Algorithmes d'évolution pour les réseaux de neurones

Mihail CRUCIANU

Rapport de recherche 187, Février 1997
Révisé Avril 1997

Laboratoire d'Informatique
Ecole d'Ingénieurs en Informatique pour l'Industrie
64 avenue Jean Portalis
37200 TOURS
télécopie : (+33) (0) 2 47 36 14 22
courrier électronique : crucianu@univ-tours.fr

Résumé : Nous nous intéressons à l'utilisation des algorithmes d'évolution pour le développement et l'apprentissage de réseaux de neurones. Les multiples aspects qui interviennent sont présentés, avec référence à des travaux qui les concernent. Les différents problèmes qui se posent sont mis en évidence, ainsi que les solutions qui ont été apportées. Enfin, nous essayons d'extraire des nombreux travaux existants les techniques les plus prometteuses.

Mots clés : réseaux de neurones, apprentissage, algorithmes d'évolution, algorithmes génétiques, algorithmes hybrides.

Abstract : This paper deals with the design and training of neural networks using evolutionary algorithms. We present the manifold aspects involved, together with a review of part of the abundant literature on these topics. We highlight the problems one has to solve and the various solutions proposed. Among the various techniques exposed, we eventually try to select the most promising ones.

Keywords : neural networks, learning, evolutionary algorithms, genetic algorithms, hybrid algorithms.

1. Introduction

L'étude des réseaux de neurones artificiels (RNA) et l'étude des algorithmes d'évolution (AE) se sont développés en parallèle et ont souvent été en interaction durant cette dernière décennie. Nous avons distingué entre deux types d'interactions : l'utilisation conjointe des RNA et des AE pour la résolution d'un même problème, ou l'emploi des AE pour le développement de RNA. Nous nous arrêtons dans cette présentation sur ce deuxième aspect.

Les RNA (voir [Hertz, Krogh et Palmer, 1991] ou [Jodouin, 1994] pour une introduction au domaine) peuvent être regardés comme des modèles non linéaires paramétriques[1] possédant des propriétés d'approximation universelle (voir par exemple [Hornik et al., 1989], [Sontag, 1993], [Doya, 1993], [Siegelmann et Sontag, 1994]). Les RNA ont une (lointaine) inspiration biologique : un RNA est composé d'unités simples ("neurones") connectées à travers des liens caractérisés par des valeurs numériques ("poids synaptiques"). Pour résoudre (modéliser) un problème à l'aide de RNA il est nécessaire de trouver une architecture adéquate du réseau (identification du modèle) et des valeurs optimales pour les poids des connexions (estimation du modèle).

Les AE (voir [Holland, 1975], [Goldberg, 1989] ou [Fogel, 1994] pour une introduction au domaine) sont des algorithmes de recherche (et sous certaines conditions des algorithmes d'optimisation) qui peuvent être utilisés, entre autres, pour déterminer le RNA capable de modéliser un problème. D'inspiration biologique, les AE ont un caractère stochastique et travaillent sur une population d'individus, chaque individu représentant une solution alternative. Un individu est décrit par une chaîne de valeurs numériques (génome). Les individus de la population sont évalués par rapport à un critère qui dépend du problème à résoudre. Une descendance est ensuite générée en privilégiant les meilleurs individus trouvés (sélection) et en utilisant des opérateurs comme la recombinaison et la mutation. La descendance remplace la population courante et l'algorithme continue.

Différents types d'algorithmes sont connus actuellement sous l'appellation d'algorithmes d'évolution (voir [Fogel, 1994] pour des distinctions entre ces classes) : les algorithmes génétiques, les stratégies évolutives (*evolution strategies*) et la programmation évolutive (*evolutionary programming*).

Dans ce rapport nous nous intéressons à l'utilisation des algorithmes d'évolution pour le développement et l'apprentissage de réseaux de neurones. Des connaissances élémentaires concernant les RNA ainsi que les AE sont nécessaires pour comprendre une majeure partie de la problématique présentée.

Nous commençons par une discussion concernant l'intérêt que présentent les AE par rapport à d'autres algorithmes d'apprentissage pour les RNA.

---

[1] Contrairement aux modèles paramétriques classiques, les RNA ne permettent pas en général de mettre en évidence la paramétrisation qui correspond à une classes de fonctions à approximer.

Les différentes caractéristiques des RNA pouvant bénéficier d'une optimisation par AE sont indiquées dans la section 3.

Dans la section 4 nous regardons de plus près l'utilisation des AE pour les RNA, et notamment la représentation des RNA, les fonctions de coût utilisées, les opérateurs d'évolution appropriés ou les techniques mixtes. Les difficultés de mise en oeuvre sont étudiées et les meilleures solutions mises en avant.

Nous concluons enfin en essayant de distinguer, dans la multitude de voies existantes, des chemins à suivre.

La liste de références donnée est très restreinte par rapport au volume de travaux dans ce domaine ; une bibliographie plus complète bien que non exhaustive peut être trouvée dans [Alander, 1995]. Les différents articles cités dans notre texte le sont dans un but d'illustration et ne correspondent pas nécessairement aux travaux les plus représentatifs. Aussi, nous avons insisté sur les travaux récents.

2. Pourquoi les algorithmes d'évolution

La plupart des algorithmes d'apprentissage employés pour les RNA sont déterministes, de type gradient. La dépendance entre les caractéristiques modifiables du réseau — nous les appellerons paramètres par un léger abus de langage — et le critère d'évaluation est une fonction différentiable. Optimiser de façon itérative le critère d'évaluation revient alors à apporter des modifications de faible amplitude aux paramètres dans la direction du gradient du critère d'évaluation par rapport à ces paramètres : dans le sens du gradient pour une maximisation et dans les sens contraire (descente de gradient) pour une minimisation. Une exigence fondamentale pour l'utilisation des algorithmes de gradient est donc bien évidemment la différentiablité de la dépendance mentionnée (l'existence du gradient). Bien souvent une telle dépendance est trop contraignante pour la recherche d'un modèle ou, tout simplement, ne peut pas être assurée :

- Si la fonction d'activation des unités est discontinue ou non différentiable, la dépendance de la performance du réseau des poids synaptiques ne peut être que non différentiable.
- Certaines caractéristiques du réseau, comme la présence ou absence d'une unité ou connexion, sont discrètes par leur nature et ne peuvent donc donner lieu à des dépendances continues.
- L'évaluation du réseau peut se réduire parfois soit à un critère qualitatif (par exemple fonctionnement satisfaisant ou non), soit à un critère quantitatif mais discontinu (par exemple nombre de classifications correctes).

Même si le critère de base est quantitatif et différentiable, des critères d'évaluation complémentaires à domaines de variation discrets doivent parfois être employés (par exemple la complexité du réseau).

Tous ces problèmes ne se posent pas pour les AE, qui n'utilisent pas explicitement la dépendance entre les caractéristiques modifiables des réseaux et le critère d'évaluation. Grâce à

la liberté offerte dans la définition de cette dépendance, les AE présentent un intérêt indiscutable pour l'apprentissage autonome (sans intervention humaine).

L'utilisation d'algorithmes d'apprentissage déterministes pour les RNA impose aussi une autre exigence, l'absence d'optima locaux du critère d'évaluation du réseau par rapport aux paramètres. L'expérience montre que cette exigence est rarement satisfaite dans la pratique. En dehors de l'existence de solutions optimales multiples, peu gênante, nous rencontrons souvent un nombre important de solutions sous-optimales qui correspondent à des optima locaux du critère d'évaluation du réseau. Les algorithmes de recherche déterministes sont incapables de quitter un optimum local atteint et ne permettent donc pas d'assurer la complétude[2] de la recherche. Grâce à leur caractère stochastique, les algorithmes d'évolution peuvent surmonter ce problème et, sous certaines conditions, atteindre avec une probabilité élevée un optimum global quelque soit l'ensemble des solutions initiales (voir par exemple [Fogel, 1994]). Il ne faut toutefois pas oublier qu'il existe d'autres algorithmes stochastiques n'utilisant pas des populations d'individus mais présentant pourtant la propriété de complétude mentionnée (voir par exemple [Solis et Wets, 1981]). Nous remarquerons aussi (se référer à la discussion plus détaillée de la section 4.4) que les caractéristiques des RNA rendent la fonction d'évaluation plutôt "inconfortable" pour les AE : des réseaux proches peuvent avoir des performances très différentes alors que des réseaux très différents peuvent avoir des performances similaires.

