Anàlisi de l'avaluació incremental i de l'aplicació eficient de l'Algorisme Genètic en un Sistema Classificador

Ester Bernadó i Mansilla, Xavier Llorà i Fàbrega, Josep M. Garrell i Guiu

Grup de Recerca en Sistemes Intel.ligents
Enginyeria i Arquitectura La Salle. Universitat Ramon Llull
Passeig Bonanova, 8. 08022 Barcelona
{esterb, xevil, josepmg}@salleURL.edu

Resum

Aquest article parteix de l'anàlisi del sistema MOLeCS, un sistema classificador (CS) que evoluciona un conjunt de regles de classificació a partir d'algorismes genètics (GA). El sistema es descomposa en dues fases: l'avaluació de les regles i el GA. El cost de l'avaluació de les regles és elevat i depèn del nombre d'exemples d'entrenament. L'article estudia i proposa l'ús d'una avaluació incremental que permeti intercalar-hi l'aplicació del GA, accelerant d'aquesta manera l'aprenentatge i reduint el cost computacional. A més, es proposen altres esquemes de selecció i de reemplaçament del GA que són més eficients que en el sistema original. El sistema resultant, anomenat IMO, presenta característiques comunes del sistema MOLeCS i de d'altres CS actuals com l'XCS. Els resultats demostren la millora del cost computacional i de l'eficiència del GA respecte MOLeCS. Per altra banda, el sistema proposat presenta un espai solució inferior a l'XCS, cosa que permet millorar-ne el rendiment en determinats problemes.

Paraules clau: computació evolutiva, aprenentatge artificial, sistemes classificadors.

1 Introducció

En el camp dels algorismes genètics [7], podem definir un sistema classificador (CS) com un sistema d'aprenentatge que evoluciona un conjunt de regles -anomenades classificadors-, sota un esquema d'aprenentatge per reforçament. El sistema va adquirint el coneixement mitjançant: a) l'adaptació de la qualitat de les regles, a partir de la interacció de l'entorn, del qual obté recompenses i b) de l'aplicació d'un GA que intervé en la descoberta de noves regles que millorin globalment el rendiment del sistema.

Els classificadors tenen dues mesures associades: la *predicció* i el *fitness*. La predicció és un estimat de la recompensa que el classificador rep de l'entorn i és per tant una mesura de la utilitat del mateix. El *fitness* s'usa en el GA per guiar l'evolució: els classificadors amb un *fitness* elevat tindran més probabilitat de reproduir-se i repartir el seu material genètic en la població, mentre que els classificadors amb un *fitness* baix tendiran a desaparèixer.

El sistema XCS [13, 14] és un CS que representa l'estat actual de l'art en el camp de l'aprenentatge amb algorismes genètics. Les seves principals característiques són: a) la definició del *fitness*, que es basa en la precisió de la predicció de la recompensa de l'entorn i b) l'ús d'un *niched GA*, que s'aplica localment sobre les espècies definides en els conjunts d'activació. Aquestes característiques fan que el sistema evolucioni conjunts de regles complets, consistents i mínims [13, 8] i per aquest motiu s'està aplicant amb èxit en molts entorns [15].

MOLeCS [1, 2] és un sistema classificador que es basa en algorismes genètics multiobjectiu. El sistema defineix el *fitness* de una regla segons la *precisió* i la *generalització* de la mateixa. La precisió es calcula com el percentatge d'encerts de la regla, mentre que la generalització d'una regla s'enten com la capacitat de generalitzar exemples similars i es mesura com el percentatge d'exemples que es cobreixen. El sistema adreça l'aprenentatge d'un conjunt de regles mitjantçant l'evolució de regles acurades i generals, que globalment tendeixen a formar un conjunt de regles òptim.

Una de les diferències remarcables entre els CS clàssics i MOLeCS és l'avaluació incremental. En els CS, les regles s'avaluen incrementalment, exem-

ple per exemple, i el GA s'aplica periòdicament (cada cert nombre d'exemples). Mentre s'apren la qualitat de les regles, es va exercint la pressió selectiva cap aquelles regles que resulten més prometedores. En canvi, en MOLeCS hi ha dues fases diferenciades: l'avaluació de les regles i el GA, que es realitzen seqüencialment. El problema d'aquest esquema és el cost computacional, en especial quan el conjunt d'entrenament és de mida elevada. En aquest article estudiem la possibilitat de modificar l'avaluació del sistema MOLeCS i fer-la incremental a l'estil dels CS tradicionals, amb l'intent de millorar-ne el cost computacional. A més, estudiem un conjunt de millores en el GA destinades a guiar més eficientment la cerca del GA, com són el *restricted mating* i l'ús de mecanismes de *niching* més eficients. El sistema final proposat, anomenat IMO, presenta un cost inferior a MOLeCS i aplica un GA similar al sistema XCS. No obstant, l'espai solució és inferior a l'XCS, cosa que en determinats entorns en millora el rendiment del sistema.

