Les systèmes de classeurs

Un état de l'art

Olivier Sigaud

Université Pierre et Marie Curie - Paris 6
4 Place Jussieu
F-75252 Paris Cedex 05
Olivier.Sigaud@lip6.fr

RÉSUMÉ. *Les systèmes de classeurs sont des systèmes à base de règles de production qui construisent leur ensemble de règles de façon automatique. Initialement, ces systèmes visaient à modéliser l'émergence de capacités cognitives à l'aide de mécanismes adaptatifs, en particulier évolutionnistes. Suite à un renouveau de la problématique mettant davantage l'accent sur l'apprentissage, les systèmes de classeurs ont été vus ensuite comme des outils capables de traiter des problèmes de décision séquentielle de façon compacte, en représentant l'état comme composé d'observables différenciées qu'un agent peut choisir de prendre en compte ou non. Enfin, beaucoup plus récemment, les systèmes de classeurs se sont avérés très efficaces pour résoudre des problèmes de classification automatique, ce qui dynamise le champ de recherche correspondant. Dans ce contexte, l'objet de cette contribution est de présenter l'état de la recherche sur les systèmes de classeurs en insistant sur les développements les plus récents, en insistant sur la décision séquentielle plutôt que sur la classification automatique.*

ABSTRACT. *Learning Classifier Systems (LCSs) are rule-based systems that automatically build their ruleset. Initially, LCSs were dedicated to the modelling of the emergence of cognitive abilities thanks to adaptive mechanisms, particularly evolutionary processes. After a renewal of the field more focused on learning, LCSs have been reconsidered as sequential decision problem solving systems endowed with a generalization property. Finally, much more recently, LCSs have proved very efficient at solving classification tasks, which boosted the field. In this context, the aim of this contribution is to present the state-of-the-art of LCSs, insisting on recent developments, and focusing more on the sequential decision domain than on automatic classification.*

MOTS-CLÉS : *systèmes de classeurs, apprentissage par renforcement, généralisation.*

KEYWORDS: *Learning classifier systems, reinforcement learning, generalization.*

RSTI - RIA – 21/2007. Prise de décision séquentielle, pages 75 à 106

1. Introduction

Tous les systèmes de classeurs [1] ont en commun d'être des systèmes à base de règles de production qui construisent automatiquement l'ensemble de règles sur lequel ils travaillent.

Ces systèmes inventés par John Holland (Holland, 1976) ont fait l'objet d'une gestation commune avec les algorithmes génétiques (AG) (Holland, 1975). L'impact considérable des AG a valu aux recherches sur les systèmes de classeurs de longtemps rester dans l'ombre. Toutefois, ces recherches connaissent depuis quelques années un essor important, qui justifie la publication d'une présentation générale destinée à un public élargi. L'objectif de cet article est donc d'offrir à la communauté francophone susceptible de s'y intéresser, elle-même en plein essor, un panorama didactique qui présente à la fois les aspects fondamentaux des mécanismes des systèmes de classeurs et les développements les plus récents auxquels ils donnent lieu.

Afin d'atteindre cet objectif, nous commençons par présenter les deux mécanismes sur lesquels ils s'appuient, à savoir les algorithmes génétiques et l'apprentissage par renforcement. Nous brossons ensuite un rapide historique de la recherche sur les systèmes de classeurs destiné à faire apparaître trois familles principales de systèmes, à savoir les systèmes basés sur la force, sur la précision et sur l'anticipation. Nous présentons alors à la section 4 tout ce qu'il y a de commun aux différents systèmes de classeurs issus de ces trois familles, de leur formalisme de représentation à leurs mécanismes fondamentaux. Puis, dans trois sections successives, nous examinons plus en détail les aspects saillants des trois familles, en nous efforçant de couvrir les développements les plus récents auxquelles elles donnent lieu sur les plans théoriques et applicatifs. Nous consacrons un effort plus important à la lignée issue de XCS, qui est le système sur lequel portent le plus de travaux à l'heure actuelle. Nous nous efforçons enfin de faire apparaître les directions de recherche qui nous semblent les plus prometteuses et nous concluons sur les ressources à consulter pour approfondir le sujet.

2. Arrière-plan

2.1. *Algorithmes génétiques*

Nous commençons par présenter brièvement les algorithmes génétiques (Holland, 1975; Booker *et al.*, 1989; Goldberg, 1989), qui s'inspirent très librement de la théorie sur la sélection naturelle de Darwin. Ces algorithmes travaillent sur une population d'individus qui représente un ensemble de solutions possibles à un problème donné.

1. En anglais, *Learning Classifier Systems*, ou *LCS*. Le terme *Learning* a été ajouté récemment pour distinguer cette communauté de la communauté scientifiquement distincte qui travaille sur les *Multiple Classifier Systems* en mélangeant différentes techniques de classification. A noter aussi qu'on trouve souvent dans les textes français l'appellation impropre « systèmes de classifieurs ».

Les AG reposent généralement sur quatre analogies avec la biologie : ils utilisent un code, le *génotype*, des transformations simples opérant sur ce code, les *opérateurs génétiques*, la construction d'une solution à partir d'un code, le *passage du génotype au phénotype*, et un processus de sélection des solutions, la *survie du plus apte*. Le génotype, ou génome, permet de représenter ou de construire un individu, c'est-à-dire une solution potentielle au problème posé. Les opérateurs génétiques permettent d'introduire des variations au sein de ces génomes. Ces opérateurs se divisent en deux grandes catégories : les croisements, qui créent de nouveaux génomes en recombinant des sous-parties du code génétique de deux ou plusieurs individus, et les mutations, qui modifient aléatoirement le génome d'un individu. Le processus de sélection permet d'identifier les génomes méritant d'être reproduits, sur lesquels on appliquera les opérateurs génétiques.

Un AG manipule un ensemble de génomes qui sont initialisés arbitrairement et sont ensuite sélectionnés et modifiés progressivement, génération après génération. Ceux qui ne sont pas sélectionnés sont éliminés. Pour un AG, une fonction d'utilité, ou *fitness*, évalue l'intérêt d'un phénotype au regard du problème posé. C'est de cette évaluation que dépendra ensuite la survie de cette solution ou son nombre de descendants dans la population de la génération suivante. Les descendants d'un individu sont construits à partir de copies du génome parent auxquelles on applique les opérateurs génétiques. Ainsi, le processus consiste en une boucle répétée à chaque génération pour déterminer la nouvelle population :

1) on sélectionne n_e génomes parents selon la *fitness* des phénotypes correspondants,

2) on applique les opérateurs de croisement sur les génomes sélectionnés pour générer de nouveaux génomes,

3) on applique les opérateurs de mutation sur les nouveaux génomes de manière à les modifier aléatoirement,

4) on évalue chacun des phénotypes engendrés à partir de ces nouveaux génomes,

5) on retourne en 1.

Sous réserve d'un certain nombre de conditions empiriques que nous ne détaillerons pas, un tel processus conduit à une augmentation progressive de la *fitness* des individus.

La recherche sur les algorithmes génétiques est devenu un champ de recherches à part entière, que nous n'allons pas passer en revue ici. Bien que lui ayant donné naissance, les systèmes de classeurs n'ont fait jusqu'à présent qu'un usage très limité des développements issus de ce champ de recherche. En conséquence, pour une introduction aux systèmes de classeurs, il est simplement nécessaire de préciser les points suivants.

– On distingue classiquement les opérateurs de croisement dits « à un point » [2], qui coupent les deux génomes des parents en un point choisi au hasard pour recomposer de nouveaux génomes en intervertissant des parties, et les opérateurs de croisement dits « à plusieurs points », qui font de même en découpant les génomes des parents en plus de deux tronçons. Dans la plupart des systèmes de classeurs, on se contente d'utiliser des croisements à un point.

– On distingue aussi les algorithmes génétiques « générationnels », dans lesquels une partie importante de la population est renouvelée intégralement au passage d'une génération à la suivante, et les algorithmes génétiques « *steady state* », dans lesquels les individus sont renouvelés un à un au sein de la population. Dans le cadre des systèmes de classeurs, on utilise généralement un renouvellement de type *steady state*.

2.2. *Processus décisionnel de Markov et apprentissage par renforcement*

L'autre mécanisme fondamental sur lequel reposent les systèmes de classeurs est l'apprentissage par renforcement. Il est nécessaire pour présenter ce mécanisme de décrire brièvement le formalisme des processus décisionnel de Markov et l'algorithme Q-LEARNING, qui est l'algorithme d'apprentissage le plus utilisé dans le cadre des systèmes de classeurs. Cette présentation étant conçue pour être la plus succincte possible, nous renvoyons le lecteur désireux d'approfondir ces questions à (Sigaud, 2004).

2.2.1. *Processus décisionnel de Markov*

Un processus décisionnel de Markov se décrit par la donnée des éléments suivants :

– un ensemble fini S d'états discrets notés s ;

– un ensemble fini A d'actions discrètes notées a ;

– une fonction de transition $P : S \times A \rightarrow \Pi(S)$ où $\Pi(S)$ est l'ensemble des distributions de probabilité sur S. Une distribution de probabilité particulière $Pr(s_{t+1}|s_t, a_t)$ indique les probabilités pour que l'agent se retrouve dans les différents états s_{t+1} possibles quand il fait l'action a_t dans l'état s_t ;

– une fonction de renforcement $R : S \times A \rightarrow \mathbb{R}$ qui indique pour tout couple (s_t, a_t), sous forme d'un nombre réel, le signal de renforcement (punition ou récompense) que reçoit l'agent quand il fait l'action a_t dans l'état s_t.

Le formalisme des processus décisionnels de Markov permet de décrire la structure d'un ensemble de problèmes auxquels est confronté un agent et ne préjuge en rien du choix des actions de l'agent au sein de son environnement. Il permet uniquement de déterminer quelle sera la situation future de l'agent selon qu'il effectue telle ou telle action dans tel ou tel état et quelles récompenses il recevra en fonction de son comportement. Les processus décisionnels de Markov définissent donc un cadre général pour représenter un problème auquel est susceptible d'être confronté un agent réalisant des actions discrètes dans un environnement discret.

2. *One-point cross-over.*

Pour que le problème représenté soit bien un processus décisionnel de Markov, il faut que la fonction de transition vérifie une hypothèse connue sous le nom d'hypothèse de Markov, qui stipule que la distribution de probabilité spécifiant l'état s_{t+1} dépend uniquement de s_t et de l'action a_t, mais pas du passé. On a donc $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$. Cela revient à dire que, si l'hypothèse de Markov est vérifiée, une connaissance des actions réalisées précédemment ou des états parcourus dans le passé ne permet pas de déterminer plus précisément avec quelle probabilité l'agent se trouvera dans chaque état possible à l'issue de son action.

Le comportement de l'agent est décrit par une politique π qui indique pour chaque état s selon quelle distribution de probabilité l'agent doit choisir les différentes actions possibles dans cet état.