Des résultats intéressants concernant les performances relatives d'algorithmes de recherche ont été publiés récemment sous l'étiquette NFL (*No Free Lunch*, voir [Wolpert et Macready, 1995], [Wolpert, 1996a, b]). Ces résultats montrent que tous les algorithmes — dont l'algorithme de recherche purement aléatoire — sont équivalents si nous regardons leur performance moyenne sur l'ensemble des problèmes possibles, considérés équiprobables. Un algorithme peut alors être estimé supérieur à d'autres si

1° Les problèmes possibles ne sont pas équiprobables **et** la distribution de probabilités *a priori* sur l'espace des problèmes possibles est explicitement utilisée dans l'algorithme (l'algorithme est explicitement adapté aux problèmes pour lesquels il sera employé).

2° L'algorithme possède une robustesse relative : il est nettement meilleur que ses concurrents sur une faible partie de l'espace des problèmes possibles et légèrement moins performant sur le reste de cet espace. N'oublions pas qu'un des concurrents est la recherche purement aléatoire !

Les résultats NFL confirment donc qu'on ne peut pas comparer les algorithmes sans faire d'hypothèse sur la classe des problèmes à résoudre. L'expérience nous montre que, pour une classe de problèmes souvent rencontrés, l'absence de complétude des algorithmes de recherche déterministes constitue un important handicap ; dans ces cas précis les AE peuvent être très

[2] La probabilité d'atteindre un état-cible (état recherché) à partir de n'importe quel autre état est non nulle.

utiles. En revanche, dans ces mêmes cas, les AE ne présentent pas nécessairement d'avantages par rapport à leurs concurrents stochastiques !

Notons enfin que les AE peuvent s'avérer intéressants quand notre but est explicitement de développer une population de RNA répondant à certains critères, par exemple pour la modélisation par une combinaison d'estimateurs. Ainsi, dans [Opitz et Shavlik, 1996] les AE permettent d'améliorer les performances des réseaux individuels, tout en encourageant une bonne diversité dans la population finale — pour des réseaux différents, les erreurs sont maximales sur des parties différentes du corpus d'apprentissage.

3. Objet de l'optimisation par AE

Un modèle de type RNA est défini par un nombre assez important de paramètres (au sens général du terme) et tous ces paramètres peuvent faire l'objet d'une optimisation par AE.

Les poids des connexions constituent le plus souvent des variables dont les valeurs sont optimisées par AE ou par des algorithmes mixtes (voir la section 4.5). Dans certains travaux (comme [Maniezzo, 1993]) la précision utilisée pour les valeurs des poids est aussi optimisée par AE ; la précision des poids est un des paramètres de contrôle de la complexité du réseau, son optimisation devrait permettre d'améliorer la capacité de généralisation du réseau.

Pour un réseau dont le nombre de couches et les nombres d'unités par couche sont donnés nous pouvons optimiser la connectivité par AE. La connectivité est un autre paramètre de contrôle de la complexité. Réduire la connectivité d'un réseau peut avoir d'autres effets désirables : apprentissage plus rapide (notamment pour les algorithmes mixtes), extraction facilitée de connaissances à partir du réseau après apprentissage.

Le nombre et la position (distribution en couches) des unités du réseau peut aussi être optimisé par AE. En général la connectivité du réseau est optimisée en même temps. Notons que le nombre et la position des unités constituent aussi des variables de contrôle de la complexité. Dans [Schiffmann et al., 1993] on constate qu'au cours de l'apprentissage par AE le nombre d'unités augmente légèrement mais en même temps le nombre de connexions baisse fortement, ce qui produit globalement une diminution importante de la complexité (nombre de degrés de liberté) des réseaux de la population.

Quand l'AE est employé en conjonction avec un algorithme d'apprentissage complémentaire (en général un algorithme de gradient, voir la section 4.5), les paramètres de l'algorithme complémentaire peuvent être optimisés par l'AE ; il peut s'agir de la vitesse d'apprentissage, de l'inertie, de l'intervalle d'initialisation des poids, du nombre de pas d'apprentissage, du paramètre d'oubli (variable de contrôle de la complexité du modèle, voir [Hinton, 1986], [Weigend et al., 1991]). Parmi les travaux concernés citons [Fekadu et al., 1993], [Robbins et al., 1993].

Les AE peuvent être employés pour optimiser non seulement les paramètres mais aussi la **forme** d'un algorithme d'apprentissage pour les RNA. Ainsi, dans [Bengio et al., 1994] la programmation génétique ([Koza, 1992]) est employée pour trouver une règle d'apprentissage applicable à la place de la rétropropagation du gradient ([Rumelhart et al., 1986]). [Lucas, 1996] propose d'utiliser un algorithme stochastique[3] — qui peut être un AE — pour optimiser l'algorithme d'apprentissage complémentaire des réseaux. L'AE est ainsi employé non plus pour obtenir directement un modèle du problème, mais pour optimiser un algorithme d'apprentissage (algorithme d'identification et d'estimation de modèles). Dans le cadre de l'AE l'évaluation de chaque algorithme de la population porte en général sur plusieurs problèmes distincts, donc l'algorithme complémentaire est développé pour s'appliquer à différents types de problèmes.

Remarquons que chaque fois qu'un algorithme d'optimisation agit sur les variables qui contrôlent la complexité d'un réseau, pour que les conséquences sur la capacité de généralisation du réseau soient positives il est nécessaire que le corpus utilisé pour cette optimisation soit différent du corpus d'apprentissage courant du réseau. Dans le cas contraire la complexité ne ferait qu'augmenter pour maximiser les performances sur le corpus d'apprentissage courant, avec des conséquences néfastes pour la généralisation.

L'optimisation des paramètres de RNA a été le plus souvent appliquée aux perceptrons multi-couches (PMC, [Rumelhart et al., 1986]), mais aussi à d'autres types de RNA : les réseaux récurrents (voir par exemple [Angeline et al., 1994]), les réseaux RBF (*Radial Basis Function*, [Powell, 1987]), voir [Neruda, 1995], les cartes de Kohonen ([Kohonen, 1990]), voir [Hämäläinen, 1995], LVQ (*Learning Vector Quantization*, [Kohonen, 1988]), voir [Merelo et Prieto, 1995], autres types de réseaux dont certains *ad hoc*.

Les AE peuvent optimiser l'ensemble des caractéristiques (*features*) du problème qui doit être modélisé par les RNA. Ainsi, pour la reconnaissance de l'écriture manuscrite, [Kussul et Baidyk, 1995] sélectionnent par un AG les *features* qui constituent l'entrée du réseau.

Toujours en rapport avec les RNA, les AE ont été employé pour développer le corpus d'apprentissage d'un réseau. Ainsi, dans [Ventura et al., 1995] un AG permet d'obtenir, pour un certain nombre d'arguments, l'inverse difficile à calculer d'une fonction connue ; les paires <argument, valeur de l'inverse> résultantes constituent l'ensemble d'apprentissage pour un RNA qui produira une interpolation de cette fonction inverse (considérée comme appartenant à la classe des fonctions "lisses").

[3] Notre observation : pour l'exemple présenté, la dépendence entre le critère d'évaluation et les paramètres est déroutante pour l'algorithme stochastique employé...

4. Techniques employées

Nous étudions maintenant les différents aspects qui interviennent dans l'utilisation des AE pour les RNA. Même si nous présentons ces aspects séparément, il ne faut pas oublier que les choix à faire ne sont pas indépendants.

4.1. Représentation des RNA

Les opérateurs des AE agissent directement sur une représentation des individus (RNA) à optimiser, alors que l'évaluation de ces individus est effectuée à partir de leur comportement. Le choix de la représentation est indissociable du choix des opérateurs d'évolution (OE) et le couple <représentation, opérateurs> conditionne le résultat de l'application de l'AE. Idéalement, le choix effectué devrait assurer la complétude de la recherche, la validité de tout individu obtenu et la présence d'une métrique convenable sur l'espace des représentation. Dans la pratique, plusieurs problèmes se posent :

La recherche est complète dans la mesure où toute solution possible peut être trouvée, quelque soit la population initiale. Pour les RNA, l'espace de recherche est extrêmement vaste ; il est donc indispensable de restreindre l'espace de recherche aux seules solutions potentiellement utiles. Toute la difficulté est de définir ce nouvel espace et de s'assurer qu'il contient la solution du problème à résoudre. En tenant compte des résultats connus concernant les propriétés d'approximation universelle de différentes architectures de RNA (voir [Hornik et al., 1989], [Siegelmann et Sontag, 1994]), nous pouvons réduire significativement l'espace de recherche sans diminuer sa puissance de modélisation. Les représentations peuvent alors être plus compactes mais il faut définir les OE attentivement afin de garantir une recherche complète.