L'article s'estructura de la forma següent: en primer lloc, descrivim breument el sistema MOLeCS, a partir del qual s'analitza la nova proposta (secció 3). En la secció 4 es descriu el nou sistema proposat i a continuació presentem els resultats. Finalment, la secció 6 resumeix les conclusions i línies de futur.

2 MOLeCS

A continuació detallem l'algorisme del sistema MOLeCS.

```
inicialitza la poblacio [P]
MENTRE ¬ fi
  /* Avaluacio */
  PER CADA exemple d'entrenament te𝒯
    PER CADA regla re [P]
      actualitza y_r = (acc, gen)
    FIPER
  FIPER
  assigna fitness multiobjectiu segons y
  /* GA */
  selecciona G · r individus
  recombina amb probabilitats p_c, p_m
  reemplaça els nous individus amb crowding
FIMENTRE
```

on $\mathcal{T}$ és el conjunt d'entrenament, [P] la població, que està formada per un conjunt de regles, i *acc*, *gen* són la precisió i generalització de la regla. Com es pot veure, el GA s'aplica un cop s'ha finalitzat l'etapa d'avaluació de les regles, que es realitza per tots els exemples d'entrenament $\mathcal{T}$. Un tret característic de MOLeCS és l'aplicació de tècniques multiobjectiu en el càlcul del *fitness* i l'ús de *crowding* per a mantenir un conjunt divers de regles. Per a més detalls podeu consultar [1, 2].

3 Anàlisi del sistema

3.1 Avaluació incremental

L'objectiu de definir una avaluació incremental és reduir el cost computacional. La proposta consisteix en avançar l'aplicació del GA usant les avaluacions parcials de les regles, enlloc d'esperar el càlcul complet de les avaluacions. D'aquesta manera, el GA exerceix abans la pressió evolutiva: les regles que potencialment són bones es mantindran i reproduiran, mentre que les regles amb un rendiment baix s'esborraran abans, de manera que no gastaran recursos computacionals.

Per mostrar que un esquema incremental pot millorar el cost computacional del sistema, anem a realitzar un càlcul del cost d'un sistema classificador tradicional, amb un esquema incremental, comparat amb MOLeCS. En un CS tradicional, el temps computacional depèn del temps total de l'avaluació de les regles t_{avalT}, que depèn del nombre total d'exemples presentats al sistema n_{ex} i del temps emprat per cada exemple t_{ex}. A aquest, se li suma el cost de l'aplicació del GA t_{GAT}, que és el producte del nombre de cops que s'aplica l'algorisme genètic n_{GA} pel temps emprat en cada aplicació t_{GA}:

$$\begin{aligned} t_{CS} &= t_{avalT} + t_{GAT} \\ &= n_{ex}t_{ex} + n_{GA}t_{GA} \\ &= n_{ex}(t_{map} + t_{act}) + n_{GA}(t_{sel} + t_{cr} + t_m + t_{rep}) \\ &= n_{ex}\alpha r + n_{GA}(\beta r + \gamma log_2 r) + \xi \end{aligned}$$

Per cada exemple, cal fer el mapeig amb totes les regles t_{map} i l'actualització de les mateixes t_{act}. Aquests depenen del nombre de regles r. Per altra banda, el temps del GA depèn de les fases de selecció t_{sel}, creuament t_{cr}, mutació t_m i reemplaçament t_{rep}. Habitualment, el nombre d'individus implicats és constant (2 individus). Això fa que t_{cr} i t_m siguin constants, mentre que $t_{sel} + t_{rep} = \beta r + \gamma log_2 r + \xi$ [5]. En l'anterior equació, α, β, γ i ξ són constants. Per tant, el cost de l'algorisme és: $O(n_{ex}r + n_{GA}r)$. Substituint $n_{ex} = n_{GA}\theta_{GA}$, on θ_{GA} és la freqüència d'aplicació del GA, i simplificant:

$$t_{CS} \in O(\theta_{GA} n_{GA} r) \qquad (1)$$

En el sistema MOLeCS, realitzem un càlcul complet de l'avaluació t_{avalT} i apliquem el GA amb cost t_{GA} en cada iteració:

$$\begin{aligned} t_M &= (t_{avalT} + t_{GA})n_{GA} \\ &= (t_{map} + t_{fit})n_{GA} + (t_{sel} + t_{cr} + t_m + t_{rep})n_{GA} \end{aligned}$$

on n_{GA} és el nombre d'iteracions totals de l'algorisme (que coincideix amb el nombre de cops que s'aplica el GA). El temps de l'avaluació implica: a) el mapeig entre tots els exemples d'entrenament t_{map} que depèn del nombre d'exemples en el conjunt d'entrenament $|\mathcal{T}|$ i del nombre de regles r, i b) el cost d'avaluar el *fitness* segons el mètode multiobjectiu AGR $t_{fit} = \alpha r log_2 r + \beta r$ (veure [2]). Per altra part, el genètic es descomposa en les fases de selecció, creuament, mutació i reemplaçament. Amb el mètode DC, el cost és: $r + t_{avalT}$ (veure [2]). Aplicant-ho, el cost final de MOLeCS és:

$$t_M \in O(n_{GA} r(|\mathcal{T}| + log_2 r)) \quad (2)$$

La nostra hipòtesi és que el cost d'un CS és inferior a MOLeCS si el GA aprofita les avaluacions parcials de les regles. És a dir:

$$n_{GA1}\theta_{GA} < n_{GA2}(|\mathcal{T}| + log_2 r) \quad (3)$$

on distingim entre el nombre de cops que s'aplica el GA en un CS (n_{GA1}) i en MOLeCS (n_{GA2}). Per altra banda, el nombre d'exemples entre aplicacions del GA és θ_{GA} en un CS i $|\mathcal{T}|$ en MOLeCS. Habitualment $\theta_{GA} < |\mathcal{T}|$, especialment si la mida del conjunt d'entrenament és molt elevada. El nombre de cops que cal aplicar el GA per assolir l'aprenentatge pot variar en els dos esquemes. Si reduim molt θ_{GA} en el CS, molt probablement augmentarà n_{GA1} ja que el GA actuarà amb avaluacions més preliminars i potser realitzarà algunes cerques infructuoses. En canvi, si augmentem θ_{GA}, el GA actuarà sobre avaluacions més estables i per tant, augmentarem la seva probabilitat d'èxit, disminuint doncs n_{GA1}. L'objectiu és usar un valor de θ_{GA} que permeti calcular avaluacions fiables de les regles sense necessitat d'usar $|\mathcal{T}|$ exemples, de manera que la condició (3) es compleixi.

3.2 Algorisme Genètic

A continuació, analitzem els esquemes de selecció i de reemplaçament del GA aplicat en el MOLeCS, i proposem algunes variants que en milloren el rendiment i el cost temporal.

Creuament restringit

En cada iteració del MOLeCS es generen $G \cdot r$ noves regles ($0 \leq G \leq 1$), provinents de l'aplicació del GA. Aquest selecciona $G \cdot r$ regles entre tota la població [P] i posteriorment, les recombina amb els operadors de creuament i mutació. Però hi ha un problema inherent a aquest esquema: la possible generació de *letals* [5], degut al creuament entre regles que tenen poca relació entre si. Per exemple, suposem que hem seleccionat dues regles: `00###0:0` i `110###:0` corresponents al problema del multiplexor booleà de 6 entrades[1]. Ambdúes són correctes però pertanyen a *espècies* diferents, ja que cobreixen diferents tipus d'exemples. El creuament entre aquestes pot donar solucions del tipus `010###:0` que tenen poc sentit i no contribueixen a la millora de l'aprenentatge. Per pal·liar aquest efecte, es pot restringir el creuament a les solucions properes entre si, mètode que es coneix sota el nom de *restricted mating* [5].

En els sistemes classificadors incrementals, podem definir una espècie com el conjunt de regles que s'activen sota un determinat exemple (conjunt d'activació). Per exemple, donat l'exemple `010010`, el conjunt d'activació pot estar format per regles del tipus: `01##10:1` i `01#01#:1`. El creuament entre aquestes regles donarà solucions properes a l'exemple actual i probablement més eficients. Aquesta idea fou proposada per Booker [4] i usada també en el sistema XCS [13, 14].

Reemplaçament

Un dels punts claus per al bon rendiment d'un sistema classificador és el manteniment d'un conjunt divers de regles. L'objectiu és trobar un conjunt de regles que solucioni globalment el problema d'aprenentatge, i no una única regla. Però la tendència del GA és obtenir una única solució encara que hi hagi un conjunt de solucions igualment atractives, degut a la deriva genètica [6]. L'especiació o *niching* defineix un conjunt de modificacions al GA original que eviten aquest efecte, permetent així que es trobin múltiples solucions.