Lorsque la fonction de transition et la fonction de récompense sont connues a priori, les méthodes de programmation dynamique telles que *policy iteration* (Bellman, 1961; Puterman *et al.*, 1978) et *value iteration* (Bellman, 1957) permettent de trouver efficacement comment doit se comporter l'agent pour maximiser la récompense cumulée qu'il reçoit sur l'ensemble de son comportement.

Pour définir la récompense cumulée, on introduit un facteur multiplicatif γ, appelé « facteur d'actualisation [3] » et généralement compris entre 0 et 1. Le facteur γ mesure à quel point les récompenses ultérieures sont prises en compte dans le calcul de la récompense cumulée. On définit alors la récompense cumulée dite « γ-pondérée » à l'instant t :

$$Rc_\pi(t) = \sum_{k=t}^{T_{max}} \gamma^{(k-t)} r_\pi(k)$$

où T_{max} peut être fini ou infini et $r_\pi(k)$ désigne la récompense immédiate reçue à l'instant k si l'on suit la politique π.

Les méthodes de programmation dynamique consistent à introduire une fonction de valeur V^π où $V^\pi(s)$ représente la récompense globale espérée par l'agent s'il suit la politique π à partir de l'état s. La fonction de valeur $V^\pi(s_t)$ associe donc à chaque état s_t une mesure de la récompense cumulée qu'un agent recevra s'il suit la politique π à partir de l'état s_t.

On montre que, si l'hypothèse de Markov est vérifiée, la fonction V^π est solution de l'équation de Bellman (Bertsekas, 1995) :

$$\forall s \in S, V^\pi(s) = \sum_a \pi(s_t, a_t)[R(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V^\pi(s_{t+1})] \quad [1]$$

3. *Discount factor.*

Plutôt que la fonction V^π, il est souvent plus pratique d'introduire une fonction Q^π où $Q^\pi(s,a)$ mesure la récompense globale espérée par l'agent s'il suit la politique π après avoir fait l'action a dans l'état s. Tout ce qui a été dit de V^π se transpose directement à Q^π, sachant que $V^\pi(s) = \max_a Q^\pi(s,a)$.

Les fonctions optimales correspondantes sont indépendantes de la politique de l'agent, elles sont notées V^* et Q^*.

2.2.2. *Apprentissage par renforcement*

La nécessité de l'apprentissage se fait jour lorsque les fonctions de transition et de récompense ne sont pas connues a priori. La seule solution pour l'agent consiste alors à explorer les différentes actions possibles dans les différents états de son environnement à la recherche des couples (s_t, a_t) qui produisent une récompense élevée.

Les principales méthodes d'apprentissage par renforcement consistent à essayer d'estimer V^* ou Q^* de façon itérative au fur et à mesure que l'agent fait des essais et des erreurs dans son environnement. Toutes ces méthodes reposent sur une technique générale pour estimer la moyenne d'un signal stochastique reçu à chaque pas de temps, sans stocker aucune information relative aux signaux reçus dans le passé. Prenons le cas du calcul de la récompense immédiate moyenne. Sa formulation exacte sur k itérations est

$$E_k(s) = (r_1 + r_2 + \cdots + r_k)/k$$

Par ailleurs,

$$E_{k+1}(s) = (r_1 + r_2 + \cdots + r_k + r_{k+1})/(k+1)$$

donc

$$E_{k+1}(s) = k/(k+1)E_k(s) + r_{k+1}/(k+1)$$

On peut reformuler :

$$E_{k+1}(s) = (k+1)/(k+1)E_k(s) - E_k(s)/(k+1) + r_{k+1}/(k+1)$$

ou encore

$$E_{k+1}(s) = E_k(s) + 1/(k+1)[r_{k+1} - E_k(s)]$$

Formulée ainsi, on a accès à la moyenne exacte en ne stockant plus que k. Si on ne veut pas stocker k, on peut approximer $1/(k+1)$ par α, ce qui donne l'équation [2] dont on retrouve la forme générale partout en apprentissage par renforcement :

$$E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)] \qquad [2]$$

Le paramètre α est appelé « taux d'apprentissage » [4], il doit être réglé de façon adéquate car il influence la vitesse de convergence vers la moyenne exacte.

4. *Learning rate.*

Nous ne présenterons pas l'ensemble des méthodes d'apprentissage qui reposent sur ce principe d'estimation, nous ne donnerons que l'équation de mise à jour de l'algorithme Q-LEARNING, qui est la suivante :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad [3]$$

3. Historique succinct

3.1. *Approche Pittsburg contre Michigan*

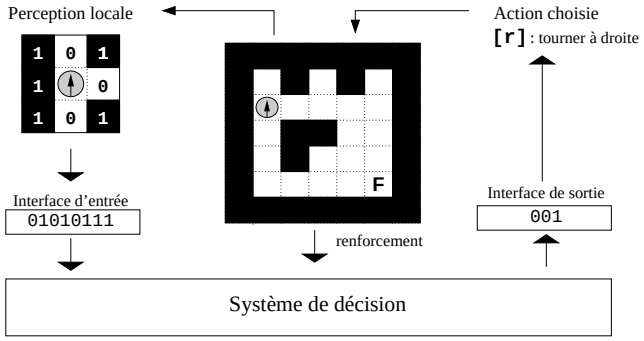

Figure 1. *Représentation d'un problème d'interaction. L'agent perçoit une situation sous la forme d'un ensemble d'attributs. Dans cet exemple, l'agent est situé dans un labyrinthe et perçoit la présence (symbole 1) ou l'absence (symbole 0) de blocs dans les huit cases environnantes. Les cases sont considérées dans le sens des aiguilles d'une montre, en commençant par le nord, si bien qu'il perçoit* `[01010111]` *dans l'exemple ci-dessus. Cette information est envoyée à son interface d'entrée. L'agent doit choisir à chaque instant entre avancer* `[f]` *tourner à droite* `[r]` *ou tourner à gauche* `[l]`*. Il indique l'action qu'il a choisie au travers de son interface de sortie*

Les systèmes de classeurs ont été inventés par (Holland, 1975) en vue de proposer un modèle de l'émergence de la cognition fondé sur des mécanismes adaptatifs. Il s'agit d'un ensemble de règles de production appelées « classeurs » combinées à des mécanismes adaptatifs chargés de faire évoluer ces règles. L'objectif était de résoudre des problème d'interaction avec un environnement de type de ceux présentés sur la figure 1, ce qui a été décrit plus tard par Wilson comme « le problème de l'Animat » (Wilson, 1985).

Lors de ces premières recherches sur les systèmes de classeurs, l'accent était surtout mis sur le parallélisme de l'architecture et les mécanismes d'évolution qui lui permettent de s'adapter en permanence aux variations de l'environnement (Golberg *et al.*, 1988). Cette approche était vue comme un moyen d'échapper à la fragilité des systèmes d'intelligence artificielle traditionnelle (Holland, 1986) face aux contextes plus

complexes que les problèmes jouets ou les domaines d'applications totalement maîtrisés. Cette période du développement des systèmes de classeurs était surtout marquée par l'opposition entre les approches de type Pittsburgh et les approches de type Michigan.

Dans l'approche de type défendue par (Smith, 1980), de l'université de Pittsburgh, le seul mécanisme adaptatif mis en œuvre était un AG qui s'appliquait à une population de systèmes de classeurs pour choisir parmi cette population le système de classeurs le plus adapté à un problème donné.

Dans les systèmes proposés par Holland et ses doctorants successifs, constituant l'école Michigan, au contraire, l'AG était combiné d'emblée avec un mécanisme d'apprentissage par renforcement et s'appliquait de façon plus subtile à l'intérieur d'un unique système de classeurs.

Même si l'on assiste à l'heure actuelle à un renouveau des approches de type Pittsburgh (Llorà *et al.*, 2002; Bacardit *et al.*, 2003; Landau *et al.*, 2005), l'approche de type Michigan s'est rapidement imposée comme le cadre standard des systèmes de classeurs, sa concurrente s'étant dissoute dans la problématique plus vaste des algorithmes évolutionnistes.

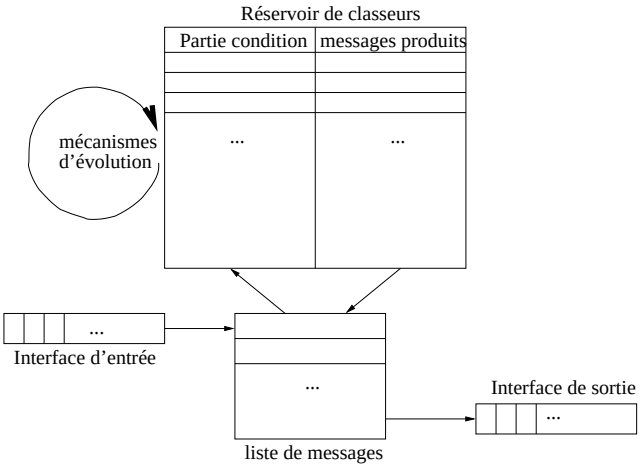

Figure 2. *Architecture du système CS1 de Holland. L'interface d'entrée produit des messages qui sont ajoutés à la liste des messages internes. Un réservoir de règles de production en déduit de nouveaux messages en déclenchant les règles dont la partie [Condition] est vérifiée par les messages originaux. Certains messages sont envoyés à l'interface de sortie et produisent des actions*

La première implémentation concrète d'un système de classeurs de type Michigan, appelée « CS1 », a été publiée par Holland et Reitman (Holland *et al.*, 1978). Ce premier modèle assez complexe est illustré sur la figure 2. L'AG se charge de créer de

nouvelles règles de production à partir des règles existantes. La *fitness* d'un classeur est mesurée par sa capacité à proposer une action efficace au moment opportun. Cette efficacité est elle-même évaluée par un algorithme d'apprentissage par renforcement nommé BUCKET BRIGADE (Holland *et al.*, 1986).

La présence d'une liste de messages internes permet de rendre compte de phénomène de mémoire dès lors que certains messages peuvent rester dans la liste de messages durant plusieurs pas de temps. Mais de nombreux mécanismes se sont avérés difficiles à mettre au point, notamment le mécanisme de suppression des messages et les interactions entre l'évaluation des classeurs et les mécanismes génétiques chargés de faire évoluer leur population (Wilson *et al.*, 1989). En raison de la difficulté à obtenir des résultats avec diverses versions de ces architectures, la recherche sur les systèmes de classeurs a connu un certain essoufflement dans les années 1980. Un historique de cette première phase du développement des systèmes de classeurs a été publié par Wilson et Goldberg en 1989 (Wilson *et al.*, 1989).

3.2. *Le renouveau*

Le premier tournant qui s'est produit dans l'histoire des systèmes de classeurs est lié à l'histoire parallèle du développement des recherches en apprentissage par renforcement, notamment avec la publication de l'algorithme Q-LEARNING (Watkins, 1989).

