UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMATICA

Temporal Case-Base Maintenance

Mantenimiento de Bases de Casos Temporales

D. Eduardo Lupiani Ruiz

2014

Temporal Case-Base Maintenance
Mantenimiento de Bases de Casos Temporales

Eduardo Lupiani Ruiz

Faculty of Computer Science
Department of Information and
Communication Engineering

University of Murcia

Advisors:
Prof. Dr. José M. Juárez Herrero
Prof. Dr. José Palma Méndez

Ph.D. Thesis:

Computer Science

Murcia, Spain 2014

Acknowledgements

Me siento incapaz de otorgarme todo el mérito de esta tesis porque han sido muchas las personas que han influido en ella, desde mis directores y compañeros de trabajo, hasta mis amigos más cercanos y mi familia. Por ello, mi más profundo y sincero agradecimiento a Pepe Palma y Pepe Juárez, por haberme enseñado a hacer ciencia y por haber sido mi guía en la tarea académica más dura que he hecho en mi vida. Me siento afortunado por haber tenido la oportunidad de trabajar con dos grandes profesionales como vosotros y de haber tenido acceso ilimitado a vuestro conocimiento y sabiduría, no puedo imaginarme la paciencia enorme que habéis tenido conmigo. Mil gracias a mis compañeros de fatigas en el laboratorio, a Antonio Morales y Antonio Gomariz, porque habéis hecho de estos cuatro años una experiencia inolvidable. Os aseguro que os echaré de menos y os deseo lo mejor en el futuro. No me olvido de Manolo, por sus consejos y por reír mis chistes cortos, ni de Roque por haberme enseñado cómo funciona el mundo educativo universitario. Tampoco de Félix, y de nuestras conversaciones sobre alta montaña. Mi agradecimiento a Rafa, Paco y Jesualdo, por haberme dado la oportunidad de empezar mi carrera investigadora, de vosotros aprendí mucho y os lo debo.

I am extremely grateful to Susan, Stewart, Thomas and Christian. I will always remember the lifetime opportunity you gave me while I was visiting you at your universities. You have taught me new ways of understanding science.

Especialmente se merecen mis agradecimientos mis padres Eduardo y Mercedes, por el apoyo que he recibido de ellos toda mi vida, por haber confiado en mí incluso cuando yo les hablaba de los más negros nubarrones en el horizonte. A mi hermana Mercedes, mi gran amiga y confidente desde que tengo memoria, y a mi hermana María Luisa, quien me sirve de inspiración de lo bien que se pueden llegar a hacer las cosas. También me acuerdo de mis abuelos, que fueron mi ejemplo de vida. Y muy especialmente estoy agradecido a Beatriz, por estar a mi lado y hacer de mí cada día una mejor versión de mi mismo. Gracias por tu incansable aliento y apoyo, por saber escuchar mis inacabables monólogos sobre mi trabajo. Espero hacerte sentir orgullosa de mi tanto como yo lo estoy de ti.

18 de Junio de 2014

Abstract

Case-Based Reasoning (CBR) is a problem-solving methodology that solves problems by analogy with previously solved problems. The basis of CBR is a *case*, an independent piece of knowledge that associates the description of a problem with its solution, where cases are retained in a knowledge-source known as a case-base.

The amount of cases may be a sign of the expertise of the CBR system for solving the problem domain. However, having a large case-base does not guarantee an improvement of the problem-solving capability. On the contrary, an accumulation of many cases may lengthen the response time of the reasoning process and, in certain scenarios, negatively affect the correct solution of certain types of problem. Case-Base Maintenance (CBM) tasks reduce the number of cases within the case-base without affecting the problem-solving accuracy of the reasoning process.

CBM is essential when CBR is used in time dependant domains where CBR has to include temporal representation techniques in case descriptions. Nevertheless, temporal representation implies more complex case structures and makes it more difficult and costly to quantify the similarity between cases. This means the case-base should be as small as possible without harming its problem solving capabilities. However, to our knowledge, no algorithm has been proposed to perform CBM in case-bases with temporal cases.

In this thesis, we propose: (i) an evaluation method to study the effects of using CBM algorithms on CBR performance; (ii) a temporal framework for use in temporal CBR; and (iii) a set of temporal CBM algorithms. In addition, a new CBM algorithm based on a multi-objective optimization evolutionary approach is proposed. Lastly, our proposals and hypotheses are tested with data of elderly people monitored at home. In particular, the experiments conducted confirm the suitability of our proposed evaluation method to study the consequences of using CBM. Moreover, the experiments also support our initial hypothesis that it is possible to successfully perform a maintenance task on temporal case-bases with the proposed temporal CBM algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

This thesis has been developed within the Artificial Intelligence and Knowledge Engineering Group (AIKE) of the University of Murcia, Spain. The thesis was funded by the PhD Fellowship BES-2010-030294 within the project AI-SENIOR (TIN2009-14372-C03-01) by the Spanish Ministry of Science and Innovation. Part of this work was also partially funded by the project DAISY (15277/PI/10) of the Seneca Research Foundation of the Province of Murcia, Spain. Both projects are aimed at developing a system to detect detect regular patterns and abnormal scenarios in a Smart Home environment and for elderly people living alone.

The AI-Senior project is intended to provide a tool to help families take care of their relatives living alone, and facilitate the rapid response of the emergency unit in case of an accident at home. Furthermore, the system allows users to live longer at home independently. Regarding the economic aspect, it may reduce the cost of social and health agencies without worsening the quality of their services.

1.2 Motivation

Case-Based Reasoning (CBR) is a methodology that makes use of past experiences to solve new problems. In a CBR system, the atomic unit of knowledge is the *case*, an independent piece of knowledge that associates the description of a problem with its solution [1]. The representation of a case is basically a tuple composed of two elements: a problem and its solution. When multiple cases are created to solve problems in a given domain, together they form the knowledge-base upon which the CBR system can perform its problem-solving process [1]. In a CBR system, the knowledge-base is called as the *case-base*. One of the most relevant features of this CBR model is that the learning process is incremental and continuous, since newly solved cases are added to the case-base, and so the learning step is integrated into the problem solving process itself.

According to [169], Case-Base Maintenance (CBM) aims to reduce the case-base size and to improve the quality of the solutions provided by the CBR system [126]. The benefits of its use are a simplification of the problem-solving process, the speeding up of the case retrieval, and even an improvement in the capability of the system to solve problems. CBM has been studied in depth in CBR literature [18, 53, 107, 126, 146, 167, 169]. A CBM algorithm is deterministic when it always builds the same maintained case-base. Otherwise, the CBM algorithm is non-deterministic.

When using CBM algorithms, the key question is whether the maintained case-base is better, equivalent to or worse than the original one. The straightforward approach to answering this question is to use classical model evaluation methods, such as Hold-Out and Cross-

Validation, to analyse and compare the outcomes of the same CBR system obtained with these two case-bases. However, these methods do not take into consideration the singularity of applying a CBM algorithm to the case-base, which reduces the case-base size, providing evaluation results with insufficient statistical support.

In the early years, CBR was focused on non-temporal dependent problems, where the problem descriptions within the cases were snapshots of the domain system. Nonetheless, in the last decade, the inclusion of temporal features within the problem descriptions has gained a relevant place in CBR systems, as it is highlighted by the number of publications related with CBR systems working with time dependant features [16, 59, 74, 78, 111, 115, 116, 159]. In these works the term *temporal case* is accepted to describe cases which have at least one temporal feature.

What is more, there are real problem domains with time dependant features that may benefit from using CBR. In particular, within the AI-Senior and DAISY projects, the spatio-temporal features are relevant for identifying a scenario occurring at home. For instance, a record of the different rooms visited by the inhabitant during the day may identify a normal day or bad night. Consequently, cases require a temporal representation to be able to model this spatio-temporal knowledge into their problem description. The importance of the temporal representation within cases is highlighted by the number of works conducted in this sense in the CBR literature, for instance using time series [76, 115, 117, 118, 122], workflows [14, 21, 26, 27, 43, 78, 111, 112, 165], episodes [33, 56] and event sequences [59, 60, 73, 100].

One common problem of temporal representations is the complexity of the temporal distance functions, which increases the complexity of the case retrieval step. This means that temporal case-bases suffer a performance problem as the case-base size increases. Such a scenarios underlines the applicability of a CBM algorithm. However, to our knowledge, little attention has been paid to the effect and consequences of using CBM algorithms with CBR systems designed to operate with temporal domains. In fact, to date no proposal has been made carry out the maintenance task with temporal case-bases.

Within the AI-Senior project, the thesis objectives are related to the temporal representation of the daily activities (or scenarios) of the elderly at home, and the use of a CBR system to infer the type of scenario occurring at home. Furthermore, this thesis aims to develop CBM algorithms that can be used for selecting a subset of representative scenarios of the daily activities, either normal or abnormal, in order to facilitate the inference process of the reasoning system. Finally, this thesis is intended to develop an evaluation methodology that quantifies with sufficient statistical support the consequences of applying a maintenance algorithm to the proposed CBR system.

We believe that traditional evaluation methods may not quantify the problem-solving capabilities of the CBR system when maintenance has been conducted in its case-base correctly,

such as Cross-Validation and Hold-Out. CBM algorithms may drastically reduce the size of the original case-base, so that insufficient cases will be the maintained in the case-base to solve the entire problem domain; in other words, the competence of the CBR system will be reduced. Furthermore, non-deterministic maintenance approaches, such as those described in [5, 70, 81, 97], build maintained case-bases with a greater variability of cases, meaning that more statistical evidence than provided by traditional evaluation methods is necessary to provide more support to evaluation results.

1.3 Hypothesis and objectives

Given a temporal case representation, our initial hypothesis is that existing CBM algorithms may be adapted to work with temporal case-bases. From this hypothesis, this thesis pursues the following objectives:

1. To conduct a thorough bibliographical review of the existing literature related to the topic of this thesis, as well as as to perform an exhaustive analysis of the techniques that could provide support to our hypothesis.
2. To propose a non-deterministic CBM algorithm which takes advantage of the implicit information in the case-base. This information is modelled as a multi-objective optimization problem.
3. To propose a specific evaluation methodology for deterministic and non-deterministic CBM algorithms.
4. To propose a temporal extension of classical CBM algorithms.
5. To develop a CBR system to monitor elderly people living alone at home and to detect abnormal scenarios, and to carry out an experimental evaluation to compare the results of the original case-base with those built by the proposed temporal CBM algorithms.

1.4 Structure of the thesis

For the sake of clarity, the organization of this thesis is as follows:

In **chapter 2** we analyse exhaustively the related technologies and theories that will be used in this thesis. More specifically, we will review works related with CBM algorithms, the approaches to represent temporal information within a case, and temporal similarity measures to perform the retrieval of temporal cases. **Chapter 3** introduces our CBM proposal as a multi-objective problem to be optimized by a non-deterministic algorithm. In **chapter 4** we present the $\alpha\beta$ evaluation method, which evaluates deterministic and non-deterministic CBM algorithms. In **chapter 5**, we propose the extension of classical CBM algorithms for temporal

case-bases: t-CNN, t-DROP1, t-ICF, t-RC-FP and t-RENN. **Chapter 6** describes the suitability of the presented temporal CBM algorithms for the practical problem of monitoring elder people living alone at home. Finally, we highlight in **chapter 7** our conclusions related with the work developed during this thesis.

Chapter 2

State of the Art

In this chapter we exhaustively revise the topics related with the thesis. In section 2.1, the Case-Based Reasoning is introduced, as well as the importance of the analogy mechanism in this type of reasoning. In section 2.2 we show a review of a set of Case-Base Maintenance algorithms and its families. The section 2.3 describes the most common statistical measures to quantify the quality of a CBR system. Section 2.4 provides an explanation about evolutionary algorithms. Later, in section 2.5 we introduces the concepts of general temporal theory and we provides the research conducted to apply CBR to temporal dependant domains.

2.1 Case-Based Reasoning

Case-Based Reasoning (CBR) is a methodology that makes use of past experiences to solve new problems [164]. In a CBR system, the atomic unit of knowledge is the *case*, an independent piece of knowledge describing one problem and its solution [1]. Accordingly, CBR finds the solution of a given problem by analogy with past solved cases. The basic representation of a case is 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 [1]. Within the CBR community, the knowledge base is known as the *case-base*.

The root of CBR is the research conducted by Schank, who developed a model of dynamic memory in order to store earlier situations that could help solving future problems [138], and the research done by Gentner, who introduced the concept of analogical reasoning [55]. Both researches were the foundations of CYRUS, the first CBR system proposed by Kolodner [86]. Others CBR works were published from these theories, such as MEDIATOR [140], CHEF [61] and JULIA [65].

Other CBR early works are based on the theories given by Bruce Porter et al., who introduce machine learning classification techniques to solve cases as well to memorise them [127]. Examples of works following this approach are PROTOS system [127], HYPO system [11], and later the combined case-based and rule-based system CABARET [141]. On the whole, to the date there are a myriad of works related with CBR applied to many different topics. In fact, the term CBR is so wide, that many used terms have been used when this has been applied, such as *Exemplar-based reasoning*, *Instance-based reasoning*, *Memory-based reasoning* and *Analogy-based reasoning*.

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 [57], signals [119, 125], time sequences [74] or graphs [21] among others.

The most commonly used implementation of a CBR system was proposed in [1]. Figure

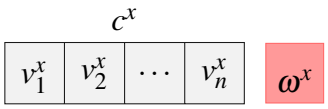


Fig. 2.1 Instance as data structure basis of a case.

2.2 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 [34];

(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 characteristic of this CBR model is its incremental and continuous learning process, where each new solved case is added to the case-base. Hence, the learning step is integrated into the problem solving process itself.

The concept of similarity plays a relevant role in CBR systems, since input problems are solved with solutions from similar solved problems to them. In short, similarity is understood as a numeric quantity that reflects the strength of the relationship between two cases, or in other words, how alike two cases are. Alternatively, distance can be used to quantify the proximity between two cases, which is a numerical quantity that measures how close two cases are to each other. Despite the concepts similarity and distance may sound similar, they quantify differently the proximity between cases. While high similarity values mean low proximity between the cases, low distance values stand for those proximate cases to each other. Furthermore, similarity is complex to measures in comparison to distance, because this is not defined with a mathematical foundation, and there are at least two reasons why similarity should have a formal basis. Firstly, one can prove properties about the CBR system in a mathematical way. Secondly, one can build engineering systems on a solid foundation [131].

Fortunately, the distance between two cases has been widely studied within the CBR community [38, 134, 168, 169]. In particular, similarity function relies on top of two distance functions: one *amalgamation or global distance function* δ_{global} and a set of *local distance functions* δ_{local} [45, 131, 134]. In fact, the global distance function is a composition of all the existing δ_{local} functions in order to measure the distance between two cases.

Figure 2.3 portrays the relations between the distance and similarity functions. Whereas the i-th local distance function δ^i_{local} computes the distance between the variable values v^x_i and

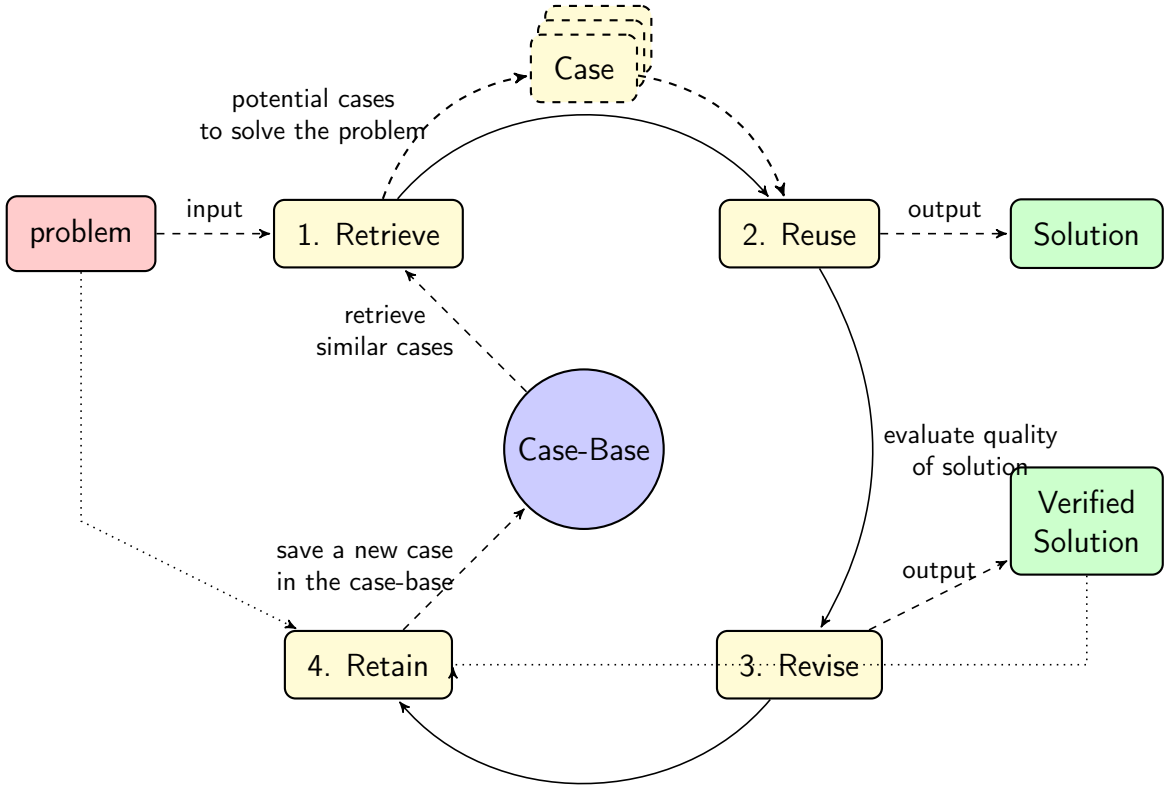


Fig. 2.2 The CBR cycle and its four steps. The CBR system using analogy-based reasoning to either build or adapt a solution to solve the input problem.

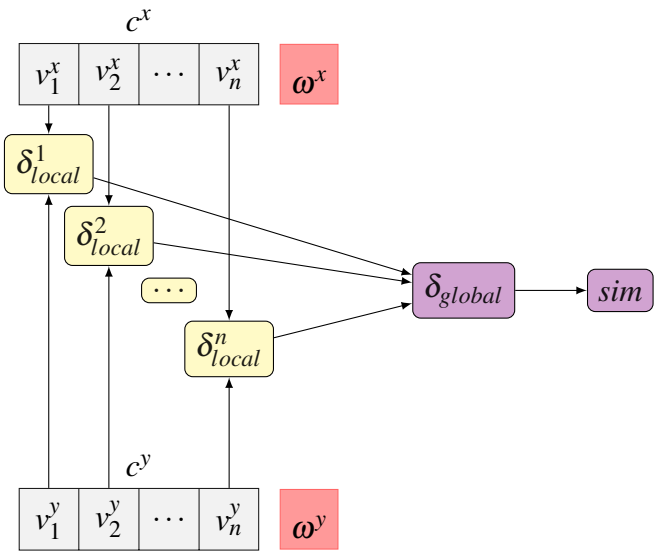


Fig. 2.3 Illustration of global, local distances and the similarity function

v_i^y. The local distance outputs are used by the global distance δ_{global}, and finally, this function is used by the function *sim* to compute the similarity between the cases c^x and c^y.

The specific description of the δ_{local}^i distance depends on the type of variable. For instance,

if the variable is numeric, this distance could be the absolute difference between the values, or if the variable is symbolic (or categorical), the distance could be either 0 if they are equal or the maximum possible distance if the symbols are different.

$$\text{numeric: } \delta_{local}(v_i^x, v_i^y) = |v_i^x - v_i^y|$$

$$\text{symbolic: } \delta_{local}(v_i^x, v_i^y) = \begin{cases} 0 \text{ if } v_i^x = v_i^y \\ \text{maximum distance otherwise} \end{cases}$$

There is a myriad of global distance functions in the literature, such as Manhattan, Euclidean or Minkowski, Chevichev, among others. Details of these measures may be found in [168, 169]. In particular, one of the most used global distance in CBM algorithms is the Minkowski distance:

$$\text{Minkowski: } \delta_{global}(\pi^x, \pi^y, r) = \left(\sum_{i=1}^{n} \left(\delta_{local}^i \left(v_i^x, v_i^y \right)^r \right) + \delta_{temp} \left(s^x, s^y \right)^r \right)^{\frac{1}{r}},$$

where $r \in \mathbb{N}^+$. From the Minkowski distance are defined the well known Euclidean ($r = 2$) and Manhattan ($r = 1$) distances.

$$\text{Euclidean: } \delta_{global}(\pi^x, \pi^y, 2) = \sqrt{(\sum_{i=1}^{n} \left(\delta_{local}^i \left(v_i^x, v_i^y \right)^2 \right) + \delta_{temp} \left(s^x, s^y \right)^2}$$

$$\text{Manhattan: } \delta_{global}(\pi^x, \pi^y, 1) = \sum_{i=1}^{n} \left(|\delta_{local}^i \left(v_i^x, v_i^y \right)| \right) + |\delta_{temp} \left(s^x, s^y \right)|$$

2.2 Case-Base Maintenance

Case-Base Maintenance (CBM) 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 [92]. In particular, according to [169], CBM aims to reduce the case-base size or to improve the quality of the solutions provided by the CBR system [126].

The CBM has been studied in depth in Machine Learning disciplines such as Instance-based Learning, Exemplar-based Learning, as well as in CBR. There are a wealth of approaches to perform CBM, such as those published in [18, 53, 107, 126, 146, 167, 169]. Using the k-nearest neighbour algorithm [34] to find the similar cases (neighbours) to a given case, 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. The task of identifying a redundant

case is easier than determining whether a case is noisy. Whereas a case could be considered as redundant when most of its neighbours have the same solution, a noisy case has a solution different to its neighbour's solutions [106]. Furthermore, 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 [132].

Some authors have proposed different taxonomies of CBM algorithms in an attempt to enhance our understanding of them. For example, [126] classify the different CBM algorithms according to the following features: (i) *Case Search Direction:* when a CBM algorithm starts with an empty case-base that is increased with selected cases from the original case-base, then the CBM algorithm is known as *Incremental* CBM; otherwise, the CBM algorithm is known as *Decremental*. (ii) *Order sensitive:* when the resulting maintained case-base is affected by the order in which the cases are explored by the CBM algorithm, then this is *Order Sensitive*; otherwise the CBM algorithm is known as *Order Insensitive*. (iii) *Selection Criteria:* if the cases for being part of the maintained case-base are selected according only considering their nearest case neighbours, then the CBM algorithm is *local*, however when all the cases are involved then is *global*. (iv) *Type of cases to retain:* the CBM algorithm retains *central cases* when it only selects cases with all their neighbours from the same solution than themselves. On the contrary, if the chosen cases have neighbours from a different type of solution, then the CBM algorithm retains *border cases* [169].

Other classifications can be made according to whether a CBM algorithm is deterministic or non-deterministic. Whereas a deterministic CBM 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.

The retrieval time is usually proportional to the case-base size among other factors, such as the number of features (attributes) and the representation complexity of each attribute. Therefore, to achieve good performance the number of learned cases in a CBR system must be kept to a minimum. However, the exact number of cases is difficult to estimate [93].

The simplest CBM algorithm consists of a random deletion of cases until the case-base reaches a certain number of cases [105]. 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: Nearest Neighbour algorithms (NN), 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 is the maintenance process itself.

Algorithms from NN family select cases according to a nearest neighbour policy. One of the first attempts to reduce case-base size was the Condensed Nearest Neighbour (CNN) [62]. Starting with a random case of each existing solution as initial case-base, the algorithm uses the rest of them as test sets and classifies them using the selected cases with a k nearest neighbours classifier (K-NN). If a case is misclassified, then this is added to the maintained 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, due to which, the Reduced Nearest Neighbour Rule (RNN)[53] 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. The Edited Nearest Neighbour (ENN) removes misclassified cases according to the solutions of their three nearest neighbours [167]. When ENN is executed multiple times, taking each output as input of the next execution, then the method is called Repeated ENN (RENN). All-kNN algorithm consists of executing k times ENN, where each execution uses from 1 to k neighbours respectively to flag a case for no selection [157]. Some authors claim ENN and its variations are actually noise removal techniques [169].

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 [3, 4]. 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 [79].

The CBM algorithms DROP1, DROP2 and DROP3 belong to the DROP family [169]. 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 explode the implicit information of

the case-base to build a maintenance model. The *Competence Model*[145, 146] defines concepts as Coverage, Reachability and Relative Coverage [147]. For each concept, a new CBM algorithm is proposed. COV and RFD use Coverage Set and Reachability cardinality to sort the case-base, respectively, and RC uses the Relative Coverage model. Finally, the sorted case-base is the input to CNN, which performs the maintenance. *Complexity Profiling* estimates the proportion of redundant and noisy cases, as well as the existing error rate in the case-base [106, 107]. 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 [106], it does not provide enough information to evaluate the case-base competence. Lastly, *Weighting, Clustering, Outliers and Internal cases Detection based on DBscan and Gaussian Means* (WCOID-DG) [144], use the competence concepts to reduce both the storage requirements and search time and to focus on balancing case retrieval efficiency and competence for a case-base. WCOID-GM is mainly based on the idea that a large case-base with weighted features is transformed to a small case-base with improving its quality.

Algorithm 1 $\delta_{global}^{temporal}$

Require: Two cases $c^x = (\pi^x, \omega^x)$, $c^y = (\pi^y, \omega^y)$
Ensure: A real number in the interval $\mathbb{R}^+$
1: $dist \leftarrow 0$
2: **for all** $v_i^x \in \pi^x, v_i^y \in \pi^y$ **do**
3: $\quad dist \leftarrow dist + \delta_{local}^i(v_i^x, v_i^y)^2$
4: **end for**
5: $dist \leftarrow dist + \delta_{temp}(s^x, s^y)^2$
6: $dist \leftarrow (dist)^{\frac{1}{2}}$
7: **return** $dist$

Condensed Nearest Neighbour. CNN

Hart published the Condensed Nearest Neighbour (CNN), the first algorithm to reduce the size of a case-base in 1968 [62]. This algorithms was incremental, order sensitive and with a local evaluation criteria.

Whereas the algorithm complexity is in the best scenario $O(k \cdot |C|)$, in the worst scenario is $O(k \cdot |C|^2)$, being k the number of neighbours.

Reduced Nearest Neighbour. RNN

Reduced Nearest Neighbour (RNN) algorithm was proposed by Gates in 1972 [53]. This algorithm is aimed to reduce the sensitive to noise of CNN algorithm. Thus, RNN add a post-stage to CNN to remove the noise cases.

Algorithm 2 NearestNeighbour

Require: A case $c^x = (\pi^x, \omega^x)$, a case-base C, a global distance δ_{global}, and $k \in \mathbb{Z}$ as the number of neighbours.

Ensure: A set of cases NN by ascending order according to their distances to c^x.

```
1: NN^x ← ∅
2: for all c^y = (π^y, ω^y) ∈ C do
3:    similarity ← sim(c^x, c^y)
4:    NN^x ← NN^x ∪ {(c^y, similarity)}
5:    sort NN^x in descending order
            according to their similarity
6:    if size of NN^x > k then
7:       remove last elements in NN
8:    end if
9: end for
10: return NN
```

Algorithm 3 correctlyClassified

Require: A case $c^x = (\pi^x, \omega^x)$, a case-base C, a global distance δ_{global}, and $k \in \mathbb{Z}$ as the number of neighbours.

Ensure: A boolean value, it returns *true* if c^x is correctly classified by a nearest neighbour in the case-base C.

```
1: NN^x ← NearestNeighbour(c^x, C, δ_global, k)
2: match ← 0
3: for all c^y = (π^y, ω^y) ∈ NN^x do
4:    if ω^x = ω^y then
5:       match ← match + 1
6:    end if
7: end for
8: if match ≥ k/2 then
9:    return true
10: end if
11: return false
```

Since RNN is based on CNN, its computational complexity in the best scenario is $O(k \cdot |C| + |M^{CNN}|)$, and in the worst $O(k \cdot |C|^2 + |M^{CNN}| \cdot |C|)$.

Edited Nearest Neighbour. ENN

Wilson presented the algorithm Edited Nearest Neighbour (ENN) in 1972 [167]. Its mains purpose was the removal of the noisy cases, assuming that this type of cases are those with most of its neighbours belonging to a different solution, being a convenient algorithm to enhance the accuracy of the CBR system [169]. Since ENN is specialised removing noisy cases, this is used by other noise sensitive CBM algorithms as a first step to remove them. A variation of ENN is Repeated Edited Nearest Neighbours (RENN), which consists on the repetition of

Algorithm 4 CNN

Require: A case-base C, $k \in \mathbb{Z}$ as the number of neighbours .
Ensure: A maintained case-base C^{σ}

$C^{\sigma} \leftarrow \emptyset$
for all $\omega \in \Omega$ **do**
  $c^x \leftarrow$ random case of C with solution ω
  $C \leftarrow C - \{c^x\}$
  $C^{\sigma} \leftarrow C^{\sigma} \cup \{c^x\}$
end for
repeat
  for all $c^x = (\pi^x, \omega^x) \in C$ **do**
    $C - \{c^x\}$
    if not $correctlyClassified(c^x, C, \delta_{global}, k)$ **then**
      $C^{\sigma} \leftarrow C^{\sigma} \cup \{c^x\}$
    else
      $C \leftarrow C \cup \{c^x\}$
    end if
  end for
until C without changes
return C^{σ}

Algorithm 5 RNN

Require: A case-base C
Ensure: A maintained case-base C^{σ}

$C^{\sigma} \leftarrow CNN(C)$
for all $c^x \in C^{\sigma}$ **do**
  $C^{\sigma} \leftarrow C^{\sigma} - \{c^x\}$
  for all $c^y \in C$ **do**
    if not $correctlyClassified(c^y, C^{\sigma})$ **then**
      $C^{\sigma} \leftarrow C^{\sigma} \cup \{c^x\}$
      break
    end if
  end for
end for
return C^{σ}

ENN until no more noisy cases can be found in the case-base.

The complexity of ENN in the best scenario is $O(3 \cdot |C|)$, and in the worst scenario$O(3 \cdot |C|^2)$.

All-kNN

Tomek modified the algorithm ENN with its proposal All-kNN [157]. Similarly to RENN, which repeats ENN, this algorithm executes ENN with incremental values of considered neigh-

Algorithm 6 RENN

Require: A case base C
Ensure: A temporal case-base C^{σ}
1: $C^{\sigma} \leftarrow C$
2: **repeat**
3: {The body of this loop is the ENN algorithm}
4: **for all** $c^x \in C^{\sigma}$ **do**
5: **if not** $correctlyClassified(c^x, C, \delta_{global}, 3)$ **then**
6: $C^{\sigma} \leftarrow C^{\sigma} - \{c^x\}$
7: **end if**
8: **end for**
9: **until** C^{σ} without changes
10: **return** C^{σ}

bours. That is, firstly ENN is executed with only one neighbour, secondly with two neighbours and so on, until k neighbours is reached.

Algorithm 7 All-kNN

Require: A case-base C, $k \in \mathbb{Z}$ as the number of neighbours
Ensure: A maintained case-base C^{σ}
1: $C^{\sigma} \leftarrow C$
2: $R \leftarrow \emptyset$
3: **for all** $c \in C^{\sigma}$ **do**
4: **for all** $n \in (1,k)$ **do**
5: **if not** $correctlyClassified(c^x, C, \delta_{global}, n)$ **then**
6: $R \leftarrow R \cup \{c\}$
7: **end if**
8: **end for**
9: **end for**
10: $C^{\sigma} \leftarrow C^{\sigma} - R$
11: **return** C^{σ}

The computational complexity of All-kNN is $O(k \cdot |C|)$.

Shrink

Kibler and Aha presented Shrink in 1987 [79]. This algorithm was proposed as well as an extension of CNN in order to reduce its noise sensitivity. Thus, a post-stage for removal misclassified cases was added after the execution of CNN algorithm.

The complexity of Shrink in the best scenario is $O(k \cdot |C| + |C^{CNN}|)$, and in the worst $O(k \times |C|^2 + |M^{CNN}|)$

Algorithm 8 Shrink

Require: A case-base C, $k \in \mathbb{Z}$ as the number of neighbours
Ensure: A maintained case-base C^{σ}

```
1: C^σ ← CNN(C)
2: for all c^x ∈ C^σ do
3:     C^σ ← C^σ − {c^x}
4:     if not correctlyClassified(c^x, C^σ, δ_global, k) then
5:         C^σ ← C^σ ∪ {c^x}
6:     end if
7: end for
8: return M^σ
```

Iterative Case Filtering. ICF

Brighton y Mellish presented in 1999 the CBM algorithm Iterative Case Filtering (ICF) [18]. A case is deleted from the case-base when the number of cases that solve it are bigger than the number of cases than the case can solve by itself. This algorithm is not order sensitive, creating always the same case-base maintenance.

DROP1, DROP2 y DROP3 Algorithms

The first published work of this algorithm family was done in 1997 with the name of RT family [170], and later was introduced with the name DROP in the year 2000 [169]. DROP1 is the basis of the rest of the algorithms, which try to improve them, although their results depends on the case-base.

DROP1 is noise sensitive since this is order sensitive. In order to change this, DROP2 sort the cases according to their closer enemy in a first stage. Given a case, another case is considered as its enemy if this contains a different solution than the case. Furthermore, DROP2 decides when the cases remain in the maintained case-base in a different way. A case will not be part of the maintained case-base if as many of its associates in the original case-base are classified correctly without the case in the maintained case-base.

DROP3 includes a previous stage to the execution of DROP2 in order to remove the noise cases from the case-base with the algorithm ENN.

RC-FP, RFC-FP, COV-FP

The algorithms COV-FP, RFC-FP, AND RC-FP were introduced by McKenna and Smyth between 2000 and 2001 [110, 149]. These algorithms are deterministic since they always because they explore the cases in certain order according to two concepts: coverage and reachability [146, 147]. Besides, because the cases are sorted according to a given heuristic by each algorithm, they always generate the same maintained case-base from the original case-base:

Algorithm 9 ICF

Require: A case-base C, a number of neighbours $k \in \mathbb{Z}$
Ensure: A maintained case-base C^{σ}

```
1: TOREMOVE ← ∅
2: for all c^x ∈ C do
3:    if not correctlyClassified(c^x, C, δ_global, k) then
4:       TOREMOVE ← TOREMOVE ∪ {c^x}
5:    end if
6: end for
7: C^σ ← C
8: C^σ ← C^σ − TOREMOVE
9: repeat
10:   for all c^x = (π^x, ω^x) ∈ C^σ do
11:      for all c^y = (π^y, ω^y) ∈ NearestNeighbour(c^x, C^σ, δ_global, k) do
12:         if (ω^x = ω^y) then
13:            COVERAGE^x ← COVERAGE^x ∪ {c^y}
14:         end if
15:      end for
16:      for all c^y ∈ C^σ − c^x do
17:         for all c^z ∈ t-NearestNeighbour(c^y, C^σ, δ^temporal_global, k) do
18:            if δ_global(c^x, c^z) = 0 then
19:               REACHABLE^x ← REACHABLE^x ∪ {c^y}
20:            end if
21:         end for
22:      end for
23:   end for
24:   TOREMOVE ← ∅
25:   for all c^x ∈ C^σ do
26:      if |REACHABLE^x| > |COVERAGE^x| then
27:         TOREMOVE ← TOREMOVE ∪ {c^x}
28:      end if
29:   end for
30:   C^σ ← C^σ − TOREMOVE
31: until not progress in C^σ
32: return C^σ
```

$$Coverage(c^x, C) = \{c^y \in C | c^y \in NearestNeighbour(c, C) \wedge c^x \text{ solves } c^y\} \tag{2.1}$$

$$Reachability(c^x, C) = \{c^y \in C | c^x \in NearestNeighbours(c^y) \wedge c^y \text{ solves } c^x)\} \tag{2.2}$$

For each concept an algorithms is proposed COV-FP and RFC-FP. Whereas COV-FP sorts the cases according to their *Coverage*, RFC-FP sorts them by their *Reachability* values. Ad-

Algorithm 10 DROP1

Require: A case-base C, a number of neighbours $k \in \mathbb{Z}$
Ensure: A maintained temporal case-base C^{σ}

$C^{\sigma} \leftarrow C$
for all $c^x \in C^{\sigma}$ **do**
  $NEIGHBOUR^x \leftarrow \text{NearestNeighbours}(c, C^{\sigma}, \delta_{global}, k+1)$
  for all $c^y \in NEIGHBOUR^x$ **do**
    $ASSOCIATE^y \leftarrow ASSOCIATE^y \cup \{c^x\}$
  end for
end for
for all $c^x \in C^{\sigma}$ **do**
  $with \leftarrow 0$
  $without \leftarrow 0$
  $C^{\sigma'} \leftarrow C^{\sigma} - \{c^x\}$
  for all $c^y \in ASSOCIATE^x$ **do**
    if $correctlyClassified(c^y, C^{\sigma})$ **then**
      $with \leftarrow with + 1$
    end if
    if $correctlyClassified(c^y, C^{\sigma'})$ **then**
      $without \leftarrow without + 1$
    end if
  end for
  if $without \geq with$ **then**
    $C^{\sigma} \leftarrow C^{\sigma'}$
    for all $c^y \in ASSOCIATE^x$ **do**
      $NEIGHBOUR^y \leftarrow NEIGHBOUR^y - \{c^x\}$
      $c^z \leftarrow \text{NearestNeighbour}(c^y, C, \delta_{global}, 1)$
      $NEIGHBOUR^y \leftarrow NEIGHBOUR^y \cup \{c^z\}$
    end for
    for $c^y \in NEIGHBOUR^x$ **do**
      $ASSOCIATE^y \leftarrow ASSOCIATE^y - \{c^x\}$
    end for
  end if
end for
return C^{σ}

ditionally, RC-FP uses an heuristic known as *RelativeCoverage* which combines the coverage and reachability of the cases to sort them. In particular, the main features of each heuristic are the heuristic they use, wich are the following:

Reach for Cover (RFC) This heuristic estimates the complexity of the case to be solved. That is, this heuristic measures the proportion of cases that can solve the case. If few cases are able to do so, then the cases is understood as difficult to solve, and should be

Algorithm 11 DROP2

Require: A case-base C, a number of neighbours $k \in \mathbb{Z}$
Ensure: A maintained temporal case-base C^{σ}

$C^{\sigma} \leftarrow C$
$C^{\sigma} \leftarrow$ sort the cases in C^{σ} accordingly to their nearest enemy distance
for all $c^x \in C^{\sigma}$ **do**
  $NEIGHBOUR^x \leftarrow \text{NearestNeighbours}(c, C^{\sigma}, \delta_{global}, k+1)$
  for all $c^y \in NEIGHBOUR^x$ **do**
    $ASSOCIATE^y \leftarrow ASSOCIATE^y \cup \{c^x\}$
  end for
end for
for all $c^x \in C^{\sigma}$ **do**
  $with \leftarrow 0$
  $without \leftarrow 0$
  $C^{\sigma'} \leftarrow C^{\sigma} - \{c^x\}$
  for all $c^y \in ASSOCIATE^x$ **do**
    if $correctlyClassified(c^y, C^{\sigma})$ **then**
      $with \leftarrow with + 1$
    end if
    if $correctlyClassified(c^y, C^{\sigma'})$ **then**
      $without \leftarrow without + 1$
    end if
  end for
  if $without \geq with$ **then**
    $C^{\sigma} \leftarrow C^{\sigma'}$
    for all $c^y \in ASSOCIATE^x$ **do**
      $NEIGHBOUR^y \leftarrow NEIGHBOUR^y - \{c^x\}$
      $c^z \leftarrow \text{NearestNeighbour}(c^y, C, \delta_{global}, 1)$
      $NEIGHBOUR^y \leftarrow NEIGHBOUR^y \cup \{c^z\}$
    end for
  end if
end for
return C^{σ}

Algorithm 12 DROP3

Require: A case-base C, a number of neighbours $k \in \mathbb{Z}$
Ensure: A maintained temporal case-base C^{σ}

$C^{\sigma} \leftarrow ENN(C)$
$C^{\sigma} \leftarrow DROP2(C^{\sigma})$
return C^{σ}

retained in the case-base. Formally:

$$RFC(c^x, C) = \frac{reachability(c^x, C)}{|C|} \tag{2.3}$$

The RFC-FP algorithm is noise sensitive since odd or infrequent cases are retained.

Maximal Cover (COV) On the contrary, COV heuristic measures the proportion of cases problem that can solve by a single case.

$$COV(c^x, C) = \frac{coverage(c^x, C)}{|C|} \tag{2.4}$$

With this heuristic, cases within the clusters of solutions are retained, and cases near the frontier between two solutions are removed. Thus, RFC-FP is less sensitive to noise.

Relative Cover(RC) This one tries to measure the relation between what a case can contribute the the knowledge of the case-base and if its knowledge is redundant. Therefore, cases which can be solved by many other cases and at the same time do not solve any other case have high probability of being removed from the case-base.

$$RelativeCoverage(c, C) = \sum_{c^y \in Coverage(c,C)} \frac{1}{|Reachability(c^y, C)|} \tag{2.5}$$

Lastly, the algorithms share the same basis, which can be seen in the algorithm 13.

Algorithm 13 COV-FP, RFC-FP, RC-FP algorithm basis

Require: A raw case base C
Ensure: A mined case base C^σ
1: R $\leftarrow$ RENN(C)
2: $C^\sigma \leftarrow \emptyset$
3: **while** R $\neq \emptyset$ **do**
4: c $\leftarrow$ Next case in R according to COV, RFC or RC
5: $C^\sigma \leftarrow C^\sigma \cup \{c\}$
6: R $\leftarrow$ R - $Coverage(c, R)$
7: $C^\sigma \leftarrow c$
8: **end while**
9:
10: **return** C^σ

The complexity of this algorithms is $O(|C|)$, plus the complexity of sorting the cases according to used heuristic. For instance, a quick-sort ordering has a complexity of $|C|^2$ in the worst scenario, and $|C| \cdot log(|C|)$ in the average scenario.

2.3 Measures for evaluating a CBR system

The accuracy of a given CBR system can be understood as the ability to solve the input problem with the right solution. A common wide practice to analyse a CBR system is the use of a test

set, which is a set of cases from the original case-base, and to build a confusion matrix. Table 2.1 shows an example of a confusion matrix for a CBR system with $\{s_1, s_2, \ldots, s_n\}$ possible solutions and N test cases.

		Actual solution				
		$\mathbf{s_1}$	$\mathbf{s_2}$	$\ldots$	$\mathbf{s_n}$	Total
Output solution	$\mathbf{s_1}$	$p_{1,1}$	$p_{1,2}$	$\ldots$	$p_{1,n}$	$\sum_{i=1}^{n} p_{1,i}$
	$\mathbf{s_2}$	$p_{2,1}$	$p_{2,2}$	$\ldots$	$p_{2,n}$	$\sum_{i=1}^{n} p_{2,i}$
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
	$\mathbf{s_n}$	$p_{n,1}$	$p_{n,2}$	$\ldots$	$p_{n,n}$	$\sum_{i=1}^{n} p_{n,i}$
					Total	N

Table 2.1 Confusion Matrix for a multi-solution classification.

Once all the cases in the test set are solved, then it is possible to calculate statistical measures such as *accuracy*, *recall* (sensitivity) and *specificity* (true negative rate) of one particular solution, and *global recall* and *global specificity*, among others [120]. The formal definitions of these measures are given in the expressions 2.6, 2.7, 2.8, 2.9 and 2.10, respectively.

$$accuracy = \sum_{i=1}^{n} p_{i,i}/N \tag{2.6}$$

$$recall(s_i) = p_{i,i}/\sum_{j=1}^{n} p_{j,i} \tag{2.7}$$

$$specificity(s_i) = \left(\sum_{j\in[1,n]j\neq i} p_{j,j}\right) / \left(\sum_{j,l\in[1,n]j,l\neq i} p_{j,l}\right) \tag{2.8}$$

$$globalrecall = \sum_{i=1}^{n} recall(s_i)/n \tag{2.9}$$

$$globalspecificity = \sum_{i=1}^{n} specificity(s_i)/n. \tag{2.10}$$

Other relevant measures are the achieved reduction rate by the method, and the difference between the retrieval time before and after the appliance of the CBM algorithm [143]. Given the original case-base M and the maintained case-base M^{σ} obtained using the CBM algorithm σ, the reduction rate is defined as:

$$\text{Reduction rate} = \frac{|M^{\sigma}|}{|M|} \tag{2.11}$$

2.4 Evolutionary algorithms

The Evolutionary Algorithms (EAs) are inspired in biological evolution [66], since they simulate biological processes to search for a solution to an optimization problem. According to [176], 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 crucial 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 essential 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 [46, 176]. 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 [25]. Unlike an EA, a MOEA's fitness function returns a value per each objective [178]. Expression 2.12 defines formally the

optimization problem to minimize n objectives:

$$minimize(\Phi(x)) = minimize(\phi_1(x), \phi_2(x), \ldots, \phi_n(x)), \tag{2.12}$$

where x is an individual, Φ 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 [40]. This dominance determines which individual is closer to the optimization objectives. Expression 2.13 defines formally the relation:

$$\begin{aligned} & x \prec y \iff \\ & \forall \phi_i(x), \phi_i(y) \in \Phi(x) : \phi_i(x) \leq \phi_i(y) \wedge \\ & \exists \phi_j(x), \phi_j(y) \in \Phi(x) : \phi_j(x) < \phi_j(y), \end{aligned} \tag{2.13}$$

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.

2.5 Temporal reasoning in CBR

Temporal Reasoning consists of formalizing of the time and providing means to represent and reason about the temporal aspects of the knowledge [160]. This is useful for several areas where the current situation depends on the context of time where the situation is happening, such as medical diagnosis and explanation, reactive system specification, planning, process supervision and natural language understanding.

Thus, performing temporal reasoning requires two main elements: an *extension of the language* for representing the temporal aspects of the knowledge, and a *temporal reasoning system*. On the one hand, the temporal language is needed to represent temporal entities as *fact* and *events*, which for the sake of simplicity, may be seen as the static and dynamic parts of the involved domain, respectively. On the other hand, the Temporal Reasoning System makes use of the temporal language to perform two different tasks that complement each other: assertion of the *temporal consistency* of the given temporal entities, and *querying-answering*. Whereas the first task is focused on determining the truth of the stored knowledge in the Temporal Reasoning System, the querying-answering task consists on providing temporal entities that answers certain querying.

The temporal language represents the idea of time with the use of temporal arguments, which introduces a simple way to describe time in our human language [160], and with a temporal ordering relation $\leq$, the temporal reasoning system may determine the time sorting

between two different temporal arguments. So as to represent both temporal arguments and ordering relation, the temporal language needs of primitives for determining the structure of time. From these primitives, then it is possible to define the temporal relation between temporal arguments.

The main primitives representations are based on the concepts *instants* (also known as points)[20, 77, 108, 109], *intervals* [7–9, 82], and even hybrid approaches [50]. Once the temporal primitives are defined, then it is possible to determine the structure of the time. (i) *Point-based:* the basic primitive is an instant of time, so with this primitive is possible the representation of instantaneous facts through the time. (ii) *Interval-Based:* a primitive contains data regarding the beginning and ending of certain dynamic events. (iii) *Point and Interval-Based Temporal Representations:* this is a mixed representation of the previous primitives, thus this able the creation of a language that represents both instantaneous facts and dynamic events.

In the early years, CBR methodology was focused on non-temporal dependent problems, where the problem descriptions within the cases are snapshots of the domain system. Nonetheless, in the last decade, the inclusion of temporal features within the problems descriptions has gained a relevant role in the CBR systems, as is highlighted by the amount of publications related with CBR systems working with time dependent features [16, 59, 74, 78, 111, 115, 116, 159]. In these works generally the term *temporal case* is understood as cases which have a temporal feature. In particular, among the most studied temporal features are time series, workflows and event sequences, whereas the use of each temporal feature relies on the type of problem which the CBR system is dealing with.