El mecanisme de *niching* emprat en MOLeCS és el *crowding*, en el qual un individu en reemplaça un altre en base a la seva similitud i al *fitness*. El problema d'aquests algorismes és que requereixen un esforç computacional elevat, perque

[1] La codificació correspon a $A_1 A_0 I_3 I_2 I_1 I_0 : S$, on A_i són les adreces, I_i les entrades del multiplexor i S la sortida [13]. El caràcter # fa de comodí.

s'ha de calcular la similitud entre els individus i a més, sovint aquesta similitud pot ser imprecisa si la representació de les regles és complexa (especialment quan treballem amb problemes amb un elevat nombre d'atributs reals). Per tant, aprofitant la redefinició d'un nou sistema, ens plantegem el fet d'usar esquemes de *niching* més senzills i/o eficients. Tenim dues possibles aproximacions:

- *Reemplaçar en la mateixa espècie.* Considerem una espècie com el conjunt d'activació, d'acord amb Booker [4]. Per tant, podem reemplaçar un individu per un altre individu del mateix conjunt d'activació, escollit amb una probabilitat inversament proporcional al *fitness*. Així, mantenim la diversitat en la població, alhora que exercim una pressió selectiva.
- *Delete T3.* En el sistema XCS, un individu s'esborra amb probabilitat proporcional a la mida promig de les espècies on ha participat (on l'espècie es defineix de manera similar a Booker). La pressió selectiva cap als bons individus s'exerceix fent que la probabilitat d'esborrat sigui major si els individus tenen un *fitness* molt baix [9].

Els dos esquemes semblen bons a priori, però a més coneixem el bon rendiment del mètode $T3$, contrastat per molts experiments, sota el marc del sistema classificador XCS. Per aquest motiu, ens decidim a incorporar aquest mètode en el nostre sistema, tot i que no descartem la implementació i testeig del primer mètode com a línia de futur.

4 Descripció del sistema

4.1 Representació

Un classificador té una regla `condició:classe` i un conjunt de paràmetres associats. La codificació d'aquesta regla depèn del tipus d'atributs del problema. En aquest article, hem usat problemes amb atributs booleans i per tant, la codificació de la condició és una cadena: $\{0, 1, \#\}^k$, on k és el nombre d'atributs i # és el caràcter comodí que s'activa per qualsevol valor de l'atribut. La classe associada és un enter. Els paràmetres de cada classificador es detallen en la taula 1.

La precisió de la regla *acc* es calcula com el percentatge d'encerts: el nombre d'exemples correctament classificats respecte els exemples coberts:

$$acc = \frac{exc}{exm} \tag{4}$$

Taula 1: Paràmetres del classificador en IMO.

exc	# d'exemples correctament classificats
exm	# d'exemples coberts
t_{born}	iteració en que es crea la regla
ns	estimació de la mida de l'espècie
acc	precisió de la regla
gen	generalització de la regla
f	fitness

La generalització és el nombre d'exemples que la regla cobreix, respecte el nombre d'exemples que la regla "ha vist" (la vida de la regla):

$$gen = \frac{exm}{t_{act} - t_{born}} \tag{5}$$

on t_{act} és la iteració actual.

Aquests càlculs es realitzen de manera acumulativa sobre els exemples que passen en el sistema durant la vida d'una regla. Per tant, estem pressuposant que la població d'exemples és invariant; és a dir, els exemples que es presenten al sistema pertanyen a un determinat concepte el qual no varia en el temps. Si variés, es podria realitzar aquest càlcul usant una finestra de temps i/o ponderant els exemples més recents. No obstant, si el concepte no varia el primer mètode és menys sensible a l'ordre dels exemples i per tant, el seu càlcul pot resultar més fiable. És per això que hem triat el primer mètode esmentat.

El *fitness* f es basa en les mesures de precisió i de generalització per tal de potenciar, a l'igual que en MOLeCS, les regles acurades màximament generals. Aquest càlcul es realitza mitjançant l'aplicació de tècniques d'optimització multiobjectiu, usant un dels esquemes disponibles en MOLeCS [2].

4.2 Actualització

En cada iteració, prové un exemple de l'entorn. A partir d'aquest, es calcula el conjunt de classificadors actius [M] que són tots aquells que satisfan la condició. Entre aquest conjunt, se selecciona el conjunt de classificadors que prediu correctament la classe associada a l'exemple, denotat com conjunt-correcte [C]. L'actualització de la qualitat de les regles, d'acord amb l'exemple actual, segueix l'esquema de la taula 2.