Les algorithmes désormais classiques d'apprentissage par renforcement tels que Q-LEARNING reposent sur une énumération explicite de tous les états possibles qu'un agent est susceptible de rencontrer dans son environnement. Or, parce qu'ils représentent la situation d'un agent sous la forme d'un ensemble de perceptions différenciées appelées « attributs », les systèmes de classeurs permettent d'éviter une énumération exhaustive de l'ensemble des états grâce à un mécanisme de généralisation [5] que nous allons détailler dans la suite.

Cette propriété de *généralisation* est apparue peu à peu comme la spécificité centrale des systèmes de classeurs vis-à-vis du formalisme standard de l'apprentissage par renforcement dans lequel l'état est considéré comme une entité indifférenciée, ce qui a conduit Lanzi à définir les systèmes de classeurs comme des systèmes d'apprentissage par renforcement dotés de capacités de généralisation (Lanzi, 2002).

5. Il faut noter que l'emploi du terme « généralisation » dans la communauté de recherche sur les systèmes de classeurs prête à confusion. En effet, en apprentissage artificiel, on appelle classiquement « généralisation » la capacité d'un système à produire la bonne réponse sur des exemples qu'il n'a pas encore observés, à partir de ce qu'il a appris sur d'autres exemples. Les systèmes de classeurs sont bien dotés de cette capacité, mais celle-ci résulte d'un mécanisme plus spécifique de « factorisation » (voir la section 8) de la représentation d'état qui est confondu avec la propriété de généralisation qui en résulte. Malgré cette remarque, nous nous conformerons à l'usage de la communauté pour faciliter l'assimilation de la littérature du domaine.

Une étape importante du renouvellement de cette problématique de recherche est la mise en évidence par Dorigo et Bersini des similarités entre l'algorithme BUCKET BRIGADE (Holland, 1986) utilisé jusque là dans les systèmes de classeurs et l'algorithme Q-LEARNING (Dorigo *et al.*, 1994). Au même moment, Wilson proposait une version radicalement simplifiée des architectures initiales, dans laquelle les messages internes avaient disparu, en publiant le système « ZCS » [6] (Wilson, 1994).

ZCS assimile la force d'un classeur à l'espérance de renforcement reçu en déclenchant le classeur concerné, donnant naissance à la lignée des systèmes « basés sur la force » [7]. La *fitness* des classeurs est donc directement égale à la qualité du classeur, c'est-à-dire à l'estimation de récompense attendue calculée par l'apprentissage par renforcement. Ainsi, l'AG élimine les classeurs qui conduisent l'agent à atteindre moins de récompenses que les autres.

Après ZCS, Wilson a imaginé un nouveau système plus subtil, baptisé « XCS » (Wilson, 1995), dans lequel la force représente cette fois la capacité du classeur à *prédire* précisément la quantité de renforcement que son déclenchement permet de recevoir. Les classeurs déclenchés sont choisis eux aussi en fonction de l'estimation de récompense attendue, mais l'AG évalue les classeurs en fonction de leur capacité à prédire exactement la récompense qu'ils vont recevoir. XCS s'est avéré très performant et constitue le point de départ d'une nouvelle lignée de systèmes « basés sur la précision » [8].

Enfin, quelques années plus tard, Stolzmann proposait un système de classeurs à anticipation nommé ACS (Stolzmann, 1998; Butz *et al.*, 2000), créant ainsi la lignée des systèmes de classeurs « basés sur l'anticipation » [9]. Comme nous allons le voir, cette troisième lignée se distingue sensiblement des deux autres. Sur le plan historique, elle provient de recherches en psychologie qui relèvent de l'apprentissage latent (Tolman, 1932; Seward, 1949). Plus précisément, Stolzmann est un étudiant de Hoffmann (Hoffmann, 1993) qui a bâti une théorie psychologique de l'apprentissage appelée « théorie du contrôle anticipatif du comportement » [10] en s'inspirant des travaux de (Herbart, 1825). Les sources historiques de cette troisième lignée ont été décrites ailleurs (Sigaud, 2004).

Le développement de ces trois lignées constitue le cœur de l'époque moderne de la recherche sur les systèmes de classeurs. Pour terminer cet historique, il faut indiquer qu'à l'issue d'une seconde synthèse de la recherche sur les systèmes de classeurs réalisée par Lanzi et Riolo en 1999 (Lanzi *et al.*, 2000a), un second tournant s'est produit. Bien que, comme nous l'avons montré, la nouvelle impulsion donnée à la recherche sur les systèmes de classeurs ait porté sur les problèmes de décision séquentielle représentant l'interaction entre un agent et son environnement, les excellents résultats du système XCS sur des problèmes de fouille de données (Bernadó *et al.*, 2001)

6. *Zeroth-level Classifier System.*
7. *Strengh-based Learning Classifier Systems.*
8. *Accuracy-based Learning Classifier Systems.*
9. *Anticipation-based Learning Classifier Systems* ou *Anticipatory Learning Classifier Systems.*
10. *Anticipatory Behavioral Control.*

conduisent actuellement à une extension spectaculaire des travaux en direction des problèmes de classification automatique, direction amorcée par (Booker, 2000) et (Holmes, 2002).

4. Éléments fondamentaux

Pour présenter les aspects les plus fondamentaux des systèmes de classeurs, nous allons adopter le point de vue proposé par (Lanzi, 2002), qui consiste à les définir comme des systèmes d'apprentissage par renforcement dotés de capacités de généralisation.

Une façon simple de réaliser concrètement l'algorithme Q-LEARNING consiste à construire au fur et à mesure de l'expérience de l'agent une table appelée *Q-table* contenant des triplets $< s, a, p >$, où s et a représentent l'état et l'action courants de l'agent, et p son estimation courante de la récompense espérée associée à ce couple (s, a). L'idée est d'ajouter une ligne dans la table à chaque fois que l'on essaye une nouvelle action, que ce soit dans un état connu ou nouveau, et de mettre à jour à chaque pas de temps la valeur de p conformément à l'équation (3) dans la ligne correspondante au couple (s, a) courant.

Le problème avec une telle représentation qualifiée de « tabulaire » est que le nombre d'états peut être très élevé, si bien que la table peut devenir trop lourde en mémoire. Comme l'indique très clairement (Lanzi, 2002), pour résoudre ce problème, il faut choisir une autre représentation dotée d'une capacité de *généralisation*. En pratique, cela peut être réalisé en construisant une table contenant des triplets $< c, a, p >$, dans laquelle les éléments c ne désignent plus un état unique, mais un ensemble d'états qui ont tous en commun de vérifier la condition c.

Les systèmes de classeurs se caractérisent par leur façon spécifique de représenter les conditions c, et par les processus qu'ils mettent en œuvre pour trouver des conditions qui permettent d'obtenir des représentations à la fois compactes et efficaces. Ce sont ces aspects que nous allons détailler à présent.

4.1. *Formalisme de représentation des classeurs*

Un système de classeurs est composé d'une population de règles appelées « *classeurs* ». Chaque classeur représente un triplet $< c, a, p >$, donc il contient au moins une partie [Condition], une partie [Action], et une estimation de la récompense espérée si l'on déclenche ce classeur.

Formellement, la partie [Condition] des classeurs est constituée d'un vecteur de tests. Il y a autant de tests que d'attributs, chaque test portant sur un attribut spécifique. Dans le cas particulier, le plus répandu, où le test porte sur une valeur que doit prendre l'attribut pour que la condition soit vérifiée, le test fait apparaître uniquement la valeur spécifiée, d'où une confusion classique entre test et attribut. Il existe un test particulier,

noté « # » et appelé « *don't care symbol* », qui signifie que la condition stipulée par le classeur sera vérifiée quelle que soit la valeur de l'attribut correspondant. Plus globalement, la partie [Condition] d'un classeur est vérifiée si chacun des tests qu'elle spécifie est vérifié par la situation courante. Dans ce cas, le classeur peut être activé.

Historiquement, les premiers systèmes de classeurs fonctionnaient uniquement avec des vecteurs de tests booléens. Chaque test de la partie [Condition] prend alors sa valeur dans l'ensemble $\{0, 1, \#\}$. Plus récemment, de nombreux systèmes ont été étendus pour prendre en compte des valeurs discrètes [11] ou continues [12]. Dans ce second cas, un test spécifie généralement un intervalle dans lequel doit se situer la valeur de l'attribut pour que la condition soit vérifiée. Il existe aussi des systèmes de classeurs dans lesquels chaque condition individuelle est exprimée sous la forme d'une expression symbolique [13] que doit vérifier l'attribut pour que la condition soit vérifiée (Ahluwalia *et al.*, 1999; Lanzi *et al.*, 1999).

Le symbole # est le moyen mis en œuvre par les systèmes de classeurs pour assurer la propriété de généralisation. Grâce à la présence de symboles # dans la partie [Condition] des classeurs, deux situations sont considérées comme équivalentes lorsque seules les attributs auxquels le classeur est sensible ont les mêmes valeurs dans ces situations. Pour le dire autrement, un classeur qui dispose d'un # pour l'un des tests de sa partie [Condition] subsume tous les classeurs qui prennent une valeur spécifique donnée pour ce même test et qui spécifient les mêmes valeurs pour les autres attributs et la même action à accomplir.

Une fois posée la représentation qu'ils manipulent, reste à définir les mécanismes de fonctionnement des systèmes de classeurs. L'objectif est de mettre au point un système qui apprend par renforcement, donc on trouve au cœur du système un mécanisme de *sélection de l'action* qui s'appuie sur l'apprentissage de la fonction valeur des différentes actions. Par ailleurs, les systèmes sont dotés d'une capacité de généralisation qui repose sur des mécanismes d'*évolution de la population de classeurs* pour atteindre le bon degré de généralité. Nous présentons immédiatement ces deux catégories de mécanismes, puis nous verrons lorsque nous énumérerons les différents systèmes que c'est principalement la façon de gérer l'interaction entre ces deux mécanismes qui distingue les systèmes les uns des autres.

4.2. *Sélection de l'action*

Compte tenu de la capacité de généralisation des classeurs, il est fréquent que plusieurs classeurs différents voient leur partie [Condition] vérifiée dans la même situation. On appelle « ensemble des classeurs activables », noté [M] [14], l'ensemble des classeurs susceptibles d'être activés dans une situation donnée. Il est aussi fré-

11. Par exemple, MACS (Gérard *et al.*, 2005).
12. Par exemple, XCSF (Wilson, 2001).
13. *S-expression.*
14. Pour *Match-set*.

quent que les différents classeurs au sein de [M] stipulent des actions différentes. Il faut donc adjoindre à un système de classeurs un mécanisme d'arbitrage qui décide de l'action qui sera effectivement accomplie par l'agent en fonction des classeurs présents dans [M]. Ce mécanisme d'arbitrage est notamment responsable de la gestion du compromis entre exploration et exploitation, crucial pour l'apprentissage par renforcement. Nous verrons les différents mécanismes possibles de *choix de l'action* en précisant le fonctionnement des systèmes de classeurs existants. Tous ces mécanismes reposent sur l'apprentissage de la valeur de la récompense espérée associée au déclenchement de chacun des classeurs. Nous passerons en revue dans la suite les différents mécanismes d'*évaluation des classeurs* responsables du calcul de cette valeur.