Pour que l'AE fonctionne normalement il est nécessaire que toute représentation obtenue par application des OE corresponde à un RNA de l'espace de recherche (la représentation est valide). Plus on réduit l'espace des représentations pour améliorer l'efficacité de la recherche, plus il est difficile de définir des OE capables d'assurer non seulement la complétude de la recherche, mais aussi la validité des solutions obtenues. Nous avons toujours la possibilité de filtrer les candidats après application des OE afin de garder uniquement les solutions valides (voir par exemple [Fekadu et al., 1993]), mais cela réduit l'efficacité de la recherche.

L'efficacité de la recherche est affectée aussi par le caractère multimode de la traduction représentation $\rightarrow$ comportement dans le cas des RNA. En effet, en raison de symétries architecturales ou tout simplement de possibilités multiples de calculer la même fonction cible, un nombre important de réseaux peuvent présenter des comportements identiques sur le corpus d'apprentissage. Afin de réduire cette multiplicité des solutions il faut imposer des contraintes sur les représentations et les OE, mais il devient alors difficile

de savoir si la complétude de la recherche n'est pas affectée. Une partie des solutions qui ont été proposées sont présentées dans la section 4.4.

Le couple <représentation, opérateurs> définit une métrique sur l'espace de recherche par rapport à la facilité d'atteindre une solution à partir d'une autre. Idéalement, cette métrique devrait assurer que plus une solution est intéressante, plus elle peut être obtenue facilement à partir de l'état courant. Dans la pratique, une métrique garantissant un accès également facile à toutes les solutions possibles présenterait déjà beaucoup d'intérêt. Plus on impose de contraintes sur les représentations et plus les OE sont complexes, plus il devient difficile de s'assurer que la métrique résultante possède les propriétés désirées (voir aussi la section 4.4).

Regardons maintenant les différentes types de représentations proposées pour l'architecture des RNA et les valeurs des poids. Les autres paramètres éventuellement optimisés par l'AE étant en nombre très réduit, leur représentation est élémentaire : à chaque paramètre correspond une chaîne binaire ou une valeur réelle.

La technique la plus simple est celle du codage direct : à chaque élément à représenter correspond une chaîne binaire (pour les algorithmes génétiques) ou une valeur réelle (pour les stratégies d'évolution) et la représentation du réseau est la concaténation de ces représentations individuelles. Le regroupement des représentations des éléments individuels est en rapport avec les OE utilisés et sera étudiée dans la section 4.4. La longueur des représentations est le plus souvent fixe car cela facilite la définition des OE et le travail des AE. Nous pouvons alors considérer des RNA ayant un nombre maximal fixé d'unités et représenter l'architecture par la matrice d'incidence du graphe que constitue le réseau (par exemple [Fekadu et al., 1993]) ; la matrice est transformée en vecteur pour obtenir une chaîne binaire. Différentes contraintes sur les architectures peuvent se traduire en contraintes sur cette matrice, permettant de simplifier la représentation et la définition des OE. Nous pouvons par exemple restreindre la recherche à des RNA non récurrents en considérant uniquement des matrices triangulaires et en représentant dans la chaîne binaire seulement le triangle des coefficients non nuls ; quelque soit les OE, les représentations produites seront toujours valides. Si les poids sont optimisés aussi par l'AE, nous pouvons associer à chaque élément binaire de la matrice d'incidence la représentation de la valeur du poids ([Maniezzo, 1993] code et optimise aussi la précision des poids) ; cette valeur est prise en compte uniquement si le poids existe. Une telle technique de représentation est très générale (peut produire n'importe quel réseau) mais la taille des représentation croît avec le carré du nombre d'unités et, pour des représentations binaires des valeurs, avec la précision des poids. Etant donnée les difficultés des AE à optimiser des représentations très longues (voir par exemple [Fogel, 1994]), d'autres solutions plus économiques ont été étudiées. Il faut toutefois remarquer que c'est le nombre de degrés de liberté pris en compte par l'algorithme d'optimisation qui donne en dernière instance la taille de

la représentation, donc réduire cette taille revient à restreindre les degrés de liberté. Toute la difficulté est de faire en sorte que cette restriction affecte le moins possible l'espace de recherche (voir les résultats sur l'approximation universelle), et en tout cas d'éviter qu'elle ne se traduise, pour le problème donné, par une recherche incomplète.

Certains auteurs (voir par exemple [Happel et Murre, 1994]) utilisent des RNA modulaires, les modules étant dans une certaine mesure prédéfinis. Il reste alors à représenter la structure et les poids des connexions entre les modules, d'où une réduction importante de la taille des représentations et donc de la complexité de la recherche par AE. Bien sûr, les RNA obtenus ne sont pas minimaux, mais leurs performances en généralisation ne sont pas mois bonnes pour autant — la complexité du modèle (RNA) ne se réduit pas au nombre de composantes.

Dans plusieurs travaux une approche générative de la traduction représentation (génotype) → RNA à évaluer (phénotype) a été employée (morphogénèse). Ainsi, Gruau (voir [Gruau, 1992]) introduit l'idée suivante : chaque RNA est le résultat d'un processus de croissance qui part d'un réseau minimal simple et qui utilise les opérateurs définis dans un arbre de réécriture. Les arbres de réécriture sont les chromosomes et un algorithme de type programmation génétique ([Koza, 1992]) agit sur ces arbres. Ceci permet de spécifier une préférence pour certaines architectures jugées utiles *a priori* et de définir plus facilement des OE adéquats. [Gruau et al., 1996] présente une extension de cette technique et la compare à un codage direct ; les résultats montrent une meilleure efficacité pour l'approche générative sur des problèmes de taille importante (*scalability*). Une autre extension des travaux de Gruau est proposée dans [Friedrich et Moraga, 1996] : des modules "intéressants" sont détectés dans les arbres de réécriture et des OE spécifiques sont définis afin de permettre l'utilisation préférentielle de ces modules pour obtenir la nouvelle génération. En l'absence de comparaison il est difficile d'évaluer l'intérêt de cette extension.

Dans [Michel et Biondi, 1995] chaque réseau est constitué au départ d'une seule cellule qui contient un chromosome qui code un ensemble de règles de production ; chaque règle possède plusieurs préconditions — correspondant à des gènes activateurs et des suppresseurs — et plusieurs actions, comme l'ajout d'une cellule ou l'établissement d'une connexion. Les auteurs affirment que ce mécanisme permet de produire tout réseau possible ; il semble toutefois difficile de définir des OE adéquats et d'évaluer la métrique obtenue. La technique n'a pas encore été comparée à des techniques classiques.

Une approche générative très différente est proposée dans les travaux de Nolfi et Parisi (voir par exemple [Nolfi et Parisi, 1991], [Cangelosi et al., 1994]). Le processus de croissance a une forte inspiration biologique : les neurones d'un réseau apparaissent à des endroits plus ou moins précis, des "axones" poussent ensuite et permettent d'établir des connexions avec d'autres neurones. Les représentations traitées par l'AE indiquent quels neurones apparaissent en premier et où, quels sont les angles de croissance pour les liens, etc. Un certain degré de

non déterminisme peut être présent dans ce processus de croissance et produit un "bruit" (qui peut être important) dans l'évaluation des individus par l'AE. Les auteurs observent l'évolution des réseaux et essayent d'expliquer ces observations : les architectures modulaires sont privilégiées, la *fitness* peut subir des modifications rapides, la différence entre l'individu moyen et le meilleur individu augmente au fur et à mesure que les générations se succèdent. Cette technique a été développée sur des problèmes simples de vie artificielle ; il semble difficile d'envisager son utilisation pour l'évolution de réseaux complexes.

Les comparaisons entre les différentes techniques sont peu nombreuses et en général partielles. Les temps de calcul très importants pour des problèmes de taille réelle et l'existence de nombreux paramètres à régler dans l'application des AE rend les comparaisons difficiles à effectuer. [Roberts et Turega, 1995] propose la classification suivante pour les techniques de représentation : codage fort (codage direct), codage faible (le codage indique l'interconnexion entre des modules d'architecture standard, prédéfinie), codage intermédiaire (systèmes de réécriture). Des tests sont ensuite effectués afin de comparer les performances des différentes techniques sur plusieurs types de problèmes de taille variable. Les résultats plutôt mitigés indiquent que la classification mentionnée n'est pas vraiment significative, mais le nombre de problèmes étudiés est assez réduit.

