Robotique évolutionniste : influence des pressions de sélection sur l'émergence d'une forme de mémoire interne

Tony Pinville

Thèse de Doctorat
de l'université Pierre et Marie Curie

Spécialité : **Informatique (EDITE)**

Présentée par : **M. Tony Pinville**

Pour obtenir le grade de
Docteur de l'Université Pierre et Marie Curie

Robotique évolutionniste : influence des pressions de sélection sur l'émergence d'une forme de mémoire interne

Thèse soutenue le 30 janvier 2012

Rapporteurs	Alain Dutech	-	Univ. de Lorraine - LORIA
	Pierre De Loor	-	Univ. Européenne de Bretagne - LabSticc
Directeur	Stéphane Doncieux	-	Univ. Pierre et Marie Curie - ISIR
Co-encadrant	Jean-Baptiste Mouret	-	Univ. Pierre et Marie Curie - ISIR
Examinateurs	Nicolas Bredèche	-	Univ. Pierre et Marie Curie - ISIR
	Cyril Fonlupt	-	Univ. du Littoral Côte d'Opale - LISIC

Résumé

L'un des défis actuels de la robotique consiste à concevoir des machines capables d'apprendre continuellement des savoir-faire nouveaux dans un monde continu, non contrôlé et changeant. La robotique évolutionniste aspire à atteindre cet objectif en se reposant sur l'utilisation d'algorithmes évolutionnistes afin d'optimiser des architectures de contrôle pour robot. Cette méthode a permis de construire avec succès des robots réels exhibant des comportements réactifs complexes. L'une des prochaines étapes est désormais de réussir à concevoir des architectures de contrôle pourvus de capacités plus cognitives telles que le raisonnement, la mémoire ou la prise de décision. La mémoire est un élément central de la cognition, mettre en œuvre des méthodes permettant aux robots d'acquérir une capacité de mémoire peut donc être vu comme un premier jalon nécessaire à la réalisation de comportements cognitifs de plus haut niveau.

Partant de ce constat, l'objectif de cette thèse est d'étudier la synthèse automatique d'architectures de contrôle pour robot, capable de réaliser des tâches nécessitant le développement d'une forme de mémoire interne. Nous émettons l'hypothèse que faire émerger une capacité de mémoire au sein d'une architecture de contrôle est un problème trompeur, la robotique évolutionniste ayant tendance à générer des agents prenant seulement en considération leurs perceptions courantes.

Nous proposons une approche basée sur l'utilisation de différentes pressions de sélection afin d'éviter une convergence prématurée vers des individus au comportement réactif. Nous montrons que pour favoriser l'émergence de mémoire ainsi que la capacité de généralisation, il est nécessaire : d'utiliser une fitness discrète non dirigiste qui n'introduit pas un gradient pouvant tendre vers un optimum local ; de mettre en place des mécanismes de diversités comportementales afin d'explorer efficacement l'espace de recherche ; de développer différents objectifs auxiliaires.

Pour valider cette approche nous nous basons sur différents protocoles inspirés des neurosciences.

Remerciements

Cette formidable aventure n'aurait été possible sans ces deux personnes : Stéphane Doncieux, qui m'a accordé sa confiance, il a été le directeur de thèse rêvé, à l'écoute, disponible, dévoué, professionnel, motivant ; Audrey, qui m'a encouragé à passer le pas et supporté tout au long de ces trois ans (même lorsque je me déplaçais sur une seule jambe !), jusque dans la rédaction. En un mot : merci.

Une immense pensée également pour mes parents, maman, papa, Pierre, qui ont su éveiller ma curiosité et m'ont toujours soutenu quels que soient mes choix, et pour mes petites sœurs, Nina, Didi et Marilou.

Un grand merci à tous les membres de l'ISIR pour leur inoubliable accueil. Merci, tout particuièrement, à *the natural born chercheur*, Jean-Baptiste Mouret, source intarissable de connaissances et d'informations, qui m'a tant aidé lors de nos différentes discussions, grâce à ses relectures et ses nombreux conseils. Merci à toute la team J01 : Jean I qui a débuté le même jour que moi, mais dont la rédaction de thèse s'est terminé quelque peu après ;) ; Paul et ses graphiques de consommation électrique ; Sylvain et sa transférabilité ; Ilaria et ses *Che palle !* ; Charles I et ses divisions par 37 ; Charles II et ses codex allemands du XVIIe ; Jean II et ses cassages en tout genre ; Florian et ses scripts TikZ ; Erwan et Antoine les petits nouveaux qui assurent la relève. Merci aussi aux J01 à temps partiel : Alain à partir de 16h00 pour le thé ou 14h00 pour troller avec Jean en corrigeant les dernières coquilles de mon manuscrit ; Didier à partir de 17h pour une analyse fine de n'importe quel lancer de fléchettes. Et un merci pêle-mêle à Benoit, Olivier, Medhi, Serena, Camille et Michèle. Ma grande fierté reste d'avoir réussi à mettre la majeure partie des personnes citées précédemment à la course à pieds !

Et puis il y a les amis de toujours : dédicace toute spéciale à Vince, Nico, Jaco et Mohammed qui m'a appris que les limites n'existent que dans notre tête, le cacoin/8.6 crew, la bande des luzarchois et associés.

Une petite pensée également pour tous mes anciens collègues d'ADDE/Claritas, notamment Hervé qui m'a mis le pied à l'étrier du monde du travail (il y a 13 ans déjà !) et de Bourse Direct, la direction, la dev, la prod et tout particulièrement Carole, qui m'a montré ce que signifie être enthousiaste et passionné par son métier.

Je remercie enfin mon jury qui a accepté de prendre le temps de se pencher sur mon manuscrit.

Un dernier petit mot pour tous les musiciens m'ayant accompagné durant ces trois ans : Art Tatum, Thelonius Monk, Miles Davis, John Coltrane, Art Blakey, Charles Mingus, Eric Dolphy, Archie Shepp, Tony Williams, Herbie Hancock, James Brown, Sly & the Family Stone, Parliament/Funkadelic, The Meters, The Temptations, Gil Scott-Heron, Johnny Guitar Watson, Cameo, Pleasure, Bar-kays, Ohio Players, Chuck Brown, Defunkt, Prince, War, Mandrill, Irakere, Fania All Stars, Fela Kuti, Jimi Hendrix et tant d'autres.

Table des matières

Chapitre 1

Introduction

Voyageur, il n'y a pas de chemin,
le chemin se fait en marchant.

Antonio Machado

Sommaire

1.1 Contexte

Un robot peut être défini comme un agent physique réalisant des tâches dans l'environnement dans lequel il évolue. Un agent [1] désigne une entité qui perçoit son environnement par des capteurs et agit sur cet environnement par des effecteurs.

Avant même l'existence du terme robot, les ingénieurs John Hammond Jr. et Benjamin Miessner construisirent en 1912 l'un des premiers spécimens : l'Electric Dog [2], une machine électrique à roues dont le mécanisme la faisait s'orienter vers les sources lumineuses et les suivre selon des trajectoires à la fois réactives et complexes. Le mécanisme s'inspirait du comportement phototropique d'un certain nombre d'insectes, et notamment des papillons de nuit. C'est en 1921 que l'écrivain tchèque, Karel Capek, invente le terme "Robot" dans sa pièce R.U.R. : Rossums Universal Robots. Il est créé à partir du mot tchèque "robota" qui signifie travail forcé, tâche pénible, servitude. En 1941, le célèbre auteur de SF, Isaac Asimov, invente le terme "Robotique" et prédit l'avènement de la robotique industrielle non-dangeureuse, au service de l'homme. A la fin des années 1940, des programmes de recherche sont entrepris aux Etats-Unis pour le développement de bras mécaniques contrôlés à distance afin de manipuler du matériel radioactif. C'est à partir des années 1960 que la robotique industrielle se développe. En 1961, le premier robot industriel, Unimate, est installé dans une usine de General Motors afin d'extraire des pièces d'une cellule de fonderie sous pression (Nof, 1999). En 1972, Nissan ouvre la première chaîne de production complètement robotisée.

1. Agent vient du latin *agens*, participe présent du verbe *ago* ("agir")
2. http://davidbuckley.net/DB/HistoryMakers/HM-ElectricDog1912.htm

Parallèlement à l'émergence des robots industriels se développent les premiers robots mobiles. Créées en 1950, les tortues de Walter (1950) sont dotées d'un capteur de contact et d'un capteur de lumière. Elles sont attirées par une lumière faible et repoussées par une lumière forte. Plus tard, en 1967, est conçu "Shakey", le premier robot mobile intégrant perception, planification et exécution [3] (Nilsson, 1984). Le projet Shakey avait pour objectif de créer un robot capable d'enchaîner une série d'actions : circuler dans une pièce pour trouver un bloc, le déplacer au sommet d'une plate-forme après avoir poussé un plan incliné contre elle, et ainsi de suite. Cependant, même si le système de raisonnement fonctionnait correctement, Shakey avait souvent des problèmes dans la génération des informations symboliques nécessaires à la planification et extraites à partir des données brutes des capteurs. De plus, Shakey fut seulement capable de réaliser les différentes actions de façon indépendante, avec une forte probabilité d'échec (Crevier, 1993).

A partir des années 1980 naît l'intelligence artificielle située (Brooks, 1991). Celle-ci est basée sur le fait que le "cerveau ne fait pas tout tout seul" et qu'avant d'imiter le cerveau humain, il serait plus réaliste de s'inspirer du comportement des animaux. Selon ce point de vue l'autonomie doit émerger de l'interaction avec le milieu. Brooks (1986) propose à cet effet une nouvelle architecture : l'architecture de subsomption. Au lieu d'imposer des représentations symboliques et des modèles aux robots, Brooks pense que ceux-ci doivent être immergés dans un environnement réel. Cette architecture repose sur un couplage direct entre capteurs et effecteurs. On retrouve ce principe au sein de l'approche animat (Wilson, 1991; Meyer et Guillot, 1990) qui vise à concevoir des systèmes artificiels simulés ou des robots réels inspirés des animaux, aptes à exhiber de façon autonome des capacités adaptatives dans un environnement complexe, dynamique et imprévisible (Meyer et Guillot, 1990).

D'après le rapport de World Robotics [4], la plupart des robots d'aujourd'hui peuvent se ranger dans deux catégories : d'une part la robotique industrielle ; d'autre part la robotique de service.

Au sein de la robotique industrielle nous retrouvons principalement des robots manipulateurs physiquement ancrés sur leur lieu de travail. Ils sont présents notamment dans des usines d'assemblage et généralement mis en place pour remplacer l'homme dans la réalisation de tâches précises, répétitives et dangereuses (peinture, charge lourde, environnement hostile...). Ces robots sont utilisés en industrie automobile ainsi que dans l'industrie électronique. La robotique industrielle représentait à la fin 2010 plus d'un million d'unités opérationnelles dans le monde [5].

La robotique de service peut se diviser en deux grandes catégories : les robots assistants téléopérables et les robots autonomes. Les robots assistants téléopérables participent à la réalisation d'une tâche en collaboration étroite avec un opérateur humain. Ils sont très utilisés dans l'industrie militaire comme par exemple le Pack-

3. http://www.ai.sri.com/shakey/
4. www.worldrobotics.org
5. source : World Robotics 2011

Bot [6] d'iRobot ou le Remotec ANDROS [7] qui ont servi pour différentes opérations en milieu hostile comme lors de déminages ou d'opérations à Fukushima. Un robot comme le BigDog de Boston Dynamics [8] a été développé afin de se déplacer sur la neige, la glace, ou en terrain très accidenté. L'usage de robots assistants téléopérables se développent dans de nombreux domaines comme le secteur spatial, avec des robots comme Sojourner, Spirit [9] et récemment Curiosity [10] développés par la NASA pour explorer Mars. On retrouve aussi des robots assistants téléopérables dans des domaines aussi divers que la vulcanologie (Bares et Wettergreen, 1999), l'exploration sous-marine (Singh et al., 2005), les véhicules d'assistance (Murphy, 2004) les services de nettoyage pour avion (Skywash (Schraft et Wanner, 1993)), ou encore dans le milieu médical, notamment avec le robot Da Vinci [11] qui assiste le chirurgien dans différentes opérations.

Les robots autonomes quant à eux se distinguent des autres types de robots par leur faculté à réaliser des tâches sans aide d'intervenants extérieurs. Parmi ceux-ci, certains évoluent dans un lieu contrôlé afin d'exécuter des tâches spécifiques, comme les robots de manutention proposés par Kiva systems [12] ou BA systèmes [13]. L'exemple le plus connu de robot autonome est le robot aspirateur, comme le Roomba dont les ventes ont explosées ces dernières années avec 5 millions d'unités vendues dans le monde. World Robotics prévoit que 14.4 millions de robots de service seront vendus aux particuliers sur la période 2011-2014. Des robots autonomes, comme le NurseBot [14], sont développés pour l'aide aux personnes agées ou ayant un handicap. De récents progrès ont été également réalisés au niveau des véhicules autonomes, comme le Stanley, qui a pu traverser le désert des Mojaves à une moyenne de 22 mph (Thrun et al., 2007) ou comme la Google Car désormais homologuée pour circuler dans l'état du Nevada. Différents robots autonomes sont utilisés dans des secteurs comme l'agriculture avec des tracteurs autonomes [15] ou des robots pour pour la taille des vignes [16]. Enfin, de nombreux robots humanoïdes tels que l'Asimo de Honda (Sakagami et al., 2002), le Romeo [17] d'Aldebaran, le HRP-4 [18] ou encore l'Icub (Metta et al., 2008) sont dotés de certaines capacités d'autonomie.

Un robot mobile autonome évoluant dans un monde réel non contrôlé doit faire face à des environnements partiellement observables, dynamiques, non-déterministes et continus. Partiellement observables, en raison du fait que ses capteurs ne lui permettent pas l'accès à un état complet de l'environnement à chaque

6. http://www.irobot.com/us/robots/defense.aspx
7. http://www.msha.gov/SagoMine/robotdetails.asp
8. http://www.bostondynamics.com/
9. http://marsrovers.jpl.nasa.gov/home/index.html
10. http://www.nasa.gov/mission_pages/msl/index.html
11. http://www.intuitivesurgical.com/products/davinci_surgical_system/
12. http://www.kivasystems.com
13. http://www.basystemes.com/
14. http://www.cs.cmu.edu/~nursebot/web/scope.html
15. http://www.kuleuven.be/english/news/tractor
16. http://wall-ye.com/
17. http://projetromeo.com
18. http://global.kawada.jp/mechatronics/hrp4.html

instant (des caméras ne peuvent, par exemple, pas voir dans les coins). Dynamiques, car l'environnement peut changer alors que l'agent n'agit pas. Non-déterministes, car les capteurs et effecteurs d'un agent sont sujets à différentes incertitudes notamment dues à des frictions, des glissements, du bruit, etc.

Parvenir à développer un agent artificiel autonome s'avère être un problème extrêmement difficile (Pfeifer et Bongard, 2006). La plupart des robots d'aujourd'hui utilisent des algorithmes déterministes pour la prise de décision, tel que des algorithmes de planification (LaValle, 2006). Aucun de ses algorithmes ne permet de répondre pleinement à une caractéristique clé des problèmes robotiques : l'incertitude. Celle-ci naît du fait que le robot manque d'information critique pour mener à bien ses tâches. Ce manque d'information peut provenir de différents facteurs (Thrun et al., 2005) : de l'environnement qui peut être dynamique ; des capteurs et des effecteurs qui ne sont jamais parfaits, car ils possèdent d'une part une limitation physique ne permettant pas d'obtenir un monde totalement observable et d'autre part ils sont sujets au bruit, perturbant les mesures des capteurs et les actions des effecteurs ; des capacités de calcul limités : un robot doit agir en temps réel, ses capacités de calcul sont donc limitées et ne permettent pas l'utilisation d'algorithmes ayant une complexité élevée.

Par ailleurs, cette capacité à faire face à un haut degré d'incertitude lié à un environnement complexe est présente chez tous les systèmes biologiques. L'observation de la nature peut être alors une source d'inspiration : étant donné l'autonomie développée chez les espèces biologiques, il peut s'avérer intéressant de prendre en considération quelques capacités comportementales extraites du comportement animal dans la conception d'un robot autonome.

Pour cela, nous pouvons nous appuyer sur les sciences cognitives qui ont pour objectif de comprendre les mécanismes d'acquisition, de traitement, de stockage, de récupération des connaissances par les organismes biologiques (homme et/ou animal) et par des systèmes artificiels (Wilson et Keil, 2001). Elles regroupent à la fois des modèles informatiques issus de l'intelligence artificielle et des techniques expérimentales venant de la psychologie, pour construire des théories précises et testables sur l'esprit humain (Wilson et Keil, 2001).

En psychologie cognitive, on distingue différents mécanismes impliqués dans les activités cognitives (Reed, 2011) :

- le *traitement de l'information*, qui intègre la reconnaissance de forme, l'attention et différentes formes de mémoire qui comprend la mémoire de travail à court terme et la mémoire à long terme ;
- des *mécanismes de représentation* tels que les codes de mémoire, les images visuelles, la catégorisation et l'organisation sémantique ;
- l'*organisation de la connaissance* qui comprend le langage, la compréhension et la mémoire de texte, la résolution de problèmes, l'expertise et la créativité, ou la prise de décision.

Cette catégorisation peut servir de grille de lecture dans la conception d'un robot autonome. Suivant les tâches à exécuter, un robot autonome se doit d'être doté de mécanismes de traitement de l'information, de représentation et d'organisation de

la connaissance, lui permettant de prendre des décisions.

Quels sont les différents courants au sein des sciences cognitives tentant d'expliquer le fonctionnement de ces mécanismes ?

Varela et al. (1991) distingue trois approches [19] :

- Le cognitivisme (McCarthy et al., 1955; Pylyshyn, 1984), approche symbolique ou computationnelle : elle est basée sur la métaphore cerveau-ordinateur. Dans la vision cognitiviste, la cognition est perçue comme une faculté de traiter de l'information basée sur la manipulation de symboles à partir de règles afin de représenter l'environnement.
- Le connexionnisme (McCulloch et Pitts, 1943; Rosenblatt, 1958; Rumelhart et al., 1988) : son objectif est de pouvoir concevoir un système capable d'exhiber un comportement intelligent sans stocker, récupérer de l'information sous forme d'expressions symboliques structurées. Cette approche repose sur l'utilisation d'un réseau d'unités élémentaires qui possèdent chacune un certain niveau d'activation. Dans ce cas, la cognition est basée sur l'émergence d'états globaux dans un réseau de composants simples, le plus souvent des réseaux de neurones.
- La cognition incarnée (Pfeifer et Bongard, 2006) ou enaction (Varela et al., 1991) : l'autonomie est la base fondamentale de cette approche. Elle aborde la cognition comme l'histoire du couplage structurel entre un organisme et son environnement. On peut y distinguer trois caractéristiques importantes (De Loor et al., 2009) : l'absence de représentation de l'environnemt a priori ; la plasticité, l'organisme est viable parce qu'il est capable d'absorber les perturbations provoquées par son environnement et de s'y adapter ; la co-évolution, la modification du monde par l'organisme impose en retour une modification de celui-ci. L'IA située (Brooks, 1991), qui rejette l'idée de représentation a priori, rejoint les principes de cette approche.

Le cognitivisme a montré différentes limites comme le problème d'ancrage de symboles (Harnad, 1990) ou "*symbol grounding problem*". Il est en effet difficile pour un agent de transformer ses perceptions (et ses actions) en représentations abstraites, en symboles pouvant être manipulés par le cerveau. Pfeifer et Scheier (2001) évoquent aussi le problème de la mise en situation, "*Frame problem*" et d'homonculus.

Nous avons fait ici le choix de nous baser sur une approche énactive ou cognition incarnée pour la conception d'une architecture de contrôle d'un robot autonome. Ceci est motivé par le fait qu'au sein de cette approche, l'autonomie est considérée comme fondamentale. De plus, cette approche qui considère que le corps et le cerveau forment un tout indissociable capable d'exhiber des comportements intelligents sans avoir recours à des symboles ou à des représentations abstraites, tend à se développer au sein des sciences cognitives (Wilson, 2002).

Selon Varela, un attribut dans le monde – l'espace, les couleurs, les formes, ... – n'est pas quelque chose qui préexiste. Celui-ci est configuré par les rapports

19. les différences entre ces approches tendent à s'estomper (Gershenson, 2004)

entre l'organisme et l'environnement. L'environnement ne contient pas d'attributs, c'est l'interaction entre l'environnement et l'organisme qui fait émerger ces attributs (Benkirane et al., 2002). Cette absence de représentations d'un monde prédonné implique que l'environnement est perçu différemment selon les perceptions propres à chaque organisme biologique. La chauve-souris qui utilise le principe de l'écholocation pour se déplacer (Moss et Sinha, 2003) possédera une représentation différente du monde par rapport au faucon doté d'une vue perçante (Fox et al., 1976).

Concevoir un "cerveau" pour un robot ne peut donc être qu'une simple copie d'un cerveau d'un organisme existant, quel que soit son degré de complexité. Un robot possède en effet des moyens de percevoir le monde qui l'entoure (lasers, caméras, ...) radicalement différent que ceux de n'importe quel organisme biologique. Malgré la diversité des systèmes nerveux présents dans la nature, on retrouve néanmoins certains invariants. Les unités fonctionnelles présentes dans tout système nerveux sont les cellules nerveuses ou neurones. Elles sont spécialisées dans la réception de stimuli, la conduction de l'excitation et la transmission d'un signal à d'autres cellules. Certaines structures neurales peuvent être communes chez des organismes phylogénétiquement éloignés comme le faucon et la chauve-souris. Le colliculus supérieur [20], par exemple, est une structure proche que l'on retrouve chez ces deux espèces (Hamdi et Whitteridge, 1954). Cette structure sous-corticale est impliquée dans la coordination des mouvements des yeux et de la tête.

Les neurones sont les briques fondamentales de n'importe quel organisme biologique doté d'un système nerveux. Contrôler un robot à l'aide d'un réseau de neurones artificiels capable de s'adapter à un environnement donné, peut donc constituer un bon point de départ pour la conception d'un robot autonome. Le défi se situe ensuite sur la manière d'organiser ces neurones afin d'élaborer une structure neurale qui sera adaptée aux caractéristiques propres du robot autonome. Les systèmes nerveux des organismes biologiques ont emergés par évolution, s'inspirer de ce principe pour mettre au point notre contrôleur pour robot peut être une solution.

La robotique évolutionniste (Harvey et Husbands, 1992; Meyer et al., 1998; Floreano et Nolfi, 2000) (ou ER pour *Evolutionary Robotics*) est une méthode pouvant répondre à cette problématique. Apparue dans les années 1990, l'objectif de cette approche est de développer des méthodes de conception automatiques afin de créer des robots autonomes effectuant des tâches dans des environnements non structurés et, ceci, sans la nécessité que le robot soit directement programmé par des humains. L'ER repose sur l'idée d'utiliser des algorithmes évolutionnistes afin d'optimiser des contrôleurs ou des morphologies pour des robots. Les algorithmes évolutionnistes (Eiben et Smith, 2003) (voir partie 2.1 pour plus de détails) sont une méthode d'optimisation stochastique inspirée de la théorie de l'évolution. De nombreux types de contrôleurs ont été optimisés en ER. On y retrouve principalement des réseaux de neurones (Floreano et Mondada, 1996), des programmes génétiques (Koza, 1992), des contrôleurs à base de logique floue (Hoffmann et Pfister, 1996; Barberá et Skar-

20. Par convention, le terme "colliculus supérieur" est le plus souvent réservé à un mammifère et le terme "tectum optique" pour une espèce non-mammifère et les vertébrés en général.

meta, 2002) ou simplement un ensemble de paramètres. Selon Nelson et al. (2009), les réseaux de neurones représenteraient 40% des travaux en robotique évolutionniste tandis que la programmation génétique 30%. La robotique évolutionniste a été testée sur des tâches variées telles que : l'évitement d'obstacles et le rechargement automatique dans un environnement simple (Floreano et Mondada, 1996) ; le pilotage de robots marcheurs (Kodjabachian et Meyer, 1997; Hornby et al., 2000) ou de robots volants (Mouret et Doncieux, 2006; Shim et Husbands, 2007) via un réseau de neurones ; le contrôle des nuées de robots (Großet al., 2006; Baele et al., 2009).

Jusqu'à présent, les recherches dans ce domaine se sont plutôt focalisées sur des tâches réactives comme des tâches de locomotion d'évitement d'obstacles ou de recherche de nourriture. Nolfi (2002b) a pu montrer la puissance et la limite de contrôleurs réactifs. Les comportements les plus complexes obtenus à l'heure actuelle, consistent en 3 ou 4 sous-comportements coordonnés (Nelson et al., 2009) comme la tâche de ramassage de balles (Doncieux et Mouret, 2010a; Mouret et Doncieux, 2012), ce qui ne permet pas de concevoir un robot autonome pleinement opérationnel. Un des défis actuel est déterminer si la robotique évolutionniste est capable de synthétiser des contrôleurs aux capacités cognitives plus complexes (Nelson et al., 2009).

1.2 Problématique

Parmi les mécanismes impliqués dans les activités cognitives, quels sont ceux susceptibles d'aider dans la conception de robots réalisant des tâches plus complexes ?

Pour cela, un agent doit tout d'abord être capable de traiter efficacement les informations provenant de son environnement. Nous avons vu précédemment que Reed (2011) distingue différents mécanismes impliqués dans le traitement de l'information, dont la reconnaissance de formes, l'attention et différentes formes de mémoire.

La mémoire joue un rôle critique dans la plupart, voire dans tous les aspects de la cognition. Piaget et Inhelder (1968) ont mis en exergue des dépendances entre mémoire et intelligence. En IA, Stanfill et Waltz (1987) considèrent que l'utilisation intensive de la mémoire pour rappeler des épisodes spécifiques du passé – plutôt que des règles – doit être la fondation d'une machine dotée de raisonnement [21]. En psychologie, Baddeley (1992) considère la mémoire de travail comme un élément indispensable pour réaliser des tâches cognitives complexes impliquant de l'apprentissage ou du raisonnement. La mémoire de travail étant un système s'occupant à la fois du traitement et du maintien des informations à court terme.

Mettre en œuvre des méthodes permettant aux robots d'acquérir une capacité de mémoire peut donc être vu comme un premier jalon nécessaire à l'acquisition de

21. *"The intensive use of memory to recall specific episodes from the past rather than rules-should be the foundation of machine reasoning." (Stanfill et Waltz, 1987)*

comportements cognitifs de plus haut niveau.

Partant de ce constat, l'objectif de cette thèse est *d'étudier la synthèse automatique d'architectures de contrôle pour robot capables de réaliser des tâches nécessitant le développement d'une forme de mémoire interne.*

Nous émettons l'hypothèse que synthétiser une architecture de contrôle pour agent capable de résoudre des tâches de mémoire est un problème *trompeur*[22] à plusieurs niveaux.