4.3. *Evolution de la population*

La préoccupation consistant à associer le bon degré de généralité à chaque classeur est cruciale pour la mise au point d'un système de classeurs. Il s'agit de rechercher l'ensemble de classeurs couvrant tous les états possibles de la façon la plus compacte possible, compte tenu de l'objectif général qui est de produire un agent dont le comportement converge vers un optimum. Les mécanismes qui permettent d'assurer cette propriété varient d'un système à l'autre, mais ils se traduisent tous par des ajouts et des suppressions de classeurs.

Dans les systèmes basés sur la force et sur la prédiction, la spécialisation et la généralisation des classeurs existants sont entièrement réalisées par un AG qui fait évoluer les parties [Condition] des classeurs dans l'espoir de générer des classeurs à la fois plus efficaces et plus généraux. On fait donc confiance à une heuristique de type essai-et-erreur pour converger vers le bon degré de généralité. L'AG fonctionne sur une population de classeurs de taille limitée et induit entre ces classeurs une compétition. Dans le cas des systèmes basés sur l'anticipation, enfin, il existe des heuristiques plus spécifiques pour assurer la spécialisation et la généralisation des classeurs, même si un algorithme génétique assure la généralisation dans ACS2. Nous passerons donc en revue les différents mécanismes de *création et suppression de classeurs* des différents systèmes.

Nous allons à présent passer en revue les trois grandes familles de systèmes de classeurs que nous avons dégagées.

5. Les systèmes basés sur la force : ZCS

5.1. *Présentation*

Les systèmes de classeurs basés sur la force sont les systèmes de classeurs les plus simples qui soient. Chaque classeur ne dispose que d'une évaluation, qui lui tient lieu à la fois d'évaluation de la récompense cumulée que l'agent peut obtenir en le déclenchant et de *fitness* pour le mécanisme de création et de suppression de classeurs. Le

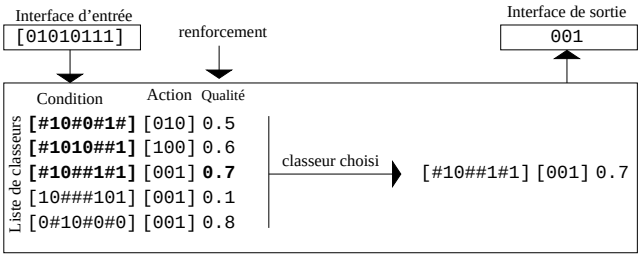

Figure 3. *Dans un système de classeurs de type ZCS, les classeurs sont composés d'une partie [Condition], une partie [Action], et une valeur qui reflète à la fois la* fitness *du classeur et la récompense cumulée que l'agent peut espérer recevoir sur le long terme s'il le déclenche. Parmi les classeurs appris, le système identifie ceux dont les conditions sont compatibles avec la situation courante, puis il choisit un classeur dans cet ensemble avec un mécanisme de roue de la fortune. L'action spécifiée par ce classeur est activée si bien que, dans l'exemple, l'agent exécute l'action* $[001]$

système le plus typique de cette catégorie est ZCS. Ce système fonctionne avec une population de classeurs de taille fixée P. Le principe de fonctionnement de l'interaction d'un agent commandé par ZCS avec son environnement est décrit sur la figure 3.

5.2. *Choix de l'action*

Une fois déterminé l'ensemble [M] des classeurs activables, le choix de l'action activée s'effectue dans ZCS selon un mécanisme de « roue de la fortune » [15] (Goldberg, 1989), qui induit une forme d'exploration. En effet, au lieu de choisir systématiquement l'action considérée comme la meilleure d'après les valeurs, on affecte à chaque classeur de [M] une probabilité d'être déclenchée égale au rapport entre sa valeur et la somme des valeurs de tous les classeurs de [M], puis c'est un tirage aléatoire qui décide de l'action effectivement choisie.

5.3. *Évaluation des classeurs*

Le mécanisme de propagation des récompenses au sein des classeurs mis en œuvre par ZCS est proche de l'algorithme *Bucket Brigade* proposé originellement par (Holland, 1986), mais s'en distingue notablement par l'absence de mécanisme d'enchère.

15. *Roulette wheel.*

Plus précisément, si on note [A] [16] l'ensemble des classeurs de [M] qui proposent l'action a_t choisie effectivement, dans un premier temps, tous les classeurs qui ont proposé l'action a_{t-1} choisie lors du pas de temps précédent se partagent équitablement une fraction $\gamma.\alpha$ de la somme des valeurs des classeurs de [A]. Dans un deuxième temps, tous les classeurs de [A] se partagent équitablement une fraction α de la récompense r_t reçue pour l'exécution de a_t. Enfin, tous les classeurs de [M] qui ne font pas partie de [A] voient leur valeur réduite d'une taxe τ. Ce mécanisme est plus proche de SARSA (Sutton, 1996) que de Q-LEARNING.

5.4. *Création et suppression de classeurs*

L'évolution de la population de classeurs est commandée dans ZCS par un AG et un opérateur de couverture [17].

– À chaque pas de temps, il y a une probabilité p d'invocation de l'AG. S'il est invoqué, l'AG utilise un mécanisme de roue de la fortune basé sur la *fitness* pour choisir deux classeurs dans l'ensemble de la population. Deux descendants de ces classeurs sont engendrés à l'aide d'un opérateur de croisement à un point et d'un opérateur de mutation. Ces descendants reçoivent la moitié de la *fitness* de chacun de leurs parents. Ils remplacent au sein de la population deux autres classeurs choisis par un mécanisme de roue de la fortune basé sur l'inverse de la *fitness*.

– L'opérateur de couverture est invoqué dans chaque situation où l'ensemble [M] est vide ou bien contient des classeurs dont la *fitness* moyenne est Φ fois plus faible que la *fitness* moyenne de la population globale. Cet opérateur intègre dans la population un nouveau classeur activable dans la situation courante, dont l'action est choisie au hasard et dont la *fitness* est égale à la *fitness* moyenne de la population. Chaque test de la partie [Condition] peut prendre la valeur # avec une probabilité fixée à 33 %.

5.5. *Paramètres*

Pour tous les paramètres de ZCS, (Bull *et al.*, 2002) donne les valeurs par défaut suivantes : taille de la population $P = 400$, *fitness* initiale $S_0 = 20.0$, taux d'apprentissage $\alpha = 0.2$, facteur d'actualisation $\gamma = 0.71$, taxe $\tau = 0.1$, taux de déclenchement de l'AG $p = 0.25$, taux de croisement $P_c = 0.5$, taux de mutation $P_m = 0.002$, taux de déclenchement de l'opérateur de couverture $\Phi = 0.5$.

5.6. *Évolution récente*

Au moment où nous rédigeons cette synthèse, la dernière publication portant spécifiquement sur les mécanismes internes de ZCS date de 2002 (Bull *et al.*, 2002). Cette

16. Pour *Action-set*.
17. *Covering.*

situation traduit un quasi-abandon des systèmes de classeurs basés sur la force au profit des systèmes basés sur la précision. Des travaux plus généraux portant sur cette famille de systèmes continuent tout de même à être publiés (Bull, 2004a; Bull, 2005).

Avant cet abandon, ZCS avait fait l'objet de deux type d'extensions, que l'on retrouvera aussi dans le cadre de XCS : l'ajout de registres internes (Cliff *et al.*, 1994) et l'ajout d'un mécanisme de chaînage des classeurs (Tomlinson *et al.*, 1998). Ces deux mécanismes sont destinés à traiter des problèmes non-markoviens, nous présenterons leur analogue proposé en extension à XCS.

Par ailleurs, ZCS a donné lieu à quelques applications. En particulier, (Bull, 1998; Bull, n.d.) a utilisé ce système pour représenter des agents économiques sur un marché simplifié, (Cao *et al.*, 1999; Cao *et al.*, 2001) l'a utilisé pour le contrôle de simulations de carrefours routiers et (Miramontes Hercog *et al.*, 2002) s'est servi de ZCS pour résoudre des problèmes classiques de comportement collectif.

Le principal problème auquel est confronté ZCS est le fait que des classeurs trop généraux susceptibles d'être activés à la fois pour faire une action adéquate près du but et une action inadéquate loin du but font la moyenne entre une espérance de récompense élevée – quand ils sont près du but – et une espérance de récompense faible – quand ils sont loin du but – et cette moyenne peut être supérieure à l'espérance de récompense de classeurs qui proposent une action adéquate loin du but, si bien que l'AG ne va conserver que les premiers, alors que ce sont les seconds qui sont nécessaires pour bien se comporter loin du but.

(Bull *et al.*, 2002) montre sur un problème jouet que, moyennant un paramétrage adéquat, on parvient à résoudre cette difficulté, mais cette publication n'a pas suffit à compenser le fait que XCS propose une solution beaucoup plus élégante à ce problème, assurant la suprématie des systèmes basés sur la précision vers lesquels nous allons nous tourner à présent.

6. Les systèmes basés sur la précision : XCS

6.1. *Présentation*

Le système XCS, imaginé par (Wilson, 1995) peu après la mise au point de ZCS, est le système de classeurs le plus étudié et le plus utilisé depuis quelques années (Kovacs, 2002). Les nombreuses recherches auxquelles il a donné lieu ont permis de considérablement progresser dans la compréhension des mécanismes qui font son succès et c'est le système de classeurs qui fait l'objet des avancées les plus significatives, comme nous allons le faire apparaître dans ce qui suit. Le lecteur trouvera dans (Butz *et al.*, 2002) une description algorithmique très détaillée de XCS qui fait référence et dans (Butz *et al.*, 2004) une analyse synthétique de ses mécanismes et de leurs interactions.

L'idée fondamentale au cœur de XCS consiste à découpler les processus d'apprentissage par renforcement et d'évolution des règles en introduisant une fonction

de *fitness* non plus proportionnelle à l'espérance de récompense, mais à la précision de la prédiction associée à cette espérance. Les classeurs qui sont conservés ne sont plus ceux qui prédisent une récompense élevée, mais ceux qui prédisent avec acuité la récompense qui sera effectivement reçue. Par conséquent, la différence entre ZCS et XCS est que le second conserve au sein de sa population les classeurs qui se déclenchent loin d'une source de récompense, et prévoient donc une récompense faible, pourvu qu'il la prévoie exactement. Ainsi, la couverture de l'espace d'état est plus complète avec XCS qu'avec ZCS.

Par ailleurs, le mécanisme de généralisation de XCS s'efforce de regrouper sous une même partie [Condition] les situations pour lesquelles l'espérance de récompense est similaire, faute de quoi le classeur ne pourrait être précis. Pour en savoir davantage sur la comparaison entre les deux familles, nous invitons le lecteur à se reporter à (Kovacs, 2004).