Time series are sequences of values taken from frequent observations of continuous features. In order to clarify this concept, let be the example where the temperature of a room is observed. Figure 2.4 depicts two plots, where the plot at the left portrays the real values of one hypothetical temporal feature that represents the temperature of a given environment. From those values, it is possible to create an approximation by taking samples within a regular frequency. Thus, the plot at the right is an example of samples taken from the plot at the left. With time series the relation of time is given by the instant in which the continuous feature is observed. Since times series are usually a sequence of discrete points, they are seen as vectors of one dimension with each element as one instant. Therefore, in order to compare two time series distances such as Manhattan or Euclidean are used to compute the distance between them. Nonetheless, because dealing with large time series with many points implies larger time computation, then it is frequent to perform a dimension reduction through the use techniques such as discrete Fourier transform (DFT) [2], Discrete Wavelet Transform (DWT) [23], and Dynamic Time Warping [137].

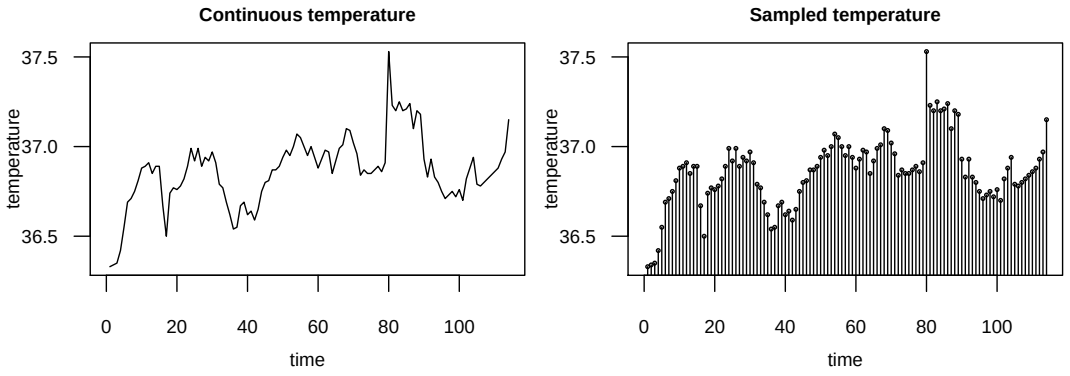


Fig. 2.4 Examples of time series. Whereas the left plot is a continuous time series, the plot at the right is a discrete time series.

Time series are used in those problems where there is involved some type of continuous feature. For instance, process supervision [51], prediction faulty situation [76], process forecast [121], robot control [129], detection of renal failures [118], respiratory sinus arrhythmia diagnosis [122], failure detection on haemodialysis machines [128] and time course prognoses [139]. Furthermore, the CBR community has been addressed theoretical works in order to propose theoretical frameworks that encompass this type of CBR systems [71, 115], and even specialized for the medical domain [117].

Episodes are strings of traces, where this traces may represent a taken action represented by an action. In order to compute the similarity of two episodes, there are different approaches, such as Levenshtein distance [94], Needleman-Wunsch [135], Smith-Waterman [142] string alignments, or a variation o them[177]. Within the CBR research, this type of time structure has been successfully applied in the recognition of task done by computer interactions with the users [33], and in activity task recognition [56].

Workflows are representations of business activities, although they can be used as well to represent industrial process, medical treatments and scientific experiments. They are used with processes which need certain task ordering for their correct execution. Therefore, workflows modelling involves the translation of high-level process requirements into schemas that can be executed by appropriate workflow engines [102]. Generally, the representation of the workflows are deterministic finite automata, where each node represents a task of the process, and the representation of time could be *implicit* or *explicit*. Whereas implicit workflows restrict the order in which the task could be executed or performed, and the explicit workflows are represented with the use of timed automata [10] or temporal constraint satisfaction problem [41] in order to set the windows of time in which the tasks are able for execution [96]. Since workflows are the model of the operation of a process, thus they are subject to execution by a workflow engine, which

generate a task sequences. This task sequence could be understood as event sequences, which are explained next.

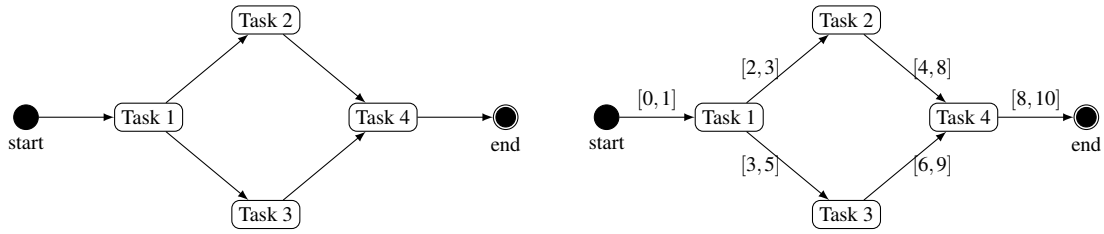


Fig. 2.5 Example of workflow as temporal structure

In figure 2.5, the workflow at the left is a finite automata that represent a process of four tasks. While that workflow only set the order in which the task are executed, the workflow at the right set the windows of time in which the task are subject to execution. When comparing two workflows between them, there are different approaches such as edit distance [112], graph matching [21, 43], graph matching enhanced with semantic labels [14], using temporal similarity for workflows based on the comparison of temporal constraint networks [26]. Finally, workflows has been used successfully in many fields such as medical [27], business [78, 165] and scientific [111] processes.

Event Sequences consists of ordered heterogeneous events in time, where each event is composed of an event type and a time-stamp that represent when the event occurs [104, 114]. While the event types could be either categorical or numeric values, the time-stamp could be a date or just simply a number representing an instant of time. Generally, event sequences are used when the temporal feature presents certain repetitive patterns and there is not a theoretical background to model such behaviour. In contrast with time series, where the temporal feature is observed with a given frequency, with event sequences the time between samples is irregular and not fixed. Besides, the time series are restricted to the observation of one temporal feature, and the event sequences could involve two or more different temporal features. So as to clarify these differences, let be again the example of the recorded temperature of a hypothetical environment, but this time only the local maximum and minimum values are recorded. Consequently, the plot at the left on figure 2.6 is the same plot as in figure 2.4, but with the maximum and minimum values labelled with triangles. In order to create the event sequence, from this collection of values, then two event types are defined: maximum and minimum, and later, they are arranged into the event sequence according to their appearance over time. The plot at the right of figure 2.6 portrays that corresponding event sequence.

The plot at the left in figure 2.6 shows the local maximum and minimum values of a time series. From these values, the it could be possible to arrange them in an event sequence

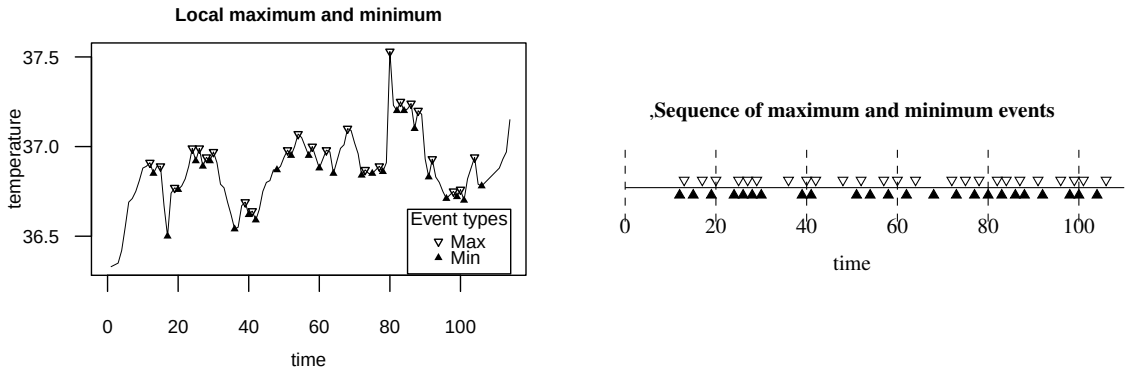


Fig. 2.6 Time-series represented with an event sequence.

as is shown in the plot at the right. Moreover, event sequences could be used so as to represent the execution of workflows, by defining every exiting task in the workflow as an event type, and the time-stamp representing when the task starts. Figure 2.7 portrayed examples of event sequences representing different executions of the workflows shown in figure 2.5. The sequences Seq_1 and Seq_2 are event sequences with events as instants, by comparison with Seq_3, which is an event sequence with events as intervals in order to represent the duration of the different tasks. Heterogeneous event sequences cannot use

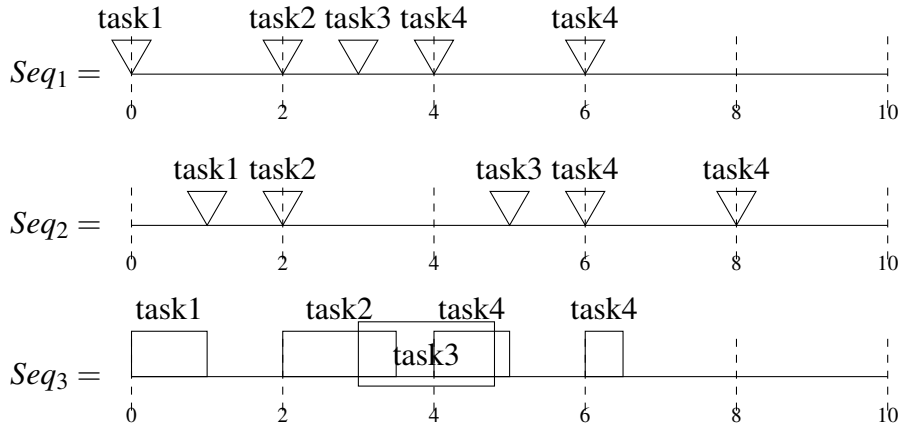


Fig. 2.7 Representation of workflow executions with event sequences.

time series similarity techniques and new proposals are required to compare this kind of cases [133]. To this end, usually the distance between event sequences is done through edit distance [104, 114], which has been adapted from Levenshtein distance [94], heuristics [68, 173], vector based [59] and based on a temporal model of possibilistic temporal constraint networks [74]. Generally, the temporal case structure used in this systems follows the frameworks described in [74] or [117]. Unlike workflows, where a priori it is possible to know the business model, the event sequences are used in those temporal features that present certain repetitive patterns and there is not enough theoretical

background to model such behaviour.

Event sequences have been used in applications with CBR systems:

- In [159] the approach is based on *episodes* of cases, where a case is designed as an static snapshot to record a set of parameters values that characterised an oil drilling process, and the complete operation process is built by linking different cases through the time. Since episodes always have the same number of cases, then the similarity between cases is done using sequence matching pattern.
- For supporting decision systems in oil well drilling [59, 60]. In contrast to the aforementioned work [159], a case contains a sequence of complex events, where these complex events could represent information related with well oil drilling operations. Since the event sequences are transformed into vectors of fixed length, the similarity between cases is done operations of distance between vectors, as for instance cosine similarity and two new distance proposals.
- For monitoring of patients in an intensive care burn unit [73]. In this work, cases are event sequences that represent the treatment of a patient, where the events represent intervals. Their proposal to compute the similarity between temporal case is to transform the event sequences into a possibilistic temporal constraint networks, which are later used as input of a similarity function.
- For prediction of length of stay prediction from event sequences representing clinical treatments [68]. In this work the event types are the different clinical events, such as admission and discharge, and they are arranged into *patient traces*, which follows the structure of an event sequence. In order to compute the distance between two cases, the authors propose similarity function based on a heuristic approach based on two considerations: The higher the number of matched events between two event sequences, the higher the similarity, and the time-stamped information of clinical events should be taken into account.

Finally, to sum up the differences between the three different time structures, the following dimensions are proposed: *temporal representation*, *values types*, *frequently sampling*, and *heterogeneous*.

Temporal representation: the time structure could be classified as *durable* and *instantaneous*. While a durable time structure implies the use of intervals, instantaneous time structures contains only instant as temporal primitive.

Value types: Regardless the temporal representation, the time structure represent the value of the observed temporal features, which can be either numeric or categorical (or symbolic).

Frequently sampled: this means if the time structure requires frequent observations of the temporal features, which implies that given an unit of time then there must be the same number of samples for every time structure.

Heterogeneous : a time structure is heterogeneous if it is able to represent different observations from different temporal features.

The table 2.2 shows the classification of the three explained time structures according to the proposed dimensions.

	Temporal Representation		Value types			
	Instants	Intervals	Numeric	Categorical	Frequently Sampled	Heterogeneous
Time series	✓		✓	✓	✓	
Episodes	✓			✓		
Workflows	✓	✓		✓		
Event Sequences	✓	✓	✓	✓		✓

Table 2.2 Classification of time series, workflows and event sequences according to the proposed dimensions: temporal representation, values types, frequent sampling and heterogeneous.

Chapter 3

A Non-deterministic CBM Algorithm Using Multi-Objetive Optimization

In the present chapter, we propose a CBM algorithm as a non-deterministic Multi-Objective Evolutionary Algorithm (MOEA). To this end, a fitness function is introduced to measure three different objectives based on the Complexity Profile model. Our hypothesis is that the MOEA performing CBM 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 CBR system. We have compared this approach with well known CBM algorithms. From the experimental results, MOEA performing CBM shows regularly good results in three different sets of case-bases, regardless of the amount of redundant and noisy cases, and with a significant potential for further improvement.

This chapter has been developed with the help of Susan Craw and Stewart Massie from the Robert Gordon University.

3.1 CBM as an multi-objective optimization problem

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 [25]. 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 [5, 70, 81], 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.

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 [126, 169, 170], 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 [106, 107]. In the last decades,

Multi-Objective Evolutionary Algorithms (MOEAs) have been applied successfully in multi-objective optimization problems [25, 46, 176]. 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 chapter is organised as follows. The next section gives an overview of existing work. Section 3.2 describes how to represent the levels of redundant and noisy cases of a case-base. Section 3.3 explains how to perform CBM with a MOEA that optimises three different types of objectives. In section 3.4, we evaluate the MOEA with different case-bases, and other CBM algorithms. Lastly, sections 3.5 and 3.6 discuss the experimental results and highlight our conclusions, respectively.

3.2 Representing the redundancy and noise levels of a case-base

Massie *et al.* [107] 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 3.1 describes the complexity function for a case:

$$complexity(c,k) = 1 - \frac{1}{k}\sum_{i=1}^{k} p(c,i), \tag{3.1}$$

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:

$$error(M,k) = \frac{1}{|M|}\sum_{c\in M} complexity(c,k). \tag{3.2}$$

$$noise(M,k) = \frac{|\{c \in M | complexity(c,k) \geq \varepsilon\}|}{|M|}. \tag{3.3}$$

$$redundancy(M,k) = \frac{|\{c \in M | complexity(c,k) = \rho\}|}{|M|}, \tag{3.4}$$

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 $\varepsilon = 0.5$ and $\rho = 0$ confirm that Complexity Profiling is correlated with the accuracy of the CBR system[107].

From the local complexities to create graphically the complexity profile of a given case-base is possible. The cases are ranked in ascending order of complexity along the horizontal axis and the local complexity of each is plotted on the vertical axis. Figure 3.1 shows a typical profile. With $\varepsilon = 0.5$ 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.

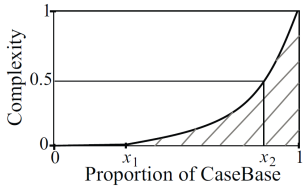


Fig. 3.1 Complexity Profile of Case-Base

3.3 Perfoming CBM with a MOEA

So as to perform CBM with a MOEA, two key issues must be carried out. On the one hand, to represent a case-base as an individual of the population. On the other hand, the definition of a fitness function must be done in order to evaluate the suitability of the individual for being an adequate maintained case-base.

3.3.1 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 $x = x_1 x_2 \ldots x_{n-1} x_n$ 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:

$$\mathscr{M} : X \rightarrow \wp(M)$$
$$\mathscr{M}(x) = \overline{x} = \{c_i \in M | x_i = true\}.$$

For example, given the individual x with all elements set to true, $\mathscr{M}(x) = M$, otherwise if all elements are set to false then $\mathscr{M}(x) = \emptyset$. 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 3.2 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 $\overline{x}$.

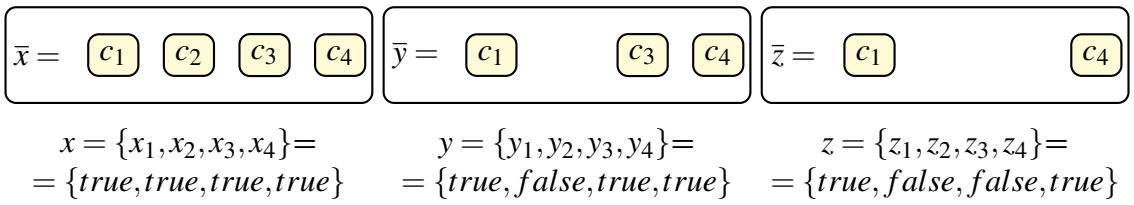


Fig. 3.2 Three case-bases $\overline{x}, \overline{y}$ and $\overline{z}$ and their corresponding individuals x, y and z.

3.3.2 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 $\mathbf{O_{reduncancy}}$ 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:

$$\begin{aligned} \Phi :& X, \mathbb{N} \rightarrow \mathbb{R}^3 \\ \Phi(x,k) =& (O^x_{size}, O^x_{reduncancy}, O^x_{error}) = \\ =& \begin{cases} (O^x_{size} = f_{size}(x,k), \\ O^x_{reduncancy} = redundancy(\bar{x},k), \\ O^x_{error} = error(\bar{x},k)) \end{cases} \end{aligned} \tag{3.5}$$

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:

$$\begin{aligned} f_{size} :& X, \mathbb{N} \rightarrow \mathbb{R} \\ f_{size}(x,k) =& (threshold(M) - length(x))^2, \\ threshold(M) =& (|M| * (1 - redundancy(M,k)). \end{aligned} \tag{3.6}$$

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 $\bar{x}$ and the redundancy threshold. This objective is squared to penalize those individuals with a greater number of cases. The values returned by functions $redundancy(\bar{x},k)$ and $error(\bar{x},k)$ in the fitness function (expression 3.5) oppose each other since a lower error rate often results in a higher redundancy and vice versa.

3.3.3 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:

$$\begin{aligned} &\Phi(x) \prec \Phi(y) \iff \\ \iff &\begin{cases} \left(O^x_{size} \leq O^y_{size} \wedge O^x_{redundancy} \leq O^y_{redundancy} \wedge O^x_{error} \leq O^y_{error}\right) \wedge \\ \wedge \left(O^x_{size} < O^y_{size} \vee O^x_{redundancy} < O^y_{redundancy} \vee O^x_{error} < O^y_{error}\right) \end{cases} \end{aligned} \tag{3.7}$$

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

3.3.4 NSGA-II

In this chapter we use the MOEA NSGA-II [40] 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 14, given a population P returns a list of the non-dominated fronts $\mathscr{F}$, where the individuals in front $\mathscr{F}_i$ dominate those individuals in front $\mathscr{F}_{i+1}$. 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 15).

Some parameters have to be set up at the beginning, such as the number of generations and the number of individuals N 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 16. In each iteration, population P_t and Q_t are joined to create the population R_t, whose number of

Algorithm 14 fast-nondominated-sort(P)

Require: A population P, a fitness function Φ
Ensure: list of the non-dominated fronts $\mathscr{F}$

for $x \in P$ **do**
  for $y \in P$ **do**
    if $\Phi(x) \prec \Phi(y)$ **then**
      $S_x \leftarrow S_x \bigcup \{q\}$
    else
      if $\Phi(y) \prec \Phi(x)$ **then**
        $n_x \leftarrow n_x + 1$
      end if
    end if
  end for
  if $n_x = 0$ **then**
    $\mathscr{F}_1 \leftarrow \mathscr{F}_1 \bigcup \{x\}$
  end if
end for
$i = 1$
while $\mathscr{F}_i \neq \emptyset$ **do**
  $\mathscr{H} \leftarrow \emptyset$
  for $x \in \mathscr{F}_i$ **do**
    for $y \in S_x$ **do**
      $n_y \leftarrow n_y - 1$
      if $n_y = 0$ **then**
        $\mathscr{H} \leftarrow \mathscr{H} \bigcup \{y\}$
      end if
    end for
  end for
  $i = i + 1$
  $\mathscr{F}_i \leftarrow \mathscr{H}$
end while
return $\mathscr{F}$

Algorithm 15 crowding-distance-assignment($\mathscr{I}$)

Require: A set of individuals $\mathscr{I}$
Ensure: Each individual within $\mathscr{I}$ with a density measure.

$l \leftarrow |\mathscr{I}|$
for $i \in [1, N]$ **do**
  $\mathscr{I}[i] \leftarrow 0$
end for
for each objective m **do**
  $\mathscr{I} \leftarrow \text{sort}(\mathscr{I}, m)$
  $\mathscr{I}[1]_{\text{density}} \leftarrow \infty$
  $\mathscr{I}[l]_{\text{density}} \leftarrow \infty$
  for $i \in [2, (l-1)]$ **do**
    $\mathscr{I}[i]_{\text{density}} \leftarrow \mathscr{I}[i]_{\text{density}} + (\mathscr{I}[i+1].m - \mathscr{I}[i-1].m)$
  end for
end for

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 3.3.5.

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 3.3 depicts graphically, the process for creating a new generation of individuals.

Algorithm 16 NSGA-II main loop

Require: A fitness function Φ, 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: $\mathscr{F} \leftarrow$ fast-nondominated-sort(R_t, Φ)
6: **while** $|P_{t+1}| < N$ **do**
7: crowding-distance-assignment$(\mathscr{F}_i)$
8: $P_{t+1} \leftarrow P_{t+1} \bigcup \mathscr{F}_i$
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**

From the population R_t, the individuals are sorted by non-dominance between them. Later the set $\mathscr{F}$ of fronts are created, where $\mathscr{F}_1$ are non dominated individuals, $\mathscr{F}_2$ are dominated by the individuals in $\mathscr{F}_1$ and so on. 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}. Finally, the population P_{t+1} is used to create the population Q_t. For further details of NSGA-II algorithms see [40].

3.3.5 Creation of a new population and 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 operations mutation and crossover 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

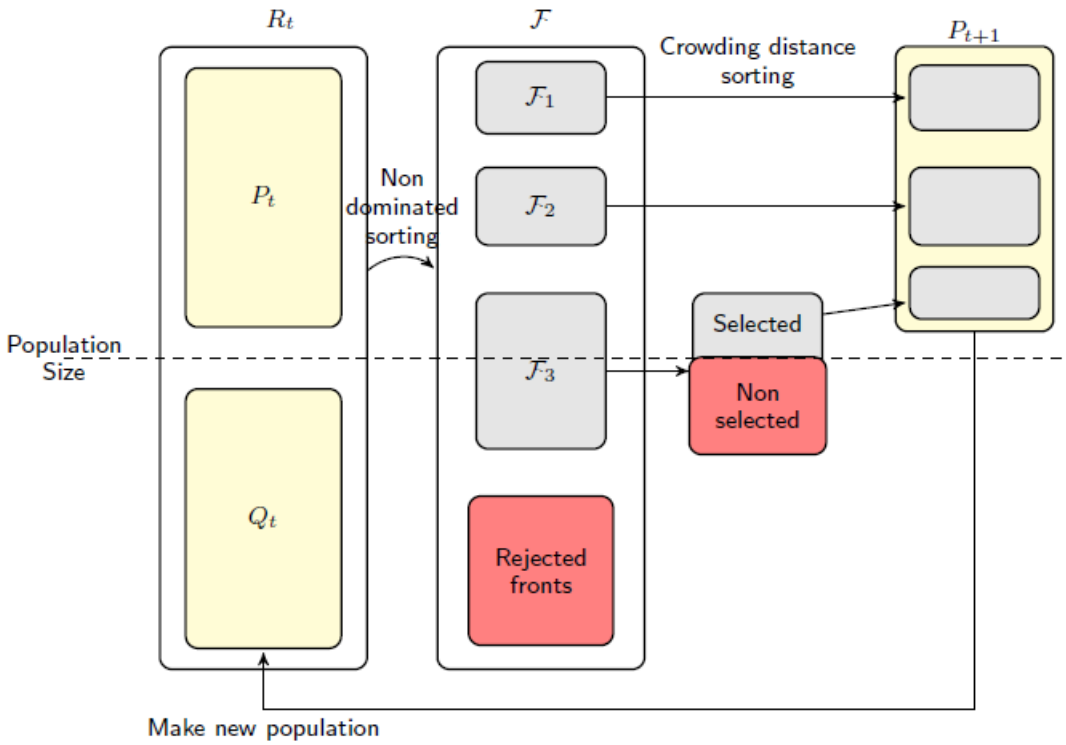


Fig. 3.3 Basic NSGA-II operation to create the next population

tournament is repeated until Q_{t+1} reaches the limit population.

Given an individual $x = x_1x_2\ldots x_n$, 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 the individual x and given a second individual $y = y_1y_2\ldots y_n$, 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:

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

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

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

3.3.6 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 $|M| * (1 - redundancy(M,k))$ (expression 3.5). Figure 3.4 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 $\mathbf{O_{error}}$.

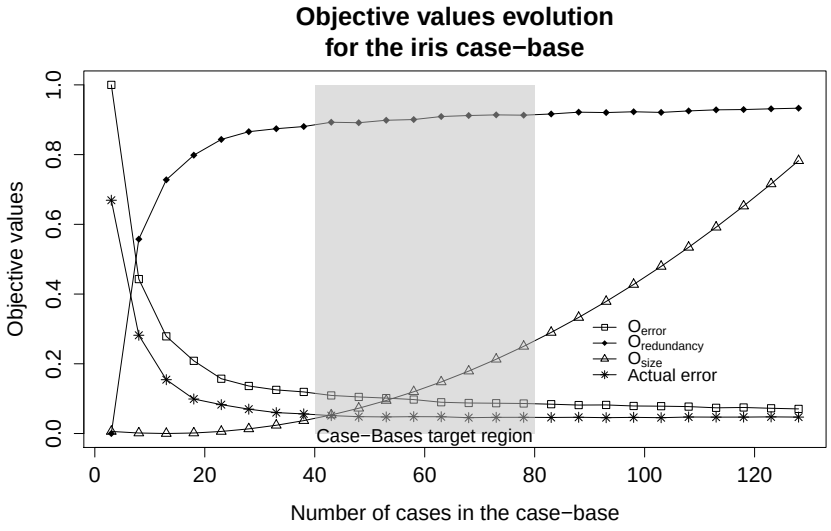


Fig. 3.4 Evolution of the objectives values for Iris dataset.

3.4 Experiments

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 [49]. 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 works with different types of cases-bases. The details of each case-base is shown in the table 3.1, and the figure 3.5 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.

Redundancy: anneal, breast-w, ionosphere, soybean

Noisy: bridges, contraceptive, yeast, vehicles

Mix: australian, hepatitis, sonar, vowel

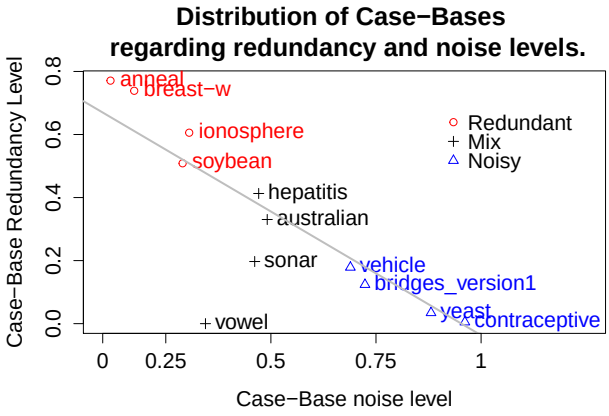


Fig. 3.5 Clustering of case-bases in relation to redundancy and noise levels. There are three clusters: *redundant*, *noisy*, and *mix*.

Table 3.1 is the summary of relevant information of 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.

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

- Population size $\in \{30, 50\}$
- Cross-over probability $\in \{0.9, 0.925, 0.95\}$

		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

Table 3.1 A summary of relevant information of the datasets in the evaluation

- 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.

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 3.6 to 3.11 show all the gathered results of NSGA-II for each set of case-bases: *redundant*, *mix* and *noisy* case-bases. The purpose of the figures 3.6,3.8 and 3.10 is to show that the NSGA-II algorithm converges to a set of solutions that have similar accuracies. While the figures 3.7, 3.9 and 3.11 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 return the best accuracy average.

Table 3.2 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 with 30 individual is enough to get maintained case-bases that converge to a stable accuracy and reduction rates.

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 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

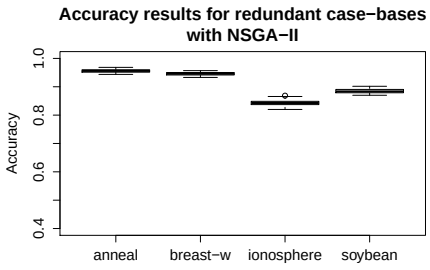


Fig. 3.6 Accuracy results for the redundant case-bases for all the possible settings combinations.

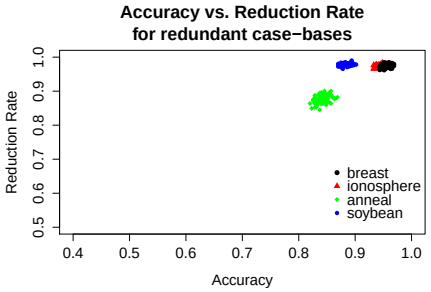


Fig. 3.7 Distribution of Accuracy against reduction rate for the redundant case-bases for all the possible settings combinations.

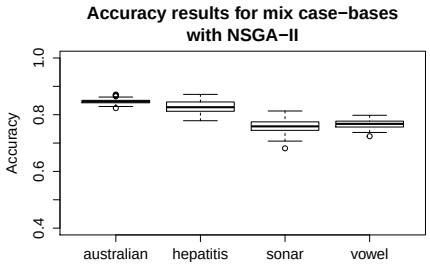


Fig. 3.8 Accuracy results for the *mix* case-bases for all the possible settings combinations.

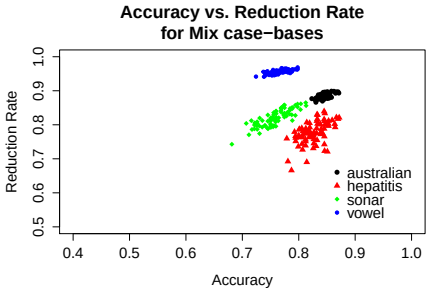


Fig. 3.9 Distribution of Accuracy against reduction rate for the *mix* case-bases for all the possible settings combinations.

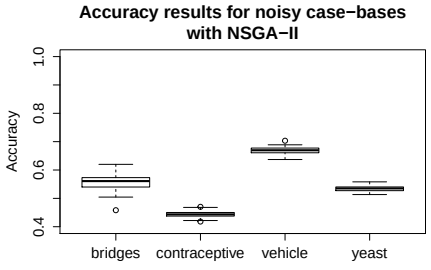


Fig. 3.10 Accuracy results for the *noisy* case-bases for all the possible settings combinations.

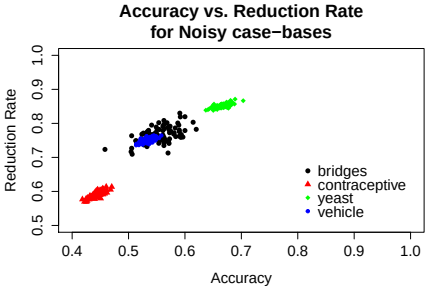


Fig. 3.11 Distribution of Accuracy against reduction rate for the *noisy* case-bases for all the possible settings combinations.

	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	0.95408	**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

Table 3.2 Evaluation results summary

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 3.12,3.13 and 3.14, 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 for accuracy means that the CBR system is returning frequently the proper 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.

3.5 Discussion

As can be seen in figure 3.12, 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

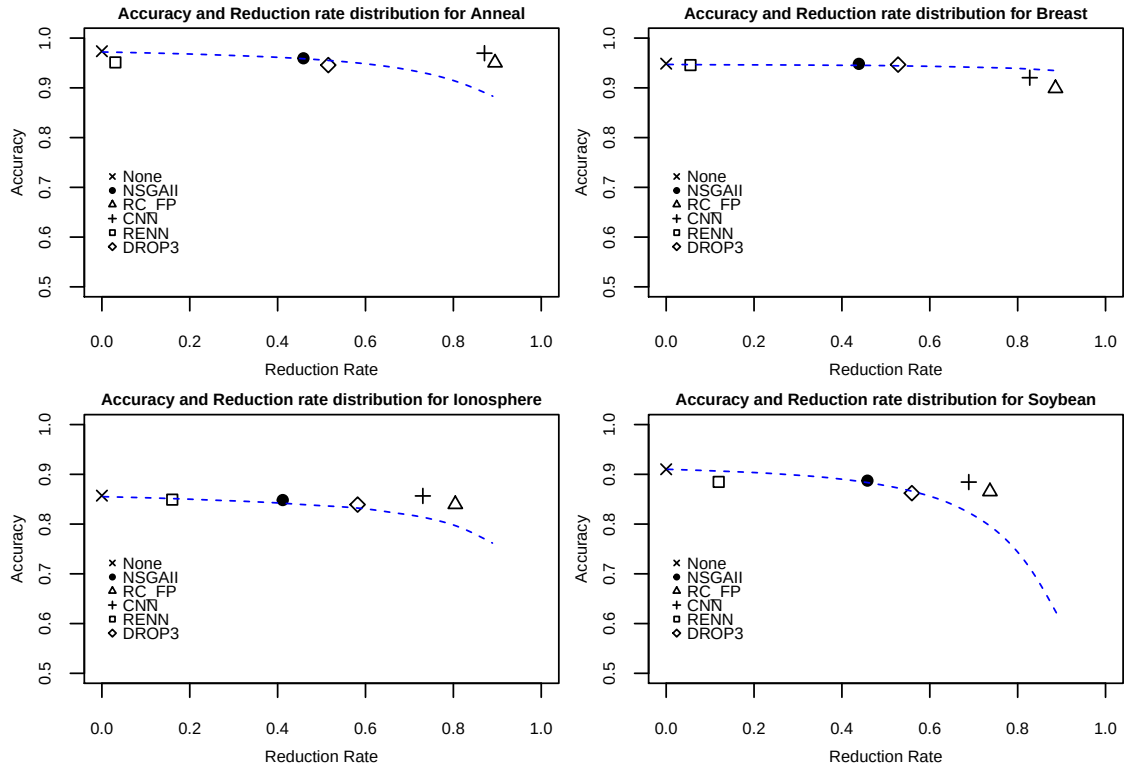


Fig. 3.12 Accuracy and reduction rate distribution for all the studied CBM algorithms for the redundant case-bases: anneal, breast, ionosphere and soybean.

good, the number of 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 3.13, 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 (3.14) have high levels of noise and low redundancy levels. With

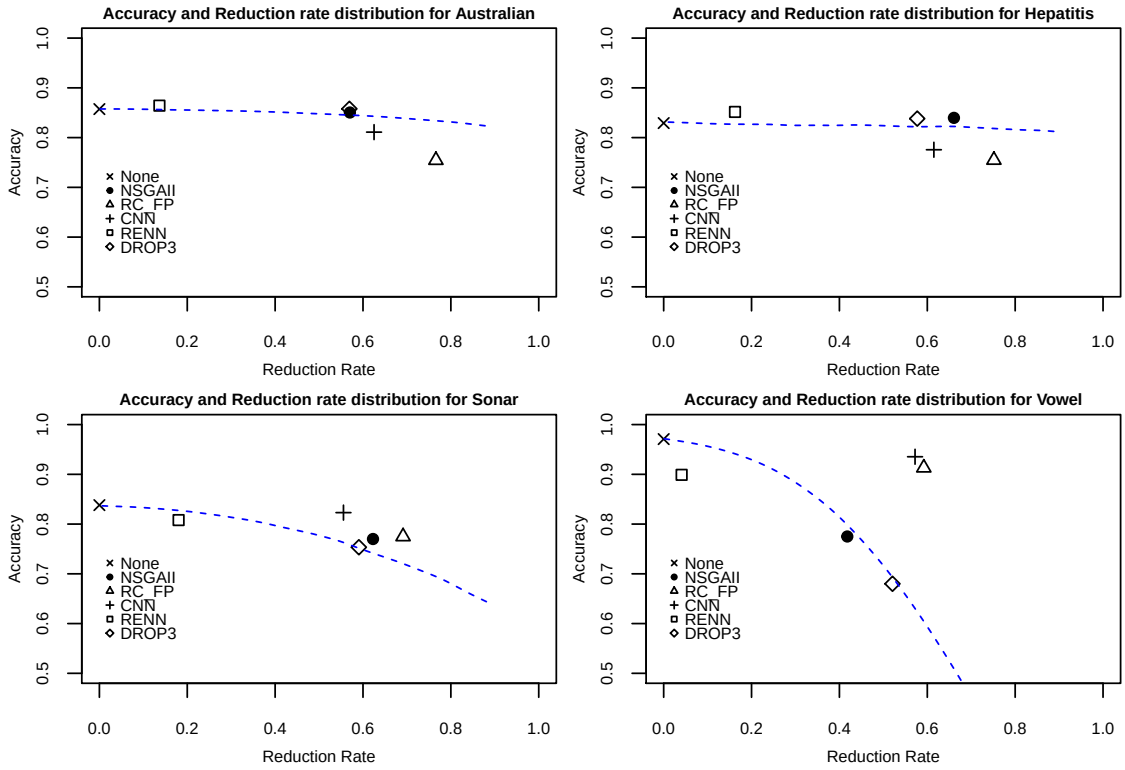


Fig. 3.13 Accuracy and reduction rate distribution for all the studied CBM algorithms for the mix case-bases: australian, hepatitis, sonar and vowel.

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 this size of case-base, it is likely than the maintained case-base is not over-fitted. 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

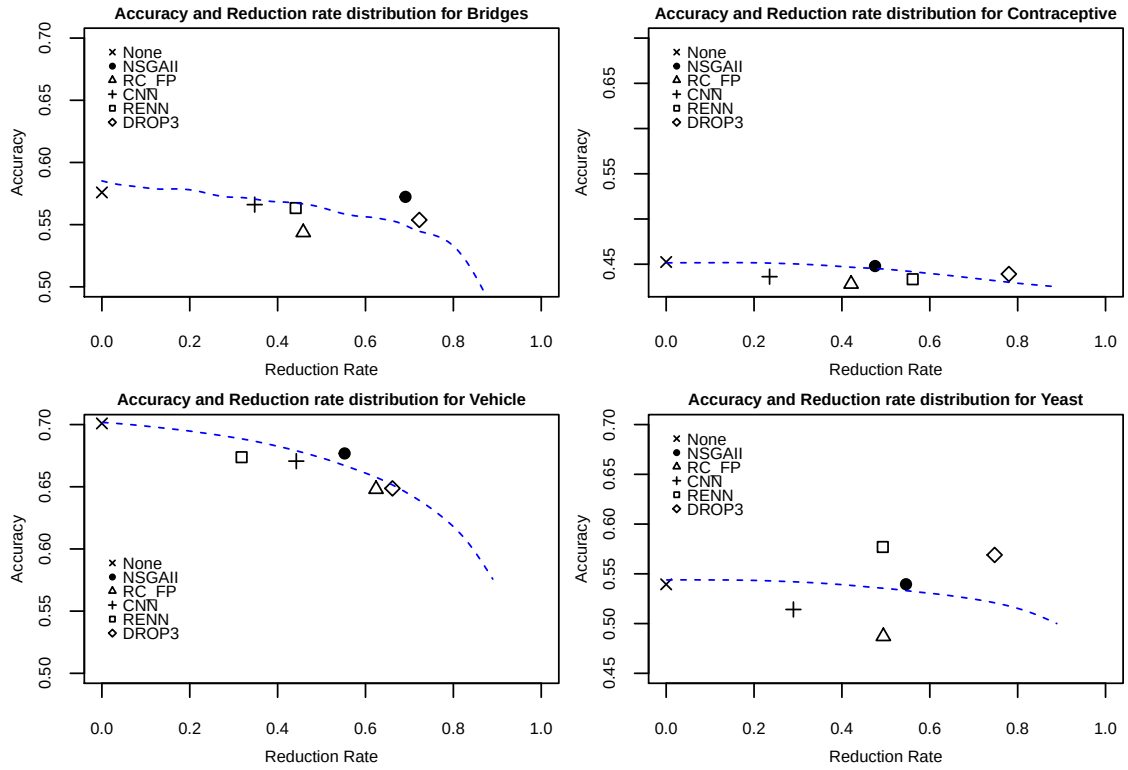


Fig. 3.14 Accuracy and reduction rate distribution for all the studied CBM algorithms for the noisy case-bases: bridges, contraceptive, vehicle and yeast.

cases to be deleted from the case-bases.

3.6 Conclusions

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 chapter we have approached the CBM task as a multi-objective optimization problem that may be solved with a MOEA. In particular, the optimization problem is divided into three different ans 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 a 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 [58, 72, 90], and achieve good maintained case-bases.

Chapter 4

Evaluating Case-Base Maintenance Algorithms

The success of a CBR system closely depends on its knowledge-base, named the case-base. The life cycle of CBR systems usually implies updating the case-base with new cases. However, it also implies removing useless cases for reasons of efficiency. This process is known as Case-Base Maintenance (CBM) and, in recent decades, great efforts have been made to automatise this process using different kind of algorithms (deterministic and non-deterministic). Indeed, CBR system designers find it difficult to choose from the wealth of algorithms available to maintain the case-base. Despite the importance of such a key decision, little attention has been paid to evaluating these algorithms. Although classical validation methods have been used, such as Cross-Validation and Hold-Out, they are not always valid for non-deterministic algorithms. In this chapter, we analyse this problem from a methodological point of view, providing an exhaustive review of these evaluation methods supported by experimentation. We also propose a specific methodology for evaluating CBM algorithms (the $\alpha\beta$ evaluation). Experiment results show that this method is the most suitable for evaluating most of the algorithms and datasets studied.

4.1 Introduction

CBR is based on experience-solving old problems to find a solution to new problems [1]. In this way, given that a problem and its solution conform a case, when new problems are solved then new cases are created and stored in a case-base. Although the learning ability of CBR is an important advantage, this characteristic is not free of drawbacks. For example, the addition of new cases to the case-base could degrade a system's performance [67, 113, 158] because case-bases with many cases need a long time to retrieve cases similar to an input query. Some reasons for increase in retrieval time are the scalability of the data structures that represent the case-base, and the case descriptions for both problem and solution. Consequently, calculation of the similarity function between cases is complex too.

Instead of incrementing hardware power, the alternative way to tackle performance problems is to use Case-Base Maintenance (CBM), whose main purpose is to update the existing case-base in order to maintain problem-solving competence [126], defined as the ability to solve new problems within an acceptable interval of time.

Two different strategies can be considered in CBM. The first strategy focuses on optimisation of the software that implements the case-base, such as tuning the parameters of the retrieval step [35] or changing the data structure used to model the case-base. The second strategy is aimed at removing cases from the case-base according to a given deletion policy [92], deleting redundant or noisy cases. In other words, the goal is to obtain smaller case-bases with the same problem-solving competence. The resulting case-base may not only replace the original case-base, but it may also be used as an index to enhance the retrieval of similar cases

[81, 148].

Algorithms that delete cases have been widely studied in the Machine Learning field [3, 18, 62, 89, 146], as well as in the CBR community [36, 37, 75, 93, 126, 132, 158]. All these works show that reducing the number of cases through a CBM process is an effective approach to decreasing the retrieval time.

When using CBM algorithms, the question that remains is whether the maintained case-base is better, equivalent to or worst than the original one. The straightforward approach to answering this question is to use classical Machine Learning evaluation methods, such as Hold-Out and Cross-Validation, to analyse and compare the outcomes of the same CBR system obtained with these two case-bases. These methods divide the case-base into a number of training and test sets, and while the training set plays the role of the case-base, the test set contains descriptions of all the input problems to be solved. In such a way, every problem in the test set is solved using the cases within the training set, which enables the accuracy, sensitivity and specificity to be calculated. Whereas Hold-Out uses one training set and one test set, the Cross-Validation usually involves up to five or ten training and test sets in order to finally average the results and obtain more reliable results. Thus, once the CBM has built the maintained case-base from the original, the evaluation method uses both the original and the maintained case-bases and builds the training and test sets. However, CBM algorithms may drastically reduce the size of the original case-base, and there will not be enough cases in the maintained case-base to resolve all the problem from the domain; that is, the competence of the CBR system is reduced.

To avoid this inconvenience, the evaluations for the CBR systems are performed in a different way. This approach consists of using the CBM algorithm on the training set created, instead of the CBM algorithm on the complete case-base. This strategy can be used both in Hold-Out evaluation [5, 36, 81, 95, 110, 126] and in Cross-Validation [42, 107].

However, this evaluation strategy has its own drawbacks. Non-deterministic CBM approaches, such as those described in [5, 70, 81, 97], return different maintained case-base outputs when multiple executions are carried out using the same original case-base, and it is necessary to execute the CBM algorithm several times taking the training set as input in order to obtain an average and achieve more reliable evaluation results.

In this chapter, we propose a novel evaluation method designed to deal with deterministic and non-deterministic CBM algorithms. In order to demonstrate the suitability of our approach, we recount the exhaustive experimentation made to compare the results given by our method and by other known evaluation methods.

Although different evaluation strategies are used in many publications, providing correct results, these results are difficult to compare precisely because they are obtained by different evaluation methods: for instance, building a test set with 40% or 30% of the cases, or repeating

the Cross-Validation up to 10 times or executing it only once. The present chapter is intended to clarify the differences between the best known evaluation methods to reveal their advantages and disadvantages. Furthermore, a new evaluation method is presented to solve those disadvantages that arise when working with non-deterministic algorithms.

The remainder of this chapter is structured as follows: the next section revises the related work to evaluation of CBM algorithms. Section 4.3 introduces our $\alpha\beta$ evaluation proposal for evaluating of CBM algorithms. Section 4.4 shows the experimentation results after performing a set of CBM algorithms with a set of experimental case-bases from the UCI repository [49]. Section 4.5 is a discussion of the results from section 4.4. Finally, in section 4.6, we give our conclusions.

4.2 How CBM is evaluated

Once a CBM finishes its execution, the next step is to figure out whether the resulting maintained case-base is better, worse or equivalent to the original one. Generally, this is done through a comparison of evaluation results or statistical measures, for instance accuracy, false positive and the kappa index, which are given by the CBR system using the original and maintained case-bases.

Two of the most common evaluation methods are Cross-Validation and Hold-Out, which provide a good estimations of statistical measures, e.g. accuracy [63]. The basis of both evaluations is similar since they divide the case-base into subsets of cases, where at least one of them is the test set and the remaining became part of the training set. Usually, the training set conforms the case-base of the CBR system, whereas the test set are the input problems to solve. Whereas the Hold-Out divides the case-base just into two to create the test set and training set, usually using 30% of the cases to create the test set, Cross-Validation divides the case-base into n subsets, known as folds. The number of folds decides the number of evaluations, thus 10 Cross-Validation folds make up 10 training sets and test sets. Generally, five or ten folds are suitable to perform an evaluation [17, 85].

Working with one particular evaluation method rather than the other depends on the type of case-base in question. Hold-out is suitable when the training and test sets are large enough for them to be representative of the original case-base and the problem domain. However, in situations where this is not possible, perhaps because the case-base is quite small, Cross-Validation is the more appropriate evaluation approach.

The question remains as to how these evaluation methods would be applied. The first idea is to apply the CBM algorithm to the original case-base, and then to evaluate the maintained case-base with either Hold-Out or Cross-Validation. However, this approach has an important drawback: the resulting maintained case-base would probably be smaller than the original

case-base due to the removal of redundant and noisy cases; consequently, the test and training sets generated are not equivalent to the original case-base. In this case, statistical measures are not reliable because they are not representative of reality.

For this reason, both Cross-Validation and Hold-Out have been widely used in the CBR literature with some modifications. Instead of first applying the CBM algorithm, the evaluation method is used on the original case-base to generate the training and test sets. Then, the CBM is applied to the training sets, which are equivalent to the original case-base. Below, we shall comment on the most common evaluation method on the CBR field.

