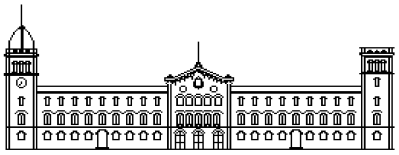

**Treball final de grau**

**GRAU DE MATEMÀTIQUES**

**Facultat de Matemàtiques**
**Universitat de Barcelona**

---

# PLATAFORMA DE RAONAMENT BASAT EN CASOS PER A LA GESTIÓ DEL CONEIXEMENT

---

**Irene Martí Diéguez**

Directora: Maria Salamó Llorente
Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. UB

Barcelona, 24 de juny de 2014

# Agraïments

M'agradaria agrair a la meva tutora del Treball de Final de Grau de Matemàtiques, la professora Maria Salamó Llorente, tot el seu suport, la seva dedicació i paciència durant la realització d'aquest treball. Sense la seva col·laboració aquest projecte no hauria estat possible.

# Índex

# Índex de figures

# Índex d'algorismes

# Índex de taules

# Capítol 1

# Introducció

Case-Based Reasoning, **CBR**, is a kind of reasoning based on solving new problems using solutions obtained of previous problems.

As something natural, human beings use this case-based reasoning to face problems that arise every day. For instance: the mechanic that repairs a car remembering another car having the same problems is using **case-based reasoning**. That is why CBR is called an *expert system* since is a system that tries to imitate the behavior of an expert human being, that is to say, it follows the same steps that a human being would take trying to solve different problems. Continuing with the example of the mechanic, an expert system in charge of diagnosing mechanical car problems would follow the same steps that a mechanic does.
In this way, the advantages of creating a CBR are clear: to make easier experts' task use CBR as a help system on decision making.

In addition, human beings do something more: we update our knowledge; that is to say, we forget useless information and we learn and remember useful one. This is as important to experts systems as it is to human beings.
On the one hand, an expert system that does not get new information is a high quality system nowadays but it will become obsolete in the future. On the other hand, an expert system that does not forget information and keeps on getting more and more will produce a huge database, excessively big to be processed in order to solve new problems or a *mediocre expert* system by means of cutting back the amount of information used.

## 1.1 Motivació i àmbit del projecte

Al fer el Grau en Matemàtiques vaig començar a relacionar les matemàtiques i la informàtica, així vaig descobrir que hi ha molts nexes en els dos graus que m'agraden. Per això vaig fer la menció en Informàtica.

Aquest treball se centra en l'àrea de l'**aprenentatge artificial, AA**, que té una base molt àmplia de matemàtiques i informàtica. Dins d'aquest món de l'$AA$, el treball està enfocat als sistemes de CBR. És a dir, tracta de *sistemes experts* que són capaços de trobar la solució d'un problema comparant-lo amb altres problemes que formen part d'una base de coneixement ja existent. Aquest tipus de sistemes van començar a ser estudiats a principis de la dècada dels 80 [I]. Com veurem més endavant en els sistemes CBR hi ha quatre fases fonamentals:

[I] Va ser el professor Roger Schank i els seus estudiants de la universitat de Yale els primers en tractar els sistemes CBR.

**Fase 1. Recuperar** el problema o problemes que són més similars al problema que estem tractant
**Fase 2. Reutilitzar** la informació i el coneixement que tenim dels problemes anteriors per resoldre el problema actual
**Fase 3. Revisar** la solució proposada
**Fase 4. Retenir** les parts de l'experiència actual que ens poden ser útils per la resolució de futurs problemes

Des del principi s'han treballat de manera exhaustiva les tres primeres fases, això ha donat pas a la creació de bons sistemes de CBR. Però la fase de *Retenir* no va ser tan estudiada. Calia guardar problemes nous per mantenir la base de coneixement actualitzada, però això feia que la seva mida augmentes cada cop més, amb els problemes que això comportava. Va ser a mitjans dels anys 90 quan la fase *Retenir* va començar a despertar més interès. Per una banda van començar a plantejar-se si tots els problemes que es guardaven realment calien ser guardats, potser s'afegia un problema que simplement donava informació ja existent. Per una altra banda van pensar que per mantenir la base de coneixement actualitzada, però dintre d'uns límits coherents de mida, calia oblidar problemes, és a dir, estudiar els problemes menys útils i eliminar-los de la base de coneixement.
Així es van començar a plantejar els *Problemes de Classificació*, que es basen en decidir quins problemes cal guardar i quins no (*Mètodes de Retenir, "Retain"*) i quins cal esborrar (*Mètodes d'Oblidar, "Forget"*).

Els sistemes de CBR són "relativament nous", això fa que encara quedi molta feina a realitzar per construir un sistema de CBR amb el màxim rendiment. Dins de tots els àmbits que hi ha en els sistemes CBR, el meu treball intenta acostar-nos a aquest objectiu mitjançant l'estudi dels *Problemes de Classificació*.

## 1.2 Objectius

L'objectiu principal del treball és estudiar diferents tècniques pel manteniment de la base de coneixement d'un sistema de CBR per tal d'aconseguir millorar el rendiment del CBR.
El manteniment de la base de coneixement es basa en dues fases fonamentals:

- La **Fase de Retenir** és la fase que s'encarrega de decidir quins problemes guardar en la base de coneixement i quins no són necessaris
- La**Fase d'Oblidar** és la fase que s'encarrega de decidir quins problemes es poden eliminar de la base de coneixement perquè es consideren que són perjudicials o no necessaris

En aquest treball es fa una descripció acurada i s'implementen diferents mètodes de *Retenir* i *Oblidar*. A més a més, es fa una comparativa usant diferents mesures de referència (com el número de problemes de la base de coneixement o la precisió obtinguda a l'executar cada mètode) que ens ajuden a decidir quins són els mètodes que milloren el sistema de CBR.

## 1.3 Organització de la memòria

L'estructura de la memòria és la següent: En el Capítol 2 s'explica més profundament que és un sistema de CBR. En el Capítol 3 s'explica el manteniment de la base de casos i es detallen una sèrie d'estratègies per *retenir* i *oblidar*. En el Capítol 4 s'analitzen els resultats obtinguts. I el Capítol 5 acaba amb la conclusió i es proporciona una visió general del treball futur.

En l'annex 1 es detalla la implementació dels mètodes no explicats en el Capítol 4. I en l'annex 2 s'explica com executar el codi.

# Capítol 2

# Raonament basat en casos

En aquest capítol s'explica el que és un *sistema de Raonament Basat en Casos* i es veuen les seves aplicacions.

## 2.1 Definició de sistema de CBR

Per explicar que és un sistema de CBR es pot començar veient la definició formulada per Riesbeck i Shank l'any 1989 [21]:

> *"A case-based reasoner solves new problems by adapting solutions that were used to solve old problems"*

El **Raonament Basat en Casos** proposa que, a mesura que es resolen nous casos d'un tipus de problema donat, el sistema de resolució d'aquests problemes vagi aprenent de la seva pròpia experiència, i aquesta nova informació apresa s'usarà per resoldre casos posteriors. Cal destacar que aquesta integració no es troba normalment en la majoria dels mètodes d'aprenentatge artificial, com l'aprenentatge inductiu, l'aprenentatge basat en explicacions o l'aprenentatge evolutiu.

Els sistemes de CBR es basen en una unitat mínima anomenada **cas**, per tant, és interessant veure la seva definició:

> *"A case is a piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner"* [27]

Un cas és un conjunt d'**atributs** (o *característiques*) que descriuen una experiència anterior ja viscuda, per tant, es pot considerar el cas com a cert. Un conjunt de casos formen la **base de casos** (o *base de coneixement*). S'usa aquesta base de casos per aconseguir els objectius del raonament, és a dir, que el sistema expert arribi a una conclusió per donar una solució.

El **Raonament Basat en Casos** és un paradigma de resolució de problemes capaç d'utilitzar el coneixement adquirit en experiències anteriors. Un problema nou es resol buscant un problema passat, igual o similar, i utilitzant la seva solució en el nou problema. A més, els sistemes de CBR retenen la nova experiència, si cal, per tal de ser utilitzada en problemes futurs [1]. En la terminologia de CBR, un **cas** és una experiència anterior que ha estat apresa, de manera que pot ser usada en la resolució de problemes en el futur, es coneix com un *cas passat*, un *cas anterior*, un *cas emmagatzemat* o un *cas retingut*. I un *cas nou* o un *cas sense resoldre* és la descripció d'un nou problema a resoldre.

Una característica molt important en el Raonament Basat en Casos és la seva connexió amb l'**aprenentatge**. Un sistema de CBR no és només un mètode particular de raonament, també és un paradigma d'aprenentatge que permet l'aprenentatge sostingut mitjançant l'actualització de la base de casos després de resoldre un problema. És a dir, l'aprenentatge en els sistemes de CBR es produeix com un subproducte natural de la resolució de problemes.

Les tasques principals que han de complir els mètodes d'un sistema de CBR són:

- La identificació del cas nou a resoldre
- Trobar un o més casos passats similars al nou
- Usar aquests casos recuperats per suggerir una solució pel cas actual
- Avaluar la solució que s'ha suggerit
- Actualitzar la base de casos, si s'ha retingut el nou cas

## 2.2 El cicle de CBR

El cicle de CBR definit per Aamodt & Plaza [1] consta de quatre fases:

1. **RECUPERAR** (*Retrieve*) el cas o els casos més similars,
2. **REUTILITZAR** (*Reuse*) la informació i el coneixement d'aquest cas per resoldre el nou problema,
3. **REVISAR** (*Revise*) la solució obtinguda, i
4. **RETENIR** (*Retain*) el nou cas per ser utilitzat en la resolució de problemes futurs.

A la Figura 2.1 s'il·lustra aquest cicle. Una descripció d'un problema inicial defineix un nou cas. Es recuperen un o més casos similars al cas nou de la base de casos. El cas recuperat es combina amb el nou cas mitjançant la reutilització, això dona lloc a un cas resolt, és a dir, es té una solució proposada pel cas inicial. Es comprova aquesta solució durant el procés de revisió, això dona lloc a una solució confirmada. I finalment, si el cas és útil per a resolucions futures es guarda en la base de casos. El cicle de CBR és iteratiu: el cicle s'aplica una vegada per cada nou cas a resoldre.

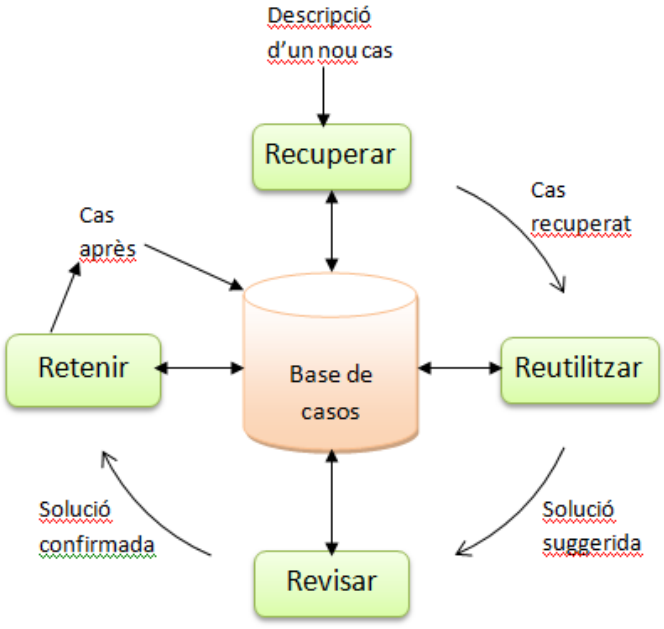


Figura 2.1: Fases del cicle de CBR

S'ha vist el cicle de CBR com un cicle de fases seqüencials. Per descriure encara més el

cicle de CBR canviem a una visió orientada a tasques, on cada pas, o subprocés, és vist com una tasca que el sistema de CBR ha d'assolir. En l'àmbit del coneixement, un sistema és vist com un agent que té objectius, i els mitjans per assolir aquests objectius.
Una descripció del sistema es pot realitzar des de tres perspectives: tasques, mètodes i models de coneixement del domini. Les tasques són creades pels objectius del sistema, i una tasca es porta a terme mitjançant l'aplicació d'un o més mètodes. Per tal que un mètode sigui capaç de realitzar una tasca, cal coneixement sobre el domini general, així com informació sobre el problema actual.

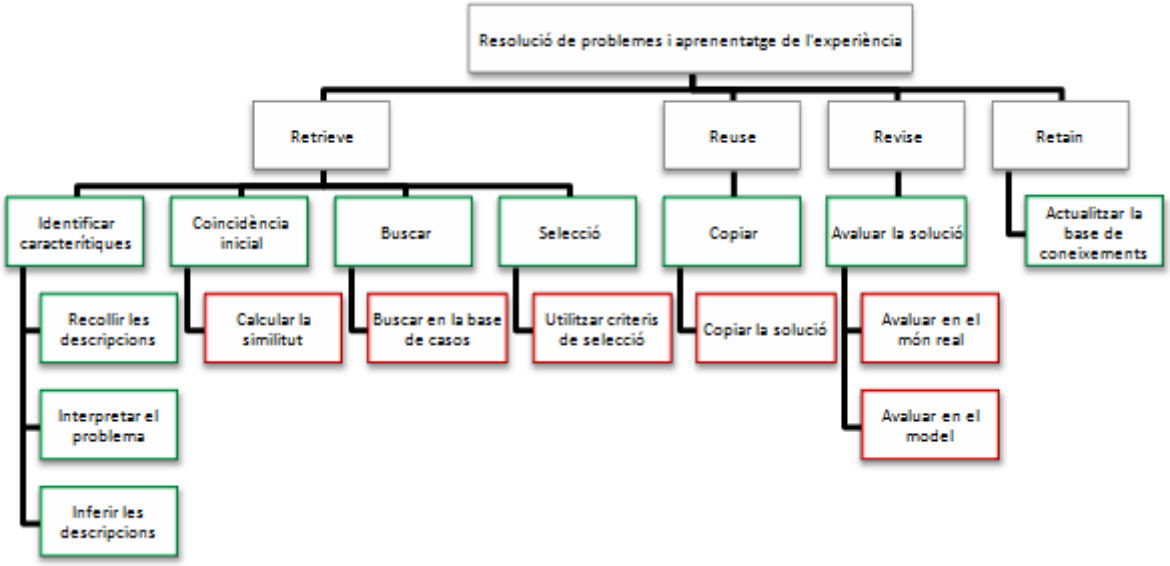


Figura 2.2: Tasques del cicle de CBR

Es veu la separació de tasques en la Figura 2.2. Les tasques tenen el requadre en verd, mentre que els mètodes estan en requadres vermells. La tasca de nivell superior és *"resolució de problemes i aprenentatge de l'experiència"* i el mètode per dur a terme aquesta tasca és el *Raonament Basat en Casos*. Es divideix la tasca de nivell superior en les quatre principals tasques corresponents als quatre processos del la Figura 2.1, *Retrieve*, *Reuse*, *Revise* i *Retain*. Aquestes quatre tasques són necessàries per dur a terme la tasca de nivell superior. Per la seva part, aquestes tasques estan dividides en subtasques que ajuden a complir l'objectiu de la tasca corresponent. Els mètodes especifiquen els algorismes que controlen les execucions de les subtasques i utilitzen els coneixements per realitzar aquestes accions.

No existeixen sistemes de CBR universals que siguin apropiats per qualsevol domini, sinó que són mètodes adaptats a la resolució de problemes i aprenentatge en dominis particulars. Juntament amb el model de tasques presentat anteriorment, els problemes relacionats amb la investigació del CBR poden ser agrupats en cinc àrees:

- Representació del coneixement
- Mètodes de recuperació
- Mètodes de reutilització
- Mètodes de revisió
- Mètodes de retenció

La primera àrea correspon a la forma de guardar el coneixement. I les altres àrees es corresponen al cicle de CBR que s'ha vist anteriorment. Veiem més detingudament cadascuna

d'aquestes 5 àrees.

### 2.2.1 Representació del coneixement

Un sistema de CBR depèn en gran mesura de l'estructura i el contingut de la seva base de casos. Recordem que la *base de casos* és un conjunt de casos, i cada *cas* es guarda amb la seva descripció (conjunt d'*atributs*) i amb la seva *solució*.
Com un nou cas es resol recordant un cas passat adequat, la recerca de casos i el processos de coincidència han de ser alhora eficaços i raonablement eficients en el temps.

### 2.2.2 Mètodes de recuperació, *Retrieve*

La tasca de *Retrieve* comença amb la descripció del nou cas a resoldre i s'acaba quan s'ha trobat el cas o casos ja resolts que més s'assemblen al nou cas. Per obtenir aquests casos similars s'usen les *Funcions de Distància* [17].

Tot i que hi ha diverses funcions de distància, la més utilitzada és la Funció de distància Euclidiana, que es defineix a la Funció 2.1 com:

$$ME(x,y) = \sqrt{\sum_{a=1}^{m} (x_a - y_a)^2} \tag{2.1}$$

on $x_a$ i $y_a$ són dos vectors d'entrada i $m$ és el número de variables d'entrada en l'aplicació.

Una funció alternativa és la Funció de distància Manhattan, que es defineix en la Funció 2.2 com:

$$MM(x,y) = \sum_{a=1}^{m} |x_a - y_a| \tag{2.2}$$

Les Funcions de distància Euclidiana i Manhattan són equivalents a la Funció de r-distància Minkowsky, amb r=2 i r=1 respectivament, que es defineix en la Funció 2.3 com:

$$M_r(x,y) = \sqrt[r]{\sum_{a=1}^{m} (x_a - y_a)^r} \tag{2.3}$$

Una debilitat d'aquestes funcions de distància és que si un dels atributs d'entrada té un rang relativament gran, pot dominar sobre els altres atributs. Per exemple, si una aplicació té dos atributs, $A$ i $B$, i $A$ pot tenir valors des de 1 fins a 1000, i $B$ té valors des de 1 fins a 10, aleshores, la influència de $B$ en la funció de distància serà anul·lada per la influència d'$A$. Per aquest motiu, les distàncies acostumen a estar normalitzades dividint la distància per cada atribut entre el seu rang, així les distàncies valdran entre 0 i 1.

Fins ara no s'ha mirat com són aquests atributs. Si són numèrics es poden aplicar les funcions de distància definides anteriorment. Però, què passa si els atributs són nominals? Una manera de treballar amb casos que tinguin tant atributs nominals com numèrics és usar una funció de distància heterogènia que utilitzi diferents funcions de distància pels diferents tipus d'atributs. Una aproximació que s'acostuma a utilitzar és la *Mètrica Heterogènia Euclidiana Superposada* (**HEOM**) [17].

La funció que defineix la distància entre dos valors nominals $x_a$ i $y_a$ d'un atribut $a$ és la Funció 2.4 .

$$d_a(x_a, y_a) = \begin{cases} 1, & \text{si } x_a \text{ o } y_a \text{ no tenen valor} \\ nom(x_a, y_a), & \text{si } a \text{ és un atribut nominal} \\ num(x_a, y_a), & \text{en cas contrari} \end{cases} \tag{2.4}$$

Els valors desconeguts dels atributs es treballen retornant una distància igual a 1. Les funcions per calcular la distància entre atributs nominals, *nom($x_a$,$y_a$)*, i la distància entre atributs numèrics, *num($x_a$,$y_a$)*, es defineixen en 2.5 i 2.6 respectivament.

$$nom(x_a, y_a) = \begin{cases} 0, & \text{si } x_a{=}y_a \\ 1, & \text{en cas contrari} \end{cases} \tag{2.5}$$

$$num(x_a, y_a) = \frac{|x_a - y_a|}{range_a} \tag{2.6}$$

On $max_a$ i $min_a$ són respectivament. el màxim i el mínim valor de l'atribut $a$.

Així, la distància entre dos vectors d'entrada $x_a$ i $y_a$ ve donada per la Funció 2.7.

$$HEOM(x, y) = \sqrt{\sum_{a=1}^{m} d_a(x_a, y_a)^2} \tag{2.7}$$

Si es generalitza amb la Funció Minkowsky s'obté la Funció 2.8.

$$HMOM(x, y) = \sqrt[r]{\sum_{a=1}^{m} d_a(x_a, y_a)^r} \tag{2.8}$$

On $d_a$($x_a$,$y_a$) és la Funció 2.4 i *num($x_a$,$y_a$)* es calcularà amb la distància corresponent.

### 2.2.3 Mètodes de reutilització, *Reuse*

La tasca de *Reuse* agafa el cas o els casos més propers obtinguts gràcies als *mètodes de recuperació* i es queda amb el cas més proper. La subtasca *"copiar"* agafa la classe solució del cas recuperat en la fase *Retrieve* i la copia en el nou cas com a solució d'aquest.

### 2.2.4 Mètodes de revisió, *Revise*

La tasca de *Revise* comprova la solució obtinguda gràcies als mètodes de recuperació i els de reutilització. En aquesta comprovació es pot obtenir èxit (al nou cas se li ha donat una solució que es considera correcte), o fracàs (la solució que s'ha donat al nou cas no es considera correcte). En el cas que s'obtingui un fracàs, es pot aprendre d'aquest fracàs en la fase següent, *Retain*.

### 2.2.5 Mètodes de retenció, *Retain*

La tasca de *Retain* és una de les tasques que engloben la finalitat d'aquest treball, és per això que es mira més extensament en el capítol 3.

## 2.3 Aplicacions existents

Les arrels del CBR es troben en els treballs de Roger Schank sobre memòria dinàmica i sobre el paper que tenen els records de situacions anteriors (casos) i els seus patrons (MOP's) en la resolució de problemes i el seu aprenentatge [23].

El primer sistema que es podria anomenar CBR va ser el sistema *"CYRUS"*, desenvolupat a la Universitat de Yale per Kolodner, component del grup de treball de Schank[12] [13]. Aquest sistema es basava en el model de memòria dinàmica de Schank i en la teoria MOP de resolució de problemes i el seu aprenentatge. Era bàsicament un sistema que responia preguntes i que tenia coneixement sobre diversos viatges i meetings del Secretari d'Estat d'Estats Units, Cyrus Vance.

Un altra base pel sistema de CBR va ser desenvolupada per Bruce Porter i el seu grup de la Universitat de Texas, Austin [4]. Van desenvolupar el sistema *"PROTOS"* [5], que integrava un coneixement de domini general i un coneixement específic en una estructura unificada de representació.