[Calabretta et al., 1996] propose un codage diploïde pour les RNA et le compare à un codage classique, haploïde. Contrairement au cas du codage habituel, haploïde, pour un codage diploïde chaque individu contient deux génomes (représentations) distincts mais relativement similaires ; un seul des deux génomes s'exprime dans le phénotype de l'individu, en fonction des gènes de dominance. Le codage diploïde assure une variabilité plus importante entre les individus que le codage haploïde, grâce à la mutation des gènes de dominance. La comparaison confirme le fait que dans un environnement qui change entre plusieurs états stables, les individus diploïdes s'adaptent plus vite car ils gardent une mémoire supplémentaire : les génomes non dominants peuvent s'exprimer après une mutation des gènes de dominance.

Nous pouvons enfin mentionner [Marti, 1992] qui optimise aussi la technique de représentation utilisée dans le cadre de l'AE.

4.2. Critère d'évaluation

La performance de chaque RNA est exprimée dans le cadre de l'AE par un scalaire (la *fitness*) qui permet d'obtenir un classement suffisamment discriminant des individus[4]. La *fitness* cumule la contribution des différentes composantes intervenant dans l'évaluation.

[4] Toutefois, dans [Michel et Biondi, 1995] la *fitness* utilisée est binaire (survie/disparition) et l'évolution est possible grâce a la complexification progressive de l'environnement ; l'intérêt de cette approche n'est pas clair.

La principale composante de la *fitness* est une mesure de l'inverse de l'erreur du RNA, comme l'inverse de l'erreur RMS ou le pourcentage de réponses correctes. Remarquons seulement qu'une discrétisation trop grossière du domaine de variation de la *fitness* peut rendre difficile le classement des individus et donc diminuer l'efficacité de la recherche par AE. Dans les applications de type vie artificielle des RNA et AE combinés nous pouvons rencontrer une mesure de la performance globale du réseau, comme le renforcement cumulé (voir par exemple [Nolfi et Parisi, 1991, 1993]).

Les résultats concernant l'estimation de la généralisation pour les RNA et l'utilisation de la validation croisée (voir [Zhu et Rohwer, 1996] pour des clarifications) ont été pris en compte relativement tard dans les travaux concernant l'application des AE aux RNA. Par conséquence, une évaluation de la *fitness* sur un corpus de validation distinct du corpus d'apprentissage n'est presque jamais utilisée pour l'arrêt de l'AE.

De plus en plus souvent, des composantes dépendant du type de modèle employé ou de l'application traitée sont présentes dans la *fitness*. Ainsi, pour une généralisation de LVQ, dans [Merelo et Prieto, 1995] on prend en compte aussi la distorsion du modèle, c'est à dire la distance moyenne entre les vecteurs à classer et les vecteurs représentants de classe correspondants. Dans le but de renforcer la variabilité des comportements des RNA de la population (ce qui ne signifie pas nécessairement variabilité des génomes), [Opitz et Shavlik, 1996] introduit dans la *fitness* l'écart entre les prédictions du réseau et les prédictions moyennes.

Des estimations de la complexité du modèle (RNA) développé commencent aussi à être présentes dans la *fitness*, comme la taille du RNA (voir par exemple [Mandischer, 1993], [Arena et al., 1993], [Fekadu et al., 1993], [Merelo et Prieto, 1995]) ou la durée de l'apprentissage complémentaire (voir [Mandischer, 1993], [Fekadu et al., 1993]). En général, pour les travaux moins récents, le but de la diminution de la complexité du modèle n'est pas l'amélioration de la généralisation mais tout simplement la réduction du temps de calcul.

4.3. Algorithmes d'évolution

Une grande diversité d'AE ont été employés pour développer des RNA, soit seuls, soit en conjonction avec d'autres algorithmes (voir la section 4.5). Les AE classiques (voir [Holland, 1975], [Goldberg, 1989]) sont les plus répandus : les individus de chaque population (génération) sont évalués par rapport à un critère qui dépend du problème à résoudre ; une descendance est ensuite générée en privilégiant les meilleurs individus trouvés (sélection) et en utilisant des OE ; la descendance remplace la population (génération) courante et l'algorithme continue.

Les RNA sont en général complexes et l'évaluation de chaque réseau est coûteuse, donc l'efficacité de la recherche a une importance capitale pour que l'AE soit utilisable. Pour améliorer l'efficacité, il faut réduire le nombre d'évaluations d'individus différents et maintenir à

un niveau élevé la diversité dans la population. La diversité est surtout importante quand des opérateurs de recombinaison font partie des OE employés.

Malheureusement, réduire le nombre d'opérations ne peut se réaliser qu'en diminuant la taille des populations, avec des effets très négatifs sur la diversité. Les AE classiques ne sont pas jugés capables de trouver un bon compromis entre la réduction du nombre d'opérations et le maintien de la diversité, donc un nombre assez important de techniques alternatives ont été essayées :

Pour les AE avec remplacements individuels (ou partiels) nous ne pouvons pas parler de générations (*steady state*). Les individus de la population ne sont pas tous remplacés en même temps, mais plutôt un par un (respectivement groupe par groupe). Pour chaque étape, le moins performant des individus est éliminé et à sa place s'installe un nouvel individu. Le nombre d'opérations diminue fortement pour une même taille de la population, mais la population doit être plus large afin d'assurer une bonne diversité (voir par exemple [Alba et al., 1993]).

Les AE parallèles — plusieurs sous-populations distinctes sont maintenues et interagissent via des OE de recombinaison périodiquement toutes les *n* générations — permettent de maintenir une meilleure diversité et ont été essayés dans [Alba et al., 1993]. Une technique voisine est celle des AE à "îles" mis en oeuvre sur des RNA dans [Gruau et al., 1996].

Pour les AE avec voisinage (voir [Merelo et Prieto, 1995] pour une application aux RNA) les contraintes sont encore plus fortes ; les individus sont situés sur un maillage 2D, la recombinaison et la sélection s'effectuent en prenant en compte uniquement les voisins.

Enfin, nous pouvons citer la technique présentée dans [Baluja et Caruana, 1995] et qui consiste en une modification plus radicale des AE classiques. Chaque individu est représenté à l'aide d'un chromosome classique, à composantes binaires. Un chromosome spécial, unique, est employé pour engendrer de manière probabiliste les individus d'une génération : chaque composante du chromosome spécial est une probabilité, la probabilité pour que le bit correspondant dans les chromosomes des individus prenne la valeur 1. Après évaluation de la *fitness* de chaque individu de la population, les probabilités du chromosome spécial sont mises à jour à partir des valeurs effectives des bits correspondants dans les meilleurs individus de la population.

Nous devons insister sur le fait que les comparaisons effectuées sont peu nombreuses (parfois même inexistantes) et il est donc difficile d'évaluer l'intérêt de ces différentes propositions.

Si la quasi-totalité des travaux concernant l'application des AE aux RNA font appel à des AE à un seul niveau — toutes les variables optimisées le sont en même temps — il est utile de mentionner que des AE multi-niveaux ont été testés. Dans [Alba et al., 1993] trois AE sont imbriqués : l'AE interne optimise les valeurs des poids pour une architecture donnée, l'AE

intermédiaire optimise la connectivité pour une configuration donnée des unités et l'AE externe optimise cette configuration des unités. Etant donnée l'énorme charge de calcul que cette proposition implique, son intérêt semble être strictement conceptuel.

Différents travaux ont montré l'intérêt d'utiliser des connaissances du domaine pour engendrer la population initiale de RNA. Par exemple, [Hämäläinen, 1995] fait appel aux AE pour optimiser la connectivité des cartes de Kohonen et emploie des cartes standard (cercle, maillage 2D), choisies en fonction du problème, pour la population initiale.

4.4. Opérateurs et paramètres

Rappelons d'abord que le choix des OE à utiliser est indissociable du choix de la représentation des RNA. Nous pouvons distinguer entre deux grandes familles d'opérateurs : les opérateurs qui agissent sur les représentations individuelles et ceux qui opèrent sur plusieurs représentations pour produire une nouvelle. Nous appellerons ici les premiers des mutations et les seconds des recombinaisons. La sélection des individus peut être regardée aussi comme un opérateur ; en l'absence d'éléments spécifiques aux RNA nous n'en parlerons pas dans ce cadre.