D'une part, la mise en place d'un protocole expérimental nécessitant une forme de mémoire n'est pas nécessairement trivial. En effet, il est souvent admis (Beer, 1996) que seul un agent non-réactif – i.e. un agent qui agit en fonction de ses perceptions courantes aussi bien que de ses perceptions passées – peut faire face à des tâches impliquant un problème d'ambiguïté des perceptions (*perceptual aliasing* (Whitehead et Ballard, 1991)). Ce problème désigne l'incapacité d'un système de perception à distinguer de manière unique tous les lieux d'un environnement. Il apparaît lorsque des situations multiples ne sont pas distinguables des perceptions immédiates et requiert des réponses différentes du système. Or Nolfi (2002b) a montré que des agents réactifs – des agents réagissant directement et exclusivement par rapport à l'état courant de leurs capteurs, donc non pourvus d'une forme de mémoire – pouvaient faire face à ces ambiguités perceptives sur des tâches de catégorisation. Cela rejoint les observations faites sur de nombreux animaux dotés d'un comportement réactif (Tinbergen, 1951; Lorenz, 2003; Balkenius, 1995) et pouvant tout de même faire face à ce problème d'ambiguité des perceptions, omniprésent dans leur environnement naturel. De nombreux chercheurs ont montré que les animaux peuvent faire face à des problèmes d'ambiguité de perceptions en utilisant l'environnement comme mémoire externe (O'Regan, 1992; Hutchins, 1995). Cette mémoire externe fait référence à l'utilisation d'objets ou de caractéristiques de l'environnement. O'Regan et al. (2001) soutient que l'environnement d'un agent naturel dans son ensemble est susceptible de servir comme mémoire externe. L'utilisation de l'environnement comme mémoire externe a été observée sur des agents robotiques (Van Dartel et al., 2005) sur une tâche de catégorisation comportant un problème d'ambiguité des perceptions.

D'autre part, l'émergence d'agents prenant seulement en considération leurs perceptions courantes constitue un minimum local (Nolfi, 2001). Il est en effet difficile de faire émerger via des méthodes évolutionnistes plusieurs capacités en même temps. L'émergence d'une forme de mémoire seule n'apporte pas directement un avantage sélectif supérieur : en plus de posséder une capacité de mémoire, l'agent doit aussi être en mesure d'en extraire des informations sensorimotrices utiles (Nolfi, 2001).

22. Le concept de problème trompeur (*deceptive problems*) a été introduit par Goldberg (1989) comme un défi pour les algorithmes génétiques. Pour ce type de fonctions, les algorithmes génétiques auront tendance à converger vers des solutions sous-optimales.

1.3 Contributions

Afin de faire face à ces difficultés, nous proposons une approche basée sur l'utilisation de différentes pressions de sélection afin d'éviter une convergence prématurée vers des individus au comportement réactif. Nous montrons que pour synthétiser automatiquement des agents capables de réaliser des tâches nécessitant une forme de mémoire interne, il est nécessaire :

- d'utiliser une fitness discrète non dirigiste qui n'introduit pas un gradient pouvant tendre vers un optimum local ;
- de mettre en place des mécanismes de diversité comportementale afin d'explorer efficacement l'espace de recherche ;
- de développer différents objectifs auxiliaires.

Pour valider cette approche nous nous basons sur différents protocoles inspirés des neurosciences.

Ce mémoire est composé de cinq chapitres. Le premier chapitre présente un état de l'art sur la robotique évolutionniste. Il contient une présentation des différents algorithmes évolutionnistes les plus usités à ce jour dans ce domaine, un aperçu des différents codages permettant de représenter et de faire évoluer les réseaux de neurones artificiels ainsi qu'un panel des différents types de tâches abordées en évolution.

Le deuxième chapitre met en exergue la difficulté de faire émerger de la mémoire au sein d'un individu à travers des tâches simples : un individu ayant tendance à exploiter l'environnement au lieu de développer une forme de mémoire interne. Nous proposons dans ce chapitre de s'inspirer des tâches issues des neurosciences afin de mettre au point un protocole expérimental où une forme de mémoire est nécessaire.

Dans le troisième chapitre nous montrons que l'insertion d'a priori au sein de la fitness n'est pas un moyen efficace permettant de faire émerger plus facilement une forme de mémoire. L'utilisation d'un mécanisme d'exploration – comme la diversité comportementale ou la nouveauté – donne quant à elle des résultats satisfaisants.

Deux différentes pressions de sélection sont proposées au sein du quatrième chapitre : l'approche *ProGAb* qui propose de favoriser les individus dotés d'une capacité de généralisation et l'approche *Coher* basée sur la comparaison et la cohérence des comportements des individus. Ces deux méthodes ont montré leur efficacité quant à l'émergence d'une forme de mémoire au sein des individus.

Le dernier chapitre propose des pistes pour le développement d'une forme de mémoire plus aboutie, en abordant notamment la problématique du passage à l'échelle, via l'utilisation d'un codage inspiré des neurosciences.

Chapitre 2

Fondations

Dans la vie il n'y a pas de solutions. Il y a des forces en marche : il faut les créer et les solutions suivent.

Antoine de Saint-Exupéry

Sommaire

2.1 Algorithmes évolutionnistes (AEs)

Les Algorithmes Evolutionnistes (ou évolutionnaires) (Eiben et Smith, 2003) sont des méta-heuristiques stochastiques d'optimisation globale, s'inspirant de la théorie de l'évolution. La souplesse d'utilisation de ces algorithmes pour des fonctions objectifs non régulières, à valeurs vectorielles, et définies sur des espaces de

TABLE 2.1 – Tableau comparatif des termes utilisés en optimisation et en évolution.

Algorithme évolutionniste	Optimisation
individus	solutions candidates
population	ensemble de solutions candidates
fonction de fitness	fonction objectif
génération	itération

recherche non standard (e.g. espaces de listes, de graphes, ...) permet leur utilisation pour des problèmes qui sont pour le moment hors d'atteinte des méthodes plus classiques.

Un problème d'optimisation globale peut se formaliser de cette manière :

$$\text{Trouver } x^* \text{ tel que } f(x*) <= f(x), \forall x \in S$$

avec f comme fonction objectif et S l'espace de recherche.

Définition 1 (Heuristique) *Une heuristique est un algorithme qui permet d'identifier au moins une solution réalisable à un problème d'optimisation, mais sans garantir que cette solution soit optimale (Michalewicz, 2004). Elle permet d'aborder des problèmes ne pouvant être résolus par des techniques de résolution exactes, en particulier les problèmes difficiles d'optimisation combinatoire.*

Définition 2 (Métaheuristique) *Une métaheuristique est une stratégie générale, applicable à un grand nombre de problèmes, à partir de laquelle on peut dériver une heuristique pour un problème particulier.*

2.1.1 Inspiration

Apparue dans les années 30, la théorie synthétique de l'évolution (ou néodarwinisme) (Mayr, 1942) intègre à la théorie darwinienne (Darwin, 1859), la théorie de l'hérédité mendelienne (Mendel, 1865) et la génétique des populations (Haldane, 1932).

Elle donne une explication des mécanismes de différenciation d'individus au sein d'une population ainsi que le principe de transmission de caractéristiques d'individus à leur descendance. Les caractéristiques d'un organisme sont en grande partie codées dans ses gènes, une séquence d'ADN différente pour chaque individu. Les sources de cette diversité sont des mutations pouvant apparaître dans ces gènes, ainsi que des recombinaisons (brassages génétiques) se produisant lors de la reproduction sexuée.

Les algorithmes évolutionnistes sont une abstraction du principe évoqué précédemment pouvant se résumer de la manière suivante :

– Il existe au sein de chaque espèce de nombreuses variations, chaque individu étant différent.

- Les ressources naturelles étant finies, il naît rapidement plus d'être vivants que la nature ne peut en nourrir ; il en résulte une lutte pour l'existence (*struggle for life*) entre chaque espèce.
- Les individus survivants possèdent des caractéristiques qui les rendent plus aptes à survivre. Darwin baptise ce concept : sélection naturelle.
- Les organismes survivants transmettent leurs avantages à leur descendance. L'accumulation au cours des générations des petites différences entre chaque branche généalogique crée de nouvelles espèces de plus en plus aptes à survivre.

2.1.2 Historique

Les algorithmes évolutionnistes (AE) ont des origines diverses : à la fin des années 1950, certains mécanismes d'évolution naturelle sont simulés via des programmes informatiques (Fraser, 1957). Une solution fonctionnelle fut proposée par John Holland en 1975, sous le nom d'algorithme génétique (Holland, 1975) dans le but de modéliser ces mécanismes. Ils ont été utilisés comme optimiseurs par (De Jong, 1975) et popularisés par Goldberg (Goldberg, 1989). Les stratégies d'évolution sont apparues en 1965 (Rechenberg, 1965, 1973; Schwefel, 1981), les programmes évolutionnistes datent du début des années 1960 (Fogel, 1962, 1991), alors que la programmation génétique est apparue plus récemment (Cramer, 1985; Koza, 1992). Ces différents AEs ont aujourd'hui en grande partie fusionné, car, comme l'a démontré Michalewicz (1994), le principe de ces algorithmes peut s'appliquer à toute représentation.

Domaines d'application Le champ d'application des algorithmes évolutionnistes couvre, de nos jours, un spectre très large : d'applications réelles complexes comme le contrôle du flux de *pipelines* de gaz, le *design* de profils d'ailes d'avion, le routage aérien, la conception d'antennes (Hornby et al., 2011; Lohn et al., 2004) et la conception de filtres électroniques (Koza et al., 2003), à des problèmes plus théoriques et combinatoires, comme en théorie des jeux et en modélisation économique ou financière (Coello et Lamont, 2004).

2.1.3 Principes

Les AEs empruntent un vocabulaire quelque peu différent des méthodes d'optimisation classiques (cf. tableau 2.1 pour un comparatif des termes).

Avant de décrire les différents principes de ces algorithmes, il semble nécessaire de définir quelques notions.

Définition 3 (génotype) *Le* génotype *correspond à la partie d'un individu (parent) transmise lors de la création d'un nouvel individu (enfant) et sur laquelle agissent directement les opérateurs de mutation et de croisement.*

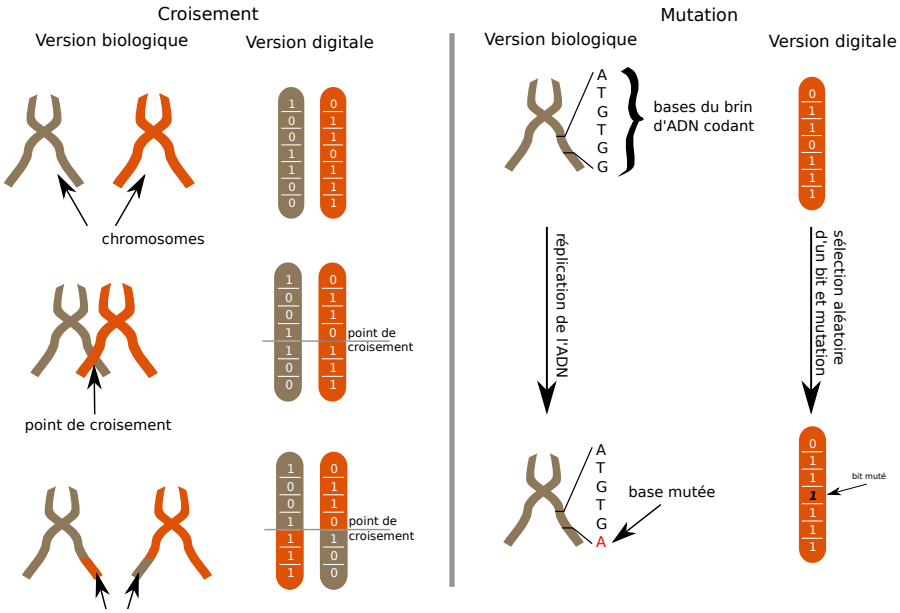


FIGURE 2.1 – Illustration des opérateurs de mutation et de croisement utilisés dans un algorithme évolutionniste, ainsi que des phénomènes biologiques dont ils sont inspirés.

Définition 4 (phénotype) *Le* phénotype *est généré à partir du génotype et correspond aux caractères observables et testables de l'individu.*

Définition 5 (fitness) *La fonction d'adaptation, souvent appelée "fitness" fournit une évaluation de la performance des individus et a une influence sur la sélection et la reproduction des individus.*

Définition 6 (population) *La* population *est l'ensemble des solutions potentielles, chacune définie par un génotype et un phénotype.*

Définition 7 (croisement) *Le* croisement *est un opérateur permettant de générer un nouveau génotype par combinaison de $n > 1$ génotypes parents.*

Définition 8 (mutation) *La* mutation *est un opérateur permettant de générer un nouvel individu à partir d'un individu parent dont certaines parties ont été modifiées aléatoirement.*

Tous les AEs présentent les points communs suivants : ils manipulent une population d'individus, un individu peut subir des modifications aléatoires, un processus de sélection permet de déterminer quels individus conserver et quels individus rejeter.

Un individu se caractérise par son génotype qui peut avoir différentes représentations : "binaire" ($G = 0, 1^N$) , "réelle" ($G = [0, 1]^N$ ou $\mathbb{R}^N$), ou sous forme de graphe.

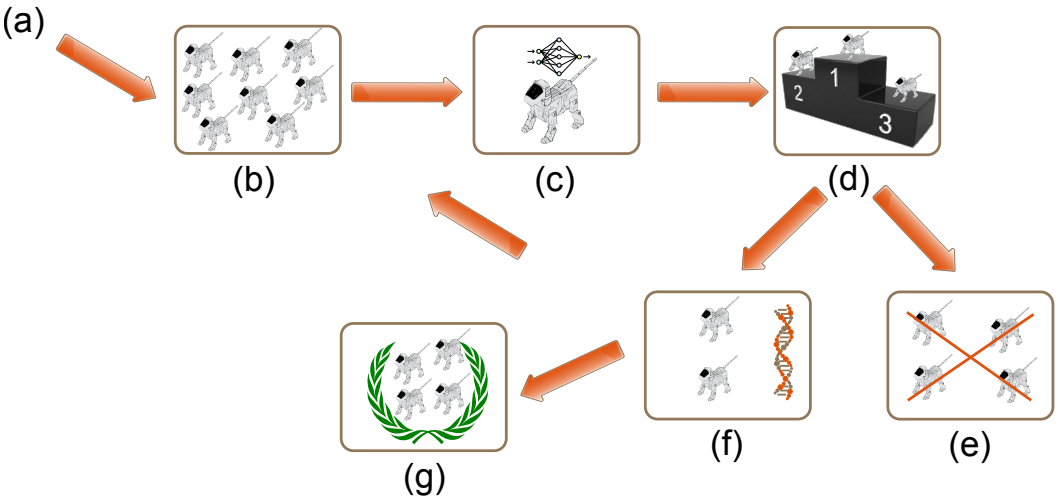


FIGURE 2.2 – Principe général d'un algorithme évolutionniste. (a) Une population de solutions candidates est générée aléatoirement ; (b) on obtient une population (c) la performance (la fitness) de chaque solution est évaluée ; (d) les solutions sont classées en fonction de leur fitness ; (f) les solutions les mieux classées sont sélectionnées pour générer une descendance par croisement puis mutation ; (e) la population parente est détruite ; le cycle recommence en (b). (g) Après n générations, une population de solutions a été trouvée.

C'est dans cet espace génotypique que sont appliqués les différents opérateurs de variabilité : les mécanismes de recombinaisons et de mutations.

Les algorithmes évolutionnistes utilisent principalement la mutation ponctuelle comme illustré sur la figure 2.1. Par exemple, en supposant que le génotype soit une chaîne de bits, on peut définir pour chaque bit une probabilité d'inversion : le génotype de l'individu muté peut alors différer de celui de l'individu original de plusieurs bits.

L'opérateur de croisement dépend essentiellement du codage utilisé. Certains travaux semblent indiquer que l'utilisation d'un opérateur de croisement rend l'algorithme plus efficace vis-à-vis de l'optimisation (Jansen et Wegener, 1999), mais il demeure parfois difficile de définir un opérateur de croisement qui ait un sens pour le codage utilisé, par exemple dans le cas d'évolution de graphes (Eiben et Schippers, 1998; Stanley et Miikkulainen, 2002). Certains travaux se limitent à ne définir que des opérateurs de mutation (e.g. (Bäck et al., 1991; Mouret et Doncieux, 2012)).

Le génotype est ensuite décodé en phénotype afin d'être évalué. Par exemple, un graphe, représentant le génotype, peut être converti en un réseau de neurones, représentant le phénotype.

Voici le pseudo-code général d'un algorithme évolutionniste (cf. figure 2.2 pour un schéma récapitulatif) :

```
construction et évaluation d'une  population initiale ;
Jusqu'à atteindre un  critère d'arrêt :
   sélection d'une partie de la population,
   reproduction des individus sélectionnés,
   mutation de la descendance,
   évaluation du degré d'adaptation de chaque individu,
   remplacement de la population initiale par une nouvelle population.
```

Sélection L'un des points clés d'un algorithme évolutionniste se situe dans le compromis entre explorer l'espace de recherche, afin d'éviter de stagner dans un optimum local, et exploiter les meilleurs individus obtenus, afin d'atteindre de meilleures valeurs aux alentours. Trop d'exploitation entraîne une convergence vers un optimum local, ou convergence prématurée, alors que trop d'exploration entraîne la non-convergence de l'algorithme.

Différentes méthodes de sélection des individus existent selon les algorithmes :

- La sélection par tournoi (Brindle, 1981; Miller et Goldberg, 1995) : consiste à effectuer un nombre arbitraire de tournois entre n individus ($n \geq 2$) et de choisir le ou les plus performants. Plus n augmente, plus ce type de sélection induit une pression importante, étant donné que le tournoi a plus de chance d'impliquer les individus de la population ayant les plus hautes valeurs de fitness.
- La sélection par roulette (Baker, 1987) : donne à chaque individu une probabilité d'être sélectionnée proportionnelle à sa performance.
- La sélection par rang : dans ce cas, la population est d'abord triée par fitness. Chaque individu se voit associé un rang en fonction de sa position. La sélection se fait ensuite de même manière que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation.

Remplacement Certains parents sont remplacés par certains des enfants en fonction de leurs performances respectives. Les stratégies d'évolution (ES) n'utilisent que le remplacement, et non la sélection, pour biaiser la recherche. Dans les ES, le remplacement est déterministe et peut prendre deux formes, notées (μ, λ)-ES et $(\mu + \lambda)$-ES. Dans le remplacement (μ, λ)-ES, les μ meilleurs individus parmi les λ enfants constituent la population suivante P^{t+1} . P^t n'est pas conservée. Il est ainsi possible que le meilleur individu de la population $t+1$ soit moins bon que le meilleur individu de la population t. Le remplacement $(\mu + \lambda)$-ES construit P^{t+1} en gardant les μ meilleurs individus de P^t et des λ enfants. Ainsi, si les meilleurs individus de P^t restent compétitifs par rapport aux enfants, ils seront conservés (c'est la propriété d'"élitisme").

Il existe de nombreux algorithmes évolutionnistes efficaces, comme la stratégie d'évolution avec adaptation de matrice de covariance (CMA-ES) (Hansen et Ostermeier, 2001; Hansen et Koumoutsakos, 2003) qui a été reconnue comme étant une des méthodes les plus performantes dans le cadre d'optimisation continue *black box*

(Hansen et al., 2010).

Cependant, de nombreux problèmes ne peuvent être réduits à un seul objectif. Prenons le cas d'un aspirateur autonome. Celui-ci doit posséder une grande capacité d'aspiration, une bonne autonomie lui permettant d'explorer efficacement différentes pièces, tout en ayant un prix d'achat raisonnable. Cela nécessite d'optimiser plusieurs objectifs simultanément pouvant être contradictoires. Il y a donc plusieurs compromis plutôt qu'une solution optimale unique.

2.1.4 Algorithmes multi-objectifs

Pareto (1897) a formulé le concept suivant : dans un problème multiobjectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères. Cet équilibre a été défini comme l'optimum de Pareto.

Définition 9 (dominance de Pareto) *Une solution x_1 est dite dominante par rapport à la solution x_2 si les conditions 1 et 2 sont vraies :*

1. *la solution x_1 n'est pas plus mauvaise que x_2 sur tous les objectifs ;*
2. *la solution x_1 est strictement meilleure que x_2 pour au moins un objectif.*

Un point x_i est dit Pareto-optimal s'il n'est dominé par aucun autre point.

Définition 10 (Front de Pareto) *Le front de Pareto est l'ensemble des solutions non dominées de l'espace de recherche.*

Un front de Pareto est représenté sur la figure 2.3.

Nous pouvons distinguer 3 grandes classes de méthodes pour résoudre un problème multiobjectif :

- les méthodes agrégées ;
- les méthodes non-agrégées non basées sur l'optimum de Pareto ;
- les méthodes basées sur l'optimum de Pareto.

2.1.4.1 Les méthodes agrégées

Dans cette classe de méthodes, les différents objectifs sont regroupés en un seul pour être optimisés.

Somme pondérée Cette méthode consiste à additionner tous les objectifs en leur affectant un coefficient de poids. Ce coefficient représente l'importance relative que le décideur attribue à l'objectif.

$$minimise \sum_{i=1}^{k} w_i f_i(x) \text{ avec } w_i \leq 0$$

avec w_i représentant le poids affecté au critère i et $\sum_{i=1}^{k} w_i = 1$.

Cette méthode peut, en faisant varier les valeurs du vecteur poids w, retrouver l'ensemble de solutions supportées si le domaine réalisable est convexe. Cependant,

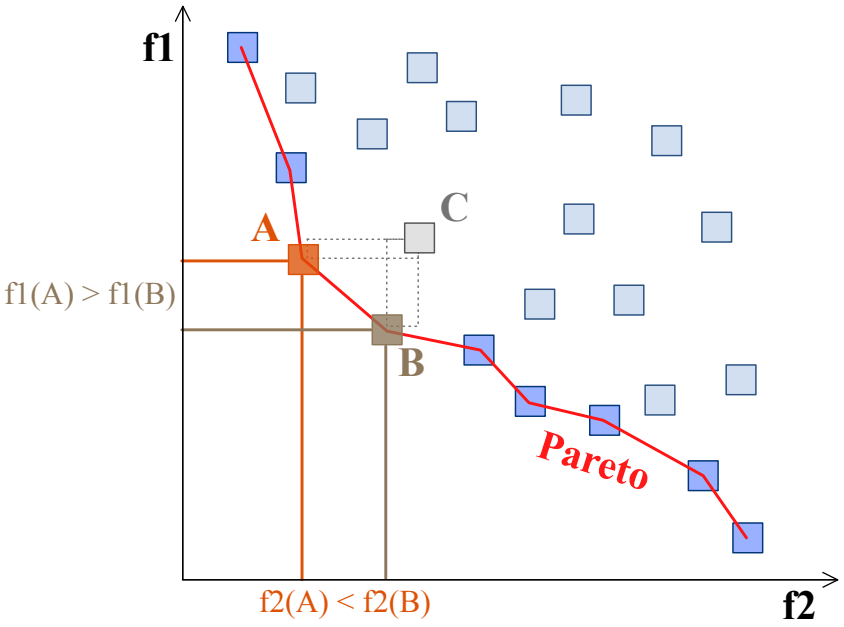


FIGURE 2.3 – Sur la figure suivante, on cherche à minimiser les deux objectifs f1 et f2. Les solutions non dominées sont les points situés sur le front de Pareto, indiqué en rouge. Le point C n'est pas sur le front de Pareto parce qu'il est dominé par les points A et B (graphique issu de Wikipedia).

elle ne permet pas de trouver les solutions enfermées dans une concavité (les solutions non supportées). Les résultats obtenus avec de telles méthodes dépendent fortement des paramètres choisis pour le vecteur de poids w. Les poids w_i doivent également être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate. Une approche généralement utilisée consiste à répéter l'exécution de l'algorithme avec des vecteurs poids différents.

Jin et al. (2001) ont proposé une agrégation dynamique des poids et Kim et De Weck (2006) ont présenté une méthode de somme pondérée adaptative.

Goal programming Dans cette méthode, le décideur fixe un but T_i à atteindre pour chaque objectif f_i (Charnes et Cooper, 1961). Ces valeurs sont ajoutées au problème comme des contraintes. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre.

$$min \sum_{i=1}^{k} |f_i(x) - T_i| \text{ avec } x \in F$$

T_i représente la valeur à atteindre pour le i^{eme} objectif.

Le min-max Il minimise le maximum de l'écart relatif entre un objectif et son but associé par le décideur (Coello et al., 1995).

$$\min \max_i \left(\frac{f_i(x) - T_i}{T_i} \right) \text{ avec } i, \ldots, k$$

avec T_i le but à atteindre pour le i^{eme} objectif. avec w_i le poids à un objectif.

Atteindre un point-cible Dans cette approche (Charnes et Cooper, 1957; Ijiri, 1965; Duckstein, 1981) le décideur spécifie l'ensemble des buts T_i qu'il souhaite atteindre et les poids associés w_i. La solution optimale est trouvée en résolvant le problème suivant :

$$\text{minimiser } \alpha \text{ tel que } T_i + \alpha.w_i >= f_i(x)$$

Les objectifs T_i représentent le point de départ de la recherche dans l'espace et les poids w_i indiquent la direction de recherche dans l'espace.

La méthode par ε-contraintes Méthode basée sur la minimisation d'un objectif f_i en considérant que les autres objectifs f_j avec $j \neq i$ doivent être inférieurs à une valeur ε_j. En général, l'objectif choisi est celui que le décideur souhaite optimiser en priorité.

$$\text{minimiser } f_i(x) \text{ avec } f_j(x) \leq \varepsilon_j, \forall j \neq i$$

De cette manière, un problème simple objectif sous contraintes peut être résolu. Le décideur peut ensuite réitérer ce processus sur un objectif différent jusqu'à ce qu'il trouve une solution satisfaisante.

Critiques Il existe plusieurs inconvénients quant à l'utilisation de méthodes agrégées : la difficulté de déterminer les différents paramètres ; elles nécessitent de nombreuses connaissances a priori, notamment lors de l'affectation d'un coefficient de poids à chaque objectif ; il est de plus nécessaire d'effectuer de nombreux tests pour déterminer l'influence de chaque objectif.

2.1.4.2 Méthodes non-agrégées et non Pareto

Vector Evaluated Genetic Algorithm (VEGA) Première extension d'un algorithme génétique simple pour la résolution d'un problème multiobjectif (Schaffer, 1984). La différence avec un algorithme génétique simple se situe au niveau de la sélection. Pour k objectifs et une population de n individus, une sélection de n/k individus est effectuée pour chaque objectif. k sous-populations seront créées contenant chacune les n/k meilleurs individus pour un objectif particulier. Les k sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille n.

La méthode lexicographique (Fourman, 1985) Les objectifs sont préalablement rangés par ordre d'importance par le décideur (Fourman, 1985). L'optimum est ensuite obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite.

2.1.4.3 Les méthodes basées sur l'optimum de Pareto

La première utilisation de la dominance au sens de Pareto remonte à (Goldberg, 1989) pour résoudre des problèmes proposés par (Schaffer, 1984).

Définition 11 (Convergence) *La convergence permet de mesurer à quelle distance les solutions trouvées sont du vrai front de Pareto.*

Définition 12 (Dispersion) *La distribution indique comment les solutions sont bien réparties sur le vrai front de Pareto (ou son approximation).*

Résoudre un problème multi-objectifs consiste à converger le plus rapidement possible vers le front de Pareto (convergence), tout en gardant une bonne distribution des solutions sur le front de Pareto (dispersion).