Au-delà de cette différence, les mécanismes mis en œuvre par XCS s'écartent significativement de ceux de ZCS. Nous les passons en revue dans ce qui suit.

6.2. *Choix de l'action*

Le mécanisme de gestion du compromis entre exploration et exploitation est géré de différentes manières dans différentes études expérimentales portant sur XCS. Le choix de l'action déclenchée se fait soit en considérant chaque action individuellement, soit en regroupant les classeurs qui proposent la même action dans un « tableau de prédiction » [18]. Parmi les mécanismes, on trouve le recours à ϵ-$greedy$, l'alternance d'essais de pure exploration et d'essais de pure exploitation, ou bien le mécanisme de roue de la fortune utilisé aussi dans ZCS.

6.3. *Évaluation des classeurs*

L'algorithme d'évaluation des classeurs est plus proche dans l'esprit de l'algorithme Q-LEARNING que de BUCKET BRIGADE. Il repose de façon systématique sur la règle de mise à jour d'une estimation que nous avons donnée avec l'équation [2]. Le mécanisme de renforcement local, appliqué exclusivement aux classeurs de [A], est le suivant :

– on commence par ajuster l'espérance de récompense p à partir de la récompense instantanée reçue r de la façon suivante : $p \leftarrow p + \beta(r - p)$,

– puis on calcule l'erreur de prédiction associée à cette espérance $\epsilon \leftarrow \epsilon + \beta(|r - p| - \epsilon)$

18. *Prediction array.*

– on en déduit la précision de la prédiction [19] *brute* $k = \begin{cases} 1 & \text{si } \epsilon < \epsilon_0 \\ \alpha(\frac{\epsilon}{\epsilon_0})^{-\nu} & \text{sinon} \end{cases}$ où $\nu > 0$.

– on ajuste ensuite la précision de la prédiction *relative* en fonction de la valeur qu'elle prend pour les autres classeurs $k' = \frac{k}{\sum_{x \in A} k_x}$

– enfin, on en déduit la *fitness* f des classeurs : $f \leftarrow f + \beta(k' - f)$

Dans le cas particulier des problèmes de décision séquentielle, Butz a obtenu un gain de performance significatif en remplaçant le mécanisme ci-dessus par une technique de descente de gradient pour l'évaluation de la récompense cumulée que peut fournir chaque classeur (Butz *et al.*, 2003b).

6.4. *Création et suppression de classeurs*

Comme dans ZCS, la création de classeurs est assurée conjointement par un algorithme génétique et un opérateur de couverture. Mais, au contraire de ce qui se fait dans ZCS, l'algorithme génétique de XCS est appliqué dans [A] et non pas dans la population globale. Cela induit une compétition entre les classeurs qui s'activent dans les mêmes états plutôt qu'une compétition globale. Cela contribue donc à résoudre le problème principal de ZCS, à savoir qu'une règle forte mais fausse applicable près du but l'emporte sur une règle juste mais applicable loin du but. Cet algorithme est déclenché tous les θ_{GA} pas de temps. Les parents sont choisis selon un mécanisme de roue de la fortune avec une probabilité proportionnelle à leur *fitness*. Récemment, Butz (Butz *et al.*, 2003c) a montré que l'on obtenait une augmentation significative de la performance du système XCS en choisissant l'action par un mécanisme de tournoi [20] au sein de [M], que ce soit pour des problèmes de classification ou de décision séquentielle.

Deux rejetons de ces parents sont insérés dans la population globale, après application d'un opérateur de mutation soit libre, soit guidé par la perception courante. On parle de mutation guidée lorsqu'un bit ne peut pas muter vers une valeur opposée à celle qui est présente dans la perception courante.

L'opérateur de couverture se charge d'ajouter des classeurs lorsque des actions sont absentes de [A]. Il ajoute un classeur qui propose l'action manquante, et dont la partie [Condition] est compatible avec la perception courante, mais généralisée par l'ajout de # avec une probabilité P(#) pour chaque test.

La population globale de XCS est de taille bornée. Quand on crée de nouveaux classeurs, si la taille limite de la population est atteinte, il faut en supprimer autant de la population globale qu'on en crée. Le mécanisme de suppression attribue à chaque classeur une probabilité d'être supprimé proportionnelle à une estimation de la taille moyenne des *action set* dans lesquels ce classeur intervient. Puis un classeur est éli-

19. *Accuracy.*
20. *Tournament selection.*

miné par un mécanisme de roue de la fortune fondé sur cette probabilité. Pour estimer la taille moyenne des *action set* dans lesquels le classeur intervient, on utilise un estimateur a_s auquel on applique une règle de mise à jour classique (voir équation 2) à chaque fois que le classeur participe à un *action set* : $a_s = a_s + \beta(taille - a_s)$ où $taille$ est la taille de l'*action set* courant.

À ces mécanismes généraux de création et de suppression s'ajoutent deux mécanismes supplémentaires :

– sachant que les mécanismes de création de classeurs peuvent engendrer des classeurs identiques à d'autres classeurs déjà insérés dans la population, plutôt que de conserver plusieurs exemplaires du même classeur dans la population, ce qui n'est efficace ni en mémoire ni en temps de calcul, on définit une notion de *macro-classeur* qui associe à chaque classeur existant un compteur N appelé « numérosité » [21] du nombre de fois où il est représenté dans la population. Pour déterminer si la taille maximale de la population est atteinte, on fait donc la somme des numérosités N de chacun des classeurs présents. De même, on prend en compte la numérosité des classeurs dans le mécanisme de sélection de l'action ;

– par ailleurs, il existe un mécanisme optionnel supplémentaire dit de « subsomption » [22] qui renforce la pression de généralisation au sein du système. Ce mécanisme consiste, à chaque fois qu'on crée un nouveau classeur, à vérifier s'il n'existe pas un autre classeur plus général et suffisamment fiable – en pratique, suffisamment évalué et doté d'une précision de la prédiction élevée – qui subsume ce nouveau classeur. Si c'est le cas, au lieu d'insérer le nouveau classeur dans la population, on augmente la numérosité de celui qui le subsume. Ce mécanisme n'est pas employé systématiquement car détecter si un nouveau classeur est subsumé par un autre coûte cher en temps de calcul, si bien que la subsomption ralentit le système.

6.5. *Paramètres*

Pour tous les paramètres de XCS, on trouve souvent les valeurs par défaut suivantes : taille de la population $P = 800$ ou $P = 2000$ selon la nature du problème, taux d'apprentissage et d'estimation $\alpha = 1$ et $\beta = 0.2$, temps entre deux déclenchements de l'AG $\theta_{GA} = 25$, taux de croisement $\xi = 0.8$, taux de mutation $\mu = 0.04$, taux de # dans l'opérateur de couverture $P_{\#} = 0.6$.

6.6. *Évolution récente*

Nous avons déjà mentionné un certain nombre d'avancées récentes se traduisant par des augmentations de performance pour XCS, à savoir l'utilisation d'un mécanisme de tournoi dans l'algorithme génétique (Butz *et al.*, 2003c) et d'un mécanisme

21. *Numerosity.*
22. *Subsumption.*

de descente de gradient dans l'évaluation des classeurs (Butz *et al.*, 2003b). Il nous reste à dire que, très récemment, Butz et ses collaborateurs (Butz *et al.*, to appear, 2006) ont mis en évidence l'apport de techniques importées des algorithmes évolutionnistes pour détecter la présence de « briques élémentaires » [23] au sein des parties [Condition] des classeurs lorsque la structure de la partie [Condition] du problème favorise la présence de telles briques.

Par ailleurs, d'autres avancées se traduisent par une extension du domaine d'application de XCS. Les plus anciennes de ces avancées concernent la résolution de problèmes dans lesquels l'hypothèse de Markov n'est pas vérifiée. Parmi ces problèmes dits « non markoviens », le cas des perceptions ambiguës [24] a été le plus étudié par la communauté de recherche sur les systèmes de classeurs. Le problème se produit lorsque le système peut recevoir la même perception pour des états de l'environnement qui sont en fait différents et dans lesquels l'action la plus appropriée n'est pas forcément la même.

Les deux principaux mécanismes utilisés pour résoudre ces problèmes avec des systèmes de classeurs sont le chaînage de classeurs et la gestion de registres internes. Dans le premier cas, réalisé par CXCS (Tomlinson *et al.*, 2000) après ZCCS (Tomlinson *et al.*, 1998) qui reposait sur ZCS, on compte sur la prise immédiate d'une suite de décisions juste avant une situation ambiguë pour ne pas avoir à prendre de décision au moment où l'ambiguïté se produit.

Dans le second cas, réalisé par XCSM et XCSMH (Lanzi, 1998) après ZCSM (Cliff *et al.*, 1994) qui reposait sur ZCS, on ajoute à la partie [Condition] des tests qui portent sur la valeur de registres internes, valeur modifiée ensuite par la partie [Action]. Toute la difficulté consiste à faire en sorte que les situations ambiguës soient effectivement distinguées par différentes valeurs des registres internes aux moments adéquats. Pour une revue plus détaillée de tous les mécanismes dédiés à la résolution de problèmes d'ambiguïtés perceptives, nous renvoyons le lecteur à (Landau *et al.*, 2006).

D'autres extensions du domaine d'application de XCS ont trait à la gestion d'espaces d'états et d'action continus. XCSR (Wilson, 2000) propose de gérer des espaces d'états continus avec des tests qui portent sur l'inclusion de la valeur réelle d'une perception dans un intervalle de $\mathbb{R}$.

Dans XCSR, Wilson propose de coder ces intervalles avec deux réels qui spécifient le centre de l'intervalle et la dispersion autour de ce centre. Par la suite, Stone et Bull (Stone *et al.*, 2003) ont montré que ce codage induisait un biais nuisible à la généricité du système et ont proposé un codage alternatif consistant à spécifier les deux bornes de l'intervalle dans un ordre quelconque. Parallèlement, Wilson a proposé dans (Wilson, 2004) une extension de XCS à la gestion de récompenses qui sont une fonction continue de l'état du système.

23. *Building block.*
24. *Perceptual aliasing.*

Dans la lignée de ces travaux sur le codage de fonctions continues, une des voies de recherche les plus actives à l'heure actuelle dans le prolongement de XCS résulte du nouveau regard théorique porté par Wilson sur son système en tant que technique générique d'approximation de fonctions. Le système résultant, XCSF (Wilson, 2001), fait l'objet de nombreuses études et extensions dont les plus récentes semblent très prometteuses (Lanzi *et al.*, to appear, 2006).