El treball realitzat per Edwina Rissland i el seu grup (entre els quals hi havia experts en Dret) de la Universitat de Massachusetts, Amhearst, tractava sobre el paper dels raonaments usats en sentències judicials [22]. Els casos, aquí *"precedents"*, s'utilitzaven per interpretar una situació d'un tribunal avaluant els arguments de les dues parts. Això va donar com a resultat el sistema "HYPO" [3].

A Europa, la investigació sobre els sistemes de CBR van començar una mica més tard que als Estats Units. Tot i això, el treball en el camp del CBR sembla haver sigut més fortament dirigit cap al desenvolupament de sistemes experts i investigació en adquisició de coneixements que als Estats Units. Entre els primers resultats es troba l'obra del sistema de CBR pel diagnòstic tècnic complex dins del sistema de "MOL TKE", realitzat per Michael Richter, juntament amb Klaus Dieter Althoff i altres en la Universitat de Kaiserslautern [2]. Això va donar lloc al sistema PATDEX [20].

Actualment, les activitats relacionades amb els sistemes de CBR s'estan estenent, tant als Estats Units com a Europa. Hi ha un creixent nombre de treballs sobre sistemes de CBR en gairebé qualsevol revista d'*Aprenentatge Artificial*. A Europa, Alemanya sembla haver pres una posició de lideratge en termes de nombre d'investigacions.

# Capítol 3

# Manteniment de la base de casos

En aquest capítol s'explica el que és el *Manteniment de la Base de Casos* i s'especifiquen els mètodes que s'implementen en aquest treball.

## 3.1 Definició de Manteniment de la Base de casos

El creixent nombre de sistemes de CBR a gran escala ha portat a una major consciència de la importància del **Manteniment de la Base de Casos** (*Case-Based Maintenance,* ***CBM***). Diverses investigacions han tractat sobre aquest problema, tenint en compte qüestions com el manteniment de la consistència i el control del creixement de la base de casos. Malgrat això, no hi ha un marc general per descriure els mètodes de *Manteniment de la Base de Casos.*

La definició de *Manteniment de la Base de Casos* (*Case-Based Maintenance,* ***CBM***) formulada per Leake i Wilson l'any 1998 [14], es presenta a continuació:

> *"Case-base maintenance implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives."*

El *Manteniment d'una Base de Casos* es pot fer a *nivell de representació* (p.ex. canviant els casos que defineixen el domini), a *nivell de coneixement* (p.ex. l'addició o l'eliminació d'un o més casos), a *nivell de la comptabilitat de la informació* (p.ex. usant informació sobre la freqüència amb que s'usa un cas), o a *nivell d'implementació* (p.ex. canviant els atributs que defineixen els casos). Els objectius de rendiment proporcionen criteris per avaluar el comportament i les tasques d'un sistema de CBR donada una base de casos inicial i una seqüència de casos resolts. Els objectius de rendiment poden ser *objectius quantitatius* (p.ex. el temps de resolució de problemes o la mida de la base de casos), o *objectius qualitatius* (p.ex. ampliar el rendiment del sistema).

En aquest treball es presenta un CBM a *nivell de coneixement* amb uns *objectius qualitatius*, és a dir, es proposen diferents models per augmentar el rendiment del sistema de CBR. Aquests models es veuen en la Secció 3.4.

## 3.2 Classificació dels mètodes de CBM

El CBM hauria de cobrir tots els aspectes que ajuden a mantenir un sistema de CBR d'alta qualitat. Les tècniques desenvolupades pels CBM van des de metodologies particulars que defineixen fases, passos i tasques per integrar el CBM en el procés de CBR, fins

a programes específics que permeten als administradors del sistema de CBR activitats de manteniment [10].

### 3.2.1 Tipus de Manteniment

Les diferents metodologies de manteniment sovint cobreixen els processos que descriuen el flux de treball i les seves tasques. Es distingeixen tres tipus de manteniment: [I]

- **Manteniment Correctiu** dels errors de processament (errors de funcionament i d'implementació)
- **Manteniment Adaptatiu** per evitar errors que es produeixen al canviar l'entorn del programa
- **Manteniment Perfectiu** per eliminar les ineficiències del processament, per millorar el rendiment o per tasques de manteniment

A la Secció 2.2 s'ha vist que es pot dividir el CBR en 4 subprocessos. Aquest procés va ser la base per la formulació d'una de les metodologies més importants i usades, la metodologia **SIAM** [19], que és una metodologia pel desenvolupament i el manteniment de sistemes de CBR.

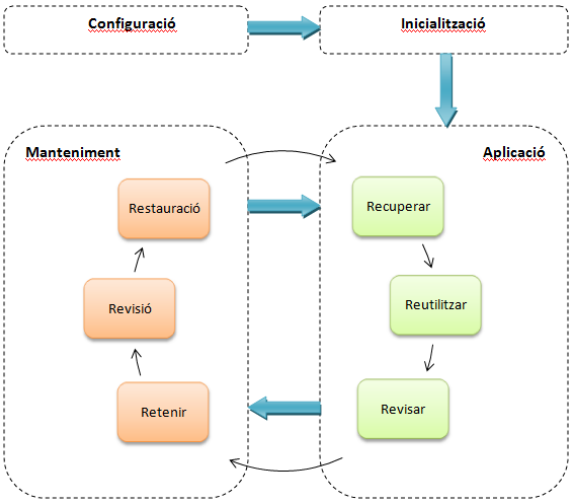


Figura 3.1: Model de procés de la metodologia SIAM

La Figura 3.1 mostra el cicle del procés SIAM, que esta format per 4 fases: *configuració*, *inicialització*, *aplicació* i *manteniment*. La fase de *configuració* aborda totes les qüestions que un sistema de CBR ha de tractar en les primeres etapes (per exemple, l'establiment dels objectius o el disseny). En la fase d'*inicialització* s'omple la base de casos. Així, el sistema de CBR està preparat pel seu ús, començant aleshores les fases d'*aplicació* i de *manteniment*. La fase d'*aplicació* conté les tres fases del model de CBR original, *Retrieve*, *Reuse* i *Revise*. La fase de *manteniment* comença així que el coneixement (cas) s'ha d'aprendre. Aquesta fase conté les fases *Retain*, *Review* i *Restore*. Si en la fase *Retain* el cas estudiat es après per la base de casos, aleshores es passa la fase de *Revisió*, on s'observa l'estat actual del sistema de CBR, es mesura la seva qualitat i si és necessari es passa a la fase de *Restauració*.

[I] Aquesta classificació de tipus de manteniment es basa en el principis de manteniment de software, definits per Swanson l'any 1976 [24]

### 3.2.2 Tasques de Manteniment

Les tasques de manteniment que porten a terme les fases de *Revisió* i *Restauració* són:

- **Mesurar** (Tasca de la fase de Revisió) Cal avaluar la qualitat del CBR d'alguna manera per saber el que cal reparar.
- **Observar** (Tasca de la fase de Revisió) En escenaris interactius les mesures de qualitat es poden visualitzar per facilitar al expert en el domini l'anàlisi d'aquestes mesures. En escenaris automatitzats les mesures actuals de qualitat es comparen amb d'altres mesures anteriors. L'objectiu principal de la tasca d'*Observar* és decidir si hi ha una disminució de la qualitat i aleshores executar la següent fase.
- **Suggerir** (Tasca de la fase de Restauració) Es dona una llista d'accions per poder reparar la disminució de la qualitat.
- **Seleccionar** (Tasca de la fase de Restauració) De la llista que s'obté en la tasca de *Suggerir* s'ha de triar la opció més adequada.
- **Modificar** (Tasca de la fase de Restauració) Finalment, les operacions de reparació seleccionades en la tasca de *Seleccionar* s'executen en la tasca de *Modificar*. El resultat de les operacions s'hauria de controlar novament tornant a la tasca de *Mesurar*.

Cada tasca de manteniment dins del *Manual de Manteniment* és una unitat de la descripció del nostre model, i consisteix en diferents elements: *entrades*, *activitats*, *mètodes*, *recursos* i *resultats* [19]. Les *entrades* descriuen la situació abans de començar la tasca, per exemple els resultats de la tasca executada prèviament. Les *activitats* són les tasques que es realitzen usant els *mètodes*. Es requereixen *recursos* per dur a terme la tasca. Finalment, es retornen els *resultats* d'executar la tasca.

## 3.3 Treball relacionat

Hi ha diverses publicacions sobre el manteniment dels sistemes de CBR, es mostren algunes de les que han tingut més impacte en aquesta àrea de recerca.

Racine i Yang fan un manteniment usant les mesures de *incoherència* i *redundància* [18]. La incoherència es pot detectar mitjançant el coneixement que es contradiu, i la redundància es veu en els casos que proporcionen la mateixa informació.

Leake i Wilson [15] usen dos relacions com a mesura. En primer lloc, defineixen la relació entre la similitud dels problemes i la similitud de les solucions. En segon lloc, defineixen la relació entre els problemes anteriors i els nous.

Smyth i Keane [11] i Zhu i Yang [25] tenen com a objectiu preservar el nivell de rendiment de la base de casos minimitzant a la vegada la seva mida. Defineixen la *cobertura* i *l'accessibilitat* com dues mesures per assolir aquest objectiu. La *cobertura d'un cas* és el conjunt de tots els casos que poden ser resolts per aquest cas. L'*accessibilitat d'un cas* és el conjunt de tots els casos que s'utilitzen per resoldre aquest cas.

## 3.4 Mètodes de CBM implementats

En aquesta Secció s'expliquen els mètodes implementats per fer el CBM:

- **Mètode CBR bàsic**: Es realitza el cicle dels sistemes de CBR sense fer tasques de CBM. S'utilitzaran els resultats obtinguts en aquesta implementació, com a referència per observar com millora el rendiment del nostre sistema de CBR gràcies a les tasques de CBM.
- **Mètode ACBR**: És un mètode que fa les tasques del CBM durant l'execució del cicle de CBR. Els primers passos són els mateixos que en cicle de CBR, però abans de fer

la fase de *Retain* fa una fase de *Review*, on decideix quins casos de la base de casos esborrar, i si el cas que s'està estudiant cal ser guardat o no. Aquestes decisions es prenen segons un valor de la mesura de ***"bondat"***. L'objectiu d'aquest mètode és mantenir les bases de casos compactes, de manera que l'eficiència no es vegi afectada negativament.

- **Mètode RDCL**: És un mètode que fa les tasques de CBM abans de l'execució del cicle de CBR. Estudia les mesures de ***"cobertura"***, ***"accessibilitat"***, ***"responsabilitat"*** i ***"dissimilitud"*** i depenent dels resultats esborra o no el conjunt de casos corresponent. Després executa el cicle de CBR amb la nova base de casos.

A continuació es veuran aquests mètodes amb més deteniment.

### 3.4.1 Definicions necessàries

Abans de veure els mètodes que s'han implementat s'explica el funcionament i es defineixen les variables que s'usaran.

Es vol veure com evoluciona el rendiment amb els diferents tests. S'agafa la base de casos amb la que es vol fer el CBR i es divideix usant una validació de *10-folds*, és a dir, es parteix la base de casos en 10 trossos, un tros s'agafa com a conjunt de *Test* i els altres 9 seran el conjunt de *Train*. Com s'ha dividit en 10 trossos, hi haurà 10 conjunts de *Test*, els diferents "folds", i els altres restants seran els conjunts de *Train*. Aquest procediment garanteix que els resultats obtinguts amb el sistema d'aprenentatge s'acosten al valor real en l'explotació del sistema. Es considera que la nova base de casos és el conjunt de *Train* i l'anomenem **CB**. Usarem els casos del conjunt de *Test*, al què anomenarem **TS**, com a casos per fer el test. A cadascun d'aquests casos l'anomenarem $c_{new}$, $c_{new} \in$ TS. Aleshores es fa servir un simulador per donar solució als casos del conjunt TS per tal d'avaluar el rendiment de cada algorisme. La solució del sistema de CBR serà comparada en la fase de Revisió amb la solució real, que està en la definició del cas de test.

A continuació, es defineixen les variables que s'usaran durant els algorismes. Sigui **c** qualsevol cas de la base de casos. Sigui **dist** la funció de distància usada en cada test. Sigui **k** el número de casos que s'agafen en la fase de *Retrieve* com a casos recuperats. Sigui **K** el conjunt d'aquests casos recuperats. Sigui $c_{sol}$ el cas solució que s'agafa entre els casos de K, $c_{sol} \in$ K, en la fase de *Reuse*. Sigui **clas** l'indicador de si el $c_{sol}$ ha classificat correctament o no al $c_{new}$.

Siguin $goodness_0$ i **goodness** les bondats inicial i actual d'un cas. Sigui **mc** la *classe majoritària*, és a dir, la classe que més es repeteix dins del conjunt de casos recuperats K.

### 3.4.2 Mètode CBR

Ja s'ha vist que el model CBR clàssic defineix el cicle dels sistemes CBR en quatre fases diferents: recuperar, reutilitzar, revisar i retenir. En la Figura 2.1 de la Secció 2.2 es mostra el cicle de CBR, on un nou cas es resol primer recuperant el cas més similar de la base de casos, a continuació, la reutilització d'aquest cas recuperat ens proporciona una solució, després es fa la revisió d'aquesta solució, i finalment, es reté el nou cas mitjançant la incorporació a la base de casos. Fixem-nos que la base de casos participa en tot el cicle de CBR (això és vital per a l'eficàcia de resolució de problemes del sistema). Per aquesta raó, el paper de la memòria i el manteniment de les bases de casos és de gran interès en els sistemes de CBR. De fet, el rendiment del sistema sempre depèn de la qualitat de la base de casos.

A continuació es mostra com funciona el cicle de CBR en el mètode implementat:

1. Es comença per la fase de **Retrieve**, on *es recupera* el cas o casos més similars al nou cas. Per fer-ho s'usarà la distància *Minkowsky*, amb r=1 (distància *Manhattan*, veure Funció 2.2), r=2 (distància *Euclidiana*, veure Funció 2.1) o r=3 (veure Funció 2.3) per atributs numèrics i la *Mètrica Heterogènia Minkowsky Superposada*, veure Funció 2.8, per atributs nominals. El mètode trobarà la solució al problema de tres maneres diferents: agafant el cas més proper, agafant els 3 casos més propers o agafant els 5 casos més propers.
2. A continuació, es fa la fase de **Reuse**, on s'obté el cas o casos recuperats en la fase anterior *reutilitzant* la seva informació, en concret, es vol l'atribut *"solució"* (és la *classe*). Si s'ha fet la fase de *Retrieve* amb més d'un cas, s'agafa aquell tal que la seva distància sigui menor.
3. En la fase de **Revise** es *revisa* si el cas més proper obtingut en la fase anterior té la mateixa classe que el *cas de test*, indicant així si el cas ha sigut classificat correcta o incorrectament.
4. Per acabar, la fase de **Retain** s'encarrega de *retenir* el nou cas si aquest no ha sigut classificat correctament, incrementant així la mida de la base de casos.

L'Algorisme 1 representa al model en pseudocodi per resoldre un nou cas $c_{new}$, i proporciona l'explicació de la Figura 2.1.

**Algorisme 1:** Algorisme del mètode CBR

**Data**: base de casos, *CB*; cas de test, $c_{new}$; funció de distància, *dist*; número de casos que es volen com a solució, $k$

```
1 // Fase de Retrieve
2 K = Retrieve(CB, c_new, dist, k)
3 // Fase de Reuse
4 c_sol = Reuse(K)
5 // Fase de Revise
6 clas = Revise(c_new, c_sol)
7 // Fase de Retain
8 Retain(CB, c_new, clas)
```

L'Algorisme 1 del cicle de CBR rep com a dades la base de casos, el cas de test, la funció de distancia que es vol fer servir i el numero de casos que es volen com a solució i realitza les 4 fases del cicle de CBR.

A continuació es veuen alguns dels algorismes corresponents a les fases del cicle de CBR.

L'Algorisme 2 de la fase de *Retrieve* rep com a dades la base de casos, el cas de test, la funció de distància que es vol fer servir i el número de casos que es volen com a solució. Per cada cas de la base de casos es calcula la distància entre els seus atributs i els atributs del cas que s'està estudiant i es guardem en un array (línies 4,5,6). Aquesta distància es calcula amb la funció de distància que l'hi hagi arribat. S'agafen les $k$ distàncies més petites i es busquen els casos corresponents. Es retornen els $k$ casos més semblants.

No es fa l'Algorisme de la fase de *Reuse*. En aquesta fase es busca el cas més proper al $c_{new}$ entre els casos obtinguts en la fase de *Retrieve*. Aquest cas és el $c_{sol}$.

Tampoc es fa l'Algorisme de la fase de *Revise*. En aquesta fase s'agafa el $c_{sol}$ i es compara la seva classe (o solució) amb la classe del $c_{new}$, si són iguals es diu que el cas s'ha classificat correctament, *clas=true*, en cas contrari es diu que s'ha classificat incorrectament, *clas=false*.

**Algorisme 2:** Algorisme de la fase de recuperar

**Data**: base de casos, *CB*; cas de test, $c_{new}$; funció de distància, *dist*; número de casos que es volen com a solució, $k$

**Result**: el conjunt de casos recuperats, $K$

```
1  auxDist → array de distàncies
2  auxK → array amb les k distàncies més petites
3  auxC → // Per cada c ∈ CB calculem la distància entre els atributs del c_new i els
   atributs del c.
4  foreach c ∈ CB do
5  |   auxDist = calcDist(c, c_new, dist);
6  |   dist.add(auxDist);
7  // Agafem les k les distàncies més petites
8  for i=0 to k do
9  |   min_{dist∈auxDist}(dist).add(auxK);
10 // Agafem els k casos corresponents a les distàncies de auxK
11 for j=0 to k do
12 |   K.add(c_{dist_j});
13 return K;
```

**Algorisme 3:** Algorisme de la fase de retenir

**Data**: base de casos, *CB*; cas de test, $c_{new}$; indicador de si el cas s'ha classificat correctament o no, *clas*

```
1  // Si el c_new ha sigut mal classificat,
2  if clas=false then
3  |   CB.add(c_new);
4  // D'aquesta manera s'actualitza la base de casos, afegint un cas nou
```

L'Algorisme 3 de la fase de *Retain* rep com a dades la base de casos, el cas de test i un indicador dient si el cas s'ha classificat correcta o incorrectament. Si el cas ha sigut mal classificat el sistema guarda el cas nou perquè considera que no té prou informació i per això el classifica malament. Aquest tipus de mètode de *Retain* s'anomena ***"different class"*** i és el mètode més bàsic. Es farà servir com a referència per comparar amb els altres mètodes.

Al fer el cicle de CBR el sistema guarda certa informació que s'usarà després per fer la comparació entre els mètodes. Guarda el número de casos que s'han classificat correctament i el número de casos que ho han fet incorrectament. Respectivament, aquestes dades permeten saber el percentatge d'encerts (que serà la mesura que es mirarà per decidir si el mètode millora o empitjora el seu rendiment) i el número de casos que s'han afegit a la base de casos.

### 3.4.3 Mètode ACBR

En aquesta Secció es proposa un model de Raonament Basat en Casos que s'estén del cicle de CBR bàsic (Figura 2.1). Es presenta el model i el seu objectiu de desenvolupar la base de casos mitjançant la incorporació de casos nous i l'oblit dels que no serveixen. Aquest model fa el manteniment de la base de casos amb un enfocament iteratiu, que promou els casos que són molt adequats per a la resolució dels problemes nous i redueix la influència dels casos que donen un error de classificació. La *retenció* i l'*oblit* es basen en la mesura de **Bondat** que es calcula iterativament sobre cada cas [16].

Es proposa un model de raonament general anomenat **Raonament Basat en Casos Adaptatiu** (*Adaptive Case-Based Reasoning,* ***ACBR***). La Figura 3.2 mostra aquest cicle adaptatiu. La fase de *Revisió* pot ser vista com un pas addicional en el cicle de CBR. Com a conseqüència d'això, s'amplia el cicle clàssic del CBR i es redefineix un cicle adaptatiu. Aquesta fase de *Revisió* es divideix en dues etapes principals: l'*actualització de la bondat* i l'*oblit*. Les estratègies de retenció són considerades com a part de la fase de *Retenir*, per tant fem el manteniment de la nostra base de casos en les dues últimes fases del cicle adaptatiu del CBR (és a dir, les fases de *Revisió* i *Retenir*).

Així, el funcionament del cicle d'ACBR és el mateix que el del cicle de CBR en les tres primeres fases, però després canvia (es pot seguir l'Algorisme 1, però la implementació a partir de la fase de *Revise* serà diferent). A més, abans de començar a executar el cicle, cal donar als casos de la base de casos un valor inicial de **bondat** que, per defecte fem que sigui *goodness* = 0.5. El fonament d'aquesta inicialització és que tots els casos es consideren igualment útils quan l'experiència de resolució de problemes és insuficient.