Méthodes de première génération

Multiple Objective Genetic Algorithm (MOGA) Dans cette méthode (Fonseca et Fleming, 1993), chaque individu de la population est rangé en fonction du nombre d'individus qui le dominent. Par la suite une fonction de notation permet de prendre en compte le rang de l'individu et le nombre d'individus ayant le même rang. Les individus non dominés sont de rang 1. L'évaluation de la fitness de chaque individu s'effectue en calculant le rang de l'individu puis en affectant la fitness de chaque individu par application d'une fonction de changement d'échelle sur la valeur de son rang. Cette fonction est en général linéaire.

L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum ce qui n'est pas satisfaisant. Pour éviter cette dérive, les auteurs utilisent une fonction de partage. L'objectif est de répartir la population sur l'ensemble du front de Pareto.

La technique de partage agit sur l'espace des objectifs. Cela suppose que deux actions qui ont le même résultat dans l'espace des objectifs ne pourront pas être présentes dans la population. Cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Cependant les performances dépendent de la valeur du paramètre utilisé dans la technique de partage.

Non dominated Sorting Genetic Algorithm (NSGA) Dans NSGA (Srinivas et Deb, 1994), le calcul de la fitness s'effectue en séparant la population en plusieurs groupes en fonction du degré de domination au sens de Pareto de chaque individu. L'algorithme se déroule ensuite comme un algorithme génétique classique. La sélection est basée sur le reste stochastique mais peut être utilisée avec d'autres heuristiques de sélection (tournoi, roulette pipée, etc.).

Niched Pareto Genetic Algorithm (NPGA) Cette méthode (Horn et al., 1994) utilise un tournoi basé sur la notion de dominance de Pareto. La comparaison se fait sur deux individus pris au hasard avec une sous-population de petite taille également choisie au hasard. Si un seul de ces deux individus domine la sous-population, il est alors positionné dans la population suivante. Dans les autres cas, une fonction de sharing est appliquée pour sélectionner l'individu.

Synthèse Du fait de la non-conservation des individus Pareto-optimaux de génération en génération dans ces méthodes dites non-élitistes, la convergence vers le front de Pareto est lente. De plus, elles maintiennent difficilement de la diversité sur ce front de Pareto. Pour contrecarrer ces limitations, des méthodes dites "élitistes" se sont développées.

Méthodes élitistes

Strength Pareto Evolutionary Algorithm (SPEA II) Cette méthode (Zitzler et al., 2001) utilise le concept de Pareto pour comparer les solutions. Une des caractéristiques de SPEA II est de maintenir un ensemble de solutions Pareto-optimales dans une archive. La fitness de chaque individu est calculée en fonction des solutions stockées dans celle-ci. De plus, toutes les solutions de l'archive participent à la sélection. Afin de maintenir une taille d'archive raisonnable, une méthode de clustering est utilisée visant à garder seulement les solutions les plus représentatives. Une nouvelle méthode de niche, basée sur Pareto, est utilisée afin de préserver la diversité. L'avantage essentiel est qu'elle n'exige pas de réglage de paramètres de sharing.

Pareto Archived Evolution Strategy (PAES) L'algorithme PAES (Knowles et Corne, 2000) est inspiré d'une stratégie d'évolution (1+1) (Rechenberg, 1973). Il n'est pas basé sur une population et n'utilise qu'un seul individu à la fois pour la recherche des solutions. Une population annexe de taille déterminée est utilisée afin de permettre de stocker les solutions temporairement Pareto-optimales. Si une nouvelle solution est non-dominée par un membre de l'archive, il est inclus dans l'archive, supprimant à cette occasion tous les membres qu'il domine. Si l'archive excède une taille maximum, l'acceptation d'une nouvelle solution est décidée par une mesure de densité basée sur une division en grille de l'espace des objectifs.

Pareto Envelope based Selection Algorithm (PESA) PESA (Corne et al., 2000) utilise une petite population interne et une population externe plus large. Une division en grille de l'espace des phénotypes est utilisée pour maintenir une diversité (application d'une mesure de densité) durant le processus. De plus, cette mesure de densité est utilisée pour permettre à des solutions d'être retenues dans une archive externe de la même manière que dans PAES (Knowles et Corne,

2000). La différence entre PESA et PESA-II (Corne et al., 2001), se situe au niveau de la sélection. La sélection se fait d'abord sur une zone. Un individu est ensuite sélectionné dans cette zone. Le but de cette approche étant de réduire le coût computationnel associé au classement de Pareto.

Non-dominated Sorting Genetic Algorithm II (NSGA-II) Outre la volonté d'assurer une bonne convergence vers le front de Pareto, ainsi qu'une bonne dispersion des solutions dans l'espace des objectifs, l'objectif de NSGA II (Deb et al., 2002) est de proposer un algorithme d'optimisation multi-objectifs rapide. Cet algorithme sera utilisé au cours des différentes expériences au sein de cette thèse. Nous allons le décrire plus en détail.

Une des caractéristiques de NSGA-II est d'introduire un classement des individus en plusieurs fronts successifs suivant la relation de dominance et de proposer également de sélectionner les solutions en fonction d'un critère de performance et de la densité de solutions dans leur voisinage.

À l'initialisation de l'algorithme, on génère une population de départ taille M.

Pour chaque génération de l'algorithme, les opérations suivantes sont effectuées :

- Grouper la population parent P_t et enfant Q_t de la génération précédente, chacune de taille M, en une population $R_t = P_t \cup Q_t$. Réaliser un rangement de la population R_t par fronts $\mathcal{F}_i$ successifs avec $i = 0$ le front de Pareto.
- Définir une nouvelle population parente pour la génération suivante $P_{t+1} = \emptyset$ et un indice $i = 0$ définissant le front de Pareto actif. Tant que $|P_{t+1}| + |\mathcal{F}_i| < M$, réaliser $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ et $i = i + 1$.
- Utiliser la procédure de classement par densité de peuplement (voir ci-dessous) au sein du front $\mathcal{F}_i$ et inclure les $M - |P_{t+1}|$ solutions les plus isolées à P_{t+1}. On a désormais $|P_{t+1}| = M$.
- Générer par sélection par tournoi, mutation et croisement, la population fille Q_{t+1} de taille M à partir de P_{t+1}.

La procédure de classement est illustrée par la figure 2.4 et l'algorithme de calcul de la distance de densité est fourni ci-dessous.

```
l=|I|
pour chaque i de F:
  définir I[i]=0
pour chaque objectif m de M:
  classer F par rapport à l'objectif m
  définir la distance des extrémités du front (i=0 et i=l) à la valeur max
  pour i = 1 à l-1:
    I[i]=I[i]+(Vm[i+1]-Vm[i-1])
```

avec i un individu du front de Pareto F, $I[i]$ la mesure de densité pour l'individu i et $Vm[i]$ correspond à la valeur de l'objectif m pour le point i.

Le processus de sélection par distance au sein d'un même front permet de favoriser les individus les plus éloignés les uns des autres tant qu'ils appartiennent à

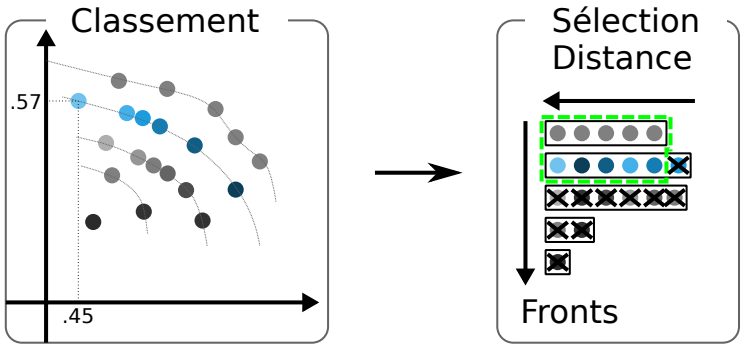


FIGURE 2.4 – schéma expliquant le processus de sélection de NSGA-II. Le classement de la population se fait par fronts successifs et en fonction de la distance d'un individu à ses plus proches voisins sur un même front. Lors de la sélection, on conserve les meilleurs fronts et, lorsque la sélection se fait parmi les individus d'un même front, on sélectionne ceux qui sont les plus isolés sur le front.

un même front. L'objectif est d'obtenir, à la convergence de l'algorithme, la quasi totalité de la population sur un seul front : le front des solutions Pareto optimales accessibles du problème. Une fois cette situation atteinte, la sélection se fait uniquement en fonction de la distance entre solutions, en vue de couvrir au mieux ce front.

Finalement, la sélection des individus pour la création d'une population fille Q_{t+1} à partir de la population P_{t+1} se fait en utilisant un opérateur de tournoi basé sur la distance ("Crowded Tournament Selection Operator") fourni dans la définition 13. Cet opérateur est appliqué entre n individus tirés au hasard dans la population P_{t+1}.

Définition 13 *Un individu i gagne un tournoi contre un individu j si et seulement si une des deux conditions suivantes est vraie :*

- *i a un meilleur rang que j (l'indice du front de Pareto de i est inférieur à celui de j).*
- *Si i et j appartiennent au même front mais si i est plus isolé que j sur le front de Pareto (sa mesure de distance est plus grande).*

L'implémentation de l'algorithme, proposée initialement par Deb, a une complexité en $O(MN^2)$. Une implémentation de l'algorithme utilisant des arbres binaires pour le classement des solutions par fronts (Jensen, 2003) permet de réduire cette complexité à $O(Nlog^{M-1}N)$, mais son implémentation est délicate si le nombre d'objectifs est supérieur à 2.

ε-Multiobjective Evolutionary Algorithm, (ε-MOEA) L'ε-MOEA (Laumanns et al., 2002; Deb et al., 2005) repose sur une version modifiée de la relation de dominance : l'ε-dominance (Laumanns et al., 2002; Gro\csan et Oltean,

2004; Deb et al., 2005). Soit $\boldsymbol{\varepsilon}$ un vecteur quelconque de n réels strictement positifs, n étant le nombre d'objectifs optimisés, on dit que la solution $\boldsymbol{x_1}$ ε-domine la solution $\boldsymbol{x_2}$ si :

$$\begin{cases} \forall i \in [1,\ n],\ f_i(\boldsymbol{x_1}) + \varepsilon_i \geq f_i(\boldsymbol{x_2}) \\ \exists i \in [1,\ n],\ f_i(\boldsymbol{x_1}) + \varepsilon_i > f_i(\boldsymbol{x_2}) \end{cases}$$

Avec l'ε-dominance, une solution domine toutes les solutions qui ne sont pas suffisamment meilleures qu'elle, sur au moins un objectif, c'est-à-dire si la différence sur un objectif ne dépasse pas la valeur ε_i correspondante. A noter que le front de Pareto ε-approché défini par la relation d'ε-dominance n'est pas unique.

La relation d'ε-dominance possède de bonnes propriétés théoriques (voir par exemple (Laumanns et al., 2002)). Elle assure notamment une bonne convergence et une bonne dispersion des solutions du front de Pareto ε-approché vis-à-vis du front de Pareto "exact" défini par la relation de dominance. Le front de Pareto ε-approché a également une taille bornée en fonction des valeurs d'ε utilisées.

L'algorithme ε-MOEA (Laumanns et al., 2002; Deb et al., 2005) utilise cette relation d'ε-dominance en plus de la relation de dominance classique. Il repose sur deux ensembles d'individus, une population et une archive contenant les solutions non-dominées trouvées. A chaque étape, deux parents sont sélectionnés, un dans la population et un dans l'archive, et sont utilisés pour générer deux enfants. Ces nouveaux individus sont ensuite intégrés à la population selon la relation classique de dominance et ajoutés à l'archive selon la relation d'ε-dominance. L'idée principale est d'assurer une bonne dispersion des solutions non-dominées dans l'espace des objectifs, en ne gardant que quelques solutions représentatives le long du front dans l'archive. Suivant les valeurs ε_i fixées pour chaque objectif, l'utilisateur peut ainsi intégrer de l'information dans le processus d'optimisation en indiquant l'écart sur chaque objectif à partir duquel deux solutions seront effectivement différentes de son point de vue (Laumanns et al., 2002).

L'algorithme ε-MOEA permet ainsi d'obtenir de bonnes performances en termes de convergence et de dispersion des solutions tout en ayant un coût computationnel moins élevé que SPEA et NSGA-II. Néanmoins, le choix des valeurs ε_i a une influence évidente sur l'approximation finale du front de Pareto. D'autre part, la densité de solutions peut également être variable le long du front et l'utilisation d'une constante ε_i par objectif risque d'entraîner un sous-échantillonnage de certaines zones par rapport à d'autres.

2.1.4.4 Algorithmes utilisant l'indicateur d'hypervolume

L'hypervolume (Zitzler et Thiele, 1999; Knowles, 2002) est un indicateur permettant de mesurer et comparer la qualité des solutions finales dans des algorithmes à base de population. Cet indicateur représente le volume d'espace à N dimensions dominé par une ou plusieurs solutions à un problème à N objectifs (voir figure 2.5). Il est calculé relativement à un point arbitraire facilement accessible et dominé. L'hypervolume couvert entre ce point et les points positionnés sur le front

de Pareto de l'expérience représente l'hypervolume de ces solutions. L'ajout d'une solution non-dominée au front ajoute automatiquement de l'espace à l'hypervolume de ce front. L'indicateur d'hypervolume a été originellement proposé et employé comme mesure de performance pour comparer différents MOEA (Zitzler et Thiele, 1999). Il a par la suite été intégré dans le processus d'optimisation (Knowles, 2002). Dans ce cas, il décrivait une stratégie pour maintenir une archive séparée et bornée de solutions dominées basées sur cet indicateur.

L'hypervolume est le seul indicateur connu pour être strictement monotone en ce qui concerne la dominance de Pareto : à chaque fois qu'une approximation du front de Pareto domine entièrement un autre, alors la valeur de l'indicateur d'hypervolume sera plus grand (Bader et Zitzler, 2011).

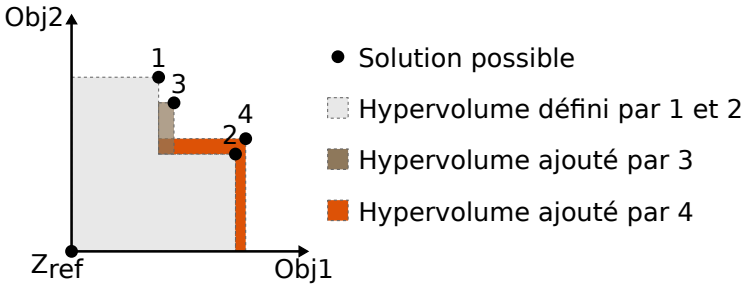


FIGURE 2.5 – illustration du concept d'hypervolume : l'hypervolume, représenté ici pour 2 objectifs correspond à la surface dominée par les solutions 1 et 2 par rapport au point de référence Z_{ref}. L'ajout de solutions, si elles sont non dominées (3 et 4) augmente la surface de l'hypervolume.

De nombreux MOEA intègrent l'indicateur d'hypervolume durant le processus d'optimisation. Parmi eux nous pouvons noter : ESP (Huband et al., 2003), IBEA (Zitzler et Künzli, 2004) (en combinaison avec l'indicateur ε), SIBEA (Brockhoff et Zitzler, 2007), SMS-EMOA (Beume et al., 2007) ou HypE (Bader et Zitzler, 2011).

Une version mutli-objectifs de CMA-ES, MO-CMA-ES (Igel et al., 2007) a été développée. Le principe est le suivant : une population d'individus, qui adaptent leur stratégie de recherche comme dans CMA-ES, est maintenue. Pour la sélection, Les individus sont triés en fonction de leur niveau de non-dominance reprenant le même principe que dans NSGA-II. Afin de classer les individus qui ont le même niveau de dominance, deux critères sont considérés : la distance par densité de peuplement et l'hypervolume ajouté par l'individu.

Différentes implémentations ont été proposées pour tenter de réduire ce coût computationnel comme (Fonseca et al., 2006) basée sur une méthode "*dimension-sweep*" avec une complexité de $O(N^{n-2} \log N)$, (Beume et Rudolph, 2006) $O(N \log N + N^{n/2})$, ou encore (Yang et Ding, 2007) qui décrit un algorithme avec une complexité de $O((n/2)^N)$.

D'autres méthodes ont été envisagées, comme automatiquement réduire le nombre d'objectifs (Brockhoff et Zitzler, 2007) ou approximer les valeurs de l'indicateur d'hypervolume en utilisant des méthodes de Monte Carlo (Everson et Field-

TABLE 2.2 – Tableau récapitulatif de différents algorithmes évolutionnistes.

Nom	MO	Archive	Complexité*	Complexité après opt.*
CMA-ES (Hansen et Koumoutsakos, 2003)			$O(N^3)$	$O(N^2)^{\ddagger}$
Hypervolume (Knowles, 2002)	*		$O(N^{n-1})$	$O((n/2)^N)^{\text{II}}$
MO-CMA-ES (Igel et al., 2007)	*		$O(N^{n-1})$	
SMS-EMOA (Beume et al., 2007)	*			
ε-MOEA (Laumanns et al., 2002)	*	*		
NSGA-II (Deb et al., 2002)	*		$O(MN^2)$	$O(N \log^{M-1} N)^{\dagger}$
SPEA (Zitzler et Thiele, 1999)	*	*	$O(M(N+A)^2)$	-
SPEA2 (Zitzler et al., 2001) $M = 2$	*	*	$O((N+A)^2)$	$O(N \log^{M-1} N)^{\dagger}$
SPEA2 (Zitzler et al., 2001) $M \geq 3$	*	*	$O(M(N+A)^2)$	-
MOGA (Fonseca et Fleming, 1993)	*		$O(MN^2)$	-
PAES (Knowles et Corne, 2000)	*	*	$O(MNA)$	$O(N \log^{M-1} A \log\log A)^{\dagger}$
PESA (Corne et al., 2000)	*	*	$O(MNA)$	$O(N \log^{M-1} A \log\log A)^{\dagger}$
PESA-II (Corne et al., 2001)	*	*	$O(MNA)$	$O(N \log^{M-1} A \log\log A)^{\dagger}$

* M représente le nombre d'objectifs, N la taille de population, A l'archive
† proposé par (Jensen, 2003)
‡ si la matrice de covariance est calculée tous les $N/10$ générations (Hansen et Ostermeier, 2001)
II proposé par (Yang et Ding, 2007)

send, 2002; Bader et Zitzler, 2011). L'idée principale dans le deuxième cas repose sur le fait que ce ne sont pas les valeurs de l'indicateur d'hypervolume qui sont importantes, mais plutôt le classement induit par cet indicateur.

2.1.5 Synthèse

Nous avons pu voir que dans le cadre d'optimisation mono-objectif continue *black box*, CMA-ES (voir le tableau 2.2 pour un récapitulatif de différents algorithmes évolutionnistes) est un des algorithmes les plus performant (Hansen et al., 2010). Dans le cadre multi-objectifs, l'utilisation d'une méthode élitiste est nécessaire afin de garantir un taux de convergence rapide, ce qui élimine tous les algorithmes multi-objectifs de première génération. D'après Wagner (Wagner et al., 2007), NSGA-II et SPEA2 ont des difficultés quand le nombre d'objectifs augmente ($n > 3$), alors que ε-MOEA gère bien l'optimisation massivement multiobjective (Wagner et al., 2007). L'utilisation d'un indicateur d'hypervolume (Zitzler et Thiele, 1999; Knowles, 2002) semble prometteuse mais sa complexité d'exécution reste trop importante pour utiliser efficacement dans le cadre d'optimisation de contrôleur, qui nécessitent un grand nombre d'évaluation. D'autre part, l'utilisation d'une archive dans un algorithme évolutionniste, comme dans ε-MOEA, SPEA, SPEA2, PESA ou PESA-II, exclu la possibilité de traiter des objectifs dynamiques, c'est-à-dire dont la valeur change au cours du temps ou en fonction des individus présents dans la population. Un objectif devient dynamique notamment dans le cas où l'on désire maintenir une certaine diversité dans la population d'individus via un objectif de diversité spécifique. Cette approche sera détaillée au chapitre 4.

Notre choix s'est donc porté sur NSGA-II, car il combine différents atouts :

– c'est un algorithme multi-objectifs où l'on observe des taux de convergence rapides ;

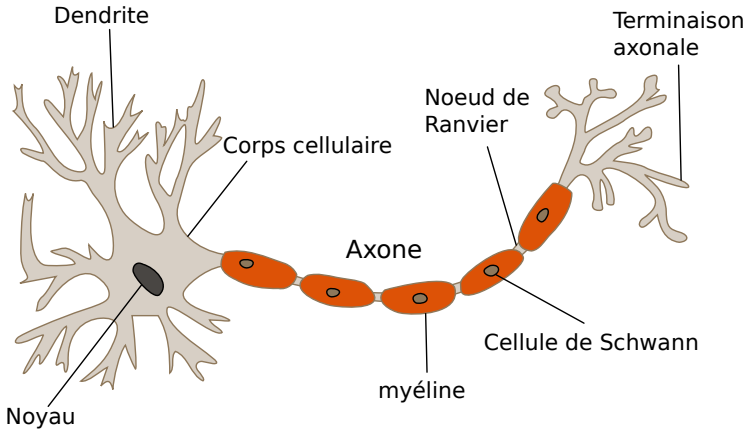


FIGURE 2.6 – schéma d'un neurone biologique

– il gère les objectifs dynamiques (dont la valeur change au cours du temps ou en fonction des individus présents dans la population) ;
– c'est un des algorithmes évolutionnistes multi-objectifs les plus utilisés actuellement Coello et Lamont (2004); Coello (2006) dans des domaines variés (par exemple en chimie Nandasana et al. (2003), en médecine Lahanas et al. (2003, 2004), en écologie Bekele et Nicklow (2007) ou encore en ingénierie Ray (2004); Reed et al. (2007)).

2.2 Évolution de réseaux de neurones

2.2.1 Modèles de neurones

2.2.1.1 Neurone biologique

Le cerveau humain contient près de 86 milliards de neurones (Azevedo et al., 2009), et il existe environ 200 types de neurones. Dans un neurone nous pouvons distinguer trois régions principales (cf. figure 2.6) : le corps cellulaire qui contient le noyau du neurone ainsi que la machine biochimique nécessaire à la synthèse d'enzymes ; les dendrites, qui se divisent comme les branches d'un arbre, recueillent l'information d'autres neurones et l'acheminent vers le corps de la cellule ; l'axone, généralement très long et unique, il conduit l'information du corps cellulaire vers d'autres neurones avec qui il fait des connexions appelées synapses.

Au niveau des synapses, la transmission de l'information se fait par l'intermédiaire de molécules chimiques : les neuromédiateurs. Quand un signal électrique arrive au niveau de la synapse, il provoque l'émission de neuromédiateurs excitateurs ou inhibiteurs qui vont se fixer sur les récepteurs dendritiques de l'autre côté de l'espace inter-synaptique. Lorsque suffisamment de molécules excitatrices se sont fixées, un signal électrique est émis dans les dendrites. Le neurone compare alors la somme de tous ces signaux à un seuil. Si la somme excède ce seuil, le neurone émet un signal électrique (émission d'un potentiel d'action) le long de son axone. Sinon,

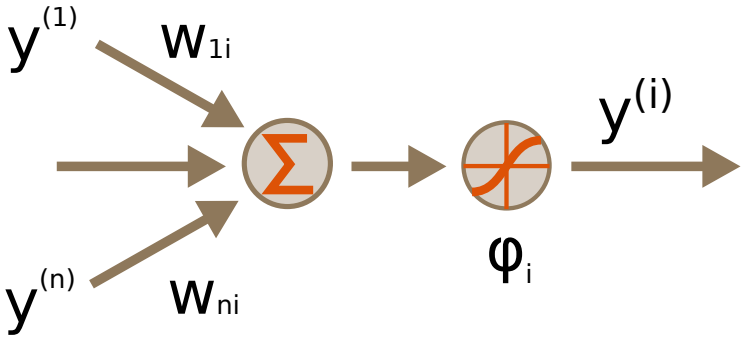


FIGURE 2.7 – schéma d'un neurone générique. Une première fonction combine les différentes entrées (le plus souvent une somme) alors qu'une seconde fonction transforme ce résultat pour générer la valeur de sortie du neurone.

il reste inactif et ne stimule pas les neurones auxquels il est connecté.

2.2.1.2 Neurone artificiel

Un neurone artificiel modélise plus ou moins fidèlement le fonctionnement d'un neurone biologique. Il peut être défini comme une fonction algébrique non linéaire, paramétrée, à valeurs bornées, de variables réelles appelées entrée. On identifie trois éléments de bases : un ensemble de poids de connexions, un seuil, et une fonction d'activation (Haykin, 1998; Floreano et Mattiussi, 2008).

Voici une petite revue de quelques modèles de neurones du plus simplifié au plus biomimétique.

Mc Culloch's and Pitts C'est la première modélisation hautement simplifiée d'un neurone décrit par McCulloch et Pitts (1943). Cette modélisation décrit une unité logique de seuil (Threshold Logic Unit (TLU)) qui applique une fonction de transfert φ aux entrées du neurone.

$$y^{(i)} = \varphi(\sum_{j=1}^{n} w_{j,i} y^{(j)}) \tag{2.1}$$

Avec $y^{(i)}$ la valeur de sortie du neurone, φ la fonction de transfert, $y^{(1)}, y^{(n)}$ les différentes entrées et $w_{j,i}$ les poids associés.

Dans la formulation originelle McCulloch et Pitts, les neurones avaient une sortie binaire (0 ou 1), mais deux des fonctions les plus utilisées sont la fonction seuil et la fonction sigmoïde :

$$\varphi(x) = \frac{1}{1 + e^{-\lambda(y^{(i)} + \text{bias})}} \tag{2.2}$$

avec *bias* correspondant au biais et λ au coefficient de pente.

Neurones à fonction radiale (RBF) Les neurones à fonction radiale (Radial Basis Function ou RBF) (Buhmann, 2003) représentent une autre variété de neurones artificiels dont l'intensité de la réponse est inversement proportionnelle à la distance entre les entrées et un point précis dans l'espace de ces entrées. Les réseaux de fonctions à base radiale sont notamment utilisés dans la classification, l'approximation et la reconnaissance de parole. Leur but est d'approximer un comportement désiré par une collection de fonction, appelées noyaux. Un noyau est caractérisé par un centre Ci et des champs récepteurs r.

Intégrateur à fuite L'intégrateur à fuite (Leaky integrator, LI) possède une dynamique du premier ordre [1] avec une non-linéarité liée à la fonction de décision (souvent une sigmoïde) (Floreano et Mattiussi, 2008). Le terme intégrateur à fuite est une référence aux circuits électriques parce qu'un neurone polarisé se comporte électriquement comme un condensateur, avec un courant de fuite comme sur la figure 2.8. Les réseaux de neurones intégrateurs à fuite sont très utilisés pour modéliser le comportement d'une assemblée de neurones [2] (Dayan et Abbott, 2005).