7. Les systèmes basés sur l'anticipation : ALCS

7.1. *Présentation*

S'ils ont un grand nombre de caractéristiques en commun avec les systèmes basés sur la force et sur la précision, les systèmes de classeurs à anticipation s'en écartent sur un point fondamental. Au lieu de classeurs de la forme [Condition] → [Action], ils manipulent des classeurs de la forme [Condition] [Action] → [Effet]. La partie [Effet] représente les effets attendus de l'action [Action] dans toutes les situations qui vérifient la partie [Condition] du classeur. Acquérir un tel ensemble de règles revient à apprendre ce que l'on appelle en apprentissage par renforcement un modèle des transitions, ce qui fait des systèmes de classeurs à anticipation des systèmes d'apprentissage par renforcement indirect [25], dont le prototype est constitué par les architectures DYNA (Sutton, 1990). Les systèmes de classeurs à anticipation peuvent donc être vus comme combinant deux propriétés cruciales pour les systèmes d'apprentissage par renforcement. Comme les architectures DYNA, ils apprennent un modèle des transitions, ce qui les dote de propriétés d'anticipation et leur permet d'apprendre plus vite. Comme les systèmes de classeurs classiques, ils sont dotés d'une capacité de généralisation, ce qui leur permet de construire un modèle plus compact que les architectures DYNA tabulaires (Gérard *et al.*, 2003).

D'un point de vue historique, (Riolo, 1991) est le premier à avoir présenté un système de classeurs intégrant une forme d'anticipation explicite. Son système, CFSC2, est directement inspiré du cadre original des systèmes de classeurs à messages internes proposé par Holland (Holland *et al.*, 1978). Le premier système de classeurs à anticipation conçu après les simplifications apportées par Wilson au domaine (cf. section 3) est ACS (Stolzmann, 1998; Butz *et al.*, 2000). Au cœur d'ACS, l'algorithme ALP (*Anticipatory Learning Process*) est une formalisation de la théorie psychologique du contrôle anticipatif du comportement développée par (Hoffmann, 1993). La problématique de l'anticipation y joue donc un rôle central. Au fil des années, ACS a été enrichi par Butz pour devenir ACS2 (Butz, 2002). Parallèlement, Gérard a proposé YACS (Gérard *et al.*, 2002) et MACS (Gérard *et al.*, 2005).

Dans ACS, ACS2 et YACS, la partie [Effet] de chaque classeur indique quels sont les attributs qui changent et quels sont ceux qui ne changent pas. À cette fin, les parties [Effet] peuvent contenir un symbole "=", qui signifie que l'attribut ne change pas. Par

25. *Model-based Reinforcement Learning.*

exemple, appliqué à la situation [1031], le classeur [#0#1] [0] [=10=] prédit que la situation résultant de l'application de l'action [0] sera [1101]. Appliqué à [2011], il prédit [2101]. Ce formalisme permet de représenter des régularités comme par exemple *« quand l'agent perçoit un mur au nord, quoi qu'il perçoive dans les autres directions, se déplacer vers le nord le conduit à rester dans la même case, donc aucun changement ne sera perçu dans sa situation »*, ce qui s'exprime par le classeur suivant : [1#######] [Nord] [========].

Par opposition, dans MACS, on représente des régularités entre différents attributs avec un classeur de la forme [#1#######] [Est] [1????????], où le symbole "?" signifie que le classeur ne sait pas prédire la valeur de l'attribut considéré. L'introduction de ce nouveau symbole permet donc de prédire séparément la valeur des différents attributs au pas de temps suivant. En l'occurrence, dans MACS, on choisit de ne prédire qu'un seul attribut par partie [Effet].

Des résultats expérimentaux sur la compacité du modèle bâti par MACS et sur sa vitesse de convergence (Gérard *et al.*, 2005) ont montré que ce système produit un modèle un peu plus compact que YACS, qui lui-même construit des modèles quatre fois plus compacts qu'ACS. En outre, MACS construit ce modèle trois fois plus vite que YACS, et 9 fois plus vite qu'ACS en nombre de pas de temps. La conséquence de ces performances est que MACS est capable de traiter dans des délais raisonnables des problèmes dont la structure est plus complexe que celle des problèmes traités par les systèmes concurrents. Cette efficacité provient de choix conceptuels plus clairs, mais s'obtient aussi au prix d'une complexité du logiciel sensiblement plus grande que celle des autres systèmes. Par ailleurs, MACS est doté d'un domaine d'application plus restreint. En effet, les heuristiques ont été définies spécifiquement pour traiter des problèmes déterministes plutôt que stochastiques et, comme Gérard y insiste dans sa thèse (Gérard, 2002), l'extension à la résolution de problèmes dans lesquels l'hypothèse de Markov n'est pas vérifiée s'avère problématique. Dans ce qui suit nous insistons sur les mécanismes de MACS, en présentant brièvement ce qui les distingue des mécanismes des autres systèmes.

7.2. *Choix de l'action*

ACS, ACS2 et YACS utilisent des solutions classiques de gestion du compromis entre exploration et exploitation déjà mentionnées pour choisir l'action déclenchée à chaque instant. En revanche, afin de fonctionner efficacement sur des problèmes de grande taille, MACS est doté d'un mécanisme d'exploration active qui repose sur une politique combinant hiérarchiquement trois critères :

– l'agent choisit en priorité les actions qui lui apportent de l'information sur des transitions qu'il n'a pas encore effectuées assez souvent ;

– il choisit ensuite les actions qui lui permettent de maximiser sa récompense externe sur le long terme, ce qui correspond à l'objectif de tout apprentissage par renforcement ;

– enfin, si plusieurs actions s'avèrent équivalentes vis-à-vis des deux critères précédents, il choisit de préférence les actions qu'il n'a pas choisies depuis le plus longtemps, ce qui permet de faire face aux cas d'environnements qui évoluent au cours du temps.

7.3. *Évaluation des classeurs*

Dans MACS, à chacun des critères énoncés précédemment est associée une fonction de récompense immédiate et un processus d'itération de la valeur qui permet d'en déduire une récompense attendue sur le long terme. Pour que cette propagation s'effectue efficacement, il est nécessaire que le modèle des transitions soit le plus précis possible, d'où le choix de favoriser l'exploration de l'environnement au détriment de la recherche de récompenses externes.

On distingue donc dans MACS deux choses qui sont moins clairement séparées dans les autres systèmes :

– d'une part, les couples (situation, action) se voient affecter les trois valeurs correspondant aux trois critères énoncés dans la section précédente ;

– d'autre part, les classeurs eux-mêmes disposent de différents indicateurs de fiabilité permettant de savoir s'il faut faire évoluer ou non le modèle des transitions qu'ils constituent. Ces indicateurs sont utilisés par les heuristiques de création et de suppression de classeurs décrites ci-dessous.

7.4. *Création et suppression de classeurs*

Pour obtenir un modèle des transitions aussi général que possible et aussi exempt d'erreur que possible, les systèmes de classeurs à anticipation reposent sur la combinaison de deux heuristiques :

– on applique aux classeurs qui ne sont pas assez précis une heuristique de spécialisation des conditions ;

– on applique aux classeurs qui ne sont pas assez généraux une heuristique de généralisation des conditions.

L'utilisation conjointe de ces deux heuristiques permet de faire converger la généralité et la précision des classeurs à un niveau optimal.

Pour ce qui est de la spécialisation, tous les systèmes de classeurs à anticipation font appel à la même idée : quand un classeur général prédit parfois bien et parfois mal la situation suivante, c'est qu'il est trop général. Il faut donc spécialiser sa partie [Condition] pour qu'il ne s'applique plus que dans les situations où il prédit bien. ACS, ACS2 et YACS choisissent au hasard un test qui spécifiait le symbole # pour le transformer en un test spécialisé, tandis que MACS dispose d'une heuristique supplémentaire qui permet de choisir efficacement le test à spécialiser.

Le processus de généralisation est plus complexe. Dans ACS et ACS2, il repose sur un algorithme génétique qui tente de remplacer des classeurs spécialisés par des classeurs plus généraux qui les subsument. Dans MACS, ce processus est réalisé par un algorithme plus complexe qui s'appuie sur la fiabilité estimée des classeurs spécialisés pour déterminer de façon rationnelle si une généralisation sera bénéfique ou non.

Pour connaître de façon plus précise le détail de chacun de ces algorithmes, le lecteur est invité à se référer aux articles spécialisés sur les différents systèmes.

7.5. *Paramètres*

Chacune des heuristiques repose sur une mémoire partielle des dernières prédictions correctes et incorrectes de chaque classeur. Le nombre de prédictions précédentes prises en compte est l'un des paramètres de MACS, que l'on fixe généralement à 5. Là encore, pour tous les autres paramètres, nous renvoyons le lecteur aux descriptions détaillées des différents systèmes.

7.6. *Évolution récente*

Dans YACS et MACS, le degré maximal de généralisation était recherché uniquement dans le modèle des transitions, sans souci de généraliser le modèle des récompenses immédiates ni le modèle de la fonction valeur. Ces deux modèles apparaissent dans ces systèmes sous la forme d'une table associant une valeur à chaque état rencontré, ce qui relativise l'argument de compacité sur lequel s'appuient ces systèmes. Récemment, Butz (Butz *et al.*, 2003a) a proposé le modèle XACS qui construit également un modèle compact de la fonction valeur au sein d'ACS2, en utilisant XCS.

La tendance de recherche la plus notable dans ce domaine consiste à tenter de modifier les mécanismes de XCS pour y adjoindre une forme d'anticipation explicite. Aucune de ces tentatives n'est convaincante pour l'instant. Par ailleurs, les recherches sur les systèmes de classeurs à anticipation trouvent un prolongement naturel dans le domaine des processus décisionnels de Markov factorisés, comme nous l'évoquerons dans la section suivante.

8. Tendances et directions futures

Les recherches sur les systèmes de classeurs modernes sont encore le fait d'une communauté assez restreinte, mais qui se développe rapidement. Un des effets concrets de cette expansion devrait être une meilleure utilisation du potentiel applicatif de ces systèmes. En effet, une des forces principales des systèmes de classeurs réside dans leur capacité d'expression élevée combinée à un formalisme très lisible pour un expert humain, ce qui en fait un outil de choix pour des applications indus-

trielles complexes. Pourtant, paradoxalement, il y a peu d'applications industrielles des systèmes de classeurs qui fassent l'objet de publications académiques. Un ouvrage collectif (Bull, 2004b) fait tout de même le point sur la question. À noter que, particulièrement en France, les systèmes de classeurs sont souvent utilisés pour résoudre des problèmes complexes de décision séquentielle dans les jeux vidéo (Sanza, 2001; Sanchez, 2004; Robert, 2005). En outre, comme nous l'avons mentionné en introduction de cet article, les excellents résultats de XCS sur des problèmes de classification automatique (Bernadó *et al.*, 2001) induisent un effort de recherche appliquée important dans le domaine de la fouille de données (Holmes, 2002).

Par ailleurs, sur le plan théorique, deux voies importantes de développement peuvent être distinguées.