Au sens le plus général, une mutation opère sur une seule représentation existante pour en produire une nouvelle. La modification effectuée peut changer ou non le nombre d'éléments de la représentation. Dans les AE classiques, l'opérateur de mutation agit sur les valeurs binaires ou réelles de ces éléments avec une probabilité constante ou qui diminue dans le temps ; cette probabilité reste cependant indépendante de l'identité des éléments d'une représentation et de l'identité des individus de la population. Plusieurs variations ont été proposées :

Pour [Angeline et al., 1993] plusieurs mutations sont possibles en même temps, ce qui augmente le nombre de solutions différentes faciles à atteindre à partir d'une solution donnée. Afin de diminuer le caractère destructif des mutations, le taux est inversement proportionnel à la *fitness* de l'individu et les modifications de faible importance sont privilégiées (avec l'hypothèse sous-jacente que la fonction d'erreur est lisse).

La même préoccupation de réduire le caractère destructif des mutations est présente dans [Schäfer et Braun, 1995], où la mutation agit sur l'existence des connexions et des unités (sauf unités d'entrée). Ainsi, la probabilité de mutation pour une connexion augmente avec la diminution de la valeur de son poids et pour une unité avec la réduction du nombre de connexions impliquées (une unité peu connectée disparaît plus facilement). Quand une unité est ajoutée, elle est complètement interconnectée avec les couches voisines. Quand une unité est éliminée, des connexions directes entre ses prédécesseurs et ses successeurs la remplacent.

Dans [Gruau et al., 1996] la mutation est appliquée après la recombinaison et le taux de mutation augmente avec la similarité des parents. La mutation a ainsi un effet compensateur par rapport à la recombinaison.

De nombreux auteurs ont constaté que des mutations de faible amplitude mais appliquées aux vecteurs de poids (donc à tous les poids) sont à préférer aux mutations localisées, de forte amplitude.

Des opérateurs de mutations particuliers sont l'ajout ou la suppression d'éléments dans les représentations. Ces opérateurs sont de plus en plus employés, malgré les difficultés engendrées pour l'AE par la taille variable des chromosomes (voir par exemple [Michel et Biondi, 1995]). [Merelo et Prieto, 1995] présente des OE adaptés à G-LVQ (généralisation de LVQ) : duplication avec modification de l'unité qui gagne le plus souvent, élimination de l'unité qui gagne le moins souvent, addition d'une unité dont les poids sont initialisés aléatoirement.

Contrairement aux opérateurs de mutation, les opérateurs de recombinaison opèrent sur plusieurs représentations (les "parents") à la fois pour en produire une nouvelle. L'intérêt d'un tel opérateur est de combiner des individus distincts qui peuvent représenter chacun une solution partielle au problème d'optimisation. Pour les algorithmes génétiques la recombinaison est l'opérateur principal et la mutation ne sert que de garantie de complétude de la recherche. Le plus connu des opérateurs de recombinaison est le croisement, mais une multitude d'autres opérateurs, plus ou moins *ad hoc*, ont été employés :

Dans [Maniezzo, 1993] deux chromosomes servent à produire un nouveau chromosome à l'aide d'une règle de décision bit par bit qui s'inspire de l'algorithme simplex ([Dantzig, 1966]).

[Robbins et al., 1993] présente un opérateur d'héritage qui traite de façon distincte chaque caractéristique codée dans le génome : le nombre de couches et les paramètres de l'algorithme d'apprentissage complémentaire (voir la section suivante) sont hérités d'un parent et ensuite mutés ; un certain nombre de couches d'éntrée sont héritées d'un parent, les autres couches de l'autre. En l'absence de comparaisons ces choix sont difficiles à justifier.

Plutôt que de combiner un nombre réduit de parents, la méthode décrite dans [Friedrich et Moraga, 1996] fait une statistique sur la distribution de modules de chromosomes dans la population entière et, dans la recombinaison, privilégie les modules qui sont les plus répandus (jugés plus prometteurs que les autres). La justification est malheureusement minime et nous sommes en droit de nous demander dans quelle mesure les modules d'arbres de réécriture correspondent à des modules fonctionnels dans les RNA développés.

Le croisement (*crossover*) est un opérateur de recombinaison traditionnel, encore le plus utilisé. Deux chromosomes sont coupés au(x) même(s) endroit(s) (le croisement peut être à un ou à plusieurs points de coupure) et les segments obtenus sont interchangés pour obtenir deux nouveaux chromosomes. L'opérateur peut être défini aussi pour des chromosomes de longueurs différentes.

Les OE et tout particulièrement le croisement doivent faire face à deux spécificités "inconfortables" des RNA : des réseaux dont les génomes sont très différents peuvent avoir des performances identiques et en même temps des réseaux très proches peuvent avoir des comportements très différents. Le premier problème est bien connu sous le nom de "conventions concurrentes" (*competing conventions*, voir entre autres [Thierens et al., 1993], [Angeline et al., 1994], [Neruda, 1995]) ; ainsi, pour un perceptron multi-couches à une seule couche cachée de n neurones nous avons $n! \cdot 2^n$ solutions équivalentes (correspondant à la permutation des unités cachées et à l'inversion, pour chaque unité, du signe des poids des connexions entrantes et sortantes). Nous pouvons en fait définir dans ce cas une décomposition de l'espace de recherche en $n! \cdot 2^n$ sous-espaces équivalents. Le nombre de solutions augmente encore si on prend en compte la possible existence de modèles différents du problème à résoudre. Dans une population quelconque il est ainsi probable que nous trouvions exclusivement des RNA appartenant à des sous-espaces distincts. Une combinaison entre deux RNA appartenant à des sous-espaces différents engendre une descendance pour laquelle certaines structures sont dupliquées tandis que d'autres structures manquent. Il est évident alors que l'application directe d'un opérateur de recombinaison classique a un effet destructif avec une probabilité proche de 1. En plus (voir le deuxième problème mentionné), un RNA qui ne modélise qu'une partie de la tâche possède en général une *fitness* très faible, il est alors peu probable qu'il soit retenu par la sélection et donc qu'il puisse contribuer après recombinaison à l'atteinte d'une solution complète. Le problème est moins grave quand la modularité du modèle cible est importante, comme c'est souvent le cas dans les systèmes de contrôle de robots ou d'*animats* (voir par exemple [Nolfi et Parisi, 1991]).

Comment résoudre le problème ? Il suffirait de restreindre la recherche à un seul de ces sous-espaces mais un sous-espace entier. Devant la difficulté de cette mission, avec (voir par exemple [Angeline et al., 1994], [Baluja et Caruana, 1995]) ou sans mention explicite, un certain nombre de travaux sur l'utilisation des AE pour les RNA ne font tout simplement pas appel à la recombinaison. Il est légitime alors de se demander si des AE qui font appel uniquement à la mutation présentent un intérêt par rapport aux algorithmes de recherche stochastique sans population (voir par exemple [Solis et Wets, 1981]) ; la réponse sera probablement très dépendante du problème de modélisation par RNA abordé. Bien que ce ne soit malheureusement pas une préoccupation de tous les auteurs, de nombreuses recherches ont été entreprises pour rendre le croisement utile dans le cas des RNA.

[Munro, 1993] limite les échanges entre deux chromosomes aux unités cachées qui travaillent de façon similaire mais pas identique. Malheureusement, l'effet de "brassage" du croisement disparaît et l'opérateur se réduit à un opérateur de mutation multiple.

Dans [Alba et al., 1993] la distance de Hamming entre deux chromosomes doit être inférieure à une valeur limite pour que le croisement puisse être appliqué. Il est difficile

d'évaluer dans quelle mesure cette technique permet de garder un seul sous-espace, complet, parmi les sous-espaces équivalents.

En imposant des contraintes probabilistes de localité sur les connexions entre des couches successives, [Braun et Weisbrod, 1993] éliminent — avec une forte probabilité — des sous-espaces équivalents, mais aussi une partie de l'espace de recherche utile.

Une méthode qui prend en compte les types de symétries possibles dans les perceptrons multi-couches est proposée dans [Thierens et al., 1993] : si la couche cachée du réseau a plus de poids négatifs que de poids positifs, les signes de tous les poids sont inversés ; avant croisement, les unités cachées du réseau sont ordonnées afin d'être aussi similaires que possible avec les unités correspondantes de l'autre réseau ; le croisement ne peut pas "casser" une unité. Ici encore il est difficile d'évaluer dans quelle mesure cette méthode permet de garder un seul sous-espace, complet.

Pour [Korning, 1996] la distribution des poids d'une unité le long du chromosome devrait encourager l'AE à faire un choix rapide entre les différents sous-espaces équivalents. Le croisement est uniforme à deux points de coupure et on peut seulement échanger le poids d'une même connexion entre deux RNA (hypothèse : l'architecture de tous les réseaux est la même).

