n$, where are preferred those individuals that represent infrequent solutions in the population set. That is, if the algorithm has to choose among two individuals of the same front, the individual with the the most infrequent case-base in the population is chosen, even if that individuals represents a worst maintained case-base. The purpose is to increment the sight of the algorithm in order to find alternative individuals, which could steer the search to better maintained case-base in the future. To define formally the operator $\geq_n$, let x, y be two individuals, then $x \geq_n y$ if $(x_{rank} < y_{rank})$ or $((x_{rank} = y_{rank}) \wedge (x_{density} > y_{density})$, where x_{rank} represents the front where the individual belongs. The *crowding-distance-assignment* procedure calculates the density per each individual (Algorithm 2).

Some parameters have to be set up at the beginning, such as the number of generations and the number N of individuals in a population. Each generation t implies an iteration of the algorithm, where two populations P_t and Q_t of N individuals are used. When NSGA-II starts, the initial population P_0 is generated randomly. Binary tournament selection, recombination, and mutation operators are used with individuals from P_0 to create a child population Q_0. Once P_0 and Q_0 are initialized, NSGA-II runs its main loop, which we can see in Algorithm 3. In each iteration, population P_t and Q_t are joined to create the population R_t, whose number of individuals is $2N$. After that, the individuals in R_t are sorted according to their dominance and crowding distances. The sorted individuals are added to population P_{t+1}. At the end of each iteration P_{t+1} is truncated to N individuals, and Q_{t+1} is generated using binary tournament selection, recombination, and mutation operators. These operations are explained below, in subsection 4.5.

Once NSGA-II finishes, the final population P_t will contain as many individuals as potential solutions, and the non-dominated individuals are mapped to their corresponding case-bases. The case-base with the minimum error rate is chosen as the solution of the CBM algorithm. If two or more case-bases have the same error rate, then the algorithm chooses the first case-base found.

Figure 5 depicts graphically the process for creating a new generation of individuals. From the population R_t, the individuals are sorted by non-dominance between them. Later the set $\mathcal{F}$ of fronts are created, where $\mathcal{F}_1$ are non dominated individuals, $\mathcal{F}_2$ are dominated by the individuals in $\mathcal{F}_1$ and so on. Finally, the population P_{t+1} is used to create the population Q_t. For further details of NSGA-II algorithms see (Deb et al (2002)).

Algorithm 1 fast-nondominated-sort(P)

```
Require: A population $P$, a fitness function $\Phi$
Ensure: list of the non-dominated fronts $\mathcal{F}$
 1: for $x \in P$ do
 2:     for $y \in P$ do
 3:         if $\Phi(x) \prec \Phi(y)$ then
 4:             $S_x \leftarrow S_x \bigcup \{q\}$
 5:         else
 6:             if $\Phi(y) \prec \Phi(x)$ then
 7:                 $n_x \leftarrow n_x + 1$
 8:             end if
 9:         end if
10:     end for
11:     if $n_x = 0$ then
12:         $\mathcal{F}_1 \leftarrow \mathcal{F}_1 \bigcup \{x\}$
13:     end if
14: end for
15: $i = 1$
16: while $\mathcal{F}_i \neq \emptyset$ do
17:     $\mathcal{H} \leftarrow \emptyset$
18:     for $x \in \mathcal{F}_i$ do
19:         for $y \in S_x$ do
20:             $n_y \leftarrow n_y - 1$
21:             if $n_y = 0$ then
22:                 $\mathcal{H} \leftarrow \mathcal{H} \bigcup \{y\}$
23:             end if
24:         end for
25:     end for
26:     $i = i + 1$
27:     $\mathcal{F}_i \leftarrow \mathcal{H}$
28: end while
29: return $\mathcal{F}$
```

Algorithm 2 crowding-distance-assignment($\mathcal{I}$)

```
Require: A set of individuals $\mathcal{I}$
Ensure: Each individual within $\mathcal{I}$ with a density measure.
 1: $l \leftarrow |\mathcal{I}|$
 2: for $i \in [1, N]$ do
 3:     $\mathcal{I}[i] \leftarrow 0$
 4: end for
 5: for each objective $m$ do
 6:     $\mathcal{I} \leftarrow \text{sort}(\mathcal{I}, m)$
 7:     $\mathcal{I}[1]_{\text{density}} \leftarrow \infty$
 8:     $\mathcal{I}[l]_{\text{density}} \leftarrow \infty$
 9:     for $i \in [2, (l-1)]$ do
10:         $\mathcal{I}[i]_{\text{density}} \leftarrow \mathcal{I}[i]_{\text{density}} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)$
11:     end for
12: end for
```

Algorithm 3 NSGA-II main loop

```
Require: A fitness function $\Phi$, $N$ as population size, $g$ as the number of generations, $prob_{mut}$ as probability mutation and $prob_{cross}$ as crossover probability.
Ensure: a population $P_t$ of potential solutions
 1: $P_0 \leftarrow$ initial population, and $Q_0 \leftarrow \emptyset$
 2: for t = 0 to $g$ do
 3:     $R_t \leftarrow P_t \bigcup Q_t$
 4:     $i \leftarrow 1$
 5:     $\mathcal{F} \leftarrow$ fast-nondominated-sort($R_t, \Phi$)
 6:     while $|P_{t+1}| < N$ do
 7:         crowding-distance-assignment($\mathcal{F}_i$)
 8:         $P_{t+1} \leftarrow P_{t+1} \bigcup \mathcal{F}_i$
 9:         $i \leftarrow i + 1$
10:     end while
11:     sort($P_{t+1}, \geq_n$)
12:     $P_{t+1} \leftarrow P_{t+1}[0 : N]$
13:     $Q_{t+1} \leftarrow$
        $\leftarrow$new-pop($P_{t+1}, prob_{mut}, prob_{cross}$)
14: end for
```