Cada nou exemple provoca l'actualització dels paràmetres exc i ns del conjunt-correcte [C] i d'exm en el conjunt d'activació [M]. Això afecta també als

Taula 2: Avaluació incremental de la qualitat d'un classificador segons l'exemple actual

PER CADA classificador $j \epsilon$ $[M]$
  SI $j \epsilon [C]$
    $exc_j = exc_j + 1$
    $ns_j = ns_j + \beta(|[C]| - ns_j)$
  FISI
  $exm_j = exm_j + 1$
FIPER
PER CADA $j \epsilon$ $[P]$
  actualitzar acc_j *i* gen_j
  actualitzar f_j
FIPER

valors *acc* i *gen* acumulats en cada regla i al valor de *fitness* que s'ha d'actualitzar.

Un tret característic dels sistemes classificadors actuals és l'ús del *covering* [3, 12, 13]. Donat un exemple d'entrada, si no es troba cap classificador en [P] que el satisfaci, se'n crea un de nou amb una condició adequada i una classe associada (en principi, aleatòria). En IMO, hem augmentat l'àmbit d'aplicació del *covering* a les situacions en que [C] és buit. En aquests casos, es crea un nou classificador amb una condició que satisfaci l'exemple, i amb la classe coincident amb l'exemple. L'objectiu d'aquest mecanisme és guiar la cerca del GA cap a l'obtenció de conjunts correctes de classificadors.

4.3 Algorisme Genètic

L'algorisme genètic s'aplica cada θ_{GA} exemples, on θ_{GA} és un paràmetre a escollir per l'usuari. El GA realitza la selecció entre els individus del conjunt-correcte [C], permetent així un creuament restringit. Els individus seleccionats es creuen i muten amb probabilitats p_c i p_m (per gen) respectivament. Els fills resultants es reintrodueixen en la població, seguint el mateix esquema que l'XCS. Si la població és plena, cal esborrar classificadors de [P]. La probabilitat d'esborrat de cada classificador es calcula segons:

$$p_{del} = \begin{cases} ns \cdot \overline{f_{[P]}}/f & \text{si } exp > \theta_{del} \wedge f < \delta \cdot \overline{f_{[P]}} \\ ns & \text{altrament} \end{cases} \quad (6)$$

on *exp* és l'experiència del classificador, o el nombre d'actualitzacions que ha sofert (exm), δ i θ_{del} són dues constants a determinar per l'usuari i $\overline{f_{[P]}}$ és el promig del *fitness* de la població.

4.4 Test

L'IMO pot treballar en dos règims: entrenament i test. En l'entrenament, es realitza *covering* sobre [C], s'actualitzen els paràmetres dels classificadors i el GA es pot activar. En canvi, en el test tots aquests processos es desactiven.

El funcionament en mode test és el següent. Donat un exemple, es calcula el conjunt de classificadors actius [M] i a partir d'aquest, cal predir la classe de sortida. Aquesta predicció es realitza en base a les classes que es proposen en [M], d'acord amb una votació ponderada per *fitness*.

5 Experimentació

5.1 Configuració del sistema

El fet d'incorporar el creuament restringit en el sistema IMO, provoca una pressió implícita cap a la generalització de les regles. Aquest fet queda explicat per la hipòtesi de generalització [13], segons la qual els classificadors generals participen en més conjunts d'activació i com que el GA s'aplica localment en aquests conjunts, aquests tenen més probabilitats de reproduir-se. Per això, la configuració del sistema per aquest article no incorpora la generalització en el *fitness*. Com a treball futur seria interessant aprofundir-hi.

Hem configurat els paràmetres del sistema segons els valors emprats en la bibliografia i l'experimentació prèvia: $f=(acc)^k$, k=10, θ_{GA}=25, β=0.2, p_c=0.8, p_m=0.01, θ_{del}=20, δ=0.1. La mida de la població (el nombre màxim de regles) és $25 \cdot |[O]|$, on $|[O]|$ és el nombre de regles de la solució òptima. El factor de 25 és per permetre la formació d'espècies al voltant de cada solució òptima.

5.2 Experiments amb IMO

El sistema IMO s'ha testejat en un conjunt de problemes artificials de diferent dificultat. En aquest article, mostrem alguns resultats significatius pels problemes del multiplexor, paritat i decodificador. Els resultats en la figura 1 mostren com evoluciona el percentatge d'encerts del sistema al llarg de les iteracions d'aprenentatge. Cada punt de la corba és el promig de les últimes 50 prediccions, realitzades en mode test.