Ja hem vist que l'ACBR implementa la fase de *revisió* a través de dos passos principals: **Actualitzar la bondat** dels casos que han fet *Retrieve* i **Oblidar** els casos que no són útils. *Actualitzar la bondat* s'inspira en l'**aprenentatge per reforç** [6] i depèn d'una *taxa de modificació o d'aprenentatge* $\alpha$ i d'un valor de *recompensa*. Definim $R$ com el conjunt de valors de recompensa associats a tots els casos obtinguts en la fase de *Recuperació*. En concret $R$ ve donada per la Funció 3.1, on $r$ és la *funció de recompensa* r: C $\longrightarrow$ {0, 1} (C és un conjunt de casos) i RC = $c_1$, $c_2$, ..., $c_k$ és el conjunt dels $k$ casos recuperats en la fase de *Retrieve*:

$$R = \{r(c_k), c_k \in RC\} \tag{3.1}$$

La funció de recompensa $r$ assigna 1 als casos $c_k$ tals que la seva classe coincideixi amb la classe del cas que s'està estudiant, i assigna 0 en cas contrari. Per actualitzar el valor de

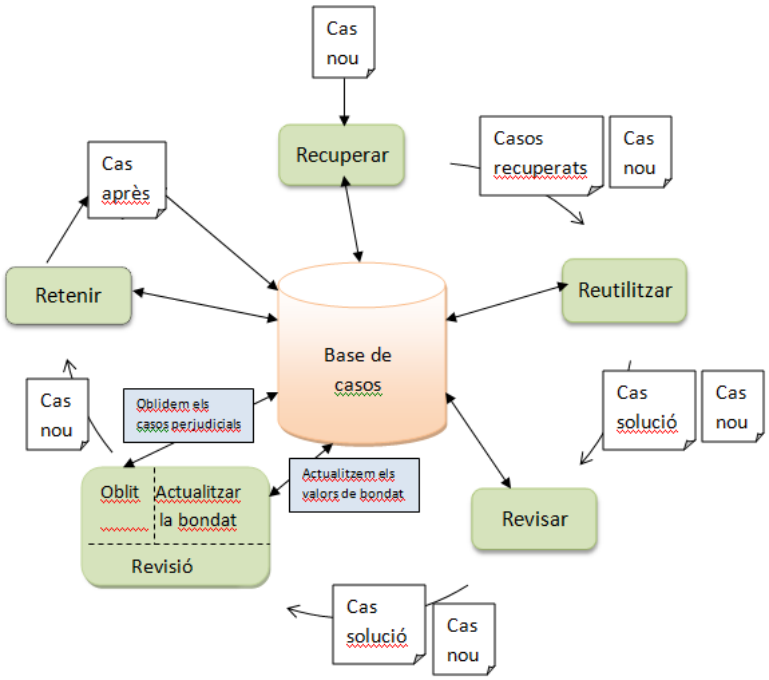


Figura 3.2: Fases del cicle d'ACBR

*bondat* dels casos recuperats s'aplica la Funció 3.2, la qual modifica el valor de bondat de qualsevol cas $c$. La taxa d'aprenentatge pot ser qualsevol valor, tot i que normalment $\alpha \in \{0.1\ ,0.2\}$ En la implementació del mètode s'ha agafat $\alpha = 0.2$.

$$goodness_j(c) = goodness_{j-1}(c) + \alpha(r(c) - goodness_{j-1}(c)) \tag{3.2}$$

Intuïtivament, es veu que aquesta fórmula segueix un enfocament d'*aprenentatge per reforç*, perquè la recompensa es genera quan un cas ajuda a la correcta classificació, i per tant, la seva mesura de bondat s'incrementa. En cas contrari, es genera una retroalimentació negativa.
En el segon pas de la fase de *revisió* és crida a una estratègia d'*Oblit* que revisa tots els casos recuperats per considerar si algun d'ells s'hauria d'esborrar de base de casos.

Finalment, s'implementa la fase de *retenir* invocant una estratègia de retenció. Aquesta estratègia decideix si el cas nou s'ha d'afegir a la base de casos, per fer-ho considera el cas que s'està estudiant, el cas que la fase de *Reuse* ha retornat com a *cas solució*, els casos recuperats i la base de casos. Si es produeix la retenció del cas, l'actual base de casos s'amplia amb un cas nou i amb el seu valor de bondat associat. D'aquesta manera, l'estratègia de retenir retorna una nova versió de la base de casos. En general, el model ACBR millora o degrada la bondat d'un cas en funció de la seva precisió de resolució. De fet, l'aspecte més notable d'aquest model de *raonament basat en casos d'adaptació* és que el sistema de CBR utilitza valors de bondat per millorar la base de casos durant el seu propi procés de resolució de problemes, ja que aquets valors són usats per les estratègies de *retenció* i *oblit*.

El model ACBR es diu que és adaptatiu perquè la base de casos es desenvolupa a través del temps. Això es fa tenint en compte l'experiència generacional (és a dir, la història dels episodis de resolució de problemes) de cada cas. Aquesta experiència generacional es representa a nivell local mitjançant la mesura de *bondat*. L'evolució de la base de casos es produeix mitjançant la *retenció* i l'*oblit* dels casos. A continuació es mostren diferents estratègies de retenció que, a part de decidir si és necessari mantenir un cas, també calcula la seva bondat

associada. A més, es presenta una estratègia d'oblit que utilitza valors de bondat per decidir si un cas ha de ser oblidat o no.

#### 3.4.3.1 Estratègia de retenció del *Grau de Desacord*

L'estratègia de retenció del **Grau de Desacord** (*Degree of Disagreement,* ***DD***) es basa en una mesura de *desacord* del conjunt de casos recuperats, K. L'algorisme proposat utilitza el conjunt K com un "comitè virtual". Així, cada membre de la comissió pot ser vist com un cas de K, i el vot és la classe associada a aquest cas. Es calcula el desacord entre les possibles solucions amb la Funció 3.3, on *# caseRetrieve* és el número de casos recuperats, *# sameClass* són els casos del pas de Retrieve que tenen la mateixa classe que la del cas solució i *# classes* és el número de classes de la base de casos.

$$d = \frac{\#caseRetrieve - \#sameClass}{(\#classes - 1)\#sameClass} \tag{3.3}$$

L'Algorisme 4 es correspon amb l'estratègia de retenció del grau de desacord. Com a entrada arriba la base de casos, *CB*, el cas de test, $c_{new}$, el conjunt de casos recuperats, *K*, el cas solució, $c_{sol}$, i el número de casos que es volen com a solució, *k*. Siguin *r* el número de casos recuperats que tenen la mateixa classe que la del cas solució i *c* el número de classes possibles dels casos de la base de casos. Un cop s'han obtingut aquestes dades, es calcula el grau de desacord, *d* aplicant la Funció 3.3 (línia 10). Si d és superior a un llindar determinat (en la nostra implementació hem agafat *threshold* = 0.3), l'estratègia de retenció considera que és útil retenir $c_{new}$, ja que pot ajudar en la solució de futurs desacords. Si es reté $c_{new}$, el sistema associarà un valor de bondat que correspon al valor màxim de bondat dels casos recuperats (línies 13,14,15).

**Algorisme 4:** Algorisme del Grau de Desacord, DD

**Data**: base de casos, *CB*; cas de test, $c_{new}$; casos recuperats, *K*; número de casos que es volen com a solució, *k*; cas solució, $c_{sol}$

```
1  r = 0
2  g = 0
3  c → número de classes possibles dels casos de CB
4  d → grau de desacord entre les possibles solucions
5  threshold = 0.3 → llindar predefinit
6  // Calculem el grau de desacord entre les possibles solucions
7  foreach case ∈ K do
8      if classe(case) = classe(c_sol) then
9          r ++
10 d = (k - r) / ( (c - 1) * r)
11 // Decidim si afegim o no c_new a CB
12 if d ≥ threshold then
13     g = max_{case∈K}(goodness(case));
14     goodness_0(c_new) = g;
15     goodness(c_new) = g;
16     CB.add(c_new)
```

El fonament d'aquesta estratègia de retenció és que el *desacord* pot generar solucions equivocades. És important tenir en compte que el DD és un *mètode no supervisat*: per afegir el cas a la base de casos no es considera si $c_{new}$ s'ha classificat correctament o no, sinó que, simplement, es basa en la majoria de les classes de K.

#### 3.4.3.2 Estratègia de retenció *Perjudicial*

L'estratègia de retenció **Perjudicial** (*Detrimental,* ***DE***) compara la classe del cas solució amb la *classe majoritària* per tal de decidir si el nou cas s'ha de mantenir o no. Com requereix la classe del cas, es tracta d'una estratègia de supervisió, i per tant, és més informativa que l'estratègia de retenció DD. El principi fonamental d'aquesta estratègia és que un nou cas es reté si ha estat mal classificat i si la classe majoritària de K produeix un error de classificació.

**Algorisme 5:** Algorisme Perjudicial, DE

**Data**: base de casos, *CB*; cas de test, $c_{new}$; casos recuperats, *K*; indicador de si el cas s'ha classificat bé o malament, *clas*

```
1 mc → classe majoritària
2 // Decidim si afegim o no c_new a CB
3 if (clas = false) && (classe(c_new) ≠ mc) then
4     goodness_0(c_new) = 0.5;
5     goodness(c_new) = 0.5;
6     CB.add(c_new)
```

L'Algorisme 5 es correspon amb l'estratègia de retenció perjudicial. Inicialment, s'analitza si el cas que s'està estudiant, $c_{new}$, s'ha classificat bé o malament, si s'ha classificat malament, *clas = false*. Si aquest és el cas, es fan servir els casos recuperats de K per l'obtenció de la classe majoritària mc. Si la classe majoritària també classifica $c_{new}$ incorrectament, el sistema considera que no conté suficients casos per produir una recuperació satisfactòria, es diu aleshores que s'ha fet una *recuperació perjudicial*. Per tal de compensar aquest error de classificació, $c_{new}$ s'afegeix a la base de casos, junt amb el seu valor de bondat. En l'estratègia DD, s'agafava el màxim valor de bondat, però no es pot aplicar aquesta política en l'estratègia DE, ja que no hi ha casos adequats en el conjunt de casos recuperat K. El que es fa és adoptar la política més simple, s'inicialitza la bondat del cas amb un valor intermedi de 0.5. Si *clas = true* (classe($c_{new}$) = classe($c_{sol}$)), aleshores es suposa que el sistema conté prou informació per produir una bona classificació i, per tant, $c_{new}$ serà descartat. A més, si $c_{new}$ no s'ha resolt correctament però la classe majoritària mj és capaç de resoldre-ho, $c_{new}$ es descartat perquè la base de casos no ha estat capaç de resoldre el cas però presenta prou diversitat de casos per resoldre casos similars en el futur.

#### 3.4.3.3 Estratègia de retenció de *Bondat Mínima*

L'estratègia de retenció de **Bondat Mínima** (*Minimal Goodness,* ***MG***) és una estratègia sense supervisió que decideix si un cas ha de ser après o no sense tenir en compte la solució (és a dir, la classe) del cas que s'està estudiant. En lloc de la classe del $c_{new}$, MG utilitza la bondat dels casos recuperats tals que la seva classe és la classe majoritària. S'anomena estratègia de retenció de bondat mínima perquè es requereix un valor mínim de bondat. Així, aquesta estratègia serà útil per provar la utilitat de la bondat.

L'Algorisme 6 es correspon amb l'estratègia de retenció de bondat mínima. S'utilitza el conjunt dels casos recuperats K per calcular la classe majoritària mc i s'agafa el subconjunt r $\subseteq$ K, format pels casos que tenen la classe igual a la classe majoritària (línia 11). Sigui auxr el valor màxim de bondat dels casos de r (línia 12). Si aquesta bondat auxr és menor que un llindar definit aleshores $c_{new}$ s'emmagatzema a la base de casos. El llindar que controla si s'afegeix un nou cas es calcula fent la mitjana entre els valors $cb_{max}$ i $cb_{min}$, que corresponen a les bondats màxima i mínima dels valors de bondat dels casos de la base de casos CB tals que la seva classe coincideix amb la classe majoritària, Funció 3.4. Per tant, per tal de poder calcular el llindar, primer s'ha d'obtenir cb, que és el conjunt de valors de bondat de tots els casos de la CB tals que tenen la mateixa classe que mc. Aquest llindar permet al model augmentar la diversitat dels grups que tenen casos amb baixos valors de bondat i restringeix el creixement dels grups que estan ben adaptats i tenen alts valors de bondat. Si auxr està per sota d'aquest llindar, aquesta estratègia de retenció decidirà emmagatzemar el $c_{new}$, i per tant es necessitarà el seu corresponent valor de bondat, que serà goodness($c_{new}$) = auxr.

$$threshold = \frac{g_{max} + g_{min}}{2} \tag{3.4}$$

**Algorisme 6:** Algorisme de Bondat Mínima, MG

**Data**: base de casos, *CB*; cas de test, $c_{new}$; casos recuperats, *K*

```
1  auxr = 0
2  cb_max = 0
3  cb_min = 0
4  thereshold = 0
5  r → array d'enters
6  cb → array d'enters
7  mc → classe majoritària
8  // Calculem el llindar
9  foreach case ∈ K do
10     if classe(case) = mc then
11         r.add(goodnes(case))
12 auxr = max_{goodness∈r}(goodness)
13 foreach case ∈ CB do
14     if classe(case) = mc then
15         cb.add(goodnes(case))
16 cb_min = min_{goodness∈cb}(goodness)
17 cb_max = max_{goodness∈cb}(goodness)
18 thereshold = (cb_max + cb_min) / 2
19 // Decidim si afegim c_new a la base de casos
20 if threshold ≥ auxr then
21     goodness_0(c_new) = auxr;
22     goodness(c_new) = auxr;
23     CB.add(c_new)
```

Igual que DD, MG és una estratègia sense supervisió. La principal diferència entre ells és la condició per a la retenció. DD es basa en un desacord entre totes les possibles solucions, mentre que MG es basa en un límit de bondat.

#### 3.4.3.4 Estratègia de retenció de l'*Aprenentatge basat en la Classificació Errònia*

L'estratègia de retenció de l'**Aprenentatge basat en la Classificació Errònia** (*Learning based on Missclassification,* ***LE***) és l'estratègia més simple que es proposa per a la introducció de l'experiència generacional en un sistema de CBR. La seva regla bàsica és: si un cas està mal classificat, s'afegeix a la base de casos. La comprovació d'errors de classificació requereix que l'estratègia sigui supervisada i així, aquesta regla s'aplica per evitar més errors de classificació. Igual que amb totes les estratègies de retenció, una vegada es decideix que un nou cas $c_{new}$ necessita ser emmagatzemat, aquesta estratègia ha d'assignar un valor de la bondat a aquest cas. Aquest valor bondat es prendrà des del cas més proper al cas que s'està estudiant. El cas més proper, cs, és el cas de K més proper a $c_{new}$ tal que la seva classe és igual a la classe del $c_{new}$. L'Algorisme 7 es correspon amb l'estratègia de retenció d'aprenentatge basat en la classificació errònia.

**Algorisme 7:** Algorisme d'Aprenentatge basat en la Classificació Errònia, LE

```
Data: base de casos, CB; cas de test, c_new; indicador de si el cas s'ha classificat
      correctament o no, clas; casos recuperats, K
1  cr → classes dels casos de K
2  r → array de casos
3  cs → cas més similar a c_new dins dels casos de r
4  // Decidim si afegim c_new a la base de casos
5  if clas = false then
6      foreach case ∈ K do
7          if classe(case) = classe(c_new) then
8              r.add(case)
9      cs = r[0] ;
10     // K té els casos endreçats de més proper a menys proper goodness_0(c_new) =
       goodness(cs);
11     goodness(c_new) = goodness(cs);
12     CB.add(c_new)
```

#### 3.4.3.5 Estratègia d'oblit d'*Oblit per Bondat*

En el procés del CBR és tan essencial oblidar com recordar, ja que *recordar* pot acabar sent negatiu (tot i que els casos emmagatzemats siguin correctes) i *oblidar* pot portar a una millora substancial en el rendiment. La capacitat del ACBR d'oblidar passa a la fase de *revisió*, que considera els casos que s'eliminaran mitjançant l'aplicació d'una estratègia d'oblit. L'estratègia que es proposa aplica una estratègia simple per decidir quins casos han de ser oblidats (és a dir, eliminats de la base de casos) depenent de la disminució del seu valor de bondat. S'anomena **Oblit per Bondat** (*Oblivion by goodness,* ***O***). Aquesta política elimina de la base de casos aquells casos recuperats tals que els seus valors de bondat actuals són inferiors als valors de bondat inicials. D'aquesta manera, s'eliminen els casos que produeixen errors de classificació al principi o en diverses ocasions durant els episodis de resolució de casos. Només aquells casos que en algunes ocasions permeten al sistema classificar correctament, es mantenen. Per tant, per exemple, s'eliminarà un cas si aquest classifica un nou cas incorrectament la primera vegada que és recuperat, perquè el seu valor bondat serà menor que l'inicial. Per contra, un cas tindrà oportunitats addicionals si ha classificat correctament altres casos abans. Això significa que el seu valor de bondat ja ha

augmentat, i per tant el seu valor de bondat actual encara pot disminuir i no caure per sota del seu valor inicial.

**Algorisme 8:** Algorisme d'Oblit per Bondat, O

```
Data: base de casos, CB; casos recuperats, K;
// Decidim si esborrem o no un cas de la base de casos
foreach case ∈ K do
    if goddness(case) ¡ goodness_0(case) then
        CB.delete(case)
```

L'Algorisme 8 mostra el procés d'aquesta estratègia d'oblit. L'oblit de casos es realitza sobre els casos recuperats de K. S'esborra un cas *case* ∈ K de la base de casos CB si el seu valor actual de bondat (goodness(case)) és menor que l'inicial ($goodness_0$(case)).

### 3.4.4 Mètode RDCL

En aquesta Secció és proposa el mètode **RDCL** [9], un mètode de *Manteniment de la Base de Casos*, que es realitza abans del CBR. Abans d'executar el cicle s'arregla la base de casos, esborrant aquells casos que o no són necessaris (donen informació repetida), o són perjudicials(donen informació errònia). Per decidir esborrar o no un cas, usem els **perfils RDCL**, els quals permeten fer un anàlisi de cada cas. Analitzar així aquests casos permet la seva categorització en funció de la seva utilitat i l'efecte que cada cas té en la competència global de la base de casos. Sigui **T** el *conjunt de Train.*

Smyth i Keane [11] van proposar dos conjunts per modelar la competència global d'un cas:

- El **conjunt d'Accessibilitat** d'un cas $c \in T$ és el conjunt de casos que poden classificar correctament al cas $c$ (Funció 3.5).
- El **conjunt de Cobertura** d'un cas $c \in T$ és el conjunt de casos que pot classificar correctament el cas $c$ (Funció 3.6).

Delany i Cunningham [7] van estendre aquest model amb una propietat addicional:

- El **conjunt de Responsabilitat** d'un cas $c \in T$ és el conjunt de casos que el cas $c$ classifica incorrectament (Funció 3.7).

Més tard, Delany [8] va afegir un últim conjunt:

- El **conjunt de Dissimilitud** d'un cas $c \in T$ és el conjunt de casos que classifiquen incorrectament al cas $c$ (Funció 3.8).

Per tant, es treballa sobre els perfils RDCL:

$$RSet(c) = \{c' \in T | classifies(c', c)\}(ReachabilitySet) \tag{3.5}$$

$$CSet(c) = \{c' \in T | classifies(c, c')\}(CoverageSet) \tag{3.6}$$

$$LSet(c) = \{c' \in T | misclassifies(c, c')\}(LiabilitySet) \tag{3.7}$$

$$DSet(c) = \{c' \in T | misclassifies(c', c)\}(DissimilaritySet) \tag{3.8}$$

on,

- **classifies(a,b)** vol dir que el cas $b$ contribueix a la correcta classificació del cas $a$, és a dir que $b$ és el cas més similar a $a$ i que $a$ té la mateixa classe que $b$.

- **misclassifies(a,b)** vol dir que el cas $b$ contribueix a la incorrecta classificació del cas $a$, és a dir que $b$ és el cas més similar a $a$ però que la classe de $a$ és diferent que la classe de $b$.

El conjunt de *dissimilitud* complementa al conjunt d'*accessibilitat*. De la mateixa manera, el conjunt de *responsabilitat* complementa al conjunt de *cobertura*.

El conjunt de cobertura d'un cas $c$ identifica la *utilitat* de $c$ en la base de casos. El conjunt de responsabilitat d'un cas identifica el *dany* que $c$ causa en la base de casos.

Si el cas *c'* ha classificat correctament al cas $c$, el cas *c'* s'inclourà en el conjunt d'accessibilitat del cas $c$. Si el cas *c'* ha classificat incorrectament al cas $c$, el cas *c'* s'inclourà en el conjunt de dissimilitud del cas $c$.

Per tant:

- Si el cas c ha sigut correctament classificat $\Rightarrow$ |RSet(c)| > 0
- Si el cas c ha sigut incorrectament classificat $\Rightarrow$ |DSet(c)| > 0
- Si el cas c és útil $\Rightarrow$ |CSet(c)| > 0
- Si el cas c és perjudicial $\Rightarrow$ |LSet(c)| > 0

El perfil RDCL d'un cas $c$ es deriva d'usar la lletra inicial (R o D i/o C i/o L) de qualsevol d'aquests conjunts que no estiguin buits. Per tant, un cas $c$ pot tenir una de les vuit possibles combinacions d'aquestes lletres indicant:

1. si el cas $c$ es classifica correctament o no (|RSet(c)| > 0 o |DSet(c)| > 0)
2. si el $c$ cas és útil (|CSet(c)| > 0)
3. si el $c$ cas és perjudicial (|LSet(c)| > 0)

Cadascuna de les vuit possibles combinacions per un cas $c$ es defineix i s'explica a continuació:

- **R**: El cas $c$ és classificat correctament, i no serveix per classificar cap altre cas.

$$|RSet(c)| > 0, |CSet(c)| = 0, |LSet(c)| = 0, |DSet(c)| = 0 \tag{3.9}$$

- **D**: El cas $c$ és classificat incorrectament, i no serveix per classificar cap altre cas.

$$|RSet(c)| = 0, |CSet(c)| = 0, |LSet(c)| = 0, |DSet(c)| > 0 \tag{3.10}$$

- **RC**: El cas $c$ és classificat correctament, i és útil (és a dir, classifica correctament altres casos).

$$|RSet(c)| > 0, |CSet(c)| > 0, |LSet(c)| = 0, |DSet(c)| = 0 \tag{3.11}$$

- **RL**: El cas $c$ és classificat correctament, i és perjudicial (és a dir, classifica incorrectament altres casos).

$$|RSet(c)| > 0, |CSet(c)| = 0, |LSet(c)| > 0, |DSet(c)| = 0 \tag{3.12}$$

- **DC**: El cas $c$ és classificat incorrectament, i és útil.

$$|RSet(c)| = 0, |CSet(c)| > 0, |LSet(c)| = 0, |DSet(c)| > 0 \tag{3.13}$$

- **DL**: El cas $c$ és classificat incorrectament, i és perjudicial.

$$|RSet(c)| = 0, |CSet(c)| = 0, |LSet(c)| > 0, |DSet(c)| > 0 \tag{3.14}$$

- **RCL**: El cas $c$ és classificat correctament, i és útil per alguns casos i perjudicial per altres.

$$|RSet(c)| > 0, |CSet(c)| > 0, |LSet(c)| > 0, |DSet(c)| = 0 \tag{3.15}$$

- **DCL**: El cas $c$ és classificat incorrectament, i és útil per alguns casos i perjudicial per altres.

$$|RSet(c)| = 0, |CSet(c)| > 0, |LSet(c)| > 0, |DSet(c)| > 0 \tag{3.16}$$

L'Algorisme 9 mostra el procés del mètode RDCL. Ja s'ha dit que el mètode RDCL es produeix abans d'executar el cicle de CBR, ja que la base de casos amb la que s'ha de fer el test es veu modificada pel mètode RDCL. L'algorisme rep el conjunt de Test, *TS*, el conjunt de Train, *CB*, i el conjunt RDCL que es vol esborrar *delRDCL*. S'inicialitza la base de casos amb el conjunt de Train. Aquests casos tindran com a atributs el valor d'accessibilitat, $R$, el valor de dissimilitud, $D$, el valor de cobertura, $C$, i el valor de responsabilitat, $L$. Es munten els conjunts RSet, DSet, RCSet, RLSet, DCSet, DLSet, RCLSet i DCLSet amb les condicions vistes anteriorment. S'esborren els casos de la base de casos del conjunt que vulguem esborrar, això ens dona una nova base de casos, *CB'* amb la que es fa el sistema de CBR.