Le modèle LPDS (Locally Projected Dynamic System) Variante du modèle LI dont les propriétés permettent de vérifier mathématiquement la stabilité des réseaux dans lesquels il est utilisé. Il a notamment été utilisé dans le cadre de modèles contractants des ganglions de la base (Girard et al., 2008). Dans le cas d'une simulation utilisant la méthode d'Euler pour ses intégrations, son comportement est calculé de la manière suivante :

$$p_t^{(i)} = \sum_{j \in C} w_{i,j} y_t^{(j)} \tag{2.3}$$

$$a_{t+dt}^{(i)} = \max(0, \min(1, a_t^{(i)} + \frac{p_t^{(i)} - a_t^{(i)} + T_i}{\tau^{(i)}}.dt)) \tag{2.4}$$

$$y_{t+dt}^{(i)} = \begin{cases} a_{t+dt}^{(i)} & \text{si i est excitateur} \\ -a_{t+dt}^{(i)} & \text{si i est inhibiteur} \end{cases} \tag{2.5}$$

$y_{t+dt}^{(i)}$ étant la variable de sortie du neurone i, T_i et $\tau^{(i)}$ des constantes propres du neurone et $p_t^{(i)}$ la somme des entrées à l'instant t. La principale différence entre ce modèle et un LI vient des opérateurs min et max qui limitent les valeurs possibles pour la variable d'état interne du neurone. L'intérêt est de fortement réduire le temps maximum de retour à l'état de stabilité dans le cas où le neurone est soumis à des entrées ayant une forte intensité (dans le cadre d'une entrée inhibitrice très

1. On appelle un élément du premier ordre, un système décrit par l'équation différentielle du premier ordre : $\frac{dx(t)}{dt} = f(x(t), t)$ où la fonction f définit le système dynamique étudié

2. Une assemblée de neurones est un groupe de neurones qui entretiennent entre eux des connexions synaptiques renforcées, de sorte qu'ils ont plus de chance d'être actifs tous ensembles en même temps.

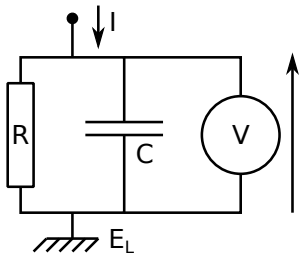


FIGURE 2.8 – circuit minimal d'un neurone LI.

forte, un neurone LI peut voir sa variable d'état atteindre des valeurs très fortement négatives. Après l'arrêt de cette entrée, la variable interne mettra un temps important avant de revenir à l'état d'équilibre, même avec une entrée excitatrice).

Les modèles impulsionnels Modèles dont le but est de décrire la série d'impulsions générée par un neurone. La "sortie" du modèle est une suite d'instants, le temps des potentiels d'action. L'intérêt de ces modèles est de pouvoir illustrer une capacité de synchronisation contrairement aux modèles précédents. Or il a été montré que la synchronisation neuronale existe dans le système nerveux et a un rôle fonctionnel (Brette, 2003). Selon Maass (1997), ils sont computationnellement plus efficaces que les autres réseaux de neurones.

Il existe différents types de modèles impulsionnels :

- Le modèle de Hodgkin et Huxley (1952) (HH) : ce modèle mathématique décrit la dynamique temporelle du neurone dans différents compartiments et reproduit les principaux "modes" de fonctionnement du neurone biologique. Le modèle est défini par quatre équations différentielles inter-dépendantes.
- Le modèle Integrate & Fire (IF) et ses dérivés : proposé par Lapicque (Lapicque, 1907) se place à un niveau de détail de modélisation moins précis que le modèle de Hodgkin & Huxley. Le neurone est modélisé tel un circuit électrique composé d'un condensateur et d'une résistance électrique (dipôle RC). Le modèle est décrit, mathématiquement, par une seule équation différentielle.
- Leaky Integrate & Fire : adaptation du modèle précédent afin que le potentiel de membrane tende à revenir à sa valeur de repos, en l'absence de stimulation, ce qui est plus proche de la réalité biologique.
- Le Spike Response Model (SRM) (Gerstner et Kistler, 2002) : il s'agit d'une modélisation phénoménologique du neurone, au sens où l'on ne considère pas les mécanismes sous-jacents, mais seulement le comportement du neurone.
- Le modèle d'Izhikevich (Izhikevich, 2003) fait intervenir deux équations différentielles couplées, permettant de reproduire une vingtaine de comportements dynamiques différents, ce qui le rend utile pour les modélisateurs qui souhaitent que leurs modèles restent proches des observations biologiques.

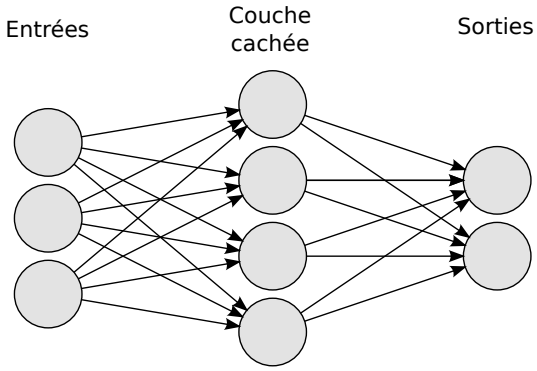


FIGURE 2.9 – Réseau feed-forward avec une couche cachée.

2.2.2 Réseaux de neurones

Le choix de la topologie d'un réseau dépend de la tâche que l'on souhaite résoudre

Les réseaux "feed forward" Ce sont des réseaux dont la structure suit une logique de traitement de l'information au travers de couches de neurones successives, de l'entrée vers la sortie, sans retour de l'information en amont (voir figure 2.9). C'est par exemple le cas des perceptrons et perceptrons multi-couches (Rosenblatt, 1958; Rumelhart et al., 1986). Dans ces réseaux la dynamique est dirigée par la présentation des exemples d'entrée. Les activations se propagent en sens unique, de la couche d'entrée à la couche de sortie. Ils sont utilisés pour de la classification, reconnaissance des formes (caractères, parole, ...) (LeCun et al., 1989) ou pour de la prédiction.

Cartes auto-organisatrices Ces cartes (Fukushima, 1975; Kohonen, 1982; Rumelhart et Zipser, 1985) sont inspirées de la structure du cortex, notamment visuel, dans lequel on peut observer une connectivité locale . En d'autres termes, chaque neurone est connecté aux entrées et à ses voisins. Parmi les différentes applications réalisées à l'aide des cartes auto-organisatrices, un assez grand nombre sont des tâches de classification non supervisées : comme une aide dans l'analyse d'observations satellitaire (Yacoub et al., 2001) ou la recherche documentaire (Kaski et al., 1998). Pour une liste d'applications se référer à Kohonen (2001).

Réseaux Récurrents il s'agit de réseaux dont la structure, peut comporter des récurrences (voir figure 2.10). Ces récurrences peuvent changer radicalement la dynamique qui pourra s'instaurer dans un réseau de neurones et l'amener à s'auto-entretenir. La notion de réseau récurrent est étudiée et mise en application dans une mémoire auto-associative (Hopfield, 1982). L'utilisation de récurrence sera reprise dans le contexte des perceptrons multicouches, avec le réseau de Jordan (Jordan, 1986) et le réseau de Elman (Elman, 1990). Dans ces deux modèles, l'activation

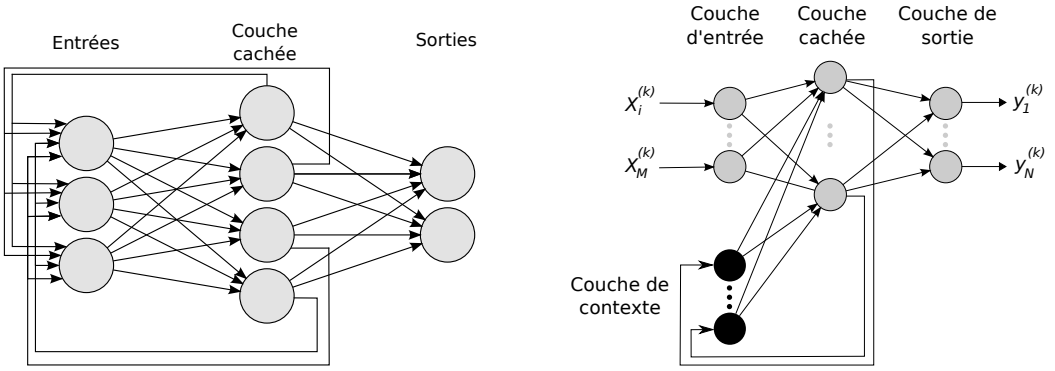


FIGURE 2.10 – Architectures de réseaux récurrents. (a) Réseau récurrent simple. (b) Réseau de type Elman.

de la couche de sortie (dans le cas Jordan) ou de la couche cachée (dans le cas de Elman) est dupliquée en retour dans la couche d'entrée. Les réseaux récurrents utilisant des intégrateurs à fuite peuvent être désignés sous le nom de réseaux de neurones récurrents à temps continus (continuous time recurrent neural network, CTRNN) (Beer, 1995; Yamauchi et Beer, 1996). Ils sont connus pour être théoriquement capables de répliquer n'importe quels systèmes dynamiques et il a été montré que des petits CTRNN sont capables de dynamiques complexes (Beer, 1995, 2006; Bongard, 2011b).

Les réseaux *Echo State* Les réseaux *echo state* (Jaeger, 2002) sont composés d'une couche cachée faiblement et aléatoirement connectée (autour de 1% de connectivité). Les taux de connectivité et les poids de connexion de la couche cachée sont fixés au préalable et doivent respecter la propriété d'*echo state* (cette propriété est décrite à la section 3.2.2.2).

Ce type de réseau a été testé sur différentes applications robotiques telles que la détection d'évènements complexes dans la navigation d'un robot autonome (Antonelo et al., 2007) ou des tâches impliquant une forme de mémoire (Hartland et al., 2009).

2.2.3 Optimisation de paramètres au sein d'une topologie fixe

Une des méthodes les plus simples en ER utilisant des réseaux de neurones consiste à définir au préalable une topologie ad hoc qui restera fixe durant le processus d'évolution et de ne faire évoluer que certains paramètres comme les poids de connexion (d'autres paramètres peuvent évoluer tels que le biais ou la pente de la fonction de transfert). Ce type d'approche a été très utilisé notamment sur des réseaux simples sans couche cachée (Floreano, 1994), sans bouclage (Bongard et Pfeifer, 2002) et surtout de nombreux types de réseaux récurrents comme des CTRNNs (Beer, 1995; Ulbricht, 1996; Gallagher, 1999; Slocum et Downey, 2000;

Paine, 2005; Maniadakis et al., 2010), des réseaux séquentiels en cascade ("Sequential Cascaded Networks", SCN) (Pollack, 1991) utilisés dans différentes expériences (Ziemke, 1996, 1999; Ziemke et Thieme, 2002), ou des réseaux de Elman (Elman, 1990) comme dans (Miglino et al., 1994).

Cependant définir une topologie fixe pour un problème donné n'est pas trivial. Bien que la plupart des expériences en neuroévolution avec topologie fixe utilisent une couche cachée totalement connectée, le fait de déterminer le nombre de noeuds cachés est un processus d'essai-erreur. L'évolution de réseaux à topologie variable pallie à ce problème en laissant le processus d'évolution la possibilité de modifier la topologie du réseau. Gruau et al. (1996) a évoqué le fait que faire évoluer la structure permet de sauver le temps perdu par les humains à essayer de déterminer la topologie des réseaux pour un problème particulier.

2.2.4 Optimisation de réseaux à topologie variable

Dans un processus d'optimisation de réseaux de neurones, outre la modification de différents paramètres du réseau, le processus d'évolution peut créer ou supprimer des connexions au sein d'un réseau (Jakobi, 1997) ainsi que potentiellement ajouter ou supprimer de nouveaux neurones (Gruau, 1993; Stanley et Miikkulainen, 2002). Cette étape élargit considérablement les possibilités offertes par la neuro-évolution (Yao, 1993; Branke, 1995) en permettant notamment l'évolution de réseaux ayant des topologies efficaces mais peu intuitives et donc souvent très différentes de celles conçues à la main ou par des méthodes d'ingénierie classiques.

L'évolution de la topologie de réseaux de neurones requiert de générer un graphe, dont la conversion vers ou depuis un génotype, n'est pas triviale. L'alternative suivante est utilisée :

- Une première approche, les codages directs, regroupent toutes les méthodes où l'obtention du phénotype d'un individu ne s'est fait que par des transformations bijectives de son génotype. Une méthode habituellement utilisée est de directement soumettre une représentation du graphe du réseau au processus d'évolution, ce qui implique de définir des opérateurs de mutation et ou de croisement adaptés qui agissent directement sur la topologie du réseau.
- Une seconde approche, les codages indirects utilisent un génotype sous forme compacte, et définissent un processus permettant de transformer ce génotype compact en réseau de neurones. L'indirection entre le génotype et le phénotype est souvent appelé processus génératif ou développemental en référence au développement biologique.

2.2.4.1 Les codages directs

Dans un codage direct, le génotype contient une représentation de la topologie du réseau de neurones, et aucune transformation de ce graphe n'est réalisée pendant le passage du génotype au phénotype. Il y a une relation bijective entre le génotype d'un individu et son phénotype.

Les codages directs ont été utilisés dans plusieurs travaux (Mouret et Doncieux, 2008; Doncieux et Meyer, 2003; Mouret et Doncieux, 2006; Stanley et Miikkulainen, 2002, 2003; Moriguchi et Honiden, 2011; Mouret et Doncieux, 2012). ils reposent principalement sur la création de nouveaux individus par mutation et ne permettent souvent pas d'effectuer de croisement, car cela implique de faire correspondre entre elles des parties ou des neurones de deux réseaux afin de les échanger. Quand ces deux réseaux n'ont pas la même topologie, il est délicat voir impossible de définir ces correspondances pour des neurones autre que les entrées et les sorties. Deux solutions ont cependant été proposées La première consiste à réaliser des analyses sur les réseaux évolués afin de définir des sous-réseaux communs aux parents (Mouret et Doncieux, 2008). La seconde est utilisée par le codage *NEAT* pour *Neuro-Evolution of Augmented Topologies* (Stanley et Miikkulainen, 2002; Whiteson et al., 2005; Stanley et Miikkulainen, 2003), codage direct le plus utilisé en neuroévolution.

Ce codage possède sur trois caractéristiques principales :

- il emploie une méthode permettant le croisement entre différentes topologies : La différentiation et les comparaisons entre réseaux reposent sur des numéros d'innovations uniques, associés, à leur création, à chaque nœud du réseau. Ces numéros, copiés lorsqu'un réseau génère des enfants, permettent notamment de mettre en correspondance les neurones de deux réseaux et donc de définir un opérateur de croisement ;
- il protège les innovations structurelles en utilisant des méthodes de spéciations : il utilise notamment des niches protégées par l'algorithme en fonction des différences génotypiques entre réseaux :
- la topologie des réseaux débute par une structure minimale et croit de manière incrémentale pendant le processus d'optimisation : le processus d'évolution démarre avec des réseaux ayant tous la même topologie (le plus souvent, ces réseaux sont sans récurrence, sans neurone caché et chacune des entrées est connectée à toutes les sorties) et pendant l'évolution, les réseaux se complexifient progressivement et les nouvelles topologies sont favorisées en protégeant les individus ayant une topologie originale.

Ses opérateurs de mutations, qui permettent de modifier un réseau de neurones directement, sont les suivants :

- modification du poids d'une connexion ;
- modification des paramètres d'un nœud : ces paramètres tels que le biais interne, ne sont pas présents dans les premières versions du codage NEAT ;
- création d'une nouvelle connexion entre deux neurones ;
- suppression d'une connexion existante ;
- création d'un nouveau neurone : une connexion est supprimée et remplacée par un nouveau neurone ayant une connexion entrante et une connexion sortante à la place de l'ancienne connexion.

2.2.4.2 Les codages indirects

Les codages indirects (ou génératifs) sont des codages où une transformation est nécessaire pour générer le phénotype d'un individu à partir de son génotype. Cet ensemble inclut les codages développementaux dans lesquels l'indirection entre le phénotype et le génotype est une séquence d'opérations souvent inspirée du développement d'organismes biologiques. De manière générale, l'utilisation de codages indirects ne se limite pas à la neuro-évolution mais permet l'évolution d'autres structures telles que des circuits électroniques (Mattiussi, 2005) ou des structures mécaniques (Kowaliw et al., 2007).

L'objectif principal des codage indirects est de faire évoluer rapidement des structures plus grandes et plus complexes que celles obtenues par des codages directs (Hornby et Lipson, 2001). Pour cela ils s'inspirent de certaines propriétés par ailleurs présentes dans de nombreuses structures organiques et d'ingénierie (Hartwell et al., 1999; Variano et al., 2004; Hornby, 2005) :

- Modularité : la modularité d'une structure correspond à la localisation structurelle d'une fonction.
- Régularité : la régularité d'une structure est la compressibilité de la représentation de la structure. Cela correspond à l'idée qu'un codage indirect doit pouvoir générer une structure complexe et de grande taille à partir d'une représentation compacte.
- Hiérarchie : une structure est hiérarchique si sa "composition" est récursive.
- Répétitivité : un réseau est dit répétitif si un de ses sous-ensembles est présent plusieurs fois avec peu ou pas de variations entre les répétitions.
- Symétrie : un réseau est dit symétrique si l'ensemble ou un sous-ensemble du réseau présente un centre ou un axe de symétrie.

Des codages génératifs tels que MENNAG (Mouret, 2008) visent notamment à faciliter la réutilisation de modules fonctionnels. MENNAG utilise une grammaire permettant à l'évolution de faire évoluer des programmes pour générer des structures (des réseaux de neurones). Les mécanismes d'obtention de modularité fonctionnelle au sein des algorithmes évolutionnistes restent un problème ouvert (Altenberg, 2004; Mouret, 2008; Hansen, 2003). Les travaux réalisés jusqu'à présent tendent à lier la modularité fonctionnelle à la pression de sélection plus qu'au codage utilisé (Kashtan et Alon, 2005; Mouret et Doncieux, 2008; Clune et al., 2012).

La structure hiérarchique par défaut est celle d'un arbre où un nœud de départ se décompose en branches successives. Le processus de développement de tous les organismes multicellulaires basés sur un œuf unique est hiérarchique. Ce processus est copié par les codages cellulaires (Gruau, 1994) et l'ensemble des codages basés sur les *L-systems* (Hornby et Lipson, 2001; Hornby et Pollack, 2002; Prusinkiewicz et Lindenmayer, 1990). Une autre notion présente dans des codages plus récents est celle de graphes sans échelle, qui conservent des propriétés (connectivité, nombre de connexions par nœud...) identiques malgré des tailles différentes. Cette propriété se retrouve dans de nombreuses structures naturelles telles que les réseaux sociaux ou les réseaux de neurones (Eguíluz et al., 2005) et est recherchée par des codages

tels que DSE (Suchorzewski, 2011) ou HyperNEAT (Woolley et Stanley, 2011).

On retrouve des propriétés de symétrie et de répétition dans les réseaux issus du codage *Cellular Encoding* (Gruau, 1994) ou du codage GENRE (Hornby et Pollack, 2002)). Pour obtenir ces propriétés AGE (*Analog Genetic Encoding*) (Mattiussi et Floreano, 2007; Mattiussi, 2005) utilise un processus s'inspirant des mécanismes biologiques d'affinité entre protéines. pour générer des graphes à partir d'une séquence de caractères évoluée par l'algorithme. Il s'est avéré que ce codage était efficace pour faire évoluer d'autres types de graphes et a notamment été utilisé pour concevoir des circuits électroniques (Mattiussi, 2005).

Une autre approche a été développée principalement par Stanley avec HyperNEAT (Stanley, 2007; Stanley et al., 2009). HyperNEAT utilise un codage indirect appelé CPPN (Stanley, 2006, 2007) (*Compositional Pattern Producing Networks*) qui peut produire des schémas de connexions avec des symétries et des motifs répétés. L'avantage de cette approche de pouvoir exploiter la géométrie d'une tâche en mappant les régularités de celles-ci à la topologie du réseau. Il est possible de générer des réseaux de neurones dotés de millions de connexions basés sur des motifs géométriques. HyperNEAT est l'un des codages indirects les plus étudiés ces dernières années (Stanley, 2007; Stanley et al., 2009; Risi et Stanley, 2010; Risi, 2011; Clune et al., 2009; Drchal et al., 2009; Risi, 2011; Suchorzewski, 2011). Une présentation plus exhaustive des codages génératifs peut être vu dans (Tonelli, 2012).

2.2.5 Synthèse

Nous avons pu faire un bref aperçu sur l'évolution de réseaux de neurones avec une description de différents modèles de neurones ainsi que des topologies les plus couramment utilisées.

Modèles de neurones. Le choix d'un type de modèles de neurones dépend de différents éléments. L'utilisation d'algorithmes évolutionnistes nécessite un grand nombre d'évaluations (parfois $> 10^6$) avant de trouver une solution acceptable. Utiliser des modèles de neurones impulsionnels est donc très coûteux en temps de calcul. Ils dépendent de plus de nombreux paramètres, ce qui élargit l'espace de recherche si l'on cherche à les faire évoluer. Notre choix s'est donc porté sur les lPDS (Girard et al., 2008), variante des neurones intégrateurs à fuite. Les réseaux de neurones intégrateurs à fuite permettent de modéliser le comportement d'une assemblée de neurones. Les lPDS ont de plus déjà été utilisés en neuroévolution (Mouret et al., 2010).

Réseaux de neurones. Nous avons décrit précédemment trois méthodes possibles pour optimiser des réseaux de neurones via évolution :

- optimisation de paramètres au sein d'une topologie fixe ;
- optimisation de réseaux à topologie variable via un codage direct ;
- optimisation de réseaux à topologie variable via un codage indirect.

Afin de simplifier l'étude de l'influence des pressions de sélection sur l'émergence de mémoire, nous nous intéresserons seulement aux deux premières méthodes au cours des différentes expériences dans la suite de cette thèse.

2.3 Robotique évolutionniste (ER)

Apparu dans les années 1990, l'objectif de la robotique évolutionniste (Harvey et Husbands, 1992; Meyer et al., 1998; Floreano et Nolfi, 2000)(ou ER pour Evolutionary Robotics) est de développer des méthodes de conception automatiques afin de créer des contrôleurs pour robots autonomes, effectuant des tâches dans des environnements non structurés et, ceci, sans la nécessité que le contrôleur soit directement programmé par des humains. Cette approche repose sur l'idée d'utiliser des algorithmes évolutionnistes afin d'optimiser des contrôleurs ou des morphologies pour des robots.

Dans la suite de cette thèse, nous nous intéresserons à l'optimisation de contrôleurs et nous avons choisi d'utiliser des réseaux de neurones comme architecture de contrôle.

La robotique évolutionniste peut être formalisée en tant que problème d'optimisation. L'action a du robot est est obtenue grâce à la formule suivante :

$$a = g_x(p(t) + p(t-1) + p(t-2) + \cdots)$$

où $a \in \mathbb{R}^n$, $p(t)$ correspond aux perceptions du robot au temps t avec $p \in \mathbb{R}^k$ et $g_x()$ correspond à l'architecture de contrôle. Il s'agit donc de trouver $g_x(.)$ minimisant $f(g_x(.))$ avec différentes contraintes :

$$h_j(g_x(.)) \leq 0, j = 1, 2, \cdots, q$$

$$l_j(g_x(.)) = 0, j = 1, 2, \cdots, r$$

d'ailleurs si

$$g_x(p(t) + p(t-1) + p(t-3) + \cdots = g_x(p(t)))$$

le robot peut être considéré comme réactif car son action ne dépend pas des perceptions précédentes.

Nous allons nous intéresser dans cette section aux différentes types de tâches qui ont été réalisées via ER, pour des revues plus détaillées voir (Mataric et Cliff, 1996; Harvey et al., 1997; Meyer et al., 1998; Meeden et Kumar, 1998; Floreano et Nolfi, 2000; Pratihar, 2003; Walker et al., 2003; Pfeifer et al., 2003; Jin, 2005b; Teo et Abbass, 2005; Nelson et Grant, 2007; Nelson et al., 2009; Doncieux et al., 2011).

De nombreuses tâches réalisées en ER sont souvent inspirées de comportements d'animaux (voir tableau 2.3). Nous allons décrire différentes catégories :

- tâches de navigation ;
- tâches nécessitant un comportement cognitif minimal ;
- tâches sensorimotrices intégrant une forme de mémoire ;
- tâches intégrant un changement de règle de comportement.