En el problema *mux*6 (multiplexor de 6 entrades), el sistema assoleix el 100% d'encerts en la iteració 2000 aproximadament; és a dir, quan

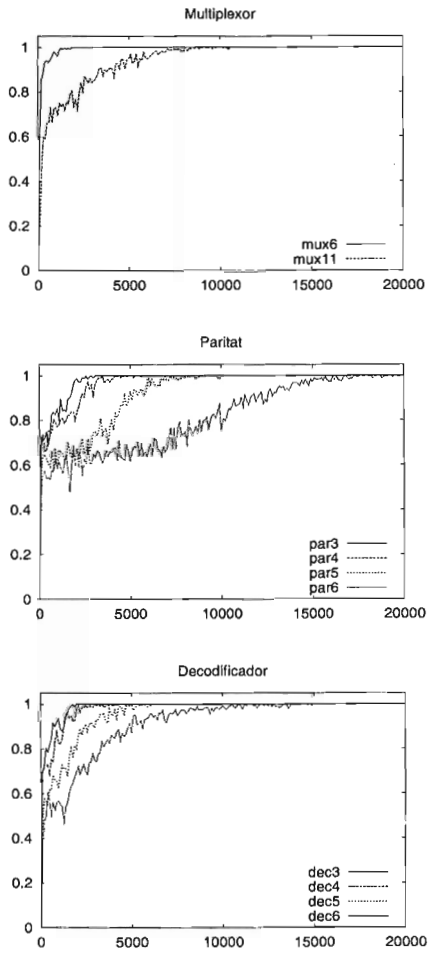

Figura 1. Resultats obtinguts amb IMO en els problemes del multiplexor, paritat i decodificador.

el sistema ha vist 2000 exemples. En el problema $mux11$, l'aprenentatge s'assoleix en la iteració 10000. Pot semblar, a partir d'aquests resultats, que el rendiment del sistema depèn del nombre d'atributs del problema i per tant, de l'espai de cerca. Però de fet, la diferència entre $mux6$ i $mux11$ no és només el nombre d'atributs, sinó la mida de la solució final. El conjunt de regles òptim (denotat per [O]) de $mux6$ conté 8 regles del tipus `01##0#:0`, mentre que en $mux11$ conté 16 regles. Aquest és el factor que contribueix més significativament a l'evolució de l'aprenentatge.

La mida de l'espai solució $|[O]|$ en la paritat és 2^n on n és el nombre d'atributs. L'evolució de l'aprenentatge en funció de $|[O]|$ té un comportament similar al multiplexor. En $par3$ el sistema triga 2500 iteracions, en $par4$ 6000 i així successivament. Per confirmar, doncs, la dependència de l'aprenentatge en la mida de l'espai solució, hem realitzat alguns experiments (que no incloem per motius d'espai) on incrementem el nombre d'entrades en el problema de la paritat, sense afectar la mida de la solució final. Això ho realitzem mitjançant l'addició de bits redundants al final de l'exemple, que el sistema haurà de generalitzar com a `#`. En aquests casos, el nombre d'iteracions necessàries per assolir el 100% d'encerts es manté pràcticament igual que el problema original sense aquests bits de més. Aquest comportament també es va observar en el sistema XCS [14, 10], donat que els dos sistemes usen el mateix esquema de GA.

En el problema de la paritat, les regles han de predir si el nombre d'uns és senar o parell, la qual cosa no permet generalitzar més d'un exemple en una mateixa regla: tots els bits de la condició han d'estar especificats per a que la regla sigui acurada. En aquest sentit, el problema ha permès comprovar el bon rendiment del sistema, malgrat la pressió implícita del GA cap a la generalització.

Per últim, hem realitzat un experiment amb el problema del decodificador, on el nombre de classes és 2^n i la classe associada a un exemple correspon a la decodificació binària del mateix. El problema té té característiques similars a la paritat, pel que fa a la generalització de les regles i a la mida de la solució final. Els resultats obtinguts per a un nombre creixent d'entrades (i creixent en [O]) són similars als de la paritat, però fins i tot una mica millors. Els primers estudis realitzats al respecte apunten cap al motiu següent, que expliquem sota els problemes $par5$ i $dec5$. Suposem que, en $par5$, degut a la pressió per la generalització, s'evoluciona un classificador del tipus `#####:0`. Aquesta evolució és factible, sobretot en les primeres iteracions, ja que tots els classificadors que tenen com a mínim un `#` són igualment inacurats (acc=0.5) i per tant, els més generals es reproduiran més ràpidament. En canvi, en $dec5$ aquest classificador té $acc = 1/32$. La seva evolució és més difícil perque té altres competidors més acurats (del tipus `00###:0`, amb precisió $acc = 1/8$) i fins i tot, altres competidors de la mateixa generalització (exemple `#####:14`) i per tant, es generaran menys còpies que en el cas de la paritat. Quan el sistema comenci a descobrir els classificadors acurats, tendirà a eliminar més ràpidament els classificadors generals en el problema del decodificador que en la paritat, fent que la velocitat d'aprenentatge sigui major.