**Algorisme 9:** Algorisme del mètode RDCL

```
Data: conjunt de Test, TS; conjunt de Train, T; conjunt RDCL a esborrar, delRDCL
Result: la nova base de casos, CB'
// S'inicialitza la base de casos CB amb els conjunts TS i T
CB → base de casos
// S'executa el test RDCL per actualitzar els valors R, D, C i L dels casos de CB
foreach case ∈ CB do
    executeRDCL(CB, case)
    // Es munten els conjunts RSet, DSet, RCSet, RLSet, DCSet, DLSet, RCLSet i
    DCLSet.
    if R(case)>0 && C(case)=0 && L(case)=0 && D(case)=0 then
        RSet.add(case);
    else if R(case)=0 && C(case)=0 && L(case)=0 && D(case)>0 then
        DSet.add(case)
    else if R(case)>0 && C(case)>0 && L(case)=0 && D(case)=0 then
        RCSet.add(case)
    else if R(case)>0 && C(case)=0 && L(case)>0 && D(case)=0 then
        RLSet.add(case)
    else if R(case)=0 && C(case)>0 && L(case)=0 && D(case)>0 then
        DCSet.add(case)
    else if R(case)=0 && C(case)=0 && L(case)>0 && D(case)>0 then
        DLSet.add(case)
    else if R(case)>0 && C(case)>0 && L(case)>0 && D(case)=0 then
        RCLSet.add(case)
    else
        DCLSet.add(case)
// S'esborra un d'aquests conjunts. Sigui delRDCL el conjunt que es vol esborrar
// (delRDCL=RSet o delRDCL=DSet o delRDCL=RCSet o delRDCL=RLSet o
delRDCL=DCSet o delRDCL=DLSet o delRDCL=RCLSet o delRDCL=DCLSet)
foreach case ∈ delRDCL do
    CB.delete(case)
return CB;
```

En l'Algorisme 10 es mostra el mètode *"executeRDCL"*. Per fer aquest cicle s'agafa com a conjunt de Test la pròpia base de casos CB, $c_{new} \in$ CB. Per tant, la base de casos CB i el

conjunt de Test són iguals. Aquest cicle és gairebé igual que el cicle de CBR amb excepció del pas de *Retain*, que es canvia pel pas de *Catalog*.

**Algorisme 10:** Algorisme del cicle RDCL, "executeRDCL"

**Data**: base de casos, *CB*; cas de test, $c_{new}$
1 // *Fase de Retrieve*
2 K = **Retrieve(CB,** $c_{new}$**, "euclidean", 1)**
3 // *Fase de Reuse*
4 $c_{sol}$ = **Reuse(K)**
5 // *Fase de Revise*
6 clas = **Revise(**$c_{new}$**,** $c_{sol}$**)**
7 // *Fase de Catalog*
8 **Catalog(CB,** $c_{new}$**,** $c_{sol}$**, clas)**

Les fase de *Retrieve*, *Reuse* i *Revise* s'executen igual que en el cicle de CBR, amb la distància Euclidiana i amb 1 cas de recuperació, ja que el cicle RDCL es fa amb aquestes configuracions.

A l'Algorisme 11 es mostra la fase de *Catalog*. On es cataloga els atributs R, D, C i L del $c_{sol}$ i del $c_{new}$ depenent de si el $c_{sol}$ ha classificat al cas $c_{new}$ correctament o incorrectament. Si el $c_{sol}$ ha classificat correctament al $c_{new}$, el $c_{sol}$ pertanyerà al RSet del $c_{new}$ i el $c_{new}$ pertanyerà al CSet del $c_{sol}$. En cas contrari, el $c_{sol}$ pertanyerà al DSet del $c_{new}$ i el $c_{new}$ pertany al LSet del $c_{sol}$. De fet no cal saber quins casos són, l'únic que cal saber és si els conjunts RSet($c_{new}$), CSet($c_{sol}$), DSet($c_{new}$) i LSet($c_{sol}$) són buits o no.

**Algorisme 11:** Fase de Catalog del cicle RDCL

**Data**: base de casos, *CB*; cas de test, $c_{new}$; cas solució, $c_{sol}$; indicador de si el cas s'ha classificat correctament o no, clas

```
1  i = 0
2  j = 0
3  // Es cataloguen els atributs R, D, C i L del c_sol i del c_new
4  // Si el cas s'ha classificat correctament
5  if clas = true then
6  |  // c_sol ∈ RSet(c_new) ;
7  |  i = R(c_new) ;
8  |  i++ ;
9  |  R(c_new) = i ;
10 |  // c_new ∈ CSet(c_sol) ;
11 |  j = C(c_sol) ;
12 |  j++ ;
13 |  C(c_sol) = j
14 else
15 |  // Si el cas s'ha classificat incorrectament ;
16 |  // c_sol ∈ DSet(c_new) ;
17 |  i = D(c_new) ;
18 |  i++ ;
19 |  D(c_new) = i ;
20 |  // c_new ∈ LSet(c_sol) ;
21 |  j = L(c_sol) ;
22 |  j++ ;
23 |  L(c_sol) = j
```

# Capítol 4

# Implementació i resultats

En aquest Capítol es detallen les tecnologies utilitzades i l'anàlisi del sistema. A més, en la Secció 4.3 es mostren els resultats, on es veu l'ús favorable del CBM.

## 4.1 Aspectes generals de la llibreria i de les eines

Per la realització d'aquest projecte s'ha utilitzat el llenguatge de programació Java pel disseny del sistema. Les entrades o *inputs* es controlen des de la classe *"Main"* del programa i els resultats o *outputs* queden reflexats en fitxers excels que es guarden en la memòria de l'ordinador. Les entrades són els diferents datasets que s'utilitzen per testejar els algorismes i les diferents configuracions amb les que s'executa el sistema de *Raonament Basat en Casos*. Les sortides són les dades que retorna el sistema un cop s'ha executat el cicle.

Cadascun dels datasets utilitzats contenen un cert número de casos. Aquests casos estan definits per una sèrie d'atributs (característiques), que poden ser tant numèrics com nominals. Amb els diferents datasets es realitzen els diferents tests, els quals proporcionen resultats que s'usen per analitzar com varia el rendiment del sistema.

Per complir amb l'objectiu de modularitat del sistema [I], la implementació del sistema es presenta en interfícies Java. Aquestes interfícies Java proporcionen els següents avantatges: organitzar la programació, obligar a que certes classes utilitzin els mateixos mètodes (noms i paràmetres) i establir relacions entre les classes que no estan relacionades.

Aquest projecte pretén analitzar diferents mètodes de CBM i veure com afecten en el rendiment del sistema de CBR. És per això que no s'ha implementat tot el cicle de CBR, la implementació bàsica venia donada per una altre TFG [26]. És evident però, que en el programa han d'estar tots els mètodes, no es pot fer un *Manteniment de la Base de Casos* sense un sistema de CBR. Els mètodes propis dels sistemes de CBR s'han hagut d'adaptar a aquest projecte i el que s'ha implementat són els algorismes de CBM. També s'ha hagut de fer un mètode *Main* que permeti executar un codi per consola, agafant les dades i recuperant les solucions, ja que en el treball que s'ha utilitzat per implementar el CBR es feia mitjançant un sistema Web.

[I] La modularitat d'un sistema és la capacitat que té un sistema de ser estudiat o entès com la unió de diferents parts que interactuen entre si i que treballen en un tasca necessària i diferenciada per aconseguir un objectiu comú

## 4.2 Detalls d'implementació

En aquesta Secció s'explica l'estructura de la implementació del sistema de CBR.

El programa consta de diferents paquets, on hi ha les interfícies i les classes que les implementen. Es pot veure en la Figura 4.1.

Cada interfície i classe tenen els seus propis atributs i mètodes que les defineixen. S'expliquen en detall cadascuna d'elles (només s'expliquen els mètodes implementats en aquest projecte, a l'Annex 1 es troben les explicacions dels mètodes que s'han adaptat).

La classe **Main** s'encarrega d'executar el codi del cicle de CBR. En la Figura 4.2 es veuen els mètodes que ha d'implementar:

- **main**: Mètode principal del programa. Carrega les dades necessàries per fer un test (dataset i configuració del test), executa aquest test i proporciona un Excel amb els resultats.
- **doTest**: Mètode que amb les dades enviades pel Main realitza un test.
- **changeName**: Mètode per canviar el format de certs noms que després s'han de buscar
- **writeInf**: Mètode que escriu en un Excel la informació dels datasets dels que es fa el test
- **writeResults**: Mètode que escriu en un Excel els resultats del test

La interfície **ICaseBase** defineix els mètodes que ha d'implementar una base de casos. La classe **CaseBase** representa a una base de casos i implementa a la interfície ICaseBase. En la Figura 4.3 es veuen els mètodes que han d'implementar:

- **getCaseC**: Mètode que retorna el cas que es demana

La interfície **ICase** defineix els mètodes que ha d'implementar un cas de la base de casos. La classe **Case** representa a un cas i implementa a la interfície ICase. En la Figura 4.4 es veuen els atributs i els mètodes que han d'implementar:

- *Atributs*:
  - **initialGoodnessValue**: valor inicial de bondat del cas
  - **currentGoodnessValue**: valor actual de bondat del cas
  - **coverageValue**: valor de cobertura del cas
  - **reachabilityValue**: valor d'accessibilitat del cas
  - **dissimilarityValue**: valor de dissimilitud del cas
  - **liabilityValue**: valor de responsabilitat del cas
- *Mètodes*:
  - **setInitialGoodnessValue**: Mètode que assigna el valor inicial de bondat a un cas
  - **getInitialGoodnessValue**: Mètode que retorna el valor inicial de bondat a un cas
  - **setCurrentGoodnessValue**: Mètode que assigna el valor actual de bondat a un cas
  - **getCurrentGoodnessValue**: Mètode que retorna el valor inicial de bondat a un cas
  - **setCoverageValue**: Mètode que assigna el valor de cobertura a un cas
  - **getCoverageValue**: Mètode que retorna el valor de cobertura a un cas
  - **setReachabilityValue**: Mètode que assigna el valor d'accessibilitat a un cas
  - **getReachabilityValue**: Mètode que retorna el valor d'accessibilitat a un cas
  - **setDissimilarityValue**: Mètode que assigna el valor de dissimilitud a un cas
  - **getDissimilarityValue**: Mètode que retorna el valor de dissimilitud a un cas
  - **setLiabilityValue**: Mètode que assigna el valor de responsabilitat a un cas

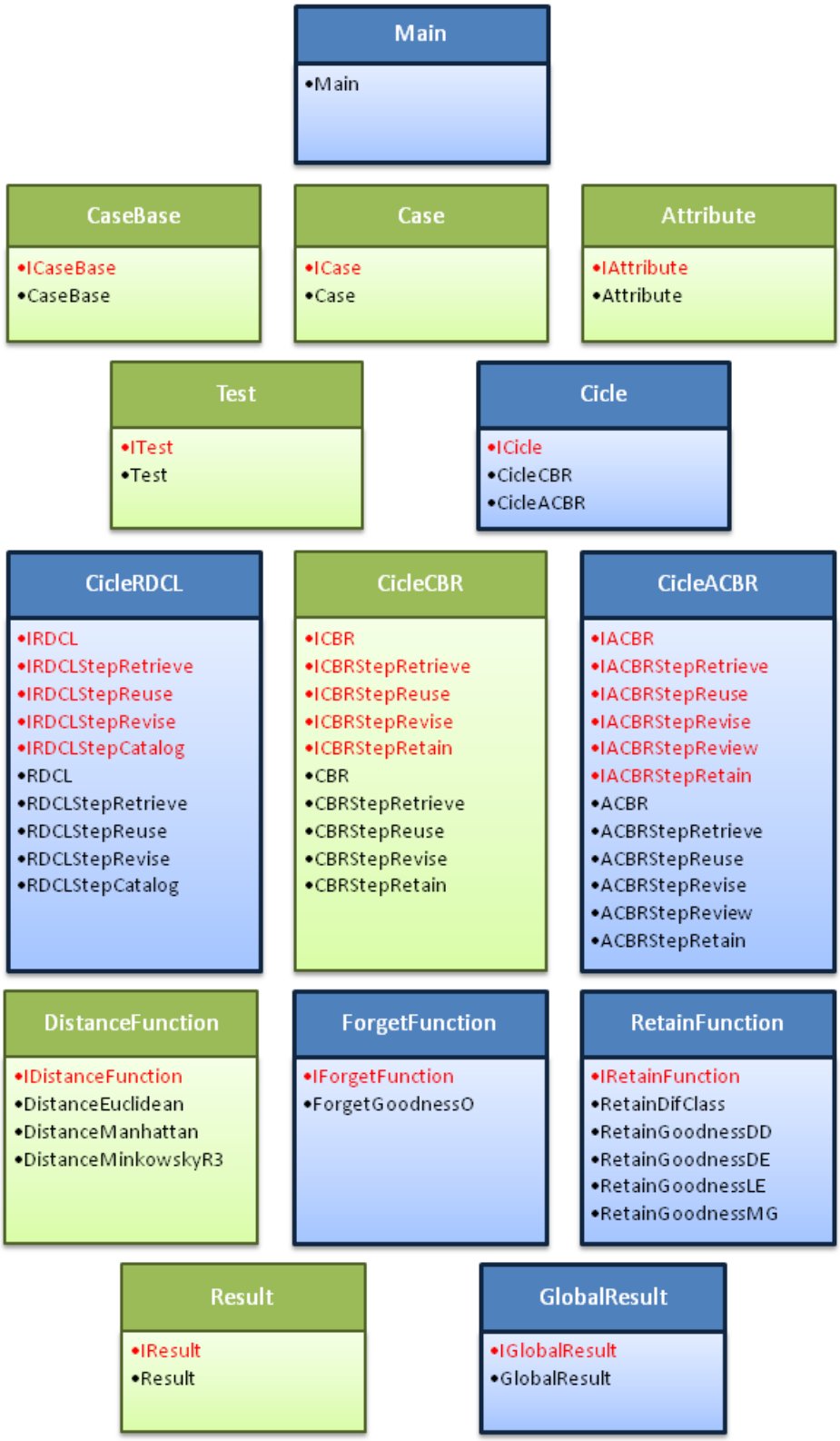


Figura 4.1: Paquets del programa que implementa el CBR. Els paquets que només s'han hagut d'adaptar tenen la vora més prima i el fondo verd i els paquets que s'han implementat tenen la vora més gruixuda i el fondo blau. Dins dels paquets, les interfícies estan escrites en vermell i les classes en negre.

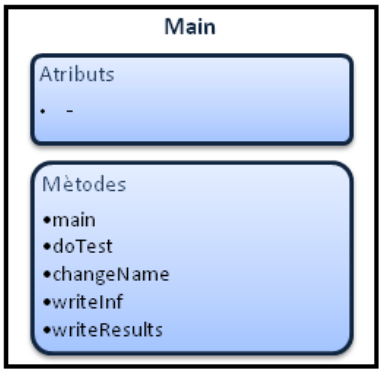


Figura 4.2: Mètodes i atributs de la classe Main. Com podem veure la classe Main no té atributs

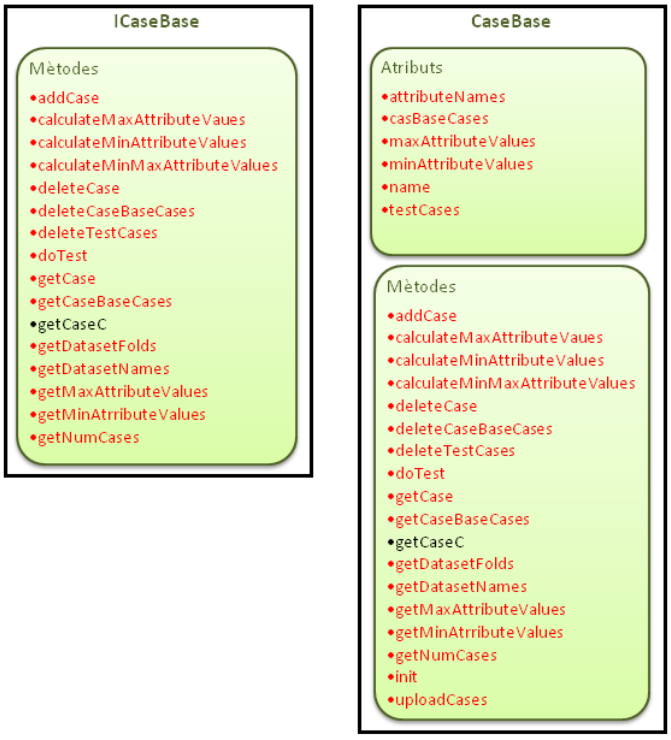


Figura 4.3: Mètodes i atributs de la interfície ICaseBase i la classe CaseBase. Els atributs i mètodes escrits en vermell són mètodes adaptats

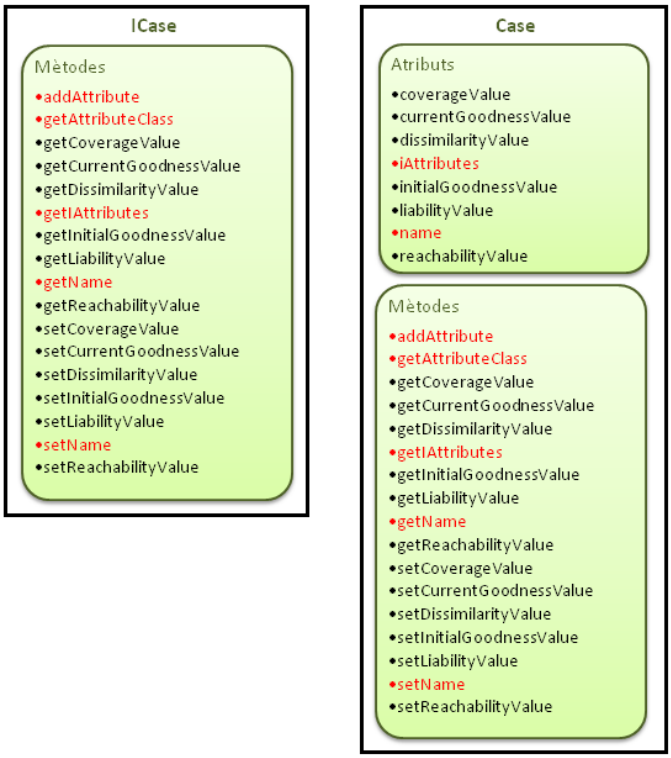


Figura 4.4: Mètodes i atributs de la interfície ICase i la classe Case. Els atributs i mètodes escrits en vermell són mètodes adaptats

– **getLiabilityValue**: Mètode que retorna el valor de responsabilitat a un cas

La interfície **IAttribute** defineix els mètodes que ha d'implementar un atribut d'un cas de la base de casos. Les classes **NumericalAttribute** i **NominalAttribute** representen a un atribut numèric i un atribut nominal respectivament, i implementen a la interfície IAttribute. En la Figura 4.5 es veuen els mètodes que han d'implementar.