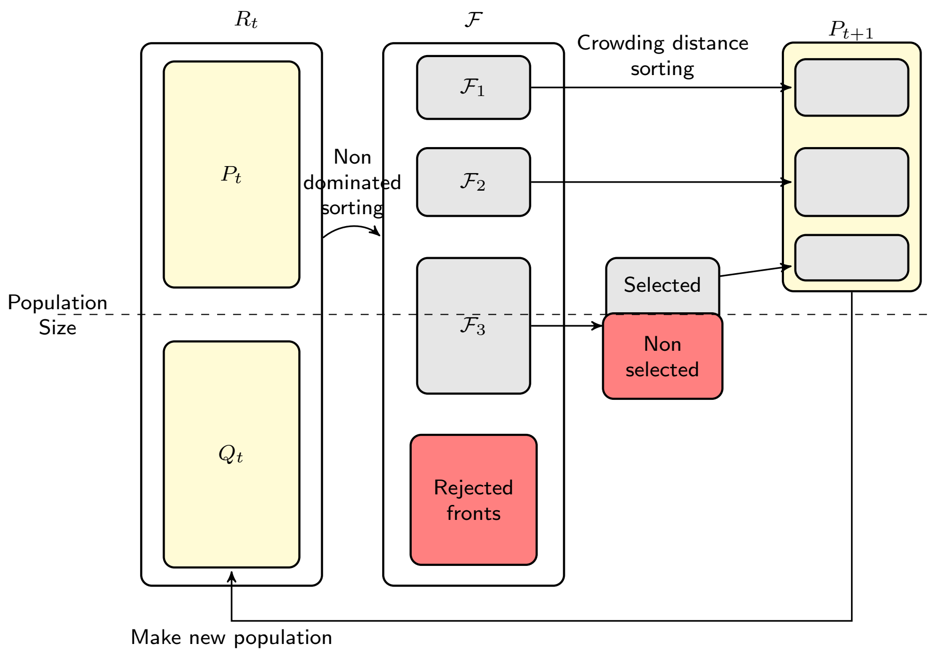

Fig. 5 This figure depicts the basic NSGA-II operation to create the next population P_{t+1}. The non dominated sorting process is performed on the population R_t to generate the fronts. Later, the next population P_{t+1} is created from the fronts until the population size is reached. If the size is exceeded, then the individuals from the last included front are not selected for being included in P_{t+1}.

4.5 Creation of a new Population and the Mutation and Crossover Operations

EAs, like MOEAs, imitate the natural reproduction with a procedure that creates a *mating pool*, which contains the fittest and lucky individuals that became parents of a new generation of individuals.

The mutation and crossover operations are two of the main processes to create a new individuals, and they broaden the space of solutions. Whereas the mutation only involves one individual, the crossover involves two different individuals. Both crossover and mutations operations have many different implementations. Nonetheless, here we explain only the operator that the MOEA algorithm is using to perform CBM. Our MOEA uses, as mutation operator, the bit-flip mutation; as crossover, the single-point crossover; and as selection strategy, the binary tournament.

To implement the binary tournament selection, two individuals are randomly chosen from the current population. This selection is with replacement, so the same individual may be selected repeatedly. Once two individuals are selected, only the fittest individual is stored in the population Q_{t+1}. In draw case, then one of them is chosen randomly. The binary tournament is repeated until Q_{t+1} reaches the limit population.

Given an individual $x = x_1x_2 \dots x_n$, the mutation operator flips the value of each gene x_i with probability $prob_{mut}$. Subsequently, if the gene had a true value then it became false and vice versa. For instance, if $prob_{mut} = 0.01$ then 1 out of 100 genes will be flipped by the mutation operator every time that a new population is created.

With individual x and, given a second individual $y = y_1y_2 \dots y_n$, the crossover mixes the genes of both individuals to create two new individuals with a probability $prob_{cross}$. Every time that two individuals are selected to cross over, an index l is randomly generated in the range $(1, n)$. The genes after the index l in the individuals x, y are swapped. Formally, given l as the single-point to crossover, and the individuals x, y, then the children x', y' are generated as follows:

$$\text{parents=} \begin{cases} x = x_1x_2 \dots x_{l-1}x_lx_{l+1} \dots x_n \\ y = y_1y_2 \cdots y_{l-1}y_ly_{l+1} \cdots y_n \end{cases} ;$$

$$\text{childrens=} \begin{cases} x' = x_1x_2 \dots x_{l-1} \overbrace{y_ly_{l+1} \cdots y_n} \\ y' = y_1y_2 \cdots y_{l-1} \underbrace{x_lx_{l+1} \dots x_n} \end{cases}$$

where the symbols $\overbrace{\quad}$ and $\underbrace{\quad}$ point out to the swapped genes between the parents.

4.6 Interpreting the MOEA approach

A MOEA using our proposed fitness function tends to search for the minimum error rate and to delete the maximum number of cases, without exceeding a threshold of number of non-redundant cases that corresponds to $|M| * (1 - \mathit{redundancy}(M, k))$ (expression 8). Figure 6 depicts the target cases-bases of the fitness function for Iris dataset. That is, case-bases with a lower number of cases and with a similar error rate to the original case-base. To build the figure, we have created 100 case-bases selecting from 3 to 130 random cases from Iris. Therefore, we have 100 case-bases of 3 cases, 100 cases-bases of 4 cases, and so on. Finally, a ten folds Cross-Validation evaluation is used to measure the error rate of each case-base. For each set of 100 cases-bases the error rate given by the Cross-Validation is averaged. Every evaluation use the *3*-NN classifier.