TABLE 2.3 – Revue de différents articles en ER, inspiré de Walker et al. (2003); Nelson et al. (2009)

Article	Fit*	Type†	Tâche	Robot	Contrôleur‡
Earon et Barfoot (2000)	Ag	R-R	Evolution de posture	Hexapode	
Hornby et Lipson (2001)	Ag	S-R	Locomotion (co-evolution du corps)	TinkerBot	Paramètres
Zufferey et al. (2002)	Ag	R-R	Locomotion, évitement de mur	Dirigeable	NN
Macinnes et Paolo (2003)	Ag	S-R	Locomotion (co-evolution du corps)	Lego-Servo	NN
Zykov et al. (2004)	Ag	R-R	Evolution de posture	Hexapode	Paramètres
Chemova et Veloso (2004)	Ag	R-R	Evolution de posture	Sony AIBO	Paramètres
Jakobi (1998)	C	S-R	Locomotion, évitement d'obstacles	Octopode	NN
Floreano et Mondada (1996)	C	R-R	Locomotion, évitement d'obstacles	Khepera	NN
Lund et Miglino (1996)	C	S-R	Locomotion, évitement d'obstacles	Khepera	NN
Banzhaf et al. (1997)	C	R-R	Locomotion, évitement d'obstacles	Khepera	GP
Gomi et Ide (1998)	C	R-R	Evolution de posture	Octopode	Paramètres
Matellán et al. (1998)	C	R-R	Locomotion, évitement d'obstacles	Khepera	Logique floue
Nordin (1998)	C	S-R	Locomotion, évitement d'obstacles	Khepera	GP
Liu (1999)	C	R-R	Déplacement d'objets	x	GP
Seok et al. (2000)	C	R-R	Phototaxie, évitement d'obstacles		FPGA
Ziegler et Banzhaf (2001)	C	S-R	Locomotion, évitement d'obstacles	Khepera	Graphe
Miglino et al. (1998)	IE	S-R	Retour au nid, évitement d'obstacles	Khepera	NN
Nakamura et al. (2000)	IE	S-S	Ramassage d'objets	Khepera	NN
Pasemann et al. (2001)	IF	S-R	Retour au nid, évitement d'obstacles	Khepera	NN
Harvey et al. (1994)	IF	R-R	Retour au nid		NN
Lee et al. (1997)	IF	S-R	Déplacement d'objet, retour au nid	Khepera	GP
Filliat et al. (1999)	IF	S-R	Locomotion, évitement d'obstacles	Khepera	NN
Barlow et al. (2005)	IF	S-R	Retour au nid	EvBot	GP
Floreano et Urzelai (2000)	Ad	R-R	Retour au nid	Khepera	NN
Watson (2002)	Ad	R-R	Phototaxie	x	NN
Nelson (2004)	Ad	S-R	Locomotion, évitement d'obstacles	EvBot	NN
Hornby et al. (2005)	Ad	R-R	Evolution de posture	Sony AIBO	Paramètres
Kamio et Iba (2005)	Ad	S-R	Déplacement d'objets, retour au nid	Sony AIBO	GP
Parker et Georgescu (2006)	Ad	S-R	Phototaxie et évitement d'obstacles	LEGO m	GP
Trianni et Dorigo (2006)	Ad	S-R	Locomotion, évitement d'obstacles	Swarm-bot	NN
Hoffmann et Pfister (1996)	Ad	S-R	Retour au nid, évitement d'obstacles	x	Logique floue
Thompson (1995)	Ad	S-R	Locomotion, évitement d'obstacles		FPGA
Ishiguro et al. (1999)	Ad	S-R	Déplacement d'objet, retour au nid	Khepera	NN
Ebner et Zell (1998)	Ad	S-R	Locomotion, évitement d'obstacles	RWI B21	GP
Nolfi (1996)	Ad	S-R	Ramassage d'objets	Khepera	NN
Schultz et al. (1996)	Ad	S-R	Evitement d'obstacles	Nomad 200	Règles
Sprinkhuizen-kuyper et al. (2000)	Ad	S-R	Déplacement d'objets	Khepera	NN
Wolff et Nordin (2001)	Ad	R-R	Optimisation de posture	Eivina	Paramètres
Nehmzow (2002)	Ad	R-R	photo-orientation, évitement d'obstacles	Lego	GP
Okura et al. (2003)	Ad	R-R	Locomotion, évitement d'obstacles	Khepera	FPGA
Quinn et al. (2001)	Ad	S-R	mouvements coordonnées	x	NN
Boeing et al. (2004)	Ad	S-R	Evolution de posture	Andy Droid	Spline
Capi et Doya (2005)	Ad	S-R	Retour au nid	Cyber rodent	NN
Trianni et Dorigo (2006)	Ad	S-R	Locomotion coordonnée	Swarm-bot	NN

* **Type de fitness utilisée** (voir section 4.1.1 pour plus de détails). *C : Fitness comportementale, IF : Fitness par increment fonctionnel, Ad : Fitness adaptée, IE : Fitness par incréments environnementaux Ag : Fitness aggrégée*

† **Type de robots utilisé**. *R-R : phase d'évaluation sur robots réels, S-R : phase d'évaluation sur robot simulé et transfert sur robot réel, S-S : expérience entièrement sur robots simulés.*

‡ **Abbréviations**. *NN : réseau de neurones, GP : programmation génétique, FPGA : circuit logique programmable.*

2.3.1 Tâches de navigation

Tâches de locomotion ou d'évitement d'obstacles. Tâches dans lesquelles les robots doivent évoluer afin d'être capables de se déplacer dans un environnement, tout en évitant des obstacles statiques, voir même mobiles dans certains cas. Lund et Miglino (1996), par exemple, proposent une tâche d'évitement d'obstacles utilisant un robot Khepera. L'architecture est un simple réseau de neurones sans couche cachée. Dans Schultz et al. (1996) un robot marcheur est optimisé afin d'éviter les obstacles. Les contrôleurs sont d'abord évalués en simulation puis transférés sur robot réel pour vérification. Le robot possède des capteurs infrarouges et tactiles.

Comportement phototaxique. Tâches où le robot doit détecter une source de lumière et se diriger vers elle.

Des contrôleurs capables de réaliser un comportement phototaxie dans un environment contenant de nombreux obstacles ont été optimisés dans Parker et Georgescu (2006). Un robot construit à partir de kits LEGO Mindstorm et équipé de deux photodétecteurs et de capteur tactile à été utilisé. Seok et al. (2000) ont présenté l'évolution d'un comportement phototaxique et d'évitement d'objets sur un robot équipé d'un sonar et de capteur de lumière.

Recherche de nourriture avec dépose d'objets (*foraging*). Dans ce type de tâche, le robot doit trouver des objets dans un environnement, les ramasser, les porter vers un autre lieu et les déposer. Nous pouvons citer comme exemple une tâche de ramassage de balles dans une arène proposé par (Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012), où le robot doit effectuer une série de sous-tâches : naviguer, trouver une balle, ramasser la balle, atteindre la corbeille, relâcher la balle, ... Ce type de tâche est considéré comme une des tâches les plus complexes en robotique évolutionniste (Nelson et al., 2009), car elle contient une séquence d'évènements qui n'est pas facilement réalisable avec un système purement réactif.

Tâche de retour au nid. Ici, le robot doit retourner dans son nid et la localisation de cet emplacement n'est pas indiquée par une source de lumière et peut ne pas être indiquée du tout. Floreano et Mondada (1996) ont optimisé un robot Khepera possédant des capteurs infrarouges, des lumières, et des capteurs de niveau de batterie. L'objectif de ce robot était d'être capable de se déplacer tout droit le plus rapidement possible ainsi que de retourner à sa base afin de recharger ses batteries. Le robot optimisé était capable d'intégrer de l'information sur l'environnement et sur la consommation d'énergie afin de permettre au robot de retourner à la station de recharge périodiquement, juste avant que son niveau d'énergie tombe en dessous d'un seuil critique. Vickerstaff et Di Paolo (2005a,b) ont optimisé des robots simulés (dotés de compas, de capteurs de vitesse et de capteurs de lumières) pouvant exhiber un comportement de retour au nid. Les robots qui sont initialement placés dans leur nid, sont sélectionnés pour leur capacité à retourner au même endroit après avoir atteint un nombre variable de balises lumineuses placées aléatoirement.

2.3.2 Tâches impliquant un comportement cognitif minimal

Le terme de comportement cognitif minimal a été défini par (Beer, 1996). Beer définit ce terme comme étant un compromis entre une tâche assez complexe pour être "*representation-hungry*" (Clark, 1997) et soulever des questions cognitivement intéressantes tout en restant assez simple dans le but d'être facilement analysable. Un problème "*representation-hungry*" employé par Clark (1997) représente les cas où il semble qu'il y a nécessité d'une représentation interne au sein d'un contrôleur.

Pour illustrer ce comportement cognitif minimal, Beer (1996) a proposé une tâche où un agent visuellement guidé a été optimisé pour s'orienter et atteindre des objets tout en étant capable de discriminer différents objets. L'agent attrape les objets circulaires et évite les objets en forme de losange (Beer, 2003).

2.3.3 Tâches sensorimotrices intégrant une forme de mémoire

Gallagher (1999); Slocum et Downey (2000) proposent une extension du comportement cognitif minimal évoqué par (Beer, 1996). Dans leur tâche, le robot évolue pour être capable d'attraper des objets tombant verticalement après avoir seulement brièvement observé sa position, pour ensuite se déplacer vers le bon emplacement.

Une des premières tâches utilisées en robotique évolutionniste impliquant une certaine forme de mémoire a été le problème de panneau de signalisation (Ulbricht, 1996; Jakobi, 1997; Rylatt et Czarnecki, 2000; Linaker et Jacobsson, 2001; Bergfeldt et Linaker, 2002; Blynel et Floreano, 2003; Kim, 2004; Ziemke et Thieme, 2002; Ziemke et al., 2004) (*"road-sign problem"*). Dans cette cette tâche le robot doit choisir une direction en fonction d'un stimulus de départ, comme par exemple l'emplacement d'une source de lumière. C'est une tâche à réponse différée (Delayed Response task, cf. 3.3), habituellement considérée comme demandant une certaine forme de mémoire à court terme puisque l'agent doit se souvenir durant une certaine période quel était l'emplacement de la source de lumière. Le problème de panneau de signalisation le plus simple consiste en un labyrinthe en T (fig. 3.4) où le robot doit tourner à gauche ou à droite en fonction de l'emplacement d'un stimulus situé au début du corridor (Ulbricht, 1996; Jakobi, 1997; Linaker et Jacobsson, 2001). Ce protocole a été utilisé dans différents cas : pour tester la capacité de généralisation au niveau temporel (Ulbricht, 1996), pour éprouver la robustesse d'un contrôleur en vue d'être transféré sur un robot réel (Jakobi, 1997; Koos et al., 2012), dans des versions étendues (Linaker et Jacobsson, 2001; Ziemke et Thieme, 2002; Ziemke et al., 2004), ou comme tâche de mémoire (Ziemke et Thieme, 2002; Kim, 2004).

Nolfi (2002a) a optimisé un robot Khepera simulé capable de naviguer dans un environnement comportant deux pièces et de déterminer dans quelle pièce il se trouve. Dans l'expérience décrite dans cet article, le robot ne pouvait pas faire une méthode d'intégration de chemin, du fait que la position de départ du robot était initialisée aléatoirement. De plus, dans l'expérience, le robot se retrouvait dans des états de capteurs identiques à différents endroits de l'environnement. Il était donc nécessaire de discriminer le lieu en intégrant des informations à travers le temps.

Dans deux autres expériences proposées par (Nolfi et Tani, 1999; Tani et Nolfi, 1999), l'objectif est de déterminer comment des robots à roues – qui disposent de capteurs de distance et se déplacent dans un environnement à plusieurs pièces – peuvent développer une capacité de prédire les états des capteurs à venir en segmentant les états des capteurs en catégories et en anticipant les catégories des états des capteurs à venir.

Dans la tâche de Capi et Doya (2005), l'agent doit visiter alternativement un emplacement où se trouve de la nourriture et un autre où se trouve de l'eau, après avoir visité son nid. L'agent doit apprendre des réponses réactives de bas niveau afin d'atteindre le prochain emplacement récompensé tout en évitant les autres. A plus haut niveau, l'agent doit sélectionner le prochain but à visiter. Cette tâche est présentée comme étant une tâche non-markoviennes séquentielles où la mémoire du précédent évènement ne peut pas déterminer la prochaine action.

Hartland et al. (2009) utilisent le peigne de Tolman (Tolman, 1948). Celui-ci est composé d'un corridor comportant de nombreuses branches s'ouvrant sur le même côté à intervalles réguliers. L'objectif du robot est de tourner exclusivement dans la troisième branche dans l'environnement. Cela impose au robot d'identifier un contexte temporel, le seul permettant de distinguer les différentes branches.

2.3.4 Tâches intégrant un changement de règle de comportement

Dans ce type de tâche, le comportement de l'agent dépend du contexte. En fonction d'une entrée externe celui-ci doit changer son comportement. Comme par exemple sur une tâche impliquant un comportement phototaxique où, en fonction d'une récompense le robot doit éviter ou s'approcher d'une source de lumière Meeden (1995); Ziemke (1996).

Maniadakis et Tani (2009) et Maniadakis et al. (2010) proposent une extension du "roadsign problem". Dans ce cas, la règle change durant l'expérience : soit le robot doit tourner du même côté que le stimulus visuel, soit il doit se diriger du côté opposé. Pour apprendre à tourner du bon côté du labyrinthe en T, l'agent reçoit un signal de récompense ou de punition. Après un nombre aléatoire d'essais, la règle change et l'agent doit être capable de s'adapter et à réapprendre à tourner du bon côté. C'est une tâche inspiré du tri de carte du Wisconsin étudiée chez différents animaux comme chez l'homme ou chez les rats (Joel et al., 1997).

2.3.5 Synthèse

Nous avons pu faire un bref aperçu de différentes tâches effectuées en robotique évolutionniste. Parmi elles, un grand nombre de tâches sont réactives et ne nécessitent donc pas nécessairement une forme de mémoire au sein de l'agent. Parmi les tâches de mémoire, la tâche "panneau de signalisation" fait partie des plus fréquemment utilisées. Celle-ci sera notre point de départ au chapitre suivant, pour tester notre hypothèse sur le caractère trompeur de l'émergence d'une forme de mémoire au sein d'un agent.

Chapitre 3

Synthèse de capacité de mémoire

La mémoire est la sentinelle de l'esprit.

William Shakespeare

Objectifs Illustrer le caractère trompeur de la synthèse de capacité de mémoire au sein d'un agent mobile. Proposer un protocole expérimental favorisant l'émergence de mémoire.

Démarche

- Description des différents types de mémoire ;
- illustration à l'aide de tâches issues de l'état de l'art de la difficulté d'obtenir des individus doté d'une forme de mémoire ;
- proposition d'un protocole expérimental inspiré de tâches issues des neurosciences.

Résultats

- Les agents agents obtenus ont tendance à externaliser leur mémoire ;
- Lorsque le protocole expérimental ne permet pas aux individus d'externaliser leur mémoire, le processus d'optimisation peine à trouver des solutions satisfaisantes.

Sommaire

3.1 Mémoire

La mémoire peut être vue comme la capacité de se rappeler des expériences passées. Suivant les domaines de recherche la définition de la mémoire diffère :

- Psychologie : *"Human memory is a system for storing and retrieving information, information that is, of course, acquired through our senses."* Baddeley (1997)
- Psychologie cognitive : *"Consider memory to mean the mental processes of a acquiring and retaining information for later retrieval, and the mental storage system that enables these processes. Operationally, memory demonstrated when the processes of retention and retrieval influence your behavior or performance in some way, even if you are unaware of the influence." (Ashcraft, 1994)*
- *"Memory is an indexed encyclopedia ; stimuli evoke the appropriate index entries, which point, in turn, to the relevant information."* Vera et Simon (1993)
- Cybernétique : *"[Memory] is a concept that the observer invokes to fill in the gap caused when part of the system is unobservable."* (Ashby, 1956)
- Philosophie : *Ensemble d'idées simples plus vives qu'elles ne sont dans l'imagination, et conservées dans l'ordre et la forme des impressions d'où elles sont issues.* (Hume, 1739)
- Neuroscience cognitive : la mémoire est étroitement liée aux mécanismes perceptifs. Tout sentiment de familiarité, toute activité de reconnaissance et d'identification relativement à un objet perçu nécessite forcément l'activation de représentations en mémoire (Houdé et al., 1998).
- Informatique : la mémoire est un dispositif physique faisant partue de tout ordinateur, capable d'enregistrer à des adresses spécifiées une quantité fixe d'informations et de les restituer après un temps indéterminé (Houdé et al., 1998).

Nous étudierons dans la section suivante les différents types de mémoires décrites en psychologie cognitive.

3.1.1 Types de mémoire

Au niveau de la mémoire, une distinction peut être faite selon le caractère des états mentaux : un état mental transitoire est défini comme de la mémoire à court terme (MCT) ; un caractère stable comme de la mémoire à long terme (MLT).

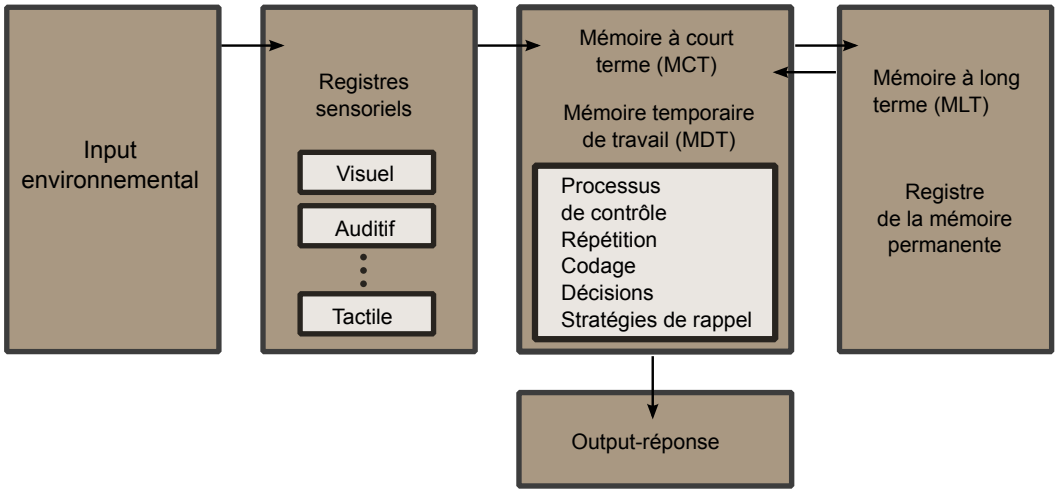


FIGURE 3.1 – Flux d'information dans le système mnésique. Source (Atkinson et Shiffrin, 1971)

Un des modèles fondateur de la psychologie cognitive, le modèle modal proposé par Atkinson et Shiffrin (1968, 1971) (visible à la figure 3.1) repose sur 3 types de mémoires :

- une mémoire sensorielle (ou Registre d'Informations Sensorielles ; RIS) ;
- une Mémoire à Court Terme (MCT) ;
- une Mémoire à Long Terme (MLT).

La distinction entre ces mémoires se fait en fonction du temps pendant lequel l'information peut être retenue, et en fonction des types de traitement qu'elles effectuent.

La mémoire sensorielle. Elle permet de stocker pendant un intervalle très court ($t < 1$ sec) un stimulus (un son, une lumière) venant de l'environnement, si cette information mérite attention le stimulus est transféré dans la MCT.

Mémoire à court terme (MCT). La MCT permet de garder en mémoire une information ($t < 1$ min) et de pouvoir la restituer pendant ce délai. Cela correspond par exemple à une tâche où il faut restituer dans l'ordre, une série d'éléments qui viennent d'être énoncés.

Une fois stockée en MCT, l'information peut faire l'objet de différents traitements cognitifs comme l'autorépétition qui permet d'augmenter la durée de rétention, le processus d'encodage pour transfert l'information dans la mémoire à long terme, le processus inverse de récupération d'informations en MLT pour les transférer en MCT ou le processus de décision intervenant dans la production et la mise œuvre d'une réponse face au stimulus initial.

La MCT porte sur un nombre limité d'information (empan), particulièrement

sensibles à toute activité interférente. Sa capacité est limitée à 7 items plus ou moins 2 (Miller, 1956), voire 4 (Cowan, 2001).

La mémoire de travail (MDT) est une extension du concept de mémoire à court terme, proposé par (Baddeley et al., 1974). La MDT permet d'effectuer des traitements cognitifs sur les éléments qui y sont temporairement stockés et est impliquée dans des processus faisant appel à un raisonnement. Elle est utile par exemple, pour restituer dans l'ordre inverse, une série d'items qui vient d'être énoncée.

Les différentes composantes de la MDT jouent un rôle clé dans plusieurs activités quotidiennes telles que le raisonnement, la compréhension du langage, l'apprentissage de vocabulaire ou la lecture.

Mémoire à long terme (MLT). la MLT contient l'ensemble des connaissances acquises par le sujet au cours de son existence. Elle est en charge de différerents mécanismes comme la gestion de ces informations (organisation, modifications en cas de nouvelle connaissances, convervation) ainsi que le recouvrement qui consiste à transferer des informations en MLT en MCT afin de prendre une décision. Différentes types de MLT peuvent être distinguées.

Mémoire explicite (Graf et Schacter, 1985) (déclarative (Cohen et Squire, 1980)) Elle implique les stratégies intentionnelles de recherches d'informations. Ce sont des connaissances qui peuvent être représentées à l'aide du langage naturel ou sous la forme d'image mentales, en principe accessible à la conscience. Elle peut être séparée en 2 catégories :

- la mémoire épisodique (Tulving, 1972) qui regroupe les informations pouvant êtres situées dans le temps et dans l'espace et dont la récupération est fortement liée à des effets de contexte ;
- la mémoire sémantique (Tulving, 1972) qui regroupe des connaissances générales du monde, indépendamment du moment et du lieu et de leur constitution, et dont la récupération est fortement liée à des effets de contexte.

Mémoire implicite (Graf et Schacter, 1985) (non déclarative) Il s'agit d'une forme de mémoire où l'on ne retient pas l'expérience qui en est à l'origine. Le rappel d'un souvenir encodé dans la mémoire implicite se fait automatiquement, sans les efforts de rappel nécessaires à la mémoire explicite.

Elle contient :

- la mémoire procédurale (Cohen et Squire, 1980) impliquée dans la réalisation des activités perceptivomotrices ou cognitives dont le contenu est difficilement accessible à la conscience ;
- les conditionnements émotionnels.

3.1.1.1 Synthèse

Selon Atkinson et Shiffrin (1971), il y a trois états possibles d'une connaissance : un état de non-acquisition ; un état d'acquisition instable (stockage en MCT) ; un

état d'acquisition stabilisé (stockage en MLT). Ce modèle postule que l'information provenant de l'extérieur est d'abord traitée dans les registres sensoriels. L'information est alors transmise à la mémoire à court terme pour y être encodée. Ce registre transitoire est conçu comme le passage obligé pour les informations avant la transmission en mémoire à long terme. La MDT, extension de la MCT, est donc un élément central au sein du système mnésique. Nous nous intéresserons donc plus particulièrement à cet élément dans la suite du chapitre.

3.1.2 Mémoire de travail

Parmi les théories proposées sur la mémoire de travail (Miyake et Shah, 1999), le modèle le plus influent est celui proposé initialement par Baddeley et al. (1974). Le modèle de (Baddeley, 2000) ajoute un élément supplémentaire par rapport au modèle de (Baddeley et al., 1974), le buffer épisodique. Ce modèle est composé de quatre éléments :

- L'administrateur central (appelé aussi système central exécutif) : il sélectionne, coordonne et contrôle les opérations de traitements. Selon Baddeley, il fait référence aux processus attentionnels.
- La boucle phonologique : cette composante manipule les informations sonores. Elle est l'espace d'auto-répétition mentale.
- Le calepin visuo-spatial : celui-ci manipule les informations visuelles et spatiales. Il joue un rôle dans la manipulation des images mentales.
- Le buffer épisodique : il récupère les informations de plusieurs types différents, parfois plusieurs en même temps, afin de les consolider en MLT, et également pour faire le lien entre ces deux types d'informations. Il est contrôlé par l'administrateur central.

3.1.2.1 Bases neurophysiologiques de la MDT

De nombreux travaux ont montré que le cortex préfrontal (PFC) semble être la structure du cerveau la plus liée à la mémoire de travail (Fuster, 2008). En particulier, une activité persistante élevée dans des neurones du PFC a été observée chez les singes lors de tâches nécessitant de mémoriser des éléments (Goldman-Rakic, 1987; Funahashi et al., 1989). Différentes types d'informations semblent être maintenus dans le PFC comme des informations spatiales (Funahashi et al., 1989), des objets récemment vus (Miller et al., 1996), des règles d'action abstraites (Miller et al., 1996) ou des informations verbales (Demb et al., 1995).

Il existe de nombreux modèles tentant de décrire la manière dont cette activité persistante pourrait être générée. Voici les grandes catégories décrites par Durstewitz et al. (2000) :

- l'activité est maintenue par l'intermédiaire de fortes connexions récurrentes excitatrices au sein d'une assemblée de neurones (Hebb, 1949). La plupart des recherches ont été menées au sein de cette catégorie de modèle.

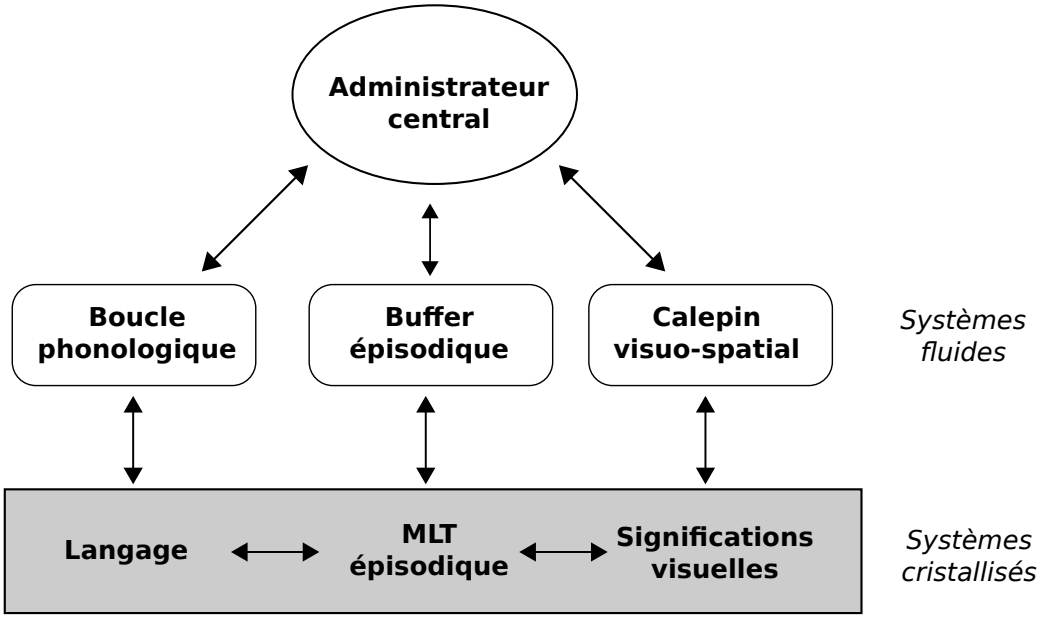


FIGURE 3.2 – Modèle de mémoire de travail d'après Baddeley (2000).

– l'activité circule en boucle appelées "synfire chains" (Abeles, 1991) . Une "synfire chains" est constituée de neurones répartis dans des couches successives, liées entre elles par des connexions unidirectionnelles. Du fait des émissions quasi-simultanées des neurones d'une couche, les neurones de la couche suivante vont être activés de manière synchrone, et vont à leur tour propager leurs activités sur les neurones de la couche suivante, et ainsi de suite.
– l'activité est maintenue par l'intermédiaire des courants membranaires qui autorise une bistabilité cellulaire (Marder et al., 1996; Lisman et al., 1998).
– de modèles attracteurs à état continus pouvant représenter des variables continues comme une position spatiale ou la fréquence d'un stimulus (Machens et Brody, 2008)

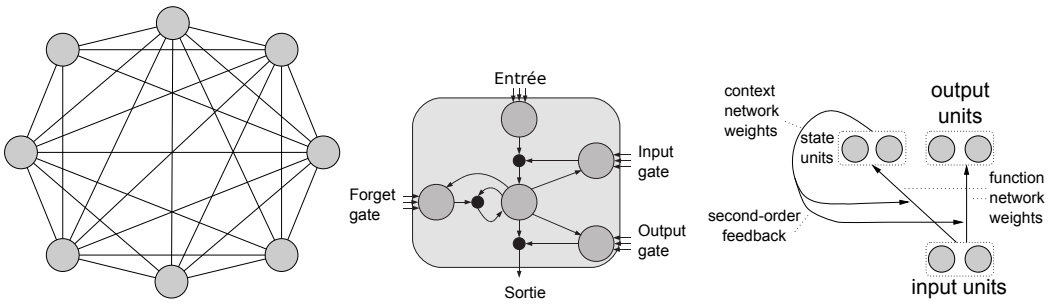
Ces catégories ne sont pas mutuellement exclusives.

3.1.3 Modèlisation de la mémoire

3.1.3.1 Réseaux de neurones

De nombreux réseaux de neurones ont été proposés afin de modéliser différentes types de mémoire (voir figure 3.3 pour une illustration) :

Réseau Hopfield (Hopfield, 1982). Ce type de réseau est dit associatif. Il est basé sur un réseau de neurones de type McCulloch et Pitts (1943) avec tous les neurones connectés entre eux sans auto-connexion. Ce modèle peut être utilisé comme mémoire associative : dans ce cas une forme mémorisée est retrouvée par



Hopfield *LSTM* *Sequential cascaded network*

FIGURE 3.3 – Différents types de réseaux de neurones récurrents.

une stabilisation du réseau s'il a été stimulé par une partie adéquate de cette forme.