La interfície **ITest** defineix els mètodes que ha d'implementar un test. La classe **Test** representa a un test i implementa a la interfície ITest. En la Figura 4.6 es veuen els mètodes que han d'implementar:

- **executeTestRDCL**: Mètode que executa un test RDCL

La interfície **ICicle** defineix els mètodes que ha d'implementar un cicle. Les classes **CicleCBR** i **CicleACBR** representen a un cicle de CBR i a un cicle d'ACBR respectivament, i implementen a la interfície ICicle. En la Figura 4.7 es veu el mètode que han d'implementar:

- **doCicle**: Mètode que executa un cicle de CBR o un cicle d'ACBR

Les interfícies **IRDCL** i **IRDCLStepRetrieve**, **IRDCLStepReuse**, **IRDCLStepRevise** i **IRDCLStepCatalog** defineixen els mètodes que ha d'implementar un cicle RDCL i els mètodes que han d'implementar el passos de Retrieve, Reuse, Revise i Catalog respectivament. Les classes **RDCL** i **RDCLStepRetrieve**, **RDCLStepReuse**, **RDCLStepRevise** i **RDCLStepCatalog** representen a un cicle RDCL i als passos de Retrive, Reuse, Revise i Catalog respectivament, i implementen a la interfície IRDCL i a les interfícies IRDCLStepRetrieve, IRDCLStepReuse, IRDCLStepRevise i IRDCLStepCatalog respectivament. En la Figura 4.8 es veuen els mètodes que han d'implementar:

- **makeRDCLCicle**: Mètode que executa un cicle RDCL
- **executeStepRetrieve**: Mètode que executa el pas de Retrieve d'un cicle RDCL
- **executeStepReuse**: Mètode que executa el pas de Reuse d'un cicle RDCL

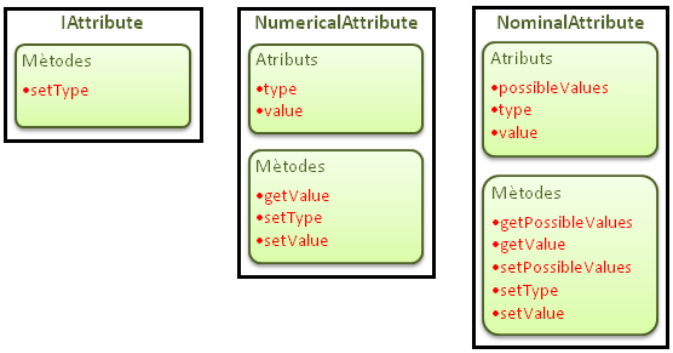


Figura 4.5: Mètodes i atributs de la interfície IAttribute i les classes NumericalAttribute i NominalAttribute. Tots els atributs i mètodes són mètodes adaptats

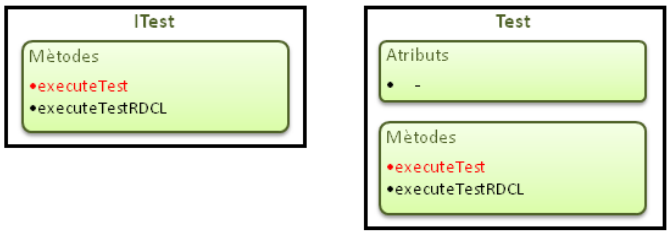


Figura 4.6: Mètodes i atributs de la interfície ITest i la classe Test. Els mètodes escrits en vermell són mètodes adaptats i la classe no té atributs

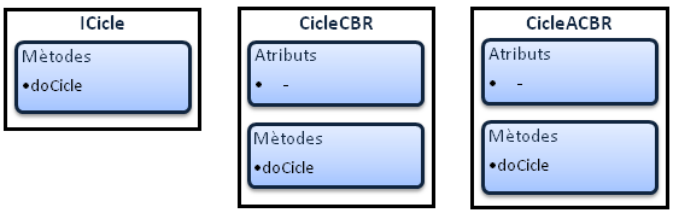


Figura 4.7: Mètodes i atributs de la interfície ICicle i les classes CicleCBR i CicleACBR. Tots els mètodes són implementats i les classes no tenen atributs

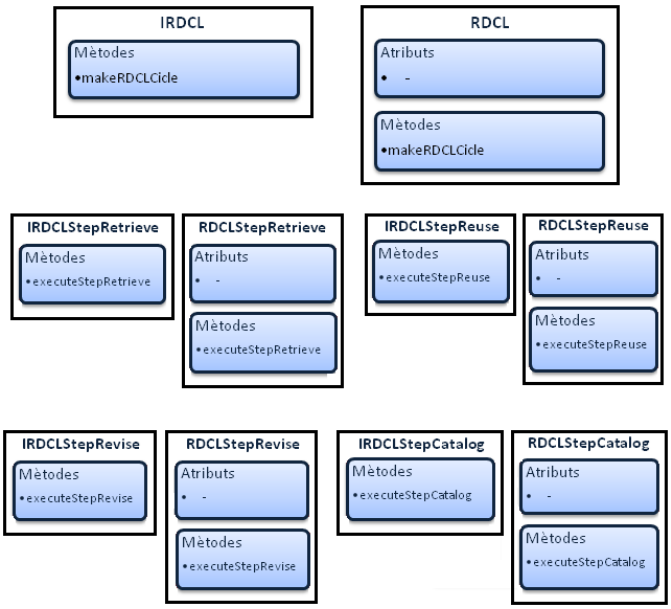


Figura 4.8: Mètodes i atributs de les interfícies IRDCL, IRDCLStepRetrieve, IRDCLStepReuse, IRDCLStepRevise i IRDCLStepCatalog i les classes RDCL, RDCLStepRetrieve, RDCLStepReuse, RDCLStepRevise i RDCLStepCatalog. Tots els mètodes són implementats i les classes no tenen atributs

- **executeStepRevise**: Mètode que executa el pas de Revise d'un cicle RDCL
- **executeStepCatalog**: Mètode que executa el pas de Catalog d'un cicle RDCL

Les interfícies **ICBR** i **ICBRStepRetrieve**, **ICBRStepReuse**, **ICBRStepRevise** i **ICBRStepRetain** defineixen els mètodes que ha d'implementar un cicle de CBR i els mètodes que han d'implementar el pasos de Retrieve, Reuse, Revise i Retain respectivament. Les classes **CBR** i **CBRStepRetrieve**, **CBRStepReuse**, **CBRStepRevise** i **CBRStepRetain** representen a un cicle de CBR i als pasos de Retrieve, Reuse, Revise i Retain respectivament, i implementen a la interfície ICBR i a les interfícies ICBRStepRetrieve, ICBRStepReuse, ICBRStepRevise i ICBRStepRetain respectivament. En la Figura 4.9 es veuen els mètodes que han d'implementar.

Les interfícies **IACBR** i **IACBRStepRetrieve**, **IACBRStepReuse**, **IACBRStepRevise**, **IACBRStepReview** i **IACBRStepRetain** defineixen els mètodes que ha d'implementar un cicle d'ACBR i els mètodes que han d'implementar el pasos de Retrieve, Reuse, Revise, Review i Retain respectivament. Les classes **ACBR** i **ACBRStepRetrieve**, **ACBRStepReuse**, **ACBRStepRevise**, **ACBRStepReview** i **ACBRStepRetain** representen a un cicle d'ACBR i als pasos de Retrieve, Reuse, Revise, Review i Retain respectivament, i implementen a la interfície IACBR i a les interfícies IACBRStepRetrieve, IACBRStepReuse, IACBRStepRevise, IACBRStepReview i IACBRStepRetain respectivament. En la Figura 4.10 es veuen els mètodes que han d'implementar:

- **makeACBRCicle**: Mètode que executa un cicle d'ACBR
- **executeStepRetrieve**: Mètode que executa el pas de Retrieve d'un cicle d'ACBR
- **executeStepReuse**: Mètode que executa el pas de Reuse d'un cicle d'ACBR
- **executeStepRevise**: Mètode que executa el pas de Revise d'un cicle d'ACBR
- **executeStepReview**: Mètode que executa el pas de Review d'un cicle d'ACBR
- **executeStepRetain**: Mètode que executa el pas de Retain d'un cicle d'ACBR

La interfície **IDistanceFunction** defineix els mètodes que ha d'implementar una funció

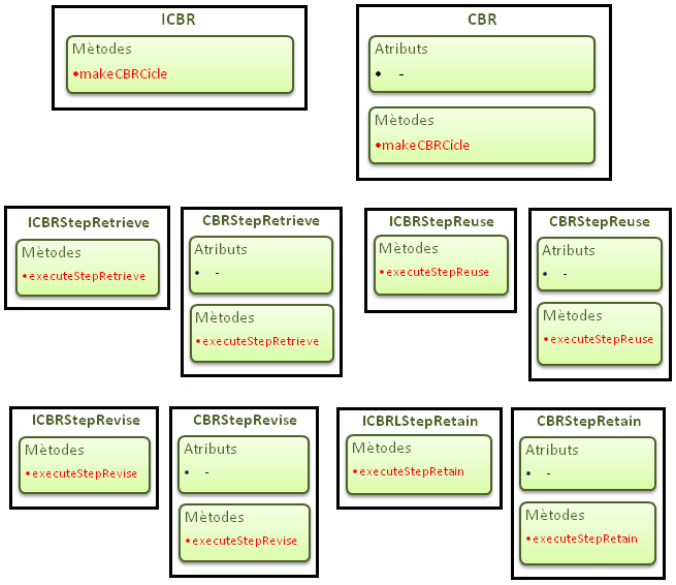


Figura 4.9: Mètodes i atributs de les interfícies ICBR, ICBRStepRetrieve, ICBRStepReuse, ICBRStepRevise i ICBRStepRetain i les classes CBR, CBRLStepRetrieve, CBRStepReuse, CBRStepRevise i CBRStepRetain. Tots els mètodes són adaptats i les classes no tenen atributs

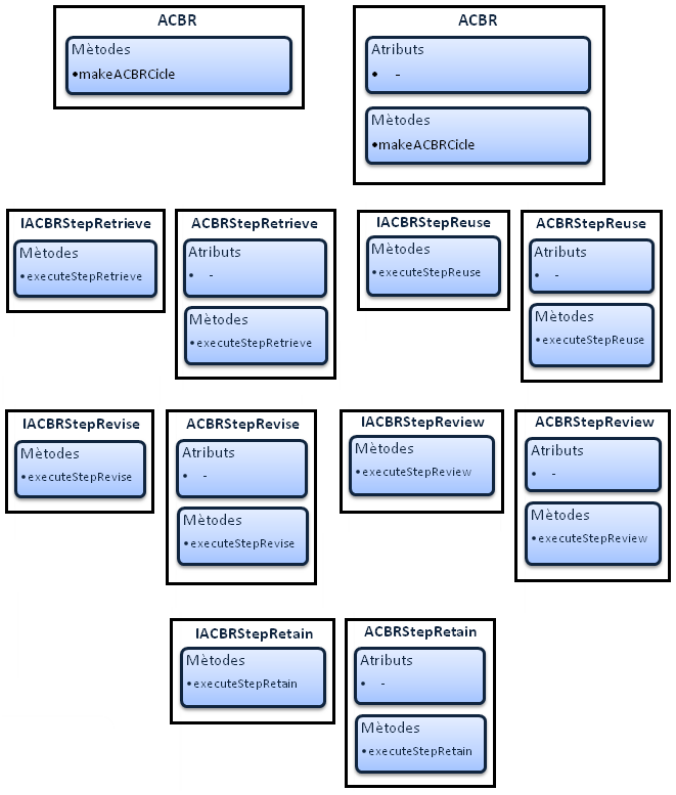


Figura 4.10: Mètodes i atributs de les interfícies IACBR, IACBRStepRetrieve, IACBRStepReuse, IACBRStepRevise, IACBRStepReview i IACBRStepRetain i les classes ACBR, ACBRLStepRetrieve, ACBRStepReuse, ACBRStepRevise, ACBRStepReview i ACBRStepRetain. Tots els mètodes són implementats i les classes no tenen atributs

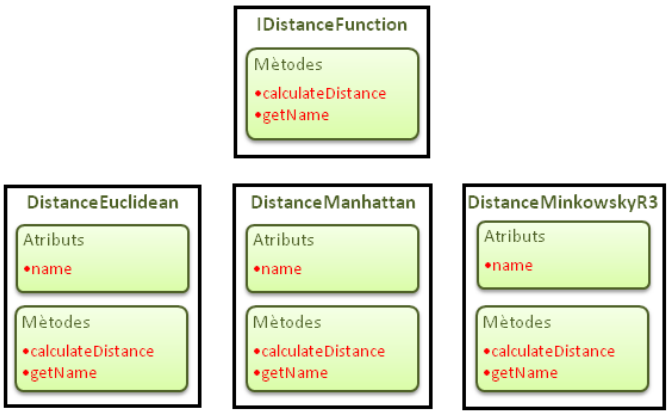


Figura 4.11: Mètodes i atributs de la interfície IDistanceFunction i les classes DistanceEuclidean, DistanceManhattan i DistanceMinkowskyR3. Tots els mètodes són adaptats

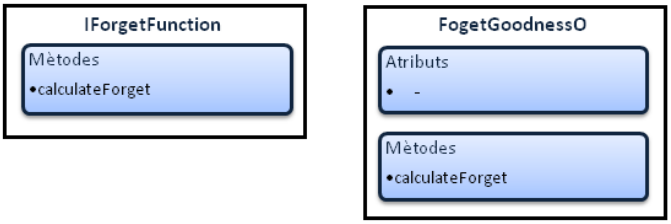


Figura 4.12: Mètodes i atributs de la interfície IForgetFunction i la classe ForgetGoodnessO. Tots els mètodes són implementats i no hi ha atributs

de distància. Les classes **DistanceEuclidean**, **DistanceManhattan** i **distanceMinkowskyR3** representen a les funcions de distància Euclidiana, Manhattan i MinkowskyR3, i implementen a la interfície IDistanceFunction. En la Figura 4.11 es veuen els mètodes que han d'implementar.

La interfície **IForgetFunction** defineix els mètodes que ha d'implementar una funció de forget. La classe **ForgetGoodnessO** representa a la funció de forget *Oblivion by goodness* i implementen a la interfície IForgetFunction. En la Figura 4.12 es veuen els mètodes que han d'implementar.

La interfície **IRetainFunction** defineix els mètodes que ha d'implementar una funció de Retain. Les classes **RetainDifClass**, **RetainGoodnessDD**, **RetainGoodnessDE**, **RetainGoodnessLE** i **RetainGoodnessMG** representen a les funcions de Retain *Different Class*, *Degree of Disagreement*, *Detrimental*, *Learning based on misclassification* i *Minimal Goodness* respectivament, i implementen a la interfície IRetainFunction. En la Figura 4.13 es veuen els mètodes que han d'implementar.

La interfície **IResult** defineix els mètodes que ha d'implementar el resultat d'un test. La classe **Result** representa a un resultat i implementa a la interfície IResult. En la Figura 4.14 es veuen els atributs i els mètodes que han d'implementar:

- *Atributs*:
  - **numCasesAdd**: número de casos que s'afegeixen a la base de casos al fer un test
  - **numCasesDelete**: número de casos que s'esborren de la base de casos al fer un test
  - **numCasesDeleteRDCL**: número de casos que s'esborren de la base de casos al fer un cicle RDCL

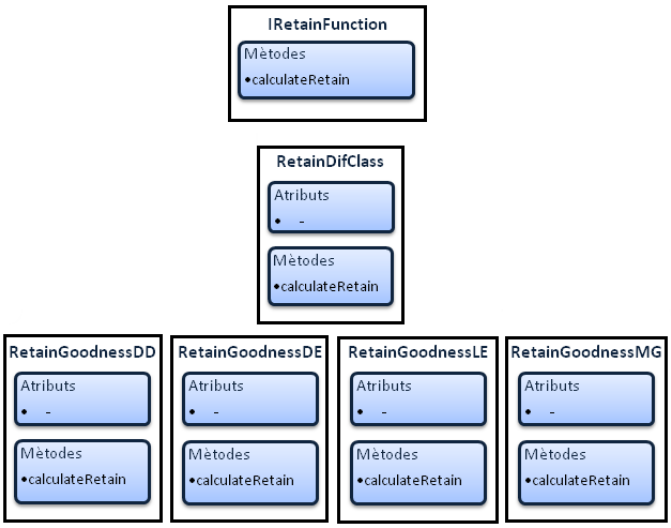


Figura 4.13: Mètodes i atributs de la interfície IRetainFunction i les classes RetainDifClass, RetainGoodnessDD, RetainGoodnessDE, RetainGoodnessLE i RetainGoodnessMG. Tots els mètodes són implementats i no hi ha atributs

- *Mètodes*:
  - **setNumAddCase**: Mètode que assigna el número de casos que s'han afegit a la base de casos al fer un test
  - **getNumAddCase**: Mètode que retorna el número de casos que s'han afegit a la base de casos al fer un test
  - **setNumDeleteCase**: Mètode que assigna el número de casos que s'han esborrat de la base de casos al fer un test
  - **getNumDeleteCase**: Mètode que retorna el número de casos que s'han esborrat de la base de casos al fer un test
  - **setNumDeleteCaseRDCL**: Mètode que assigna el número de casos que s'han esborrat de la base de casos al fer un cicle RDCL
  - **getNumDeleteCaseRDCL**: Mètode que retorna el número de casos que s'han esborrat de la base de casos al fer un cicle RDCL

La interfície **IGlobalResult** defineix els mètodes que ha d'implementar el resultat de diferents tests. La classe **GlobalResult** representa a diferents resultats i implementa a la interfície IGlobalResult. En la Figura 4.15 es veuen els atributs i els mètodes que han d'implementar:

- *Atributs*:
  - **numCasesAdd**: número de casos que cada test afegeix a la base de casos corresponent al fer el test
  - **numCasesDelete**: número de casos que cada test esborra de la base de casos corresponent al fer el test
  - **numCasesDeleteRDCL**: número de casos que cada test esborra de la base de casos corresponent al fer el cicle RDCL
  - **numCasesEnd**: número de casos finals de cada base de casos
  - **numCasesStart**: número de casos inicials de cada base de casos
  - **numCasesTest**: número de casos de test per cada base de casos
  - **percentSuccess**: percentatge d'encerts de cada test
- *Mètodes*:

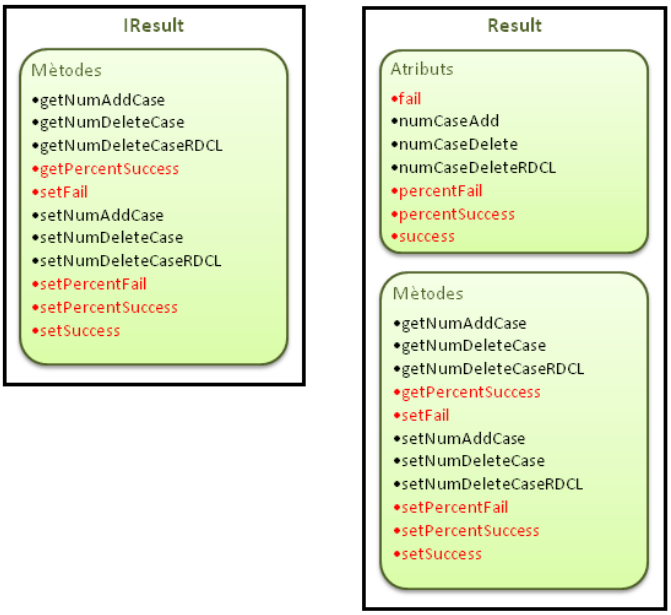


Figura 4.14: Mètodes i atributs de la interfície IResult i la classe Result. Els atributs i mètodes escrits en vermell són mètodes adaptats

- **getNumCasesAdd**: Mètode que retorna el número de casos que s'han afegit a la base de casos al fer cadascun dels tests
- **getNumCasesDelete**: Mètode que retorna el número de casos que s'han esborrat de la base de casos al fer cadascun dels tests
- **getNumCasesDeleteRDCL**: Mètode que retorna el número de casos que s'han esborrat de la base de casos al fer el cicle RDCL de cadascun dels tests
- **getNumCasesEnd**: Mètode que retorna el número de casos finals de cada base de casos
- **getNumCasesStart**: Mètode que retorna el número de casos inicials de cada base de casos
- **getNumCasesTest**: Mètode que retorna el numero de casos de test per cada base de casos
- **getPercentSuccess**: Mètode que retorna el percentatge d'encerts de cada test

## 4.3 Simulacions i resultats

En aquesta Secció s'analitzen els resultats obtinguts en els tests. Per fer el test s'han utilitzat 14 conjunts de dades de referència, que es descriuen breument a la Taula 4.1. Tots aquests conjunts de dades s'obtenen de la *UCI Machine Learning repository*. Els diversos datasets han estat escollits per proporcionar un nombre variable de casos, atributs (nominals i numèrics) i classes.

En aquest projecte s'intenta trobar mètodes que maximitzin el rendiment del cicle de CBR alhora que minimitzen la mida de la seva base de casos. Basant-nos en aquestes condicions, es pot veure quins són els "millors" tests i quins són els "pitjors". Recordem quins tests s'han fet:

- **CBR**: S'utilitza la funció de Retain *difClass*.
- **CBR amb RDCL**: Es fa el pas de RDCL i s'utilitza la funció de Retain *difClass*.

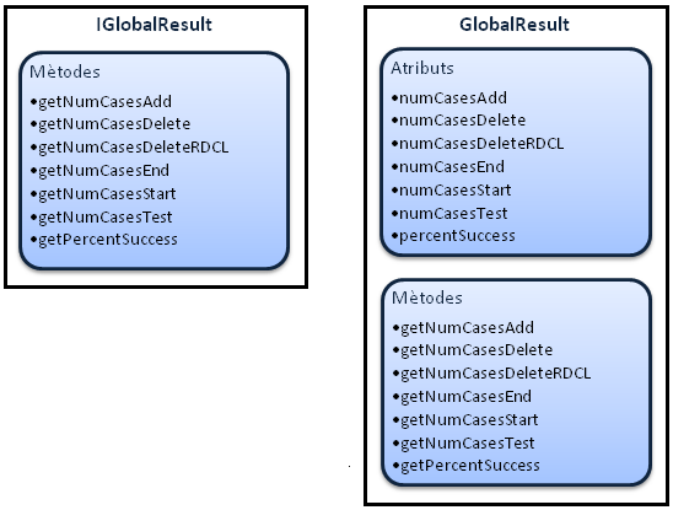


Figura 4.15: Mètodes i atributs de la interfície IGlobalResults i la classe GlobalResults. Els atributs i mètodes estan tots implementats

| Datasets | # Casos | # Atributs | # Classes |
|---|---|---|---|
| breast-w | 699 | 9 | 2 |
| bupa | 345 | 6 | 2 |
| cmc | 1473 | 9 | 3 |
| glass | 214 | 9 | 7 |
| heart-statlog | 270 | 13 | 2 |
| ionosphere | 351 | 34 | 2 |
| iris | 150 | 4 | 3 |
| lymph | 148 | 18 | 4 |
| pima-indians | 768 | 8 | 2 |
| sonar | 208 | 60 | 2 |
| vehicle | 846 | 18 | 4 |
| waveform | 5000 | 40 | 3 |
| wine | 178 | 13 | 3 |
| zoo | 101 | 17 | 7 |

Taula 4.1: Taula d'informació dels datasets que hem usat en aquest projecte. # Casos = número de casos del dataset; # Atributs = número d'atributs o de característiques dels casos del dataset; # Classes = número de classes o solucions dels casos del dataset

- **ACBR**: S'utilitzen les funcions de Retain *DD*, *DE*, *LE* i *MG* i es fa o no Forget amb la funció *O*.
- **ACBR amb RDCL**: Es fa el cicle del ACBR amb el pas de RDCL, s'utilitzen les funcions de Retain *DD*, *DE*, *LE* i *MG* i es fa o no Forget amb la funció *O*.

I cadascun d'aquests tests s'han fet pels 14 datasets anteriors, 3 funcions de distància (Minkowsky amb r=1, r=2 i r=3) i 3 funcions de Retrieve (k=1, k=3 i k=5). Al final del estudi, s'espera aconseguir les configuracions que facin un millor *Manteniment de la Base de Casos* per tenir un alt rendiment del CBR. A la Figura 4.16 es poden veure totes les possibles configuracions per cada model on, de manera general, es té que les configuracions estan formades per: 1 dels 14 datasets, 1 de les 3 funcions de distància, 1 de les 3 funcions de Retrieve, 1 dels 8 conjunts del pas de RDCL, 1 de les 5 funcions de Retain [II] i 1 de les 2 funcions de forget [III].