The plot shows for each case-base size the average values of the three objectives $\mathbf{O_{size}}$, $\mathbf{O_{redundancy}}$ and $\mathbf{O_{error}}$. Additionally, the actual error rate is shown as well. The purpose of the fitness function is to find a case-base located in shadowed area, where the case-bases have still error and redundancy levels similar to the original case-base. The search of the maintained case-base will push forward to smaller case-base sizes, but at the same time, there must be a balance between the values of the objectives $\mathbf{O_{redundancy}}$ and $\mathbf{O_{error}}$.

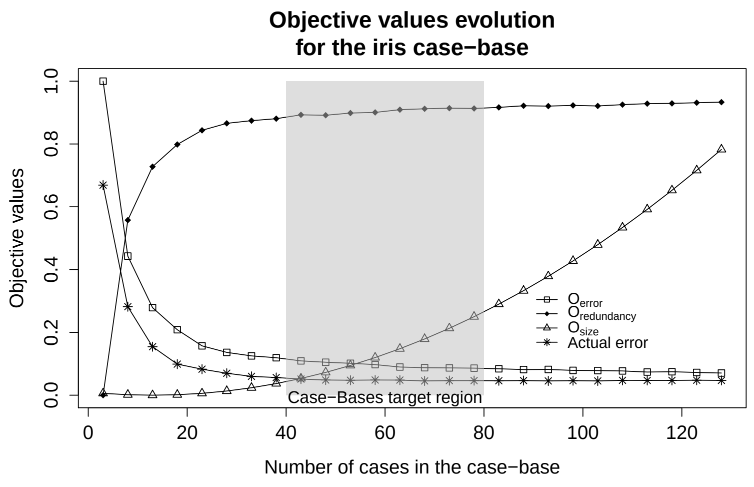

Fig. 6 Evolution of the objectives values for Iris dataset.

5 Experimentation

5.1 Selected Datasets and Characteristics

In order to study if the CBM algorithms are able to cope with noisy and redundant case-bases, we have selected a set of twelve datasets from the UCI repository (Frank and Asuncion (2010)). We have performed a pre-processing of the data in each dataset to ease the experiments. Pre-processing includes the deletion of duplicate cases, the replacing missing values with the mean of the attribute values, and finally, all the features values in each case-base are normalised in the interval $[0, 1]$.

In order to build the case-bases from the datasets, every instance attribute will be part of the problem description, and the class attribute will be the solution.

Later, we have clustered the case-bases according to their level of noise and redundancy given by the complexity profile measure. The reason is to study how our proposal work with different types of cases-bases. The details of each case-base is shown in the Table 1, and the Figure 7 depicts three clusters of case-bases: *redundancy*, *noisy* and *mix*. *Redundant* cases-base are those case-bases with redundancy higher than 0.5, *noisy* are the case-bases with noise level higher than 0.5, and *mix* are those case-bases with redundancy and noise levels lower than 0.5.

5.2 Multi-Objective Evolutionary Algorithm Settings

For each case-base,the following parameters of the evolutionary algorithm are used:

Table 1 A summary of relevant information of the datasets, such as the number of instances, features and classes, before and after the processing, as well as their complexity profile values, where the labels **Insts.**, **Feats.** stands for the number of instances and features of the dataset.

		Original			After	Complexity Profile		
	Dataset	**Insts.**	**Feats.**	**Classes**	**Insts.**	**Error**	**Noise**	**Redundancy**
Redund.	anneal	898	39	6	886	0.1243	0.1185	0.7709
	breast-w	699	10	2	463	0.1788	0.1749	0.7387
	ionosphere	351	35	2	350	0.2929	0.3057	0.6057
	soybean	683	36	19	631	0.3054	0.29	0.5087
Noisy	bridges	105	13	6	105	0.7029	0.7238	0.1238
	contraceptive	1473	10	3	1425	0.9042	0.9614	0.0049
	yeast	1484	10	10	1453	0.8385	0.8809	0.0344
	vehicle	846	19	4	846	0.6566	0.6891	0.1785
Mix	australian	690	15	2	690	0.4809	0.4913	0.3304
	hepatitis	155	20	2	155	0.4561	0.471	0.4129
	sonar	208	61	2	208	0.5048	0.4615	0.1971
	vowel	990	14	11	990	0.568	0.3444	0

Redundancy: anneal, breast-w, ionosphere, soybean
Noisy: bridges, contraceptive, yeast, vehicles
Mix: australian, hepatitis, sonar, vowel

- Population size $\in \{30, 50\}$
- Cross-over probability $\in \{0.9, 0.925, 0.95\}$
- Mutation-probability $\in \{0.03, 0.05\}$
- The number of generations is 100.
- 10 executions with every parameter settings.
- The implementation of the MOEA is NSGA-II.

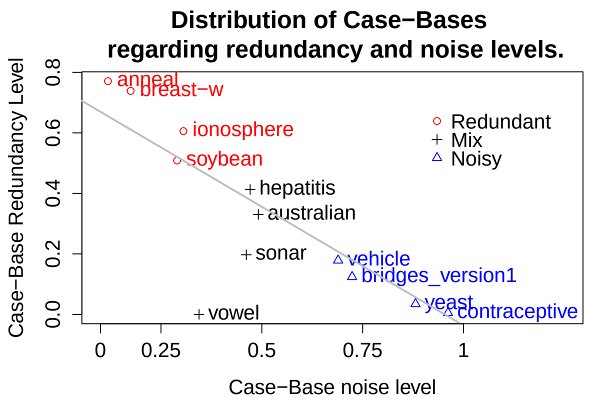

Fig. 7 Clustering of case-bases in relation to redundancy and noise levels. There are three clusters: *redundant*, *noisy*, and *mix*.