In [5, 81] the accuracy of the maintained CBR system is studied using Hold-Out, and a k-nearest neighbour (K-NN) is used as a classifier. Following the same evaluation method, in [95] compare the retrieval time and the size reduction rate.

Another evaluation that uses Hold-Out and a K-NN classifier is presented in [110] and [36].In [110] each case-base is divided randomly into two partitions: an 80% training set and a 20% test set. In [36] the evaluation divides the case-base at random into three splits: 60% for the training set, 20% for the test set and the remaining is ignored. In both papers, CBM algorithm is applied to each training set in order to reduce their size. Later, the maintained training sets are validated with the test set to obtain the accuracy. The process is repeated from 10 to 30 times. At the end, the authors average the accuracies to obtain the final accuracy estimation before comparing it with the accuracy given by the original case-base.

[37] propose a more complex variation of Cross-Validation. The authors suggest splitting the case-base into three: 60% for the training set, 20% for the test set and 20% for cross-validation. These sets are used in a two-step evaluation process. Given a set of CBM algorithms to evaluate, in the first step all the CBM algorithms are applied to the training set and the resulting case-bases are validated using the cross-validation set. In the second step, the case-base with best results is chosen and the test set is used to obtain the final evaluation results. The observed measures are the percentage of cases deleted and the accuracy given using a K-NN classifier.

In [107], a 10-fold Cross-Validation is used. The CBM algorithms are applied to each training set and the resulting maintained case-base size is recorded. Test set accuracy is measured for the original case-base and for each of the maintained case-bases created by the CBM algorithms. The evaluation is performed up to 10 times, and finally, the accuracy results are averaged and compared with the accuracy average given by the original case-base.

Some authors propose new measures such as Competence to evaluate a case-base [126]. This measure highlights the proportional improvement in terms of accuracy of the maintained case-base against the original. The authors use Hold-Out as evaluation method, using 30% of the cases as test set and the remaining as training set. The CBM algorithm is applied to the training set and validated on the test set. The observed measures are the reduction size

ratio and their proposed Competence measure. The process is repeated up to 20 times, and the results are averaged to compare them with the original measures. The classifier 1-NN is chosen to retrieve the most similar cases to an input problem.

4.3 $\alpha\beta$ evaluation method

In this chapter, we propose the $\alpha\beta$ evaluation method. Whereas classical methods are aimed to evaluate the results of a CBR system, such as Cross-Validation and Hold-Out, the aim of $\alpha\beta$ is to evaluate the CBM algorithm and its effects on the CBR system after the maintenance is finished. The main differences with classical methods are the use of the CBM algorithm on the training set instead of applying it on the original case-base, and the introduction of two new parameters: α and β. The α parameter is aimed at enhancing the reliability of the results by repeating the CBM algorithm under observation is non-deterministic, while the β parameter is intended to repeat the entire Cross-Validation process β times. Furthermore, $\alpha\beta$ evaluation performs a pre-processing of the original case-base to decrease the time needed to run the evaluation.

The $\alpha\beta$ evaluation produces two sets of decision scores for a given CBR system using a particular case-base. The first set is composed of the *Performance Decision Scores*, while the second set is composed of the *Quality Decision Scores*. Whereas the *Performance Decision Scores* comprise measures to quantify the reduction of the case-base and the CBM execution runtime, the *Quality Decision Scores* conform basic statistical measures that helps to determine the suitability of the CBR system to solve the problem domain from the given case-base. For instance, some of these statistical measures are accuracy, recall and specificity, whose definitions are given by the expressions 2.6, 2.9 and 2.10.

In order to compute the decision scores, the evaluation begins with a case-base pre-process. First, the most relevant attributes of the problem description are selected. The purpose of this attribute selection is to decrease the number of problem features of a case in order to reduce the similarity complexity between cases and to speed up the evaluation process. Attribute selection is made with a genetic algorithm, whereby each individual is a binary string with element values in the domain $\{true, false\}$. Each individual represents features selected from the problem description. If the *i*-th position is set to $true$ then the *i*-th feature is selected. That is, an individual with *n* positions represents a case-base that has *n* features to describe a problem, and the positions with the value set to $true$ are the selected features of the problem. The population size and the number of generations are set to 100. The one point cross-over probability is 0.9 and the mutation is 0.033. The fitness value of each individual is the accuracy given by a 10 fold Cross-Validation using the equivalent case-base to the individual. Once feature selection is completed, duplicate cases are deleted and where missing values are present, they are replaced

with the mean value of the corresponding feature. Finally, all the case features are normalized in the interval $[0, 1]$.

4.3.1 $\alpha\beta$ evaluation process

Once the data pre-processing finishes, the evaluation process starts. Algorithm 17 depicts in detail the steps needed to perform the $\alpha\beta$ evaluation:

Initialization of performance and quality decision scores: The first step is to initialise the Performance and Quality Decision Scores. Since quality decision scores rely on statistical measures, confusion matrices are used to compute the scores. For each given CBM algorithm, the decision scores are initialised (Algorithm 17 in lines 1 to 4). Then, confusion matrices are created, which have as many dimensions as the number of different solutions that exist in the case-base (Algorithm 17 from line 6 to 8). CM_σ refers to the confusion matrix of the σ algorithm, and $CM[x, y]$ denotes the number of times that the actual solution of the query case is x and the solution provided by the CBR system is y, while both x and y might be the same solution or not. Each given CBM algorithm σ has its own quality and performance decision score, represented as ϕ_σ and ρ_σ, respectively.

β loop: In each iteration a new training set and test set are generated. This process is repeated β times (Algorithm 17, lines 9 to 26).

Training and test sets creation: The original case-base M is divided into 10 sub-case-bases with a similar number of cases. These sub-case-bases are denoted by M_i. Stratification is used to improve the representativeness of each solution to build the sub-case-bases. Furthermore, every case in M is present in at least one of the sub-case-bases M_i. The training sets, denoted by $\overline{M_i}$, comprise those cases in M that are not in M_i. The function *complementary* create the training sets (Algorithm 17, lines 10 to 12).

Training and test sets iteration: Since CBM algorithms are executed on each $\overline{M}_i$, and the resulting case-base is evaluated using the case-bases in M_i as queries (Algorithm 17). The loop in line 11 iterates over each i-th element of both sets.

α loop: Because CBM algorithms may generate different maintained case-bases with the same input case-base, the loop generates a set of maintained case-bases for each algorithm σ in order to generate a high number of results, and, consequently, to better estimate the real values associated with the quality decision scores (Algorithm 17, lines 13 to 24).

CBM execution and confusion matrix updating: all the CBM algorithms, Σ, are performed on each $\overline{M}_i$ to generate a new maintained case-base $\overline{M}_i^\sigma$. For each case in

M_i with solution x, the CBR system using the $\overline{M}_i^{\sigma}$ returns a solution y. With both solution x and y, the confusion matrix CM_{σ} is updated (see algorithm 17, lines 14 to 23).

Calculation of decision scores: As each CBM algorithm σ is executed $10\alpha\beta$ times, the performance decision score ρ_{σ} contains the average execution time and the reduction rate. The quality decision scores ϕ_{σ} are calculated from the confusion matrix CM_{σ} (see algorithm 17, lines 27 to 30.

Algorithm 17 $\alpha\beta$ Evaluation

Require: A CBR system with a case-base M, and a set Σ of CBM algorithms.
Ensure: A set of decision scores for each CBM algorithm.
1: **for all** $\sigma \in \Sigma$ **do**
2: $\phi_{\sigma} \leftarrow (\phi_{\sigma}^{accuracy}, \phi_{\sigma}^{specificity}, \ldots)$ {quality decision scores, initialised to 0}
3: $\rho_{\sigma} \leftarrow \left(\rho_{\sigma}^{time}, \rho_{\sigma}^{reduction}\right)$ {performance decision scores, initialised to 0}
4: **end for**
5: $S \leftarrow$ number of different solutions in M
6: **for all** $\sigma \in \Sigma$ **do**
7: $CM_{\sigma} \leftarrow \begin{bmatrix} 0_{1,1} & \cdots & 0_{1,S} \\ \vdots & \ddots & \vdots \\ 0_{S,1} & \cdots & 0_{S,S} \end{bmatrix}$ {Confusion matrix of $S \times S$ dimensions}.
8: **end for**
9: **for** 1 **to** β **do**
10: test $\leftarrow \{M_i : M_i \subset M \wedge M_i \text{ is stratified} | i \in [1,10] \wedge \forall c \in M, \exists c \in M_i\}$.
11: **for all** $M_i \in$ test **do**
12: $\overline{M}_i \leftarrow complementary(M_i)$
13: **for** 1 **to** α **do**
14: **for all** $\sigma \in \Sigma$ **do**
15: $t \leftarrow current\_time$
16: $\overline{M}_i^{\sigma} \leftarrow \sigma(\overline{M}_i)$ {Perform the CBM algorithm σ}
17: $\rho_{\sigma} \leftarrow (\rho_{\sigma}^{time} + (t - current\_time), \rho_{\sigma}^{reduction} + \frac{|\overline{M}|_i^{\sigma}}{|M|})$
18: **for all** $p \in M_i$ **do** {loop over the cases in the test set}
19: $x \leftarrow solution(p)$ {Get the real solution}
20: $y \leftarrow CBR(\overline{M}_i^{\sigma}, p)$ {Get the solution given by the CBR system}
21: $CM_{\sigma}[x,y] \leftarrow CM_{\sigma}[x,y] + 1$ {Update the confusion matrix}
22: **end for**
23: **end for**
24: **end for**
25: **end for**
26: **end for**
27: **for all** $\sigma \in \Sigma$ **do**
28: $\rho_{\sigma} \leftarrow \left(\frac{\rho_{\sigma}^{time}}{\alpha \times \beta \times 10}, \frac{\rho_{\sigma}^{reduction}}{\alpha \times \beta \times 10}\right)$
29: $\phi_{\sigma} \leftarrow$ statistical_measures(CM_{σ}) {Calculate measures such as accuracy, specificity, ... }
30: **end for**
31: **return** ϕ, ρ

Figure 4.1 portrayed the data flow for the $\alpha\beta$ evaluation. The original case-base M is partitioned into 10 M_i (test sets)and $\overline{M}_i$ (training sets). Later, the CBM method σ is used on the training sets to create α maintained case-bases $\overline{M}_i^{\sigma}$, which are used to compute the Quality

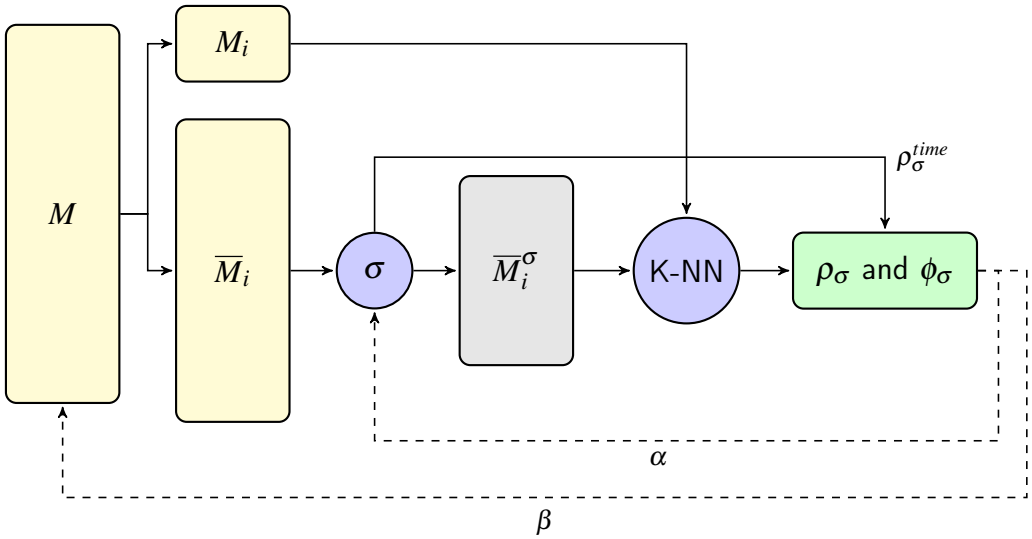


Fig. 4.1 Data flow for the $\alpha\beta$ evaluation.

and Performance Decision Scores ρ_σ and ϕ_σ. The process is repeated up to β times, and finally, the average is done to calculate the Decision Scores.

4.3.2 Values of α and β

Setting the β parameter will configure the evaluation process to repeat the Cross-Validation evaluation β times. With $\beta = 1$ and $\alpha = 1$, this evaluation is a Cross-Validation, whereby the CBM algorithms are applied to each training set. With higher values of β the Cross-Validation is repeated. This is the case of the evaluation methods adopted in [42] and [107], which repeat Cross-Validation 10 and 20 times, respectively.

On the other hand, changing the α parameter will lead to each CBM algorithm being repeated up to α times with the same training set. This parameter is useful when the CBM algorithm is non-deterministic. That is, multiple executions of a CBM with the same training set and order of cases return different case-base outputs.

4.4 Experiments

The following experiments were aimed at deepening our understanding of the $\alpha\beta$ evaluation, and, at showing how both α and β parameters can affect the decision score results. In particular experiments were made using deterministic and non-deterministic CBM algorithms with different values for both parameters. We have chosen CNN as the deterministic CBM algorithm since it is one of the simplest and most widely used CBM algorithms, while the NSGA-II was chosen as the non-deterministic algorithm [40], which uses the fitness function proposed in [97], in order to perform CBM. Such fitness function has three objectives aimed

to be minimised: (1) the number of redundant cases, (2) an estimation of the error, and (3) the difference between the current number of cases in the solution and the estimated number of non-redundant cases, which is computed at the beginning of the maintenance algorithm. When NSGA-II algorithm finishes, it obtains a set of potential solutions that represent different maintained case-bases. So as to choose the output of the algorithm, the solution representing the case-base with highest accuracy is returned, where this accuracy is given by the evaluation process, not by the fitness function.

The organization of the experimental section to obtain the quality decision scores is as follows: first, we study different configurations of α and β separately. Secondly, we compare the best results of $\alpha\beta$ with the accuracy results given by other evaluation methodologies, such as Cross-Validation, Hold-Out, and Pan's proposal (Pan) [126]. This evaluation basically consists of repeating 20 times the Hold-Out evaluation. On the other hand, the performance decision score that we evaluate is the reduction rate of the CBM algorithm. Finally, we study the complexity and runtime of each evaluation method since this will help decide which evaluation method is the best.

4.4.1 Experimental setup

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 $k = 3$. Owing to the fact that multiple solutions can be retrieved, a *majority voting system* is used. That is, CBR system output is the most common solution among the retrieved solution. In case all the solutions are different, the solution with the lowest distance to the input problem is chosen.

First, a case-base pre-processing is performed to decrease the time needed to run each experimentation. This pre-process includes selection of the most relevant attributes, deletion of duplicate cases and replacement of missing values with the mean of the attribute values. Finally, all the features values in each case-base are normalised in the interval $[0,1]$. Attribute selection is made with a genetic algorithm, where each individual is a binary string with element values in the domain $\{true, false\}$. Each individual represents the features which are selected from the problem description. If the element is $true$ then the feature is selected. That is, an individual with n elements represents a case-base with n features describing a problem, and the elements with the value set to $true$ are the selected features of the problem. The population size and the number of generation are set to 100. The one point cross-over probability is 0.9 and the mutation is 0.033. The fitness value of each individual is the accuracy given by a 10 fold Cross-Validation using the equivalent case-base to the individual.

The experiments are made with the following public datasets: *iris, liver-bupa, bridges-version1* and *zoo* from UCI repository [49]. To build the case-base, each instance in a dataset

is transformed into a case, where the problem description takes all the instance attributes except for the class, which is the solution description. All the experiments are performed 10 times in order to provide enough experimental results. Finally, the results are averaged to observe the differences between all the evaluation results.

4.4.2 Adjusting α and β parameters

In this section, the α and β influence on the evaluation results are analysed. Second, a comparative of $\alpha\beta$ against other evaluation methods will be carried out.

In order to analyse how different values of α and β influences the evaluation results, the values of α and β under consideration are $\alpha = \{1,3,5,7\}$, and $\beta = \{1,5,10,15\}$. These values are selected in order to analyse how incremental changes on α and β values affect the results of the evaluation.

Values of α for deterministic and non-deterministic CBM

Figure 4.2 depicts the accuracy results of the CNN algorithm with the four different case-bases. In this experiments the β is constant ($\beta = 1$) and α varies its value ($\alpha = \{1,3,5,7\}$).

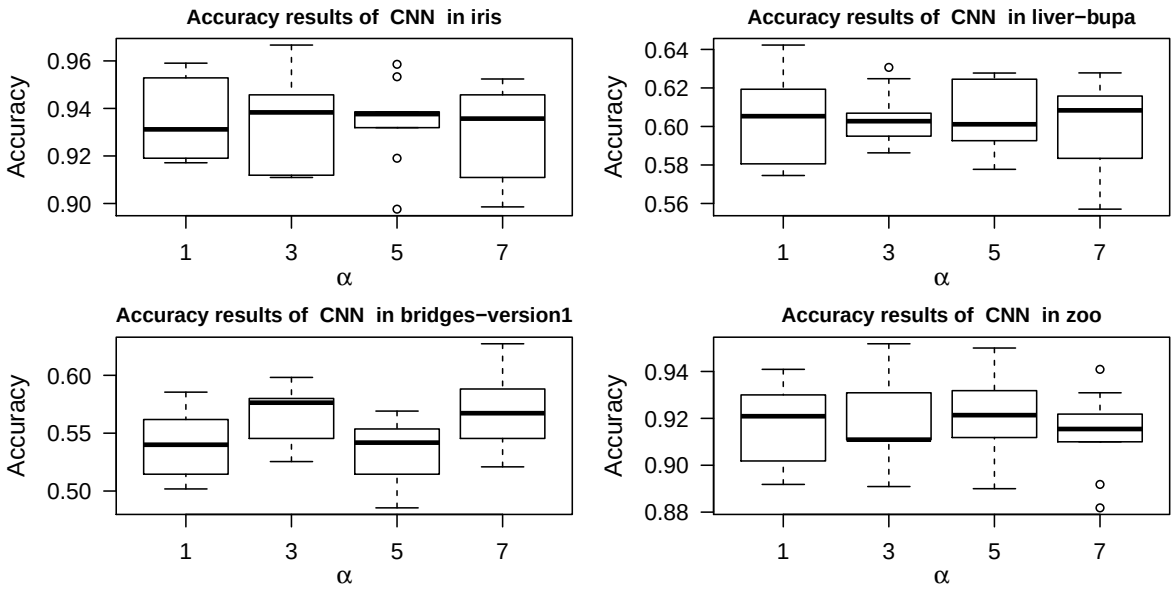


Fig. 4.2 Experiments varying the α parameter for CNN ($\alpha = \{1,3,5,7\}$, and $\beta = 1$).

Figure 4.3 shows the accuracy results for NSGA-II in each case-base. The results obtained with $\alpha = 1$ show a larger variation, while $\alpha \geq 3$ provides more regular results.

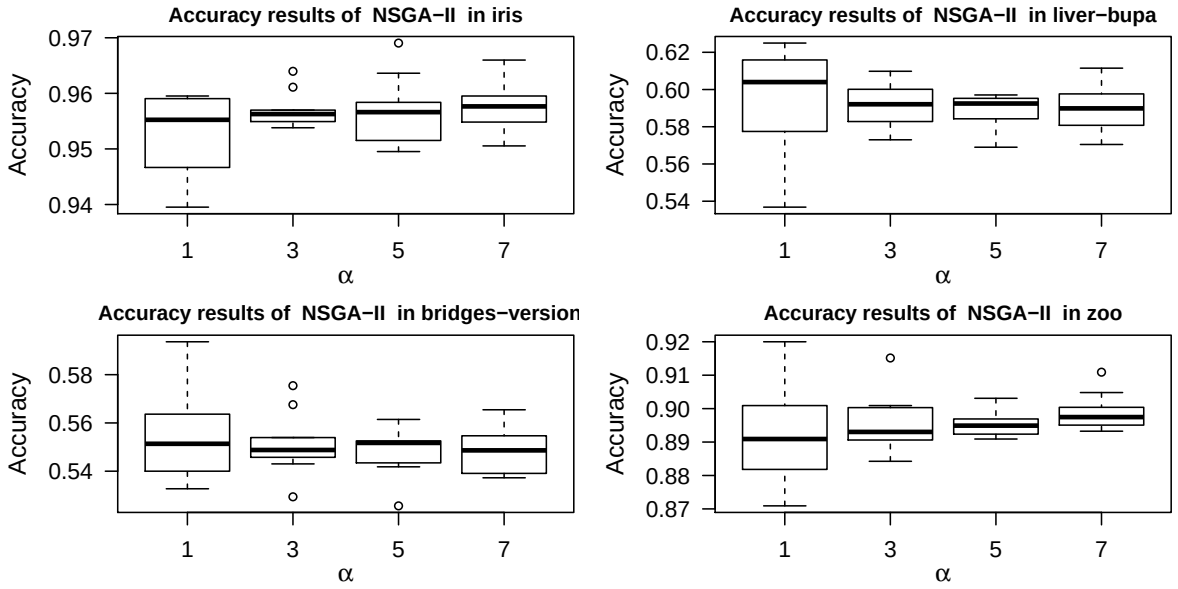


Fig. 4.3 Experiments varying the α parameter for NSGA-II ($\alpha = \{1,3,5,7\}$, and $\beta = 1$).

β values for deterministic and non-deterministic CBM algorithms

Figure 4.4 depicts the results of the recorded accuracies for each case-base with the CNN algorithm. The value of α is equal to 1, and only β varies its value ($\beta = \{1,5,10,15\}$).

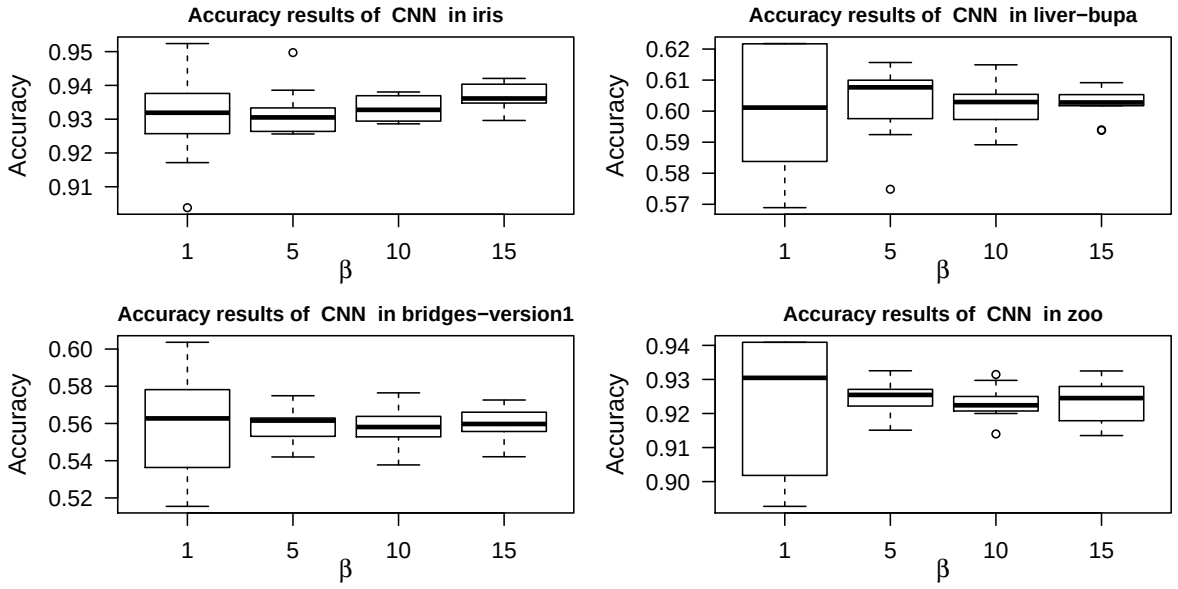


Fig. 4.4 Experiments varying the β parameter for CNN ($\alpha = 1, \beta = \{1,5,10,15\}$).

Figure 4.5 depicts the accuracy results for each case-base using NSGA-II as the applied CBM algorithm. The value of α is 1.

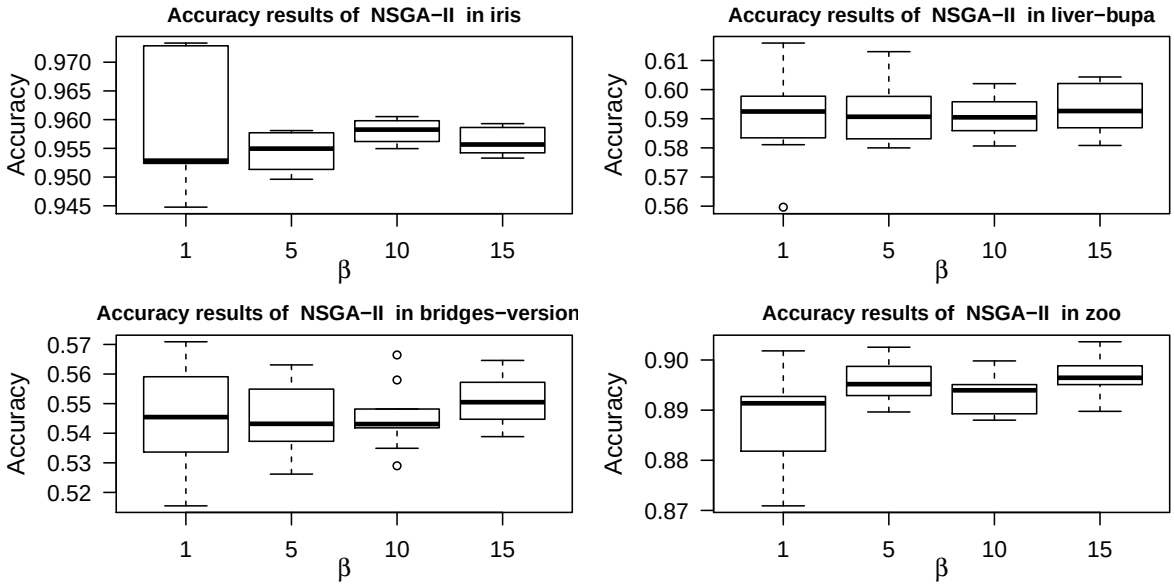


Fig. 4.5 Experiments varying the β parameter for NSGA-II ($\alpha = 1, \beta = \{1, 5, 10, 15\}$).

Comparing α and β parameters for non-Deterministic CBM

The question that remains is whether α and β can be used simultaneously to enhance the reliability of the evaluation results. The following experiment is performed using the $\alpha\beta$ evaluation with the values $\alpha = 3$ and $\beta = 5$, and NSGA-II as the CBM algorithm under observation, using the fitness function given in [97]. The results are compared with the accuracy averages given by the experiments made with $\alpha = 1, \beta = 5$ and $\alpha = 3, \beta = 1$.

Figure 4.6 depicts the accuracy results for the case-bases *iris, liver-bupa, bridges-version1* and *zoo*.

4.4.3 Comparing $\alpha\beta$ with other evaluation methods

To analyse whether $\alpha\beta$ evaluation is more suitable than other evaluation methods, we compare the $\alpha\beta$ evaluation results with those provided by Cross-Validation, Hold-Out and Pan [126]. Note that both Cross-Validation and Hold-Out take as input the maintained case-base returned by the CBM algorithm under study, whereas, both $\alpha\beta$ and Pan evaluations execute the CBM algorithm on each training set.

First, the experiments will focus on how the evaluation affects the results provided by CNN, a deterministic CBM algorithm. The values for α and β in this experiments are set to 1 and 10, respectively. Values chosen because α is only useful with non-deterministic CBM algorithms, and the experiments made in subsection 4.4.2 show that the most suitable value for β is 10 or more.

Secondly, the experiments will deal with a non-deterministic algorithm (NSGA-II) using

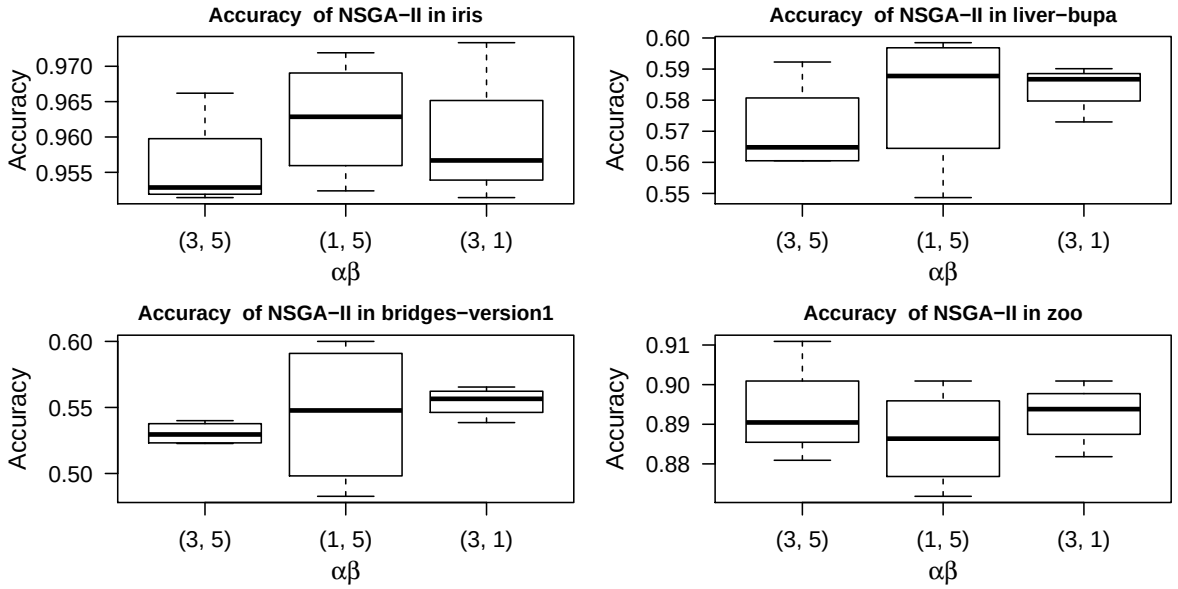


Fig. 4.6 Results of NSGA-II for three different configurations of $\alpha\beta$ evaluation: $(\alpha = 3, \beta = 5)$, $(\alpha = 1, \beta = 5)$, $(\alpha = 3, \beta = 1)$.

the fitness function to perform CBM proposed in [97]. In this occasion, the parameters values of α and β are set to 3 and 5, respectively, values that have shown good results, as figure 4.6 depicts.

Additionally, each case-base is first evaluated without any case selection. For the sake of clarity, we call *None* the CBM algorithm that does not perform any case selection.

Figure 4.7 portrays the accuracy results for each evaluation method and with the None algorithm, and figure 4.8 depicts the results of the CNN algorithm. Finally, figure 4.9 shows the results with the algorithm NSGA-II. In order to identify differences between the evaluation methods, the *Analysis of Variance* (ANOVA) is carried out. Tables 4.2, 4.4 and 4.6 show the ANOVA results for the accuracy results given by each evaluation with None, CNN and NSGA-II, respectively. With this test, the hypothesis is that the accuracy averages are equal between the results given by each evaluation method. We pay special attention to the *F value* and *p-value*, where the *F value* is the difference between the accuracy results of each evaluation method, and *p-value* is the probability for the accuracy averages to be the same. In other words, lower the *p-value*, higher the evidence that the accuracy averages are different. Table 4.1 depicts the codes used by the ANOVA tables.

We also focus on the comparison between every pair of evaluation method by carrying out the Tukey Honest Significant Differences test (Tukey HSD)[175]. Tables 4.3, 4.5 and 4.7 show the *p-value* for all the studied case-bases, from the results of two evaluation methods with Tukey HSD. Within those tables, low values of *p-value* mean low probability that the two evaluation methods have the same results variance.

Lastly, other important issue is the difference between the accuracy results given by the dif-

Label	Name	Meaning
Df	Degrees of freedom	It is related to the number of experiments done
Sum Sq	Sum of squares	The difference between the accuracy averages of each evaluation method
Mean Sq	Mean Squares	The dispersion of the results
F value	$\frac{\text{Mean square Between}}{\text{Mean square Within}}$	A large value indicates relatively more difference between the accuracy results of each evaluation method than the accuracy within groups
Pr(>F)	*p-value*	Probability that the null hypothesis is right. Lower the *p-value*, higher the evidence against the null hypothesis.

Table 4.1 Codes used in ANOVA tables 4.2, 4.4 and 4.6

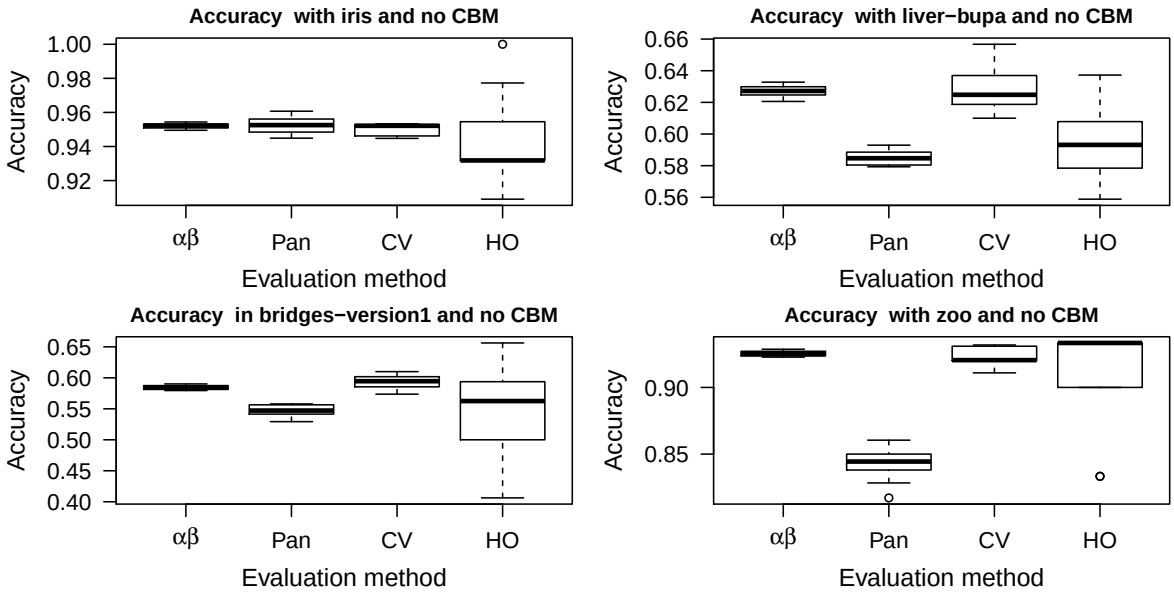


Fig. 4.7 Comparative study of the evaluation methods $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO), when None CBM algorithm is executed.

ferent evaluation processes when either they do not use a CBM algorithm or some deterministic/non-deterministic CBM algorithm. In order to study these differences, an ANOVA analysis is performed between the accuracy results given with no CBM, CNN and NSGA-II, respectively, for each evaluation process. Therefore, table 4.8 shows the ANOVA results after comparing the accuracy variances given by the $\alpha\beta$, Pan, CV and HO evaluations results.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	0.00	0.00	1.34	0.2775
	Within	36	0.01	0.00		
liver-bupa	Between	3	0.01	0.00	23.68	$< 10^{-5}$
	Within	36	0.01	0.00		
bridges-version1	Between	3	0.02	0.01	4.25	0.0114
	Within	36	0.05	0.00		
zoo	Between	3	0.04	0.01	31.73	$< 10^{-5}$
	Within	36	0.02	0.00		

Table 4.2 ANOVA for the accuracy results given by each evaluation with none CBM.

	iris	liver-bupa	bridges-version1	zoo
	p-value	*p-value*	*p-value*	*p-value*
CV-$\alpha\beta$	0.99	1.00	0.94	0.98
HO-$\alpha\beta$	0.34	$< 10^{-5}$	0.17	0.24
Pan-$\alpha\beta$	1.00	$< 10^{-5}$	0.12	$< 10^{-5}$
HO-CV	0.48	$< 10^{-5}$	0.05	0.41
Pan-CV	0.99	$< 10^{-5}$	0.03	$< 10^{-5}$
Pan-HO	0.33	0.25	1.00	$< 10^{-5}$

Table 4.3 Tukey HSD for the accuracy results given by each evaluation with none CBM algorithm.

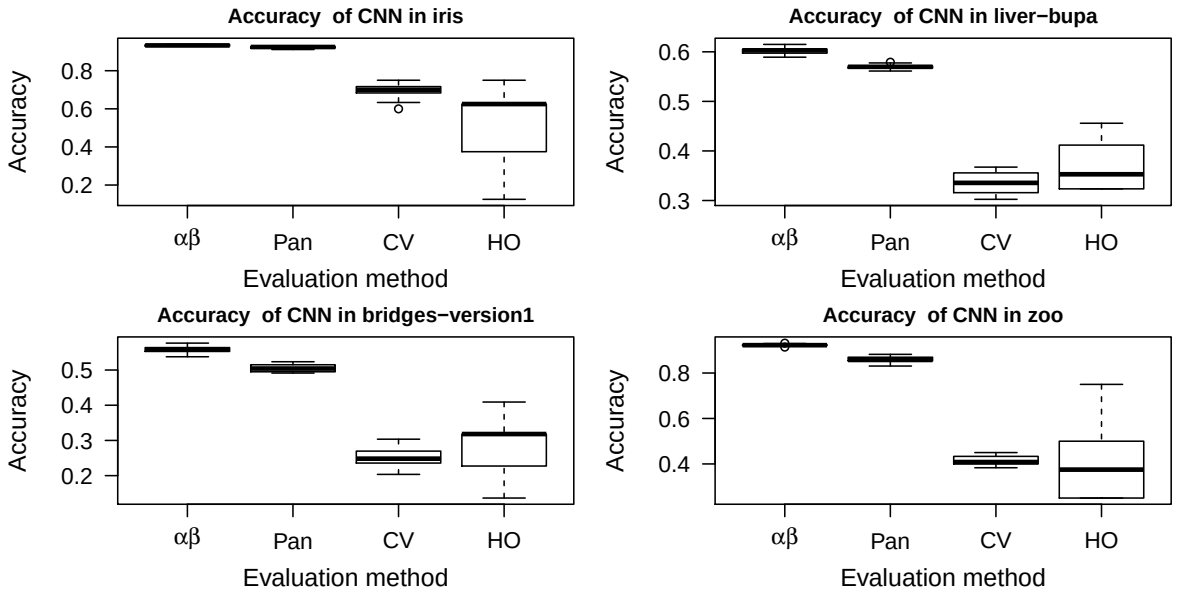


Fig. 4.8 Comparative study of the evaluation methods: $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO) evaluations, when CNN algorithm is executed.

4.4.4 Experiments of performance decision scores

Despite the importance of accuracy, the improvement achieved by a CBM algorithm in terms of performance is also fundamental. In this subsection we focus on the reduction rates and the execution time of each CBM algorithm.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	1.10	0.37	36.20	$< 10^{-5}$
	Within	36	0.36	0.01		
liver-bupa	Between	3	0.56	0.19	248.84	$< 10^{-5}$
	Within	36	0.03	0.00		
bridges-version1	Between	3	0.68	0.23	114.26	$< 10^{-5}$
	Within	36	0.07	0.00		
zoo	Between	3	2.35	0.78	112.43	$< 10^{-5}$
	Within	36	0.25	0.01		

Table 4.4 ANOVA for the accuracy results given by each evaluation with CNN.

	iris	liver-bupa	bridges-version1	zoo
	p-value	p-value	p-value	p-value
CV-$\alpha\beta$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$
HO-$\alpha\beta$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$
Pan-$\alpha\beta$	1.00	0.07	0.06	0.33
HO-CV	0.01	0.06	0.16	0.97
Pan-CV	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$
Pan-HO	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$

Table 4.5 Tukey HSD for the accuracy results given by each evaluation with CNN.

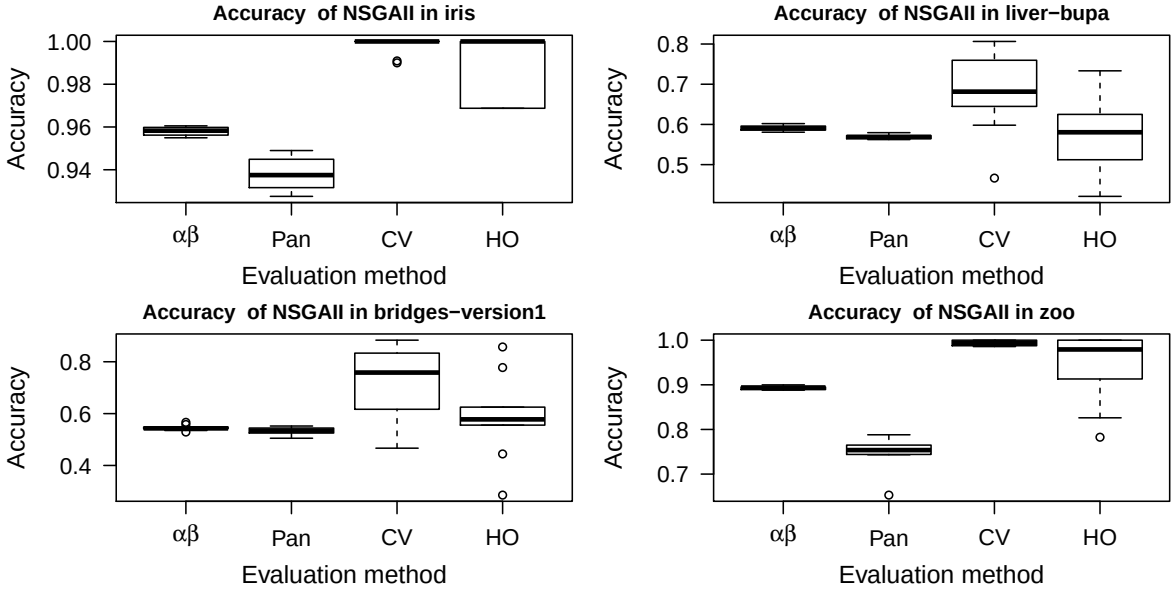


Fig. 4.9 Comparative study of the evaluation methods: $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO), when NSGA-II algorithm is executed.

Reduction rates are acquired from $\alpha\beta$ and Pan evaluation to observe whether a particular evaluation method may have any impact on this measure. Figures 4.10 and 4.11 depict the results for $\alpha\beta$ and Pan evaluation respectively. Each column is the average of all the reduction rates given by all the experiment performed using the corresponding CBM algorithm.

Prior to performing any empirical experimentation to record the runtime of each evaluation

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	0.02	0.01	103.81	$< 10^{-5}$
	Within	36	0.00	0.00		
liver-bupa	Between	3	0.08	0.03	5.65	0.0028
	Within	36	0.18	0.00		
bridges-version1	Between	3	0.24	0.08	7.36	0.0006
	Within	36	0.39	0.01		
zoo	Between	3	0.33	0.11	57.31	$< 10^{-5}$
	Within	36	0.07	0.00		

Table 4.6 ANOVA for the accuracy results given by each evaluation with NSGA-II.

	iris	liver-bupa	bridges-version1	zoo
	p-value	p-value	p-value	p-value
CV-$\alpha\beta$	$< 10^{-5}$	0.03	$< 10^{-5}$	$< 10^{-5}$
HO-$\alpha\beta$	$< 10^{-5}$	0.96	0.79	0.07
Pan-$\alpha\beta$	$< 10^{-5}$	0.90	0.99	$< 10^{-5}$
HO-CV	0.25	0.01	0.02	0.07
Pan-CV	$< 10^{-5}$	0.01	$< 10^{-5}$	$< 10^{-5}$
Pan-HO	$< 10^{-5}$	1.00	0.63	$< 10^{-5}$

Table 4.7 Tukey HSD for the accuracy results given by each evaluation with NSGA-II.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
$\alpha\beta$	Between	2	0.01	0.01	0.22	0.8041
	Within	117	3.66	0.03		
Pan	Between	2	0.02	0.01	0.39	0.6758
	Within	117	3.51	0.03		
CV	Between	2	4.12	2.06	72.65	$< 10^{-5}$
	Within	117	3.31	0.03		
HO	Between	2	3.50	1.75	49.11	$< 10^{-5}$
	Within	117	4.16	0.04		

Table 4.8 ANOVA between the accuracy results given by $\alpha\beta$, Pan, CV and HO evaluations with None, CNN and NSGA-II.

method, we define the computational complexity of each one, to ascertain which will take longer execution time. Formally, let n and m be the respective number of iterations needed to perform the CBM algorithm in the training sets of Cross-Validation and Hold-Out, with $n \geq m$. Let f be the number of folds in a Cross-Validation. The complexity orders of each evaluation are the following:

- $\alpha\beta$ evaluation: $\mathcal{O}(fn\alpha\beta)$.
- Pan $\mathcal{O}(20m)$.
- Cross-Validation: $\mathcal{O}(10n)$.
- Hold-Out $\mathcal{O}(m)$.

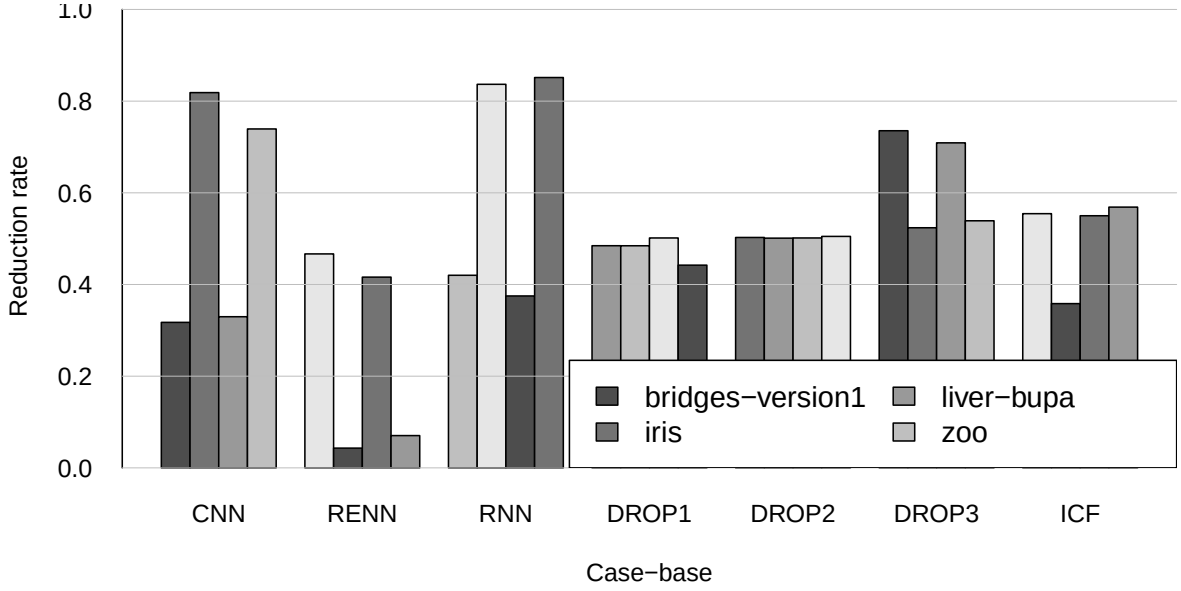


Fig. 4.10 Reduction rate for $\alpha\beta$ evaluation method.