5.3 Comparació amb MOLeCS

En la figura 2 realitzem una comparació entre el sistema IMO i MOLeCS pel problema $mux11$, respecte el nombre d'exemples que han passat en cada sistema. L'objectiu és verificar la hipòtesi que hem formulat en l'equació (3). La hipòtesi estableix que podem avançar l'aprenentatge si apliquem el GA amb les avaluacions parcials (realitzades de manera incremental), enlloc d'esperar a l'avaluació completa sobre el total d'exemples d'entrenament.

En la gràfica comparem l'evolució del sistema IMO per cada exemple d'entrenament mostrat. L'aprenentatge s'ha assolit en la iteració 10000. Donat que θ_{GA}=25, el nombre de cops que hem aplicat l'algorisme genètic per assolir el 100% d'encerts ha estat aproximadament n_{GA1}=400. Per compararho amb els resultats de MOLeCS respecte el nombre total d'exemples vistos pel sistema, hem multiplicat cada iteració de l'algorisme per $|\mathcal{T}|$=2048. L'aprenentatge en MOLeCS s'ha assolit en 40 iteracions (n_{GA2}=40), cosa que correspon a 81900 exemples. Com que el cost depèn del nombre d'exemples que es presenten el sistema, podem confirmar la hipòtesi establerta: el cost d'aprenentatge del sistema MOLeCS es redueix significativament mitjançant l'avaluació incremental implementada en el sistema IMO.

Pel que fa al nombre de cops que s'aplica el GA, pot semblar que IMO realitza una cerca menys exitosa i per això n_{GA1} és major que n_{GA2}. De fet, això no és cert, perque en cada GA de l'IMO es generen dos nous individus, mentre que el MOLeCS en genera més, en funció del paràmetre G (per G=1, es generen 400 individus per cada GA). Així, en IMO s'han testejat un total de 80 noves solucions gràcies a l'aplicació del GA, mentre que en MOLeCS el nombre ha estat molt superior. Aquest fet demostra que l'aplicació del creuament restringit en IMO millora l'eficiència de les solucions.

5.4 Comparació amb l'XCS

El càlcul diferenciat de la precisió entre el sistema IMO i l'XCS fa que el tipus de solucions que evolucionen aquests dos sistemes siguin diferents. La solució del sistema XCS és un *mapa complet* que conté les regles acurades, des de la perspectiva de la seva predicció. Per tant, s'emmagatzemem les regles que prediuen correctament la recompensa de l'entorn, encara que aquesta recompensa sigui zero. Aquest tipus de regles, que podríem anomenar com

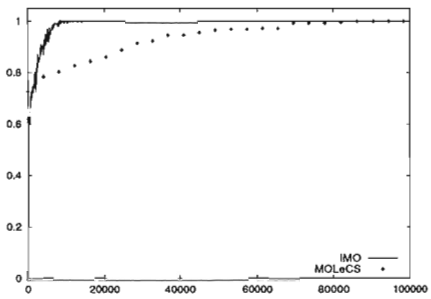

Figura 2. Comparació entre IMO i MOLeCS en el problema del multiplexor **mux11**, respecte al nombre d'exemples presentats a cada sistema.

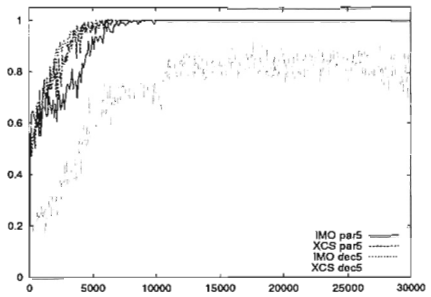

Figura 3. Comparació entre IMO i XCS en la paritat **par5** i decodificador **dec5**.

consistentment incorrectes també formen part de la solució de l'XCS. Per exemple, en el cas del multiplexor una regla d'aquest tipus seria: `00###0:1→0`, on la part dreta de la fletxa indica la predicció de la regla. En canvi, l'IMO només cerca i evoluciona les regles *consistentment correctes*, de manera que el conjunt de solucions final és menor.

En la figura 3 comparem el sistema IMO amb l'XCS pels problemes de la paritat *par*5 i el decodificador *dec*5. En *par*5, l'espai solució de l'IMO $|[O]|_{IMO}$=32 i en XCS $|[O]|_{XCS}$=64. Els dos sistemes assoleixen el 100% d'encerts pràcticament al mateix temps, tot i que l'XCS té una pendent d'aprenentatge més pronunciada.