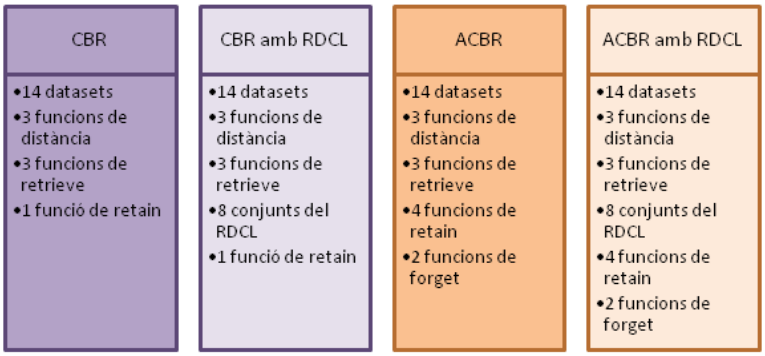


Figura 4.16: Possibles configuracions per cadascun dels models que s'executen. En color lila es veu el cicle de CBR, amb (lila fluix) i sense (lila fort) RDCl, i en color taronja es veu el cicle d'ACBR, també amb (taronja fluix) i sense (taronja fort) RDCL.

Per saber el número total de configuracions que es poden fer s'apliquen els *Principis Fonamentals de l'Anàlisi Combinatori*, que permeten estudiar les diferents seleccions que es poden fer amb els elements d'un conjunt donat.

**Teorema 1** ***Principi de Multiplicació****: Si un event o succés "A" pot ocórrer de "m" maneres diferents i un altre succés "B" pot ocórrer de "n" maneres diferents, aleshores el número de maneres diferents en que poden passar els dos successos és:* $m * n$.

**Teorema 2** ***Principi d'Addició****: Si un event o succés "A" pot ocórrer de "m" maneres diferents i un altre succés "B" pot ocórrer de "n" maneres diferents, i no és possible que els dos events es realitzin junts, aleshores l'event "A" o l'event "B" es realitzaran de* $m + n$ *maneres diferents.*

Així, per la Figura 4.16 i el Teorema 1 es pot saber el nombre total de configuracions per cadascun dels models:

- **CBR**: $14 * 3 * 3 * 1 = 126$ configuracions
- **CBR amb RDCL**: $14 * 3 * 3 * 8 * 1 = 1008$ configuracions
- **ACBR**: $14 * 3 * 3 * 4 * 2 = 1008$ configuracions
- **ACBR amb RDCL**: $14 * 3 * 3 * 8 * 4 * 2 = 8064$ configuracions

[II] Tot i que en total hi ha 5 funcions de Retain, el cicle de CBR només en pot triar 1 i el cicle d'ACBR tria 1 entre les 4 restants

[III] Es diu que hi ha 2 funcions de forget però en realitat només hi ha la opció de fer o no forget

| datasets | | 1 | | | 3 | | | 5 | |
|---|---|---|---|---|---|---|---|---|---|
| | #c | %c | %s | #c | %c | %s | #c | %c | %s |
| breast-w | 631.8 | 90.4 | 96.1 | 631.6 | 90.4 | 96.4 | 631.2 | 90.3 | 97 |
| bupa | 323.3 | 93.7 | 62.9 | 324.6 | 94.1 | 59 | 325.6 | 94.4 | 56.2 |
| cmc | 1407.4 | 95.5 | 44.5 | 1410.6 | 95.8 | 42.3 | 1408.8 | 95.6 | 43.6 |
| glass | 201.3 | 93.1 | 68.6 | 198.7 | 93.6 | 63.9 | 199.7 | 94.1 | 59.2 |
| heart-statlog | 249.8 | 92.5 | 74.8 | 249.9 | 92.6 | 74.4 | 249 | 92.2 | 77.8 |
| ionosphere | 320.5 | 91.3 | 86.9 | 321.3 | 91.5 | 84.7 | 321.7 | 91.7 | 83.5 |
| iris | 135.7 | 90.5 | 95.3 | 135.9 | 90.6 | 94 | 135.9 | 90.6 | 94 |
| lymph | 135.7 | 91.7 | 83.3 | 136.5 | 92.2 | 77.9 | 136.6 | 92.3 | 77 |
| pima-indians | 714.2 | 93 | 70.1 | 716.9 | 93.3 | 66.5 | 716.8 | 93.3 | 66.7 |
| sonar | 189.8 | 91.3 | 87.3 | 190.8 | 91.7 | 82.6 | 191.3 | 92 | 80.2 |
| vehicle | 787.3 | 93.1 | 69.3 | 790.2 | 93.4 | 65.9 | 791.1 | 93.5 | 64.8 |
| waveform | 4632.8 | 92.7 | 73.4 | 4638.4 | 92.8 | 72.3 | 4642.6 | 92.9 | 71.5 |
| wine | 161 | 90.4 | 95.6 | 160.8 | 90.3 | 96.7 | 161 | 90.4 | 95.6 |
| zoo | 91.4 | 90.5 | 95.6 | 92 | 91.1 | 89.4 | 92.3 | 91.4 | 86.6 |
| | | *92.1* | *78.9* | | **92.4** | **76.2** | | 92.5 | 75.3 |

Taula 4.2: Resultats del cicle de CBR amb la distància Euclidiana. #c = número de casos finals de la base de casos; %c = percentatge dels casos finals de la base de casos; %s = percentatge d'encerts del test. A la última fila surten els percentatges totals de número de casos i encerts

I junt amb el Teorema 2 es pot saber el nombre total de configuracions:

- $126 + 1008 + 1008 + 8064 = 10206$ configuracions

S'han realitzat totes les proves corresponents a les diferents configuracions definides. L'anàlisi dels resultats es detalla en escala donat el gran nombre de resultats a mostrar. És a dir, primer s'agafa com a referència el model CBR, que s'executa tornant uns resultats. Amb les configuracions que donin millor rendiment (és a dir, caldrà tenir en compte el percentatge d'encerts i el número de casos de la base de casos) s'analitzen els resultats en els següents algorismes d'aprenentatge i es mira l'evolució del rendiment.

Es comença doncs a fer l'anàlisi dels resultats:

A les Taules 4.2, 4.3 i 4.4 es poden veure els resultats del test del cicle de CBR amb la distància Euclidiana, Manhattan i Minkowsky amb r=3 respectivament.

Al analitzar els resultats de les Taules 4.2, 4.3 i 4.4 es veu que les configuracions que millor funcionen són aquelles en les que només s'agafa 1 cas de Retrieve (en les Taules 4.2, 4.3 i 4.4, els percentatges totals estan escrits en cursiva). Però els datasets que s'ha provat són petits i per observacions fetes fora del projecte se sap que agafant 3 o 5 casos de Retrieve s'aconsegueix millor rendiment. Per tant, no agafem configuracions que només agafin un cas de recuperació.

Així, les 3 configuracions que agafem per executar els altres models són: amb la distància Manhattan i 3 casos de recuperació, amb la distància Euclidiana i 3 casos de recuperació i amb la distància Manhattan i 5 casos de recuperació (en les Taules 4.2, 4.3 i 4.4, els percentatges totals estan escrits en negreta)

Es compara primer el model CBR amb el model CBR amb RDCL. A les taules de les Figures 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23 i 4.24 es poden veure els resultats del cicle de CBR amb el pas de RDCL esborrant els casos dels conjunt R, D, RC, RL, DC, DL, RDL i

| datasets | | 1 | | | 3 | | | 5 | |
|---|---|---|---|---|---|---|---|---|---|
| | #c | %c | %s | #c | %c | %s | #c | %c | %s |
| breast-w | 631.4 | 90.3 | 96.7 | 631.4 | 90.3 | 96.7 | 631.7 | 90.4 | 96.3 |
| bupa | 323.7 | 93.8 | 61.7 | 324.2 | 94 | 60.3 | 324.4 | 94 | 59.7 |
| cmc | 1407.5 | 95.6 | 44.5 | 1408.9 | 95.6 | 43.5 | 1409.4 | 95.7 | 43.2 |
| glass | 198.7 | 92.9 | 71.5 | 199.7 | 93.3 | 66.9 | 200.2 | 93.6 | 64.3 |
| heart-statlog | 249.1 | 92.3 | 77.4 | 250.2 | 92.7 | 73.3 | 249.1 | 92.3 | 77.4 |
| ionosphere | 319.2 | 90.9 | 90.6 | 319.9 | 91.1 | 88.6 | 320 | 91.2 | 88.3 |
| iris | 135.9 | 90.6 | 94 | 136 | 90.7 | 93.3 | 136.1 | 90.7 | 92.7 |
| lymph | 135.7 | 91.7 | 83.3 | 136.5 | 92.2 | 77.9 | 136.6 | 92.3 | 77 |
| pima-indians | 714.9 | 93.1 | 69.2 | 716.4 | 93.3 | 67.2 | 716.6 | 93.3 | 66.9 |
| sonar | 190.1 | 91.4 | 86 | 191.3 | 92 | 80.4 | 191.3 | 92 | 80 |
| vehicle | 786.9 | 93 | 69.8 | 788.3 | 93.2 | 68.2 | 790.9 | 93.5 | 65.1 |
| waveform | 4633 | 92.7 | 73.4 | 4638.5 | 92.8 | 72.3 | 4637.3 | 92.7 | 72.5 |
| wine | 160.8 | 90.3 | 96.7 | 160.6 | 90.2 | 97.8 | 161 | 90.4 | 95.6 |
| zoo | 91.4 | 90.5 | 95.6 | 92 | 91.1 | 89.4 | 92.3 | 91.4 | 86.6 |
| | | *92.1* | *79.3* | | **92.3** | **76.8** | | **92.4** | **76.1** |

Taula 4.3: Resultats del cicle de CBR amb la distància Manhattan. #c = número de casos finals de la base de casos; %c = percentatge dels casos finals de la base de casos; %s = percentatge d'encerts del test. A la última fila surten els percentatges totals de número de casos i encerts

| datasets | | 1 | | | 3 | | | 5 | |
|---|---|---|---|---|---|---|---|---|---|
| | #c | %c | %s | #c | %c | %s | #c | %c | %s |
| breast-w | 632.3 | 90.5 | 95.4 | 631.7 | 90.4 | 96.3 | 631.5 | 90.3 | 96.6 |
| bupa | 324.1 | 93.9 | 60.6 | 325.3 | 94.3 | 57 | 325.4 | 94.3 | 56.9 |
| cmc | 1407.6 | 95.6 | 44.4 | 1410.7 | 95.8 | 42.3 | 1409.4 | 95.7 | 43.2 |
| glass | 199.7 | 93.3 | 66.8 | 200.5 | 93.7 | 63 | 201.9 | 94.3 | 56.4 |
| heart-statlog | 250.3 | 92.7 | 73 | 249.8 | 92.5 | 74.8 | 249.4 | 92.4 | 76.3 |
| ionosphere | 321.1 | 91.5 | 85.2 | 321.4 | 91.6 | 84.4 | 322 | 91.7 | 82.6 |
| iris | 135.6 | 90.4 | 96 | 135.7 | 90.5 | 95.3 | 135.6 | 90.4 | 96 |
| lymph | 135.8 | 91.8 | 82.6 | 136.3 | 92.1 | 79.3 | 136.6 | 92.3 | 77 |
| pima-indians | 714.3 | 93 | 69.9 | 716.1 | 93.2 | 67.6 | 716.9 | 93.3 | 66.5 |
| sonar | 190 | 91.3 | 86.3 | 191.2 | 91.9 | 80.8 | 191.4 | 92 | 79.8 |
| vehicle | 788.5 | 93.2 | 67.9 | 191.2 | 93.5 | 64.7 | 192.8 | 93.7 | 62.8 |
| waveform | 4633.8 | 92.7 | 73.2 | 4642.5 | 92.9 | 71.5 | 4647.5 | 93 | 70.5 |
| wine | 160.9 | 90.4 | 96.2 | 160.8 | 90.3 | 96.8 | 160.9 | 90.4 | 96.2 |
| zoo | 91.4 | 90.5 | 95.6 | 92 | 91.1 | 89.4 | 92.3 | 91.4 | 86.7 |
| | | *92.2* | *78.1* | | 92.4 | 75.9 | | 92.5 | 74.8 |

Taula 4.4: Resultats del cicle de CBR amb la distància Minkowsky. #c = número de casos finals de la base de casos; %c = percentatge dels casos finals de la base de casos; %s = percentatge d'encerts del test. A la última fila surten els percentatges totals de número de casos i encerts

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 42,58 | 96,85 | 42,59 | 96,71 | 42,65 | 96,13 |
| bupa | 74,87 | 55,86 | 74,67 | 57,92 | 74,75 | 57,12 |
| cmc | 83,29 | 41,68 | 83,26 | 41,95 | 83,16 | 42,90 |
| glass | 75,05 | 62,92 | 74,39 | 69,68 | 75,00 | 63,34 |
| heart-statlog | 71,04 | 74,81 | 71,04 | 74,81 | 70,78 | 77,41 |
| ionosphere | 58,86 | 85,51 | 58,35 | 90,61 | 58,72 | 86,90 |
| iris | 61,53 | 95,33 | 61,53 | 95,33 | 61,67 | 94,00 |
| lymph | 65,54 | 76,47 | 65,47 | 77,13 | 65,74 | 74,48 |
| pima-indians | 70,07 | 67,18 | 70,07 | 67,20 | 70,16 | 66,30 |
| sonar | 64,47 | 80,87 | 64,86 | 76,92 | 65,24 | 72,87 |
| vehicle | 72,17 | 63,94 | 71,86 | 67,11 | 72,14 | 64,29 |
| waveform | 55,78 | 71,26 | 55,76 | 71,42 | 55,81 | 70,90 |
| wine | 57,92 | 95,54 | 57,64 | 98,30 | 57,92 | 95,55 |
| zoo | 54,55 | 87,60 | 54,55 | 87,60 | 55,15 | 81,99 |
| | **64,84** | **75,42** | **64,72** | **76,62** | **64,92** | **74,58** |

Figura 4.17: Taula dels resultats del model CBR amb RDCL esborrant el conjunt R

DCL respectivament.

Al analitzar els resultats de les taules de les Figures 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23 i 4.24 es veu que els millors resultats s'obtenen en la seva gran majoria amb la distància Manhattan i 3 casos de recuperació en el pas de *Retrieve*. Les 5 millors configuracions tenen les caselles de percentatges totals marcades en vermell.

Ara es compara el model CBR amb el model ACBR. A les taules de les Figures 4.25, 4.26, 4.27 i 4.28 es poden veure els resultats del cicle d'ACBR amb les funcions de Retain DD, DE, LE i MG respectivament.

Al analitzar els resultats de les taules de les Figures 4.25, 4.26, 4.27 i 4.28 es veu que els millors resultats s'obtenen amb la distància Manhattan, 3 casos de recuperació en el pas de *Retrieve* i fent oblit. Les 5 millors configuracions tenen les caselles de percentatges totals marcades en vermell.

Analitzant els resultats dels dos models es veu que la distància Manhattan amb 3 casos de recuperació en el pas de *Retrieve*, és la configuració amb la que s'obté millor rendiment.

En el model CBR amb RDCL veiem que els resultats més òptims s'aconsegueixen esborrant els casos dels conjunts DL, DCL, D i RL. Pensant en les definicions dels conjunts sembla evident que l'eliminació d'aquets conjunts és beneficiosa: els casos del conjunt D són els casos mal classificats i els casos del conjunt L són els casos perjudicials. Així, per fer el model ACBR amb el pas de RDCL s'usaran configuracions en les que s'esborrin els casos dels conjunts D, DC i DCL.

En el model ACBR qualsevol de les funcions de Retain amb forget donen un augment del rendiment, amb uns resultats gairebé idèntics. Per fer el model ACBR amb el pas de RDCL s'ha decidit usar les funcions de Retain de DE i MG. Així agafem una estratègia amb supervisió (DE) i una sense (MG).

Per tant, les configuracions que es faran servir per executar el model ACBR amb RDCL són: {manhattan}, {3 casos de Retrieve}, {esborrant els conjunts D, DL o DCL}, {amb les funcions de Retain DE o MG} i {amb forget}. Podem veure els resultats a la taula de la Figura 4.29.

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 89,11 | 96,54 | 89,11 | 96,55 | 89,14 | 96,27 |
| bupa | 81,57 | 56,04 | 81,33 | 58,50 | 81,39 | 57,91 |
| cmc | 80,01 | 42,69 | 79,86 | 44,13 | 79,93 | 43,51 |
| glass | 82,76 | 61,44 | 82,24 | 66,61 | 82,38 | 65,12 |
| heart-statl | 84,33 | 74,07 | 84,26 | 74,81 | 84,26 | 74,81 |
| ionospher | 83,79 | 84,67 | 83,48 | 87,78 | 83,53 | 87,19 |
| iris | 87,73 | 94,67 | 87,73 | 94,67 | 87,67 | 95,33 |
| lymph | 83,51 | 77,39 | 83,45 | 78,01 | 83,18 | 80,63 |
| pima-india | 83,13 | 69,14 | 83,13 | 69,15 | 83,03 | 70,07 |
| sonar | 83,85 | 80,33 | 83,94 | 79,41 | 84,28 | 75,75 |
| vehicle | 83,32 | 66,63 | 83,18 | 68,06 | 83,53 | 64,50 |
| waveform | 78,86 | 72,92 | 78,92 | 72,26 | 78,93 | 72,24 |
| wine | 86,52 | 96,22 | 86,40 | 97,24 | 86,57 | 95,64 |
| zoo | 87,62 | 90,68 | 87,62 | 90,68 | 87,92 | 87,91 |
| | **84,01** | **75,96** | **83,90** | **76,99** | **83,98** | **76,21** |

Figura 4.18: Taula dels resultats del model CBR amb RDCL esborrant el conjunt D

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 52,96 | 95,97 | 52,92 | 96,42 | 53,03 | 95,27 |
| bupa | 68,93 | 56,43 | 68,49 | 60,80 | 69,28 | 53,03 |
| cmc | 75,96 | 40,86 | 75,86 | 41,89 | 76,08 | 39,71 |
| glass | 59,81 | 57,24 | 59,49 | 60,47 | 60,00 | 55,07 |
| heart-statlog | 52,67 | 70,00 | 52,81 | 68,52 | 52,37 | 72,96 |
| ionosphere | 54,42 | 80,68 | 53,79 | 86,94 | 53,96 | 85,23 |
| iris | 36,47 | 90,00 | 36,27 | 92,00 | 36,07 | 94,00 |
| lymph | 55,27 | 72,72 | 55,20 | 73,46 | 55,68 | 69,05 |
| pima-indians | 60,95 | 61,19 | 60,92 | 61,46 | 60,90 | 61,70 |
| sonar | 50,63 | 73,88 | 50,67 | 73,41 | 51,39 | 66,24 |
| vehicle | 59,78 | 62,38 | 59,59 | 64,26 | 59,83 | 61,78 |
| waveform | 72,14 | 68,46 | 72,01 | 69,74 | 72,10 | 68,86 |
| wine | 41,97 | 94,56 | 42,02 | 93,85 | 42,47 | 89,51 |
| zoo | 45,64 | 85,90 | 45,64 | 85,90 | 46,04 | 82,23 |
| | **56,26** | **72,16** | **56,12** | **73,51** | **56,37** | **71,05** |

Figura 4.19: Taula dels resultats del model CBR amb RDCL esborrant el conjunt RC

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 90,07 | 96,41 | 90,03 | 96,85 | 90,09 | 96,27 |
| bupa | 90,29 | 59,91 | 90,23 | 60,61 | 90,32 | 59,71 |
| cmc | 93,64 | 42,02 | 93,52 | 43,17 | 93,53 | 43,11 |
| glass | 89,72 | 63,93 | 89,39 | 67,34 | 89,63 | 64,86 |
| heart-statlog | 90,89 | 74,81 | 91,07 | 72,96 | 90,70 | 76,67 |
| ionosphere | 87,24 | 85,50 | 86,92 | 88,64 | 86,95 | 88,34 |
| iris | 89,53 | 94,00 | 89,60 | 93,33 | 89,67 | 92,67 |
| lymph | 89,39 | 78,55 | 89,32 | 79,17 | 89,59 | 76,40 |
| pima-indians | 90,47 | 65,48 | 90,39 | 66,27 | 90,42 | 66,01 |
| sonar | 89,23 | 82,12 | 89,38 | 80,85 | 89,47 | 79,50 |
| vehicle | 91,56 | 65,57 | 91,26 | 68,51 | 91,56 | 65,57 |
| waveform | 90,42 | 72,88 | 90,49 | 72,20 | 90,45 | 72,64 |
| wine | 89,27 | 96,75 | 89,16 | 97,77 | 89,44 | 94,99 |
| zoo | 89,60 | 89,43 | 89,60 | 89,43 | 89,90 | 86,66 |
| | **90,09** | **76,24** | **90,03** | **76,94** | **90,12** | **75,96** |

Figura 4.20: Taula dels resultats del model CBR amb RDCL esborrant el conjunt RL

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 89,90 | 96,40 | 89,89 | 96,56 | 89,91 | 96,27 |
| bupa | 89,59 | 57,86 | 89,25 | 61,38 | 89,33 | 60,48 |
| cmc | 93,38 | 42,15 | 93,28 | 43,17 | 93,32 | 42,77 |
| glass | 91,21 | 62,60 | 90,93 | 65,49 | 91,26 | 62,07 |
| heart-statlog | 91,59 | 74,07 | 91,67 | 73,33 | 91,26 | 77,41 |
| ionosphere | 88,95 | 84,39 | 88,49 | 88,90 | 88,60 | 87,77 |
| iris | 90,53 | 94,00 | 90,60 | 93,33 | 90,67 | 92,67 |
| lymph | 90,07 | 77,21 | 90,07 | 77,24 | 90,14 | 76,41 |
| pima-indians | 91,22 | 66,14 | 91,15 | 66,92 | 91,17 | 66,67 |
| sonar | 89,47 | 81,71 | 89,62 | 80,41 | 89,66 | 79,57 |
| vehicle | 91,44 | 65,55 | 91,13 | 68,62 | 91,50 | 64,97 |
| waveform | 90,33 | 72,30 | 90,30 | 72,62 | 90,28 | 72,78 |
| wine | 89,78 | 96,75 | 89,66 | 97,77 | 89,89 | 95,57 |
| zoo | 90,89 | 88,60 | 90,89 | 88,60 | 91,29 | 84,99 |
| | **90,60** | **75,70** | **90,49** | **76,74** | **90,59** | **75,74** |