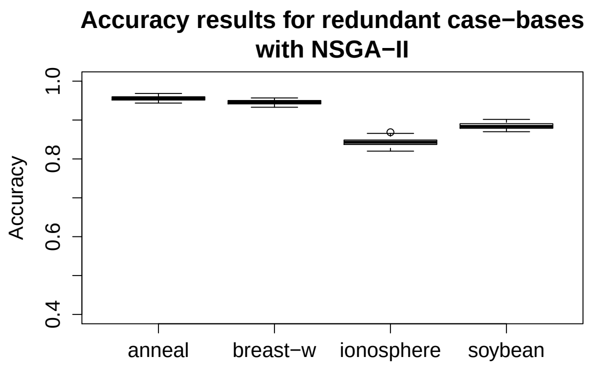

Fig. 8 Accuracy results for the redundant case-bases for all the possible settings combinations.

The MOEA NSGA-II algorithms are executed with all the parameter settings proposed earlier. Later the highest accuracy achieved by the different parameters tuning is chosen to compare the results against other CBM algorithms.

Figures 8 to 13 show the results gathered by NSGA-II over each set of case-bases: *redundant*, *mix* and *noisy* case-bases. The purpose of the figures 8,10 and 12 is to show that the NSGA-II algorithm converges to a set of solutions that have similar accuracies. While the figures 9, 11 and 13 show the relation between the accuracy and the achieved reduction rate. So as to compare the result of NSGA-II algorithm against those given by other CBM algorithms, among all the setting results, we are selecting the configuration that returns the best accuracy average.

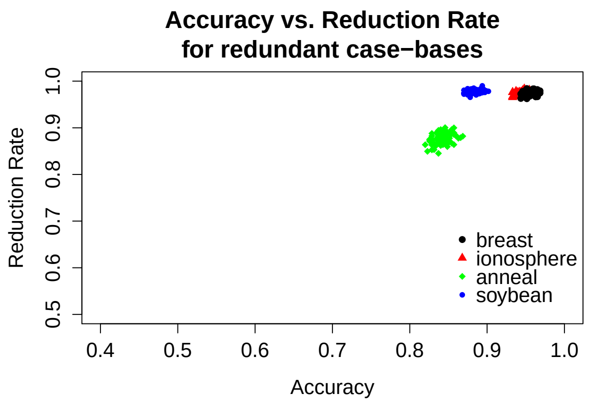

Fig. 9 Distribution of Accuracy against reduction rate for the redundant case-bases for all the possible settings combinations.

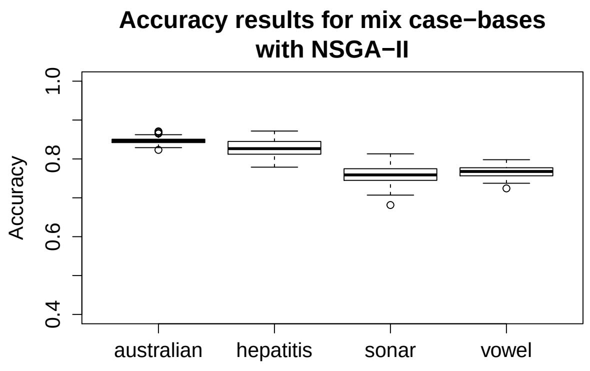

Fig. 10 Accuracy results for the *mix* case-bases for all the possible settings combinations.

Table 2 contains the accuracy averages for each parameter tuning combination. The bold results are the selected results to compare against the rest of the algorithms results. Although the parameters tuning may affect the results, it seems that a population of 30 individuals is enough obtain maintained case-bases that converge to a stable accuracy and reduction rates.

5.3 Results

All the experiments share the same CBR system configuration, that is, only the case-base is different for each experiment. The CBR system retrieves the most similar cases using a k-NN approach with $k = 3$. A voting system is used

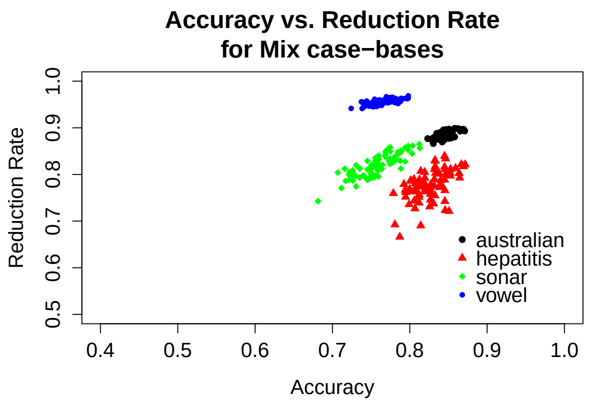

Fig. 11 Distribution of Accuracy against reduction rate for the *mix* case-bases for all the possible settings combinations.

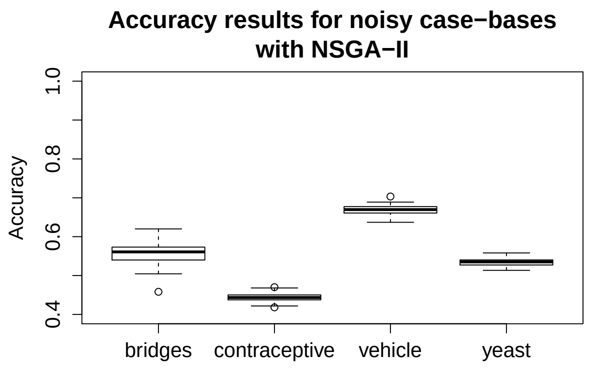

Fig. 12 Accuracy results for the *noisy* case-bases for all the possible settings combinations.

to build the solution from the retrieved set of similar cases, whereby the most common solution in the nearest neighbours is returned by the CBR system.