Réseau Elman (Elman, 1990) et de Jordan (Jordan, 1986). C'est une classe simple de réseaux récurrents. Le réseau de Jordan utilise un vecteur d'état qui contient les copies des activations de la couche de sortie du réseau l'instant précédent. Le réseau de Elman utilise les copies des activations de la couches cachées l'instant précédent pour le vecteur d'état. Dans ces deux types de réseaux le vecteurs d'état permet au réseau de stocker des informations sur les entrées précédentes. C'est pourquoi ils sont utilisés dans la prédiction de séries temporelles, les tâches de classification, l'apprentissage de scénarios. Par rapport au réseau de type feedforward qui travaille avec des fenêtres temporelles, ils ont l'avantage de tenir compte implicitement du contexte temporel, il est donc capable d'apprendre des associations aussi bien temporelles que spatiales (voir figure 2.10).

Sequential Cascaded Network (SCN) (Pollack, 1991). Dans cette architecture les poids de connexion (et/ou le biais) sont modulés dynamiquement. Le SCN consiste en : une fonction réseau qui fait correspondre les entrées aux sorties et unités d'états en utilisant une couche de poids de connexion (les poids de la fonction réseau) ; un réseau contexte qui prend comme entrée les valeurs d'activation de l'unité d'état et produit comme sortie les poids de la fonction réseau du pas de temps suivant. Ziemke (2001) ont utilisé une version étendue de ce modèle (Extended Sequential Cascaded Network (ESCN)) sur des tâches de mémoire en robotique évolutionniste.

LSTM (Hochreiter et Schmidhuber, 1997). Un neurone à large mémoire court-terme ou LSTM (Long Short-Term Memory) peut être vu comme un neurone ayant, en plus des connexions externes, une connexion auto-récurrente de coefficient constant égal à 1. Ceci permet de sauvegarder d'une itération à l'autre les états successifs du neurone. Des portes multiplicatives au niveau de l'entrée (input gate) et de la sortie (output gate) permettent de protéger respectivement l'état actuel de la mémoire et celui du reste du réseau.

3.1.4 La mémoire dans les systèmes cognitifs synthétiques

Une forme de mémoire dans un système cognitif synthétique est nécessaire dans différentes classes de problèmes. En apprentissage par renforcement par exemple, on évoque la nécessité d'avoir une capacité de mémoire lorsqu'il y a un problème d'ambigüité des perceptions (*perceptual aliasing* (Whitehead et Ballard, 1991)). Ce problème désigne l'incapacité d'un système de perception à distinguer de manière unique tous les lieux d'un environnement. Il apparaît lorsque des situations multiples ne sont pas distinguables des perceptions immédiates et requièrent des réponses différentes du système. On peut illustrer le problème d'ambigüité des perceptions avec l'exemple suivant : un agent fait face à une boîte fermée et l'action qu'il doit effectuer dépend du contenu de la boîte. Si son comportement est seulement basé sur ses perceptions immédiates, l'agent n'a pas moyen d'agir de manière optimale.

Différentes méthodes ont été proposées en RL pour intégrer une forme de mémoire. Notamment pour retenir un stimulus dans un labyrinthe en T discret (voir section 3.3 pour une description de la tâche). Pour retenir le stimulus, une approche utilisant une fenêtre de mémoire de taille, fixe (Mitchell et Lin, 1993) ou variable (McCallum, 1996), retenant les observations et actions les plus récentes a été proposée. D'autres méthodes utilisent des réseaux de neurones tel que les LSTM (Bakker, 2002; Gomez, 2005).

En robotique évolutionniste la notion d'ambigüité des capteurs ou Sensory aliasing, proche de la notion d'ambigüité des perceptions a été introduite par Nolfi (2001). Cette ambigüité des capteurs fait référence à une situation dans laquelle, du point de vue de l'agent, plus de deux états agent/environnement correspondent au même modèle sensoriel, mais requièrent des réponses motrices différentes. Une façon de résoudre ce problème est de conserver une trace des capteurs affectés par le problème d'ambigüité des capteurs.

Pour ce faire, différentes solutions ont été proposées :

- utilisation de réseaux de neurones récurrents ;
- utilisation de modulation synaptique.

Utilisation de réseaux de neurones récurrents. En incluant des connexions récurrentes, ces réseaux peuvent produire des réponses qui peuvent prendre en compte à la fois les états courants et précédent des capteurs. Les réseaux de Jordan (Jordan, 1986) ou d'Elman (Elman, 1990) ont été appliqués avec succès sur différentes tâches de navigation dans lesquelles un robot devait périodiquement retourner dans une aire pour se recharger (Nolfi et al., 1994). D'autres tentatives ont été conduites en utilisant des réseaux de neurones récurrents continus (CTRNN) (Beer et Gallagher, 1992). En s'appuyant sur des équations différentielles au lieu d'être mis à jour à des pas de temps fixe, ces réseaux peuvent produire des dynamiques continues. Ce type de réseaux a été utilisé avec succès sur différentes tâches telles que des tâches de locomotion (Gallagher et al., 1996) de navigation guidée visuellement (Harvey et al., 1994)).

Utilisation de plasticité synaptique. Ziemke et Thieme (2002) montrent notamment que des tâches nécessitant une forme de mémoire à court terme peuvent être résolues via une modulation des poids synaptiques. Cette méthode est appliquée avec succès sur une tâche de panneau de signalisation (voir par. 2.3.3 pour plus de détails). Différentes tâches intégrant une forme de mémoire ont été décrites précédemment à la section 2.3.3.

3.1.5 Défis

En ER, faire émerger des agents dotés d'une capacité de mémoire pose deux problèmes.

Tendance à l'obtention d'individus aux comportements réactifs. Il y a forte tendance en ER à obtenir des individus réactifs (Nolfi, 2001; Nelson et al., 2009). Il est de plus difficile en ER de développer plusieurs capacités en même temps (Nolfi, 2001), comme par exemple développer une capacité de mémoire tout en étant capable de se mouvoir dans un labyrinthe.

Difficulté à déterminer si un individu est doté de mémoire. Pour détecter qu'un agent possède une capacité de mémoire, il faut vérifier si celui-ci possède une représentation interne. Il est difficile d'identifier de façon claire si un état interne a un statut représentationnel ou si c'est simplement le reflet accidentel d'une corrélation avec un élément externe (Clark, 1997). L'interaction entre un individu et l'environnement externe peut être proprement décrit comme un processus de causalité réciproque continue (Clark, 1997). Dans ce cas, isoler un individu et l'environnement et essayer d'identifier quelle est la relation entre les états internes de l'agent et les features de l'environnement est parfois difficilement possible.

3.2 Expérience liminaire

Afin de mesurer la difficulté de générer des contrôleurs dotés de capacité de mémoire nous allons utiliser une tâche issu de l'état de l'art : La tâche panneau de signalisation (Ulbricht, 1996; Rylatt et Czarnecki, 2000; Ziemke et Thieme, 2002) : un robot navigue dans un labyrinthe en T et doit, suivant un stimulus rencontré pendant la traversée du premier corridor (lumière, son, ...), tourner dans une direction donnée à la jonction. Le dispositif expérimental, ainsi que le type de robot utilisé sont indiqués sur la figure 3.4.

Dans cette tâche un stimulus situé au départ du corridor doit être maintenu en mémoire afin de tourner vers le bon côté.

Notre objectif ici est de vérifier que les individus réussissant cette tâche sont dotés d'une capacité de mémoire. Pour cela, nous allons catégoriser les différents types d'individus obtenus et étudier l'influence de différents paramètres sur l'émergence de mémoire :

– influence du codage ;

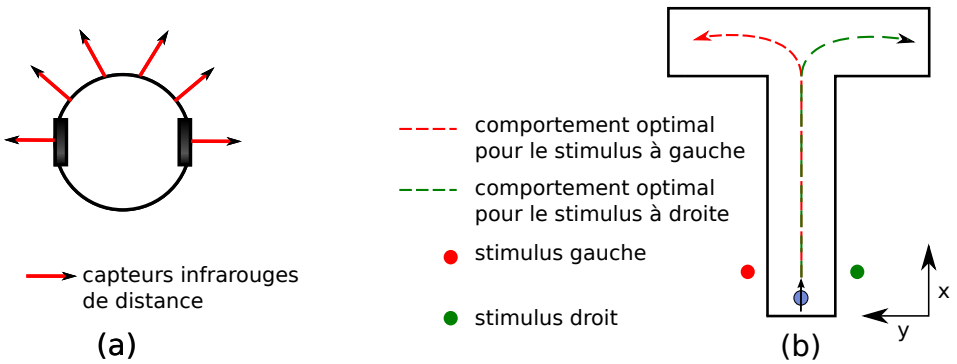


FIGURE 3.4 – (a) Robot mobile utilisé pour la tâche de navigation dans un labyrinthe en T simulé. En plus des capteurs indiqués sur le schéma, le robot dispose de deux capteurs : un pour détecter la lumière à gauche, un autre pour détecter la lumière à droite. (b) Carte du labyrinthe en T et indication du comportement optimal recherché par séquence de stimuli.

– influence du nombre de contextes utilisé durant l'optimisation ;

Un contexte désigne un ensemble de paramètres de l'environnement : position et angle de départ du robot, taille de la carte, ...

3.2.1 Catégorisation de différents types d'agents robotiques

Il est difficile en ER de caractériser le comportement des agents que l'on obtient ou que l'on souhaite obtenir par évolution. Certains termes sont évoqués comme comportement cognitif minimal (Beer, 1996), complexe (Nelson et al., 2009) sans pour autant être clairement définis et formalisés. Nous tenterons de décrire ici quelques catégories d'individus. Il est nécessaire au préalable de définir quelques notions que nous réutiliserons tout au long de ce manuscrit (ces notions sont inspirés de Nolfi (2001)).

Définition 14 (Etat interne) *Un état interne est un état (e.g. l'activation d'un état d'un neurone interne du système de contrôle de l'agent) qui peut être affecté par les états sensorimoteurs précédents d'un agent et qui peut co-déterminer, avec les états courant des capteurs, les actions motrices de l'agent.*

Définition 15 (Agent réactif) *Un agent réactif indique un agent qui n'a pas d'état interne et dans lequel l'action motrice courante est seulement fonction de l'état courant des capteurs(Nolfi, 2001).*

Définition 16 (Agent doté d'une capacité de mémoire interne) *Agent qui possède au moins une représentation interne pouvant co-déterminer au moment opportun avec les états courant des capteurs, les actions motrices de l'agent.*

Définition 17 (Agent doté d'une capacité de mémoire externe) *Désigne un agent qui au lieu d'utiliser une mémoire interne, utilise l'environnement pour*

stocker de l'information. Par exemple sa position par rapport à l'environnement peut servir de mémoire externe.

3.2.2 Protocole expérimental

Le contrôle du robot est assuré par un réseau de neurones. Les neurones à optimiser sont des intégrateurs à fuite ayant des propriétés de contractions : des lPDS (Girard et al., 2008) (voir section 2.2.1.2 pour de plus de détails).

3.2.2.1 Etude de l'influence du codage

Afin d'étudier l'influence du codage sur l'émergence de mémoire, différents types de contrôleurs seront optimisés :

- Optimisation des paramètres du réseau (*Elman*) : dans ce cas, la topologie du réseau de neurones récurrent de type Elman (Elman, 1990) reste fixe et seuls les paramètres du réseau sont optimisés (voir figure 2.10).
- Optimisation de la topologie et des paramètres du réseau :
  - Réseau *DNN* : la topologie et les paramètres sont optimisés, et les connexions récurrentes sont autorisées.
  - Réseau *DNN* Feed-Forward (*FF*) : la topologie et les paramètres sont optimisés mais les connexions récurrentes ne sont pas autorisées.

Codage DNN. Le codage *DNN* (pour Direct Neural Network) est un codage direct simple (Mouret et Doncieux, 2009b) inspiré par NEAT (Stanley et Miikkulainen, 2002). Dans celui-ci, un réseau de neurones est décrit comme un graphe orienté et cinq opérateurs de mutations sont implémentés (aucun opérateur de croisement n'est défini) :

- ajout d'une connexion ;
- retrait d'une connexion ;
- ajout d'un neurone ;
- retrait d'un neurone ;
- modification des paramètres (poids ou biais) d'une connexion par mutation polynômiale (Deb, 2001).

Plus de détails peuvent être trouvés dans (Mouret et Doncieux, 2009b), ce codage a été employé dans plusieurs papiers (Doncieux et al., 2009; Mouret et Doncieux, 2009a; Doncieux et Mouret, 2010a).

Les paramètres liés à l'encodage des réseaux de neurones utilisés pour le contrôle sont répertoriés dans la table 3.1.

3.2.2.2 Caractérisation des individus dotés d'une forme de mémoire interne

Afin de caractériser les individus possiblement doté d'une forme de mémoire interne, nous nous basons sur la propriété d'*echo state* définie par Jaeger (2002). Un réseau possède une propriété d'*echo state* si celui-ci élimine asymptotiquement

TABLE 3.1 – Paramètres liés à l'encodage des réseaux de neurones.

Paramètres	Valeurs
nb. min./max. de neurones	1 / 20
nb. min/max. de connexions	1 / 50
prob. d'ajouter un neurone	0.05
prob. d'enlever un neurone	0.05
prob. d'ajouter une connexion	0.05
prob. d'enlever une connexion	0.05
prob. de modifier un poids/biais	0.05

toute information provenant des conditions initiales. Vu que nous nous intéressons ici aux réseaux capables de maintenir des informations provenant des entrées, nous désirons détecter les réseaux qui ne possèdent pas cette propriété d'*echo state*.

Nous définissons N qui représente les neurones internes, K les neurones d'entrées, $W = (w_{ij})$ représente la matrice des poids $N * N$ correspondant aux connexions internes. σ_{max} représente la plus grande valeur singulière de la matrice W. L'activation des neurones d'entrée au pas de temps n correspond à :

$$u(n) = (u_1(n), \ldots, u_K(n))$$

l'état x, représente l'activation des neurones internes :

$$x(n) = (x_1(n), \ldots, x_N(n))$$

Jaeger (2002) prouve qu'un réseau possède cette propriété d'*echo state* pour toutes les entrées u, pour tous les états $x, x' \in [-1, 1]^N$, dans le cas où la matrice de poids W satisfait $\sigma_{max} < 1$.

Les réseaux ayant potentiellement une forme de mémoire sont donc ceux ayant une valeur singulière $\sigma_{max} > 1$ au sein de la matrice W.

Comme nous sommes dans une matrice carré :

$$\exists |\lambda_{max}| > 1 \implies \exists \sigma_{max} > 1$$

avec λ_{max} représentant la valeur propre dont la valeur absolue est maximale dans la matrice W.

Dans notre cas, les valeurs propres de la matrice W sont approximées en utilisant la librairie gsl [1]. Le calcul via cette approximation permet de nous abstraire des spécificités inhérentes à la décomposition en valeurs propres qui ne fonctionne seulement que pour certaines matrices carrées.

1. http ://www.gnu.org/software/gsl/

TABLE 3.2 – liste des différentes configurations utilisés

Configuration	Description
DNN+UNI	Opt. topologie et param. du réseau + évaluation dans un contexte
FF+UNI	Opt. topologie et param. du réseau Feed-Forward + évaluation dans un contexte
ELMAN+UNI	Opt. param. du réseau Elman + évaluation dans un contexte
DNN+MULTI	Opt. topologie et params du réseau + évaluation dans plusieurs contextes
FF+MULTI	Opt. topologie et params du réseau Feed-Forward + évaluation dans plusieurs contextes
ELMAN+MULTI	Opt. param. du réseau Elman + évaluation dans plusieurs contextes

3.2.2.3 Influence du nombre de contextes

Par contexte, nous désignons un ensemble de paramètres lié à l'environnement dans lequel l'agent évolue. Dans cette expérience un contexte est défini par : la position initiale du robot ; l'angle initial du robot ; la taille de l'environnement.

Nous étudierons l'influence du nombre de contextes sur le type de solutions obtenues. Pour cela nous optimiserons des individus sur différentes configurations :

- un seul contexte *UNI* ;
- plusieurs contextes *MULTI* : dans ce cas douze contextes différents sont utilisés pendant l'optimisation où la position et l'angle de départ sont modifiés.

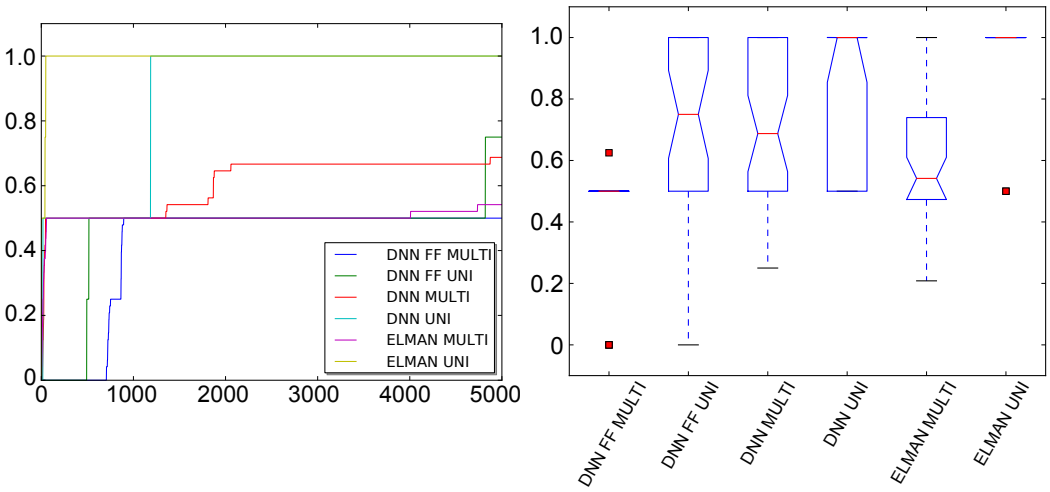
une liste des différentes configurations est présente au tableau 3.2

Cette expérience utilise l'algorithme évolutionniste multi-objectif élitiste NSGA-II (Deb, 2001), qui repose sur une sélection par tournoi et une méthode de classement des individus basée sur la relation de dominance (cf. partie 2.1.4.3 pour plus de détails). Ce travail a été implémenté dans le framework Sferes_{v2} (Mouret et Doncieux, 2010). Chacune des configurations est répétée trente fois et comporte 5000 générations.

3.2.3 Expérience : panneau de signalisation

Contexte expérimental L'agent considéré est un robot simulé à 2 deux roues et 8 capteurs : 6 capteurs de distance et 2 capteurs permettant de détecter de quel côté se situe la source lumineuse. Le robot contrôle sa vitesse par l'intermédiaire de 2 neurones de sortie correspondant à la vitesse de ses moteurs gauche et droit. Le robot doit se diriger vers la gauche lorsque le stimulus est à gauche, et à droite si le stimulus est à droite. La totalité des expériences s'effectue dans un simulateur 2D sans friction ni glissement.

Fitness. Le robot est évalué sur l'ensemble $C = \{g, d\}$ avec g et d correspondant respectivement au stimulus placé à gauche et à droite du corridor. Durant l'évaluation afin d'éviter le surentraînement, un individu doit atteindre le bon côté du labyrinthe dans $L = 12$ contextes différents. Un contexte, ici, est défini par une position et un angle initial de départ pour le robot.



(a) *(b)*

FIGURE 3.5 – Résultats obtenus sur l'expérience panneau de signalisation. (a) Evolution de médiane de la performance des meilleurs individus durant le processus d'optimisation. (b) boîte à moustache pour chaque configuration des meilleurs individus avec 30 runs par configuration.

$$f(x) = \frac{1}{2 * L * |C|} \sum_{i=0}^{L=12} \sum_{j \in C} \left(\begin{cases} 1 \text{ if } (p_{ij}(T) \in G \text{ and } j = g) \\ 1 \text{ if } (p_{ij}(T) \in D \text{ and } j = d) \end{cases} \right) \tag{3.1}$$

G et D correspondent respectivement à l'ensemble des positions à gauche et à droite du corridor du labyrinthe en T, avec $G \cap D = \varnothing$. $p_{ij}(T)$ correspond à la position finale de l'agent à la fin $(t = T)$ de la séquence j du contexte i. La constante 2 correspond à la note maximale quand l'agent se dirige correctement du bon côté du labyrinthe pour les stimuli g et d. Elle est utilisée pour normaliser la valeur de la fitness dans l'intervalle $[0, 1]$. L'agent dispose de 200 pas de temps pour se déplacer dans le labyrinthe.

3.2.3.1 Résultats

Nous allons nous intéresser à différents points :
- aux performances des différentes configurations ;
- a la catégorisation des agents obtenus ;
- aux trajectoires des différents agents.

Performance. La figure 3.5 illustre les performances obtenus pour chaque configuration. A gauche, nous pouvons voir l'évolution de la performance des meilleurs individus durant le processus d'optimisation. Chaque courbe correspondant à la médiane sur 30 runs des meilleurs individus. Le graphique à droite représente les performances des meilleurs individus de chaque configuration.

Nous pouvons constater qu'évaluer un individu dans plusieurs contextes impacte grandement les performances et ce quelle que soit la configuration utilisée. Dans le cas des réseaux FeedForward (*FF*) la médiane passe de 0.75 en lors de l'évaluation dans un seul contexte (*UNI*) à 0.5 lors de l'évaluation dans plusieurs contextes (*MULTI*). Nous retrouvons le même type de baisse pour la configuration *DNN* où la médiane passe de 1 pour *UNI* à 0.67 pour *MULTI*, ainsi que pour *Elman* (de 1 pour *UNI* à 0.57).

DNN obtient les meilleurs résultats avec une médiane de 0.68 contre 0.57 pour la structure fixe Elman et 0.5 avec le réseau Feed-Forward. Une fitness de 0.5 correspond au minimum local attractif consistant à tourner du même côté quel que soit le stimulus de départ. Si l'on regarde l'évolution de la fitness pendant le processus d'optimisation (graphique de gauche de la figure 3.5), nous constatons qu'il faut seulement quelques générations que ce soit avec un réseau à structure fixe de type Elman ou *DNN*. pour atteindre ce minimum local. Pour des réseaux FF, ce minimum local est atteint bien plus tardivement après 1000 générations, et même après 6000 générations, la médiane reste bloquée à ce niveau. Pour DNN et Elman la sortie de ce minimum se fait après respectivement 1500 et 4000 générations.

Catégorisation des individus obtenus. Le graphique 3.6 montre les différentes performances des meilleurs individus pour les configurations *DNN* et *Elman*. La barre rouge indique le pourcentage d'individus ayant obtenu la fitness maximum, la barre bleue le pourcentage d'individus ayant obtenu la fitness maximum et réussi le test de mémoire. La barre noire indique les individus ayant réussi le test de mémoire sans prendre en considération la fitness obtenue. Nous pouvons constater qu'avec *Elman*, le fait de posséder une capacité de mémoire ne se traduit pas par de bonnes performances sur le contexte *MULTI* : moins de 20% des agents sont capables de résoudre la tâche dans ce cas là. Pour le codage *DNN* nous pouvons constater qu'un grand nombre d'individus réussissent la tâche sans pour autant développer une forme de mémoire. Seulement 40% des individus réussissent la tâche en développant une capacité de mémoire dans le cas *UNI* et 8% des individus dans le cas *MULTI*.

Trajectoire. Nous constatons que les trajectoires obtenues par les agents réussissant la tâche à la figure 3.7 sont assez éloignées d'une trajectoire que l'on aurait imaginée a priori : trajectoire rectiligne dans le corridor, puis un changement de direction seulement lorsqu'un choix s'impose. L'agent exploite en fait l'environnement afin d'éviter d'avoir une représentation interne d'un stimulus précédemment vu. Cette tâche de panneau de signalisation peut donc être résolue par un agent qui ne dispose pas de représentation interne du stimulus de départ. Il peut externaliser cette mémoire en utilisant l'environnement. On peut voir différentes illustrations sur les figures 3.7, où l'on peut constater que dès le départ, l'agent se positionne d'une façon telle que sa trajectoire lui permet d'atteindre le bon côté du labyrinthe en T sans avoir besoin de mémoriser le stimulus de départ.

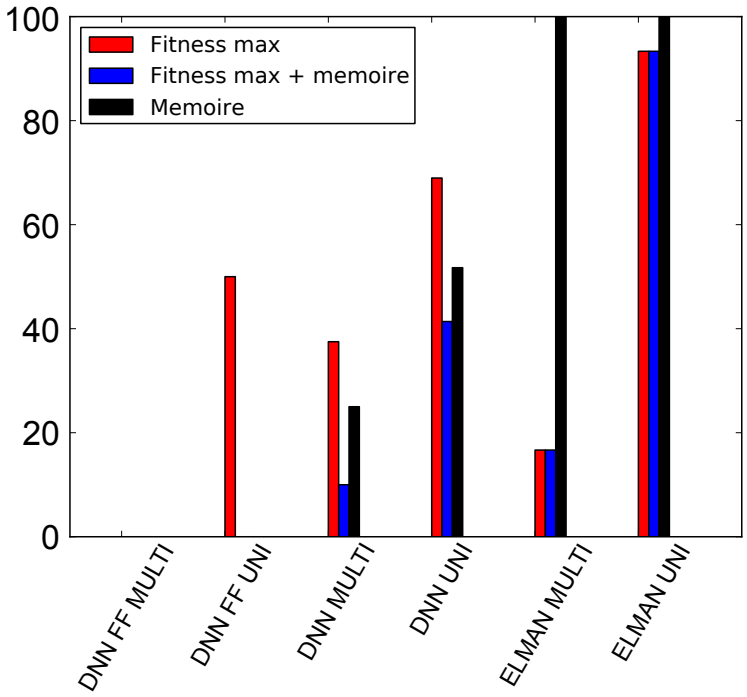


FIGURE 3.6 – Graphique représentant différentes performances des meilleurs individus pour les configurations *DNN* et *Elman*. La barre rouge indique le pourcentage d'individus ayant obtenu la fitness maximum, La barre bleue le pourcentage d'individus ayant obtenu la fitness maximum et réussi le test de mémoire La barre noire indique les individus ayant réussi le test de mémoire sans prendre en considération la fitness obtenue.