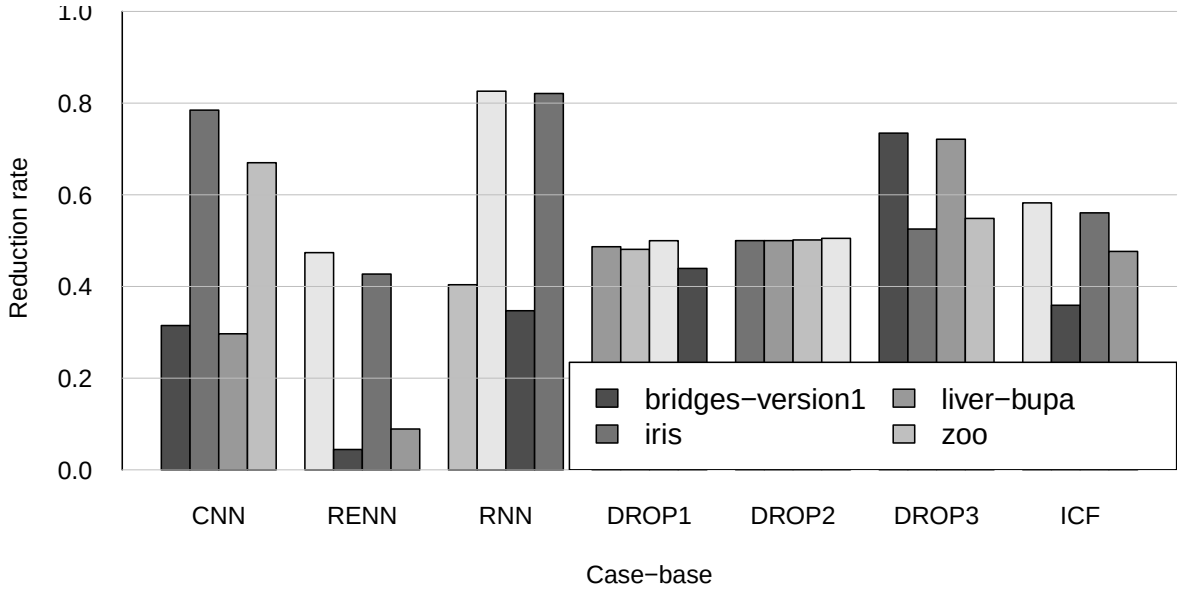


Fig. 4.11 Reduction rate for Pan evaluation method.

According to the complexity, with $f = 10$ the $\alpha\beta$ will take $\alpha\beta$ more time to finish the evaluation than Cross-Validation. Pan evaluation will take 20 times more than Hold-Out. Because $n \geq m$, the evaluations based on Cross-Validation have longer runtime than those based in Hold-Out. Figure 4.12 depicts the average runtime results for the experiment performed previously, with $\beta = 10$, $\alpha = 1$ and $f = 10$. As we expected, $\alpha\beta$ has the longest runtime, followed by Pan evaluation.

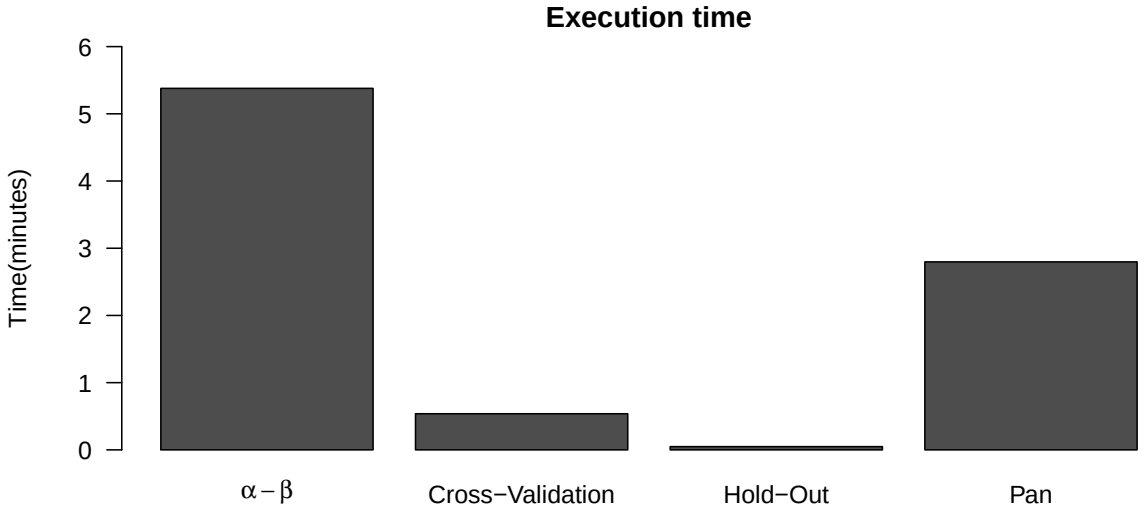


Fig. 4.12 Execution time of the different evaluation methods.

4.5 Discussion

As regards to the study of the α and β parameters, the following aspects should be mentioned:

1. According to figures 4.4 and 4.5, high values of β provide accurate results with low variance. These evidence are related to the study of parameter β for deterministic and non-deterministic CBM algorithms. Whereas in all the studied case-bases values of $\beta = 1$ show higher variance of accuracy results than those results with $\beta = 5, 10$ and 15, values of $\beta = 5$ and $\beta = 10$ seem to converge to similar accuracy averages. Therefore, the $\beta = 5$ and $\beta = 10$ seem to be enough to perform the evaluation for deterministic and non-deterministic algorithms.

2. Whereas the α parameter has no effect on the results when the CBM algorithm is deterministic, higher values of α may provide reliable accuracy results for non-deterministic CBM algorithms. According to figure 4.3 , values of α equal to 3 or 5 seem to be sufficient to obtain reliable accuracy results.

3. When using α and β parameters simultaneously with values greater than one, the method shows reliable accuracy results. However, the variance in the results is only slightly higher than when α parameter alone is used.

After the comparative study of evaluation methods ($\alpha\beta$, Hold-Out, Cross-Validation and Pan evaluations), the following should be pointed out:

1. Whereas methods based on folds, such as $\alpha\beta$ and Cross-Validation, achieve similar accuracy averages for every case-base with very little variance, those based on Hold-

Out provide uneven accuracy results with greater variation. This is probably due to the low variety in the training and test sets.

2. Hold-Out shows the greatest variance in the accuracy results.

3. When a deterministic CBM algorithm is being evaluated, neither Cross-Validation and nor Hold-Out provide a good estimation of the accuracy of the CBR system. Furthermore, $\alpha\beta$ and Pan evaluations have similar results, with low variance, although Pan evaluation underestimates the accuracy in every case-base.

4. For non-deterministic CBM algorithm, the Pan, Cross-Validation and Hold-Out evaluations do not compute reliable accuracy results in some case-bases. The variation in the results confirms this point. On the other hand, $\alpha\beta$ provides accuracy results with the lowest variation. Furthermore, the results are the closest to the original accuracy results.

In our experiments, we compare the evaluation methods according to the accuracy average using ANOVA (tables 4.2, 4.4 and 4.6). We consider the standard interpretation of the p-value < 0.05 to consider that there are significant differences between the experiment results. According to this:

5. Regarding the ANOVA test on the original case-bases, there is a significant difference between the accuracy averages provided by the different CBM algorithms for *liver-bupa*, *bridges-version1* and *zoo* case-bases. In particular, according to the TukeyHSD results in table 4.3, there is no significant difference between the accuracy average results obtained by $\alpha\beta$ and Cross-Validation, and similarly, Pan and Hold-Out evaluation methods obtain similar accuracy averages. Nonetheless, $\alpha\beta$ results are significantly different from the results of Hold-Out in the case-base *liver-bupa*, and with Pan with *liver-bupa* and *zoo*. Lastly, Pan and Cross-Validation results are significantly different in *liver-bupa*, *bridges-version1* and *zoo* case-bases. When compared the results of Pan and Hold-Out, there is only significant differences for the case-base *zoo*. Therefore, when none CBM algorithm is used all the evaluation methods are suitable to measure the accuracy of a CBR system, although Pan has different accuracy measures from the rest.

6. As can be seen in table 4.4, when deterministic CBM algorithm is studied, all the accuracy averages are significantly different. In particular, TukeyHSD results (see table 4.5) show that $\alpha\beta$ and Pan accuracy averages are significantly different to both Cross-Validation and Hold-Out in all the case-bases. What is more, $\alpha\beta$ and Pan are not significantly different from their accuracy averages, although for the case-bases *liver-bupa* and *bridges-version1* are close to being significantly different. On the one hand, Cross-Validation and Hold-Out are measuring differently the accuracy of the CBR system after

the maintenance by a deterministic algorithm with regard to their results from the original case-base. This indicates they are not suitable to measure the effects on the accuracy when a deterministic CBM algorithm is used. On the other hand, $\alpha\beta$ and Pan accuracy results are similar to the original. Both evaluation methods have similar accuracy averages with case-bases with high accuracy averages, such as *iris* and *zoo*. With regard to case-bases with low accuracy averages, such as *liver-bupa* and *bridges-version1*, their accuracy averages are quite similar.

7. When non-deterministic CBM Algorithms are analysed (see Table 4.6), $\alpha\beta$ evaluation is significantly different from Cross-Validation, in contrast with the original case-base, where both evaluation methods have similar accuracy averages. Besides, TukeyHSD results (see table 4.7) show that $\alpha\beta$ is significantly different from Pan and Hold-Out accuracy averages in *iris* and *zoo* case-bases. Whereas $\alpha\beta$ and Pan have similar accuracy averages in case-bases with low accuracy such as *liver-bupa* and *bridges-version1*, their accuracy averages are different from case-bases with high accuracy. Finally, Hold-Out and Cross-Validation only have similar accuracy averages for the case-base *iris*, and both have different accuracy averages from the original case-bases. Therefore, they are not suitable to evaluate the effects on the accuracy by a non-deterministic algorithm.

8. Similarity accuracy averages obtained by the different evaluation methods for all the considered case-bases and CBM algorithms are shown in table 4.8. With $\alpha\beta$ and Pan there is no statistical difference between the original (none algorithm), deterministic and non-deterministic accuracy results, meaning that both evaluation methods are good options to evaluate the effects of the CBM algorithms in the CBR system. In particular, $\alpha\beta$ provide the highest similarities between the accuracy averages of the CBM algorithms and these obtained in the original case-base. On the contrary, Cross-Validation and Hold-Out accuracy averages are significantly different. Hence, their results are not reliable when they are used to evaluate the effect of CBM algorithms on the CBR system, either deterministic or non-deterministic.

A glance at figures 4.10 and 4.11 shows that every reduction for every case-base is quite similar. The Pearson correlation between the data from both figures is shown in figure 4.13, with correlation values close to one.

In light of to the results of Cross-Validation and Hold-Out, one execution is not sufficient to obtain reliable measures, whether for deterministic or non-deterministic CBM algorithms.

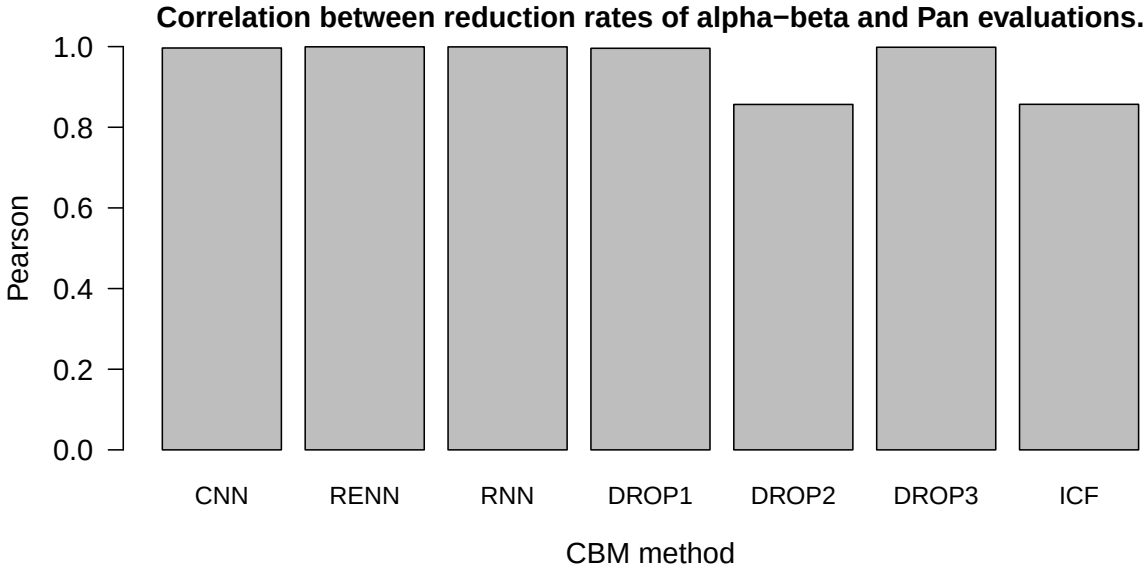


Fig. 4.13 Pearson correlation between $\alpha\beta$ and Pan evaluations.

4.6 Conclusions

We introduce the $\alpha\beta$ method to evaluate different CBM algorithms. The evaluation depends on two parameters: α to obtain reliable results when the CBM algorithm under observation is non-deterministic, and β to repeat the entire evaluation β times and obtain an average of the results. We have released a Java library that implements many CBM algorithms. Finally, exhaustive experiments have been performed in order to compare different evaluation methods.

Using public datasets, we have compared our results with those obtained using other evaluation methods in the CBR literature: Cross-Validation, Hold-Out and Pan evaluation [126]. On the one hand, the results obtained show that only one execution of Cross-Validation and Hold-Out is not sufficient to provide reliable results, whether for deterministic or non-deterministic CBM algorithms. On the other hand, according to our experiments, $\alpha\beta$ and Pan evaluation are suitable methods to evaluate deterministic CBM algorithms. Although Pan evaluation needs a lower runtime, $\alpha\beta$ is the only evaluation method able to provide reliable evaluation measurements when dealing with non-deterministic CBM algorithms. In such a scenario, the $\alpha\beta$ evaluation shows itself to be a good option for use as an evaluation method.

In certain domains, such as Intensive Care Unit decision support or personal activity monitoring at home, the concept of time is a relevant feature to perform reasoning operations [27, 73]. For this reason, in future studies, the $\alpha\beta$ evaluation, as well as the CBM algorithms, will be extended to domains where the case-base contains temporal cases. Additionally, because the $\alpha\beta$ evaluation generates the training and test sets randomly, and there are many CBM algorithms that are order sensitive, we will extend the $\alpha\beta$ evaluation to multiple execu-

tions of the CBM algorithm with different training set orders. Finally, emphasis will be placed on improving the efficiency of the $\alpha\beta$ evaluation, in an attempt to reduce the time consumed during its execution.

The ANOVA test carried out to compare accuracy averages provided by CBM methods suggests that the evaluation methods analysed measure differently the accuracy of the CBR system. Experiment results show that Cross-Validation and Hold-Out are less suitable to evaluate the effects on accuracy of CBM algorithms. However, $\alpha\beta$ and Pan show consistent results when maintained case-bases are compared with their original case-bases. Furthermore, accuracy averages provided by $\alpha\beta$ on maintained case-bases are more similar to those provided by original case-bases than in the case of Pan. In the case of non-deterministic CBM algorithms, $\alpha\beta$ provides a lower accuracy variance than in the case of Pan. Therefore, this suggests that $\alpha\beta$ provides a more robust evaluation method when CBM algorithms are applied.

Chapter 5

Temporal Case-Base Maintenance

Time plays a key role in describing stories and cases. In the last decades, great efforts have been done to develop CBR systems that can cope with the temporal dimension. Despite this kind of system finds it difficult to maintain their case-base, little attention has been paid to maintain temporal case-bases. Case-Base Maintenance (CBM) algorithms are useful tools to build efficient and reliable CBR systems. In this chapter, we propose the extension of different CBM approaches to deal with this problem. Five temporal CBM algorithms are proposed: t-CNN, t-RENN, t-DROP1, t-ICF and t-RC-FP. These algorithms make the maintenance of case-bases possible when cases are temporal event sequences.

5.1 Introduction

Traditionally, CBR systems have been designed to deal with static domains, which capture the state of the system environment and never change over the time. However, in order to capture how the system evolves through time there are some domains in which the temporal dimension is relevant for the description of system states. Example of these problems could be the evolution of the state of a variable as the time goes by, a process planning or the monitoring an activity, and even there is possible to find that the problem description of a case depends on another case problem but in a different time.

In this way, in order to represent the time there is commonly three primitives: points or instants, intervals between two points or a mix of both [160], and among the most used time structures to represent the temporal dimension we can find times series [76, 115, 118, 122, 128], episodes [33, 56, 177], workflows [14, 21, 27, 43, 78, 112, 165] and event sequences [59, 60, 68, 73, 159]. In particular, the present chapter is focused in a CBR system using event sequences and instants for representing temporal cases. The main reason to choose this data structure is because they can be used to represent most of the rest of the structures, such as the time series and episodes. The second reason is due to its suitability to the proposed CBR system to monitoring elderly people at home, which is introduced in the next chapter.

CBR systems using temporal cases may suffer the same performance problems as classical CBR as the size of the case-base increases. Therefore CBM algorithms may be used to reduce the size of the case-base and to increase the performance, as weel as to keep the problem solving correctness of the system at the same time. For the sake of clarification, given a CBR system working with cases storing event sequences, figure 5.1 depicts the queries per second that a CBR system is able to deliver according to the case-base size. The data has been gathered from a CBR system using cases with event sequences with different number of events. As can be seen, higher number of cases implies lower number of queries per second, even for event sequences with few events. Indeed the performance could be improved with better hardware and data structures. Notwithstanding, these two approaches may not solve completely the

problem because the performance problem may appear again with larger case-bases.

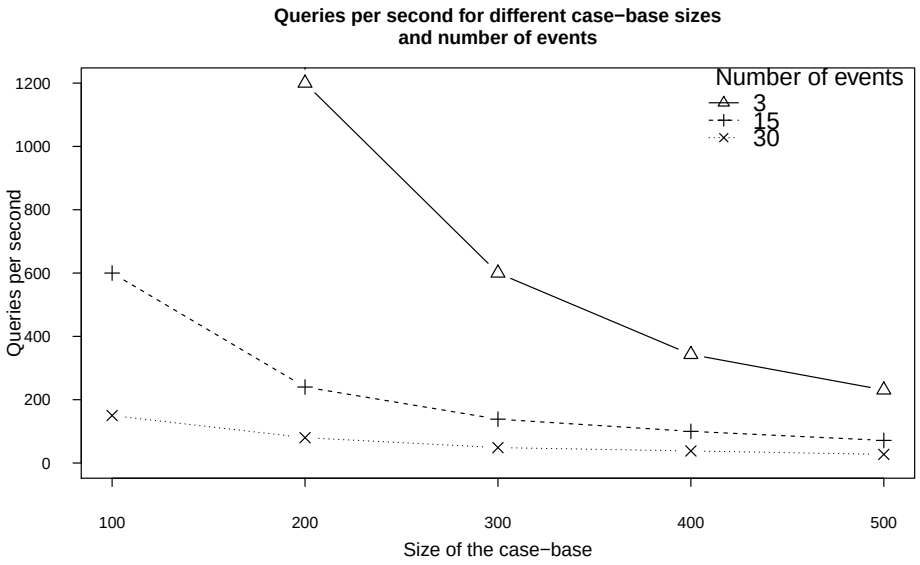


Fig. 5.1 Performance (queries per second) of a CBR system working with different temporal case-base sizes.

Therefore, the CBM is an approach that prevents performance problems caused by a large case-bases. However, up to our knowledge there is no study focused in the effect of the maintenance in temporal case-bases. Therefore, in the present chapter, the use of CBM algorithms in temporal case-bases is analysed, in order to enlighten whether the maintenance task may be performed successfully im temporal case-bases.

The remainder of this chapter is the following: In section 5.2 we describe the event sequences as a temporal representation. In section 5.3 we propose a case representation for using event sequences. In section 5.4 we introduce how to quantify the similarity between temporal cases. Later, in section 5.5 we explain our proposals for temporal CBM. In section 5.6 several experiments results are shown in order to determine the factors that may affect the quality of the maintained case-base created by a CBM algorithm. Finally, we show in sections 5.7 and 5.8 the discussion and conclusions, respectively.

5.2 Event sequences

Intelligent systems are only able to give solutions to problems in a particular domain or environment [48, 153]. In fact, the environment is represented by one variable for each existing features describing the problem domain to solve.

Definition 5.1. (Variable) A *variable* V_i represents a feature that can be observed and measured, and it is formally defined by the set of all possible values that can be observed from the feature.

The variable values are described as quantitative or qualitative values. For instance, a variable representing real numbers may be $V_{real} \subseteq \mathbb{R}$, or a set of symbolic values as $V_i = \{symbol_1, symbol_2, \ldots\}$. In general, any V_i can be defined as a set of values for describing a feature.

However, more complex representations of variables are also possible [123], such as those related with temporal dependent features. In particular, the temporal dependent values could be described through the concepts of event type, event and event sequence.

Accordingly to [114, 162], the possible values that can be found in a temporal dependent variable are called *event type*.

Definition 5.2. (Event Type) An *event type* is set of the possible values that temporal dependent variables may take. The set of all the possible event types is denoted by E.

Example 1. Let $V_1 = \{A, B, C\}$ be the temporal dependent variable, which are the values that the variable V_1 may have as time goes by. From this variable, defining the set E is possible:

$$E = \{e | e \in V_1\} = \{A, B, C\}.$$

Furthermore, the given definition of event type is expressively enough to define more complex representation. For instance, given the time-dependant variables V_1 and $V_2 = \{D, E\}$, the set of event types would be:

$$E = \{e | e \in V_1 \times V_2\} = \{(A,D), (A,E), (B,D), (B,E), (C,D), (C,E)\}.$$

Definition 5.3. (Event) Given a set of all the possible event types E, an *event* is a pair (e,t) of an event type and a time-stamp that represents the instant (time point) at which the event type occurs, formally $e \in E$, and $t \in \mathbb{N}^0$.

Example 2. Let us suppose a temporal dependent variable $V = \{A, B, C\}$, and the event types $E = \{e \in V\}$. The distribution of event types given by figure 5.2:

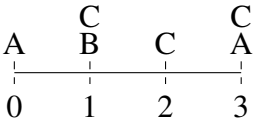


Fig. 5.2 An example of distribution of event types A, B, C through a time windows, where the numbers 0 and 3 are the minimum and maximum time-stamp, respectively.

Figure 5.2 represents the different type events occurs, thus the event type A occurs in the time-stamps 0 and 3, B occurs in the time-stamp 1, and C in the time-stmaps 1,2 and 3. Thus,

the different events that can be defined from this event type distribution are the following:

$$\text{At time } 0 : (A,0) \text{At time } 1 : (B,1)$$
$$\text{At time } 2 : (C,1) \text{At time } 3 : (C,2)$$
$$\text{At time } 4 : (A,3) \text{At time } 5 : (C,3)$$

An event sequence represents different observations of the temporal dependent variables through the time.

Definition 5.4. (Event Sequence) Given a set of event types E, an *event sequence* s_l is a total ordered set of events, and is represented as:

$$s_l = \left\langle \left(e_1^l, t_1^l\right), \ldots, \left(e_n^l, t_n^l\right) \right\rangle,$$

where e_1^l and t_i^l are the i-th event type and time-stamp of the sequence s_l. Furthermore, it is always true that $t_i \leq t_{i+1}$ for each pair of event types e_i and e_{i+1}.

Finally, the set of all event sequences of a given set of event types is defined as follows:

Definition 5.5. (Event Sequences Set) Given a set of event types E, the *event sequences set* S is the set of all the possible event sequences that can be defined from E. Formally:

$$S = \left\{ s^l = \left\langle \left(e_1^l, t_1^l\right), \ldots, \left(e_n^l, t_n^l\right) \right\rangle \middle| \forall i \in [1,n]\ e_i^l \in E, t_i^l \in \mathbb{N}^0, t_i^l \leq t_{i+1}^l \right\}.$$

Example 3. This example is intended to illustrate how the different introduced definitions can be used. Let the scenario be with a temporal dependent variable $V = \{A,B,C\}$, the set of event types $E = \{e \in V\}$, and the distribution of event types from example 2. Therefore, according to the definition 5.4, an event sequence for this distribution is the following:

$$s = \langle (A,0), (B,1), (C,1), (C,2), (A,3), (C,3) \rangle$$

Other definition for the set of event types E could be defined. For instance, being $\mathbb{B} = \{T,F\}$ the set of boolean values, let us suppose the following set $E = \mathbb{B}_A \times \mathbb{B}_B \times \mathbb{B}_C$, where each set $\mathbb{B}_A, \mathbb{B}_B, \mathbb{B}_C$ represents whether the values A,B or C of the temporal variable V occurs for a given time-stamp, respectively. In such case, the event sequence is the following:

$$s = \langle (T,F,F,0), (F,T,T,1), (F,F,F,2), (T,F,T,3) \rangle$$

5.3 Case representation of temporal problem domains

In CBR terminology, a case denotes previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems [1]. Thus, a case may be represented with a problem and its solution descriptions. Whereas the problem description is comprised of non-temporal variables, which define and represent the static features of the problem, temporal variables are related to temporal dependant features of the problem. For instance, let us suppose the scenario where a patient is being monitored: on the one hand, the observation of the person age is enough to classify a person as a child or adult. On the other hand, it is unlikely to predict the future patients' temperature from their current temperature observation, being necessary to make constantly new observations of the temperature values through the time in order to predict the temperature tendency.

In essence, the problem may have a static part and another time-dependant part [118], which may require temporal representations.

- Non-temporal component: with variables representing the static part of the problem. Example of this type of variables may be the name of a person, its age, height, etc.

- Temporal component: with a representation of time-dependant variables. For instance, the patient's temperature and oxygen saturation level during the last hour.

In the following, the main concepts related with temporal CBR systems are defined, such as temporal case, temporal problem, solution and temporal case-base.

Definition 5.6. (Temporal Problem Domain) Given a set of event sequences S, a set of non temporal dependent variables $V_1, \ldots, V_n$, a *temporal problem domain* Π is defined as:

$$\Pi = V_1 \times \ldots \times V_n \times S$$
$$\pi = (v_1, \ldots, v_n, s)$$

Definition 5.7. (Solution Set) The *solution set* Ω is the set of all the existing solutions to a given temporal problem domain Π.

$$\Omega = \{\omega_i \mid \exists \pi_i \in \Pi \ : \omega_i \text{ solves the problem } \pi_i\}$$

Definition 5.8. (Temporal Case) Given a temporal problem domain Π, and a set of solutions Ω, a *temporal case* is defined formally as follows:

$$c = (\pi, \omega),$$

where $\pi = (v_1, \ldots, v_n, s) \in \Pi$, and $\omega \in \Omega$. In order to ease the understanding, Figure 5.3 shows a graphical representation of a temporal case.

Definition 5.9. (Temporal Case-Base) Given a set of event sequences S, a set of static variables $V_1, \ldots, V_n$, and a set of solutions Ω, a *Temporal Case-Base* (TC) is a set of temporal cases:

$$TC \in \wp(V_1 \times \ldots V_n \times S \times \Omega)$$
$$c = (v_1, \ldots, v_n, s, \omega) \in TC$$

Note that TC could be the empty set, representing a temporal case-base with cases. For the sake of simplicity, hereafter, the terms case and temporal cases are used interchangeably.

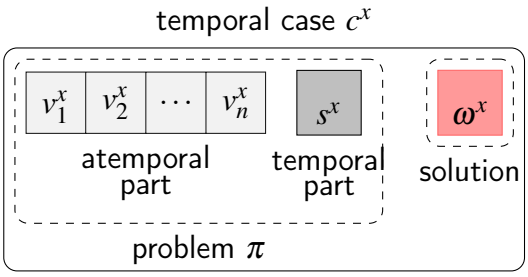
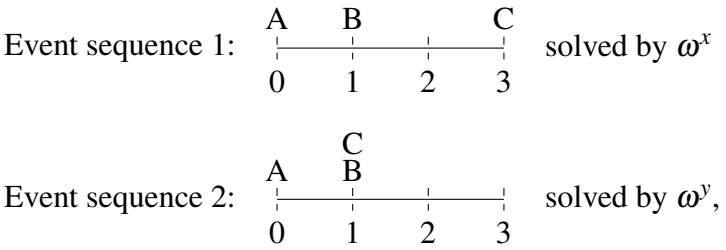


Fig. 5.3 Temporal case c. Regarding the problem description, the v_1 to v_n values are values of the non-temporal variables V_1 to V_n, respectively, and s^x is an event sequence. The problem solution is represented with ω^x.

Example 4. Similarly to the example 3, let be the following two event sequences:



where the event sequences 1 and 2 are formally defined as:

$$s^1 = \langle (A,0), (B,1), (C,3) \rangle$$
$$s^2 = \langle (A,0), (B,1), (C,1) \rangle .$$

Given the sequences s^1 and s^2, and the two possible solutions ω^1 and ω^2, a temporal case-base can be defined as: Finally, the following cases are defined, as well as the temporal

case-base TC:

$$c^1 = \left(\langle (A,0),(B,1),(C,3)\rangle, \omega^1\right)$$
$$c^2 = \left(\langle (A,0),(B,1),(C,1)\rangle, \omega^2\right)$$
$$TC = \left\{c^1, c^2\right\}.$$

5.4 Similarity between temporal cases

In this section, we formalise the concepts and of local distance, global distance and similarity related to the temporal case framework defined in section 5.3. Additionally, when dealing with temporal problem types, a *temporal distance function* δ_{temp} is required as part of the global function as well. In fact, the global distance function is a composition of all the existing δ_{local} and δ_{temp} functions in order to measure the distance between two cases. Whereas δ_{local} functions calculate the distance between two non-temporal variables, δ_{temp} functions ranges the distance between two event sequences from the temporal case. For the sake of clarity, figure 2.3 depicts graphically those distance functions. Given two temporal cases $c^x = (\pi^x, \omega^x)$ and $c^y = (\pi^y, \omega^y)$, there are one local functions δ_{local} for every variable, and a temporal local distance function δ_{temp} to measure the distance between the event sequences of both cases. In this sense, figure 5.4 shows an update of figure 2.3 by including δ_{temp} distance function to the similarity computation between two cases.

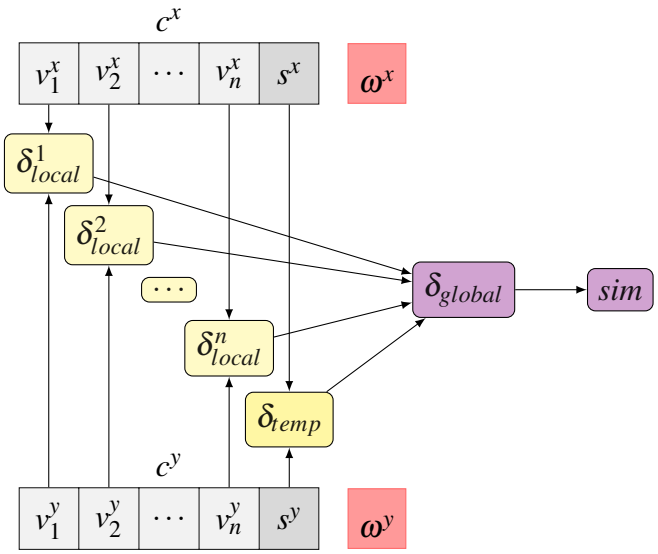


Fig. 5.4 Diagram of the similarity function *sim*, with its use of the δ_{global} and δ_{local} distance functions.

In figure 5.4, whereas the i-th local distance function δ_{local}^i calculates the distance between the variable values v_i^x and v_i^y, the temporal similarity function δ_{temp} quantifies the distance be-

tween the event sequence s^x and s^y. The local distance outputs are used by the global distance δ_{global}, and finally, this function is used by the function *sim* to compute the similarity between the cases c^x and c^y.

At this point, the definitions of similarity between two temporal cases, as well as the definition of local and temporal distance functions, are given.

Definition 5.10. (Local Distance between two Variables) Given the variable V_i, the *local distance* function δ^i_{local} is defined as follows:

$$\delta^i_{local} : V_i \times V_i \rightarrow \mathbb{R}^+$$
$$\delta^i_{local}(v^x_i, v^y_i) \in \mathbb{R}^+$$

The description of the δ^i_{local} distance depends on the type of variable. For instance, if the variable is numeric, this distance could be the absolute difference between the values, or if the variable is symbolic (or categorical), the distance could be either 0 if they are equal or the maximum possible distance if the symbols are different.

$$\text{numeric: } \delta_{local}(v^x_i, v^y_i) = |v^x_i - v^y_i|$$

$$\text{symbolic: } \delta_{local}(v^x_i, v^y_i) = \begin{cases} 0 \text{ if } v^x_i = v^y_i \\ \text{maximum distance otherwise} \end{cases}$$

Similarly, introducing the definition of the local distance between event sequences is possible.

Definition 5.11. (Local Distance between Event Sequences) The *local distance* function of two event sequences is defined as:

$$\delta_{temp} : S \times S \rightarrow \mathbb{R}^+$$
$$\delta_{temp}(s^x, s^y) \in \mathbb{R}^+$$

Every distance function between events sequences must fulfils, at least, the following properties:

- **Reflexivity:** $\forall s \in S : \delta_{temp}(s, s) = 0$.
- **Identity of indiscernible:** If $\delta_{temp}(s^x, s^y) = \delta_{temp}(s^y, s^x) = 0 \rightarrow s^x = s^y$.

As happen with local distance functions, there is a wide number of δ_{temp} function descriptions [104, 114, 173]. In the next subsection are proposed some of them.

Definition 5.12. (Global Distance between two Temporal Problems) Given the temporal problem domain Π, the *global distance* function δ_{global} is defined as:

$$\delta_{global} : \Pi \times \Pi \rightarrow \mathbb{R}^{+}$$
$$\delta_{global}(\pi^x, \pi^y) \in \mathbb{R}^{+}.$$

There is a myriad of global distance functions in the literature, such as Manhattan, Euclidean or Minkowski, Chevichev, among others. Details of these measures may be found in [168, 169]. In particular, one of the most used global distance in CBM algorithms is the Minkowski distance:

$$\text{Minkowski: } \delta_{global}(\pi^x, \pi^y, r) = \left(\sum_{i=1}^{n} \left(\delta^i_{local} \left(v^x_i, v^y_i \right)^r \right) + \delta_{temp} \left(s^x, s^y \right)^r \right)^{\frac{1}{r}},$$

where $r \in \mathbb{N}^{+}$. From the Minkowski distance are defined the well known Euclidean ($r = 2$) and Manhattan ($r = 1$) distances.

$$\text{Euclidean: } \delta_{global}(\pi^x, \pi^y, 2) = \sqrt{(\sum_{i=1}^{n} \left(\delta^i_{local} \left(v^x_i, v^y_i \right)^2 \right) + \delta_{temp} \left(s^x, s^y \right)^2}$$
$$\text{Manhattan: } \delta_{global}(\pi^x, \pi^y, 1) = \sum_{i=1}^{n} \left(|\delta^i_{local} \left(v^x_i, v^y_i \right)| \right) + |\delta_{temp} \left(s^x, s^y \right)|$$

Definition 5.13. (Similarity between two Temporal Cases) Given a temporal case-base TC, the *similarity* between two temporal cases $c^x, c^y \in TC$ is defined as follows:

$$sim : TC \times TC \rightarrow [0, 1]$$
$$sim(c^x, c^y) \in [0, 1]$$

The similarity function between cases must fulfil the following properties:

- **Reflexivity :** $\forall c \in TC : sim(c, c) = 1$.
- **Identity of indiscernible:** If $sim(c^x, c^y) = sim(c^y, c^x) = 1 \Rightarrow c^x = c^y$.

Different description of similarity function can be found in the literature. In particular, the following similarity function is proposed [47]:

$$sim(c^x, c^y) = 1 - \frac{\delta_{global}(\pi^x, \pi^y)}{max\left(\delta_{global}(\pi^x, \pi^y)\right) - min\left(\delta_{global}(\pi^x, \pi^y)\right))},$$

where the functions *max* and *min* stand for the maximum and minimum values from the co-domain of the global distance function. The reason to do so is to normalize the values of the

similarity co-domain function in the interval $[0,1]$.

5.4.1 Edit distance between two event sequences

This distance measures the related cost of transforming an event sequence into another, using insertion, deletion and alignment/move operations. Besides, the measure is also known as *Levenshtein distance* due to the last name of the author who proposed it [94], although the Levenshtein distance measure only involves strings. In [104, 114] a modification of the edit distance is proposed to manage event sequences, where the alignment/move involves the temporal distance between two events. The requirement of this algorithm is that the distance between time-stamps must be normalized within the interval $[0,1]$ and no operation cost should be higher than 1, that is, the cost of inserting or removing one event should be lower than 1. Otherwise, the cost of align two events with the same event type can be higher than the inserting and removing operations. However, tackling with this normalization is straightforward: the distance between two time-stamps have to be divided with the difference between the maximum and minimum known time-stamps values.

Example 5. Let c^x and c^y be the following two temporal cases, each one with the solution $\in \{\omega_1, \omega_2\}$:

$$c^x = (\pi^x, \omega^x) = \left(\left\langle (e_1^x, 2), (e_2^x, 5), (e_3^x, 8) \right\rangle, \omega^x\right)$$
$$c^y = (\pi^y, \omega^y) = \left(\left\langle \left(e_1^y, 0\right), \left(e_3^y, 3\right), \left(e_2^y, 9\right) \right\rangle, \omega^y\right)$$

So as to calculate the similarity between both temporal cases, only the description problems are needed though.

$$\pi^x = \left\langle (e_1^x, 2), (e_2^x, 5), (o_3^x, 8) \right\rangle$$
$$\pi^y = \left\langle \left(e_1^y, 0\right), \left(e_3^y, 3\right), \left(e_2^y, 9\right) \right\rangle$$

Assuming that the cost w for all operations is equal to 1, the resulting matrix of the algorithm 18 is the following:

		$(e_1^x, 2)$	$(e_2^x, 5)$	$(e_3^x, 8)$
	0	1	2	3
$(e_1^y, 0)$	1	0.2	1.2	2.2
$(e_3^y, 3)$	2	1.2	2.2	1.7
$(e_2^y, 9)$	3	2.2	1.6	**2.6**

The distance between the two event sequences is the value on the bottom right cell of the matrix. Hence, following the definition 5.13, and assuming that the maximum distance is 6

Algorithm 18 Calculate the edit distance δ_{temp}^{Edit} between two sequences s^x, x^y

Input: Two event sequences $s^x = \langle (e_1^x, t_1^x), \ldots, (e_n^x, t_n^x) \rangle$ and $s^y = \langle (e_1^y, t_1^y), \ldots, (e_m^y, t_m^y) \rangle$, the costs $w(e^x), w(e^y)$ of the *insertion* and *deletion* operations.
Output: Edit distance $\delta_{temp}^{Edit}(s^x, s^y)$.

$r \leftarrow$ matrix of $n \times m$ dimensions
$r(0,0) \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
  $r(i,0) \leftarrow r(i-1,0) + w(e^x)$
end for
for $j \leftarrow 1$ **to** n **do**
  $r(0,j) \leftarrow r(0,j-1) + w(e^y)$
end for
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    $update^x \leftarrow r(i-1,j) + w(e^x)$
    $update^y \leftarrow r(i,j-1) + w(e^y)$
    $align \leftarrow r(i-1,j-1)$
    if $e_i^x = e_j^y$ **then**
      $align \leftarrow align + (\frac{|t_i^x - t_j^y|}{max_t - min_t})$
    else
      $align \leftarrow align + w(e^x) + w(e^y)$
    end if
    $r(i,j) \leftarrow min\left(update_{S_x}, update_{S_y}, align\right)$
  end for
end for
return $r(n,m)$

and the minimum 0, corresponding to removing and inserting all the events and no change at all, then the normalised similarity between the temporal cases c^x and c^y is:

$$sim(c^x, c^y) = 1 - \frac{\delta_{temp}^{Edit}(\pi^x, \pi^y)}{6} = 1 - \frac{2.6}{6} = 0.567$$

5.4.2 Match & Mismatch distance

The Match & Mismatch (henceforth M&M) is an algorithm that quantifies the distance between two event sequences. This algorithm splits each event sequence into several event subsequences, one list for each event type [174]. The algorithm assumes that computing the distance for these sequences is a simpler problem [173].

For every event type the algorithm obtains an edit-distance matrix. From all the matrices built, four different measures are calculated: the time difference between the events (TD), the missing events (NM), the extra event (NE) and the number of swamping events (SN). These

measures are the basis of one overall score that quantifies the similarity between the two event sequences.

Algorithm 19 Calculate the M&M distance δ_{temp}^{MM} between two sequences s^x, s^y

Input: Two event sequences
$s^x = \langle (e_1^x, t_1^x), \ldots, (e_n^x, t_n^x) \rangle$,
$s^y = \langle (e_1^y, t_1^y), \ldots, (e_m^y, t_m^y) \rangle$,
and $w_{td}, w_{nm}, w_{ne}, w_{ns}$ as weight parameters.
Output: Match & Mismatch similarity between the two given sequences.

$TD \leftarrow 0, NM \leftarrow 0, NE \leftarrow 0, NS \leftarrow 0$
for all event type $e \in s^x$ or s^y **do**
  $S_e^x \leftarrow$ events with the event type e from s^x
  $S_e^y \leftarrow$ events with the event type e from s^y
  if $|S_e^x| \leq |S_e^y|$ **then**
    $NE \leftarrow NE + |S_e^x| - |S_e^y|$
  else
    $NM \leftarrow NM + |S_e^y| - |S_e^x|$
  end if
  $n \leftarrow |S_e^x|, m \leftarrow |S_e^y|$
  $r \leftarrow$ matrix of $(n+1) \times (m+1)$
  $r(0,0) \leftarrow 0$
  for $i \leftarrow 1$ **to** n **do**
    $r(0,i) \leftarrow 0$
    $r(i,0) \leftarrow 0$
  end for
  for $j \leftarrow 1$ **to** n **do**
    for $i \leftarrow 1$ **to** m **do**
      $update_x \leftarrow r[i-1, j] + |t_j^x - t_i^y|$
      $update_y \leftarrow r[i, j-1] + |t_j^x - t_i^y|$
      $update_{xy} \leftarrow r[i-1, j-1] + |t_j^x - t_i^y|$
      $r[i, j] \leftarrow min(update_x, update_y)$
      $r[i, j] \leftarrow min(r[i, j], update_{xy})$
    end for
  end for
  $TD \leftarrow TD + r(difference, m-2)$
end for
$NS \leftarrow$ number of swamping events
return $4 - \frac{w_{td}TD - w_{nm}NM - w_{ne}NE - w_{ns}NS}{w_{td} + w_{nm} + w_{ne} + w_{ns}}$

Given two event sequences, M&M obtains distance values within the interval $[0,1]$, where the lower values mean that the event sequences are similar, whereas higher values are given for two no similar event sequences..

Example 6. Consider the same temporal case-base as in the example 5, now we calculate the similarity between the two existing cases with M&M. For each case c^x and c^y, the events in

the event sequences are assigned to different set accordingly to their event type. Thus, having the following event types $E = \{e_1, e_2, e_3\}$, the algorithm builds up the following sets:

	$s^x_{e_1} = \{(e_1,2)\}$	$sec^x_{e_2} = \{(e_2,5)\}$	$sec^x_{e_3} = \{(e_3,8)\}$	
	$s^y_{e_1} = \{(e_1,0)\}$	$s^y_{e_2} = \{(e_2,9)\}$	$s^y_{e_3} = \{(e_3,3)\}$	TOTAL
NE	0	0	0	0
NM	0	0	0	0
TD	0.2	0.4	0.5	1.1

The number of swamping events $NS = 1$, because we can swap one events to make a match between the event sorting.

$$\pi^x = \langle (e_1,2), (e_2,5), (e_3,8) \rangle$$
$$\text{swap events } (e_2,5) \text{ and } (e_3,8)$$
$$\pi^x = \langle (e_1,2), (e_3,8), (e_2,5) \rangle.$$

Assuming that the variables NE, NM, TD and NS have the same weight and are equal to 1, and that the maximum M&M distance is 6, then the similarity between the two cases c^x and c^y is:

$$sim(c^x, c^y) = 1 - \frac{\delta^{MM}_{temp}(\pi^x, \pi^y)}{6} = 1 - \frac{4 - NE + NM + NS + TD}{6} =$$
$$= 1 - \frac{4 - 0 + 0 + 1 + 1.1}{6} = 1 - \frac{1.9}{6} = 0.683.$$

5.5 Temporal case-base maintenance

As mentioned in the chapter 2,CBM 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 [92]. In particular, according to [169], CBM aims to reduce the case-base size or to improve the quality of the solutions provided by the CBR system [126].

Therefore, CBM algorithms can be used to reduce the number of cases and maintain the accuracy of the CBR system at the same time. Whereas CBM 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 CBM algorithm is suitable for certain types of case-bases that share some indicators, such as redundancy and noise levels.

In the chapter 2, there was introduced the different dimensions in which CBM algorithms can be classified: *case search*, *direction*, *order sensitive* and *type of cases* to retain, as well as the different CBM algorithm families: *NN*, *DROP* and *Competence*. On top of those families and dimensions, a myriad of CBM algorithms have been proposed in the literature [18, 62, 97, 110, 149, 167, 169, 170]. For the sake of simplicity, table 5.1 shows the different dimensions and algorithms, as well as a some CBM algorithms of each of their combinations. That is, each cell represents a CBM algorithm of a given family algorithm: *NN*, *DROP* and *Competence*, and a given dimension: *case search direction*, *order sensitive* and *type of cases* to retain. For instance, the algorithm CNN is a member of the *NN* family, which has an incremental case search direction, is sensitive to the case ordering and retains cases in the borders of case clusters.

Dimensions vs. Family		**NN**	**DROP**	**Competence**
Case search direction	Incremental	CNN	-	RC-FP
	Decremental	RENN	DROP1,2,3	ICF
Order sensitive	Yes	CNN,RENN	DROP1	ICF
	No	-	DROP2,3	RC-FP
Type of Cases to Retain	Border	CNN	DROP1	ICF,RC-FP
	Central	RENN	DROP2,3	COV-FP

Table 5.1 Outline of CBM algorithms by their dimension and family.

However, up to our knowledge there is no study related with CBM algorithms working with temporal case-bases. Owing to this, the proposed subsection is intended to adapt one existing CBM algorithm of each dimension in order to study whether they can be successfully used with temporal case-bases or not. Table 5.2 depicts the CBM algorithms that have been adapted from the table 5.1.

Dimensions vs. Family		**NN**	**DROP**	**Competence**
Case search direction	Incremental	t-CNN	-	t-RC-FP
	Decremental	t-RENN	t-DROP1,2,3	t-ICF
Order sensitive	Yes	t-CNN,t-RENN	t-DROP1	t-ICF
	No	-	t-DROP2,3	t-RC-FP
Type of Cases to Retain	Border	t-CNN	t-DROP1	t-ICF,t-RC-FP
	Central	t-RENN	t-DROP2,3	t-COV-FP

Table 5.2 Outline of proposed temporal CBM algorithms by their dimension and family.

Next, for every algorithm in table 5.2, a pseudo code of their execution is given. Besides, some common functions are given so as to share behaviour between algorithms, such as the search of nearest neighbours or the computation of the global distance δ_{global}.

The functions $max(\delta_{global}^{temporal}(c^x, c^y))$ and $min(\delta_{global}^{temporal}(c^x, c^y))$ returns the maximum and minimum global distances that can be found from two cases in the temporal-case base TC.

Algorithm 20 *sim*

Input: Two cases $c^x = (\pi^x, \omega^x)$, $c^y = (\pi^y, \omega^y)$.
Output: A real number in the interval $[0,1]$

1: **return** $1 - \frac{\delta_{global}^{temporal}(c^x,c^y)}{max(\delta_{global}^{temporal}(c^x,c^y)) - min(\delta_{global}^{temporal}(c^x,c^y))}$

Algorithm 21 $\delta_{global}^{temporal}$

Input: Two cases $c^x = (\pi^x, \omega^x)$, $c^y = (\pi^y, \omega^y)$
Output: A real number in the interval $\mathbb{R}^+$

1: $dist \leftarrow 0$
2: **for all** $v_i^x \in \pi^x, v_i^y \in \pi^y$ **do**
3: $\quad dist \leftarrow dist + \delta_{local}^i(v_i^x, v_i^y)^2$
4: **end for**
5: $dist \leftarrow dist + \delta_{temp}(s^x, s^y)^2$
6: $dist \leftarrow (dist)^{\frac{1}{2}}$
7: **return** $dist$

Algorithm 22 t-NearestNeighbour

Input: A case $c^x = (\pi^x, \omega^x)$, a temporal case-base TC, a global distance $\delta_{global}^{temporal}$, and $k \in \mathbb{Z}$ as the number of neighbours.
Output: A set of cases NN by ascending order according to their distances to c^x.