The purpose of the experiments is not to identify the best CBM algorithm, but to show how the different algorithms react to the types of case-bases. We have selected four existing CBM algorithms: from the NN family, CNN and RENN algorithms; DROP3 from the DROP family; and the Competence Model family is represented by RC-FP.

The average accuracy and reduction rate results for the algorithms CNN, RENN, DROP1, RC_FP and NSGA-II are shown in the Figures 14, 15 and 16, where each shape represent a particular CBM algorithm. The results are normalized in the interval $[0, 1]$, so a reduction rate of 0 means no reduction at all, and 1 that the complete case-base has been deleted. Values close to 1

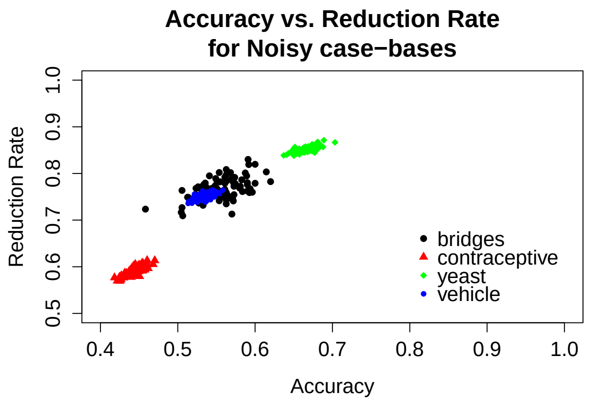

Fig. 13 Distribution of Accuracy against reduction rate for the *noisy* case-bases for all the possible settings combinations.

Table 2 Each cell is the average accuracy result for the 10 executions with the given parameter settings: cross-over and mutation probabilities and population size. The best accuracy results are highlight with bold letters.

Population=30				
	Cross-Over=0.9		Cross-Over=0.925	
Case-Base	Mut=0.03	Mut=0.05	Mut=0.03	Mut=0.05
anneal	0.95408	**0.9594**	0.95428	0.95678
breast	0.9442	0.9443	**0.94818**	0.94426
ionosphere	**0.84828**	0.84344	0.84256	0.83945
soybean	0.88338	**0.8873**	0.88336	0.88381
australian	0.84566	0.84188	**0.85043**	0.84694
hepatitis	0.8142	0.82952	0.838	0.82959
sonar	0.7656	**0.77**	0.76227	0.73898
vowel	0.76151	**0.77505**	0.76717	0.76576
bridges	0.54455	0.55254	0.56026	0.55353
contraceptive	0.44148	0.44548	**0.44798**	0.44196
vehicle	**0.6768**	0.67108	0.67506	0.66165
yeast	0.53564	0.53806	**0.53952**	0.53148

Population=50				
	Cross-Over=0.9		Cross-Over=0.925	
Case-Base	Mut=0.03	Mut=0.05	Mut=0.03	Mut=0.05
anneal	0.95634	0.95531	0.9552	0.95621
breast	0.94514	0.94601	0.94406	0.9464
ionosphere	0.8397	0.84201	0.84373	0.8417
soybean	0.88353	0.88401	0.88243	0.88684
australian	0.84637	0.84347	0.8484	0.84726
hepatiti	0.82197	**0.83967**	0.81846	0.82079
sonar	0.75725	0.75171	0.761	0.76076
vowel	0.76848	0.76747	0.76394	0.76324
bridges	0.56998	**0.57235**	0.55799	0.54718
contraceptive	0.44765	0.43999	0.44303	0.44296
vehicle	0.65833	0.66607	0.66974	0.66916
yeast	0.53035	0.53348	0.53245	0.53227

for accuracy means that the CBR system is returning frequently the correct solution to the input problem. NONE represents the results of the original case-base, hence it is possible to check visually whether the CBM algorithms either improve or worsen the original accuracy. The dashed line represent the average accuracy results of executing 100 times a random selection of cases for the given case-base size.

6 Discussion

As can be seen in Figure 14, regarding the *redundancy* case-bases, all the CBM algorithms generate maintained case-bases with similar accuracies. Because of low noisy levels, RENN removes few cases from the case-base, in contrast to CNN and DROP3 algorithms, which remove a great number of cases. Despite of the fact that having a greater reduction rate may be good, the number of