Le problème des conventions concurrentes à été résolu dans [Neruda, 1995] pour des réseaux RBF particuliers — la sortie du réseau est une combinaison linéaire entre n gaussiennes à base radiale, ce qui engendre $n!$ sous-espaces de recherche équivalents. La première étape a été de trouver les conditions d'équivalence fonctionnelle entre deux réseaux RBF du type mentionné : il s'agit uniquement des permutations d'unités RB. La deuxième étape a permis de trouver une technique de représentation canonique des réseaux (qui garantit la complétude, tout en gardant les représentations des réseaux à l'intérieur d'un seul sous-espace) : c'est la représentation pour laquelle l'ordre lexicographique est respecté entre les vecteurs de paramètres des unités RB. Enfin, la troisième étape a été de trouver un algorithme d'apprentissage permettant d'explorer le sous-espace obtenu de façon complète, sans quitter la technique de représentation canonique (sans sortir du sous-espace) : les algorithmes de gradient ne peuvent pas garantir cela, l'auteur définit donc des opérateurs de mutation et recombinaison adéquats. Pour la mutation, le respect de l'ordre lexicographique entre les vecteurs de paramètres des unités RB donne les valeurs limite de l'intervalle de variation pour un vecteur : les vecteurs voisins. Pour le croisement, le point de coupure doit être choisi de façon à respecter l'ordre lexicographique dans la descendance.

Si le problème a été résolu pour des RBF particuliers, une solution semble beaucoup plus difficile à obtenir pour les RNA les plus utilisés, les perceptrons multi-couches (sans mentionner les RNA en général).

4.5. Techniques mixtes

Pour développer des RNA, les AE sont souvent utilisés en combinaison avec des algorithmes complémentaires (AC), en général déterministes (AD), parfois stochastiques sans population. En effet, les AE peuvent fonctionner avec un critère d'évaluation non différentiable et possèdent la propriété de complétude de la recherche, mais la complexité des calculs est importante. Les AD (en général algorithmes de gradient) sont moins puissants, mais beaucoup plus efficaces pour la recherche locale d'une solution quand le critère d'évaluation est différentiable. L'algorithme mixte serait la composition entre une recherche globale par AE et une recherche locale par AC.

Différentes combinaisons peuvent être envisagées, la solution adoptée dans un cas concret sera dépendante du problème et du type de RNA traités. Nous proposons la classification suivante :

1° L'AE et l'AC traitent des paramètres différents des RNA. La distinction correspond globalement à une décomposition du critère d'évaluation en deux composantes, la première non différentiable et traitée par AE, la deuxième différentiable et traitée donc par AC. Par exemple, l'AE prend en charge l'évolution de l'architecture, alors que l'AC fait uniquement varier les poids des connexions.

2° L'AE et l'AC traitent les mêmes paramètres. Le critère d'évaluation doit normalement être différentiable (voir toutefois [Boers et al., 1995] pour une exception, l'AD employé comme algorithme complémentaire n'est pas un algorithme de gradient). Le but de la combinaison est uniquement de rendre la recherche complète.

3° L'AE et l'AC ne traitent pas exactement les mêmes paramètres mais il existe une intersection non vide entre les deux groupes de paramètres. Les AE traitent donc non seulement la composante non différentiable du critère d'évaluation, mais aussi (une partie de) la composante différentiable, dans un souci de complétude.

En règle générale, l'AE correspond à un apprentissage inter-génération et l'AC à un apprentissage intra-génération. La *fitness* est évaluée pour tous les membres de la population à la fin de l'apprentissage par AC. Comment profiter du résultat de l'amélioration des individus d'une génération par AC au moment du passage à la génération suivante ? Deux solutions ont été envisagées, les deux sont d'inspiration biologique :

Le lamarckisme (d'après le nom du biologiste Lamarck) : les acquis intra-génération obtenus grâce à l'AC sont transmis directement, à travers un génôme modifié, à la génération suivante. Cette transmission directe rend la recherche plus efficace, mais aussi plus sensible à la présence d'optima locaux dans le critère d'évaluation. Si le codage est indirect (morphogénèse) il est souvent fort difficile de déterminer le génotype qui correspond au phénotype modifié par l'apprentissage intra-génération. Le lamarckisme peut être employé dans les trois cas de la classification précédente.

L'effet Baldwin ([Baldwin, 1896]) : l'évaluation de la *fitness* a lieu après l'apprentissage intra-génération par un algorithme complémentaire, mais ces acquis ne sont pas transmis à la génération suivante ; pour les individus de la génération suivante, les paramètres traités par l'AC peuvent être initialisés aléatoirement (dans le premier cas de la classification précédente) ou avec les valeurs définies par le génotype non modifié des parents (les deux autres cas de la classification). Dans ce cas, l'AE sélectionne progressivement les individus qui présentent les meilleurs "voisinages" par rapport au critère d'évaluation — les individus qui peuvent profiter au mieux de l'apprentissage intra-génération. L'hypothèse sous-jacente est que la probabilité pour qu'un individu explore grâce à l'AC les meilleures régions de son "voisinage" est élevée ; on peut trouver des situations dans lesquelles cette hypothèse ne se vérifie pas. Remarquons aussi que le "voisinage" d'un individu est défini par l'AC utilisé ; soumettre l'AC même à l'action de l'AE aura alors comme conséquence la sélection des individus les plus aptes à apprendre. Pour une discussion plus détaillée de l'effet Baldwin on peut regarder [Turney, 1996].

S'il n'y a aucune corrélation entre le critère d'évaluation de l'AE et celui de l'algorithme complémentaire, l'effet Baldwin est absent. Il est toutefois possible que même en l'absence d'une telle corrélation le résultat soit positif, car les individus les plus robustes (les moins sensibles aux changements destructifs produits par AC ou par AE) sont privilégiés.

L'absence de transmission directe des acquis de l'apprentissage intra-génération à la génération suivante réduit la sensibilité aux minima locaux mais diminue en même temps l'efficacité de la recherche.

Un autre problème, rarement pris en compte, est celui du bruit introduit dans la recherche par AE à cause de l'évaluation de la *fitness* après l'apprentissage intra-génération. La solution est d'effectuer des apprentissages intra-génération multiples à partir de conditions initiales différentes avant d'évaluer la *fitness* (voir par exemple [Friedrich et Moraga, 1996]). Cette solution ne s'applique que si l'AE et l'AC traitent des paramètres différents des RNA.

En conclusion, si le lamarckisme peut être défini (l'association phénotype $\rightarrow$ génotype est facilement calculable) nous conseillons de faire appel à lui car il est plus efficace.

Regardons maintenant les techniques mixtes proposées dans un certain nombre de travaux :

L'effet Baldwin est explicitement mentionné dans plusieurs articles, comme [Hinton et Nowlan, 1987], [Williams et Bounds, 1994] (une critique de résultats antérieurs de Parisi, Nolfi et Cecconi) ou [Boers et al., 1995]. [Boers et al., 1995] propose, pour des réseaux modulaires, une technique déterministe de modification de l'architecture ; des unités supplémentaires sont ajoutées aux modules pour lesquels les modifications des

poids restent importantes après un certain nombre de pas de modification des poids par descente de gradient. Cette technique déterministe se superpose à l'AE.

[Braun et Weisbrod, 1993], [Schäfer et Braun, 1995] ou [Friedrich et Moraga, 1996] font appel à RPROP ([Riedmiller et Braun, 1993]) comme algorithme déterministe (algorithme de gradient) de modification des poids. Beaucoup plus rapide que la rétropropagation du gradient ([Rumelhart et al., 1986]), RPROP semble mener à une explosion des valeurs des poids, avec un mauvais effet sur la généralisation (même utilisé avec *weight decay*).

Dans [Braun et Weisbrod, 1993] les poids des connexions héritées par la descendance sont hérités aussi (lamarckisme), ce qui réduit sensiblement le temps d'apprentissage. Pour [Schäfer et Braun, 1995] aussi les poids développés par un parent sont hérités par sa descendance (hormis les poids qui apparaissent ou disparaissent), mais le lamarckisme intervient aussi dans le choix des exemples pour l'apprentissage : uniquement les exemples mal appris par la génération précédente sont employés pour l'apprentissage intra-génération dans la génération courante. L'avantage est, bien sûr, un apprentissage beaucoup plus rapide, mais le désavantage (non mentionné par les auteurs) est qu'après quelques générations l'algorithme mène à un "apprentissage du bruit" : après quelques générations les exemples mal appris seront très probablement les *outliers* de la distribution des données à modéliser...