1: $NN^x \leftarrow \emptyset$
2: **for all** $c^y = (\pi^y, \omega^y) \in TC$ **do**
3: $\quad similarity \leftarrow sim(c^x, c^y)$
4: $\quad NN^x \leftarrow NN^x \cup \{(c^y, similarity)\}$
5: $\quad$ sort NN^x in descending order
$\qquad\qquad$ according to their similarity
6: $\quad$ **if** size of $NN^x > k$ **then**
7: $\qquad$ remove last elements in NN
8: $\quad$ **end if**
9: **end for**
10: **return** NN

5.6 Experiments

While CBM algorithms have been used successfully with non-temporal case-bases, however, there is no deep study to enlighten the understanding of their performance with temporal case-bases. To this end, several *non temporal* and *temporal* factors are proposed. The objective is to study the dependency between these factors and the accuracy and reduction rates that can be achieved with the maintained case-base by a temporal CBM algorithm (hereinafter tCBM).

Among the *non temporal factors*, two factors are defined: the *redundancy* of the case-base, as the proportion of redundant cases, and the *case-base size*. The reason to use them is

Algorithm 23 correctlyClassified

Input: A case $c^x = (\pi^x, \omega^x)$, a temporal case-base TC, a global distance $\delta_{global}^{temporal}$, and $k \in \mathbb{Z}$ as the number of neighbours.
Output: A boolean value, it returns *true* if c^x is correctly classified by a nearest neighbour in the temporal case-base TC.

$NN^x \leftarrow$ t-NearestNeighbour$(c^x, TC, \delta_{global}^{temporal}, k)$
$match \leftarrow 0$
for all $c^y = (\pi^y, \omega^y) \in NN^x$ **do**
  if $\omega^x = \omega^y$ **then**
    $match \leftarrow match + 1$
  end if
end for
if $match \geq \frac{k}{2}$ **then**
  return *true*
end if
return *false*

Algorithm 24 t-CNN

Input: A temporal case-base TC, $k \in \mathbb{Z}$ as the number of neighbours .
Output: A maintained temporal case-base TC'

$TC' \leftarrow \emptyset$
for all $\omega \in \Omega$ **do**
  $c^x \leftarrow$ random case of TC with solution ω
  $TC \leftarrow TC - \{c^x\}$
  $TC' \leftarrow TC' \cup \{c^x\}$
end for
repeat
  for all $c^x = (\pi^x, \omega^x) \in TC$ **do**
    $TC - \{c^x\}$
    if not $\boxed{correctlyClassified(c^x, TC, \delta_{global}^{temporal}, k)}$ **then**
      $TC' \leftarrow TC' \cup \{c^x\}$
    else
      $TC \leftarrow TC \cup \{c^x\}$
    end if
  end for
until TC without changes
return TC'

because the purpose of the tCBM algorithms is the deletion of redundant and noisy cases from the temporal case-base.

Regarding the *temporal factors*, the length of the sequence, the density of the event in the event sequence, and the distribution of the events through the time are factors that may affect the tCBM working process. The *length of sequence* is defined as the number of instants

Algorithm 25 t-RENN

Input: A temporal case base TC
Output: A maintained temporal case-base TC'

1: $TC' \leftarrow TC$
2: **repeat**
3: **for all** $c^x \in TC'$ **do**
4: **if not** $\boxed{correctlyClassified(c^x, TC, \delta_{global}^{temporal}, 3)}$ **then**
5: $TC' \leftarrow TC' - \{c^x\}$
6: **end if**
7: **end for**
8: **until** TC' without changes
9: **return** TC'

that the event sequence could represent. For instance, a length of 100 means that the event sequence could have events in the timestamps interval $[1, 100]$. The *density* of events refers to the amount of events by the length of event sequence. Thus, a density of 0.5 means that half of the instants have an event. Lastly, the *distribution* of event types represents the case where one of the event types have higher density appearance than the rest of the event types.

Finally, the type of temporal distance between event sequences is considered as well, where the two proposed distances to study are the Edit Distance [114] and M&M [173].

For each factor, an experiment has been designed in order to cover temporal and non-temporal factors that may represent a temporal case-base, and so to study their effects on the accuracy and reduction rates of the proposed tCBMs.

5.6.1 Enumeration of the experiments carried out

The following experiments have been run in order to study which factors could affect the results of the proposed tCBMs.

On the one hand, the following experiments are aimed to study the non-temporal factors:

1. The effect of the case-base size in the tCBM results.
2. The effect of the redundancy in the tCBM results.

On the other hand, the temporal factors are studied in the following experiments:

3. The density of events in the event sequence. Given an interval of time, the density of an event sequence determines the number of events in the event sequence. That is:

$$density = \frac{num.events * |E|}{length}$$

4. The results of the accuracy for different temporal similarity measures.

Algorithm 26 t-DROP1

Input: A temporal case-base TC, a number of neighbours $k \in \mathbb{Z}$
Output: A maintained temporal case-base TC'

1: $TC' \leftarrow TC$
2: **for all** $c^x \in TC'$ **do**
3: $NEIGHBOUR^x \leftarrow$ t-NearestNeighbours$(c, TC', \delta_{global}^{temporal}, k+1)$
4: **for all** $c^y \in NEIGHBOUR^x$ **do**
5: $ASSOCIATE^y \leftarrow ASSOCIATE^y \cup \{c^x\}$
6: **end for**
7: **end for**
8: **for all** $c^x \in TC'$ **do**
9: $with \leftarrow 0$
10: $without \leftarrow 0$
11: $TC'' \leftarrow TC' - \{c^x\}$
12: **for all** $c^y \in ASSOCIATE^x$ **do**
13: **if** $correctlyClassified(c^y, TC')$ **then**
14: $with \leftarrow with + 1$
15: **end if**
16: **if** $correctlyClassified(c^y, TC'')$ **then**
17: $without \leftarrow without + 1$
18: **end if**
19: **end for**
20: **if** $without \geq with$ **then**
21: $TC' \leftarrow TC''$
22: **for all** $c^y \in ASSOCIATE^x$ **do**
23: $NEIGHBOUR^y \leftarrow NEIGHBOUR^y - \{c^x\}$
24: $c^z \leftarrow$ t-NearestNeighbour$(c^y, TC, \delta_{global}^{temporal}, 1)$
25: $NEIGHBOUR^y \leftarrow NEIGHBOUR^y \cup \{c^z\}$
26: **end for**
27: **for** $c^y \in NEIGHBOUR^x$ **do**
28: $ASSOCIATE^y \leftarrow ASSOCIATE^y - \{c^x\}$
29: **end for**
30: **end if**
31: **end for**
32: **return** TC'

5. The impact of different distribution of event types. In particular, whether the over occurrence of one particular event type over the rest may affect the accuracy of the CBR system.

6. Finding, where possible, which type of tCBM could fail performing the maintenance task when dealing with temporal case-bases.

Algorithm 27 t-ICF

Input: A temporal case-base TC, a number of neighbours $k \in \mathbb{Z}$
Output: A maintained case-base TC

1: $TOREMOVE \leftarrow \emptyset$
2: **for all** $c^x \in TC$ **do**
3: **if not** $\boxed{correctlyClassified(c^x, TC, \delta_{global}^{temporal}, k)}$ **then**
4: $TOREMOVE \leftarrow TOREMOVE \cup \{c^x\}$
5: **end if**
6: **end for**
7: $TC \leftarrow TC - TOREMOVE$
8: **repeat**
9: **for all** $c^x = (\pi^x, \omega^x) \in TC$ **do**
10: **for all** $c^y = (\pi^y, \omega^y) \in \boxed{\text{t-NearestNeighbour}(c^x, TC, \delta_{global}^{temporal}, k)}$ **do**
11: **if** $(\omega^x = \omega^y)$ **then**
12: $COVERAGE^x \leftarrow COVERAGE^x \cup \{c^y\}$
13: **end if**
14: **end for**
15: **for all** $c^y \in TC - c^x$ **do**
16: **for all** $c^z \in \boxed{\text{t-NearestNeighbour}(c^y, TC, \delta_{global}^{temporal}, k)}$ **do**
17: **if** $\boxed{\delta_{global}^{temporal}(c^x, c^z)} = 0$ **then**
18: $REACHABLE^x \leftarrow REACHABLE^x \cup \{c^y\}$
19: **end if**
20: **end for**
21: **end for**
22: **end for**
23: $TOREMOVE \leftarrow \emptyset$
24: **for all** $c^x \in TC$ **do**
25: **if** $|REACHABLE^x| > |COVERAGE^x|$ **then**
26: $TOREMOVE \leftarrow TOREMOVE \cup \{c^x\}$
27: **end if**
28: **end for**
29: $TC \leftarrow TC - TOREMOVE$
30: **until** not progress in TC
31: **return** TC

5.6.2 Experiments details

The chosen evaluation method is $\alpha\beta$, with $\alpha = 10$ and $\beta = 10$. Despite Hold-Out and Cross-Validation could have been used for the evaluation, as was shown by the conducted experiments in chapter 4, $\alpha\beta$ showed to be a reliable method to evaluate CBM algorithms, particularly with values $\alpha = 10$ and $\beta = 10$. Therefore, with $\alpha\beta$ from the original temporal case-base the training and test sets are created β times, and every tCBM algorithm is executed α times

Algorithm 28 t-RC-FP

Input: A temporal case-base TC, a value $k \in \mathbb{Z}$ as the number of neighbours.
Output: A maintained temporal case-base TC'

$R \leftarrow$ t-RENN(TC)
$TC' \leftarrow \emptyset$
while $R \neq \emptyset$ **do**
  for all $c^x = (\pi^x, \omega^x) \in R$ **do**
    $COVERAGE^x \leftarrow \emptyset$
    $REACHABLE^x \leftarrow \emptyset$
    for all $c^y = (\pi^y, \omega^y) \in$ t-NearestNeighbour$(c^x, R, \delta^{temporal}_{global}, k)$ **do**
      if $(\omega^x = \omega^y)$ **then**
        $COVERAGE^x \leftarrow COVERAGE^x \cup \{c^y\}$
      end if
    end for
    for all $c^y \in TC - \{c^x\}$ **do**
      for all $c^z \in$ t-NearestNeighbour$(c^y, TC, \delta^{temporal}_{global}, k)$ **do**
        if $\delta^{temporal}_{global}(c^x, c^z) = 0$ **then**
          $REACHABLE^x \leftarrow REACHABLE^x \cup \{c^y\}$
        end if
      end for
    end for
  end for
  $max \leftarrow -1$
  for all $c^x \in R$ **do**
    $cov \leftarrow 0$
    for all $c^y \in COVERAGE^y$ **do**
      $cov \leftarrow cov + |REACHABILITY^y|$
    end for
    if $\frac{1}{cov} > max$ **then**
      $cov \leftarrow \frac{1}{cov}$
      $c^{next} \leftarrow c^x$
    end if
  end for
  $TC' \leftarrow TC' \cup \{c^{next}\}$
  $R \leftarrow R - COVERAGE^{next}$
end while
return TC'

using the training set. At the end of every experiment, there will be up to β accuracy and reduction rate results, which are the evaluation scores of the experiment. For further details regarding the $\alpha\beta$, the evaluation method can be found in chapter 4.

Regarding the CBR system configuration, a K-NN classifier is used as retrieval mechanism,

with $K = 3$. As an adaptation process to create the output solution of the system, the most common solution is returned among the neighbours retrieved by K-NN.

Since many experiments need to be carried out for each aforementioned scenario, next a set of parameters are defined so to generate a synthetic case-base and to identify each experiment.

The following parameters are common to all the experiments:

- There are three event types $E = \{A, B, C\}$.
- The set of solutions is the following $\Omega = \{T1, T2, T3, T4, T5, T6\}$. That is, every case will have an event sequence that contains event types from E, and a solution from Ω.
- The length of the sequences may appear in the interval $length \in [1, 100]$.
- The Edit Distance is chosen as temporal distance function in all the experiments, excepting in the experiment focused in the results given by tww different temporal distance functions.

Regarding each particular experiment, the details are the following:

1. The temporal case-base size can be one of the following values $\{60, 300, 600\}$. The level of redundancy is 85%, and the density the events types is 0.3.
2. The three levels of redundancy in the case-case of this experiment are $redundancy \in \{0.85, 0.80, 0.75\}$. The size of the case-base is 600 case, so to ensure that the case-base have enough cases to cover the problem domain properly.
3. The different values of density of event types in the experimentation are $\{0.1, 0.3, 0.6\}$. The temporal case-bases for this experiment contain 600 cases, with 85% of redundant cases.
4. This experiment is run with the M&M temporal distance function. Besides, the temporal case-base size can be one $\{60, 300, 600\}$, with level of redundancy of 85%, and density of events types as 0.3.
5. Over representation of events from a particular event type. This means that one event type occurs more frequently than the rest. A case-base with over representation of an event type is labelled as *Unbalance*, and a case-base with the density of event equally distributed between all the event types is labelled as *Balanced*. The temporal case-bases for this experiment contain 600 cases, with 85% of redundant cases and a density of 0.3.
6. From the observation of all the previous experiments, we will determine whether any tCBM is not suitable for being used with temporal case-bases.

5.6.3 Synthetic generation of the temporal case-bases

The construction of the synthetic case-bases is done through the creation of cases from a set of patterns that represents different types of cases. Given the set of solutions Ω and the set of event types E, each event type $e \in E$ occurs in a type of case with solution $\omega \in \Omega$ according to the following function:

$$Occurs(e \in E, t, t_1, t_2, t_3, t_4) = \begin{cases} (t < t_1) \vee (t > t_4) : 0 \\ (t_1 \leq t) \wedge (t \leq t_2) : \frac{t-t_1}{t_2-t_1}\frac{3}{4} \\ (t_2 \leq t) \wedge (t \leq t_3) : \frac{3}{4} \\ (t_3 \leq t) \wedge (t_3 \leq t_4) : \frac{t_4-t}{t_4-t_3}\frac{3}{4} \end{cases}$$

	A				B				C			
	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
T1	0	12	17	30	20	40	50	70	60	75	85	100
T2	0	12	17	30	60	75	85	100	20	40	50	70
T3	20	40	50	70	0	12	17	30	60	75	85	100
T4	60	75	85	100	0	12	17	30	20	40	50	70
T5	20	40	50	70	60	75	85	100	0	12	17	30
T6	60	75	85	100	20	40	50	70	0	12	17	30

Table 5.3 Parameters to create each event type for every event sequence type

That is, for an instant of time t, every event type has associated a probability for occurring in the event sequence. In fact, there is the possibility of two events of two different event types could occur at the same time. The values of t_1, t_2, t_3, t_4 are given in table 5.3.

For instance, the probability to occur for the event type A, for the event sequence $T1$ in the instant 10 is:

$$Occurs(A, 0, 12, 17, 30) = \frac{10-0}{12-0}\frac{3}{4} = \frac{30}{48} = 0.625$$

The algorithm 29 shows the detailed steps to create the synthetic case-base for each experiment.

For instance, a case c with balanced event sequence with 9 events is the following:

$$c = (\langle (A,3), (A,12), (A,20), (B,38), (B,45), (B,60), (C,74), (C,87), (C,93) \rangle, "T1")$$

In contrast, an unbalance temporal case-base with 9 event sequences can be the following:

$$c = (\langle (A,3), (A,8), (A,12), (A,15), (A,20), (A,23), (B,30), (B,57), (C,60) \rangle, "T1")$$

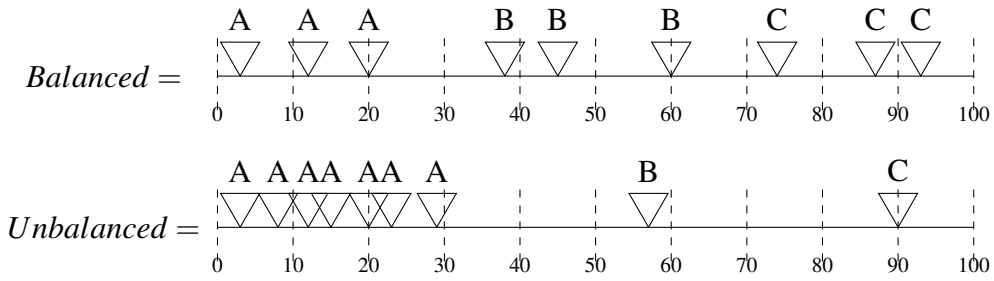


Fig. 5.5 Representation of balanced and unbalanced event sequences.

5.6.4 Results

In the following we show the results of the experiments carried out according to the goals aforementioned stated.

The effect of the case-base size in the tCBMs

We firstly analyse how the size of the original temporal case-base affects to the accuracy of the CBR system when tCBMs algorithms are used to reduce the number of cases of the case-base. Figure 5.6 illustrates the accuracy, and figure 5.7 depicts the reduction rate of the maintained case-base.

The darkness lines represent each average of every tCBM, and the blur lines are the maximum and minimum accuracies. Closer the distance between these two lines, lower variance on the accuracy results.

In figure 5.7, whereas a value of 1 means that all the cases from the original temporal case-base have been deleted, and a value equal to 0 means no deletion at all.

The effect of redundant cases in the tCBMs

In this experiment, we analyse the effects on the accuracy and reduction rates that may have the proportion of redundant cases in the case-base. Figure 5.8 shows the accuracy results for three different case-bases with the same size but with different levels of redundancy. Again, the dark lines represent the average accuracy for each tCBM, and the grey lines the maximum and minimum accuracies. Regarding the capacity of deletion cases, figure 5.9 depicts the reduction rates of each tCBM. The interpretation is similar to figure 5.7.

The effect of density of events in the tCBMs

Next, the experiment is aimed to analyse the effect of the density in the accuracy and reduction rates for each tCBM. The density is a key aspect in the study of sequences, and it determines the number of events per event sequence. Hence, higher densities implies larger the number of

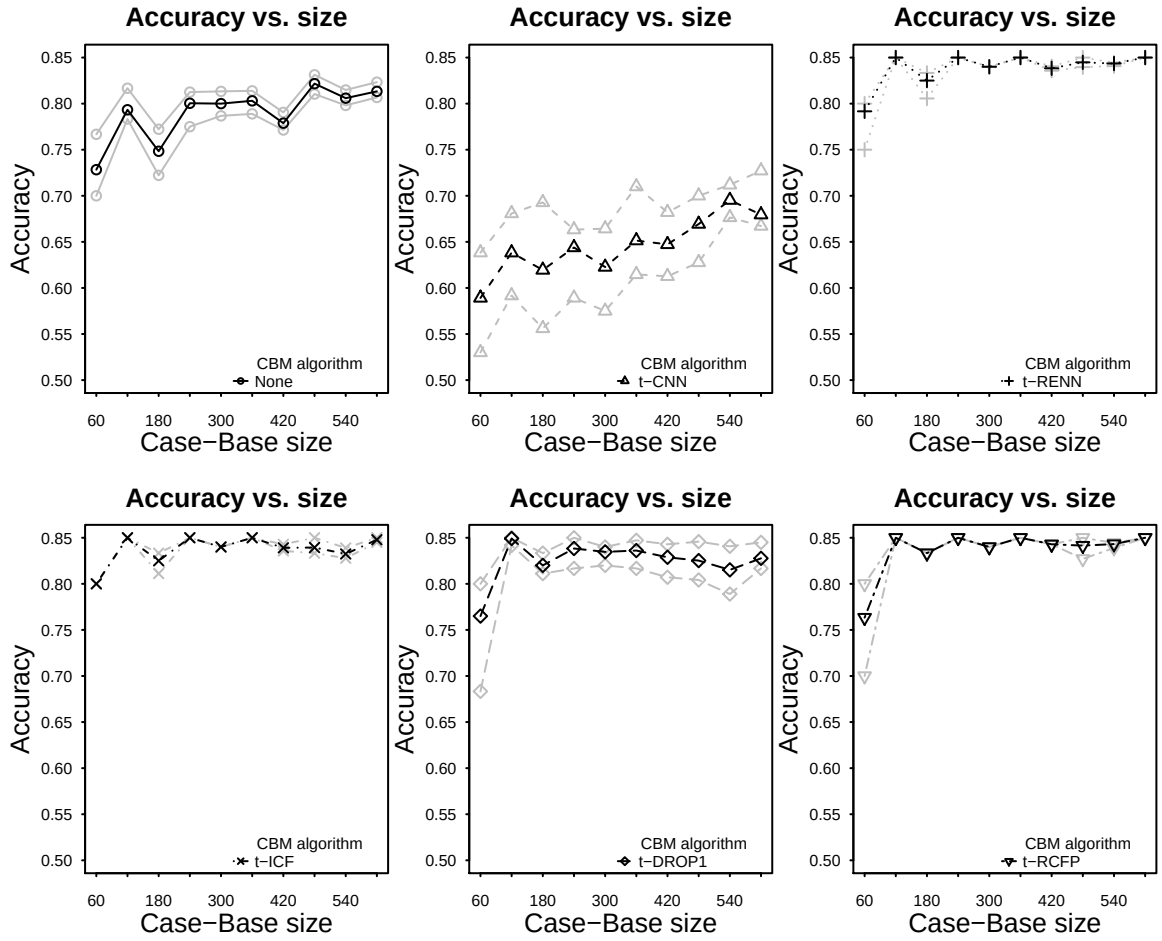


Fig. 5.6 Accuracy with different synthetic temporal case-base with different sizes.

events. Whereas figure 5.10 depicts the different accuracy results with three different cases-bases with different levels of density of events, figure 5.11 plots the reduction rates for this experiment.

The results with different temporal similarity measures

As proceeded in the previous experiment in subsection 5.6.4 with the edit distance temporal function, figures 5.12 and5.13 portray the results for the M&M temporal distance, either for their accuracy and reduction rate for case-base sizes for 60 to 600 cases.

The accuracy average results with the Edit Distance and M&M distance are different. In order to conclude if there is a statistic difference between the results from both measures, table 5.4 show the p-value from a t-test, and the accuracy averages. The null hypothesis is true difference between Edit Distance and M&M accuracy averages is less than 0. Therefore, p-values close to 1 mean that Edit Distance has greater accuracy than M&M. On the contrary, p-values close to 0 means that M&M has better accuracy.

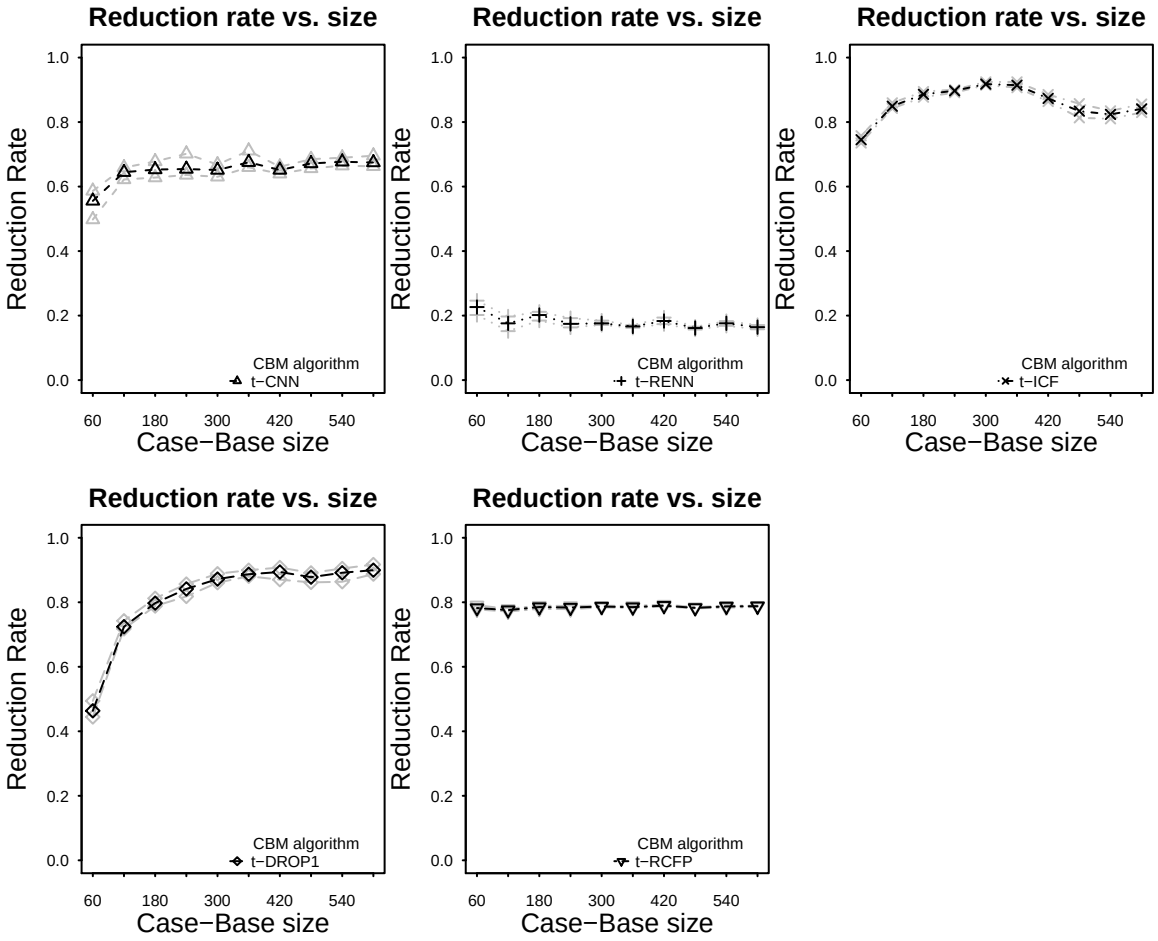


Fig. 5.7 Reduction rate with different synthetic temporal case-base and different size.

	Avg. accuracy		
	p-value	Edit distance	M&M
None	1.00	0.79	0.77
t-CNN	0.03	0.65	0.66
t-RENN	1.00	0.84	0.79
t-ICF	1.00	0.84	0.79
t-DROP1	1.00	0.82	0.76
t-RC-FP	1.00	0.84	0.78

Table 5.4 Probability of Edit Distance accuracy is greater than M&M accuracy with the synthetic case-base.

The reduction rate of the tCBMs may be affected as well by the variation of the temporal similarity measure. Figure 5.13 plots the reduction rates for the each tCBM.

Statistical study is done to analyse the true difference between the reduction rates by the Edit Distance and M&M. Table 5.5 shows the results of a t-test between the reduction rates results of Edit Distance and M&M. In this study, p-values close means the probability that Edit

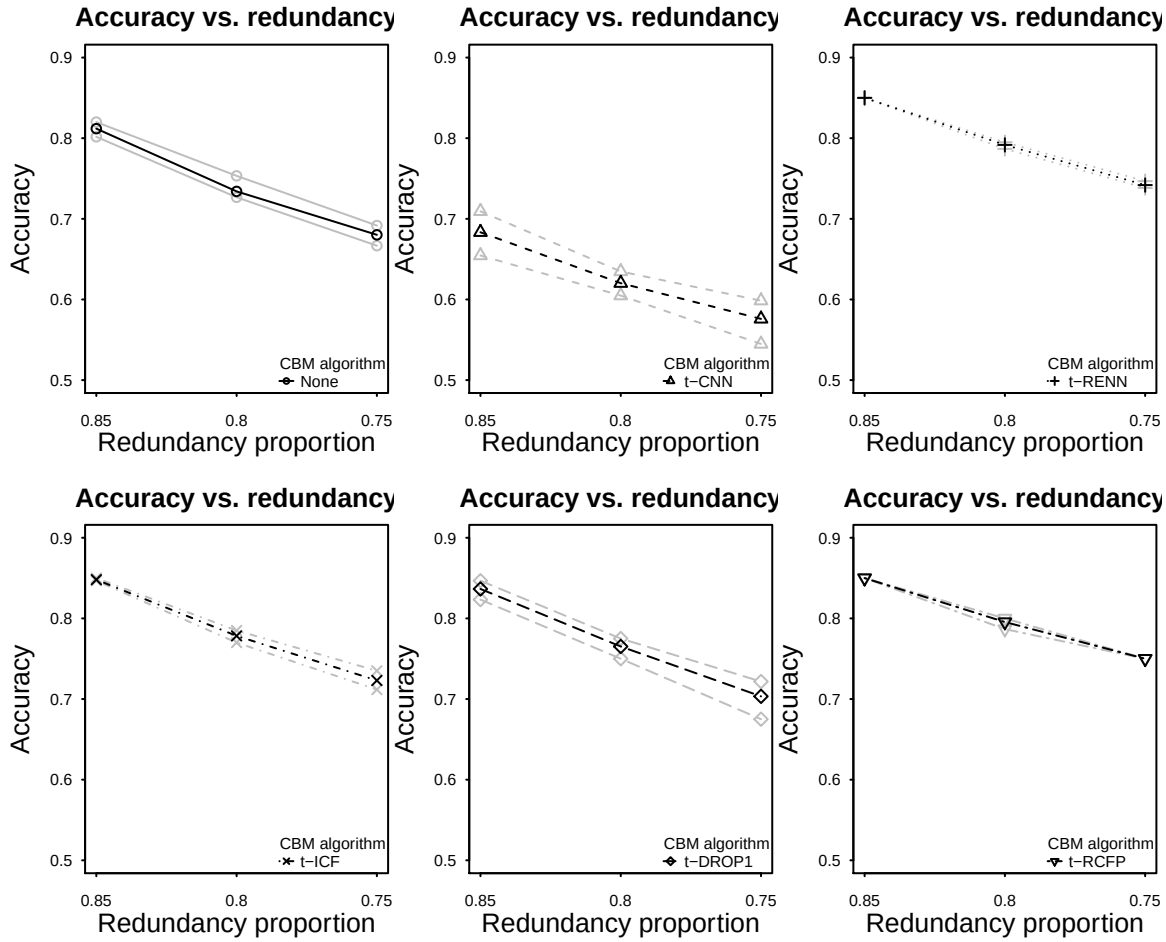


Fig. 5.8 Accuracy with different case-bases and different redundancy levels.

Distance has greater reduction rates than M&M.

		Reduction rate average	
	p-value	Edit distance	M&M
t-CNN	1.00	0.65	0.62
t-RENN	0.00	0.18	0.22
t-ICF	1.00	0.86	0.75
t-DROP1	1.00	0.81	0.74
t-RC-FP	1.00	0.78	0.78

Table 5.5 Probability of the Edit Distance function to have a greater reduction rate than M&M with the synthetic case-base.

The effect of the over-representation of a certain event type

For this experimentation, one of the event types in the event sequence is overrepresented (see Algorithm 29 for details about how to build this event sequence type). The purpose is to

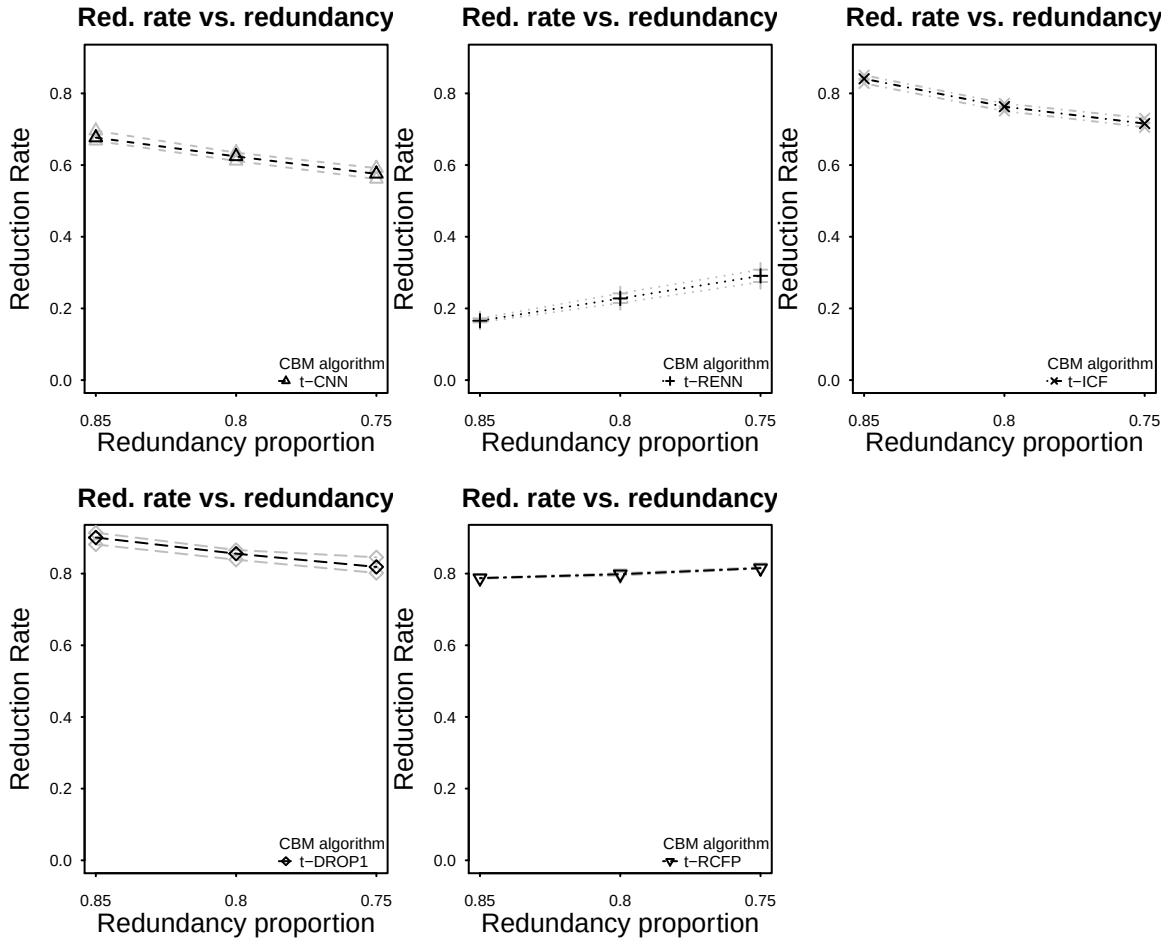


Fig. 5.9 Reduction rate with different case-bases and different redundancy levels.

observe if this factor may affect the output of the tCBM algorithms. That is, if their abilities to reduce the case-base is affected by event sequences with over representation of a given event type.

Figure 5.14 depicts the accuracy results given by the output case-bases by the considered tCBM with an original case-base with over representation of one an event type in the event sequences. Statistic study between the results of both experiments is shown in table 5.6, where the column p-value is the probability that the results given by a tCBM with a balanced and unbalanced temporal case-base would be the same. P-values closer to 0 mean that there is a statistical difference, and p-values closer to 1 belong to those results with no statistical difference.

Similarly, at a glance to the reduction rate results from the experiment 5.6.4 and the current one, the reduction rates sets seem to be fairly similar on to each other.

Figure 5.15 shows the reduction rates from the considered tCBM algorithms.

In order to study if there are statistic differences between the reduction rates with balanced and unbalanced event sequence representations, an statistical study is performed. In table 5.7,

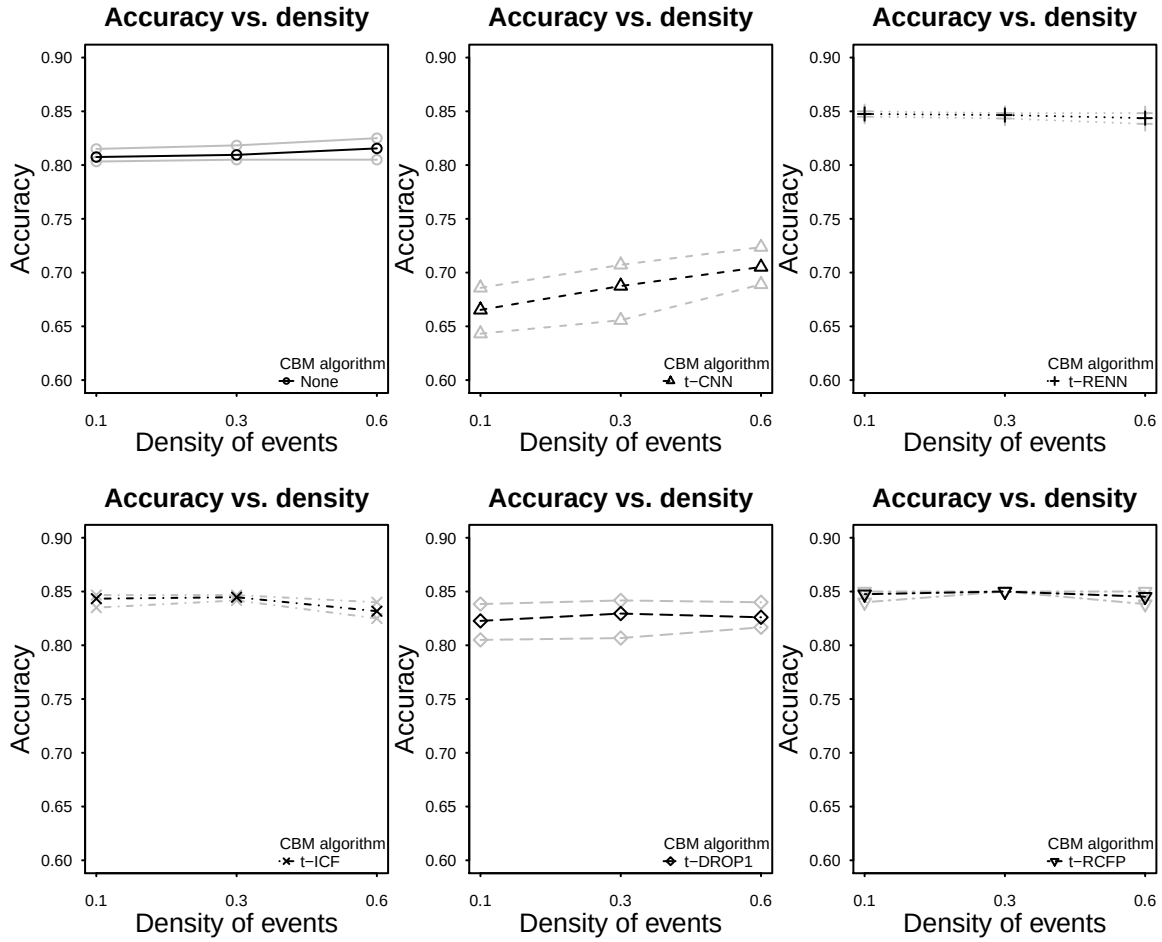


Fig. 5.10 Accuracy varying the density of events within the event sequences.

	Accuracy averages		
	p-value	Balanced	Unbalanced
t-none	0.05	0.79	0.80
t-CNN	0.00	0.65	0.67
t-RENN	0.63	0.84	0.84
t-ICF	0.00	0.84	0.82
t-DROP1	0.01	0.82	0.81
t-RC-FP	0.52	0.84	0.83

Table 5.6 Probability of an accuracy of a case-base returned by a tCBM could be affected by the over representation of an event type.

the column p-values represent the probability that two sets of reduction rate results for a given tCBM would be the statistically the same. Given two sets of reduction rate results, one from the balanced event sequences and another from the unbalanced event sequences, on the one hand, values close to 0 belong to two sets of reduction rate results with statistical difference. On the other hand, values closer to 1 appears when the two sets of reduction rate results have

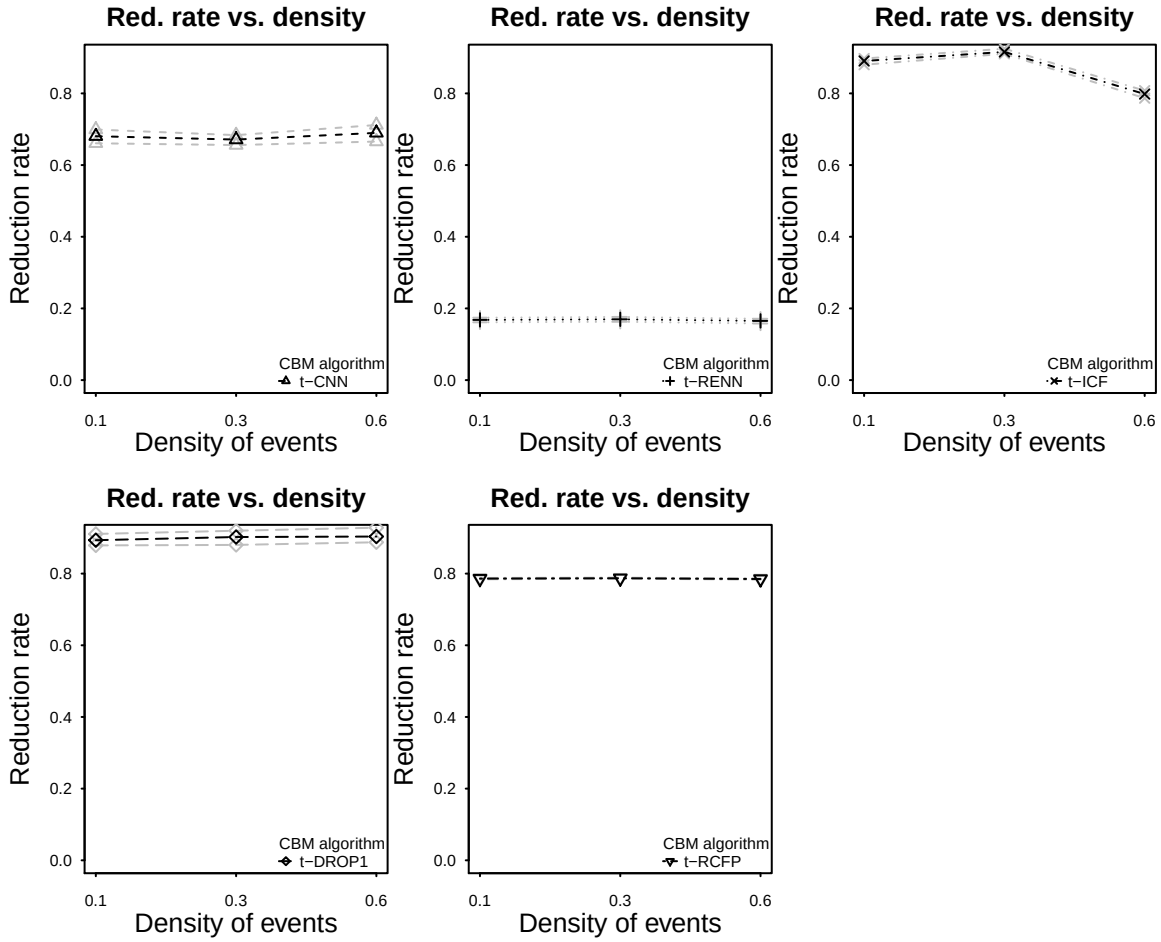


Fig. 5.11 Reduction rate with different case-bases and different density of events.

no statistical difference.

	Reduction rate average		
	p-value	Balanced	Unbalance
t-CNN	0.05	0.65	0.66
t-RENN	0.92	0.18	0.18
t-ICF	0.01	0.86	0.84
t-DROP1	0.80	0.81	0.81
t-RC-FP	0.00	0.78	0.78

Table 5.7 Probability of a reduction rate of a tCBM could be affected by the over representation of an event type.

Temporal CBM failing with maintenance task

Since tCBMs delete cases from the original temporal case-base, there is the possibility that the reduction performed affects the accuracy of the system. This maintenance task could fail

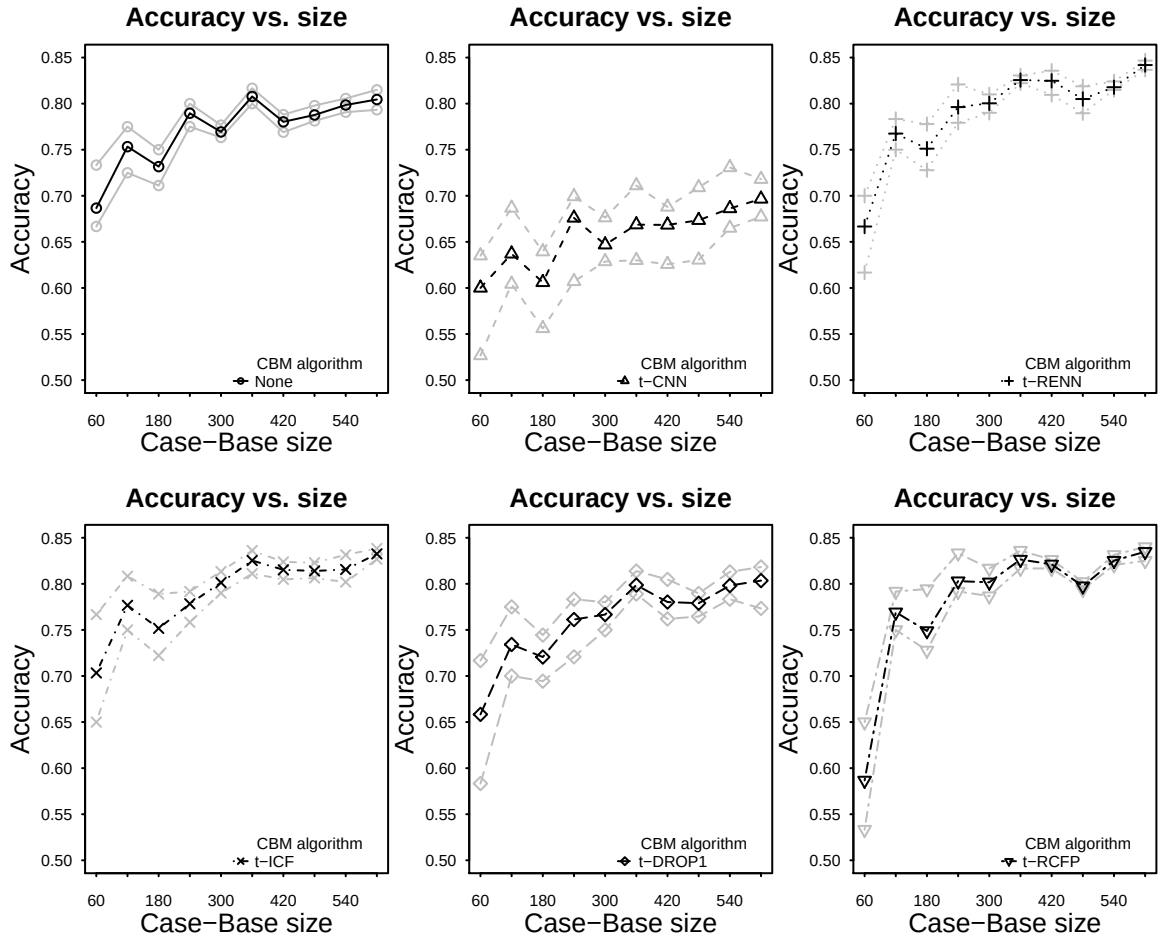


Fig. 5.12 Accuracy with the M&M temporal distance function with different synthetic temporal case-base with different sizes.

when the accuracy of the CBR system drop significantly. In order to observe whether a tCBM fails in the maintenance task, from all the accuracy results given by the previous experiments are considered, and each tCBM results are compared against the original temporal case-base. Table 5.8 shows the results of a t-test. In the column p-value appears the probability of a given tCBM returning a temporal case-base with worst accuracy than the original temporal case-base.

	p-value	None Accuracy avg.	Accuracy average
cnn	1.00	0.74	0.63
renn	0.00	0.74	0.79
icf	0.00	0.74	0.78
drop1	0.00	0.74	0.77
rcfp	0.00	0.74	0.80

Table 5.8 Probability of the accuracy given by a maintained temporal case-base is better than the original temporal case-base.