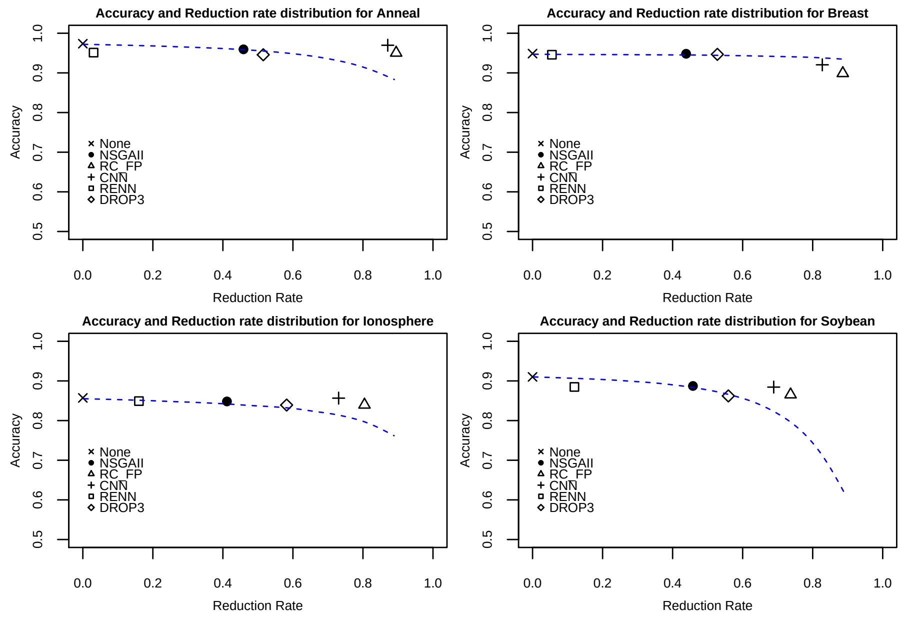

Fig. 14 Accuracy and reduction rate distribution for all the studied CBM algorithms for the redundant case-bases: anneal, breast, ionosphere and soybean.

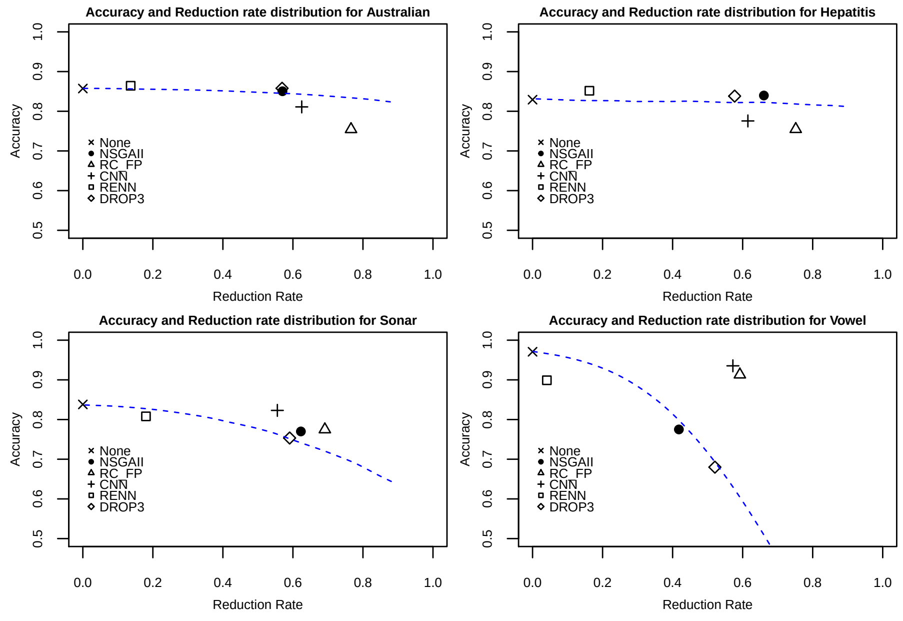

Fig. 15 Accuracy and reduction rate distribution for all the studied CBM algorithms for the mix case-bases: australian, hepatitis, sonar and vowel.

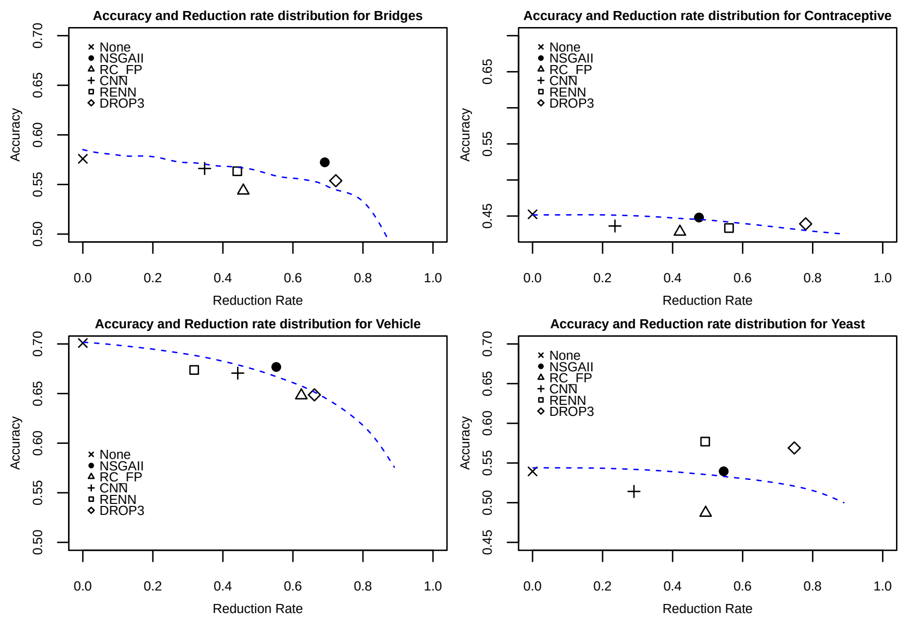

Fig. 16 Accuracy and reduction rate distribution for all the studied CBM algorithms for the noisy case-bases: bridges, contraceptive, vehicle and yeast.

deleted cases is so high that is likely that the maintained case-bases will be over-fitted given the great variance of accuracy results typically found with a lower amount of cases. The NSGA-II and RC_FP achieve similar reduction rates, around half of the case-bases, but NSGA-II is the only one that always achieve higher accuracy than the estimated mean accuracy in every case-base.