En canvi, en *dec*5, $|[O]|_{IMO}$=32, mentre que $|[O]|_{XCS}$=192. La diferència entre la mida de les solucions és major i això es fa notar també en la velocitat d'aprenentatge, que en l'XCS és pitjor. Fins i tot, l'XCS té certa dificultat en assolir un encert superior al 80%, la qual cosa indica que el problema és difícil per l'XCS. La causa està en que l'XCS evoluciona un gran nombre de regles *inconsistentment incorrectes* (p.ex., `1####:2` → `0`) que tenen

una generalització superior a les regles *consistentment correctes* (p.ex., 00010:2 → 1000), cosa que dificulta la seva evolució. En canvi, en l'IMO totes les regles consistentment incorrectes són rebutjades i per tant, no interfereixen en la descoberta de les "bones regles".

6 Conclusions i línies de futur

En aquest article hem proposat un sistema classificador inspirat en el sistema MOLeCS, el qual incorpora una avaluació incremental de les regles i un conjunt de millores en el GA. L'avaluació incremental permet que el GA usi les avaluacions parcials de les regles, accelerant així l'aprenentatge, tal com es demostra en els resultats.

Les modificacions introduïdes en el GA estan extretes del sistema XCS i estan destinades a millorar el procés de cerca, mitjançant el creuament restringit a individus de la mateixa espècie i la millora dels mecanismes de *niching*. IMO i XCS comparteixen així el mateix esquema de GA, però no el mateix mecanisme d'avaluació. En concret, IMO basa la precisió de les regles en el percentatge d'encerts (aprenentatge supervisat), mentre que l'XCS basa la precisió en la predicció de la recompensa de l'entorn (aprenentatge per reforçament). Això fa que l'espai solució d'IMO sigui més reduit que XCS, ja que només s'emmagatzemen les regles consistenment correctes i no un mapa complet. I quan aquesta diferència és considerable, el rendiment d'IMO és millor.

Com a línia de futur, seria interessant extendre aquests resultats a altres problemes de més complexitat incorporant, a més, estimadors no esbiaixats del percentatge d'encerts [11]. Així mateix, han quedat obertes altres qüestions com són: la implementació i estudi d'un altre esquema de *niching*, que reemplaci els pous individus en l'espècie actual i l'estudi de l'efecte del GA i el *fitness* multiobjectiu en la generalització de les regles.

Referències

[1] Ester Bernadó and Josep M. Garrell. MultiObjective Learning in a Genetic Classifier System (MOLeCS). In *Butlletí de l'ACIA, 22. 3r Congrés Català d'Intel.ligència Artificial CCIA'2000.*

[2] Ester Bernadó and Josep M. Garrell. MOLeCS: Using Multiobjective Evolutionary Algorithms for Learning. In *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, pages 696–710, 2001.

[3] Pierre Bonelli and Alexandre Parodi. An Efficient Classifier System and its Experimental Comparison with two Representative learning methods on three medical domains. In *Fourth International Conference on Genetic Algorithms (ICGA'91)*, pages 288–295. Morgan Kaufmann, 1991.

[4] Lashon B. Booker. *Intelligent behavior as an adaptation to the task environment.* PhD thesis, University of Michigan, 1982.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.

[6] D.E. Goldberg and P. Segrest. Finite Markov chain analysis of Genetic Algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Lawrence Erlbaum, 1987.

[7] John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[8] Tim Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions. In *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, 1997.

[9] Tim Kovacs. Deletion Schemes for Classifier Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 329–336. Morgan Kauffmann, 1999.

[10] Tim Kovacs and Manfred Kerber. What makes a problem hard for XCS? In *Third International Workshop on Learning Classifier Systems (IWLCS-2000)*, 2000.

[11] J.Kent Martin and D.S. Hirschberg. Small Sample Statistics for Classification Error Rates I : Error Rate Measurements. Technical Report No. 96-21, Department of Information and Computer Science, University of California, Irvine, 1996.

[12] Stewart Wilson. ZCS: A Zeroth Level Classifier System. *Evolutionary Computation*, 2:1–18, 1994.

[13] Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[14] Stewart W. Wilson. Generalization in the XCS Classifier System. In *Genetic Programming: Proceedings of the Third Annual Conference.* San Francisco, CA: Morgan Kaufmann, 1998.

[15] Stewart W. Wilson. State of XCS Classifier System Research. Technical Report No.99.1.1, Prediction Dynamics, Concord MA, 1999.