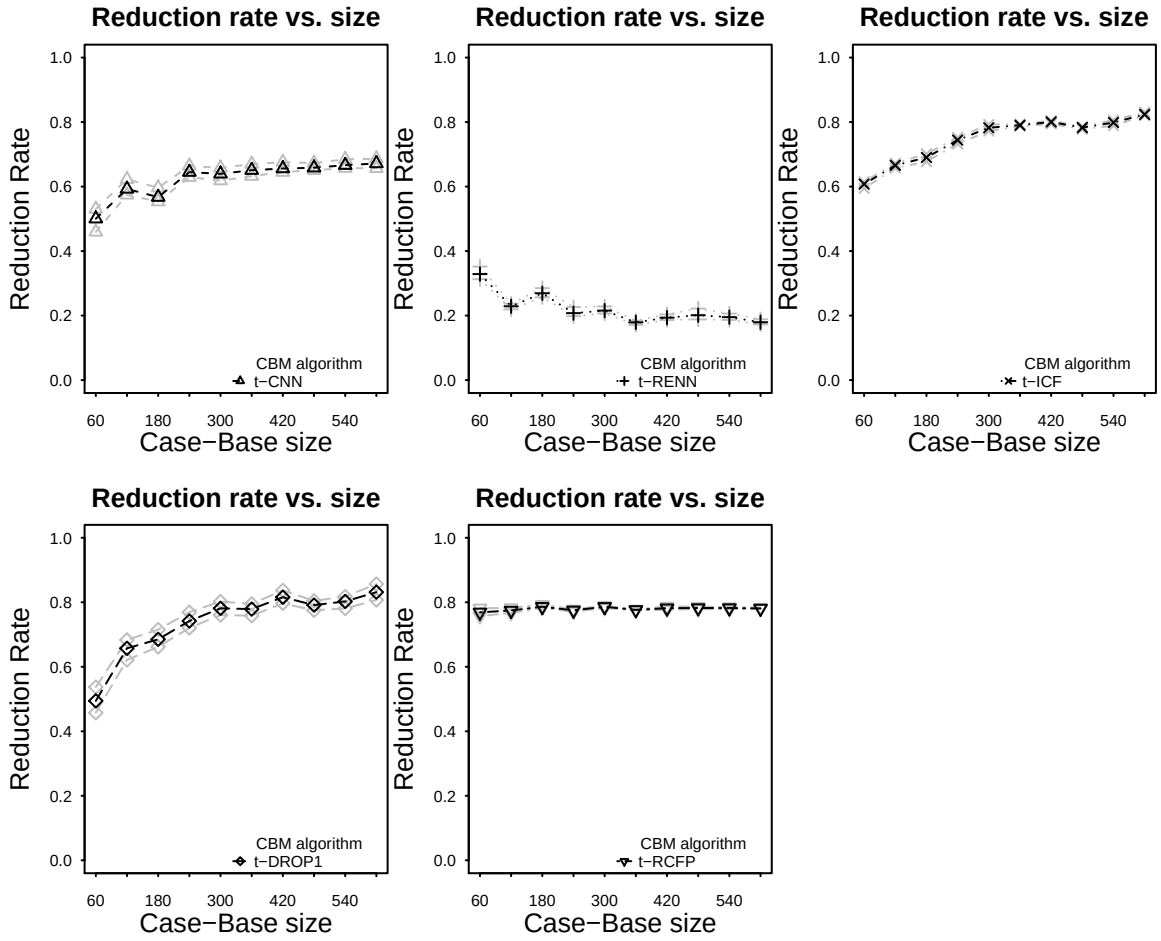


Fig. 5.13 Reduction rates with different case-bases and sizes with the M&M temporal distance function.

5.7 Discussion

Regarding the case-base size, from figure 5.6 can be seen than that it is a relevant factor for the tCBM. In fact, lower the case-base sizes implies higher variance of the accuracy results, even for the original temporal case-base, which could indicate that the temporal case-base is not covering the problem domain. However, despite most of the tCBMs could be used to improve the accuracy of the CBR system even with few cases, t-CNN fails in this purpose, returning temporal case-bases with lower accuracy than the original. Furthermore, from figure 5.7 the case-base size seems to be a relevant factor in the reduction rate of some tCBMs algorithms. While algorithms such as t-RENN and ICF are not affected, other algorithms are, such as t-CNN, t-DROP1 and t-RC-FP. In fact, these algorithms are affected up to certain case-base size, from which its reduction rate is stable, being lower this reduction with the smaller the case-bases. The results point out that other factors are the cause of the tCBM algorithms reduction ability, such as the proportion of redundant cases, or temporal factors related to the

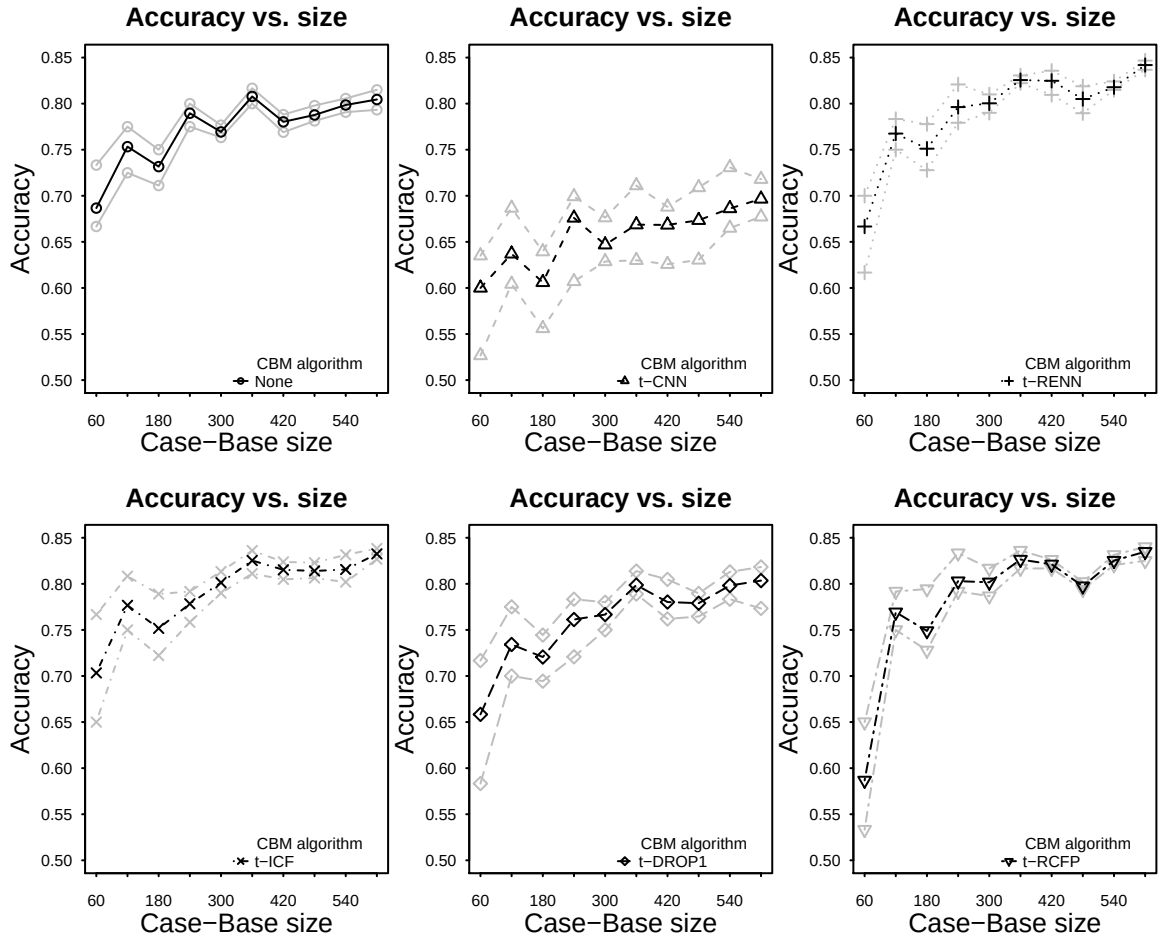


Fig. 5.14 Accuracy regarding the synthetic temporal case-base with the case-bases with over representation of an event type.

event sequences.

Figure 5.8 confirms that the number of redundant cases in the temporal case-base is another relevant factor in the creation of a good maintained case-case. In this sense, lower the amount of redundant cases then lower the accuracy that it is possible to achieve with the maintained case-base. Nonetheless, most of the tCBMs algorithms return maintained case-bases with better accuracies than the original case-base, with the exception of t-CNN.

According to figure 5.9, the number of redundant cases is a factor to be considered in the reduction ability of the algorithms. In this way, the algorithms with the purpose of deleting redundant cases, such as t-CNN, t-DROP1 and t-ICF, have greater reduction rates with case-bases with high redundancy levels, and lower reduction rates with low redundancy levels. Regarding t-RENN, which is a noise case remover, lower the number of redundant cases, then higher its reduction ability. It is remarkable the good reduction ability of t-RC-FP, with high reduction rate with every level of redundancy levels.

Regarding the temporal factors, figures 5.10 and 5.11 depict that the different densities of

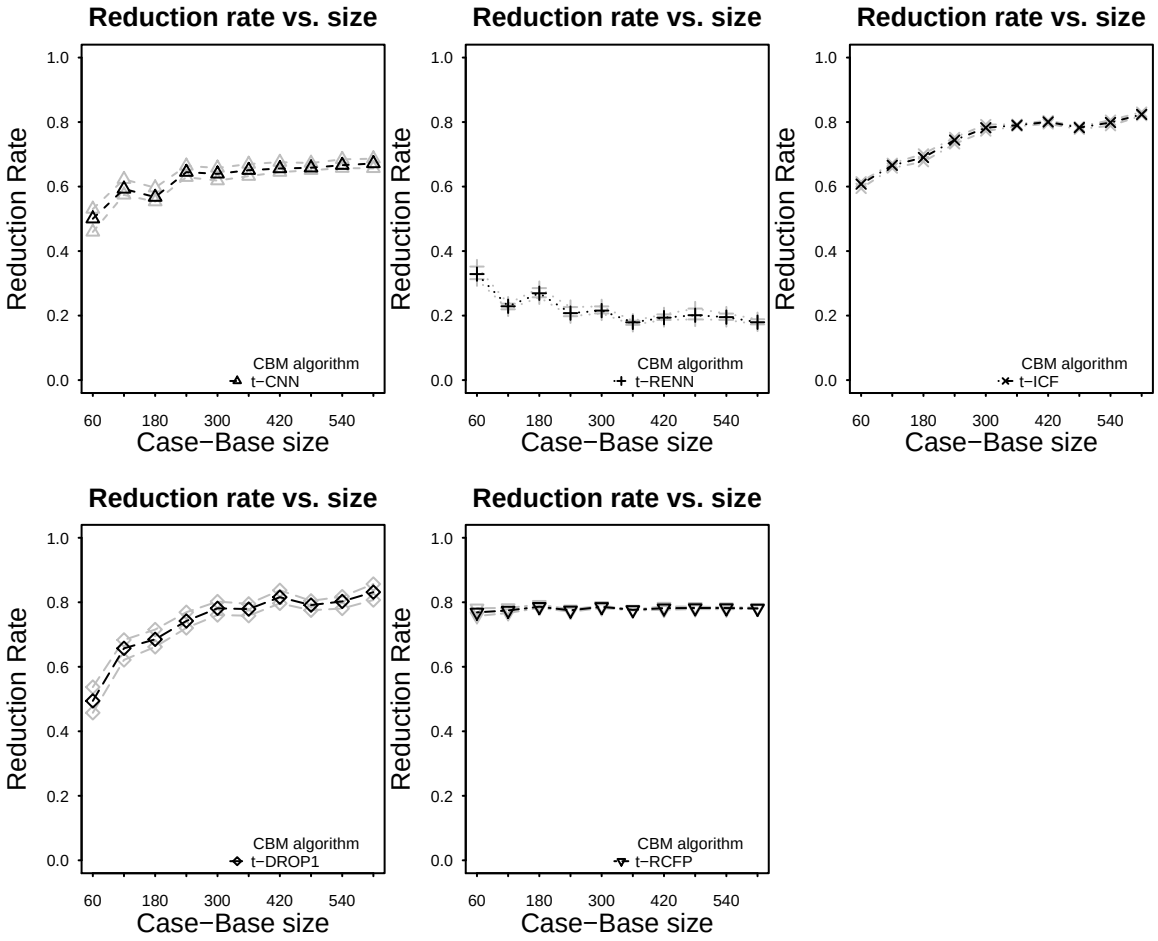


Fig. 5.15 Accuracy regarding the synthetic temporal case-base with the M&M temporal distance function.

the events is not a relevant factor to the creation of the maintained case-base, since the returned accuracies and reduction rates are not different from temporal case-bases with different density levels.

The experiments carried out confirm that the temporal distance functions is a relevant factor (see figures 5.12 and 5.13), and it should be taken into consideration to improve the problem solving ability of the CBR system. In fact, the experimental results depicted in tables 5.4 and 5.5, confirms that there are statistical differences between the results given by the CBR system using the Edit Distance and M&M distance functions, for both accuracy and reduction rate results.

Taking into consideration the over representation of an event type, comparing figures 5.14 and 5.6, they seem to be very similar and with no difference between them. However, the statistical study between the accuracy results given by both scenarios show that there is a statistical difference, although the distances between the accuracy averages of both experiments are quite similar to be taken into consideration. The same occurs for the reduction rates. Re-

sults shown in figures 5.15 and 5.7 are similar, but the statistical study from table 5.7 highlights that there are statistical differences for some algorithms, such as t-CNN, t-ICF and t-RC-FP. In this case, the representation of the event sequence could affect the reduction rate of the tCBM algorithms.

With regards to figure 5.6 and table 5.8, only the t-CNN algorithm creates maintained case-bases with lower problem solving ability than the original case-base, which is given to an excess of deletion of cases. This situation may be given to the deletion policy of the algorithm, which reduces significantly the size of the case-base in such a way that the temporal cases are not enough to cover the complete problem domain. Furthermore, the results of t-CNN are different from the rest due to its good results with other non-temporal case-bases [97–99]. Although its results may be driven by the particularity of the synthetic case-base used in this experiments.

5.8 Conclusions

In the present chapter, formal definitions of temporal cases using event sequences are given. Furthermore, because computing the similarity is a key aspect of a CBR system, two different temporal distance functions are used: the Edit Distance [104, 114] and M&M [173, 174].

Similarly with non-temporal case-bases, a CBR system using a temporal case-base could suffer from performance problems and worsening of its accuracy when the amount of stored cases is high. Thus, CBM algorithms can be used in order to reduce the size of the case-base and to improve the performance of the system. In particular, a set of temporal CBM algorithms have been proposed, one per each algorithm family and dimension, in order to perform the performance task with temporal case-bases: t-CNN, t-RENN, t-DROP1, t-ICF and t-RC-FP.

Several experiments have been carried out in order to study whether the proposed algorithms may be used with temporal case-bases. To this end, synthetic temporal case-bases have been created to study a set of factors that may affect the results of the CBM algorithm. These factors are classified into two categories: temporal and non-temporal factors. Whereas the non-temporal factors are the size of the case-base and the proportion of redundant cases, among the temporal factors are the density of events, the temporal distance function and the representation of the event sequence. From the experiments results given by the maintenance algorithms, the accuracy and reduction rate are completely dependent of the non-temporal factors. Despite of the fact that the temporal factors affect directly to the maintenance results, there are no statistical differences between the set of results between the different experimental results. Furthermore, most of the proposed algorithms, t-RENN, t-DROP1, t-ICF and t-RC-FP calculate maintained case-bases with better accuracies than the original case-base in all the experiments. That is, the maintenance task may create a maintained case-base with better ac-

curacy and lower number of cases. However, if the reduction is high then the results could be over-fitted to the evaluation. The only algorithm that seems to be failing in the maintenance task is t-CNN, which always obtains a case-base with lower accuracy than the original. In this way, more experiments need to be carried out with this algorithm in order to gather more evidence and to provide a final conclusion. Lastly, according to the experiments results, the adaptation of the existing proposed CBM algorithms is feasible in order to work with temporal case-bases. The next step is to make use of the proposed temporal CBMs with an Ambient Assisted Living system that make use of sensor network.

Algorithm 29 Temporal CB creator

Input: A case-base size $size \in \mathbb{Z}$, a density value $density \in \mathbb{R}^+$, a proportion of redundant cases $redundancy \in \mathbb{R}^+$, an event sequence length $length \in \mathbb{Z}$, a set of event types E, a set of solutions Ω, a table of parameters p, and a boolean value $b \in \mathbb{B}$ to create either a balanced or unbalanced temporal case-base.

Output: A temporal case-base TC

$TC \leftarrow \emptyset$
for $\omega \in \Omega$ **do**
  for $j = 1$ **to** $\frac{size}{|\Omega|}$ **do**
    $s \leftarrow <>$ {An empty event sequence}
    {number of events per event sequence}
    $n \leftarrow density * length$
    if balanced **then**
      for all $e \in E$ **do**
        {number of events per event type}
        $n^e \leftarrow \frac{density * length}{|E|}$
      end for
    end if
    {Iterate over the instants}
    for $t = 1$ **to** $length \wedge n > 0$ **do**
      for all $e \in E \wedge n > 0$ **do**
        if $n^e = 0 \wedge \neg balanced$ **then**
          continue
        end if
        {parameters for *Occurs* function}
        $t_1, t_2, t_3, t_4 \leftarrow p[\omega, e]$
        if $Occurs(e, t, t_1, t_2, t_3, t_4) >$ random number in $[0,1] \in \mathbb{R}$ **then**
          {Add new event}
          $s \leftarrow s + (e, t)$
          $n \leftarrow n - 1$
          if $balanced$ **then**
            $n^e \leftarrow n^e - 1$
          end if
        end if
      end for
    end for
    $TC \leftarrow TC \cup \{c = (s, \omega)\}$ {Add temporal case to temporal case-base}
  end for
end for
{Introduce noise cases from the redundancy parameter}
$noise \leftarrow size - redundancy * size$;
while $noise > 0$ **do**
  $c^x \leftarrow$ random case from TC not selected previously
  Change solution ω^w of c^x to another different solution $\omega^y \in \Omega$
  $noise \leftarrow noise - 1$
end while
return TC

Chapter 6

Using CBR to Detect Risk Scenarios of Elder People Living Alone at Home

The present chapter introduces a temporal CBR system for detecting domestic accidents that may lead to serious complications if the elderly resident is not attended quickly. In temporal CBR systems the cases are composed of event sequences. Several experiments have been conducted with different CBR system configurations in order to test this approach. Results from these experiments show that the proposed approach is able to detect unsafe scenarios. Furthermore, in order to test the proposed temporal CBM algorithms in chapter 5, the proposed CBR system is evaluated using their maintained case-bases. Results are similar to those given by the original case-base, and support our hypothesis that maintenance can be conducted successfully with temporal dependant domains.

This chapter has been developed with the support of Thomas Roth-Berghofer and Christian Sauer from the University of West London.

6.1 Introduction

According to the World Health Organization and the US National Institute of Ageing/Health [166], industrialized countries are facing the problem that the population's average age is drastically increasing. Information Technology and Artificial Intelligence (AI) may play a relevant role in looking after people living alone at home, as well as in providing care assistance. Examples of this key role are some policies such as the Ambient Assisted Living initiative promoted by the European Union [1]. In all of these initiatives, Smart Homes are encouraged as a tool to detect unsafe scenarios at home [19], as for instance falls, which are one of the major causes of serious accidents for the elderly people living alone.

Smart Home systems are usually based on agent architectures [32], where each agent is responsible of one particular task, such as the control of the home environment [130], the assistance of home inhabitants [87, 88, 156] or monitoring of residents' health status [30]. In order to fulfill their purpose, Smart Homes require a set of sensors to gather home data and deliver them to agents. To this end, different types of sensors have been considered, ranging from intrusive devices such as cameras or wearable sensors [31, 156], to pervasive approaches such as movement or pressures devices connected by wireless sensor networks [6, 136]. Once data is collected and processed, AI techniques can carry out some inference processes in order to interpret the scenario based on the processed data.

The use of CBR in an agent in Smart Home systems has several advantages [32, 91]. First, the learning process is implicit in the CBR cycle, so agents using CBR can learn from specific situations as time goes by. Thus, the system adapts itself to the resident's specific needs. Second, the response time can be reduced because CBR avoids resolving already solved problems, which may involve a great amount of information and computation in a Smart Home environ-

[1] http://www.aal-europe.eu/

ment. Third, an expert can define personalised cases to represent a customised problem and its solutions. Finally, as CBR systems use similar past solved cases to solve a current problem, these cases can be used to provide explanations on why a concrete solution is proposed [39, 150].

Some authors have used CBR to solve problems in Smart Homes [12, 91]. However, to our understanding, little attention has been paid to the temporal dimension in the development of these CBR systems, since the use of the time dimension is limited to determine the context of the case [101].

We believe the techniques presented in previous chapters are suitable to solve some of the problems aforementioned. Therefore, in this chapter we propose a CBR system able to detect potential unsafe scenarios in a Smart Home, as for instance falls, using a spatial-temporal approach. This CBR system is based on the retrieval of previous cases which represent the different locations visited by the elderly resident during one of their daily activities. These cases are represented by event sequences, where each event consists of a location and a timestamp, and the type of scenario detected. The proposed CBR system has been designed for *proDIA* monitoring system [15]. This commercial system consists of a network of pervasive sensors, such as motion detection infra-red sensors, pressure sensors (located in bed, chairs, sofas, etc) and magnetic sensors to detect door opening and closing. A prototype of this system has been placed in 100 houses in the Region of Murcia, Spain.

However, a CBR system may suffer of different problems. Retaining many cases in the case-base may imply an increment of the retrieval time of the similar cases. This problem is relevant because the equipment placed at home usually is based on low profile hardware with low computing power. Consequently, in order to ease this problem we are interested on the effects and consequences of using a temporal Case-Base Maintenance algorithm (tCBM) in this system. The benefits of its use are the reduction of the case-base size, making interpretable the explanation of the system outcome and enhancing the performance of the system at the same time.

This chapter is organized as follows: in section 6.2 a review about Smart Homes and the uses of AI techniques are presented. Section 6.3 presents the architecture of the CBR system and how the spatio-temporal representation is modelled within cases. Section 6.4 shows the experiments conducted to determine the best configuration of the proposed system as well as observing the consequences and results of using tCBM algorithms. Finally, discussions and the conclusions are provided in sections 6.5 and 6.6, respectively.

6.2 Smart Homes and AI

Smart Homes are meant to ease the activities of daily living of the people at home. In fact, this goal is so wide that the term *smart home* can be used in many different systems with different purposes: the automation and control of home environment [130], assistance of independent elderly people [64, 87, 88, 124, 136, 152, 156], monitoring of elderly health status [13, 30], or improving the leisure activities at home [152].

Moreover, examples of the importance of this technology are the development of many research initiatives, such as CASAS project [28], MavHome [29], MIT's PlaceLab [69], CARE project [88], the Aware Home [80], Georgia Tech Aware Home [64] or H-SAUDE [30].

The implementation of Smart Homes is traditionally based on agents [32], where each agent is responsible of one particular task, for instance, the monitoring of activities of daily living. In order to fulfil their purpose, each agent uses a set of devices or sensors to gather the required data from the environment. Usually, these devices are nodes of a wireless sensor network [6], such as video cameras [156], pervasive [136] or wearable sensors [31, 163]. Despite of the fact that these type of sensors can be used to fulfil different objectives of Smart Homes, each device type offers a set of advantages and disadvantages [24, 44].

6.2.1 Recognizing activities of daily living in a Smart Home

Nowadays, it is possible to record of the actions taken at home making use of cheap sensors infrastructure [44]. A lot of effort has been put into developing systems able to detect Activities of Daily Living (hereinafter ADL) [31, 80, 87, 88, 103, 124, 163]. ADL detection makes possible to discover people's behavioural patterns and assist them at home.

To detect the people's behavioural patterns with Smart Home systems [13], many projects have been developed with the purpose of recognizing Activities of Daily Living (hereinafter ADL) in order to provide assistance to people at home. These projects have in common the use of cheap sensors to record the actions taken at home. These sensors are usually as pervasive as possible [152], and among the most common there are binary sensors (such as infrared, door-installed and pressure sensors) [87, 88, 124, 124], RFID tags [31, 163] (either as bracelets or hidden in the clothes), and even GPS devices in combination with accelerometers [103]. Sensors to detect smoke and fire are also in use for the sake of the elderly safety [124]. Video cameras are also used, but they could comprise the people's privacy [22]. Alternative approaches proposes the uses of vibration sensors to be placed on the floor to track the movement of the people at home [80]. The number of sensors to be placed at home is not a simple decision, although the tendency is to install as few as possible [154].

These types of system are characterised by their spatial and temporal features. While the system is aware about the current person location at home, it keeps a record of the previous lo-

cation. From that information, the system can infer what activity is undergoing at home. There are different approaches to represent the data collected by sensors to be used by the system, as for instance time series or event sequences [52]. Alternatively, to enhance the understanding of the collected data, complex data models can be built from the information provided by sensors. For example, Markov based models can be built from time series generated by sensors [54]. The main differences between time series and event sequences are the frequency in which the data from sensors are gathered and the heterogeneity of the collected data. On the one hand, time series collect data from one sensor with a given frequency. In case that several sensors were deployed, the system has to deal with several time series, one per sensor. On the other hand, event sequences collect data without fixed frequency, and they are able to keep record of different sensor data in a common data structure. Lastly, whereas event sequences usually have lower data space requirements than times series, the operations required in event sequence processing are more complex than those with time series.

Figure 6.1 portrays an example of how several binary sensors can be deployed within a house for ADL monitoring. Sensors of this type only sends two values, either *true* or *false*, to indicate whether it has been activated or not. Thus, movement sensors will activate a binary signal when an object movement is detected within their range. At the same time, pressure sensors will activate a binary signal when they are pressed.

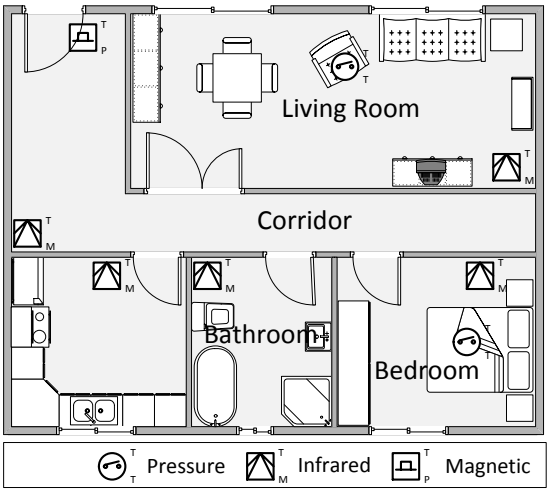


Fig. 6.1 Smart home and arranged sensors.

6.2.2 Smart Homes and CBR

CBR has been used in Smart Home systems for different purposes. In [171], a CBR system is used to assist where to place sensors in a Smart Home. The decision is done according to the

resident's physical disabilities, such as their cognitive abilities, mobility, dexterity and other personal details. However, the system does not consider the temporal information to reflect the change in a resident's physical state over the time.

A CBR architecture for Smart Home to enhance the inhabitants comfort at home is proposed in [101]. In this approach, cases comprise actions occurring at home and how the Smart Home reacts to them to enhance comfort. For instance, lowering the AC temperature or adjusting the light brightness in a room. The case structure is a frame with slots for representing the user information, data gathered from the sensors, and a time-stamp to represent when the observation was made.

In [91], a CBR system is used to detect problems at a home and to propose actions to amend them. However, most of the cases' descriptions do not include the spatial-temporal data of the actions taken at home. According to [91], the success of the system relies on the quality of the retained cases in the case-base. Thus, the use of good case engineering practices are recommended not only to create the first set of cases, but also to create cases personalised to each specific user. Furthermore, since the case learning task is included in the CBR cycle, the case-base size increases with time, making the inclusion of case-base maintenance policies necessary to maintain the quality and performance of the case-base.

The core of the Smart Home proposed in [32] is a CBR system. This system is placed in a residential ward where nursery staff provides care of patients. The purpose of the CBR system is to plan the tasks to be done by the staff. The case representation keeps data about the patient's health status, details of one task already done by a staff member and what is the next task to be performed. Besides, each task has a priority level and is associated with a beginning and ending time-stamp.

AmICREEK uses a CBR system to detect the situation taking place within the system's environment [83, 84]. AmICREEK is based upon a three layer architecture [15]: the *perception* layer as the middle-ware gathering the information from the sensor network, the *awareness* layer is a CBR system that detects the context in which the action is taking place and the goal of the action. Lastly, there is the *sensitivity* layer, in which a sequence of tasks is built in order to satisfy the goal given by the CBR system according to the system's context. The authors tested this approach in a hospital ward but the cases only contained one time-stamp to represent the time in which they were created.

6.3 A Smart Home for alarm detection

Our proposal of CBR system has been designed for being used within the commercial system *proDIA*. The architecture of the *proDIA* system is built upon three levels: *sensor level*, the *communication level* and the *data processing level*. Figure 6.2 depicts *proDIA* three level

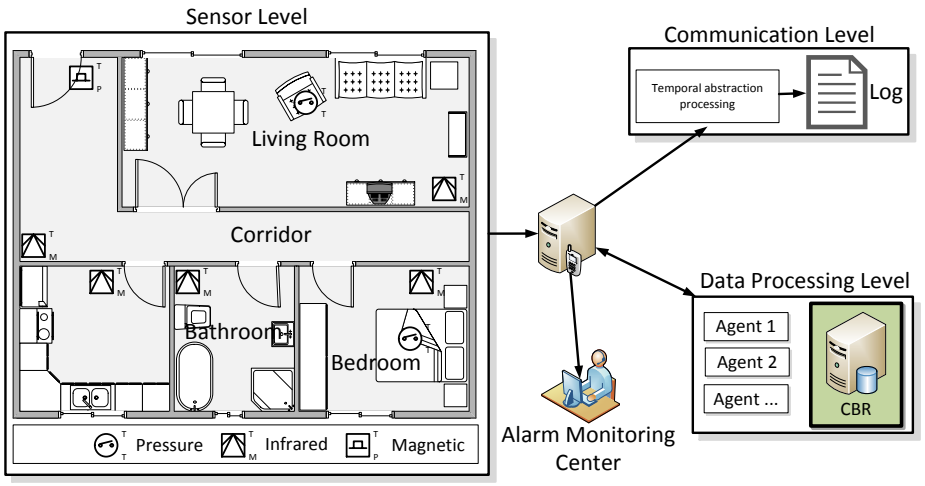


Fig. 6.2 *proDIA* three level architecture and example of sensors deployment.

architecture and an example of different sensor deployment.

The *sensor level* manages the sensor-data acquisition from the wireless sensor network. This network uses three types of sensors: infrared motion detection sensors, pressure sensors and magnetic sensors to detect whether the main door is opened or closed. The basic configuration of the system includes the placement of motion sensors in every location, and one single magnetic sensor to detect whether the main door is opened or closed. Furthermore, pressure sensors may be located in places such as sofas and beds to detect whether the person is resting or lying on one of them. With these pressure sensors, detecting the location of the person is possible, even if the person is not moving.

The second architecture level is the *communication level*, where the data provided by the sensor level is recorded. Using the IEEE-802.15.4 communication standard, the data gathered by the sensors is kept in a home-station (mini-PC), which synchronises the data sent by the sensors according to the timestamps in which they are received. The communication level creates a log of the data with a given frequency, which is used by the *data processing level*. The *log* is a comma separated values (CSV) file, where each line contains information related to the readings sent by the sensors, such as the identifier of the sensor, the time-stamp in which the sensor is activated, the location of the sensor and the content of the sensor's reading. Every newly created log starts empty, so the frequency in which the log is created will determine the amount of data stored in it. Therefore, if long observation periods of the home are required, the log must register data using a low frequency of collection of data. On the contrary, log files created with a high frequency contain less data since they represent short observation periods. For instance, table 6.1 shows a log file representing the data sent by sensors when the users arrive at home.

Starting from the information provided by the communication level, the *data processing*

Time-stamp	sensor id.	Location	Message	Rationale/Justification
7932	sensor197680	Corridor	MOVE: true	getting house
7932	sensor197683	Door	isOPENDOOR: true	
7940	sensor197683	Corridor	isOPENDOOR: false	
7959	sensor347050	Bedroom	MOVE: true	going to the bedroom
7972	sensor530111	Bedroom	PRESS: true	sitting down on the bed
7980	sensor197680	Corridor	MOVE: false	
8054	sensor530111	Bedroom	PRESS: off	standing up
8113	sensor197680	Corridor	MOVE: true	going to the toilet
8121	sensor197680	Bedroom	MOVE: false	
8122	sensor536770	Bathroom	MOVE: true	getting in the bathroom

Table 6.1 Example of activity log.

level attempts to infer the state in which the elderly is, that is, the complete situation in accordance to the situation context is described. The system relies on the assumption that the location of the resident, the activity or absence of it, and the moment of the day in which these facts are registered are enough to detect possible emergency situations. For example, if the resident suffers a fall and, consequently, loses their consciousness, or broke a bone in such a way that prevents them from moving, detection of this situation is based on an excessive time of inactivity being measured in a context in which this is abnormal (i.e. the attendee is in the house and she is not supposed to be resting or sleeping). To this end, a behavioural model was developed, based on a finite state automaton. Once an abnormal situation is detected, the system sends an alarm to the Alarm Monitoring Centre, where a specific predefined protocol of action is triggered.

6.3.1 CBR system configuration

We propose to include CBR in the *proDIA* system to check whether the daily activity at home is normal or abnormal, indicating that an unsafe scenario is taking place. To this end, the approach followed is to keep a record of the movement of the resident at home within given time-frame. The CBR system checks whether a current activity, or event sequence, is similar to previously recorded activities/event sequences.

According to the definition of case given in [1], a case consists of a problem and a solution. In order to classify and detect unsafe situations at home, the cases represent a daily activity and its type. Thus, whereas the problem represents the visited locations during one daily activity, the solution is a label describing the type of activity (called scenario). The set of valid values for the solution is not limited, being possible the inclusion of new *solution* labels on demand when the CBR is running.

The event sequences consist of heterogeneous events ordered in time, where each event is composed of an event type and a time-stamp that represents when the event occurs [104, 114]. Thus, each event in the sequence is a tuple composed of the location visited by the person and the corresponding time-stamp.

The following expressions are provided to detail the case representation (see expression 6.1) and the event sequence (see expression 6.2).

$$\text{case } c = (sequence, solution) \tag{6.1}$$

$$solution \in \{normal, scenario_1, scenario_2, \ldots\}$$

$$\begin{aligned} sequence = \langle &(loc_1, t_1), \ldots, (loc_2, t_2), (loc_n, t_n) \rangle \mid \\ &\mid \forall loc \in \{Corridor, Bedroom, \ldots\} \wedge \\ &\wedge \ (t_i, t_{i+1} \in \mathbb{N}^+ \wedge t_i \leq t_{i+1}) \end{aligned} \tag{6.2}$$

Given a log, the main steps of the CBR system are the following:

1) The CBR system reads the log when a new one is created. This log contains the data from sensors chronologically ordered according to their time-stamps.

2) An event sequence is built from the collected sensor data. Later, this event sequence is used as a *input* query to the retrieval step.

3) The CBR system retrieves from its case-base those cases with the most similar event sequence to the *input*.

4) According to the retrieved cases, the system tries to infer the type of the activity that best matched the *input*.

5) When the activity is classified as an abnormal scenario (according to the predefined solution labels), the system sends a message to the Alarm Monitoring Centre. Then, the expert decides which action is the most suitable to the scenario according to pre-established protocols.

6) Finally, a new case is retained in the case-base when an *abnormal* scenario is classified correctly.

6.3.2 Temporal distance between cases

The CBR system uses the edit distance between event sequences proposed in [104, 114]. This distance measure calculates the *cost* of transforming an event sequence into another. This cost is represented as the number of operations needed to perform the transformation. The operations are applied on the query event sequence, until it matches with the retrieved event sequence, which is known as a pattern. Therefore, a high number of transformation operations

stand for two not very similar event sequences. On the contrary, two similar event sequences need few transformation operations. The set of available operations are *insertion*, *deletion* and *displacement*. While the insertion is used when the query does not contain an event that is present in the pattern, the deletion is used if the query contains an event not appearing in the pattern. The displacement operation is used when two events in both the query and pattern match the event type (that is, they refer to same home location). Whereas insertion and deletion operations have an associated constant cost, displacement operation cost depends on the distance between the time-stamps of the events types implied in the displacement. In order to ensure the proper operation of the edit distance, the displacement operation costs has to be lower than the insertion and deletion operations cost. Thus, the difference between the time-stamps is normalized between 0 and 1, and the cost of the insertion and deletion operations is set to values higher or equal to 1. Algorithm 30 presents a dynamic programming approach for searching the minimum number of operations to transform a query into a pattern.

Algorithm 30 Edit distance between two event sequences x, y [104, 114]

Input: Two event sequences $x = \langle (loc_1^x, t_1^x), \ldots, (loc_n^x, t_n^x) \rangle$ and $y = \langle (loc_1^y, t_1^y), \ldots, (loc_m^y, t_m^y) \rangle$, with $loc \in \{Bedroom, Corridor, \ldots\}$, the costs $w(loc^x), w(loc^y)$ of the *insertion* and *deletion* operations.

Output: Edit distance between the two given sequences.

1: $r \leftarrow$ matrix of $n \times m$ dimensions
2: $r(0,0) \leftarrow 0$
3: **for** $i \leftarrow 0$ **to** m **do**
4: $\quad r(i,0) \leftarrow r(i-1,0) + w(loc^x)$
5: **end for**
6: **for** $j \leftarrow 0$ **to** n **do**
7: $\quad r(0,j) \leftarrow r(0,j-1) + w(loc^y)$
8: **end for**
9: **for** $i \leftarrow 1$ **to** m **do**
10: $\quad$ **for** $j \leftarrow 1$ **to** n **do**
11: $\quad\quad update_x \leftarrow r(i-1,j) + w(loc^x)$
12: $\quad\quad update_y \leftarrow r(i,j-1) + w(loc^y)$
13: $\quad\quad align \leftarrow r(i-1,j-1)$
14: $\quad\quad$ **if** $loc^x = loc^y$ **then**
15: $\quad\quad\quad align \leftarrow align + (\frac{|t_i^x - t_j^y|}{max(t) - min(t)})$
16: $\quad\quad$ **else**
17: $\quad\quad\quad align \leftarrow align + w(loc^x) + w(loc^y)$
18: $\quad\quad$ **end if**
19: $\quad\quad r(i,j) \leftarrow min(update_x, update_y, align)$
20: $\quad$ **end for**
21: **end for**
22: **return** $r(n,m)$

6.3.3 Alarm scenarios analysis

Four different scenarios have been considered based on different common scenarios that usually occur at home: a *normal* daily activity, a *bad night* due to sickness, a fall and a later loss of consciousness, and last, a fall in which the person remains conscious and crawl to find help. That is, cases have a particular solution identifying the type of scenario that is represented by their event sequence.

A normal scenario is understood as a daily routine activity, in which nothing unexpected

happens. For instance, figure 6.3 depicts in a time-line the different events that compose the event sequence of a normal scenario case within a time window of 24 hours duration. The different numbers represent the location at home (Corridor=0, Kitchen=1, Living Room=2, Toilet = 3, Bedroom = 4, Out = 5), and the night and middle of the day hours are represented with a darker and the brighter background colour, respectively. For the sake of clarity, the corridor locations are portrayed with black triangles.

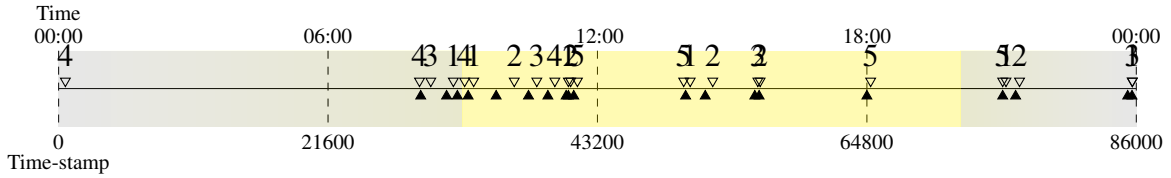


Fig. 6.3 Case representing 24 hours of a normal day. The codes represent locations (Corridor=0, Kitchen=1, Living Room=2, Toilet = 3, Bedroom = 4, Out = 5)

Another example of case representing a normal scenarios is depicted in figure 6.4. This occasion the case contains an event sequence covering 6 hours of activity at home.

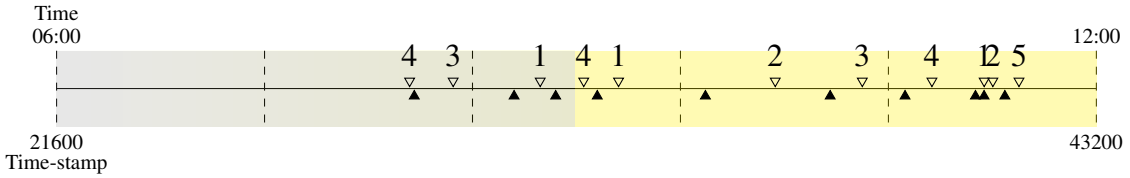


Fig. 6.4 Case representing 6 hours of duration of a normal day.

The *bad night* template represents the locations visited when the resident has not been able to sleep due to a sickness status. In this scenario, the event sequence represents regular visits to the bathroom during the night. Figure 6.5 portrays an example of a bad night scenario. When compared with figure 6.3, it can be seen that from midnight to 8:00AM, the person has frequently gone to the bathroom during the night, and the rest of the day is similar to a normal day with a few variations, such as getting up late from bed.

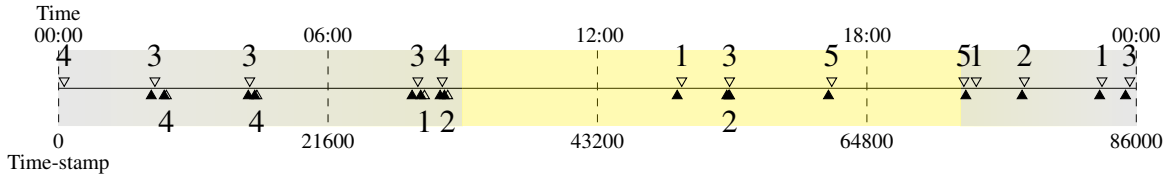


Fig. 6.5 Case representing a resident's bad night within a case of 24 hours duration.

Fall scenarios describe two types of falls: a fall where the person stays motionless after losing consciousness and another where the person stays conscious and may crawl on the

floor. Both scenarios usually occur in the bathroom as a consequence of a fall in the bathtub and the difference between them is the activity of the location visited.

After a fall, a conscious person tries to move or crawl to other different location to call for help. In this scenario, since the movement within the house is slow, in a given time two movement sensors in different locations detect activity, alternating the records of the resident location. For instance, figure 6.6 shows a case of 12 hours duration for a scenario of fall without loss of consciousness. Once the fall occurs the person crawls on the floor moving forward slowly. When the person reaches a door, the movement sensors of the current and next locations start sending activation signals at the same time, until the person moves definitively to the next room.

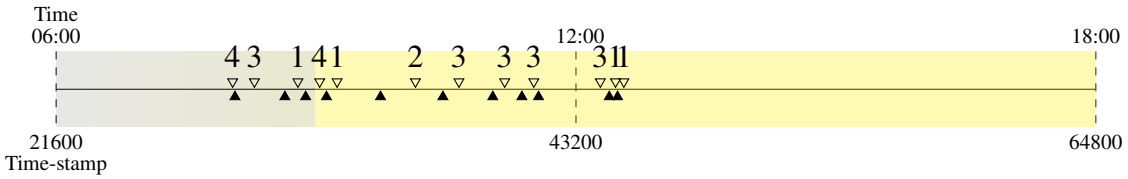


Fig. 6.6 Case of 12 hours duration representing a fall scenario where resident tries to ask for help.

On the contrary, if a person loss their consciousness, no movement sensor will be activated. Figure 6.7 represents a case of this scenario. Once the person suffers the fall, no sensors detect any activity any more. When comparing figure 6.7 with the normal scenario in figure 6.3, it is infrequent to have a long period of inactivity at home when the person is in the bathroom.

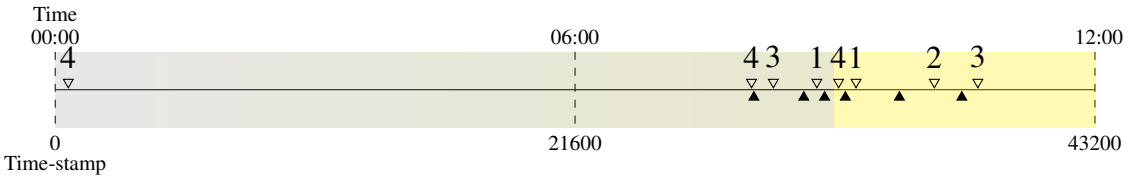


Fig. 6.7 Case of 12 hour duration representing a fall scenario that provokes a loss of consciousness to the resident.

6.4 Experiments

In this section, experiments to evaluate the suitability of techniques presented in chapter 5 are conducted. Therefore, these experiments has been designed to study whether a CBR system is able to detect the scenarios defined in the previous section 6.3.3 using both original and maintained case-bases.

In performed experiments, different synthetically generated case-bases have been used. The CBR system retrieves the most similar case using 1-NN strategy. This number of neighbours is chosen since a low number of abnormal cases are expected to be retained in the case-base. What is more, since a single case is retrieved by each query, no adaptation process is performed. That is, the solution of the retrieved case is the outcome of the system, describing the type of scenario that is taking place.

The chosen temporal similarity function is the temporal Edit distance (see algorithm 30). Consequently, owing to the fact that the case problem description comprises only an event sequence, the global similarity function is determined by the chosen temporal similarity function.

In the experiments, t-CNN, t-RENN, t-ICF, t-DROP1 and t-RCFP algorithms have been applied to construct the maintained case-bases. Details of these algorithms can be found in chapter 5.

In all the conducted experiments, the accuracy of the system, the true positive rates of the different scenarios, and the false positives rates of the system are calculated. These statistics will allow us to discuss what particular scenarios the CBR system is able to detect.

The rest of this section is structured as follows. Firstly, the generation of the synthetic case-base is explained to ease the understanding of cases problem description. Secondly, an experiment is conducted in order to tune different parameters such as the maximum duration of the event sequences and the required number of cases to detect the proposed abnormal scenarios. Finally, using the best CBR system configuration, a set of experiments are carried out to study how the tCBM algorithm may affect the capabilities of the CBR system in detecting the abnormal scenarios.

6.4.1 Generating a synthetic temporal case-base

In this research, synthetic temporal case-bases have been used. The main reason to use artificial data instead of real life logs is to keep control of the different existing scenarios at home, and being able to study if the CBR system is able to detect them.

The algorithm for temporal cases generation can be easily explained by the workflow schema presented in figure 6.8. The different locations of the resident in a normal day are represented by the workflow tasks. Given the size of the case-base, a beginning date, and the duration of the event sequence, the algorithm creates the event sequence as follows:

1. The tasks in a grey box produce an event occurring approximately at the given time, regardless the duration of the previous task.

2. Each task in white background produces an event with a duration approximately to the indicated. During that time, no other locations are visited by the person. Every task

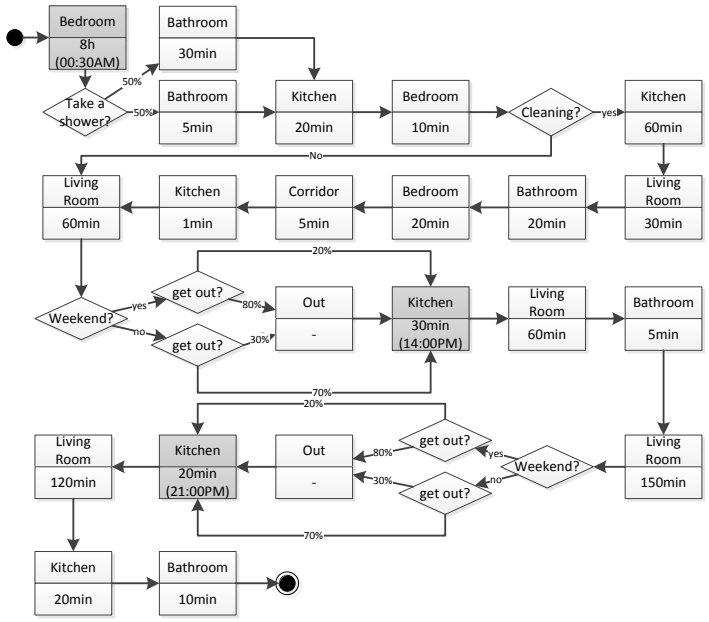


Fig. 6.8 Workflow of activities for a normal behaviour at home.

begins exactly when the previous task ends.