Regarding the *mix* case-bases from Figure 15, the RENN is the algorithm with the lowest reduction rate for all the studied case-bases, and it is able of improve the original accuracy of the system when the redundancy and noisy levels of the case-base are large enough, such as hepatitis and australian. Nonetheless, with sonar and vowel case-bases, RENN gets lower accuracy than the estimated mean, in contrast to CNN and RC_FP, which get better results for these case-bases, and achieve good reduction rates in this type of case-base. In case-bases with low redundancy and noise level lower than 0.5, both CNN and RC_FP seem to create maintained case-bases with much better accuracy than the estimated accuracy mean. DROP3 algorithm achieve a reduction of bigger than half of the cases and similar accuracy to the estimated accuracy mean. NSGA-II creates maintained case-bases with similar accuracy to the estimated accuracy mean, and with a reduction rate slightly above half of the cases, excepting for the vowel case-base.

The *noisy* case-bases (16) have high levels of noise and low redundancy levels. With these case-bases type, the RENN algorithm achieves greater reduction rates than in the previous case-bases. Nonetheless, this algorithm is designed to remove noisy cases. CNN does not reduce the size of the case-base as much as in the case of the *redundant* and *mix* case-bases, and its accuracy results are under the estimated mean. DROP3 produces the smallest maintained case-bases for this type of case-bases, thus again it is possible that the maintained cases-bases would be over-fitted. RC-FP removes around half of the cases, but the accuracy is lower than the estimated mean in all the case-bases. NSGA-II is the only algorithm that produces a case-base with better accuracy than the estimated accuracy on all the case-bases. Regarding the reduction rate, it removes around half of the cases on three case-bases but for the bridges case-base.

According to the experiment results, there is no best CBM algorithm for reducing the size of all the considered case-bases. However, MOEA NSGA-II algorithm performs consistently with *redundant*, *mix* and *noisy* case-bases alike, because it is able to create case-bases with similar accuracies to the original case-bases, reducing the number of cases to around half of the initial size. With a maintained case-base size as given using our approach, it is not likely to have an over-fitted CBR system. The only exception seems to be those case-bases with low redundancy and noisy levels, such as the vowel case-base. However, in these types of case-bases CBM is complex without worsening the error rate, because there are insufficient amount of redundant or noisy cases to be deleted from the case-bases.

7 Conclusion

Finding a well-maintained case-base is a complex problem because we are working in a theoretically weak domain. On the one hand, experimental results show that lower number of cases in the maintained case-base often decreases the problem-solving ability of the system. On the other hand, it is not possible to determine the exact amount and selection of cases that produces the optimal case-base. Furthermore, finding the lowest error rate and the smallest maintained case-base may not be the best solution, because it may be that existing cases are not fully representative of future problems and that the case-base would be over-fitted to the existing cases.

In this work we have approached the Case-Base Maintenance task as a multi-objective optimization problem that may be solved with a Multi-Objective Evolutionary Algorithm. In particular, the optimization problem is divided into three different and simultaneous objectives: (1) minimizing the distance of the current number of cases in the case-base to an estimation of the amount of non-redundant cases in the initial case-base; (2) reducing the proportion of redundant cases and; (3) minimizing the estimation of the error rate achieved with the maintained case-base. With this optimization problem, our purpose is to find a case-base, with a lower proportion of no redundant and noisy cases, that is still able to solve problems at a similar level to the original case-base. In order to solve the optimization problem, we have chosen the MOEA NSGA-II, but other MOEA may also be suitable.

We have tested the suitability of our approach with different case-bases and compared the results achieved to those given by other existing CBM algorithms from the literature. The case-bases were classified according to their redundancy and noise levels into three different types: *redundant*, *mix* and *noisy*. Regardless of the type of case-base, the MOEA NSGA-II is the most consistent algorithm, because it performs well with all of them, creating maintained case-bases with similar accuracy to the original case-base and with maintained case-base sizes that avoid the over-fitting problem. Furthermore, given the number of cases of the returned maintained cases-bases, it is likely that the accuracy is higher than the estimated mean accuracy for that amount of cases. The only exception to this behaviour is vowel which has low redundancy and appears to be noisy.

However, there are some drawbacks of using MOEA to perform CBM. Firstly, like the rest of the CBM algorithms, there is no guarantee of finding an optimal solution within finite time, and other CBM algorithms may create better maintained case-bases. This problem is relative though, because none of the studied CBM algorithm is able to find an optimal solution either. Secondly, the runtime could be a limitation, in particular where CBM cannot be performed off-line and the CBR system is stopped until the CBM process finishes. For this reason MOEAs are not suitable in all scenarios. Nonetheless, using a MOEA could be suitable when the case-base is built for the first time from a raw set of data, and where time is not the most important restriction. Finally, the use of a MOEA may require the tuning of the parameters related to the size of the population and the crossover and mutation operator, although it is possible to use some recommended parameters as given in (Jong and Spears (1990); Grefenstette (1986); Laumanns et al (2001)), and achieve good maintained case-bases.

Acknowledgements This work was partially funded by the Seneca Research Foundation of the Region of Murcia under project 15277/PI/10, and by the Spanish Ministry of Science and Innovation+European FEDER+PlanE funds under the project TIN2009-14372-C03-01.