D'une part, les recherches foisonnantes visant à améliorer les capacités de XCS s'accompagnent de travaux théoriques destinés à mieux comprendre les fondements de l'efficacité de ce système. De tels travaux bénéficient des progrès parallèles qui sont réalisés sur la question de la convergence des approches évolutionnistes (Poli *et al.*, 1998), mais il semble que cette voie de recherche soit encore loin d'être épuisée (Drugowitsch *et al.*, 2006).

D'autre part, du côté de l'apprentissage par renforcement, la découverte récente du cadre théorique des processus décisionnels de Markov factorisés (Boutilier *et al.*, 1995) suscite de sérieux espoirs de convergence entre la communauté de recherche sur les systèmes de classeurs et la communauté beaucoup plus vaste des chercheurs spécialistes de l'apprentissage par renforcement. En effet, ce cadre théorique, dans lequel l'état est décomposé sous la forme d'une collection de variables aléatoires (Boutilier *et al.*, 2000), correspond exactement au formalisme des systèmes de classeurs, chaque test de la partie [Condition] correspondant à l'une des variables aléatoires (Sigaud *et al.*, 2004). La propriété de généralisation des systèmes de classeurs est l'équivalent de la propriété de factorisation qui justifie l'existence de ce nouveau cadre théorique. Cependant, alors que les travaux sur les processus décisionnels de Markov factorisés se sont cantonnés à résoudre des problèmes de planification en considérant que le modèle des transitions est connu à l'avance, nous venons de publier un travail dans lequel nous apprenons ce modèle des transitions (Degris *et al.*, to appear, 2006) en nous inspirant directement de nos recherches antérieures sur les systèmes de classeurs à anticipation. Cette voie de recherche reste encore largement à explorer et devrait faire l'objet de développements conséquents dans les années qui viennent.

9. Conclusion

Dans cet article, nous avons présenté les systèmes de classeurs, qui ajoutent à l'apprentissage par renforcement classique la possibilité de représenter les états sous la forme de vecteurs d'attributs et de trouver une expression compacte de la représentation ainsi obtenue. Leur formalisme permet de rassembler des processus d'apprentis-

sage et d'évolution de façon homogène, ce qui en fait une classe de systèmes particulièrement riches à l'intersection de plusieurs domaines de recherche. Ils bénéficient ainsi des avancées de ces différents domaines.

Nous espérons que cet état de l'art aura donné au lecteur désireux de s'initier à la recherche sur les systèmes de classeurs les moyens de s'orienter au sein des différents courants qui portent l'évolution rapide de ce domaine. Pour approfondir les différents aspects abordés ici, un certain nombre de nouvelles ressources sont disponibles depuis (Kovacs, 2002), qui faisait le point sur cette question en 2001. En particulier, le point d'entrée incontournable est le site dédié à la communauté de recherche sur les systèmes de classeurs, qui se trouve à l'adresse http ://lcsweb.cs.bath.ac.uk/.

Pour avoir une vue plus détaillée sur les développements récents, il faut se reporter aux actes des conférences IWLCS (Lanzi *et al.*, 2000b; Lanzi *et al.*, 2001; Lanzi *et al.*, 2002a; Lanzi *et al.*, 2002b; Stolzmann *et al.*, 2002) – dont la publication accuse un certain retard – et GECCO (Banzhaf *et al.*, 1999; Whitley *et al.*, 2000; Spector *et al.*, 2001; Langdon *et al.*, 2002; Cantu-Paz *et al.*, 2003; Deb *et al.*, 2004; Beyer *et al.*, 2005), qui rassemblent chaque année l'essentiel des travaux consacrés à ce domaine [26].

10. Bibliographie

Ahluwalia M., Bull L., « A Genetic Programming-based Classifier System », *in* W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, S. R. E. (eds), *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, Morgan Kaufmann, p. 11-18, 1999.

Bacardit J., Garrell J. M., « Evolving Multiple Discretizations with Adaptive Intervals for a Pittsburgh Rule-Based Learning Classifier System », *in* E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, J. Miller (eds), *Genetic and Evolutionary Computation – GECCO-2003*, Springer-Verlag, Berlin, p. 1818-1831, 2003.

Banzhaf W., Daida J., Eiben A. E., Garzon M. H., Honavar V., Jakiela M., E. S. R. (eds), *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, Morgan Kaufmann, 1999.

Bellman R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

Bellman R. E., *Adaptive Control Processes : A Guided Tour*, Princeton University Press, 1961.

Bernadó E., Llorà X., Garrel J. M., « XCS and GALE : a comparative study of two Learning Classifer Systems with six other learning algorithms on classification tasks », *in* P.-L. Lanzi, W. Stolzmann, S. W. Wilson (eds), *Proceedings of the fourth international workshop on Learning Classifer Systems*, 2001.

26. Les actes relatifs à l'année 2006 ne sont pas encore parus au moment où nous terminons cette synthèse.

Bertsekas D. P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 1995.

Beyer H.-G., O'Reilly U.-M., Arnold D., Banzhaf W., Blum C., Bonabeau E., Cantú Paz E., Dasgupta D., Deb K., Foste r J., de Jong E., Lipson H., Llora X., Mancoridis S., Pelikan M., Raidl G., Soule T., Tyrrell A., Watson J.-P., Zitzler E. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, ACM Press, Washington DC, june 25-29, 2005.

Booker L. B., « Do We Really Need to Estimate Rule Utilities in Classifier Systems ? », *in* P.-L. Lanzi, W. Stolzmann, S. W. Wilson (eds), *Learning Classifier Systems. From Foundations to Applications*, vol. 1813 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, p. 125-142, 2000.

Booker L., Goldberg D. E., Holland J. H., « Classifier Systems and Genetic Algorithms », *Artificial Intelligence*, vol. 40, n° 1-3, p. 235-282, 1989.

Boutilier C., Dearden R., Goldszmidt M., « Exploiting Structure in Policy Construction », *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, p. 1104-1111, 1995.

Boutilier C., Dearden R., Goldszmidt M., « Stochastic Dynamic Programming with Factored Representations », *Artificial Intelligence*, vol. 121, n° 1, p. 49-107, 2000.

Bull L., « On ZCS in Multi-agent Environments », *Lecture Notes in Computer Science*, vol. 1498, p. 471-479, 1998.

Bull L., « A Simple Payoff-based Learning Classifier System », *in* X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, A. Tino, P.and Kabán, H.-P. Schwefel (eds), *Proceedings of Parallel Problem Solving from Nature (PPSN VIII)*, Springer-Verlag, 2004a.

Bull L., « Two Simple Learning Classifier Systems », *in* L. Bull, T. Kovacs (eds), *Foundations of Learning Classifier Systems*, Springer-Verlag, 2005.

Bull L., « On using ZCS in a Simulated Continuous Double-Auction Market », p. 83-90, n.d.

Bull L. (ed.), *Applications of Learning Classifier Systems*, Springer, 2004b.

Bull L., Hurst J., « ZCS Redux », *Evolutionary Computation*, vol. 10, n° 2, p. 185-205, 2002.

Butz M., Kovacs T., Lanzi P. L., Wilson S. W., « Toward a Theory of Generalization and Learning in XCS », *IEEE Transactions on Evolutionary Computation*, vol. 8, n° 1, p. 28-46, 2004.

Butz M., Pelikan M., Llorà X., Goldberg D. E., « Automated Global Structure Extraction for Eeffective Local Building Block Processing in XCS », *Journal of Evolutionary Computation*, to appear, 2006.

Butz M. V., « An Algorithmic Description of ACS2 », *in* Lanzi *et al.* (2002a), p. 211-229, 2002.

Butz M. V., Goldberg D. E., « Generalized State Values in an Anticipatory Learning Classifier System », *in* M. V. Butz, O. Sigaud, P. Gérard (eds), *LNAI 2684 : Anticipatory Behavior in Adaptive Learning Systems*, Springer-Verlag, p. 281-301, 2003a.

Butz M. V., Goldberg D. E., Lanzi P. L., Gradient Descent Methods in Learning Classifier Systems : Improving XCS Performance in Multistep Problem, Technical Report n° 2003028, IlliGAL, 2003b.

Butz M. V., Goldberg D. E., Stolzmann W., « Introducing a Genetic Generalization Pressure to the Anticipatory Classifier Systems Part I : Theoretical Approach », *in* Whitley *et al.*

(2000), p. 34-41, 2000. Also Technical Report 2000005 of the Illinois Genetic Algorithms Laboratory.

Butz M. V., Sastry K., Goldberg D. E., « Tournament Selection : Stable Fitness Pressure in XCS », *in* E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, J. Miller (eds), *Genetic and Evolutionary Computation – GECCO-2003*, vol. 2724 of *LNCS*, Springer-Verlag, p. 1857-1869, 2003c.

Butz M. V., Wilson S. W., « An Algorithmic Description of XCS », *Journal of Soft Computing*, vol. 6, n° 3–4, p. 144-153, 2002.

Cantu-Paz E., Foster J. A., Deb K., O'Reilly U.-M., Beyer H.-G., Standish R., Kendall G., Wilson S., Harman M., Weneger J., Dasgupta D., Potter M. A., Schultz A. C., Dowsland K. A., Jonoska N., Miller J. (eds), *Proceedings of the 2003 Genetic and Evolutionary Computation Conference Workshop Program*, Springer Verlag, Chicago, IL, 2003.

Cao Y. J., Ireson N., Bull L., Miles R., « Design of a Traffic Junction Controller using a Classifier System and Fuzzy Logic », *in* B. Reusch (ed.), *Proceedings of the Sixth International Conference on Computational Intelligence, Theory and Applications (6th Fuzzy Days)*, vol. 1625 of *LNCS*, Springer-Verlag, p. 342-353, 1999.

Cao Y. J., Ireson N., Bull L., Miles R., « An Evolutionary Intelligent Agents Approach to Traffic Signal Control », *International Journal of Knowledge-based Intelligent Engineering Systems*, vol. 5, n° 4, p. 279-289, 2001.

Cliff D., Ross S., « Adding memory to ZCS », *Adaptive Behavior*, vol. 3, n° 2, p. 101-150, 1994.

Deb K., Poli R., Banzhaf W., Beyer H.-G., Burke E., Darwen P., Dasgupta D., Floreano D., Foster J. A., Harman M., Holland O., Lanzi P.-L., Spector L., Tettamanzi A., Thierens D., Tyrrell A. (eds), *Proceedings of the 2004 Genetic and Evolutionary Computation Conference Workshop Program*, Springer Verlag, Seattle, WA, 2004.

Degris T., Sigaud O., Wuillemin P.-H., « Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems », *Proceedings of the 23rd International Conference on Machine Learning (ICML'2006)*, CMU, Pensylvania, to appear, 2006.

Dorigo M., Bersini H., « A Comparison of Q-Learning and Classifier Systems », *in* D. Cliff, P. Husbands, J.-A. Meyer, S. W. Wilson (eds), *From Animals to Animats 3*, MIT Press, Cambridge, MA, p. 248-255, 1994.