3. Next task is randomly taken according to a pre-established probability distribution in decision nodes. When the decision nodes implies weekend days, the next task is chosen accordingly to the defined day. That is, the task in the branch with answer *yes* is taken when the given date corresponds to any of weekend days.

4. When the difference of time between the first and last events in the event sequence is bigger than the indicated duration of the event sequence, a new case is created and labelled with the corresponding scenario description. For instance, if the duration of a case is established in 12 hours, then two cases will be produced from the diagram of locations in figure 6.8.

Regarding the proportion of the different scenarios in the case-base, the 90% of the retained cases represents normal daily activities. In order to avoid the discrimination of abnormal scenarios, the 10% of the remaining cases is equally shared by the abnormal cases. These proportions between normal and abnormal scenarios are chosen since abnormal scenarios appear less frequently than normal scenarios in daily life.

6.4.2 Study of the temporal case-base

In this section, the effect of (1) the duration of the event sequences within the cases; and (2) the size of the case-base, are analysed. Therefore, the values of both parameters (case-base size and case durations) are tuned to study its effect on the CBR system. The proposed case durations are 6, 12, 18 and 24 hours. Therefore, sort and long observations correspond to high and low frequent readings of the log, respectively. The cases in the case-base representing normal and bad night activities cover up to 24 hours of activity at home, and the rest of the cases represent normal activities until the unsafe scenario occurs.

Figure 6.9 shows the results of the conducted experiments. For each experiment, the accuracy of the system is observed, as well as the false positive rate regarding the normal behaviour. Additionally, the true positives rates for the different behaviours are recorded to study if the CBR system succeeds to detect them. Note that the false positive rate means the proportion of unsafe scenarios that were classified as normal, which must be avoided because this type of misclassification is the most dangerous one for the residents safety.

Based on the results shown in figure 6.9, the following observations are made:

- The accuracy is stable in all the scenarios when the case-base size is over 50 cases. However, a CBR system using short observation periods has a lower accuracy than a CBR system using longer observation periods. In fact, the experiments where the observation period spanned up to 24 hours gets accuracy values close to 1, even with small case-bases.

- Retaining a large amount of cases decreases the ratio of false positives for the normal behaviour. That is, increasing the number of stored cases means a decrease of the number of times in which the CBR system miss-classifies the behaviour as normal when in fact there is an abnormal situation at the resident's home.

- The true positives rate for the normal activities is directly proportional to the accuracy of the system, which can be explained due to the predominance of this type in the case-base (90% of the cases). Nonetheless, when the observations are frequently acquired, the CBR system is increasingly unable to detect the normal activities, which increases the ratio of false alarms. That is, the system is permanently classifying the abnormal situations as normal.

- Regarding the true positive rate of bad night scenarios, in all the experiments an increment of the case-base size will ease correct classification of these scenarios, except for the shortest observations, which never classifies this activity correctly. In fact, CBR system using longer observation periods classify the bad night scenarios correctly more frequently than those using shorter observation periods.

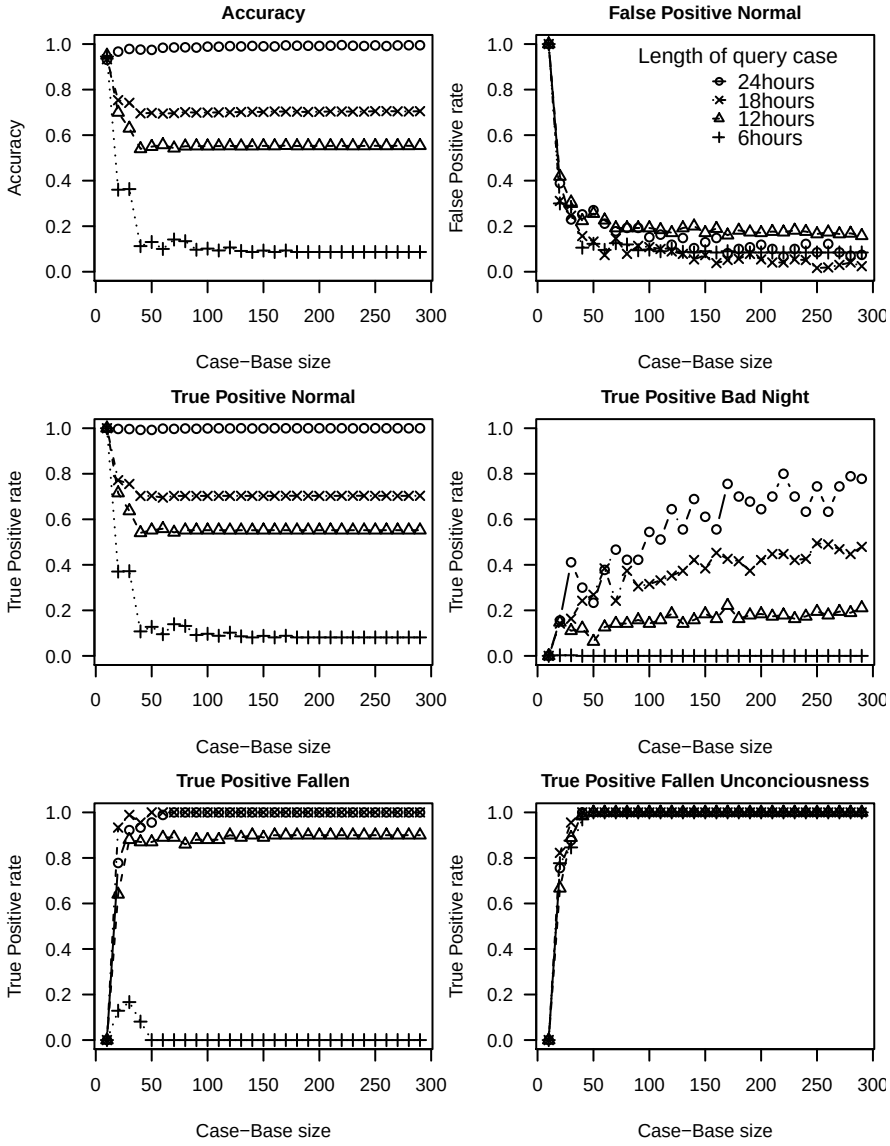


Fig. 6.9 Evaluation of the CBR system varying the case-base size and event sequence length.

- The detection of falls is possible in most observation lengths, except for the shortest observation periods, which fail to detect falls without loss of consciousness. Notwithstanding, longer observation periods are not suitable for the required quick response in the fall scenario.

It is worth mentioning that a CBR system can detect unsafe scenarios at home, although some considerations need to be addressed before using a CBR system in a real deployment. The observation of the environment needs to be done with short and long intervals in order to get a quick response to dangerous situations, such as falls, and to detect long activities. Finally, since new cases are added constantly to the case-base, there is a risk of a decreasing performance of the CBR system. This problem may be solved with the application of an appropriate CBM task.

6.4.3 Results without maintenance

According the previous section 6.4.2, the proposed CBR system retains cases with long event sequences and short event sequences at the same time. Thus, the log is read every 6, 12 and 24 hours, resulting in cases with a maximum duration equivalent to these values. For instance, given one day of observation of the home, the log will be read up to four times creating four cases of six hours of duration, two cases covering 12 hours of activity and finally one case of 24 hours of duration.

To evaluate the capability of the system detecting abnormal scenarios, $\alpha\beta$ evaluation method is conducted 10 times with $\alpha = 10$. Later, the results will be compared to those given by the CBR system using some type of temporal maintenance algorithm. Figure 6.10 shows the results of the original CBR system configuration. The shaded region belongs to the area between the maximum and minimum observed value with the corresponding case-base size.

Figure 6.10 highlights that the CBR system is able to detect every type of abnormal scenario if there are enough cases in the case-base. Furthermore, the following observations can be done:

- Despite the accuracy and the true positive rate of the normal scenario are almost 1 even with few cases in the case-base, the true positives and false positives scores need larger case-base sizes to increment their values up 1 or 0, respectively. Furthermore, with the exception of true positive of normal scenarios, the rest of the scenarios have larger variation on the results in the smaller case-bases.

- The false positive rate suffers from few variability and is stable below 0.1 in case-bases bigger than 500. The system is able to detect a bad night scenario when many cases are retained in the case-base. Besides, the variability of the true positive rate in detecting a bad nigh is significantly large, resulting in a CBR system not accurate in detecting this scenario.

- Finally, both falls scenarios are detected with larger cases-bases, but the variability of results in smaller case-bases is quite high.

Accordingly to the aforementioned results, a CBR system using larger case-bases is more likely to detect scenarios occurring at home. However, the performance of the system may decline as they are stored in the case-base. This can be solved using a tCBM to reduce the case-base size.

6.4.4 Results using tCBMs algorithms

The objective of this section is to analyses the effects of temporal CBM algorithm in the CBR system performance. That is, we want to determine if temporal CBM algorithms are able

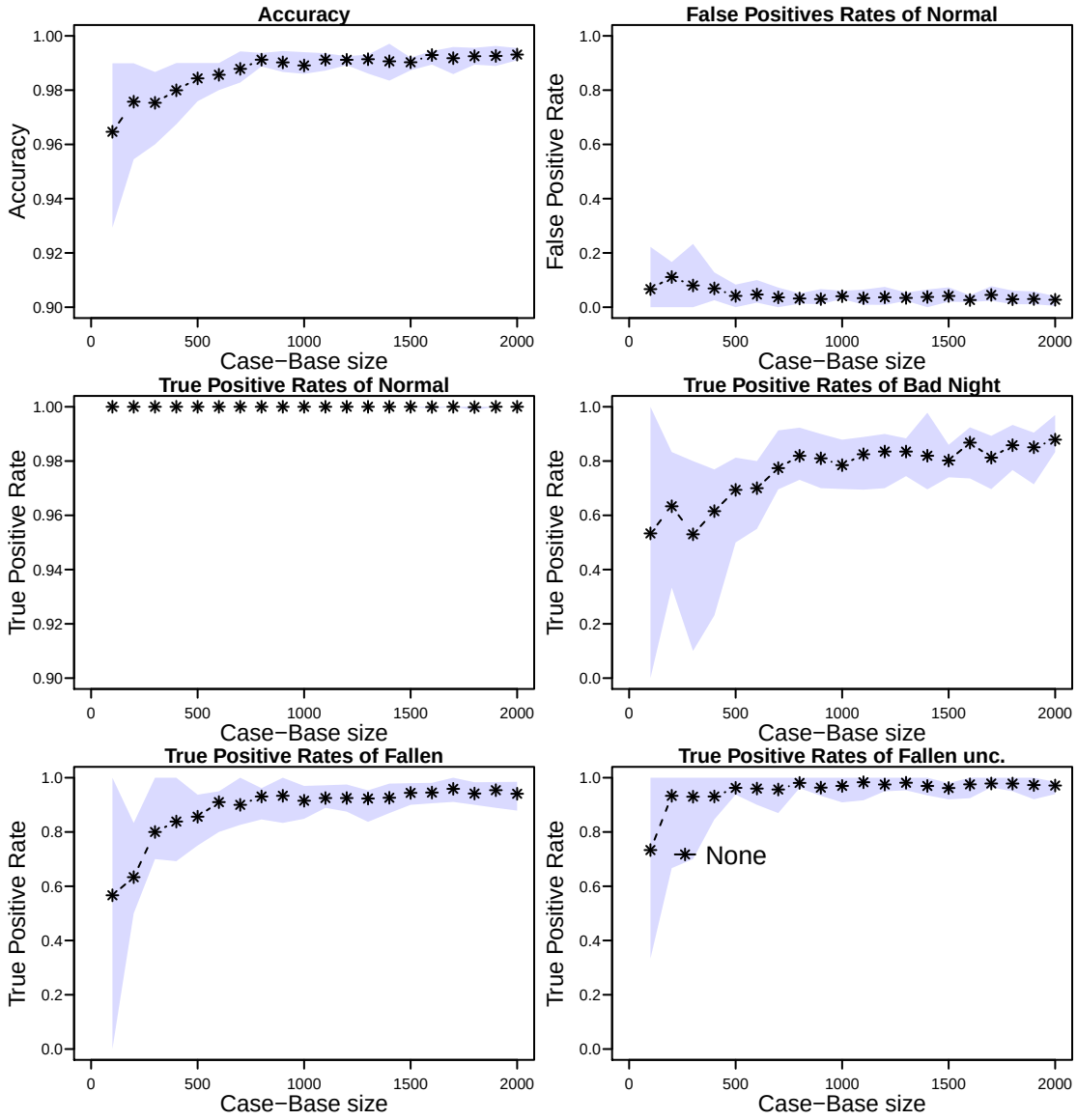


Fig. 6.10 Evaluation of the CBR system varying the case-base size and the case duration.

to produce case-bases equivalent (in terms of CBR system performance) to the original one. The considered tCBMs are t-CNN, t-DROP1, t-ICF, t-RC-FP and t-RENN, which were deeply analysed in chapter 5. The configuration of the CBR system is the same as in the previous experiments with no maintenance, using 10 times $\alpha\beta$ evaluation method with $\alpha = 10$ and $\beta = 1$. The results given by the CBR system with each CBM algorithm are shown in figures 6.11 to 6.15. For each figure, results of the tCBM are compared to the original results. While the original results are shown with asterisks, the results of the tCBM are shown with two different lines. The plotted black broad line shows the results obtained according to the original case-base size. The other line plotted in grey colour shows the same evaluation results but corresponding with the maintained case-base size. In this way, the higher the distance between the dots of the grey line and the bold lines, the higher the reduction rates achieved with the

tCBM algorithm. Additionally, the shaded area surrounding the broad line corresponds to the maximum and minimum observed values of the evaluation results according to the original case-base size. A short distance between the maximum and minimum values implies that the tCBM algorithm has converged and is creating maintained case-bases with little variability in their retained cases.

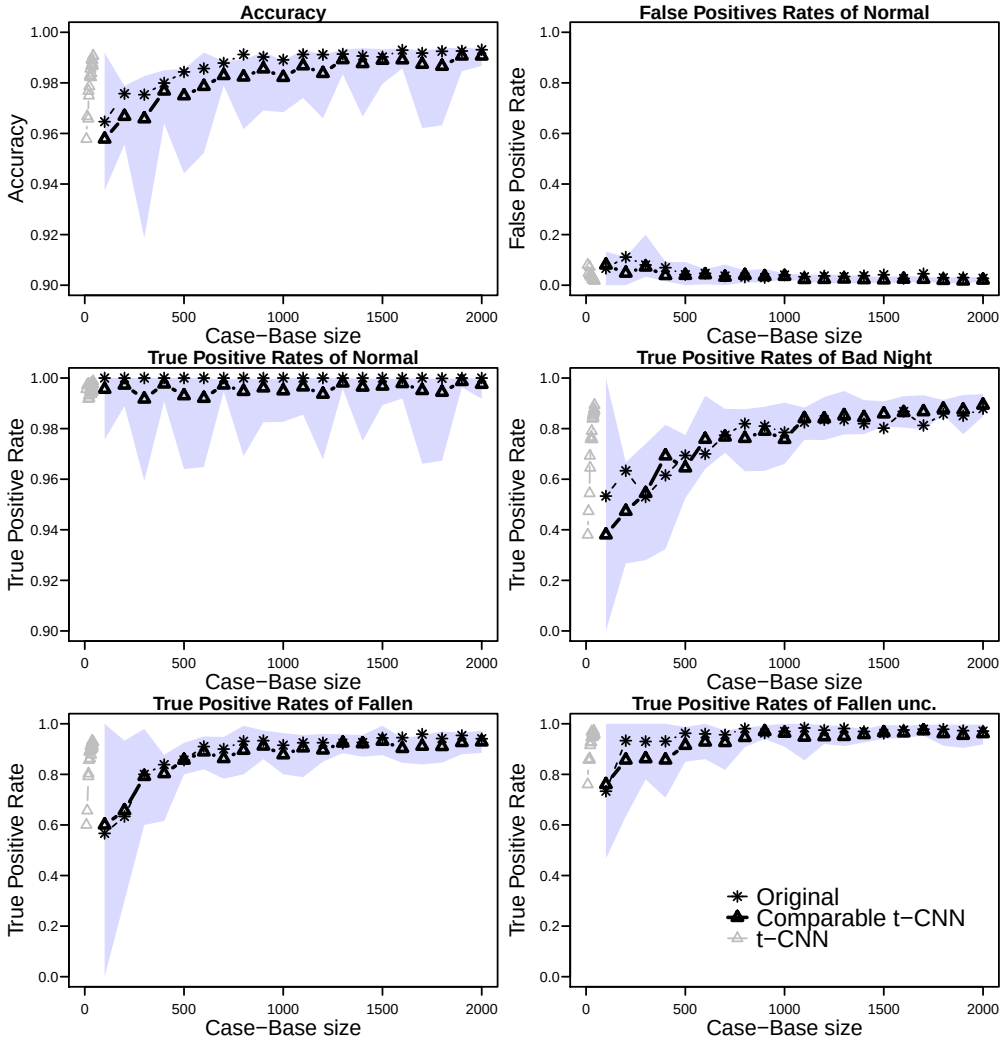


Fig. 6.11 Evaluation of the CBR system varying the case-base size and using t-CNN algorithm.

In order to test whether the original and maintained case-bases are equivalent or not(have statistically the same results), a paired *t-test* analysis is conducted on the results given by the largest original case-base and the results given by each tCBM algorithm, assuming as alternative hypothesis that the result averages from the original and maintained case-base are different. Table 6.2 shows the *p-values* obtained from this study. With *p-values* with a numerical value lower than a significant level of 0.05, there is a strong evidence that the results of the maintained case-base and the original have different averages, so they may be not considered as equivalent. The true positive rate for the normal scenario is not considered since the CBR

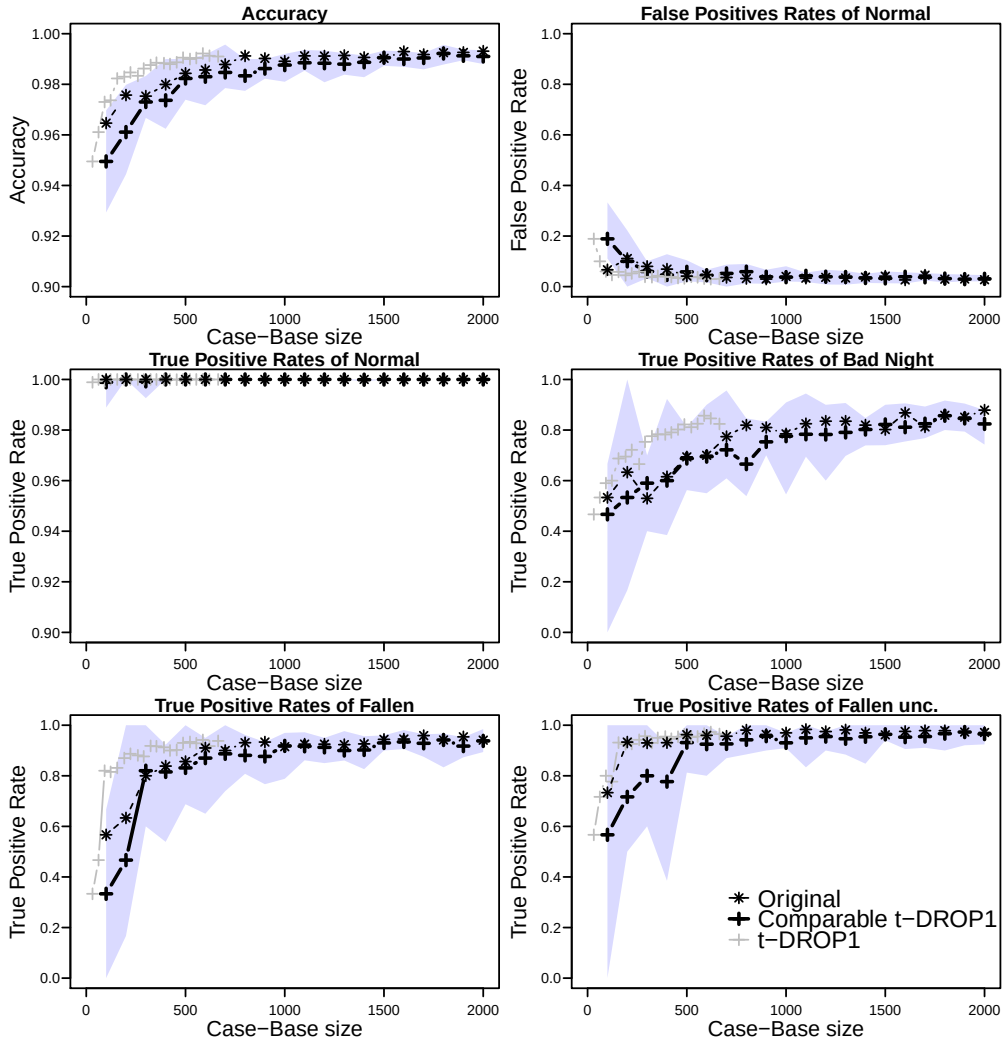


Fig. 6.12 Evaluation of the CBR system varying the case-base size and using t-DROP1 algorithm.

system reaches values almost equal to 1 with all the original and maintained cases-bases.

			Abnormal scenarios		
	Accuracy	FP Normal	Bad night	Fall	Fall unc.
t-CNN	0.02	0.14	0.34	0.41	0.16
t-DROP1	0.07	0.57	0.07	0.84	0.56
t-ICF	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	0.24	0.28
t-RCFP	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$	0.30	0.50
t-RENN	$< 10^{-5}$	0.02	$< 10^{-5}$	0.02	0.30

Table 6.2 Probability that the CBR system results with the maintained and original case-base have significantly the same average results.

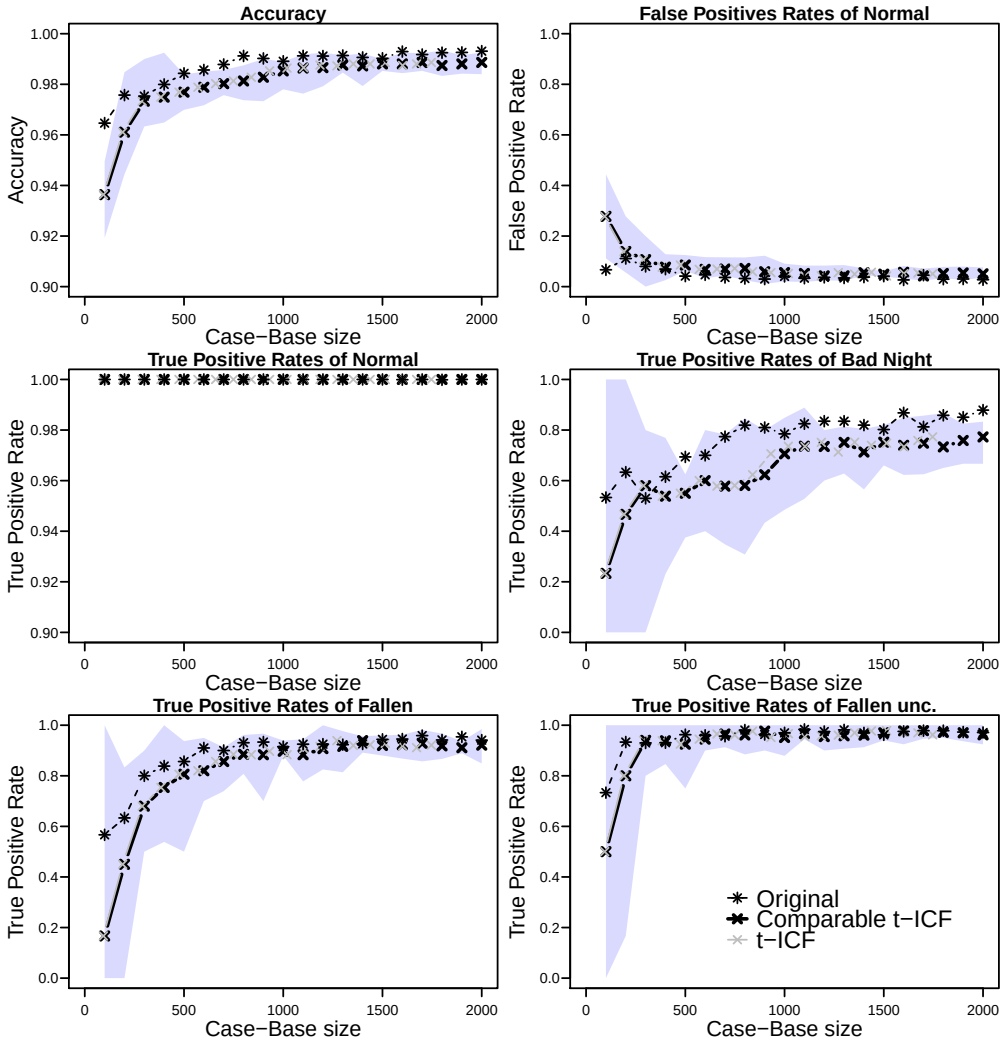


Fig. 6.13 Evaluation of the CBR system varying the case-base size and using t-ICF algorithm.

6.5 Discussion

From the results shown in table 6.2 and figures 6.11 to 6.15, the following discussion for each observed evaluation result can be drawn:

- Regarding the similarity of the results, a CBR system using the maintained case-bases from the largest original case-base has similar results than using the original case-base. The only exception is the detection of bad night scenarios the tCBM algorithms t-ICF, t-RC-FP and t-RENN.

- No maintained case-base is able to reach the same accuracy than the original case-base. However, t-DROP1 algorithm builds maintained case-bases which are significantly equal to the original case-base in terms of accuracy. Furthermore, the accuracy is similar for all the considered maintained case-bases.

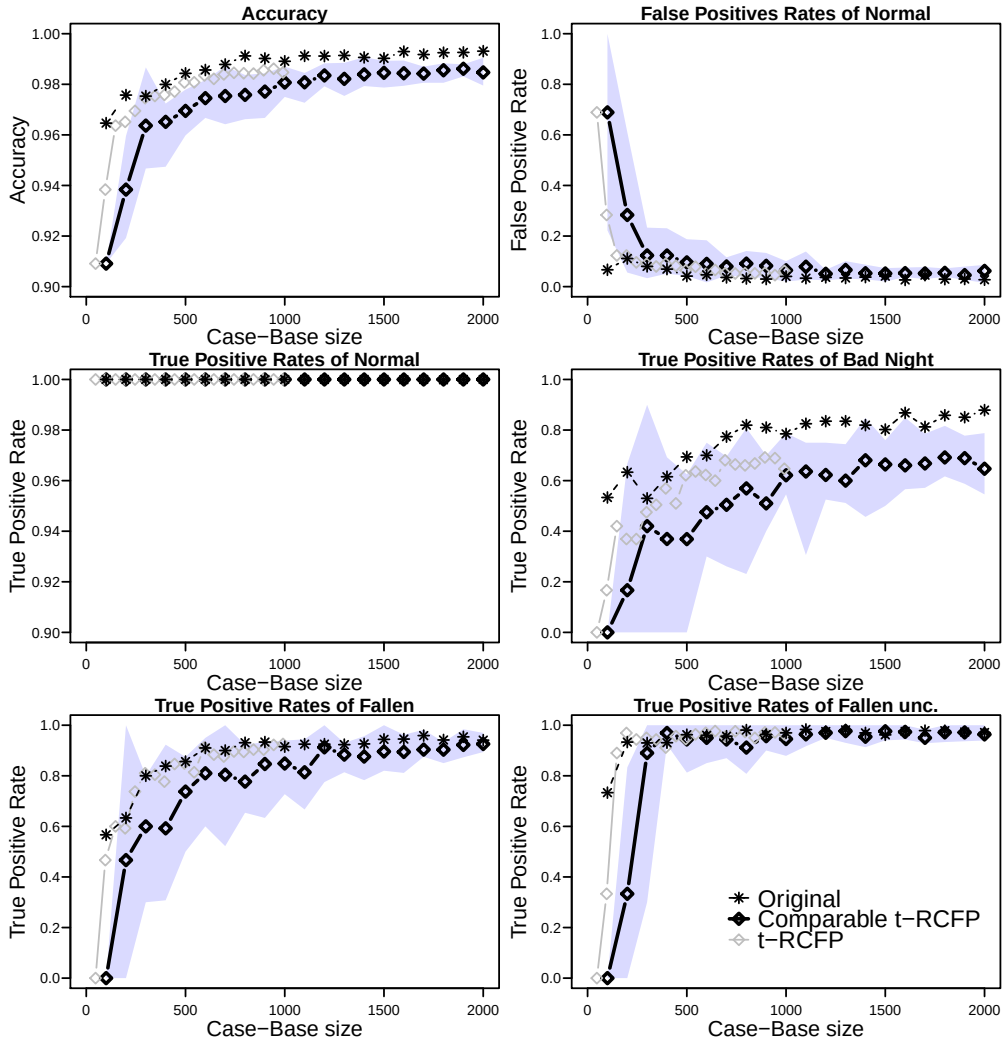


Fig. 6.14 Evaluation of the CBR system varying the case-base size and using t-RC-FP algorithm.

- Algorithms t-CNN and t-DROP1 can create a maintained case-base which is significantly equal to the original case-base in terms of false positives rates of normal scenario detection.
- Regarding the true positive rate of bad night scenarios, t-CNN and t-DROP1 are able to create a equivalent maintained case-base to the original case-base.
- t-RENN is the only algorithm which produces maintained case-bases with significantly different true positive rates of scenarios with falls without loss of consciousness.
- Regarding the detection of scenarios with loss of consciousness, all the algorithms create maintained case-bases with similar true positive rates averages to the original case-base.

When our attention is drawn to the results for each tCBM algorithm and their reduction rates it is worth highlighting:

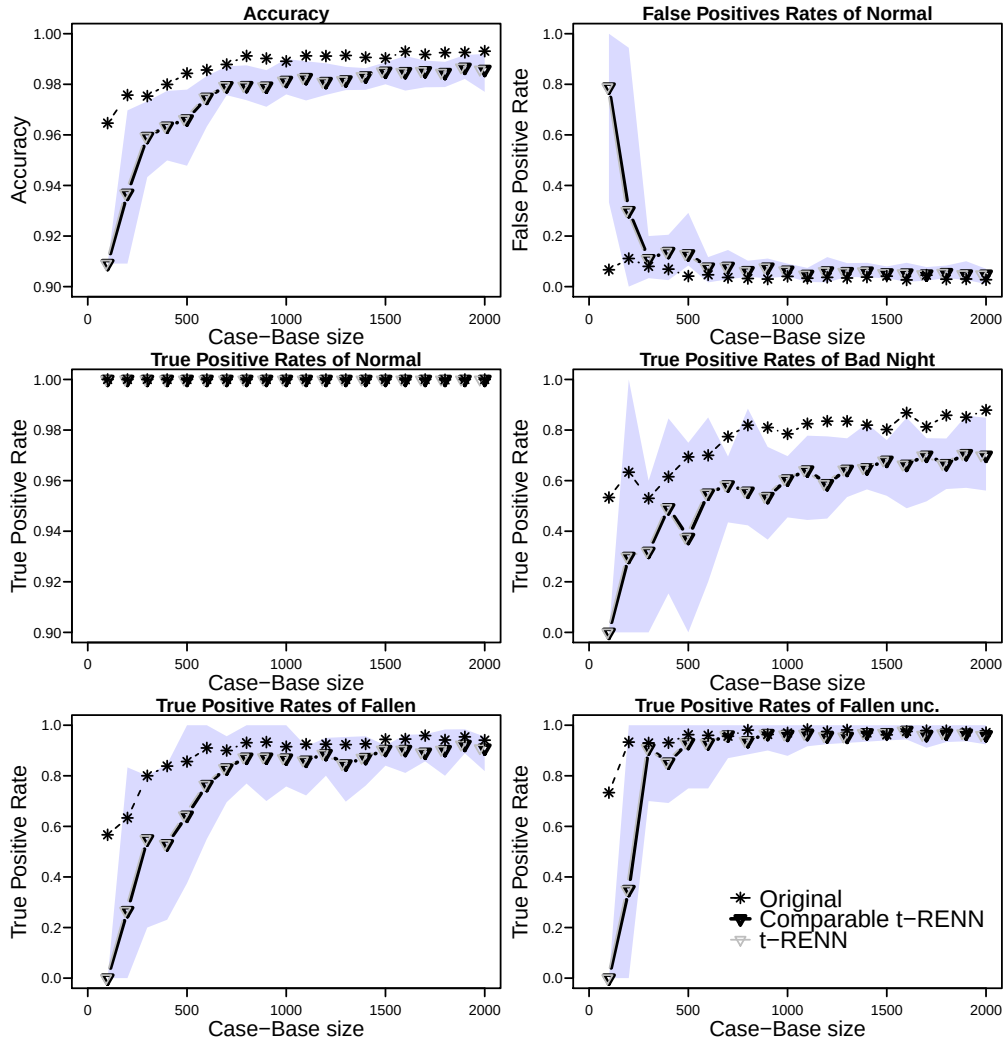


Fig. 6.15 Evaluation of the CBR system varying the case-base size and using t-RENN algorithm.

- The CBR system using case-bases maintained by t-CNN obtains very close results to those given with the original case-base. In particular, the false positive rates of the normal scenarios are slightly smaller, in spite of a slightly worsening of the right classification of normal scenarios. Furthermore, there is no statistical difference between the original and maintained case-bases for the false positives rate of normal scenarios and the proposed abnormal scenarios. The t-CNN algorithm has the most aggressive reduction rate since it is aimed to reduce the number of redundant cases. In fact, the average sizes of the maintained case-bases are always 10% lower than the original case-base size.

- The maintained case-bases built by t-DROP1 can be considered equivalent to the original case-bases since there is strong evidence supporting that the averages results with the original and maintained case-base are the same. Furthermore, the reduction rate of this

algorithm is near 25% of the original case-base size.

- Algorithm t-ICF obtains significantly the same average results to the original case-base in detecting both scenarios of falls. However, the true positive rates of bad night scenarios are slightly lower than the original case-base. Besides, there is strong evidence that the averages given with the maintained and original case-base have different averages for the accuracy, false positive rates of normal scenarios and true positive rates of bad night scenarios. The reduction rate of this algorithm is low, removing around 10% of the cases from the original case-base.

- Regarding t-RC-FP and its results in table 6.2, there is very strong evidence that this algorithm obtains similar results in detecting both scenarios of falls. Besides, the CBR system using the maintained case-bases by this algorithm gets significantly lower true positive rates of bad night scenarios than in the original case-base. The reduction rate of this algorithm is always about 50% of the original case-base.

- Algorithm t-RENN creates maintained case-bases with significantly equal true positive rates of scenarios of falls with loss of consciousness. For the rest of the results, there are strong evidence that supports that there are a significant difference between the original and maintained case-base. The reduction rate of this algorithm is the lowest of all tCBM algorithms considered, since t-RENN is aimed to reduce the number of cases surrounded by cases with different solution, which corresponds to the less frequent cases (the abnormal scenarios).

6.6 Conclusions

The main goal of this chapter is to evaluate the suitability of the techniques proposed in this thesis in a practical problem. To this end, we propose a CBR system within a Smart Home system. This CBR system gathers the activity of the monitored person (in the form of event sequence) from a set sensor, and uses temporal similarity to retrieve the most similar previously stored cases to classify the type of scenario occurring at home. We evaluate the suitability of this approach using a simulation of the activities at home considering different risky scenarios. Finally, the results using the original case-base are compared to those obtained using a set of maintained case-bases by the algorithms t-CNN, t-DROP1, t-ICF, t-RC-FP and t-RENN.

The experiments conducted demonstrate that the proposed CBR system is able to successfully detect the different proposed scenarios. However, a proper detection of scenarios requires a large number of cases. Since the proposed monitoring system is based on hardware with low computational capabilities, the size of the case-base may cause performance problems, so the

use of tCBM is justified. Therefore, several experiments have been conducted in order to confirm whether it is possible to perform a maintenance tasks in the case-base without worsening the capabilities of the CBR system.

The experimental results show that all the considered tCBM algorithms are able to create maintained cases-bases with similar capabilities in detecting the different proposed abnormal scenario. However, the success in creating a maintained case-base equivalent to the original depends on the number of cases within the case-base and the tCBM algorithm applied. On the whole, the following conclusion can be drawn:

- Despite a maintained case-base equivalent to the original case-base can be obtained, experiments prove the relevance of evaluating a set of tCBM algorithms due to the variability of their results.
- Algorithms t-CNN and t-DROP1 are able to create case-bases which statistically present the same results averages than the original case-base.
- Algorithms t-ICF, t-RC-FP and t-RENN can build similar case-bases to the original, although they are not totally equivalent since they have significantly different average results than the original case-base.
- Finally, algorithm t-RENN does not reduce significantly the size of the case-base because it usually removes cases describing abnormal scenarios, which are less frequent than normal cases. Thus, we recommend not using t-RENN in case-bases where the key cases rarely appear in the case-base.

Chapter 7

Conclusions

As a result of the research carried out during this thesis, we have reviewed in depth the existing literature related to topics such as CBM algorithms, made an evaluation of the same and discussed alternatives for temporal representation. Furthermore, we have proposed a non-deterministic CBM algorithm which takes advantage of the implicit information contained in the case-base. Then, we introduced a specific evaluation methodology for deterministic and non-deterministic CBM algorithms. We developed a set of temporal CBM algorithms to perform maintenance with temporal dependant domains. Finally, we explained the design of a CBR system to monitor elderly people living alone using the proposed temporal case representations and temporal CBM algorithms to build the case-base. Below, we discuss the main conclusions that can be drawn.

When the CBM is approached as a Multi-Objective optimization problem with an Evolutionary Algorithm, the accuracy reached with the created maintained case-bases are significantly similar to that given with the original case-base. Besides, the reduction of the case-base size is proportional to the proportion of redundant cases, avoiding the over-fitting problem.

Traditional evaluation methods do not offer enough statistical support to reliably quantify the effects on the CBR system when undergoing maintenance. To tackle this problem, we have proposed the $\alpha\beta$ evaluation method, which is able to provide reliable evaluation results with a high degree of confidence.

The experiments conducted supported our hypothesis that Cross-Validation and Hold-Out are not sufficient to provide reliable results when evaluating CBM algorithms, whether they are deterministic or non-deterministic. On the other hand, according to our experiments, among the considered evaluation methods, $\alpha\beta$ is the only one able to provide reliable evaluation measurements when dealing with both deterministic and non-deterministic CBM algorithms.

Similarly with non-temporal case-bases, a CBR system using a temporal case-base could suffer from performance problems and loss of accuracy when the number of stored cases is high. Thus, we proposed a set of tCBM algorithms: t-CNN, t-RENN, t-DROP1, t-ICF and t-RC-FP to ease these problems. After conducting several experiments, we conclude that the successful maintenance of temporal case-bases is possible. Similarly to CBM algorithms, only non-temporal factors bias the performance of the tCBM algorithms, such as the case-base size and the proportion of redundant and noise cases. Temporal factors may also affect the capability of the CBR system to solve problems, but the tCBM experimental results do not seem to be affected by them.

Finally, we tested the techniques proposed to detect abnormal situations in elderly people living alone at home. The experiments conducted empirically confirm that the proposed temporal CBR system is able to successfully detect the different scenarios, althoug the proper detection of scenarios requires a large number of cases. The experimental results show that all the considered tCBM algorithms are able to create maintained cases-bases with similar capa-

bilities to detect the different proposed abnormal scenarios. However, the success of creating a maintained case-base equivalent to the original depends on the number of cases within the case-base and the tCBM algorithm applied.

Most of the parts making up this thesis have been evaluated in peer-reviewed journals and international congresses related to the CBR field:

7.1 Contributions

- Eduardo Lupiani, Jose M. Juarez, Jose Palma: *A Proposal of Temporal Case-Base Maintenance Algorithms*. Proceedings of the 22nd International Conference on Case-Based Reasoning, ICCBR 2014.

- Eduardo Lupiani, Jose M. Juarez, Jose Palma, Christian Sauer, Thomas Roth-Berghofer: *Using Case-Based Reasoning to Detect Risk Scenarios of Elderly People Living Alone at Home*. Proceedings of the 22nd International Conference on Case-Based Reasoning, ICCBR 2014.

- Eduardo Lupiani, Jose M. Juarez, Jose Palma: *Evaluating Case-Base Maintenance algorithms*. Knowledge-Based Systems, 2014. JCR Impact factor: 4.104

- Eduardo Lupiani, Christian Sauer, Thomas Roth-Berghofer, Jose M. Juarez, Jose Palma: *Implementation of similarity measures for event sequences in myCBR*. In: Proceedings of the 18th UK Workshop on Case-Based Reasoning, UKCBR 2013.

- Eduardo Lupiani, Susan Craw, Stewart Massie, Jose M. Juarez, Jose Palma: *A Multi-Objective Evolutionary Algorithm Fitness Function for Case-Base Maintenance*. Proceedings of the 21st International Conference on Case-Based Reasoning, ICCBR 2013: 218-232.

- Eduardo Lupiani, Fernando Jimenez, Jose M. Juarez, Jose Palma: *An Evolutionary Multiobjective Constrained Optimisation Approach for Case Selection: Evaluation in a Medical Problem*. Proceedings of the 14th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2011: 383-392

- Jose M. Juarez, Eduardo Lupiani , Jose Palma: *Case Selection Evaluation Methodology*. Industrial Conference on Data Mining - Workshops, 2011, 17-29

- Eduardo Lupiani, Jose M. Juarez, Fernando Jimenez, Jose Palma: *Evaluating Case Selection Algorithms for Analogical Reasoning Systems*. Proceedings of the 4th International Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC 2011 (1): 344-353

Additionally, the hypothesis proposed in this thesis was presented and validated in the following Doctoral Consortium.

- Eduardo Lupiani: *Case-Based Maintenance in Ambient Assisted Living*. Infinite Salud Doctoral Consortium. Murcia, Spain. (2012).
- Eduardo Lupiani: *Case Selection in Ambient Assisted Living*. 20th International Conference on Case-Based Reasoning. Lyon, France. (2012). Mentor: Ralph Bergmann

Finally, the following work has been submitted and we are awaiting its acceptance and publication:

- Eduardo Lupiani, Stewart Massie, Susan Craw, Jose M. Juarez, Jose Palma: *Case-Base Maintenance with Multi-Objective Evolutionary Algorithms*. Journal of Intelligent Information Systems.

7.2 Future work

The research presented in this document has raised several questions. Firstly, the search for an optimal maintained case-base is still a weakly defined theoretical problem. Despite our efforts to model maintenance as a multi-objective optimization problem, our opinion is that knowledge inside cases may play a relevant role in the future development of both CBM and tCBM algorithms. In particular, because our proposal is based on event sequences, future research will be oriented to analysing reductions in their length through events deletion, as well as by merging two different temporal cases that cover the same time domain. For instance, if two temporal cases contain two event sequences similar to each other (same number of events and few displacements over time), it would seem feasible to create a single temporal case with an event sequence similar to both of them.

In spite of the fact that the experiments conducted support our initial hypothesis, we are not in a position to confirm that tCBM can be used successfully with other temporal representations, such workflows or time-series. Hence, more experiments need to be carried out to answer this open question.

Furthermore, our intention to integrate the proposed tCBM algorithms as part of the temporal data mining framework developed by the AIKE research group. In this way, the research group can benefit in future research from a new tool to reduce the size of the temporal data available without worsening the quality of the experimental results.

Finally, our CBR proposal to monitor elderly people living alone at home has been evaluated using synthetic data. In the future tests and experiments with real data from real people will be carried out. If the experiments show positive results, our plans is to promote the elderly monitoring system commercially.

References

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

[2] Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, volume 730 of *LNCS*, pages 69–84. Springer Berlin Heidelberg.

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

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

[5] Ahn, H., Kim, K.-j., and 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.

[6] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422.

[7] Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications Of The ACM*, 26(11):832–843.

[8] Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

[9] Allen, J. and Hayes, P. (1985). A Common-Sense Theory of Time. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*.

[10] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235.

[11] Ashley, K. D. (1991). Reasoning with cases and hypotheticals in HYPO. *International Journal of Man-Machine Studies*, 34(6):753 – 796.

[12] Augusto, J. C. and Nugent, C. D. (2006). Smart homes can be smarter. In *Designing Smart Homes*, volume 4008 of *LNCS*, pages 1–15. Springer Berlin Heidelberg.

[13] Barger, T., Brown, D., and Alwan, M. (2005). Health-status monitoring through analysis of behavioral patterns. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(1):22–27.

[14] Bergmann, R. and Gil, Y. (2011). Retrieval of semantic workflows with knowledge intensive similarity measures. In *Case-Based Reasoning Research And Development*, volume 6880 of *LNCS*, pages 17–31.

[15] Botía, J. A., Villa, A., and Palma, J. (2012). Ambient assisted living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications*, 39(9):8136–8148.

[16] Bottrighi, A., Leonardi, G., Montani, S., Portinale, L., and Terenziani, P. (2010). Intelligent data interpretation and case base exploration through temporal abstractions. In *Case-Based Reasoning. Research and Development*, volume 6176 of *LNCS*, pages 36–50. Springer Berlin Heidelberg.

[17] Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression - the x-random case. *International Statistical Review*, 60(3):291–319.

[18] Brighton, H. and Mellish, C. (1999). On the consistency of information filters for lazy learning algorithms. In *Principles of Data Mining and Knowledge Discovery*, volume 1704 of *LNAI*, pages 283–288.

[19] Broek, G. V. D., Cavallo, F., and Wehrmann, C. (2010). *AALiance Ambient Assisted Living Roadmap*. IOS press.

[20] Bruce, B. (1972). A model for temporal references and its application in a question answering program. *Artificial Intelligence*, 4:1–25.

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

[22] Caine, K. E., Rogers, W. A., and Fisk, A. D. (2005). Privacy perceptions of an aware home with visual sensing devices. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 1856–1858.

[23] Chan, K.-P. and Fu, A.-C. (1999). Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133.

[24] Chong, C.-Y. and Kumar, S. (2003). Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

[25] Coello, C. C., Lamont, G., and van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer.

[26] Combi, C., Gozzi, M., Juarez, J. M., Marin, R., and Oliboni, B. (2007). Querying clinical workflows by temporal similarity. In *Artificial Intelligence in Medicine*, volume 4594 of *LNCS*, pages 469–478. Springer Berlin Heidelberg.

[27] Combi, C., Gozzi, M., Oliboni, B., Juarez, J. M., and Marin, R. (2009). Temporal similarity measures for querying clinical workflows. *Artificial Intelligence In Medicine*, 46:37–54.

[28] Cook, D., Schmitter-Edgecombe, M., Crandall, A., Sanders, C., and Thomas, B. (2009). Collecting and disseminating smart home sensor data in the casas project. In *Proc. of CHI09 Workshop on Developing Shared Home Behavior Datasetsto Advance HCI and Ubiquitous Computing Research.*

[29] Cook, D., Youngblood, M., Heierman, E.O., I., Gopalratnam, K., Rao, S., Litvin, A., and Khawaja, F. (2003). Mavhome: an agent-based smart home. In *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 521–524.

[30] Copetti, A., Loques, O., Leite, J., Barbosa, T., and da Nóbrega, A. (2009). Intelligent context-aware monitoring of hypertensive patients. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–6.

[31] Corchado, J. M., Bajo, J., de Paz, Y., and Tapia, D. I. (2008a). Intelligent environment for monitoring alzheimer patients, agent technology for health care. *Decision Support Systems*, 44(2):382 – 396.