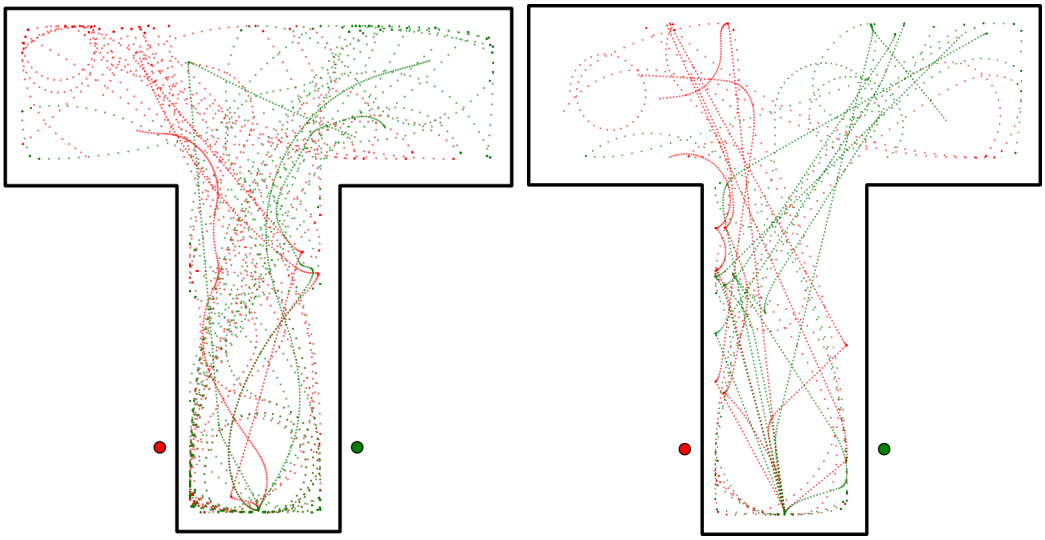


FIGURE 3.7 – Ensemble des trajectoires des meilleurs individus sur la tâche de panneau de signalisation. Les trajectoires en rouge représente les trajectoires effectuées lorsque le stimulus est à gauche. Les trajectoires vertes lorsque le stimulus est à droite.

Ces résultats confirment l'hypothèse qu'il est plus aisé d'obtenir des individus réactifs.

3.2.4 Synthèse

Cette expérience liminaire a permis de confirmer notre hypothèse sur le caractère trompeur de l'obtention d'agent doté de capacité de mémoire. La tâche de panneau de signalisation, n'a pas permis d'obtenir avec succès un nombre satisfaisant d'agents dotés de mémoire. Nous avons pu constater que le processus d'optimisation tend à trouver des individus qui exploitent l'environnement afin d'externaliser la capacité de mémoire, par exemple en se positionnant d'une manière particulière dès que le stimulus apparait et évite de ce fait la nécessité d'avoir recours à une représentation interne du stimulus.

Afin de faire émerger efficacement une forme de mémoire au sein d'un agent, il semble donc important de définir un protocole expérimental différent où une capacité de mémoire est indispensable. Une des solutions possibles est de s'inspirer des différents protocoles de tâches de mémoire qui ont été développés au sein des neurosciences.

3.3 Tâches impliquant la mémoire de travail

Afin de mettre en évidence les zones du cerveau impliquées lors d'un processus de mémoire, différentes tâches ont été développées par les neuroscientifiques, no-

TABLE 3.3 – Différents environnements utilisés chez le rat afin de tester sa mémoire

Environnement	Description
Labyrinthe classique	Consiste en une large plateforme avec une série de murs verticaux. Le rat part d'un lieu, se déplace à travers le labyrinthe jusqu'à un autre emplacement où il perçoit une récompense.
Labyrinthe en T	Le labyrinthe forme un T. L'animal part de la base du T. Une récompense peut être placée dans un des bras du labyrinthe.
Labyrinthe en T multiple	Extension du labyrinthe en T. Dans ce cas plusieurs labyrinthes en T se suivent.
Le labyrinthe en Y	Similaire au labyrinthe en T, mais possède trois bras identiques. Le rat démarre à la fin d'un bras, puis choisit l'un des deux autres. Le labyrinthe en Y est un peu plus simple que le labyrinthe en T car les virages progressifs réduisent le temps d'apprentissage par rapport aux virages brusques du labyrinthe en T.
Labyrinthe radial à bras	Contient une plateforme centrale et entre 3 et 48 bras (Olton et Samuelson, 1976). A la fin de chaque bras se trouve de la nourriture, non visible de la plateforme centrale. le rat est placé sur la plateforme centrale. Le rat doit visiter chaque bras et manger la nourriture. si l'animal a une bonne mémoire, il ne devrait pas retourner dans les bras qu'il a déjà visité.
La piscine de Morris	C'est un large bac rond rempli d'eau opaque avec deux petites plateformes cachées sous la surface de l'eau. Le rat est placé sur une plateforme de départ et doit nager jusqu'à ce qu'il trouve l'autre plateforme où se poser. Des signaux extérieurs sont placés autour de la piscine pour aider le rat à apprendre où se trouve la plateforme d'arrivée.

tamment pour des rats ou des singes. Nous allons faire un bref aperçu de celles-ci. Les tâches impliquant une forme de mémoire à court terme ou de mémoire de travail sont regroupées dans ce que l'on définit comme des tâches à réponses différées ou "Delayed-Response Tasks" (DR). Le tableau 3.3 donne une liste de différents environnements utilisés chez le rat pour pour tester sa mémoire. Pour un descriptif des tâches à réponses différées chez le singe se référer à (Buccafusco, 2008).

Le modèle de (Zipser, 1991) (le même modèle a été proposé avec des neurones à spikes (Zipser et al., 1993)) montre qu'un mécanisme utilisant seulement l'activité dynamique dans un réseau de neurones est suffisant pour réaliser des tâches de mémoire à court terme. Dans cette expérience, le réseau possède deux entrées, une entrée analogique à mémoriser et une entrée binaire déterminant à quel moment le réseau doit mémoriser l'entrée analogique et une sortie. Le réseau est entrainé via backpropagation (BPTT) pour que le neurone de sortie maintienne la valeur présente au niveau de l'entrée analogique à chaque fois que l'entrée binaire est activée. Cette valeur de sortie doit être maintenue quels que soit les changements dans l'entrée analogique jusqu'à ce que l'entrée binaire soit de nouveau active. Les patterns d'activités des réseaux obtenus ont été comparés avec succès à différentes données electrophysiologiques.

Tâches d'appariement différé (DMS). Cette tâche (Delayed Matching to Sample Tasks) nécessite que le sujet se souvienne d'un stimulus pendant une période durant laquelle le stimulus n'est plus présent. Après cette période, on présente au su-

jet le stimulus précédent et un stimulus alternatif. Le sujet obtient une récompense lorsqu'il sélectionne le stimulus précédent. De nombreux modèles ont été proposés (Dehaene et al., 1989; Moody et al., 1998; Gisiger et al., 2005) et comparés à différentes données électrophysiologiques. Le modèle de Dehaene et al. (1989) est comparé aux données électrophysiologiques d'enregistrements de neurones du PFC de singes réalisant la tâche (Fuster, 1984). Celui de Moody et al. (1998) est comparé avec des enregistrements du PFC issu des expériences de (di Pellegrino et Wise, 1993).

Tâche oculomotrice à réponse différée. Il existe différents types de tâches oculomotrices à réponse différée (Oculo-motor delayed-response task ou OMDR, ODR) le sujet fixe une lumière sur un écran. Une autre lumière, la cible, est brièvement visible sur une position variable dans la périphérie. Après un délai la lumière de fixation est retirée, signalant que la saccade doit être effectuée à la position de la cible à mémoriser. Ce type de tâche a été notamment modélisé via un réseau de neurones par Droulez et Berthoz (1991).

Tâche de saccades séquentielles différées (DSS). (Delayed Sequential Saccade Tasks) Une séquence de cibles défile et disparait. Le sujet doit maintenir la séquence en mémoire. Après un délai le sujet doit effectuer une saccade oculaire pour chacune des cibles dans l'ordre à laquelle elles ont été présentées. Mitchell et Zipser (2003) proposent un modèle du champs oculomoteur frontal (Frontal eye field) capable d'effectuer des saccades oculaires après avoir mémorisé une séquence de cible apparaissant à des positions arbitraires.

Tâche de discrimination à deux intervalles (2AFC). Dans ce type de tâche (two-interval-discrimination task), le sujet observe un premier stimulus, un son par exemple. Une période de délai est observée durant quelques secondes. Ensuite un second stimulus est présenté avec une fréquence différente. Le sujet doit faire un choix et déterminer si le deuxième stimulus possède une fréquence plus ou moins élevée que le premier stimulus.

Différents modèles bas niveau notamment celui de (Machens et al., 2005) qui présente un modèle basé sur des méchanismes biophysiques réalistes et des données neurophysiologiques récentes pouvant simuler de la mémoire de travail et des decisions binaires. Miller et Wang (2006) proposent une alternative à ce modèle. Jun et al. (2010) ont comparé ces deux modèles à des enregistrements de neurones à spike au sein du PFC lorsque des singes réalisaient une tâche à choix forcé, aucun des types de cellules prédits par les deux modèles n'étaient représentés significativement dans la population.

Tâche AX-CPT. La tâche AX-CPT est une version modifiée du Continuous Performance Test (CPT) (Rosvold et al., 1956). On peut la décrire de la manière suivante : des séquences de lettres sont présentées sur un écran et apparaissent les

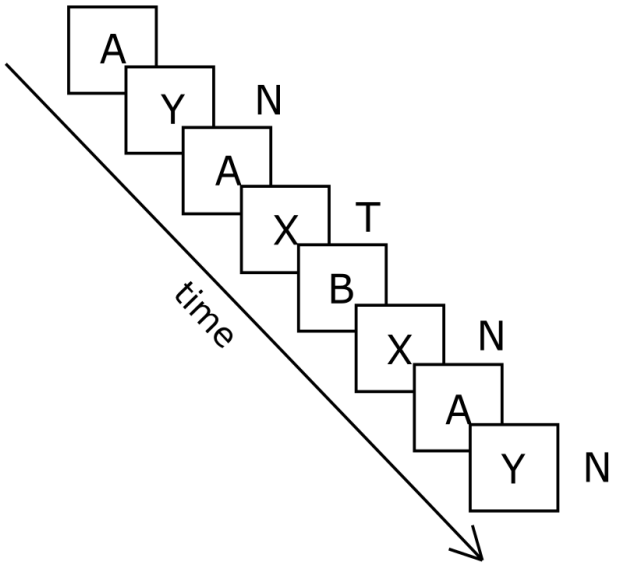


FIGURE 3.8 – Description de la tâche AX-CPT (Braver et al., 1995) : La séquence A suivi de Y ainsi que BY, BX, représente les essais "Non-cible", A suivi de X représente l'essai "cible". La séquence AX est présente dans 70% des cas. T indique lorsque la réponse doit être "cible", N lorsqu'elle correspond à "non-cible".

unes après les autres sous la forme d'une série de paires de lettres. La consigne est d'indiquer si, oui ou non, la paire cible (la lettre A suivie de la lettre X) est présentée à l'écran. Le sujet doit donc répondre "oui" lorsque la lettre X est immédiatement précédée de la lettre A et "non" dans tous les autres cas. Les essais "Non cible" sont les essais AY, BY et BX. La cible AX étant présente majoritairement au sein d'un bloc d'essais, la paire AX devient donc le contexte informationnel dominant pour le sujet qui effectue la tâche.

Introduit par Braver et al. (1995), ce paradigme est devenu un protocole standard pour étudier les syndromes impliquant un dysfonctionnement du PFC tel que la schizophrénie (Braver et al., 1999) ou évaluer l'effet de l'âge sur les performances (Braver et Barch, 2002). Différents modèles de haut niveau (Braver et al., 1995, 1999; Braver et Barch, 2002) ont été créé pour faire des prédictions en observant les performances comportementales des sujets. Le premier (Braver et al., 1995) est un modèle simple du PFC, les modèles suivant (Braver et al., 1999; Braver et Barch, 2002) essayent de définir un modèle de contrôle cognitif [2] qui simule l'interaction entre le PFC et la dopamine.

2. Le contrôle cognitif peut être défini comme la capacité à focaliser son attention sur une information pertinente à la tâche en cours et à inhiber les informations non pertinentes. Le contrôle cognitif est sollicité quotidiennement, par exemple lorsqu'on doit suivre une conversation dans un milieu bruyant.

Modèles	Tâches	Cerveau*	Réseaux de neurones	Neurones
Zipser (1991)	DR	LIP PFC	13 N / 120 Con	
Zipser et al. (1993)	DR		13 N / 120 Cn	Spike
Minami et Inui (2001)	DR	PFC	5 sets 4 Neurons	
Droulez et Berthoz (1991)	ODR	LIP	50 Mod 3 N/961Mod 4N	Spike
Mitchell et Zipser (2003)	DSS	FEF	4 arrays 32 N /19640 C	Spike
Dehaene et al. (1989)	DMS	PFC	50 Clust * 100 N	Spike
Moody et al. (1998)	DMS	PFC PMC	10-60 N	
Gisiger et al. (2005)	DMS		900 Neurons	Spike
Machens et al. (2005)	2AFC	Front Lobe	2*250 Neuron	Spike
Miller et Wang (2006)	2AFC	PFC	12 Set*2*(400+100) N	Spike
Braver et al. (1995)	CPT-AX	PFC	??	
O'Reilly et Frank (2006)	12-AX	PFC, BG	20 lay 400 Neurons	Spike

* Représente les parties du cerveau simulée dans le modèle

12-AX. Cette tâche est une extension de la tâche AX-CPT proposée par (O'Reilly et Frank, 2006). Dans celle-ci on présente une série continue de lettres et de chiffres individuels (par exemple, 1, B, Y, A, X, 2, B, C, Y). Le sujet doit répondre à l'occurrence des cibles ("X"ou"Y") à la fois en fonction de la lettre qui les précède immédiatement (respectivement" A"ou"B"), mais aussi en fonction du dernier chiffre qui a été présenté. Si le dernier chiffre présenté était un "1", il faudra répondre seulement aux "X" précédés d'un "A". Si le dernier chiffre présenté était un "2", il faudra répondre aux "Y" précédés d'un "B". Les stimuli distracteurs comme les "C" doivent être ignorés. Par conséquent, la représentation de la lettre ("X"ou"Y") associée à une réponse ne doit être mise à jour que sous certaines conditions (respectivement"1–A"ou"2–B"). De plus, la représentation du dernier chiffre présenté, doit être maintenue pendant que la représentation de la lettre la plus récente doit être mise à jour.

O'Reilly et Frank (2006) proposent un modèle du PFC et les ganglions de la base pouvant réaliser la tâche 12-AX. Ce modèle repose sur l'hypothèse que les ganglions de la base ont pour rôle de sélectionner les stimulus à mémoriser et que le PFC sert à mémoriser ces informations.

Tâche d'alternance différée (DA). Dans cette tâche (delayed alternation task), une réponse correcte dépend d'une réponse précédente. C'est une tâche à deux choix dans laquelle le sujet doit choisir alternativement deux objets pour obtenir une récompense.

3.3.1 Synthèse

Nous avons pu voir que de nombreuses tâches différentes ont été développées afin de mettre en évidence des capacités de mémoire, telles que des tâches oculomotrices ou des tâches de navigation dans un labyrinthe.

De ce bref descriptif des tâches de mémoire de travail ainsi que des modèles computationnelles associés à ceux-ci, nous pouvons retenir ceci :

– Les différentes tâches développées pour des rats ou des souris sont aisément reproductibles avec un agent robotique de type e-puck ;

– De nombreux modèles computationnelles sont disponibles prouvant qu'il est possible de développer un réseau de neurones capable de réaliser ces différentes types de tâches à réponses différées.

Pour la tâche de mémoire de travail, notre choix s'est porté sur la tâche AX-CPT et ceci pour plusieurs raisons :

– Cette tâche a été modélisée par un réseau de neurones assez simple ((Braver et al., 1995)) ;
– C'est une tâche de référence utilisée pour mettre en évidence des déficits au sein de la mémoire de travail (Perlstein et al., 2003) ;
– elle a été adaptée chez les rats (Maes et al., 2001), donc facilement adaptable pour un agent robotique mobile.

Pour l'environnement, notre choix s'est porté sur le labyrinthe en T, afin de reprendre le protocole décrit par (Maes et al., 2001). Celui-ci se déroule de la manière suivante :

– Présentation de la séquence de lettres (A ou B puis X ou Y), l'individu ne peut pas bouger ;
– Après la présentation de la séquence de lettres, l'individu doit se diriger du bon côté du labyrinthe : à gauche pour la sequence AX, à droite pour les autres séquences.

3.4 Expériences et résultats

3.4.1 Contexte expérimental

L'objectif de ces expériences est de vérifier l'impact de l'utilisation du protocole AX-CPT sur l'émergence de capacité de mémoire au sein d'un agent.

Pour cela nous utilisons deux tâches robotiques différentes intégrant le protocole AX-CPT (décrit à la section 3.3) :

– une tâche de pointage utilisant un robot humanoïde, le robot Icub ;
– la tâche de "panneau de signalisation".

Pour les deux tâches, nous utilisons le protocole suivant : une séquence de lettres est présentée à l'agent, A ou B, suivi par X ou Y après un délai. Les moteurs sont désactivés durant la présentation des lettres. En fonction de la séquence de lettres le robot doit atteindre une certaine cible : pointer la bonne cible dans le cas de l'icub ou se diriger du bon côté du labyrinthe dans la tâche du panneau de signalisation.

Dans la suite nous utilisons les notations suivantes :

– T : durée de la simulation (correspond au nombre de pas de temps t, ici $t = 350$) ;
– T_{pres} : nombre de pas de temps utilisé lors de la présentation d'une lettre ;
– T_{delay} : nombre de pas de temps utilisé lors du délai entre la présentation de deux lettres, ici $T_{pres} = T_{delay} = 50$;
– T_{end} : nombre de pas de temps à la fin de la séquence de mémoire($T_{end} = 2 * T_{pres} + T_{delay}$) ;
– L : nombre de contextes utilisés durant l'évaluation ;

- $C = \{AX, AY, BX, BY\}$: ensemble des séquences de présentation de lettres ;
- x : un individu (réseau de neurones) ;
- $\mathbf{v}$: vecteur correspondant à une séquence de lettres ($\mathbf{v} \in \{0,1\}^l$) ;
- N_{in} : nombre d'entrée d'un réseau ;
- N_{out} : nombre de sortie d'un réseau ;
- N_{max} : nombre maximum de neurones pour un réseau ($N_{max} = 50$) ;
- $N(x)$: nombre de neurones pour un individu x avec $N(x) \in [N_{in} + N_{out}, N_{max}]$;
- $\gamma(x, \mathbf{v}, t)_i$: niveau d'activation d'un neurone i au temps t ($t \in [0, T]$) ;
- $\gamma_{end}(x, \mathbf{v})_i$: niveau d'activation d'un neurone i à la fin de la séquence de mémoire (i.e. $t = T_{end}(x, \mathbf{v})$).

Dans les deux cas, l'ensemble de la tâche dure 350 pas de temps et se déroule de la façon suivante :

- $0 < t < 50$: présentation de la première lettre (A/B) ;
- $50 \leq t < 100$: période de délai, tous les capteurs sont à 0 ;
- $100 \leq t < 150$: présentation de la seconde lettre (X/Y) ;
- $150 \leq t \leq 350$: le robot peut bouger et doit atteindre la bonne cible.

La séquence à mémoriser, la séquence de présentation des lettres, est donc séparée de la séquence de déplacement.

3.4.1.1 Evolution des réseaux de neurones

Structure fixe : évolution des paramètres (Elman). Dans ce cas nous utilisons un réseau de Elman (Elman, 1990), dont la structure est fixée et seuls les paramètres du réseau peuvent évoluer.

Evolution de la topologie et des paramètres du réseau (DNN). Dans ce cas à la fois la structure et les paramètres sont évolués. Nous utilisons le codage direct simple DNN décrit à la section 3.2.2.1.

3.4.1.2 Test post-optimisation

Un test d'évaluation la capacité de mémoire obtenue a été mis en place afin de mieux comparer les résultats.

Test de mémoire. Pour réussir la tâche, l'agent doit mémoriser la première lettre (A ou B) afin de tourner du bon côté après la présentation de la seconde lettre (X ou Y). L'objectif de ce test est de caractériser la capacité de mémoire obtenue au sein des individus. Pour cela nous vérifions après le processus d'optimisation deux choses : la capacité d'un agent à différencier les séquences AX des autres séquences (BX, BY, AY) ; la robustesse de cette différenciation lorsque le délai de présentation entre les lettres est plus grand.

Nous utilisons pour cela, un délai T_{delaym} plus grand

$$T_{presm} = T_{delaym} = 4 * T_{delay} \tag{3.2}$$

avec T_{delay} représentant le nombre de pas de temps utilisés lors du délai entre deux présentations de lettres, T_{delaym} le délai pendant le test de mémoire, T_{presm} le temps de présentation d'une lettre pendant le test de mémoire.

Nous vérifions que la sortie converge vers un vecteur constant :

$$\gamma(x, \mathbf{v}, t)_i - \gamma(x, \mathbf{v}, t-n)_i| < \varepsilon \forall n \in \{1, \cdots, 100\}, \forall i \in [0, N_{out}] \tag{3.3}$$

avec $\varepsilon = 10^{-6}$ et $t = T_{presm} + T_{delaym}$)

Enfin, nous testons si pour la séquence AX, il existe au moins un neurone avec une activité différente par rapport aux autres séquences (BX, BY, AY) :

$$\exists i \in [0, N] \begin{cases} \gamma(x, \mathbf{v}_{ax}, t)_i - \gamma(x, \mathbf{v}_{ay}, t)_i > \sigma \\ \gamma(x, \mathbf{v}_{ax}, t)_i - \gamma(x, \mathbf{v}_{by}, t)_i > \sigma \\ \gamma(x, \mathbf{v}_{ax}, t)_i - \gamma(x, \mathbf{v}_{bx}, t)_i > \sigma \end{cases} \tag{3.4}$$

avec $\varepsilon = 10^{-6}$, $\sigma = 0.1$, $t = T_{presm} + T_{delaym}$ et $\mathbf{v}_{ax}$ correspondant à la séquence de lettre AX.

3.4.2 Expérience 1 : AX-CPT avec une tâche de pointage

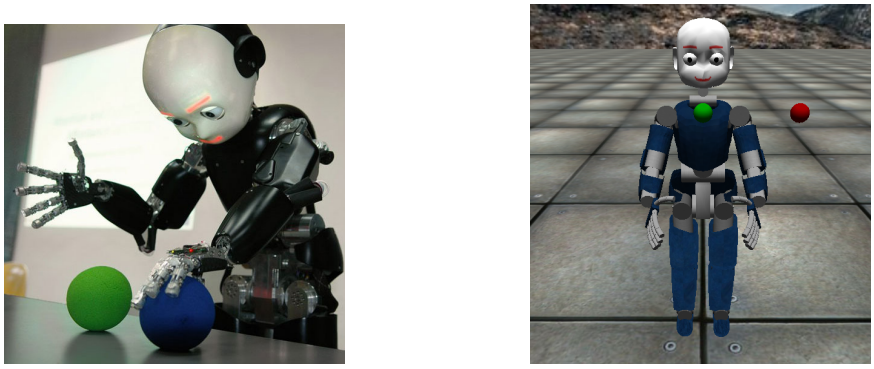
Description de la tâche. Suivant la séquence de lettre (AX,BX,AY,BY), le robot humanoïde simulé iCub (Metta et al., 2010) doit atteindre la cible correspondante. Il y a une cible spécifique pour la séquence AX, et une autre cible correspondant aux autres séquences.

Icub. L'iCub (Metta et al., 2010) (voir illustration 3.9) mesure approximativement 1m de hauteur et pèse 22 kg. Au niveau du haut du corps il possède un nombre de degrés de liberté (DDL) de 38, avec 7 pour chaque bras, 9 pour chaque main, et 6 pour la tête. Ici, nous utilisons 4 DDL du bras de l'iCub : 1) une articulation dans l'épaule qui correspond au mouvement avant-arrière quand le bras est aligné avec la gravité ; 2) une articulation dans l'épaule pour les mouvements d'adduction/abduction du bras ; 3) une articulation dans l'épaule pour les mouvements de lacet lorsque l'axe principal du bras est aligné avec la gravité.

En simulation, pour calculer la cinématique inverse nous utilisons la bibliothèque iKin (Pattacini et al., 2010) basée sur le package d'optimisation non-linéaire IPOPT [3].

Contrôleur neuronal. Le système de contrôle de l'iCub consiste en un réseau de neurones simple qui contrôle directement la direction qui est appliquée aux articulations motorisées. Les réseaux de neurones sont sélectionnés pour leur capacité à atteindre la position cible désirée. Il n'y a pas de contraintes imposée sur la manière par laquelle le problème doit être résolu : la trajectoire et la posture du bras sont laissées libres.

3. https://projects.coin-or.org/Ipopt



Réel *Simulé*

FIGURE 3.9 – Illustration du robot iCub.

Le réseau de neurones possède six neurones sensoriels, quatre entrées pour les séquences de lettres et quatre neurones de sortie. Les six neurones encodent la distance le long des trois axes entre la main du robot et les deux cibles normalisées dans l'intervalle $[-1, +1]$ (3 neurones par cible). Les quatres autres capteurs représentent chaque lettre A, B, X, Y qui reçoivent 1 si la lettres est présentée et 0 sinon. Les quatres moteurs de sortie encodent la vélocité angulaire des quatres articulations. L'activation des neurones de sortie est normalisée pour correspondre aux angles min, max ($[-95, 5]$, $[10, 120]$,$[-37, 65]$,$[20, 102]$) respectivement pour les articulations de 0 à 4.

Fitness. Le robot est évalué sur l'ensemble $C = \{AX, AY, BX, BY\}$ de séquences de lettres. Pendant l'évaluation, pour éviter le surapprentissage, un individu doit atteindre la cible correspondante dans $L = 16$ différents contextes. Un contexte est défini par une position spécifique u_{ax} pour la cible correspondant à AX et une position spécifique u_{oth} pour les autres cibles.

La fitness augmente seulement si l'agent est assez proche de la bonne cible.

$$f(x) = \frac{1}{6 * L * |C|} \sum_{i=0}^{L=16} \sum_{j \in C} \left(\begin{cases} 3 \text{ if } j = AX \text{ and} \\ \quad d(p_{ij}(x,T), u_{ax}) < \sigma \\ 1 \text{ if } j \neq AX \text{ and} \\ \quad d(p_{ij}(x,T), u_{oth}) < \sigma \\ 0 \text{ otherwise} \end{cases} \right) \tag{3.5}$$

avec σ arbitrairement fixé à 0.05, $p_{ij}(x, T)$ la position de la main x à la fin de la séquence et $d(x, y)$ est une distance euclidienne. La constante 6 correspond à la note maximale quand l'agent pointe correctement la bonne cible pour chaque séquence de lettres pour un contexte. Elle est utilisée pour normaliser la valeur de la fitness dans l'intervalle $[0, 1]$.