References

Aamodt A, Plaza E (1994) Case-based reasoning: Foundational issues , methodological variations, and system approaches. AI Communications 7:39–59

Aha D (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal Of Man-Machine Studies 36(2):267–287

Aha D, Kibler D, Albert M (1991) Instance-based learning algorithms. Machine Learning 6(1):37–66

Ahn H, Kim Kj, Han I (2007) A case-based reasoning system with the two-dimensional reduction technique for customer classification. Expert Systems with Applications 32(4):1011–1019

Brighton H, Mellish C (1999) On the consistency of information filters for lazy learning algorithms. In: Principles of Data Mining and Knowledge Discovery, Lecture Notes in Artificial Intelligence, vol 1704, pp 283–288

Bunke H, Irniger C, Neuhaus M (2005) Graph matching - challenges and potential solutions. In: Proceedings of the 13th international conference on Image Analysis and Processing, ICIAP'05, pp 1–10

Coello CC, Lamont G, van Veldhuizen D (2007) Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation

Cover T, Hart P (1967) Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13(1):21–27

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2):182–197

Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. SpringerVerlag

Francis A, Ram JA (1993) Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In: In Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning, pp 48–55

Frank A, Asuncion A (2010) UCI machine learning repository. URL http://archive.ics.uci.edu/ml

Gates G (1972) Reduced nearest neighbor rule. IEEE Transactions on Information Theory 18(3):431+

Gil Y (2012) Reproducibility and efficiency of scientific data analysis: Scientific workflows and case-based reasoning. In: Case-Based Reasoning Research And Development, Lecture Notes In Computer Science, vol 7466, pp 2–2

Göker MH, Roth-Berghofer T (1999) The development and utilization of the case-based help-desk support system HOMER. Engineering Applications of Artificial Intelligence 12(6):665 – 680

Grefenstette J (1986) Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 16(1):122–128, DOI 10.1109/TSMC.1986.289288

Hart P (1968) Condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3):515+

Holland JH (1975) Adaptation in Natural And Artificial Systems. MIT Press

Ishibuchi H, Nakashima T, Nii M (2001) Genetic-algorithm-based instance and feature selection. In: Instance Selection and Construction for Data Mining, vol 608, pp 95–112

Jong KAD, Spears WM (1990) An analysis of the interacting roles of population size and crossover in genetic algorithms. In: PPSN, pp 38–47

Juarez JM, Guil F, Palma J, Marin R (2009) Temporal similarity by measuring possibilistic uncertainty in CBR. Fuzzy Sets And Systems 160(2):214–230

Kibler D, Aha D (1987) Learning representative exemplars of concepts: An initial case study. In: Proceedings of the Fourth International Workshop on Machine Learning, pp 24–30

Kim K, Han I (2001) Maintaining case-based reasoning systems using a genetic algorithms approach. Expert Systems With Applications 21(3):139–145

Laumanns M, Zitzler E, Thiele L (2001) On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In: EMO, pp 181–196

Leake D, Wilson D (1998) Categorizing case-base maintenance: Dimensions and directions. In: Advances in Case-Based Reasoning, LNAI, vol 1488, pp 196–207

Leake D, Wilson M (2011) How many cases do you need? assessing and predicting case-base coverage. In: 19th international conference on Case-Based Reasoning Research and Development, ICCBR'11, pp 92–106

Lopez de Mantaras R, McSherry D, Bridge D, Leake D, Smyth B, Craw S, Faltings B, Maher ML, Cox MT, Forbus K, Keane M, Aamodt A, Watson I (2005) Retrieval, reuse, revision and retention in case-based reasoning. The Knowledge Engineering Review 20:215–240

Markovitch S, Scott P (1988) The role of forgetting in learning. In: Proceedings of The Fifth International Conference on Machine Learning, pp 459–465

Massie S, Craw S, Wiratunga N (2005) Complexity-guided case discovery for case based reasoning. In: 20th National Conference on Artificial Intelligence - Volume 1, AAAI'05, pp 216–221

Massie S, Craw S, Wiratunga N (2006) Complexity profiling for informed case-base editing. In: Advances in Case-Based Reasoning, LNAI, vol 4106, pp 325–339

Montani S, Portinale L, Leonardi G, Bellazzi R, Bellazzi R (2006) Case-based retrieval to support the treatment of end stage renal failure patients. Artificial Intelligence in Medicine 37(1):31–42

Olsson E, Funk P, Xiong N (2004) Fault diagnosis in industry using sensor readings and case-based reasoning. Journal of Intelligent and Fuzzy Systems 15(1):41–46

Pan R, Yang Q, Pan S (2007) Mining competent case bases for case-based reasoning. Artificial Intelligence 171(16-17):1039–1068

Riesbeck RSC (1989) Inside Case-Based Reasoning. Lawrence Erlbaum

Rissland EL (2009) Black swans, gray cygnets and other rare birds. In: Proceedings of the 8th International Conference on Case-Based Reasoning: Case-Based Reasoning Research And Development, ICCBR '09, pp 6–13

Smyth B, Keane M (1995) Remembering to forget - a competence-preserving case deletion policy for case-based reasoning systems. In: IJCAI'95, International Joint Conference on Artificial Intelligence, pp 377–382

Smyth B, Mckenna E (1999) Building compact competent case-bases. In: Case-Based Reasoning Research And Development, Lecture Notes in Artificial Intelligence, vol 1650, pp 329–342

Tomek I (1976) Experiment with edited nearest-neighbor rule. IEEE Transactions On Systems Man And Cybernetics 6(6):448–452

Watson I (1998) Is cbr a technology or a methodology? In: Tasks and Methods in Applied Artificial Intelligence, LNCS, vol 1416, pp 525–534

Wilson D (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems Man and Cybernetics SMC2(3):408–&

Wilson D, Martinez T (2000) Reduction techniques for instance-based learning algorithms. Machine Learning 38(3):257–286

Wilson DR, Martinez TR (1997) Instance pruning techniques. In: Machine Learning: Proceedings Of The Fourteenth International Conference (ICML97), Morgan Kaufmann, pp 404–411

Yu X, Gen M (2010) Introduction to Evolutionary Algorithms, Decision Engineering. Springer, DOI 10.1007/978-1-84996-129-5

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4):257–271