[32] Corchado, J. M., Bajo, J., and Paz, Y. (2008b). A CBR system: The core of an ambient intelligence health care application. In *Soft Computing Applications in Industry*, volume 226 of *Studies in Fuzziness and Soft Computing*, pages 311–330. Springer Berlin Heidelberg.

[33] Cordier, A., Lefevre, M., Champin, P.-A., Georgeon, O. L., and Mille, A. (2013). Trace-based reasoning - modeling interaction traces for reasoning on experiences. In *FLAIRS Conference*.

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

[35] Craw, S., Jarmulak, J., and Rowe, R. (2001). Maintaining retrieval knowledge in a case-based reasoning system. *Computational Intelligence*, 17(2):346–363.

[36] Cummins, L. and Bridge, D. (2009). Maintenance by a committee of experts: The MACE approach to case-base maintenance. In *Case-Based Reasoning Research And Development, Proceedings*, volume 5650 of *LNAI*, pages 120–134.

[37] Cummins, L. and Bridge, D. G. (2011). Choosing a case base maintenance algorithm using a meta-case base. In *SGAI Conf.*, pages 167–180.

[38] Cunningham, P. (2009). A taxonomy of similarity mechanisms for case-based reasoning. *Knowledge and Data Engineering, IEEE Transactions on*, 21(11):1532–1543.

[39] Cunningham, P., Doyle, D., and Loughrey, J. (2003). An evaluation of the usefulness of case-based explanation. In *In Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 122–130.

[40] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions On Evolutionary Computation*, 6(2):182–197.

[41] Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95.

[42] Delany, S. and Cunningham, P. (2004). An analysis of case-base editing in a spam filtering system. In *Advances In Case-Based Reasoning, Proceedings*, volume 3155 of *LNCS*, pages 128–141.

[43] Dijkman, R., Dumas, M., and García-Bañuelos, L. (2009). Graph matching algorithms for business process model similarity search. In *Proceedings of the 7th International Conference on Business Process Management*, BPM '09, pages 48–63.

[44] Ding, D., Cooper, R. A., Pasquina, P. F., and Fici-Pasquina, L. (2011). Sensor technology for smart homes. *Maturitas*, 69(2):131 – 136.

[45] Díaz-Agudo, B. and González-Calero, P. (2001). A declarative similarity framework for knowledge intensive CBR. In *Case-Based Reasoning Research and Development*, volume 2080 of *LNCS*, pages 158–172. Springer Berlin Heidelberg.

[46] Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. SpringerVerlag.

[47] Finnie, G. and Sun, Z. (2002). Similarity and metrics in case-based reasoning. *International Journal of Intelligent Systems*, 17(3):273–287.

[48] Finnie, G. and Sun, Z. (2003). {R5} model for case-based reasoning. *Knowledge-Based Systems*, 16(1):59 – 65.

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

[50] Freksa, C. (1992). Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227.

[51] Fuchs, B., Mille, A., and Chiron, B. (1995). Operator decision aiding by adaptation of supervision strategies. In *Case-Based Reasoning Research and Development*, volume 1010 of *LNCS*, pages 23–32. Springer Berlin Heidelberg.

[52] Galushka, M., Patterson, D., and Rooney, N. (2006). Temporal data mining for smart homes. In *Designing Smart Homes*, volume 4008 of *LNCS*, pages 85–108. Springer Berlin Heidelberg.

[53] Gates, G. (1972). Reduced nearest neighbor rule. *IEEE Transactions On Information Theory*, 18(3):431+.

[54] Ge, X. and Smyth, P. (2000). Deformable markov model templates for time-series pattern matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90.

[55] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155 – 170.

[56] Georgeon, O. L., Mille, A., Bellet, T., Mathern, B., and Ritter, F. E. (2012). Supporting activity modelling from activity traces. *Expert Systems*, 29(3):261–275.

[57] Gil, Y. (2012). Reproducibility and efficiency of scientific data analysis: Scientific workflows and case-based reasoning. In *Case-Based Reasoning Research And Development*, volume 7466 of *LNCS*, pages 2–2.

[58] Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern.*, 16(1):122–128.

[59] Gundersen, O. E. (2012). Toward measuring the similarity of complex event sequences in real-time. In *Case-Based Reasoning Research and Development*. Springer Berlin Heidelberg.

[60] Gundersen, O. E., Sørmo, F., Aamodt, A., and Skalle, P. (2013). A real-time decision support system for high cost oil-well drilling operations. *AI Magazine*, 34(1):21–32.

[61] Hammond, K. J. (1989). *Case-based Planning: Viewing Planning As a Memory Task*. Academic Press Professional, Inc.

[62] Hart, P. (1968). Condensed nearest neighbor rule. *IEEE Transactions On Information Theory*, 14(3):515+.

[63] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc.

[64] Helal, S., Winkler, B., Lee, C., Kaddoura, Y., Ran, L., Giraldo, C., Kuchibhotla, S., and Mann, W. (2003). Enabling location-aware pervasive computing applications for the elderly. In *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 531–536.

[65] Hinrichs, T. R. (1992). *Problem-solving in Open Worlds: A Case Study in Design*. PhD thesis. UMI Order No. GAX92-06083.

[66] Holland, J. H. (1975). *Adaptation in Natural And Artificial Systems*. MIT Press.

[67] Houeland, T. G. and Aamodt, A. (2010). The utility problem for lazy learners - towards a non-eager approach. In *Case-Based Reasoning Research And Development, 18Th International Conference On Case-Based Reasoning, ICCBR 2010*, volume 6176 of *LNAI*, pages 141–155.

[68] Huang, Z., Juarez, J. M., Duan, H., and Li, H. (2013). Length of stay prediction for clinical treatment process using temporal similarity. *Expert Syst. Appl.*, 40(16):6330–6339.

[69] Intille, S., Larson, K., Tapia, E., Beaudin, J., Kaushik, P., Nawyn, J., and Rockinson, R. (2006). Using a live-in laboratory for ubiquitous computing research. In *Pervasive Computing*, volume 3968 of *LNCS*, pages 349–365. Springer Berlin Heidelberg.

[70] Ishibuchi, H., Nakashima, T., and Nii, M. (2001). Genetic-algorithm-based instance and feature selection. In *Instance Selection and Construction for Data Mining*, volume 608 of *The Springer International Series in Engineering and Computer Science*, pages 95–112. Springer US.

[71] Jaczynski, M. (1997). A framework for the management of past experiences with time-extended situations. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, CIKM '97, pages 32–39.

[72] Jong, K. A. D. and Spears, W. M. (1990). An analysis of the interacting roles of population size and crossover in genetic algorithms. In *PPSN*, pages 38–47.

[73] Juarez, J. M., Campos, M., Palma, J., and Marin, R. (2011a). T-care: temporal case retrieval system. *Expert Systems*, 28:324–338.

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

[75] Juarez, J. M., Lupiani, E., and Palma, J. (2011b). Case selection evaluation methodology. In *Industrial Conference on Data Mining - Workshops*, pages 17–29.

[76] Jære, M. D., Aamodt, A., and Skalle, P. (2002). Representing temporal knowledge for case-based prediction. In *Advances In Case-Based Reasoning*, volume 2416 of *LNCS*, pages 174–188. Springer Berlin Heidelberg.

[77] Kahn, K. M. and Gorry, G. (1977). Mechanizing temporal knowledge. *Artificial Intelligence*, 9:87–108.

[78] Kapetanakis, S., Petridis, M., Knight, B., Ma, J., and Bacon, L. (2010). A case based reasoning approach for the monitoring of business workflows. In *Case-Based Reasoning. Research and Development*, volume 6176 of *LNCS*, pages 390–405. Springer Berlin Heidelberg.

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

[80] Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E. D., Starner, T., and Newstetter, W. (1999). The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, CoBuild '99, pages 191–198. Springer-Verlag.

[81] Kim, K. and Han, I. (2001). Maintaining case-based reasoning systems using a genetic algorithms approach. *Expert Systems with Applications*, 21:139–145.

[82] Knight, B. and Ma, J. (1993). Time representation: A taxonomy of temporal models. *Artificial Intelligence Review*, 7(6):401–419.

[83] Kofod-Petersen, A. and Aamodt, A. (2006). Contextualised ambient intelligence through case-based reasoning. In *Proceedings of the 8th European Conference on Advances in Case-Based Reasoning*, ECCBR'06, pages 211–225.

[84] Kofod-Petersen, A. and Aamodt, A. (2009). Case-based reasoning for situation-aware ambient intelligence: A hospital ward evaluation study. In *Case-Based Reasoning Research and Development*, pages 450–464. Springer.

[85] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI'95*, pages 1137–1145.

[86] Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory*. *Cognitive Science*, 7(4):243–280.

[87] Krishnan, N. C. and Cook, D. J. (2014). Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 10, Part B(0):138 – 154.

[88] Kröse, B., Kasteren, T. V., Gibson, C., and Dool, T. V. D. (2008). Care: Context awareness in residences for elderly. In *In Proc. of the Conf. of the International Society for Gerontechnology*.

[89] Kuncheva, L. and Jain, L. (1999). Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20(11-13):1149–1156.

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

[91] Leake, D., Maguitman, A., and Reichherzer, T. (2006). Cases, context, and comfort: Opportunities for case-based reasoning in smart homes. In *Designing Smart Homes*, volume 4008 of *LNCS*, pages 109–131. Springer Berlin Heidelberg.

[92] Leake, D. and Wilson, D. (1998). Categorizing case-base maintenance: Dimensions and directions. In *Advances In Case-Based Reasoning*, volume 1488 of *LNAI*, pages 196–207.

[93] Leake, D. and Wilson, M. (2011). How many cases do you need? assessing and predicting case-base coverage. In *Proceedings Of The 19Th International Conference On Case-Based Reasoning Research And Development*, ICCBR'11, pages 92–106.

[94] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

[95] Li, Y., Shiu, S., and Pal, S. (2006). Combining feature reduction and case selection in building CBR classifiers. *IEEE Transactions On Knowledge And Data Engineering*, 18(3):415–429.

[96] Lu, R., Sadiq, S., Padmanabhan, V., and Governatori, G. (2006). Using a temporal constraint network for business process execution. In *Proceedings of the 17th Australasian Database Conference - Volume 49*, pages 157–166.

[97] Lupiani, E., Craw, S., Massie, S., Juarez, J. M., and Palma, J. (2013a). A multi-objective evolutionary algorithm fitness function for case-base maintenance. In *ICCBR*, pages 218–232.

[98] Lupiani, E., Jimenez, F., Juarez, J. M., and Palma, J. (2011a). An evolutionary multiobjective constrained optimisation approach for case selection: Evaluation in a medical problem. In *CAEPIA*, pages 383–392.

[99] Lupiani, E., Juarez, J. M., Jimenez, F., and Palma, J. (2011b). Evaluating case selection algorithms for analogical reasoning systems. In *Foundations on Natural and Artificial Computation*, volume 6686 of *LNCS*, chapter 36, pages 344–353.

[100] Lupiani, E., Sauer, C., Roth-Berghofer, T., Juarez, J. M., and Palma, J. (2013b). Implementation of similarity measures for event sequences in myCBR. In *Proceedings of the 18th UKCBR Workshop*.

[101] Ma, T., Kim, Y.-D., Ma, Q., Tang, M., and Zhou, W. (2005). Context-aware implementation based on CBR for smart home. In *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, volume 4, pages 112–115 Vol. 4.

[102] Madhusudan, T., Zhao, J., and Marshall, B. (2004). A case-based reasoning framework for workflow model management. *Data & Knowledge Engineering*, 50(1):87 – 115.

[103] Makikawa, M. and Murakami, D. (1996). Development of an ambulatory physical activity and behavior map monitoring system. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 1, pages 71–72 vol.1.

[104] Mannila, H. and Moen, P. (1999). Similarity between event types in sequences. In *DataWarehousing and Knowledge Discovery*. Springer Berlin Heidelberg.

[105] Markovitch, S. and Scott, P. (1988). The role of forgetting in learning. In *Proceedings of The Fifth International Conference on Machine Learning*, pages 459–465.

[106] Massie, S., Craw, S., and Wiratunga, N. (2005). Complexity-guided case discovery for case based reasoning. In *Proceedings of the 20th national conference on Artificial Intelligence - Volume 1*, AAAI'05, pages 216–221.

[107] Massie, S., Craw, S., and Wiratunga, N. (2006). Complexity profiling for informed case-base editing. In *Advances In Case-Based Reasoning, Proceedings*, volume 4106 of *LNAI*, pages 325–339.

[108] McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *English*, 4(463-502):463–502.

[109] McDermott, D. (1982). A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6(2):101–155.

[110] McKenna, E. and Smyth, B. (2001). Competence-guided case-base editing techniques. In *Advances In Case-Based Reasoning, Proceedings*, volume 1898 of *LNAI*, pages 186–197.

[111] Minor, M. and Görg, S. (2011). Acquiring adaptation cases for scientific workflows. In *Case-Based Reasoning Research and Development*, volume 6880 of *LNCS*, pages 166–180. Springer Berlin Heidelberg.

[112] Minor, M., Tartakovski, A., and Bergmann, R. (2007). Representation and structure-based similarity assessment for agile workflows. In *ICCBR*, pages 224–238.

[113] Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2-3):363–391.

[114] Moen, P. (2000). *Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining*. PhD thesis, University of Helsinki.

[115] Montani, S., Bottrighi, A., Leonardi, G., Portinale, L., and Terenziani, P. (2009). Multi-level abstractions and multi-dimensional retrieval of cases with time series features. In *Case-Based Reasoning Research And Development*, volume 5650 of *LNCS*, pages 225–239.

[116] Montani, S. and Leonardi, G. (2014). Retrieval and clustering for supporting business process adjustment and analysis. *Information Systems*, 40(0):128 – 141.

[117] Montani, S. and Portinale, L. (2006). Accounting for the temporal dimension in case-based retrieval: A framework for medical applications. *Computational Intelligence*, 22(3-4):208–223.

[118] Montani, S., Portinale, L., Bellazzi, R., and Leonardi, G. (2004). Rhene: A case retrieval system for hemodialysis cases with dynamically monitored parameters. In *ECCBR*, pages 659–672.

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

[120] Mosqueira-Rey, E. and Moret-Bonillo, V. (2000). Validation of intelligent systems: a critical study and a tool. *Expert Systems with Applications*, 18(1):1 – 16.

[121] Nakhaeizadeh, G. (1994). Learning prediction of time series. a theoretical and empirical comparison of CBR with some other approaches. In *Topics in Case-Based Reasoning*, volume 837 of *LNCS*, pages 65–76. Springer Berlin Heidelberg.

[122] Nilsson, M. (2004). Building a case base for stress diagnosis: An analysis of classified respiratory sinus arrhythmia sequences. In *Proceedings of the ECCBR 2004 Workshops*, pages 55–63.

[123] Obweger, H., Suntinger, M., Schiefer, J., and Raidl, G. (2010). Similarity searching in sequences of complex events. In *Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on*.

[124] Ogawa, M., Suzuki, R., Otake, S., Izutsu, T., Iwaya, T., and Togawa, T. (2002). Long term remote behavioral monitoring of elderly by using sensors installed in ordinary houses. In *Microtechnologies in Medicine amp; Biology 2nd Annual International IEEE-EMB Special Topic Conference on*, pages 322–325.

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

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

[127] Porter, B. and Bareiss, R. (1986). *PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks*. Austin AI TR. Artificial Intelligence Laboratory, The University of Texas at Austin.

[128] Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., and Juarez, J. (2006). A case-based architecture for temporal abstraction configuration and processing. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 667–676.

[129] Ram, A. and Santamaría, J. (1997). Continuous case-based reasoning. *Artificial Intelligence*, 90(1–2):25 – 77.

[130] Rawi, M. and Al-Anbuky, A. (2009). Passive house sensor networks: Human centric thermal comfort concept. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 255–260.

[131] Richter, M. M. (2007). Foundations of similarity and utility. In *FLAIRS Conference*, pages 30–37. AAAI Press.

[132] Rissland, E. L. (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, pages 6–13.

[133] Roddick, J. and Spiliopoulou, M. (2002). A survey of temporal knowledge discovery paradigms and methods. *Knowledge and Data Engineering, IEEE Transactions on*, 14(4):750–767.

[134] Roth-Berghofer, T. (2004). Explanations and case-based reasoning: Foundational issues. In *Advances in Case-Based Reasoning*, volume 3155 of *LNCS*, pages 389–403. Springer Berlin Heidelberg.

[135] S B Needleman, C. D. W. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

[136] Safavi, A., Keshavarz-Haddad, A., Khoubani, S., Mosharraf-Dehkordi, S., Dehghani-Pilehvarani, A., and Tabei, F. (2010). A remote elderly monitoring system with localizing based on wireless sensor network. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 2, pages V2–553–V2–557.

[137] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49.

[138] Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press.

[139] Schmidt, R., Heindl, B., Pollwein, B., and Gierl, L. (1996). Abstractions of data and time for multiparametric time course prognoses. In *Advances in Case-Based Reasoning*, volume 1168 of *LNCS*, pages 377–391. Springer Berlin Heidelberg.

[140] Simpson, Jr., R. L. (1985). *A Computer Model of Case-based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation (Analogy, Machine Learning, Conceptual Memory)*. PhD thesis.

[141] Skalak, D. and Rissland, E. (1992). Arguments and cases: An inevitable intertwining. *Artificial Intelligence and Law*, 1(1):3–44.

[142] Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197.

[143] Smiti, A. and Elouedi, Z. (2011). Article: Overview of maintenance for case based reasoning systems. *International Journal of Computer Applications*, 32(2):49–56.

[144] Smiti, A. and Elouedi, Z. (2014). WCOID-DG: An approach for case base maintenance based on weighting, clustering, outliers, internal detection and dbsan-gmeans. *Journal of Computer and System Sciences*, 80(1):27 – 38.

[145] Smyth, B. (1998). Case-base maintenance. In *IEA/AIE (Vol. 2)*, pages 507–516.

[146] Smyth, B. and Keane, M. (1995). Remembering to forget - a competence-preserving case deletion policy for case-based reasoning systems. In *IJCAI-95 - Proceedings Of The Fourteenth International Joint Conference On Artificial Intelligence, Vols 1 And 2*, International Joint Conference On Artificial Intelligence, pages 377–382.

[147] Smyth, B. and Mckenna, E. (1999). Building compact competent case-bases. In *Case-Based Reasoning Research And Development*, volume 1650 of *LNAI*, pages 329–342.

[148] Smyth, B. and McKenna, E. (2001a). Competence guided incremental footprint-based retrieval. *Knowledge-Based Systems*, 14(3-4, Sp. Iss. SI):155–161.

[149] Smyth, B. and McKenna, E. (2001b). Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249.

[150] Sormo, F., Cassens, J., and Aamodt, A. (2005). Explanation in case-based reasoning - perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143.

[151] Stahl, A. and Roth-Berghofer, T. (2008). Rapid prototyping of cbr applications with the open source tool mycbr. In *Advances in Case-Based Reasoning*. Springer Berlin Heidelberg.

[152] Stefanov, D., Bien, Z., and Bang, W.-C. (2004). The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12(2):228–250.

[153] Sun, Z., Finnie, G., and Weber, K. (2004). Case base building with similarity relations. *Information Sciences*, 165(1–2):21 – 43.

[154] Suryadevara, N., Mukhopadhyay, S., Wang, R., and Rayudu, R. (2013). Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Engineering Applications of Artificial Intelligence*, 26(10):2641 – 2652.

[155] Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., and Rodríguez-Roda, I. (2005). An approach for temporal case-based reasoning: Episode-based reasoning. In *Case-Based Reasoning Research And Development*, volume 3620 of *LNCS*, pages 465–476. Springer Berlin Heidelberg.

[156] Tabar, A. M., Keshavarz, A., and Aghajan, H. (2006). Smart home care network using sensor fusion and distributed vision-based reasoning. In *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, VSSN '06, pages 145–154. ACM.

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

[158] Tsang, E. and Wang, X. (2005). An approach to case-based maintenance: Selecting representative cases. *International Journal Of Pattern Recognition And Artificial Intelligence*, 19(1):79–89.

[159] Valipour Shokouhi, S., Skalle, P., Aamodt, A., Sormo, F., et al. (2009). Integration of real-time data and past experiences for reducing operational problems. In *International Petroleum Technology Conference*.

[160] Vila, L. (1994). A survey on temporal reasoning in artificial intelligence. *AI Commun.*, 7(1):4–28.

[161] Vrotsou, K. (2010). *Everyday mining: Exploring sequences in event-based data*. PhD thesis, Department of Science and Technology Linkoping University.

[162] Vrotsou, K. and Forsell, C. (2011). A qualitative study of similarity measures in event-based data. In *Human Interface and the Management of Information. Interacting with Information*. Springer Berlin Heidelberg.

[163] Wang, L., Gu, T., Tao, X., Chen, H., and Lu, J. (2011). Recognizing multi-user activities using wearable sensors in a smart home. *Pervasive and Mobile Computing*, 7(3):287 – 298. Knowledge-Driven Activity Recognition in Intelligent Environments.

[164] Watson, I. (1998). Is CBR a technology or a methodology? In *Tasks and Methods in Applied Artificial Intelligence*, volume 1416 of *LNCS*, pages 525–534.

[165] Weber, B., Wild, W., and Breu, R. (2004). Cbrflow: Enabling adaptive workflow management through conversational case-based reasoning. In *Advances in Case-Based Reasoning*, volume 3155 of *LNCS*, pages 434–448. Springer Berlin Heidelberg.

[166] WHO, NIA, and NIH (2011). *Global health and ageing*. NIH Publications. WHO.

[167] Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions On Systems Man And Cybernetics*, SMC2(3):408–&.

[168] Wilson, D. and Martinez, T. (1997a). Improved heterogeneous distance functions. *Journal Of Artificial Intelligence Research*, 6:1–34.

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

[170] Wilson, D. R. and Martinez, T. R. (1997b). Instance pruning techniques. In *Machine Learning: Proceedings Of The Fourteenth International Conference (Icml'97)*, pages 404–411. Morgan Kaufmann.

[171] Wiratunga, N., Craw, S., Taylor, B., and Davis, G. (2004). Case-based reasoning for matching smarthouse technology to people's needs. *Knowledge-Based Systems*, 17:139 – 146.

[172] Wolf, M. and Petridis, M. (2008). Measuring similarity of software designs using graph matching for CBR.

[173] Wongsuphasawat, K., Plaisant, C., Taieb-Maimon, M., and Shneiderman, B. (2012). Querying event sequences by exact match or similarity search: Design and empirical evaluation. *Interact. Comput.*, 24:55–68.

[174] Wongsuphasawat, K. and Shneiderman, B. (2009). Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*.

[175] Yandell, B. (1997). *Practical Data Analysis for Designed Experiments*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

[176] Yu, X. and Gen, M. (2010). *Introduction to Evolutionary Algorithms, Decision Engineering*. Springer.

[177] Zarka, R., Cordier, A., Egyed-Zsigmond, E., Lamontagne, L., and Mille, A. (2013). Similarity measures to compare episodes in modeled traces. In *Case-Based Reasoning Research and Development*, volume 7969 of *LNCS*, pages 358–372. Springer Berlin Heidelberg.

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

Appendix A

A library for Case-base Maintenance: CELSEA

In this appendix, we present CELSEA[1] (CasE Library SElection Api), a framework for experimenting and evaluating case selection methods. This framework has a triple objective. First, CELSEA provides a simple infrastructure for developing CBM algorithms. Second, CELSEA includes a catalogue of implemented methods to experiment with case-bases and databases. Finally, CELSEA implements the $\alpha\beta$ evaluation proposed in section 4.3 (see figure 4.1). CELSEA is a Java library composed of five main modules (see figure A.1): *distance-classifier, cbr, case selection, evaluation* and *test*. The *distance-classifier module* implements a K-NN classifier and its catalogue of distance functions. The *CBR module* includes the case structure, the case-base and its efficient memory representation (*light memory*). This module also includes a functionality for experiment reporting in spreadsheet format (*report*). The *case selection module* is a catalogue of the following well known CBM algorithms: CNN, RNN, ENN, RENN, All-KNN, Shrink, IB2, IB3, ICF, DROP1, DROP2, DROP3,COV_RNN, RFC_CNN, RC_CNN, COV_CG, RFC_CG and RC_CG (see chapter 2). The *evaluation module* implements the Cross-Validation engine, α and β loops, etc. Finally, the *test module* includes different evaluation implementations, such as the evaluation $\alpha\beta$.

CELSEA can be used for development and experimental purposes. Experiments using CELSEA can be made using command lines providing the case-base file (in *CSV* format), a CBM file (in *XML* format) and the configuration of their parameters (in *XML* format). The CBM file consists of a list of the CBM algorithms to be analysed and their parameters configuration. Finally, the configuration file includes the implementation path to the CBM (σ), the database meta-information, and the values of the parameters described in the evaluation methodology proposed, that is: α , β, the number of partitions (or $folders$), and the classifier configuration.

[1] CELSEA project http://perseo.inf.um.es/~aike/celsea

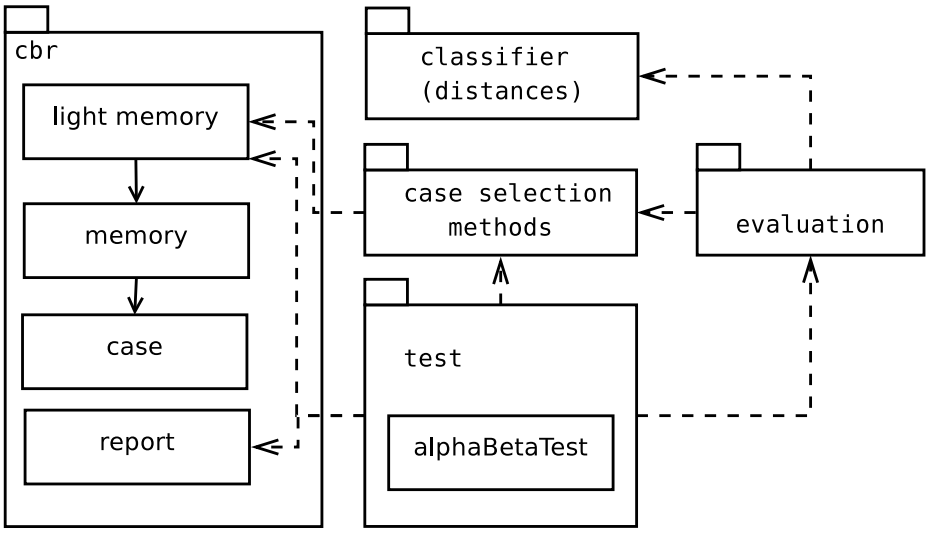


Fig. A.1 CELSEA framework: main package structure.

Appendix B

Implementation of Similarity Measures for Event Sequences in *myCBR*

The work presented in this appendix has been done in collaboration with Thomas Roth-Berghofer and Christian Sauer of the University of West London.

Within the CBR community there already exists other work on how to make use of temporal representations, such as similarity between workflows using an edit distance [112], graph matching to supervise business workflows [78] and to find software for reusing [172], and even the use of episodes (sequence of cases) in order to supervise a wastewater treatment process [155]. However, while there are approaches to compute the similarity between two event sequences [59, 114, 123, 161, 162, 173], only very few or almost none of these approaches were ever implemented in a software, which in turn is making it difficult to select the most suitable event sequence similarity measure for a specific software system.

For this reason our work presented in this paper focuses on the implementation of similarity measures between event sequences, so an actual implemented CBR system can be employed in the problem domains described before. For this purpose we are extending the *myCBR* development software to implement the event sequence similarity measures. The main reason for our choice is that the *myCBR* software is intended for building prototypical CBR applications in teaching, research, and small industrial projects with low effort [151]. Furthermore, it is designed for being easily extended with new functionalities.

The key motivation for implementing and further developing the open source software *myCBR* was and still is the need for a compact and easy-to-use tool for building prototypical CBR applications. The use of the *myCBR* tool especially aims for the sectors of teaching, research and small industrial projects with low initial development effort. To allow for the rapid prototyping of CBR systems, *myCBR Workbench* provides graphical user interfaces for modelling attribute-specific similarity measures and for evaluating the resulting retrieval quality in an integrated retrieval interface. In order to further reduce the development effort of CBR sys-

tems, especially the built up of a case-base, after defining an appropriate case representation, *myCBR Workbench* includes tools for generating the case representation automatically from existing raw data as well as importing cases from CSV data files. The accompanying Software Development Kit (SDK) allows for the integration of the developed CBR systems into other applications and allows also for the implementation of extensions into the SDK based on specific requirements such as additional similarity calculations, as it is demonstrated in the present research work.

myCBR provides a workbench, the *myCBR Workbench*, that supports the creation and maintenance of the vocabulary and similarity measures used within a CBR system and thus provides control over the knowledge model of a CBR system developed or maintained using *myCBR*. The *myCBR Workbench* is implemented as using the *Rich Client Platform* (RCP) of the *Eclipse* JDE. *myCBR Workbench* offers two different perspectives on a CBR system being developed - a perspective for the knowledge modeling and another perspective to develop and maintain the case-base(s). The conceptual idea behind the modeling perspective is that a case structure is created first being followed by the definition of the vocabulary and the creation of individual local similarity measures for each attribute. A description of a *concept* can contain one or more attribute descriptions as well as references to other concepts, hence allowing the user to create object-oriented like case representations. A description of an *attribute* can be formulated using one of the following data types: Boolean, Double, Integer, Interval, String or Symbol. For each data type *myCBR* provides a default similarity function that supports their definition and individual functionalities to further define more sophisticated similarity functions with regard to the specific data types chosen for an attribute. To ease the prototyping of CBR systems one single attribute description can have more than one similarity measure, allowing for rapid testing and experimentation to find the most suitable similarity functions.

As already described the modelling of the similarity measures in *myCBR Workbench* takes place first on the attribute level, defining the local similarity measures and then moves on to the concept level to define the global similarity measure(s) of a case representation. The attributes are then defined with their data types and value ranges. Depending on the chosen data type of an attribute the *myCBR Workbench* provides modelling GUI's for: Numerical data, providing predefined distance functions along with predefined similarity behaviour (constant, single step or polynomial similarity in-/de-crease). For the definition of similarity measures for symbolic values, *myCBR Workbench* provides table functions and taxonomy functions. A table function allows defining individual similarity values for each value pair, while a taxonomy subsumes similarity values for subsets of values. Depending on the size of a vocabulary, table similarity measures are hard to maintain and taxonomies provide a more structured overview. For symbolic values, also set similarities are provided in order to compare multiple value pairs [151].

The present paper describes how *myCBR* can be extended in order to include a new data type such as event sequences, as well as two distances to compute the similarity between event sequences. Therefore the highly modularised code structure of the *myCBR* project was extended to allow for the implementation of the new attribute and the computation of similarities for it's new data type being event sequences.

The major goal of *myCBR* is to minimize the effort for building CBR applications that require knowledge-intensive similarity measures[151], so the structure and design of the *myCBR* software is ready for being extended with new functionalities. The software is implemented in Java, so it follows an object oriented approach. In this work we are only showing the set of classes that are related with the representation of the case-base and the processes to retrieve cases. The figure B.1 depicts the class diagram of the classes realising the retrieval process and the case representation in a case-base.

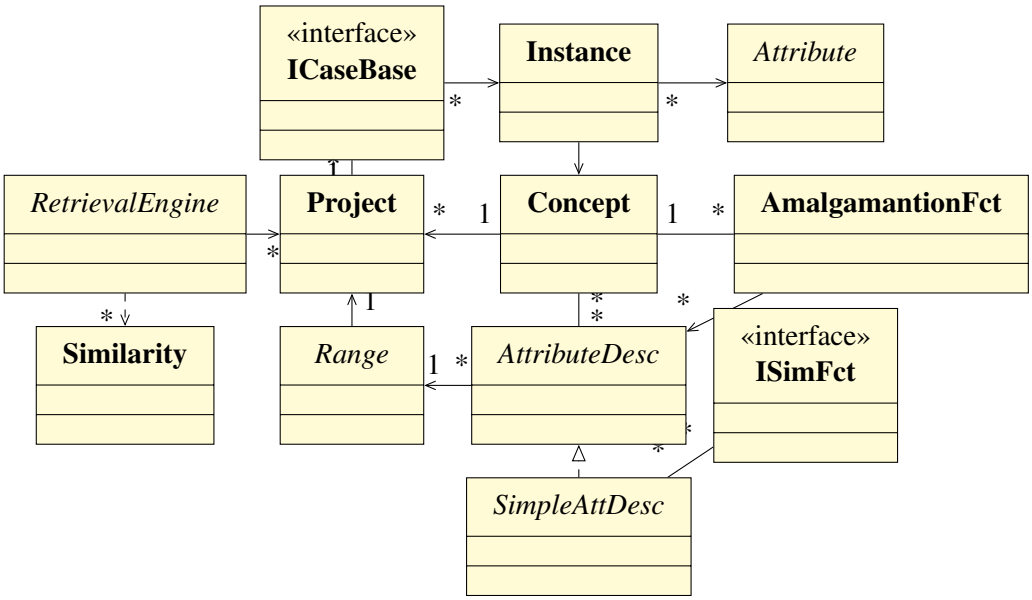


Fig. B.1 Main classes of the *myCBR* SDK related to the similarity computation process.

Where each class has its own role:

Project: A Project consists of a concept that defines the vocabulary of a project domain and one or more case bases storing problem cases from the domain.

Instance: Represents values occurring in either query and cases. It contains a set of attributes that conforms to the case structure.

Attribute: This is an interface with the basic operations of all types of attributes. Each attribute must have an implementation of this class. For instance, DoubleAttribute, StringAttribute, etc.

ICaseBase: An interface that represents the set of retained cases.

RetrievalEngine: Implementation of the process to retrieve the most similar cases to a query.

Concept: Vocabulary of the problem domain.

Range: Possible values of a particular implementation of an attribute.

AttributeDesc: Contains details about the description of one attribute of the case. For instance, there are descriptions for numbers, symbols, dates, etc.

SimpleAttDesc: This class extends the AttributeDesc class with new fields and methods for being used by a ISimFct.

ISimFct: Implementation of the similarity between two attributes with the same description, i.e. between attributes of the same data-type.

AmalgamationFct: Implementation of the global similarity function to compute the similarity between two instances / cases, considering all the existing ISimFct to create a Similarity object.

Similarity: The value that represent the similarity between two instances.

Our approach was the creation of a new attribute that represents event sequences. From the definitions given before in chapter 5, we have set the following string representation:

$$sec =< (concept, timestamp) \ldots >$$

The figure B.2 depicts the new classes that we have implemented. With them we can represent an event sequence within as an attribute of a case which allows *myCBR* to compute the similarity between two different event sequences.

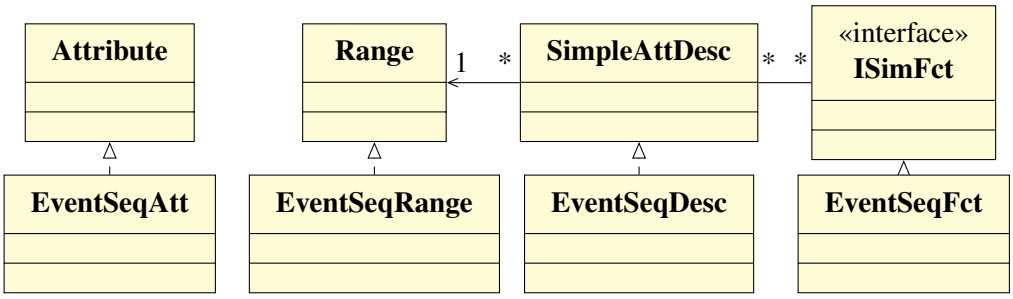


Fig. B.2 Sequence of operations to compute the similarity between two cases.

EventSeqAtt: Contains the information of a particular event sequence.

EventSeqRange: Contains the possible values of event sequences. It is responsible of creating new *EventSeqAtt*.

EventSeqDesc: Contains the details about how the event sequences are implemented.

EventSeqFct: This corresponds to the implementation of the similarity measures that we have worked on in the present paper. When an object of this class is created it is mandatory to specify which type of measure it should employ.

The implementations of the measures are expressed follow the algorithms 18 and 19 introduced in chapter 5.

Appendix C

Resumen en español

Contexto

Esta tesis ha sido desarrollada dentro del grupo Artificial Intelligence and Knowledge Engineering Group (AIKE) de la Universidad de Murcia (España) y ha sido financiada por la beca de doctorado BES-2010-030294 dentro del proyecto AI-SENIOR (TIN2009-14372-C03-01) del Ministerio de Ciencia e Innovación. Parte de este trabajo también ha sido financiado por el proyecto DAISY (15277/PI/10) de la Fundación Séneca para la Investigación de la Región de Murcia (España). Ambos proyectos centran su atención en el desarrollo de un sistema para la detección de patrones regulares y de escenarios anormales en un entorno Smart Home y la monitorización de personas mayores en el hogar.

Los proyectos AI-Senior y DAISY tienen como objetivo ofrecer una herramienta que ayude en la asistencia y cuidado de las personas mayores que viven solas, de manera que se pueda facilitar una respuesta rápida de las unidades de emergencia en caso de un accidente en el hogar. En cuanto a sus ventajas y beneficios, el sistema ofrecerá la posibilidad a sus usuarios de vivir más tiempo en su hogar de manera independiente, y podría reducir los costes de los servicios sociales y de salud pública que tienen entre sus competencias el cuidado de las personas mayores.

Motivación

El Razonamiento Basado en Casos (CBR de sus siglas en inglés, Case-Based Reasoning) es una metodología para la resolución de problemas por analogía con problemas anteriormente resueltos. La base del CBR son los *casos*, piezas independientes de conocimiento que encapsulan la asociación entre un problema y su solución, las cuales se agrupan en una base de conocimiento conocida como *base de casos*. Una de las características más importantes de CBR es que el proceso de aprendizaje es continuo e incremental, dado que a medida que

nuevos problemas son solucionados, nuevos casos son creados y almacenados en la base de casos, incrementando el conocimiento en el dominio.

Sin embargo, el almacenamiento de un mayor número de casos no es garantía de una mejor capacidad para la resolución de problemas. De hecho, en ocasiones el rendimiento del sistema puede decaer, e incluso provocar un empeoramiento de la capacidad del sistema en la resolución de problemas. Por este motivo, es esencial encontrar un equilibrio entre el número de casos almacenados que permita cubrir suficientemente el dominio. De acuerdo a [169], el Mantenimiento de Base de Casos (CBM de sus iniciales en inglés, Case-Base Maintenance) consiste fundamentalmente en reducir el tamaño de la base de casos sin afectar negativamente la capacidad en la resolución de problemas del proceso de razonamiento [126]. Los beneficios de su uso radican en una simplificación del proceso de resolución de problemas, acelerando la recuperación de casos similares y un posible incremento de la tasa de aciertos en la resolución de problemas, ya que básicamente las técnicas de CBM se centran en la eliminación de casos erróneos u obsoletos.

En particular, las técnicas para CBM han sido estudiadas en profundidad por la comunidad CBR [18, 53, 107, 126, 146, 167, 169]. Para mejorar el entendimiento y compresión de las técnicas CBM, se han propuesto distintas clasificaciones. Una de esas clasificaciones se basa en si el algoritmo CBM es determinista o no. Cuando el algoritmo es determinista, éste siempre produce la misma base de casos mantenida a partir de la base de casos original. En caso contrario, cuando el algoritmo genera diferentes bases de casos mantenidas, el algoritmo CBM es considerado no determinista.

Un aspecto fundamental cuando se utilizan algoritmos CBM, es conocer si la base de casos mantenida ofrece unos resultados equivalentes o peores que los ofrecidos por la base de casos original. Para responder a esta pregunta se suelen utilizar métodos de evaluación clásicos, como Cross-Validation o Hold-Out. Sin embargo, dichos métodos no toman en consideración la singularidad del proceso de reducción de casos realizada por el algoritmo CBM, dando como resultado unas estimaciones de tasa de aciertos que no tienen suficiente evidencia estadística que los apoyen.

Desde los inicios del CBR y hasta hace relativamente poco tiempo, estos estaban enfocados en la resolución de problemas que no eran dependientes del tiempo, donde las descripciones de los problemas representan observaciones estáticas de un problema resuelto. Sin embargo, en la última década la inclusión de características temporales en los casos está ganando un papel relevante, y muestra de ello es la cantidad de trabajos que han sido publicados al respecto [16, 59, 74, 78, 111, 115, 116, 159]. Un elemento en común de estos trabajos es el *caso temporal*, es decir, un problema resuelto en el que explícitamente se representa la dimensión temporal.

En particular, dentro de los proyectos AI-Senior y DAISY, la identificación correcta del

tipo de actividad desarrollada en el hogar necesita de una representación espacio temporal que ayude a registrar el orden en el que son visitadas las diferentes estancias de una casa. De esta forma, los casos necesitan una representación temporal para modelar esta información. Entre los esquemas de representación que han sido utilizados para representar casos temporales caben destacar las series temporales series temporales [76, 115, 117, 118, 122], los flujos de trabajo [14, 21, 26, 27, 43, 78, 111, 112, 165], los episodios [33, 56] y las secuencias de eventos [59, 60, 73, 100].

Al utilizar esquemas de representación temporal dentro de los casos, el cálculo de la función de similitud se hace más complejo, incrementando la complejidad global del proceso de recuperación de casos. Como consecuencia, las bases de casos temporales sufren problemas de rendimiento conforme el tamaño de la base de casos se va incrementando, es decir, cuantos más casos son almacenados en la base de casos, el proceso de encontrar y recuperar los casos más similares se hace más complejo. Por este motivo, al introducir casos temporales, el proceso de reducción del tamaño de la base de casos adquiere una mayor importancia.

Sin embargo, a nuestro entender, se ha prestado poca atención a los efectos y consecuencias que tiene aplicar un algoritmo CBM sobre un sistema CBR que está diseñado para trabajar con casos temporales. De hecho, de momento no se ha publicado ningún trabajo relacionado con mantenimiento de bases de casos temporales.

Uno de los objetivos de esta tesis es el desarrollo de una representación temporal que pueda ser utilizada dentro del proyecto AI-Senior. De esta manera, la representación propuesta modelará las actividades diarias llevadas a cabo por las personas dentro del hogar, y los casos de la base de casos ayudarán a la clasificación de la actividad registrada como peligrosa o normal.

Además, hay que tener en cuenta que los métodos tradicionales de evaluación, como Cross-Validation y Hold-Out) no cuantifican correctamente las capacidades de resolución de problemas de un sistema CBR después de realizarle un mantenimiento. El motivo principal es que los algoritmos CBM pueden reducir drásticamente el tamaño de una base de casos, de forma que los conjuntos de casos de prueba creados por los métodos de evaluación a partir de la base de casos mantenida, podrían no cubrir el dominio de problemas; es decir, la competencia del sistema CBR se vería reducida. Por otra parte, si tenemos en cuenta que cuando se utiliza un algoritmo CBM no determinista, como los descritos en [5, 70, 81, 97], los distintos conjuntos de casos seleccionados presentan un variabilidad más alta que en el caso de los métodos deterministas, siendo necesaria una evidencia estadística más alta que la propuesta por los métodos de evaluación tradicionales para ofrecer unos resultados de evaluación más confiables. Por ello, otro de los objetivos de esta tesis consiste en el desarrollo de una metodología de evaluación que cuantifique con una suficiente evidencia estadística las consecuencias de aplicar un algoritmo CBM al sistema CBR propuesto para la monitorización de personas mayores en el

hogar.

Hipótesis y objetivos

Dada una representación de casos temporales, nuestra hipótesis inicial es que los algoritmos CBM existentes pueden ser adaptados para trabajar con bases de casos temporales. A partir de esta hipótesis, la tesis persigue los siguientes objetivos:

1. Realizar una revisión bibliográfica en profundidad para analizar los trabajos que han tratado el problema del CBM con anterioridad.
2. El análisis y propuesta de algoritmos CBM no deterministas, con el fin de estudiar el comportamiento de este tipo de algoritmos. El elemento original de la propuesta radica en modelar el problema de CBM como un problema de optimización multiobjetivo.
3. Proponer una metodología de evaluación para algoritmos deterministas y no deterministas, así como analizar los algoritmos CBM más extendidos con la evaluación propuesta.
4. Proponer algoritmos CBM temporales a partir de los algoritmos CBM tradicionales.
5. Estudio de la idoneidad del uso de algoritmos CBM temporales en un caso práctico. En concreto el mantenimiento de un sistema CBR para la motorización de ancianos independientes que viven solos en su hogar.

Resultados y conclusiones

Como resultado de la investigación realizada durante la tesis, hemos realizado un estudio exhaustivo de las propuestas existentes para la realización de CBM, pudiendo concluir que no existen trabajos previos sobre CBM cuando tratamos con bases de casos temporales. Adicionalmente, hemos modelado el mantenimiento de una base de casos como un problema de optimización multiobjetivo. Entre los criterios a optimizar están el número de casos, la proporción de casos redundante y reducción de una estimación del error que podría dar el sistema CBR. Consecuentemente con la revisión bibliográfica realizada, podemos ofrecer las siguientes conclusiones:

Cuando se enfoca el CBM como un problema de optimización multiobjetivo con un Algoritmo Evolutivo, la tasa de aciertos alcanzadas con las bases de casos mantenidas son significativamente iguales a la obtenida por la base de casos original. Además, la reducción de la base de casos es proporcional a la cantidad de casos redundante en la base de casos, evitando un problema de sobre aprendizaje.

Los métodos de evaluación tradicionales no ofrecen suficiente soporte estadístico para cuantificar con confianza los efectos que tienen los algoritmos CBM en un sistema CBR. Para

solucionar este problema, hemos propuesto el método de evaluación $\alpha\beta$, el cual es capaz de ofrecer resultados confiables con un alto grado de confianza.

Los experimentos llevados a cabo apoyan nuestra hipótesis de que los métodos Cross-Validation and Hold-Out no son suficientes para ofrecer resultados confiables cuando los métodos CBM son evaluados, ya sean algoritmos deterministas o no deterministas. Por un lado, de acuerdo a nuestros experimentos, nuestro método $\alpha\beta$ y otras técnicas propuestas [126] son adecuados para evaluar algoritmos CBM deterministas. Sin embargo, $\alpha\beta$ es el único método de evaluación capaz de ofrecer resultados confiables cuando se evalúa un método no determinista.

Al igual que ocurre con las bases de casos no temporales, un sistema CBR temporal podría sufrir de problemas de rendimiento cuando la cantidad de casos almacenada es alta. Para paliar este problema, en esta tesis hemos presentado un conjunto de algoritmos CBM temporales: t-CNN, t-RENN, t-DROP1, t-ICF and t-RC-FP. A la luz de los resultados obtenidos en las experimentaciones, llegamos a la conclusión de que es posible realizar tareas de mantenimiento de manera exitosa cuando el sistema CBR trabaja con casos temporales. Sin embargo, el conocimiento representado dentro de los casos temporales puede afectar a la tasa de aciertos del sistema CBR cuando resuelve problemas. Hay que destacar no obstante, que sólo los factores no temporales, como el tamaño de la base de casos y la proporción de casos redundantes y ruidosos, pueden afectar al resultado de un algoritmo CBM temporal.

Finalmente, en cuanto a nuestra propuesta para detectar situaciones anormales en un hogar donde vive una persona mayor sola, los experimentos que hemos realizado demuestran su capacidad para detectar satisfactoriamente los diferentes escenarios propuestos de peligrosidad. Sin embargo, para una correcta detección se requiere una base de casos con muchos casos almacenados, lo cual repercute en la complejidad del proceso de razonamiento y confirma la necesidad de la utilización de algoritmos CBM temporales para reducir su tamaño.

Cuando realizamos la tarea de mantenimiento con los algoritmos que hemos propuesto, los resultados experimentales también muestran que es posible crear bases de casos con propiedades similares a la base de casos original. Sin embargo, el éxito de la creación de estas bases de casos temporales depende del número de casos en la base de casos original. Es decir, cuantos más casos hay en la base de casos mayor será la probabilidad de que el algoritmo CBM temporal genere una base de casos mantenida similar a la original.