Figura 4.21: Taula dels resultats del model CBR amb RDCL esborrant el conjunt DC

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 88,56 | 96,99 | 88,57 | 96,85 | 88,58 | 96,70 |
| bupa | 82,29 | 62,22 | 82,38 | 61,42 | 82,52 | 60,03 |
| cmc | 69,36 | 44,86 | 69,27 | 45,81 | 69,36 | 44,86 |
| glass | 82,99 | 65,24 | 82,71 | 68,24 | 82,94 | 65,79 |
| heart-statlog | 82,07 | 76,67 | 81,96 | 77,78 | 81,89 | 78,52 |
| ionosphere | 90,48 | 84,65 | 90,11 | 88,35 | 90,11 | 88,33 |
| iris | 89,40 | 94,67 | 89,47 | 94,00 | 89,53 | 93,33 |
| lymph | 88,92 | 77,74 | 88,92 | 77,78 | 88,99 | 76,98 |
| pima-indians | 82,10 | 70,05 | 82,01 | 70,96 | 82,15 | 69,54 |
| sonar | 90,72 | 82,21 | 90,87 | 80,85 | 91,06 | 78,64 |
| vehicle | 81,60 | 66,37 | 81,45 | 67,83 | 81,55 | 66,85 |
| waveform | 88,72 | 72,86 | 88,71 | 72,98 | 88,71 | 72,94 |
| wine | 90,34 | 96,75 | 90,22 | 97,77 | 90,45 | 95,57 |
| zoo | 90,99 | 89,43 | 90,99 | 89,43 | 91,29 | 86,66 |
| | **85,61** | **77,19** | **85,55** | **77,86** | **85,65** | **76,77** |

Figura 4.22: Taula dels resultats del model CBR amb RDCL esborrant el conjunt DL

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 89,79 | 96,41 | 89,77 | 96,56 | 89,81 | 96,13 |
| bupa | 86,67 | 57,53 | 86,41 | 60,32 | 86,46 | 59,72 |
| cmc | 90,41 | 42,76 | 90,35 | 43,31 | 90,44 | 42,43 |
| glass | 88,13 | 63,01 | 87,85 | 65,77 | 87,85 | 65,74 |
| heart-statlog | 87,41 | 76,67 | 87,56 | 75,19 | 87,37 | 77,04 |
| ionosphere | 87,72 | 84,94 | 87,41 | 88,06 | 87,38 | 88,30 |
| iris | 89,40 | 93,33 | 89,47 | 92,67 | 89,40 | 93,33 |
| lymph | 86,01 | 76,72 | 86,01 | 76,76 | 85,95 | 77,09 |
| pima-indians | 89,04 | 65,63 | 88,93 | 66,67 | 88,98 | 66,14 |
| sonar | 85,91 | 80,13 | 85,67 | 82,67 | 85,82 | 81,06 |
| vehicle | 88,27 | 64,99 | 87,98 | 67,93 | 88,10 | 66,71 |
| waveform | 86,98 | 73,24 | 86,97 | 73,38 | 87,02 | 72,88 |
| wine | 87,02 | 96,16 | 86,80 | 98,36 | 87,08 | 95,64 |
| zoo | 89,01 | 88,60 | 89,01 | 88,60 | 89,31 | 85,82 |
| | **87,98** | **75,72** | **87,87** | **76,87** | **87,93** | **76,29** |

Figura 4.23: Taula dels resultats del model CBR amb RDCL esborrant el conjunt RCL

| | euclidean 3 | | manhattan 3 | | manhattan 5 | |
|---|---|---|---|---|---|---|
| | %c | %s | %c | %s | %c | %s |
| breast-w | 89,87 | 95,99 | 89,84 | 96,27 | 89,87 | 95,98 |
| bupa | 89,33 | 57,81 | 89,04 | 60,86 | 88,93 | 62,08 |
| cmc | 89,92 | 43,31 | 89,86 | 43,85 | 89,88 | 43,72 |
| glass | 90,19 | 64,40 | 89,86 | 67,88 | 90,23 | 63,81 |
| heart-statlog | 90,37 | 75,19 | 90,44 | 74,44 | 90,15 | 77,41 |
| ionosphere | 91,00 | 85,51 | 90,63 | 89,21 | 90,66 | 88,89 |
| iris | 90,40 | 94,00 | 90,47 | 93,33 | 90,53 | 92,67 |
| lymph | 90,00 | 77,21 | 90,07 | 76,63 | 90,14 | 75,51 |
| pima-indians | 89,83 | 67,18 | 89,77 | 67,84 | 89,99 | 65,64 |
| sonar | 91,35 | 82,17 | 91,59 | 79,94 | 91,59 | 79,55 |
| vehicle | 89,68 | 66,02 | 89,60 | 66,86 | 89,74 | 65,44 |
| waveform | 89,06 | 73,20 | 89,02 | 73,56 | 89,09 | 72,88 |
| wine | 90,34 | 96,75 | 90,22 | 97,77 | 90,45 | 95,57 |
| zoo | 90,99 | 89,43 | 90,99 | 89,43 | 91,29 | 86,66 |
| | **90,17** | **76,30** | **90,10** | **76,99** | **90,18** | **76,13** |

Figura 4.24: Taula dels resultats del model CBR amb RDCL esborrant el conjunt DCL

| | euclidean 3 | | | | manhattan 3 | | | | manhattan 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | no forget | | forget | | no forget | | forget | | no forget | | forget | |
| | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s |
| breast-w | 90,62 | 96,14 | 89,34 | 96,14 | 90,57 | 96,70 | 89,46 | 96,70 | 90,36 | 96,70 | 88,48 | 96,70 |
| bupa | 97,19 | 63,24 | 85,07 | 61,75 | 96,84 | 62,36 | 85,39 | 62,61 | 96,00 | 62,05 | 76,12 | 61,13 |
| cmc | 93,65 | 44,53 | 76,71 | 44,67 | 93,46 | 44,40 | 76,68 | 44,67 | 97,62 | 44,26 | 69,07 | 43,71 |
| glass | 91,78 | 66,31 | 81,36 | 66,27 | 92,01 | 69,67 | 81,59 | 69,63 | 90,98 | 69,67 | 73,55 | 67,38 |
| heart-statlog | 93,52 | 74,07 | 86,04 | 74,81 | 93,81 | 77,04 | 86,56 | 77,41 | 92,78 | 76,67 | 81,56 | 77,41 |
| ionosphere | 91,62 | 86,64 | 87,21 | 87,48 | 91,14 | 90,60 | 87,98 | 90,60 | 90,88 | 90,60 | 85,16 | 90,02 |
| iris | 90,00 | 95,33 | 88,27 | 95,33 | 90,13 | 94,00 | 88,40 | 94,00 | 90,33 | 94,00 | 87,47 | 94,00 |
| lymph | 91,28 | 83,28 | 84,32 | 82,69 | 91,22 | 83,28 | 84,32 | 82,69 | 91,55 | 83,28 | 79,32 | 80,72 |
| pima-indians | 94,79 | 70,20 | 85,65 | 70,59 | 94,74 | 69,82 | 85,20 | 70,21 | 93,65 | 69,69 | 78,14 | 70,07 |
| sonar | 92,26 | 86,84 | 87,26 | 85,88 | 92,74 | 86,01 | 87,64 | 85,99 | 92,50 | 86,01 | 82,84 | 85,99 |
| vehicle | 92,07 | 69,43 | 82,10 | 69,76 | 91,62 | 69,81 | 82,30 | 70,16 | 92,29 | 69,69 | 76,08 | 69,22 |
| waveform | 91,62 | 73,52 | 84,00 | 74,08 | 91,59 | 73,36 | 83,87 | 73,84 | 93,99 | 73,32 | 81,59 | 74,26 |
| wine | 90,11 | 95,64 | 88,60 | 95,64 | 90,28 | 96,69 | 89,10 | 97,24 | 90,73 | 96,69 | 88,03 | 97,24 |
| zoo | 90,30 | 94,63 | 88,12 | 95,63 | 90,30 | 94,63 | 88,12 | 95,63 | 90,20 | 94,63 | 85,84 | 96,40 |
| | **92,20** | **78,56** | **85,29** | **78,62** | **92,18** | **79,17** | **85,47** | **79,38** | **92,42** | **79,09** | **80,95** | **78,88** |

Figura 4.25: Taula dels resultats del model ACBR amb la funció de Retain DD

| | euclidean 3 | | | | manhattan 3 | | | | manhattan 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | no forget | | forget | | no forget | | forget | | no forget | | forget | |
| | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s |
| breast-w | 90,26 | 96,14 | 89,06 | 96,14 | 90,23 | 96,70 | 89,18 | 96,70 | 90,24 | 96,70 | 88,33 | 96,70 |
| bupa | 92,72 | 62,93 | 80,75 | 61,43 | 92,70 | 61,75 | 81,25 | 61,99 | 92,46 | 61,75 | 72,72 | 61,70 |
| cmc | 94,74 | 44,40 | 77,83 | 44,26 | 94,73 | 44,54 | 78,01 | 44,26 | 94,33 | 44,40 | 66,44 | 43,51 |
| glass | 92,62 | 68,63 | 82,15 | 68,61 | 92,29 | 71,54 | 82,48 | 71,52 | 92,01 | 71,54 | 74,67 | 69,70 |
| heart-statlog | 91,56 | 74,44 | 84,37 | 74,44 | 91,44 | 77,04 | 84,26 | 77,04 | 91,30 | 77,04 | 80,15 | 77,41 |
| ionosphere | 91,05 | 87,21 | 86,72 | 87,77 | 90,85 | 90,60 | 87,64 | 90,60 | 90,77 | 90,60 | 85,07 | 90,02 |
| iris | 90,47 | 95,33 | 88,67 | 95,33 | 90,47 | 94,00 | 88,73 | 94,00 | 90,47 | 94,00 | 87,60 | 94,00 |
| lymph | 91,01 | 83,28 | 84,19 | 82,69 | 91,01 | 83,28 | 84,26 | 82,69 | 90,95 | 83,28 | 78,78 | 80,72 |
| pima-indians | 91,99 | 70,34 | 82,92 | 70,86 | 92,20 | 69,17 | 82,94 | 70,08 | 91,97 | 69,17 | 76,91 | 70,20 |
| sonar | 91,01 | 87,32 | 85,82 | 86,79 | 91,15 | 86,01 | 85,87 | 85,99 | 91,06 | 86,01 | 80,96 | 85,99 |
| vehicle | 92,35 | 69,32 | 82,60 | 70,01 | 92,45 | 69,69 | 82,98 | 70,16 | 92,27 | 69,81 | 75,93 | 68,98 |
| waveform | 91,64 | 73,46 | 84,02 | 74,16 | 91,67 | 73,36 | 83,99 | 73,86 | 91,37 | 73,36 | 79,19 | 74,42 |
| wine | 90,34 | 95,64 | 88,82 | 95,64 | 90,17 | 96,69 | 89,04 | 97,24 | 90,22 | 96,69 | 87,58 | 97,24 |
| zoo | 90,50 | 95,63 | 88,32 | 95,63 | 90,50 | 95,63 | 88,32 | 95,63 | 90,40 | 95,63 | 85,94 | 96,40 |
| | **91,59** | **78,86** | **84,73** | **78,84** | **91,56** | **79,28** | **84,92** | **79,41** | **91,42** | **79,28** | **80,02** | **79,07** |

Figura 4.26: Taula dels resultats del model ACBR amb la funció de Retain DE

| | euclidean 3 | | | | manhattan 3 | | | | manhattan 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | no forget | | forget | | no forget | | forget | | no forget | | forget | |
| | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s |
| breast-w | 90,39 | 96,14 | 89,18 | 96,14 | 90,33 | 96,70 | 89,27 | 96,70 | 90,33 | 96,70 | 88,43 | 96,70 |
| bupa | 93,71 | 62,93 | 81,83 | 61,72 | 93,83 | 61,75 | 82,46 | 61,99 | 93,83 | 61,75 | 74,09 | 61,70 |
| cmc | 95,55 | 44,53 | 78,74 | 43,92 | 95,55 | 44,47 | 78,85 | 44,12 | 95,55 | 44,47 | 67,54 | 43,30 |
| glass | 93,13 | 68,63 | 82,62 | 68,61 | 92,85 | 71,54 | 82,90 | 71,52 | 92,85 | 71,54 | 75,42 | 69,70 |
| heart-statlog | 92,52 | 74,81 | 85,22 | 74,81 | 92,26 | 77,41 | 85,07 | 77,41 | 92,26 | 77,41 | 81,15 | 77,41 |
| ionosphere | 91,31 | 86,93 | 86,92 | 87,77 | 90,94 | 90,60 | 87,78 | 90,60 | 90,94 | 90,60 | 85,27 | 90,02 |
| iris | 90,47 | 95,33 | 88,67 | 95,33 | 90,60 | 94,00 | 88,87 | 94,00 | 90,60 | 94,00 | 87,73 | 94,00 |
| lymph | 91,69 | 83,28 | 84,86 | 82,69 | 91,69 | 83,28 | 84,93 | 82,69 | 91,69 | 83,28 | 79,59 | 80,72 |
| pima-indians | 92,99 | 70,07 | 83,95 | 70,59 | 93,09 | 69,17 | 83,78 | 70,08 | 93,09 | 69,17 | 77,75 | 70,07 |
| sonar | 91,25 | 87,32 | 86,11 | 86,79 | 91,39 | 86,01 | 86,20 | 85,99 | 91,39 | 86,01 | 81,49 | 85,99 |
| vehicle | 93,06 | 69,32 | 83,18 | 70,01 | 93,01 | 69,81 | 83,51 | 70,16 | 93,01 | 69,81 | 76,65 | 68,98 |
| waveform | 92,66 | 73,44 | 85,02 | 74,00 | 92,66 | 73,40 | 84,96 | 73,80 | 92,66 | 73,40 | 80,43 | 74,28 |
| wine | 90,45 | 95,64 | 88,93 | 95,64 | 90,34 | 96,69 | 89,16 | 97,24 | 90,34 | 96,69 | 87,64 | 97,24 |
| zoo | 90,50 | 95,63 | 88,32 | 95,63 | 90,50 | 95,63 | 88,32 | 95,63 | 90,50 | 95,63 | 85,94 | 96,40 |
| | **92,12** | **78,86** | **85,25** | **78,83** | **92,07** | **79,32** | **85,43** | **79,42** | **92,07** | **79,32** | **80,65** | **79,04** |

Figura 4.27: Taula dels resultats del model ACBR amb la funció de Retain LE

| | euclidean 3 | | | | manhattan 3 | | | | manhattan 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | no forget | | forget | | no forget | | forget | | no forget | | forget | |
| | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s |
| breast-w | 91,39 | 96,14 | 91,10 | 96,14 | 91,32 | 96,70 | 90,99 | 96,70 | 91,40 | 96,70 | 90,92 | 96,70 |
| bupa | 93,77 | 63,22 | 81,62 | 61,14 | 93,13 | 61,75 | 81,77 | 61,99 | 93,51 | 62,04 | 73,97 | 61,40 |
| cmc | 95,14 | 44,33 | 78,47 | 44,26 | 95,09 | 44,47 | 78,57 | 44,26 | 94,82 | 44,33 | 67,46 | 43,51 |
| glass | 92,99 | 68,63 | 82,76 | 68,15 | 92,80 | 71,54 | 82,90 | 71,52 | 92,66 | 71,54 | 75,47 | 69,70 |
| heart-statlog | 91,96 | 74,07 | 84,96 | 74,44 | 92,00 | 76,67 | 84,93 | 77,04 | 91,96 | 76,67 | 81,26 | 77,41 |
| ionosphere | 91,37 | 87,21 | 87,38 | 87,77 | 91,03 | 90,60 | 88,35 | 90,60 | 91,11 | 90,60 | 86,21 | 90,02 |
| iris | 90,60 | 95,33 | 88,93 | 95,33 | 90,67 | 94,00 | 89,07 | 94,00 | 90,80 | 94,00 | 88,33 | 94,00 |
| lymph | 91,42 | 83,28 | 84,66 | 82,69 | 91,42 | 83,28 | 84,73 | 82,69 | 91,62 | 83,28 | 79,39 | 81,31 |
| pima-indians | 92,49 | 70,47 | 83,54 | 70,86 | 92,72 | 69,30 | 83,50 | 70,21 | 92,43 | 69,30 | 77,55 | 70,21 |
| sonar | 91,73 | 87,32 | 86,78 | 86,35 | 91,68 | 86,01 | 86,83 | 85,99 | 91,92 | 86,01 | 82,26 | 85,99 |
| vehicle | 92,80 | 69,32 | 83,32 | 69,89 | 92,83 | 69,69 | 83,77 | 70,04 | 92,49 | 70,05 | 77,02 | 69,10 |
| waveform | 92,62 | 73,48 | 85,97 | 74,18 | 92,55 | 73,28 | 85,61 | 73,86 | 92,02 | 73,28 | 80,61 | 74,44 |
| wine | 90,45 | 95,64 | 88,93 | 95,64 | 90,28 | 96,69 | 89,27 | 97,24 | 90,34 | 96,69 | 87,87 | 97,24 |
| zoo | 91,09 | 95,63 | 89,11 | 95,63 | 91,09 | 95,63 | 89,11 | 95,63 | 91,09 | 95,63 | 86,93 | 96,40 |
| | **92,13** | **78,86** | **85,54** | **78,75** | **92,04** | **79,26** | **85,67** | **79,41** | **92,01** | **79,29** | **81,09** | **79,10** |

Figura 4.28: Taula dels resultats del model ACBR amb la funció de Retain MG

| | DE | | | | | | MG | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D | | DL | | DCL | | D | | DL | | DCL | |
| | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s | %c | %s |
| breast-w | 87,95 | 96,70 | 87,60 | 96,70 | 88,58 | 96,70 | 89,76 | 96,84 | 89,40 | 96,70 | 90,40 | 96,70 |
| bupa | 68,14 | 61,14 | 69,62 | 63,98 | 76,26 | 61,98 | 69,01 | 61,14 | 70,29 | 63,98 | 76,90 | 61,98 |
| cmc | 62,34 | 44,19 | 52,20 | 45,13 | 72,15 | 43,65 | 62,98 | 44,05 | 52,76 | 45,34 | 72,82 | 43,65 |
| glass | 71,26 | 71,61 | 72,52 | 69,74 | 79,21 | 69,72 | 71,64 | 71,61 | 72,94 | 69,74 | 79,58 | 69,72 |
| heart-statlog | 76,11 | 79,26 | 75,04 | 78,89 | 82,19 | 77,41 | 76,89 | 78,89 | 75,67 | 78,89 | 82,93 | 77,41 |
| ionosphere | 79,69 | 89,17 | 86,55 | 90,03 | 87,26 | 91,17 | 80,37 | 89,17 | 87,24 | 90,03 | 88,06 | 91,17 |
| iris | 86,00 | 95,33 | 87,67 | 94,67 | 88,67 | 94,00 | 86,40 | 95,33 | 88,00 | 94,67 | 89,00 | 94,00 |
| lymph | 75,61 | 80,10 | 81,08 | 82,69 | 82,09 | 81,27 | 76,15 | 80,10 | 81,55 | 82,69 | 82,64 | 81,27 |
| pima-indians | 73,36 | 70,73 | 72,63 | 72,01 | 79,64 | 70,74 | 73,78 | 70,73 | 73,13 | 71,75 | 80,22 | 70,87 |
| sonar | 77,74 | 84,12 | 84,86 | 84,62 | 85,48 | 85,51 | 78,56 | 84,12 | 85,63 | 85,51 | 86,39 | 84,62 |
| vehicle | 72,99 | 69,48 | 71,29 | 70,14 | 79,28 | 69,19 | 73,74 | 69,72 | 72,04 | 69,90 | 80,09 | 69,19 |
| waveform | 70,36 | 74,54 | 80,17 | 74,38 | 80,54 | 74,58 | 71,93 | 74,60 | 81,82 | 74,44 | 82,33 | 74,58 |
| wine | 85,06 | 97,24 | 89,04 | 97,24 | 89,04 | 97,24 | 85,17 | 97,24 | 89,27 | 97,24 | 89,27 | 97,24 |
| zoo | 85,15 | 94,52 | 88,22 | 95,63 | 88,12 | 95,63 | 86,04 | 94,52 | 89,01 | 95,63 | 88,91 | 95,63 |
| | **76,55** | **79,15** | **78,46** | **79,70** | **82,75** | **79,20** | **77,31** | **79,15** | **79,20** | **79,75** | **83,54** | **79,15** |

Figura 4.29: Taula dels resultats del model ACBR amb el pas RDCL

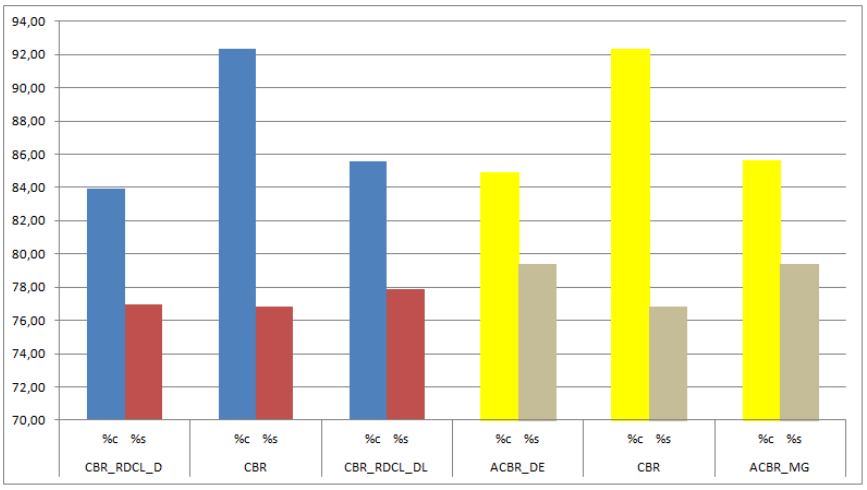


Figura 4.30: Gràfic comparatiu entre el model CBR i els models CBR amb RDCL i ACBR

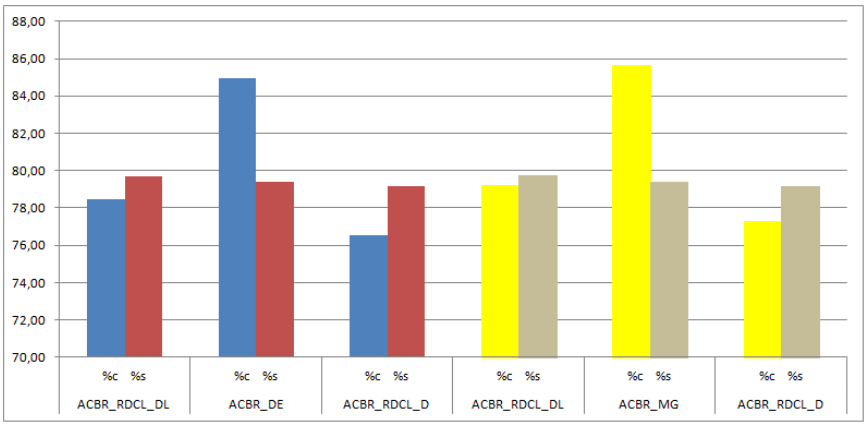


Figura 4.31: Gràfic comparatiu entre el model ACBR i el models ACBR amb RDCL

Al analitzar els resultats de la Taula 4.29, es veu que les configuracions amb les que s'obté millor rendiment són (tot i que les 6 configuracions donen resultats molt semblants): amb la distància Manhattan, 3 casos de recuperació, esborrant els casos del conjunt D, amb la funció de Retain DE i fent forget, amb la distància Manhattan, 3 casos de recuperació, esborrant els casos del conjunt DL, amb la funció de Retain DE i fent forget i amb la distància Manhattan, 3 casos de recuperació, esborrant els casos del conjunt DL, amb la funció de Retain MG i fent forget. Marquem aquestes caselles en vermell en la taula de la Figura 4.29.