Dans [Munro, 1993] un AE trouve de bonnes représentations internes pour les perceptrons multi-couches, avec un critère supplémentaire — les vecteurs d'entrée du RNA doivent être discriminés de manière non identique par les unités de la couche cachée. Un algorithme complémentaire de type gradient est employé uniquement pour modifier les poids entre la couche cachée et la couche de sortie des RNA.

Pour un réseau à 3 couches de poids, [Kussul et Baidyk, 1995] développent la dernière couche par un algorithme de gradient et les deux premières par AE.

[Boné et al., 1996] fait appel à un AE pour déterminer l'automate qui sera implémenté par un RNA récurrent, la machine séquentielle connexionniste (MSC, [Touzet, 1990]). La rétropropagation du gradient est ensuite employée pour les deux perceptrons multi-couches qui composent la MSC.

Dans [Kitano, 1994] l'AE permet de trouver des valeurs initiales convenables pour les poids de RNA, ensuite c'est la rétropropagation du gradient qui est utilisée pour l'apprentissage.

[Korning, 1996] fait appel à un algorithme déterministe particulier par lequel on change tour à tour la valeur de chaque bit du génome, si le changement mène à une amélioration de la *fitness* le changement est gardé. Les résultats ne sont pas à la hauteur, ce qui s'explique par le fait que la recherche est trop contrainte et souffre donc de la présence de minima locaux nombreux.

Enfin, deux remarques d'intérêt général :

Pour l'évaluation de la *fitness* après l'apprentissage intra-génération, selon les travaux, le corpus utilisé est le corpus d'apprentissage, un corpus distinct dit de "généralisation", ou un corpus mélangé. Nous proposons que le choix soit fait entre les deux solutions décrites ci-après. Pour la première solution, nous disposons d'un corpus d'apprentissage, de deux corpus dits de validation et de deux corpus dits de test ; l'apprentissage intra-génération utilise le corpus d'apprentissage et est arrêté grâce au corpus de validation 1 ; l'évaluation de la *fitness* est faite sur le corpus de test 1, l'AE est arrêté grâce au corpus de validation 2 et la capacité de généralisation du modèle obtenu est évaluée sur le corpus de test 2. Pour la deuxième solution, nous faisons appel à un corpus d'apprentissage, un corpus de validation et un corpus de test ; l'apprentissage intra-génération et l'évaluation de la *fitness* utilisent le corpus d'apprentissage (l'arrêt est fait grâce à un autre critère), l'AE est arrêté grâce au corpus de validation et la capacité de généralisation du modèle obtenu est estimée sur le corpus de test. La deuxième solution est probablement moins efficace que la première mais plus économique en terme de volume de données disponibles. Ces deux solutions correspondent à des conclusions obtenues dans la théorie de l'apprentissage (voir par exemple [Anthony et Biggs, 1992]), même relativisés par des résultats de type NFL (voir [Wolpert, 1996a, b], [Wolpert, 1994]). Les solutions intermédiaires, rencontrées dans la littérature, nous semblent difficiles à justifier.

En ce qui concerne la complexité du modèle développé, la proposition d'avoir une durée de l'apprentissage intra-génération inversement proportionnelle à la taille des réseaux (une durée de l'apprentissage constante, quelque soit la taille du réseau) pourrait nuire à la généralisation ; l'apprentissage peut ainsi être arrêté prématurément pour les réseaux de taille importante et trop tard — apprentissage par coeur — pour les réseaux de taille réduite[5]. Un critère plus fiable (voir la remarque précédente) devrait être employé pour l'arrêt de l'apprentissage intra-génération.

5. Conclusion

Nous avons passé en revue plusieurs aspects qui interviennent dans l'utilisation des algorithmes d'évolution pour le développement de réseaux de neurones artificiels. Cette étude permet de structurer les nombreux approches rencontrés, parfois très différents, et de mettre en évidence un certain nombre de problèmes ainsi que les solutions existantes.

Quand les AE sont-ils intéressants à appliquer aux RNA ? Tout d'abord dans les situations où le critère d'évaluation ne permet pas l'utilisation des algorithmes de gradient, soit parce que le gradient ne peut pas être défini, soit parce que les optima locaux sont excessivement nombreux. Ensuite, une bonne combinaison entre un AE et un algorithme de

[5] Cependant, plus un réseau est sous-dimensionné, plus ses performances en généralisation sont robustes au prolongement de l'apprentissage.

gradient peut permettre de trouver un bon compromis entre la robustesse et l'efficacité de la recherche. Enfin, grâce à leur généralité par rapport aux critères de coût et à la possibilité de suivre les changements d'un environnement non stationnaire, les AE peuvent présenter un intérêt pour l'apprentissage dans les RNA qui contrôlent des systèmes autonomes. Les avantages des AE par rapport à d'autres algorithmes stochastiques (surtout en absence de croisement) doivent toutefois être mieux évalués.

Nous avons remarqué que les comparaisons existantes sont peu nombreuses et peu générales. C'est la diversité des approches, ainsi que la complexité des calculs des AE appliqués aux RNA, qui font que ces comparaisons soient difficiles à réaliser.

Enfin, nous aimerions exprimer deux desiderata concernant l'utilisation des AE pour les RNA. D'abord, la relation entre les algorithmes développés et les particularités des problèmes traités doit être mieux explicitée et étudiée, sous peine de voir toute conclusion relativisée par les résultats NFL mentionnés dans la deuxième section. Ensuite, il serait grand temps de voir les résultats connus de la théorie de l'apprentissage — notamment ceux concernant la validation — pris en compte de façon systématique dans l'apprentissage par AE des RNA.

Remerciements

L'auteur tient à remercier Gilles Venturini et Gilles Verley pour la relecture attentive du manuscrit, les discussions enrichissantes et les nombreuses suggestions.

Références

Alander, J. T. (1995) *An Indexed Bibliography of Neural Networks and Genetic Algorithms: Years 1985-1994*, ftp://ftp.uwasa.fi/cs/report94-1/gaNNbib.ps.Z.

Alba, E., Aldana, J. F., Troya, J. M. (1993) GAs as heuristics for optimizing ANN design, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Angeline, P. J., Saunders, G. M., Pollack, J. B. (1994) An evolutionary algorithm that constructs recurrent neural networks, *IEEE Trans. on Neural Networks* 5: 54-65.

Anthony, M., Biggs, N. (1992) *Computational Learning Theory*, Cambridge University Press, Cambridge, UK.

Arena, P., Caponetto, R., Fortuna, L., Xibilia, M. G. (1993) MLP optimal topology via genetic algorithms, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Baldwin, J. M. (1896) A new factor in evolution, *American Naturalist* 30: 441-451.

Baluja, S., Caruana, R. (1995) Removing the genetics from the standard genetic algorithm, *Technical Report* CMU-CS-95-141, School of Computer Science, Carnegie Mellon university, Pittsburgh, Pennsylvania.

Bengio, S., Bengio, Y., Cloutier, J. (1994) Use of genetic programming for the search of a new learning rule for neural networks, *0-7803-1899-4/*1994 IEEE, p. 324-328.

Bishop, J. L., Bushnell, M. J., Usher, A., Westland, S. (1993) Genetic optimization of NN architectures for colour recipe prediction, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Branke, J. (1995) Evolutionary algorithms for neural network design and training, *Proceedings of the 1st Nordic Workshop on Genetic Algorithms and its Applications*, Vaasa, Finland, 1995.

Boers, E. J. W., Borst, M. V., Sprinkhuizen-Kuyper, I. G. (1995) Evolving neural networks using the "Baldwin effect", *Proceedings of ANNGA'95*, Alès, France, Springer-Verlag, p. 333-336.

Bone, R., Guillouet, D., Pasquier, F., Asselin de Beauville, J.-P. (1996) Détection d'automates par algorithme génétique pour l'apprentissage de la machine séquentielle connexionniste, *Rencontres Approches Connexionnistes en Sciences Economiques et en Gestion '96*, Nantes, France, p. 85-90.

Braun, H., Weisbrod, J. (1993) Evolving feedforward neural networks, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Calabretta, R., Galbiati, R., Nolfi, S., Parisi, D. (1996) Two is better than one: a diploid genotype for neural networks, *Technical Report*, Department of Neural Systems and Artificial Life, Institute of Psychology, National Research Council, 00137 Rome, Italy.

Cangelosi, A., Nolfi, S., Parisi, D. (1994) Cell division and migration in a "genotype" for neural networks, *Network: Computation in Neural Systems*.