Drugowitsch J., Barry A., Towards Convergence of Learning Classifier Systems Value Iteration., Technical Report n° CSBU-2006-03,, Department of Computer Science, University of Bath, 2006.

Golberg D. E., Holland J. H., « Guest Editorial : Genetic Algorithms and Machine Learning », *Machine Learning*, vol. 3, p. 95-99, 1988.

Goldberg D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA, 1989.

Gérard P., *Apprentissage latent et apprentissage par renforcement dans les systèmes de classeurs*, Thèse de doctorat de l'Université PARIS VI, 2002.

Gérard P., Meyer J.-A., Sigaud O., « Combining Latent Learning with Dynamic Programming in MACS », *European Journal of Operational Research*, vol. 160, p. 614-637, 2005.

Gérard P., Sigaud O., « Designing Efficient Exploration with MACS : Modules and Function Approximation », *Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO'03)*, Springer-Verlag, Chicago, IL, p. 1882-1893, july, 2003.

Gérard P., Stolzmann W., Sigaud O., « YACS : a new Learning Classifier System with Anticipation », *Journal of Soft Computing : Special Issue on Learning Classifier Systems*, vol. 6, n° 3-4, p. 216-228, 2002.

Herbart J. F., *Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil*, August Wilhem Unzer, Koenigsberg, Germany, 1825.

Hoffmann J., *Vorhersage und Erkenntnis [Anticipation and Cognition]*, Hogrefe, Göttingen, 1993.

Holland J. H., *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, MI, 1975.

Holland J. H., « Adaptation », *Progress in theorical biology (vol. 4)*, 1976.

Holland J. H., « Escaping Brittleness : The possibilities of general-purpose learning algorithms applied to parallel rule-based systems », *Machine Learning, An Artificial Intelligence Approach (volume II)*, Morgan Kaufmann, 1986.

Holland J. H., Holyoak K. J., Nisbett R. E., Thagard P. R., *Induction*, MIT Press, 1986.

Holland J. H., Reitman J. S., « Cognitive Systems based on Adaptive Algorithms », *Pattern Directed Inference Systems*, vol. 7, n° 2, p. 125-149, 1978.

Holmes J. H., « A new representation for assessing classifier performance in mining large databases », *in* W. Stolzmann, P.-L. Lanzi, S. W. Wilson (eds), *IWLCS-02. Proceedings of the International Workshop on Learning Classifier Systems*, LNAI, Springer-Verlag, Granada, september, 2002.

Kovacs T., « Learning Classifier Systems Resources », *Journal of Soft Computing*, vol. 6, n° 3-4, p. 240-243, 2002.

Kovacs T., *Strength or Accuracy : Credit Assignment in Learning Classifier Systems*, Springer, 2004.

Landau S., Sigaud O., « A Comparison between ATNoSFERES and LCSs on non-Markov problems », *Information Sciences*, 2006.

Landau S., Sigaud O., Schoenauer M., « ATNoSFERES revisited », *in* H.-G. Beyer, U.-M. O'Reilly, D. Arnold, W. Banzhaf, C. Blum, E. Bonabeau, E. Cantú Paz, D. Dasgupta, K. Deb, J. Foste r, E. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, E. Zitzler (eds), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, ACM Press, Washington DC, p. 1867-1874, june 25-29, 2005.

Langdon W. B., Cantu-Paz E., Mathias K., Roy R., Davis D., Poli R., Balakrishnan K., Honavar V., Rudolph G., Wegener J., Bull L., Potter M. A., Schultz A. C., Miller J. F., Burke E., Jonoska N. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, Morgan Kaufmann, New York, NY, july 9-13, 2002.

Lanzi P.-L., « Adding Memory to XCS », *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC98)*, IEEE Press, 1998.

Lanzi P.-L., « Learning CLassifier Systems from a Reinforcement Learning Perspective », *Journal of Soft Computing*, vol. 6, n° 3-4, p. 162-170, 2002.

Lanzi P.-L., Loiacono D., Wilson S., Goldberg D. E., « Classifier Prediction based on Tile Coding », *Proceedings of the 2006 Genetic and Evolutionary Computation Conference Workshop Program*, to appear, 2006.

Lanzi P.-L., Perrucci A., « Extending the Representation of Classifier Conditions Part II : From Messy Coding to S-Expressions », *in* W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, S. R. E. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, Morgan Kaufmann, Orlando, FL, p. 345-352, 1999.

Lanzi P.-L., Riolo R. L., « A Roadmap to the Last Decade of Learning Classifier Systems research (From 1989 to 1999) », *in* P.-L. Lanzi, W. Stolzmann, S. W. Wilson (eds), *Learning Classifier Systems : from Foundations to Applications*, Springer-Verlag, Heidelberg, p. 33-62, 2000a.

Lanzi P.-L., Stolzmann W., Wilson S. W. (eds), *Learning Classifier Systems. From Foundations to Applications*, vol. 1813 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 2000b.

Lanzi P.-L., Stolzmann W., Wilson S. W. (eds), *Advances in Learning Classifier Systems*, vol. 1996 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 2001.

Lanzi P.-L., Stolzmann W., Wilson S. W. (eds), *Advances in Learning Classifier Systems*, vol. 2321 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 2002a.

Lanzi P.-L., Stolzmann W., Wilson S. W. (eds), *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS2002)*, Granada, 2002b.

Llorà X., Garrell J. M., « Co-evolving Different Knowledge Representations with fine-grained Parallel Learning Classifier Systems », *in* W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, N. Jonoska (eds), *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO2002)*, Morgan Kaufmann, 2002.

Miramontes Hercog L., Fogarty T. C., « Co-evolutionary Classifier Systems for Multi-Agent Simulation », *in* D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, M. Shackleton (eds), *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, IEEE Press, p. 1798-1803, 2002.

Poli R., Langdon W. B., « Schema Theory for Genetic Programming with One-point Crossover and Point Mutation », *Evolutionary Computation Journal*, vol. 6, n° 3, p. 231-252, 1998.

Puterman M. L., Shin M. C., « Modified Policy Iteration Algorithms for Discounted Markov Decision Problems », *Management Science*, vol. 24, p. 1127-1137, 1978.

Riolo R. L., « Lookahead Planning and Latent Learning in a Classifier System. », *in* J.-A. Meyer, S. W. Wilson (eds), *From animals to animats : Proceedings of the First International Conference on Simulation of Adaptative Behavior*, MIT Press, p. 316-326, 1991.

Robert G., MHiCS, une architecture de Sélection de l'Action Motivationnelle et Hiérarchique à Systèmes de Classeurs pour Personnages Non Joueurs Adaptatifs, PhD thesis, Laboratoire d'Informatique de Paris 6, 2005.

Sanchez S., Mécanismes évolutionnistes pour la simulation comportementale d'acteurs virtuels, PhD thesis, Université de Siences Sociales Toulouse 1, Toulouse, 2004.

Sanza C., Evolution d'entités virtuelles coopératives par systèmes de classifieurs, PhD thesis, Université Paul Sabatier, Toulouse, 2001.

Seward J. P., « An Experimental Analysis of Latent Learning », *Journal of Experimental Psychology*, vol. 39, p. 177-186, 1949.

Sigaud O., *Comportements adaptatifs pour les agents dans des environnements informatiques complexes*, Mémoire d'Habilitation à Diriger des Recherches de l'Université PARIS VI, 2004.

Sigaud O., Gourdin T., Wuillemin P.-H., « Improving MACS thanks to a comparison with 2TBNs », *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'04*, Springer Verlag, p. 810-823, 2004.

Smith S. F., A Learning System Based on Genetic Algorithms, PhD thesis, Department of Computer Science, University of Pittsburg, Pittsburg, MA, 1980.

Spector L., D. G. E., Wu A., Langdon W. B., Voigt H. M., Gen M. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO01)*, Morgan Kaufmann, 2001.

Stolzmann W., « Anticipatory CLassifier Systems », *in* J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, R. Riolo (eds), *Genetic Programming*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, p. 658-664, 1998.

Stolzmann W., Lanzi P.-L., Wilson S. W. (eds), *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS2003)*, Chicago, IL, 2002.

Stone C., Bull L., « Towards Learning Classifier Systems for Continuous-Valued Online Environments », *in* E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, J. Miller (eds), *Genetic and Evolutionary Computation – GECCO-2003*, Springer-Verlag, Berlin, p. 1924-1925, 2003.

Sutton R. S., « Planning by Incremental Dynamic Programming », *Proceedings of the Eighth International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, p. 353-357, 1990.

Sutton R. S., « Generalization in Reinforcement Learning : Successul examples using sparse coarse coding », *in* D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems : Proceedings of the 1995 Conference*, MIT Press, Cambridge, MA, p. 1038-1044, 1996.

Tolman E. C., *Purposive behavior in animals and men*, Appletown, New York, 1932.

Tomlinson A., Bull L., « A Corporate Classifier System », *in* A. E. Eiben, T. Bäck, M. Shoenauer, H.-P. Schwefel (eds), *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature – PPSN V*, n° 1498 in *LNCS*, Springer Verlag, p. 550-559, 1998.

Tomlinson A., Bull L., « CXCS », *in* P.-L. Lanzi, W. Stolzmann, S. W. Wilson (eds), *Learning Classifier Systems : From Foundations to Applications*, Springer-Verlag, Heidelberg, p. 194-208, 2000.

Watkins C. J. C. H., Learning with Delayed Rewards, PhD thesis, Psychology Department, University of Cambridge, England, 1989.

Whitley L. D., Goldberg D. E., Cantú-Paz E., Spector L., Parmee I. C., Beyer H.-G. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO00)*, Morgan Kaufmann, 2000.

Wilson S. W., « Knowledge Growth in an Artificial Animat », *in* J. J. Grefenstette (ed.), *Proceedings of the 1st international Conference on Genetic Algorithms and their applications (ICGA85)*, L. E. Associates, p. 16-23, july, 1985.

Wilson S. W., « ZCS, a Zeroth level Classifier System », *Evolutionary Computation*, vol. 2, n° 1, p. 1-18, 1994.

Wilson S. W., « Classifier Fitness Based on Accuracy », *Evolutionary Computation*, vol. 3, n° 2, p. 149-175, 1995.

Wilson S. W., « Get Real ! XCS with Continuous-valued Inputs », *in* P.-L. Lanzi, W. Stolzmann, S. W. Wilson (eds), *LNAI 1813 : From Foundations to Applications*, Springer Verlag, p. 209-220, 2000.

Wilson S. W., « Function Approximation with a Classifier System », *in* L. Spector, G. E. D., A. Wu, W. B. Langdon, H. M. Voigt, M. Gen (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO01)*, Morgan Kaufmann, p. 974-981, 2001.

Wilson S. W., « Classifier Systems for Continuous Payoff Environments », *LNCS 3103 : Learning Classifier Systems*, Springer-Verlag, p. 824 - 835, 2004.

Wilson S. W., Goldberg D. E., « A Critical Review of CLassifier Systems », *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, Los Altos, California, p. 244-255, 1989.