En les Figures 4.30 i 4.31, es veu gràficament la millora del rendiment.

En la primera meitat del gràfic de la Figura 4.30 es veu la comparació entre el model CBR i el model CBR fent el pas de RDCL, esborrant els conjunts D i DL. En la segona meitat, es veu la comparació entre el model CBR i el model ACBR, fent les funcions de Retain DE i MG. En aquest gràfic es veu clarament com el rendiment millora molt al executar el model ACBR.

En la primera meitat del gràfic de la Figura 4.31 es veu la comparació entre el model ACBR i el model ACBR fent el pas de RDCL, esborrant els conjunts D i DL i usant en tots els casos la funció de Retain DE. En la segona meitat es veu la mateixa comparació, però usant la funció de Retain MG. Com es pot veure el percentatge d'encerts no varia gairé, però si que ho fa el percentatge de casos usats. Per tant, el rendiment encara augmentarà més.

# Capítol 5

# Conclusions

Aquest projecte presenta un total de 4 models de *Raonament Basat en Casos*: **CBR**, **CBR amb RDCL**, **ACBR** i **ACBR amb RDCL**. Aquests models presenten un raonament usant les tècniques d'*aprendre* i *oblidar* d'una manera similar a com ho fem els humans. Es considera que aquestes tècniques es combinen en els diferents models per garantir el **rendiment**, és a dir la *precisió* i una *base de casos compacte*.
Els models ACBR desenvolupen de forma iterativa el *Manteniment de la Base de Casos* mitjançant diferents estratègies de retenció i oblit que utilitzen una mesura de **bondat** per decidir quins casos esborrar i quins retenir. Els models RDCL desenvolupen de manera inicial el *Manteniment de la Base de Casos* mitjançant diferents estratègies d'oblit que utilitzen les propietats dels casos per decidir si aquests han de ser esborrats o no.

Amb les proves que s'han realitzat hem demostrat l'eficàcia de la combinació d'aquests models aplicant-los en un conjunt de 14 datasets diferents. En general, els nostres experiments indiquen que el model **ACBR amb RDCL** ofereix beneficis de rendiment. Els resultats de l'avaluació confirmen que la majoria de les configuracions del nostre model produeixen millores significatives en termes de rendiment en comparació amb el model **CBR** usat com a referència. Però el nostre anàlisi demostra que, si estem particularment interessats en el rendiment, algunes configuracions poden ser més convenients que d'altres.
El model **ACBR amb RDCL** obté un major rendiment que la configuració del model **CBR**. Aquest bon rendiment s'aconsegueix perquè la base de casos s'adapta de forma continua, el que manté una alta diversitat de casos i alhora esborra aquells que són perjudicials. Hi ha dos aspectes que controlen el rendiment del nostre model. En primer lloc hi ha una reducció de la base de casos que permet una recuperació més ràpida i per tant millora l'eficiència. En segon lloc hi ha una millora de la precisió.

En conclusió, l'avaluació dels nostres models fa evident que el manteniment de la base de cas és efectiu. Però també mostra que els estudis en aquest camp encara són inicials i que queda molt camí per recórrer. Trobar altres mètodes de **Manteniment de la Base de Casos** i les seves possibles combinacions ens ajudaran en el futur a trobar un sistema de **Raonament Basat en Casos** amb un alt nivell de rendiment.

# Bibliografia


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

[2] K. D. Althoff. *Knowledge acquisition in the domain of CNC machine centers; the MOLTKE approach*, J. Boose, B. Gaines and J.-G. Ganascia (eds.): EKAW-89; Third European Workshop on Knowledge-Based Systems, Paris, p.180-195 (1989)

[3] K. Ashley. *Modeling legal arguments: Reasonin with cases and hypotheticals*, MIT Press, Bradford Books, Cambridge (1991)

[4] R. Bareiss. *Exemplar-based knowledge acquisition: A unified approach to concept representation, classification, and learning* Academic Press, Boston (1989)

[5] R. Bareiss, B. Porter. *PROTOS: An experiment in knowledge acquisition for heuristic classification tasks*, Proceedings of the First International Meeting on Advances in Learning (IMAL), Les Arcs, France, p.159-174 (1986)

[6] A. G. Barton, R. S. Sutton. *Reinforcement Learning*, An introduction, The MIT Press (1998)

[7] P. Cunningham, S. J. Delany. *An analysis of case-based editing in a spam filtering system*, P. Funk, P. González-Calero (Eds.), Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR 2004), vol. 3155, p.128-141 (2004)

[8] S. J. Delany. *The good, the bad and the incorrectly classified: Profiling cases for case-base editing*, L. McGinty, D. C. Wilson (Eds.), Proceedings of the Eighth International Conference on Case-Based Reasoning ICCBR 09, vol.5650, p-135-149 (2009)

[9] S.J. Delany, N. Segata, B. Namee. *Profiling instances in noise reduction*, Knowledge-Based Systems, vol.31, p.28-40 (2012)

[10] I. Iglezakis, T. Reinartz, T. Roth-Berghofer. *Maintenance memories: beyond concepts and techniques for case base maintenance*

[11] T. Keane, B. Smyth. *Remembering to forget: A competence–preserving case deletion policy for case–based reasoning systems*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, (IJCAI-95), Morgan Kaufmann Publishers Inc., p.377-382 (1995)

[12] J. Kolodner. *Maintaining organization in a dynamic long-term memory*, Cognitive Science, vol.7, p.243-280 (1983)

[13] J. Kolodner. *Reconstructive memory, a computer mode!*, Cognitive Science, vol.7, p.281-328 (1983)

[14] D. B. Leake, D. C. Wilson. *Categorizing case-based maintenance: dimensions and directions*, Computer Science Department

[15] D. B. Leake, D. C. Wilson *When experience is wrong: Examining CBR for changing tasks and environments*, Proceedings of the Third International Conference on Case–Based Reasoning (1999)

[16] M. López-Sánchez, M. Salamó. *Adaptive case-based reasoning using retention and forgetting strategies*, Knowledge-Based Systems, vol.24, p-230-247 (2011)

[17] T. R. Martinez, D. Randall Willson. *Improved heterogeneous distance functions*, Journal of artificial intelligence research, vol.6, p.1-34 (1997)

[18] K. Racine, Q. Yang. *Maintaining unstructured case bases*, Proceedings of the International Conference on Case Based Reasoning, p.553-564 (1997)

[19] T. Reinartz, T. Roth-Berghofer. *A maintenance manual for case-based reasoning systems*, David W. Aha and Ian Watson, editors, Proceedings of the Fourth International Conference on Case-Based Reasoning, Vancouver, Canada, p.452-466 (2001)

[20] A. M. Richter, S. Weiss. *Similarity, uncertainty and case-based reasoning in PATDEX*, R.S. Boyer (ed.): Automated reasoning, essays in honour of Woody Bledsoe, Kluwer, p.249-265 (1991)

[21] C. Riesbeck, R. Shank. *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Yersey (1989)

[22] E. Rissland. *Examples in legal reasoning: Legal hypotheticals*, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI, Karlsruhe (1983)

[23] R. Schank. *Dynamic memory; a theory of reminding and learning in computers and people*, Cambridge University Press (1982)

[24] E. Swanson. *The dimensions of maintenance*, Proceedings of the 2nd International Conference on Software Engineering, p.492-497 (1976)

[25] Q. Yang, J. Zhu. *Remembering to add: Competence–preserving case addition policies for case base maintenance*, Proceedings of the International Joint Conference in Artificial Intelligence, p.234-239 (1999)

[26] M. Guijosa López. *Implementación en Java de un sistema clasificador basado en casos*, TFG Enginyeria tècnica en informàtica de sistemes UB (2008)

[27] Viquipèdia, Internet. *http://www.wikipedia.org/*

# Annex 1: Implementació i resultats

En la Secció 4.2 s'han vist els mètodes implementats per poder realitzar aquest projecte. Es veuen ara els mètodes específics d'un CBR bàsic:

Mètodes i atributs de la interfície ICaseBase i de la classe CaseBase (Figura 1):

- *Atributs*
  - **attributeNames**: noms dels atributs d'un cas de la base de casos
  - **casBaseCases**: casos de la base de casos
  - **maxAttributeValues**: valor màxim dels atributs d'un cas de la base de casos
  - **minAttributeValues**: valor mínim dels atributs d'un cas de la base de casos
  - **name**: nom de la base de casos
  - **testCases**: casos del conjunt de Test
- *Mètodes*
  - **addCase**: Mètode que afegeix un cas a la base de casos
  - **calculateMaxAttributeValues**: Mètode que calcula el valor màxim dels atributs d'una col·lecció de casos
  - **calculateMinAttributeValues**: Mètode que calcula el valor mínim dels atributs d'una col·lecció de casos
  - **calculateMinAttributeValues**: Mètode que es queda amb els valors mínims i màxims dels atributs entre els casos de test i els casos de la base de casos
  - **deleteCase**: Mètode que esborra un cas de la base de casos
  - **deleteCaseBaseCases**: Mètode que esborra tos els casos de la base de casos
  - **deleteTestCases**: Mètode que esborra un cas del conjunt de Train
  - **doTest**: Mètode que fa un test
  - **getCase**: Mètode que retorna el cas d'una posició determinada
  - **getCaseBaseCases**: Mètode que retorna els casos d'una base de casos
  - **getDatasetFolds**: Mètode que retorna el folds d'un fitxer donat
  - **getDatasetNames**: Mètode que retorna el nom dels datasets
  - **getMaxAttributeValues**: Mètode que retorna els valors màxims dels atributs
  - **getMinAttributeValues**: Mètode que retorna els valors mínims dels atributs
  - **getNumCases**: Mètode que retorna el número de casos d'un conjunt de casos
  - **init**: Mètode que inicialitza una base de casos
  - **uploadCases**: Mètode que guarda els casos a una base de casos a partir d'un fitxer

Mètodes i atributs de la interfície ICase i de la classe Case (Figura 4.4):

- *Atributs*
  - **iAttributes**: atributs d'un cas
  - **name**: nom del cas
- *Mètodes*
  - **addAttribute**: Mètode que afegeix un atribut a un cas

  - **getAttributeClass**: Mètode que retorna la classe (o solució) d'un cas
  - **getIAttributes**: Mètode que retorna els atributs d'un cas
  - **getName**: Mètode que retorna el nom d'un atribut
  - **setName**: Mètode que assigna un nom a un cas

Mètodes i atributs de la interfície IAttribute i de les classes NumericalAtrribute i Nominal Atribute (Figura 4.5):

- *Atributs*
  - **possibleValues**: possibles valors nominals d'un atribut
  - **type**: tipus d'atribut, pot ser nominal o numèric
  - **value**: valor d'un atribut
- *Mètodes*
  - **getPossibleValues**: Mètode que retorna els possibles valors nominals d'un atribut
  - **getValue**: Mètode que retorna el valor d'un cas
  - **setPossibleValues**: Mètode que assigna els possibles valors nominals a un atribut
  - **setType**: Mètode que retorna el tipus d'un atribut
  - **setValue**: Mètode que retorna el valor d'un atribut

Mètodes de la interfície ITest i de la classe Test (Figura 4.6):

- **executeTest**: Mètode que executa un test

Mètodes les interfícies ICBR, ICBRStepRetrieve, ICBRStepReuse, ICBRStepRevise i ICBRStepRetain i de les classes CBR, CBRStepRetrieve, CBRStepReuse, CBRStepRevise i CBRStepRetain (Figura 2):

- **makeCBRCicle**: Mètode que executa un cicle de CBR
- **executeStepRetrive**: Mètode que executa el pas de Retrieve d'un cicle de CBR
- **executeStepReuse**: Mètode que executa el pas de Reuse d'un cicle de CBR
- **executeStepRevise**: Mètode que executa el pas de Revise d'un cicle de CBR
- **executeStepRetain**: Mètode que executa el pas de Retain d'un cicle de CBR

Mètodes i atributs de la interfície IDistanceFunction i de les classes DistanceEuclidean, DistanceManhattan i Distance MinkowskyR3 (Figura 4.11):

- *Atributs*
  - **name**: nom de la funció de distància ("euclidean", "manhattan" o "minkowsky")
- *Mètodes*
  - **getName**: Mètode que retorna el nom de la funció de distància
  - **calculateDistance**: Mètode que calcula la distància entre els atributs dels casos de la base de casos i el cas de test

Mètodes i atributs de la interfície IResult i de la classe Result (Figura 4.14):

- *Atributs*
  - **fail**: número total d'errors al executar un test
  - **percentFail**: percentatge d'errors d'un test
  - **percentSuccess**: percentatge d'encerts d'un test
  - **fail**: número total d'encerts al executar un test
- *Mètodes*
  - **getPercentSuccess**: Mètode que retorna el percentatge d'encerts d'un test
  - **setFail**: Mètode que assigna el número d'errors d'un test
  - **setPercentFail**: Mètode que assigna el percentatge d'errors d'un test
  - **setPercentSuccess**: Mètode que assigna el percentatge d'encerts d'un test
  - **setSuccess**: Mètode que assigna el número d'encerts d'un test

# Annex 2: Manual d'usuari

Tot el que es descriu a continuació es refereix exclusivament al sistema operatiu Windows. Per instal·lar i executar el sistema en el nostre ordinador, necessitem els següents programes: **Java 1.6.0:** Es pot descarregar des de la pàgina web de *Sun Microsystems* http://java.sun.com/. **Eclipse** Es pot descarregar des de la pàgina web http://eclipse-sdk.softonic.com/.

Un cop es té instal·lat l'*Eclipse* s'ha d'anar a la carpeta a on es troba el codi del TFG, anomenada **"codiTFG"**, i exportar-lo cap a *Eclipse*. Un cop fet això, es fa la ruta: *codiTFG/src/Main/Main* on s'obre la classe "Main". El mètode principal del programa és el mètode "main". Aquest mètode és l'encarregat de muntar les configuracions amb les que el vol fer el test. Esta preparat per executar totes les configuracions possibles, però no és recomanables, ja que, per la gran quantitat de configuracions (10206 configuracions!!), el programa triga massa temps a executar. També esta preparat perquè cada usuari escrigui la combinació que vol fer (linies 68-88 del codi). De fet, el codi s'entrega amb les configuracions més òptimes trobades en la Secció 4.3 (veure Figura1).

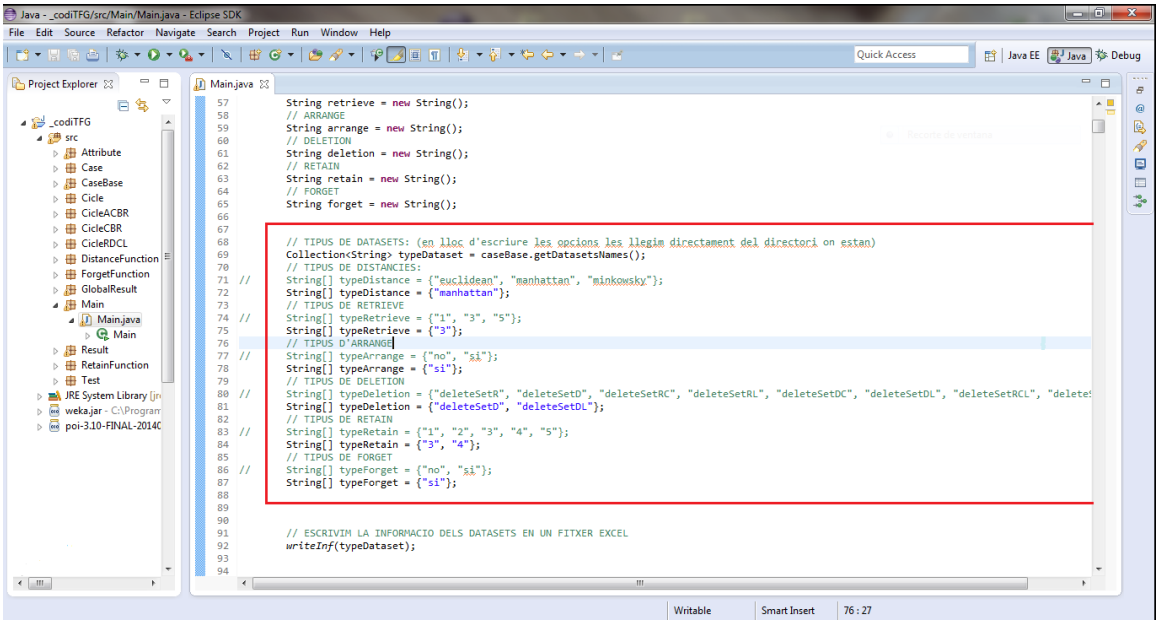


Figura 1: Pantalla Main del codi, mètode "main". El requadre en vermell indica el tros on l'usuari pot canviar les configuracions dels tests.

Des del mètode "Main" es pot demanar qualsevol tipus de configuració, excepte els datasets. El sistema està preparat per rebre qualsevol dataset (tot i que hi ha un nombre limitat de datases que es poden posar). En l'entrega d'aquest projecte, junt amb aquest codi i la memòria, s'ha ajuntat una carpeta amb els datasets que es fan servir. Per tal d'executar el codi correctament cal que l'usuari es creï una carpeta amb la següent ruta: *C:/datasetsCBR*,

i aquí dintre posar la carpeta de datasets que s'adjunta. Es podria canviar aquesta ruta, però durant el programa es llegeix diverses vegades i s'hauria de canviar en tots els llocs.

S'ha comentat que hi ha un nombre limitat de datasets, en el projecte s'ha treballat amb 14 datasets i per això aquest és el màxim permès. Aquesta limitació era necessària per poder escriure l'excel d'informació dels datasets. Es pot canviar aquesta limitació en el mètode *"writeInf"* de la classe "Main". En les linies 547-717 hi ha uns *if / else if / else* que estan fins a 14, si es vol augmentar el nombre de datasets s'han d'augmentar els *else if*. No passa res si s'executa amb un nombre menor de datasets.

Com a resultats el sistema retorna dos fitxers excel. El primer és un fitxer amb informació del dataset que s'està tractant. Es pot canviar la ubicació on es guarda aquest fitxer anant al mètode *"writeInf"* de la classe "Main", línia 721. El segon fitxer excel que retorna és un fitxer amb els resultats del test. També es pot canviar la seva ubicació, s'ha d'anar al mètode *"writeResults"* de la classe "Main", línia 1069 (veure Figura 2). Cal dir que en els dos casos es necessari posar *".xls"*.

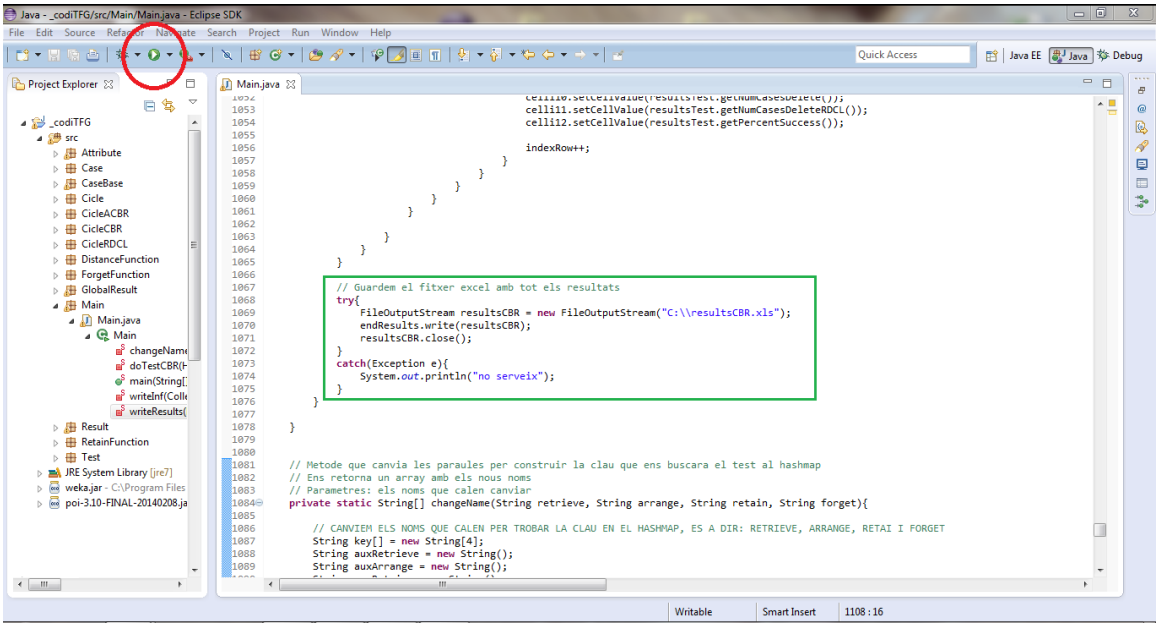


Figura 2: Pantalla Main del codi, mètode "writeResults". El requadre en verd indica el tros on l'usuari pot canviar la ubicació dels resultats del test. La rodona vermella indica el lloc on està el "Run".

Quan ja estigui tot preparat, l'usuari no té més que clicar al botó "Run" perquè el programa s'executi (veure Figura 2).

Amb aquest mètode de programa es vol deixar el codi obert, per futures implementacions en les que els models de *Raonament Basat en Casos* canviïn.