Dantzig, G. B. (1966) *Applications et prolongements de la programmation linéaire*, Dunod, Paris, 1966.

de Garis, H. (1993) Circuits of production rule GenNets, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Doya, K. (1993) Universality of fully-connected recurrent networks, submitted to *IEEE Transactions on Neural Networks*.

Fekadu, A. A., Hines, E. L., Gardner, J. W. (1993) GA design of NN based electronic nose, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Fogel, D. B. (1994) An introduction to simulated evolutionary optimization, *IEEE Trans. on Neural Networks* 5: 3-14.

Friedrich, Ch. M., Moraga, C. (1996) An evolutionary method to find good building-blocks for architectures of artificial neural networks, *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96)*, p. 951-956, Granada, Spain.

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

Gruau, F. (1992) Genetic synthesis of boolean neural networks with a cell rewriting developmental process, in *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, Juin 1992, p.55-74.

Gruau, F., Whitley, D., Pyeatt, L. (1996) A comparison between cellular encoding for genetic neural networks, *NeuroCOLT Technical Report* NC-TR-96-048, CS Department, Colorado State University in Fort Collins, CO 80523, USA.

Hämäläinen, A. (1995) Using genetic algorithm in self-organizing map design, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 364-367.

Happel, B. L. M., Murre, J. M. J. (1994) The design and evolution of modular neural network architectures, *Neural Networks* 7: 985-1004.

Hertz, J., Krogh, A., Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

Hinton, G. E. (1986) Learning distributed representations of concepts, in *Proceedings of the 8thAnnual Conference of the Cognitive Science Society*, Amherst, 1986, Hillsdale: Erlbaum, p. 1-12.

Hinton, G. E., Nowlan, S. J. (1987) How learning can guide evolution, *Complex Systems* 1: 495-502.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press.

Hornik, K., Stinchcombe, M., White, H. (1989) Multilayer feedforward networks are universal approximators, *Neural Networks* 2: 359-366.

Jodouin, J.-F. (1994a) *Les réseaux de neurones — principes et définitions*, Hermès, Paris, 1994.

Jodouin, J.-F. (1994b) *Les réseaux neuromimétiques*, Hermès, Paris, 1994.

Kitano, H. (1994) Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms, *Physika D* 75: 225-238.

Kohonen, T. (1988) Learning vector quantization, *Neural Networks*, vol. 1: 303-322.

Kohonen, T. (1990) The self-organizing map, *Proceedings of the IEEE* 78: 1464-1480.

Korning, P. G. (1996) Training neural networks by means of genetic algorithms working on very long chromosomes, *file* korning.nnga.ps.Z in Neuroprose.

Koza, J. R. (1992) *Genetic Programming*, MIT Press, Cambridge, MA, USA.

Kussul, E. M., Baidyk, T. N. (1995) Genetic algorithm for neurocomputer image recognition, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 120-123.

Kwiatkowski, L., Stromboni, J.-P. (1993) Neuromimetic algorithms processing : tools for design of dedicated architectures, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Lucas, S. M. (1996) Evolving neural-network learning behaviours with set-based chromosomes, *Technical Report*, Department of Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, UK.

Mandischer, M. (1993) Representation and evolution of neural networks, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Maniezzo, V. (1993) Searching among search spaces: hastening the genetic evolution of feedforward neural networks, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Marti, L. (1992) Genetically generated neural networks II: searching for an optimal representation, in *Proceedings of IJCNN'92*, vol. 2, p. 221-226.

Merelo, J. J., Prieto, A. (1995) G-LVQ, a combination of genetic algorithms and LVQ, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 92-95.

Michel, O., Biondi, I. (1995) From the chromosome to the neural network, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 80-83.

Mitchell, R. J., Bishop, J. M., Low, W. (1993) Using a genetic algorithm to find the rules of a neural network, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Munro, P. (1993) Genetic search for optimal representations in neural networks, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Neruda, R. (1995) Functional equivalence and genetic learning of RBF networks, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 53-56.

Nolfi, S., Parisi, D. (1991) Growing Neural Networks, *Technical Report* PCIA-91-15, Department of Cognitive Processes and Artificial Intelligence, Institute of Psychology, CNR, Rome.

Nolfi, S., Parisi, D. (1993) Self-selection of Input Stimuli for Improving Performance, *Technical Report*, Department of Cognitive Processes and Artificial Intelligence, Institute of Psychology, CNR, Rome.

Opitz, D. W., Shavlik, J. W. (1996) Generating accurate and diverse members of a neural-network ensemble, dans D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.) *Advances in Neural Information Processing Systems* 8, MIT Press, Cambridge, MA, USA.

Philipsen, W. J. M., Cluitmans, L. J. M. (1993) Using a GA to tune Potts neural networks, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Powell, M. J. D. (1987) Radial bass functions for multivariable interpolation: a review, dans Mason, J. C., Cox, M. G. (eds.) *Algorithms for Approximation*, Oxford: Clarendon Press, p. 143-167.

Riedmiller, M., Braun, H. (1993) A direct adaptive method for faster backpropagation learning: the Rprop algorithm, dans H. Ruspini (ed.): *Proceedings of ICNN'93*, San Francisco, pp. 586-591.

Robbins, P., Soper, A., Rennolls, K. (1993) Use of GAs for optimal topology determination in back propagation NNs, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Roberts, S. G., Turega, M. (1995) Evolving neural network structures: an evaluation of encoding techniques, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 96-99.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986) Learning internal representations by error propagation, in Rumelhart, D. E., McClelland, J. L. (eds.) *Parallel distributed processing: explorations in the microstructure of cognition*, Vol.2: Psychological and biological models, Cambridge, MA: MIT Press., p. 7-57.

Schäfer, J., Braun, H. (1995) Optimizing classifiers for handwritten digits by genetic algorithms, *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms'95*, Alès, France, 1995, Springer-Verlag, p. 10-13.

Schiffmann, W., Joost, M., Werner, R. (1993) Application of GAs to the construction of topologies for MLPs, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Siegelman, H. T., Sontag, E. D. (1994) Neural networks with real weights: analog computational complexity, *TCS Journal* 1994.

Solis, F. J., Wets, R. J.-B. (1981) Minimization by random search techniques, *Mathematical Operations Research* 6: 19-30.

Sontag, E. D. (1993) Some topics in neural networks and control, *Report* LS 93-02, Department of Mathematics, Rutgers University, New Brunswick.

Thierens, D., Suykens, J., Vandewalle, J., De Moor, B. (1993) Genetic weight optimization of a feedforward neural network controller, *Proceedings of ANNGA'93*, Innsbruck, Austria.

Touzet, C. (1990) Contribution à l'étude et au développement de modèles connexionnistes séquentiels, *Thèse de Docteur en Sciences*, Université des Sciences et Techniques du Languedoc, Montpellier, France.

Turney, P. (1996) Myths and legends of the Baldwin effect, *ICML'96 Workshop on Evolutionary Computing and Machine Learning*, Bari, Italy, July 2-3, 1996.

Ventura, D., Andersen, T., Martinez, T. R. (1995) Using evolutionary computation to generate training set data for neural networks, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 468-471.

Weigend, A. S., Rumelhart, D. E., Huberman, B. A. (1991) Generalisation by weight-elimination applied to currency exchange rate prediction, *Proceedings of IJCNN'91*, Seattle, IEEE 1991, p. 837-841.

Williams, B. V., Bounds, D. G. (1994) Learning and Evolution in Populations of Backprop Networks, *Research Report*, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, UK.

Wolpert, D. H. (1994) The relationship between PAC, the Statistical Physics framework, the Bayesian framework and the VC framework, dans D. Wolpert (ed.) *The Mathematics of Generalization*, Addison-Wesley, Reading, MA.

Wolpert, D. H., Macready, W. G. (1995) No free lunch theorems for search, *Technical Report SFI-TR-95-02-010*, The Santa Fe Institute.

Wolpert, D. H. (1996a) The lack of a priori distinctions between learning algorithms, *Neural Computation* 8: 1341-1390.

Wolpert, D. H. (1996b) The existence of a priori distinctions between learning algorithms, *Neural Computation* 8: 1391-1420.

Yip, P. P. C., Pao, Y.-H. (1995) A perfect integration of neural networks and evolutionary algorithms, *Proceedings of ANNGA'95*, Alès, France, 1995, Springer-Verlag, p. 88-91.

Zhu, H., Rohwer, R. (1996) No free lunch for cross-validation, *Neural Computation* 8: 1421-1426.