3.4.2.1 Résultats

Pour les meilleurs individus de chaque configuration, nous allons étudier :

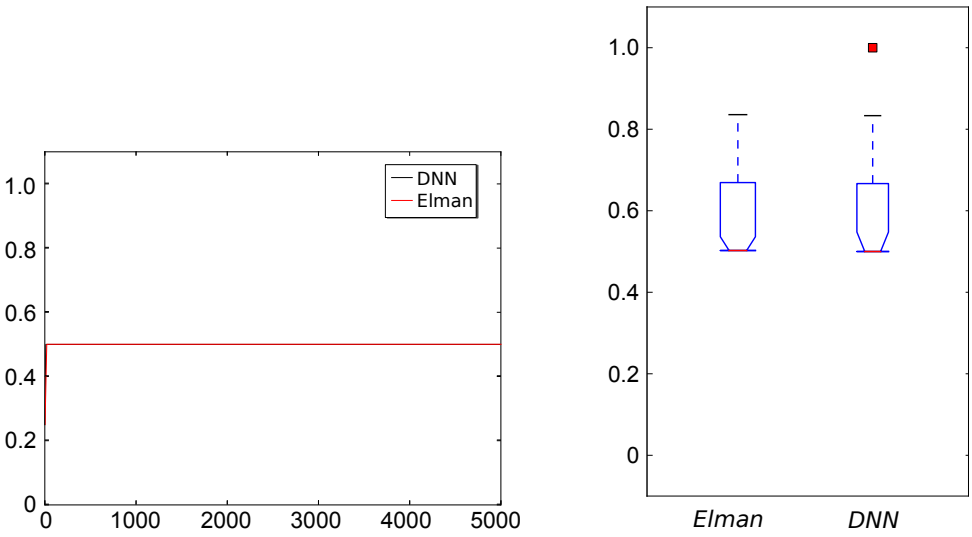


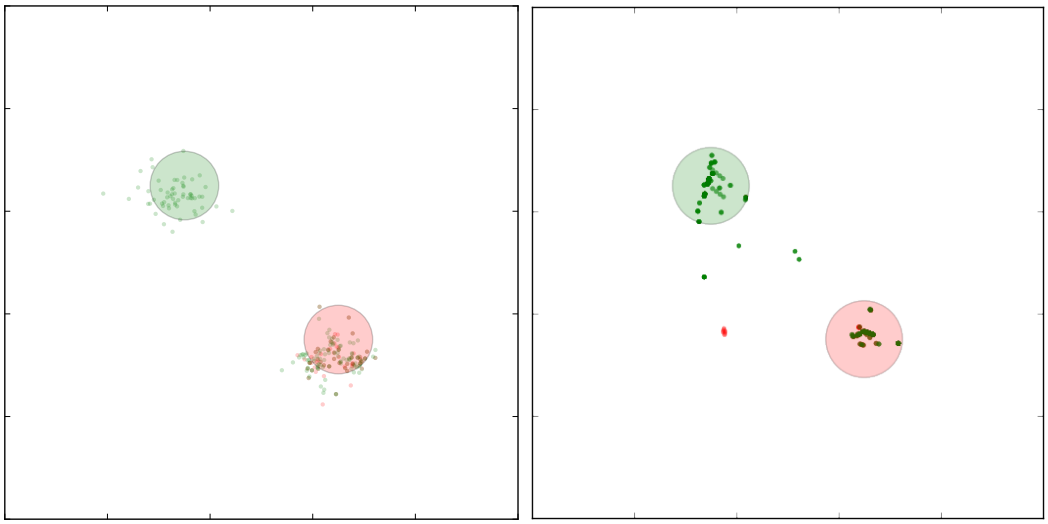
FIGURE 3.10 – (a) évolution des individus obtenus par les différentes configurations. Les courbes représentent les médianes de chaque configuration, avec 30 runs par configuration. (b) boîte à moustache représentant les meilleurs individus de chaque run pour les différentes configurations. (30 runs par configuration).

- leurs performances ;
- leurs trajectoires et leurs positions finales ;
- leur capacité de mémoire ;

Performances. A gauche de la figure 3.10 est illustrée l'évolution de la fitness pendant le processus d'optimisation. Chaque courbe correspond à la médiane des runs pour chaque configuration. Nous pouvons constater que le minimum local à 0.5 est rapidement atteint après seulement quelques générations, que ce soit avec *DNN* ou *Elman*. La fitness reste stable pendant tout le processus d'optimisation que ce soit avec *DNN* ou *Elman*. Imposer une topologie favorisant l'émergence de mémoire comme l'utilisation d'un réseau récurrent de type Elman ne favorise pas, dans ce cas les performances.

A droite de la figure 3.10 sont affichées les boîtes à moustache correspondant aux meilleurs individus de chaque configuration (*DNN* et *Elman*). Les résultats sont très similaires sur les deux configurations. Seul un individu avec la configuration *DNN* a obtenu la fitness maximale. Comme attendu, faire émerger une forme simple de mémoire au sein d'un agent n'est pas un problème trivial.

Positions finales. La figure 3.11 représente les positions de la main à la fin de la simulation pour les meilleurs individus de chaque configuration (30 runs par configuration). Les points rouges correspondent à la position de la main lors de la séquence AX et les points verts pour les autres séquences de lettres (AY,BX,BY).



Elman *DNN*

FIGURE 3.11 – positions de la main à la fin de la simulation pour les meilleurs individus de chaque configuration dans l'expérience 1 (30 runs par configuration). Les points rouges correspondent à la position de la main lors de la séquence AX et les points verts pour les autres séquences de lettres (AY,BX,BY).

En comparant les positions finales obtenues par les meilleurs individus, on peut observer des comportements similaires. Le minimum local dans lequel tombe par la plupart des individus correspond en fait à un individu qui se dirige toujours du même côté. En effet, ces individus ne sont pas capables de différencier les séquences AX des autres séquences et choisissent indifféremment la cible de gauche ou la cible de droite. Les positions finales des solutions obtenues par la configuration *DNN* sont plus précises que les solutions obtenues par *Elman*

Mémoire. Le graphique 3.12 montre différentes performances des meilleurs individus de chaque configuration obtenues sur l'expérience de pointage. La barre rouge représente le pourcentage d'individu ayant obtenu la fitness maximum. La barre bleue le pourcentage d'individu ayant obtenu la fitness maximum et réussi le test de mémoire, La barre noire indique les individus ayant réussi le test de mémoire.

On constate que les résultats obtenus quelles que soit les configurations sont très médiocres. L'utilisation d'un réseau de type Elman, propice a priori l'émergence de mémoire, ne permet pas d'obtenir des résultats significativement meilleurs que DNN sur le test de mémoire post-optimisation.

3.4.3 Expérience 2 : AX-CPT dans labyrinthe en T

C'est une extension de la tâche "panneau de signalisation" (Ziemke et Thieme, 2002; Rylatt et Czarnecki, 2000) visible à la figure 3.13. Dans notre cas, le stimulus permettant de déterminer de quel côté l'agent doit tourner dans le labyrinthe en T

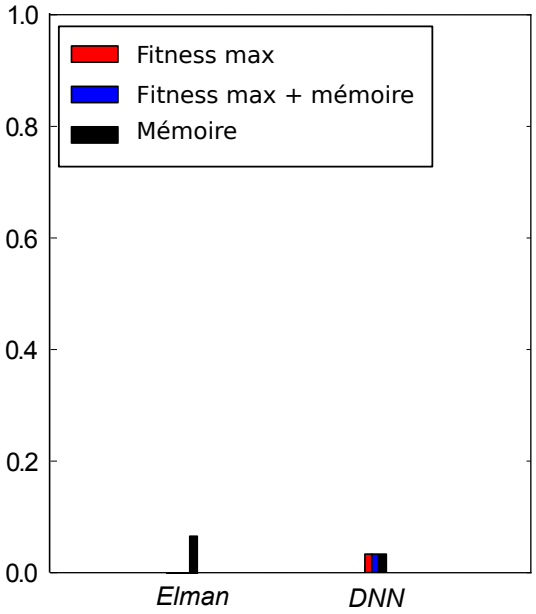


FIGURE 3.12 – Graphique représentant différentes performances des meilleurs individus de chaque configuration. La barre rouge représente le pourcentage d'individu ayant obtenu la fitness maximum, La barre bleue le pourcentage d'individu ayant obtenu la fitness maximum et réussi le test de mémoire La barre noire indique les individus ayant réussi le test de mémoire sans prendre en considération la fitness obtenue.

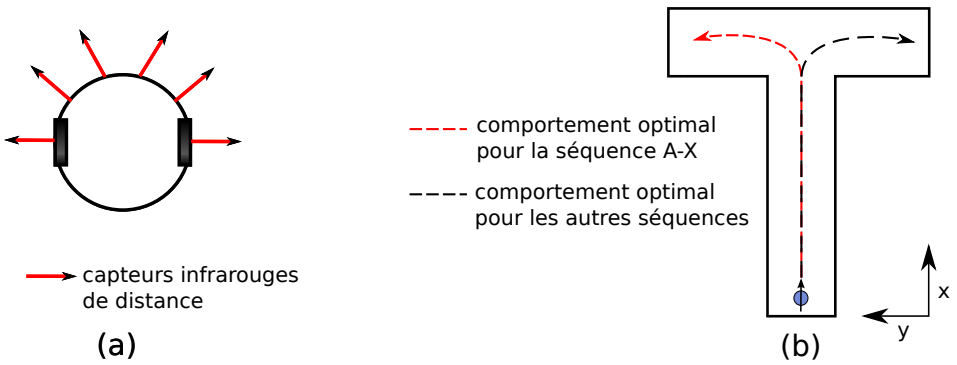


FIGURE 3.13 – (a) Robot mobile utilisé pour la tâche de navigation dans un labyrinthe en T simulé. En plus des capteurs indiqués sur le schéma, le robot dispose d'un capteur pour chacun des 4 stimuli A, B, X, Y. (b) Carte du labyrinthe en T et indication du comportement optimal recherché par séquence de stimuli.

est remplacé par une combinaison de quatre stimuli A, B, X, Y. Il y a un capteur pour chaque lettre qui reçoit 1 si la lettre est présenté et 0 sinon. L'agent doit tourner à gauche pour la séquence de lettres AX et à droite pour les autres séquences (BX,BY,AY).

Contrôleur neuronal. L'agent est un robot simulé à deux roues de type e-puck recevant comme entrées sensorielles six capteurs de distance infrarouges et quatre capteurs pour les lettres. Le robot contrôle sa vitesse par l'intermédiaire de 2 neurones de sortie correspondant à la vitesse de ses moteurs gauche et droit.

Fitness. Le robot est évalué sur l'ensemble $C = \{AX, AY, BX, BY\}$ de séquences de lettres. Durant l'évaluation afin d'éviter le surentraînement, un individu doit atteindre le bon côté du labyrinthe dans 12 contextes différents. Un contexte, ici, est défini par une position et un angle initial de départ pour le robot.

$$f(x) = \frac{1}{6 * L * |C|} \sum_{i=0}^{L=12} \sum_{j \in C} \left(\begin{cases} 3 \text{ if } (p_{ij}(T) \in G \text{ and } j = AX) \\ 1 \text{ if } (p_{ij}(T) \in D \text{ and } j \neq AX) \\ 0 \text{ otherwise} \end{cases} \right) \tag{3.6}$$

G et D correspondent respectivement à l'ensemble des positions à gauche et à droite du corridor du labyrinthe en T, avec $G \cap D = \varnothing$. $p_{ij}(T)$ correspond à la position finale de l'agent à la fin ($t = T$) de la séquence j du contexte i. La constante 6 correspond à la note maximale quand l'agent se dirige correctement du bon côté du labyrinthe à chaque séquence de lettres pour un contexte. Elle est utilisée pour normaliser la valeur de la fitness dans l'intervalle $[0, 1]$. La fonction de fitness récompense seulement les individus qui se dirigent vers la bonne cible suivant la séquence de lettres.

3.4.3.1 Résultats

Pour les meilleurs individus de chaque configuration, nous allons étudier :
- leurs performances ;
- leurs trajectoires ;
- leur capacité de mémoire.

3.4.3.2 Performance

A gauche de la figure 3.10 est illustré l'évolution de la fitness pendant le processus d'optimisation. Chaque courbe correspond à la médiane des runs pour chaque configuration.

Dans expérience le minimum local à 0.5 est obtenue au bout de quelques générations pour la configuration *DNN* et après plusiseurs centaines de générations pour pour la configuration *Elman*.

Dans cette expérience, l'utilisation d'un réseau récurrent de type Elman pénalise les performances. Il faut environ 2000 générations avec *DNN*, pour que la

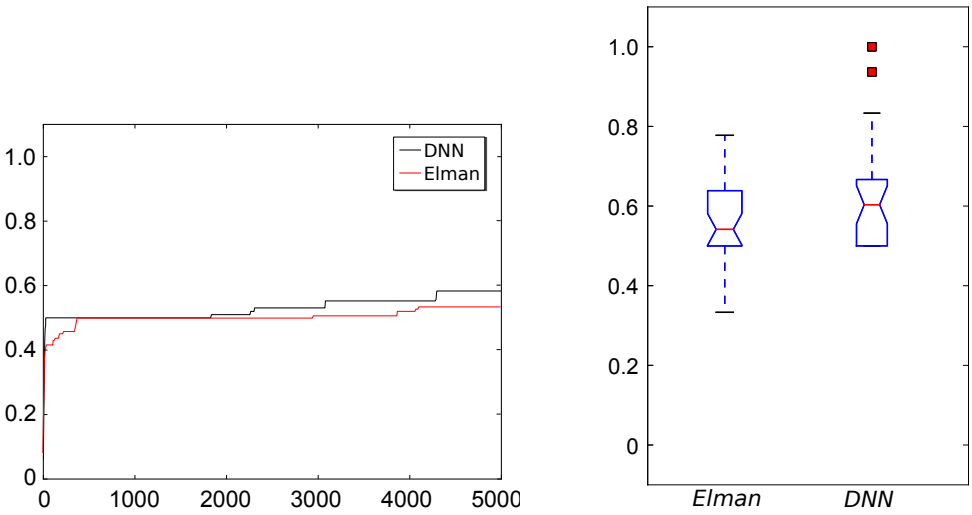


FIGURE 3.14 – (a) Evolution des individus obtenus par les différentes configurations. Les courbes représentent les médianes de chaque configuration, avec 30 runs par configuration. (b) Boîte à moustache représentant les meilleurs individus de chaque run pour les différentes configurations. (30 runs par configuration).

médiane sorte du minimum local, pour atteindre environ 0.6 à la fin du processus d'optimisation. Alors pour *Elman* ce n'est seulement qu'après 4000 générations.

A droite de la figure 3.10 sont affichées les boîtes à moustache correspondant aux meilleurs individus de chaque configuration (*DNN* et *Elman*). Les résultats sont très similaires sur les deux configurations et très proches des résultats obtenus sur l'expérience précédente. Nous constatons simplement une performance supérieure mais non significative de la configuration *DNN* par rapport à *Elman*.

3.4.3.3 Trajectoire

La figure 3.15 représente les trajectoires obtenues dans l'expérience 2 par les meilleurs individus pour les différentes configurations (30 runs par configuration). Les points rouges correspondent aux trajectoires pour les séquences. AX. Les points verts pour les autres séquences (AY,BX,BY).

Tout comme l'expérience précédente, nous pouvons constater que les trajectoires pour les séquences AX et les autres séquences sont quasi identiques que ce soit avec *DNN* et *Elman*. En comparant les trajectoires obtenues par les meilleurs individus sur les deux expériences, on peut observer des comportements similaires.

3.4.3.4 Mémoire

Le graphique 3.16 montre différentes performances des meilleurs individus de chaque configuration obtenues sur les deux expériences. La barre rouge représente le pourcentage d'individu ayant obtenu la fitness maximum, La barre bleue le pour-

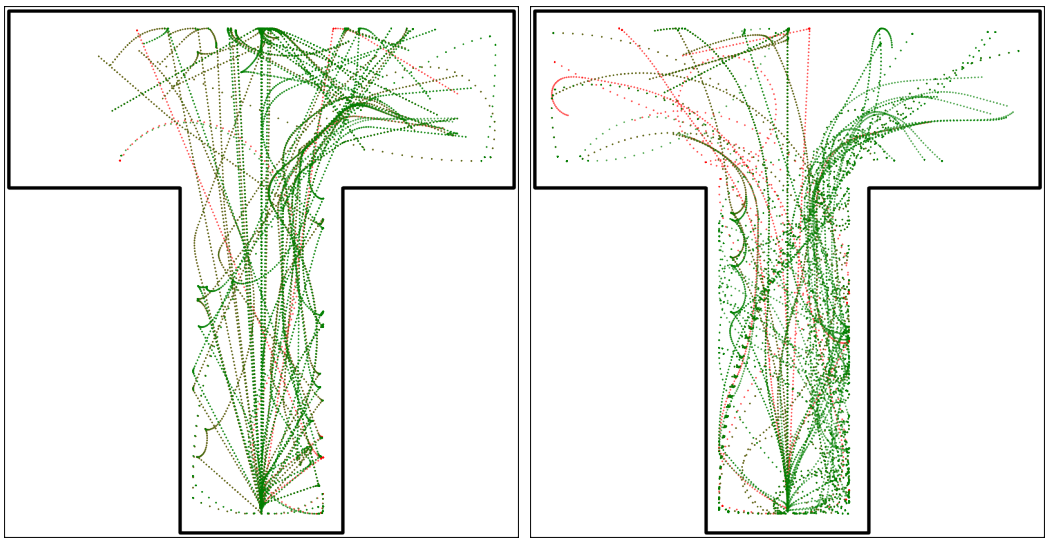


FIGURE 3.15 – Trajectoires obtenues par les meilleurs individus dans l'expérience "panneau de signalisation" pour les différentes configurations (30 runs par configuration). Les points rouges correspondent aux trajectoires pour les séquences. AX. Les points verts pour les autres séquences (AY,BX,BY).

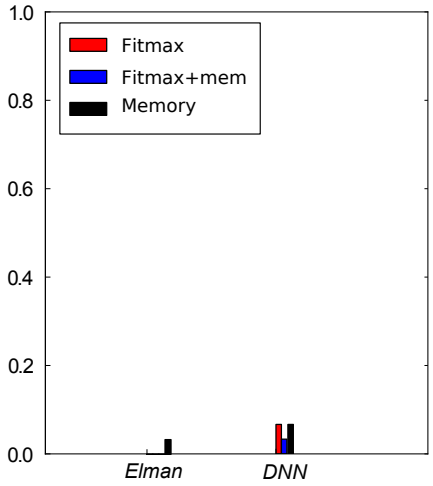


FIGURE 3.16 – Graphique représentant différentes performances des meilleurs individus de chaque configuration sur l'expérience panneau de signalisation. La barre rouge représente le pourcentage d'individus ayant obtenu la fitness maximum, La barre bleue le pourcentage d'individus ayant obtenu la fitness maximum et réussi le test de mémoire La barre noire indique les individus ayant réussi le test de mémoire sans prendre en considération la fitness obtenue.

centage d'individu ayant obtenu la fitness maximum et réussi le test de mémoire, La barre noire indique les individus ayant réussi le test de mémoire.

Tout comme l'expérience précédente, on constate que les résultats obtenus sur les deux configurations sont très médiocres. Seul avec la configuration *DNN* nous avons obtenus deux solutions obtenant la fitness maximale.

3.5 Conclusion

Nous avons validé notre hypothèse de départ sur le caractère trompeur de l'émergence d'une capacité de mémoire au sein d'un agent. Les différentes expériences liminaires ont montré l'importance du protocole expérimental pour faire émerger une forme de mémoire. Parmi les individus obtenus, peu d'individus ont développé une capacité de mémoire. En effet, l'utilisation d'un protocole expérimental censé nécessiter l'intégration de données sensorimotrices telle que la tâche de "panneau de signalisation" peut être résolue par des robots réactifs qui exploitent l'environnement en externalisant leur mémoire.

A partir d'un bref état de l'art sur les tâches de mémoire utilisées en neurosciences et des modèles computationnels associés, nous avons pu mettre en place un protocole basé sur une tâche AX-CPT impliquant la mémoire de travail. Cependant, que ce soit en utilisant un réseau récurrent (Elman) censé favoriser l'apparition de mémoire ou en faisant évoluer la topologie des réseaux (*DNN*), nous avons obtenu des taux de convergence très faibles sur les deux expériences basées sur AX-CPT. Le fait que quelques solutions aient tout de même été trouvées nous confirme que le codage utilisé (*DNN*) n'est pas en cause. L'objectif est désormais d'améliorer le taux de convergence. Pour parvenir à de meilleurs résultats nous émettons l'hypothèse que nous faisons face à un problème de pression de sélection. Pour vérifier cela, nous testerons dans le chapitre suivant plusieurs objectifs, guidés ou non vers le but pour favoriser l'obtention d'individus dotés de mémoire ainsi que différents types de diversités comportementales afin de mieux explorer l'espace de recherche.

Chapitre 4

Définition de la fitness et objectifs d'exploration

Pour explorer le champ des possibles, le bricolage est la méthode la plus efficace.

Hubert Reeves

Objectifs Améliorer le taux de convergence sur les expériences de mémoire mises en place au chapitre précédent.
Démarche Tester l'influence de :
- l'insertion d'a priori dans la fitness
- l'utilisation d'objectifs d'exploration :
  - diversité comportementale ;
  - nouveauté.

Résultats
- L'utilisation d'objectifs d'exploration (diversité, nouveauté) a un impact significatif sur l'émergence d'une forme de mémoire interne au sein des agents ;
- Contre-intuitivement, insérer de l'a priori au sein de la fitness favorise des agents convergeant vers un minimum local.

Sommaire

4.1 Introduction

Nous avons pu observer au chapitre précédent que la conception automatique d'agents dotés d'une capacité de mémoire était un problème trompeur. En effet, les agents obtenus dans nos expériences nécessitant une forme de mémoire tombent dans des minima locaux et parviennent difficilement à générer des comportements autres que réactifs.

Différentes hypothèses peuvent expliquer les résultats :

- Le codage utilisé est inadapté : l'utilisation d'un codage direct comme *DNN*[1] ne permettrait pas de faire émerger des individus capable de se mouvoir et capable d'intégrer des informations sensorimotrices dans le temps.
- La pression de sélection utilisée est inadaptée : dans ce cas précis, l'utilisation d'une fitness simple et peu directive, n'a pu mener à l'émergence de comportements intéressants. Il est peut être nécessaire d'intégrer au sein de la fitness différents gradients de sélection pour échapper aux différents minima locaux.
- la capacité d'exploration de l'algorithme évolutionniste est insuffisante : ce qui entraine une convergence prématurée.

Nous nous intéresserons dans ce chapitre aux deux derniers points.

4.1.1 Fonction de fitness en robotique évolutionniste

Afin d'exercer une pression de sélection efficace, différentes méthodes ont été proposées, comme le recours à une approche incrémentale. Le but étant de permettre la résolution de problèmes qui se sont avérés insolubles par ailleurs. Le principe général de telles approches est de décomposer le problème et de le résoudre "sous-problème" par "sous-problème" (Harvey et al., 1994; Kodjabachian et Meyer, 1997; Winkeler et Manjunath, 1998; Parker, 2001) ou en augmentant progressivement la complexité (Gomez et Miikkulainen, 1997), se ramenant ainsi à des difficultés abordables. Une autre méthode appelée façonnage de fitness ou "fitness shaping" (Nolfi et Parisi, 1995) repose sur l'idée d'ajouter des termes visant à lisser le paysage de fitness en récompensant des comportements intermédiaires ou en punissant des comportements non désirés. Ces méthodes portent sur la définition de la pression de sélection et visent à l'adapter en incluant des connaissances a priori sur le problème. Selon Nelson et al. (2009) les fonctions de fitness en ER peuvent êtres classées en fonction de cette quantité de connaissances a priori que le concepteur intègre dans le processus d'évolution[2], le tableau 4.1 résume les différentes catégories.

Fitness obtenue sur des données d'apprentissage. En supposant qu'on dispose d'un certain nombre d'entrées pour lesquelles la réponse optimale du contrôleur

1. pour plus de détails voir section 3.2.2.1.
2. pour une revue plus complète, se référer à Nelson et al. (2009)

TABLE 4.1 – Différents types de fitness et leur niveau de connaissance a priori. Inspiré par (Nelson et al., 2009)

Fitness	Incorporation de connaissance a priori
Fitness obtenue sur des données d'apprentissage	Très haut
Fitness comportementale	Haut
Fitness par increment fonctionnel	Modéré-Haut
Fitness adaptée	Modéré
Fitness par incréments environnementaux	Modéré
Fitness aggrégée	Très bas
Absence de fitness	Très bas

est connue, l'erreur entre la réponse effective du contrôleur et la réponse optimale attendue peut être utilisée comme fitness. Par exemple dans l'article (Mucientes et al., 2010), des comportements simples de navigation sont optimisés pour un robot à roues en minimisant l'erreur sur les valeurs de moteurs en fonction des entrées sensorielles du robot sur plusieurs centaines de tests.

Fitness comportementale. Une fitness comportementale indique pour un contrôleur donné si son comportement sur le robot vérifie un certain nombre de caractéristiques que doit posséder le comportement optimal. On se concentre donc ici uniquement sur le comportement du robot obtenu. Par exemple, des comportements de marche ont été obtenus par optimisation sur un robot octopode en définissant une fitness qui augmente ou diminue à chaque étape de l'évaluation en fonction de l'adéquation entre le comportement observé et des règles définissant le comportement optimal (Jakobi, 1998).

Fitness par incréments fonctionnels. Ce type de fitness est typiquement utilisé pour faciliter l'apprentissage d'un comportement complexe en le découpant en différents sous-comportements. Le processus compte alors plusieurs étapes d'optimisations successives où les contrôleurs sont optimisés pour apprendre un ensemble de ces sous-comportements, par exemple via une fitness comportementale. Cette technique a notamment été utilisée avec un robot portique pour l'apprentissage de suivi de cibles visuelles (Harvey et al., 1994) : le robot apprend d'abord à localiser des cibles fixes de grandes tailles, puis des cibles fixes de petites tailles et enfin des cibles mobiles à l'aide de 3 fonctions de fitness différentes. De nombreuses autres approches incrémentales ont été proposées (Gomez et Miikkulainen, 1996; Kodjabachian et Meyer, 1997; Urzelai et al., 1998; De Nardi et al., 2006). Dans cette catégorie, nous pouvons citer le "fitness shaping", dans ce cas la fitness est définie comme une agrégation (une somme pondérée, par exemple) de différents critères d'évaluation afin de créer un gradient de fitness. Utilisant cette idée, Nolfi et Parisi (1995) ont réussi à obtenir un perceptron multicouche capable de piloter un robot pour localiser, reconnaître et attraper un objet. La fitness était augmentée si l'individu était près de l'objet, si l'objet était en face du robot, si le robot essayait d'attraper l'objet, si le robot avait l'objet dans la pince et si le robot relâchait l'objet

au bon endroit.

Fitness par incréments environnementaux. Cette fonction de fitness est basée sur les mêmes principes qu'une fitness par incréments fonctionnels, à ceci près que la fonction de fitness n'est pas modifiée, alors que l'environnement d'évaluation se complexifie à chaque étape de l'optimisation : le robot doit alors résoudre la tâche considérée dans un environnement simple, puis un environnement un peu plus complexe Miglino et al. (1998); Nakamura et al. (2000); Barlow et Oh (2006). Par exemple Bongard (2011a), utilise une telle fonction de fitness dans un processus dit de *shaping* où un robot simulé apprend à déplacer des objets de différentes formes et à les saisir.

Fitness dite agrégée. Une fitness agrégée contient un ou plusieurs termes ayant trait à la réalisation de la tâche. Par exemple si un robot doit ramasser un ensemble de balles dans une arène et les amener dans un panier, une fitness agrégée possible est le nombre de balles dans le panier à la fin de l'évaluation (Mouret et Doncieux, 2012). Ce type de fitness contient peu d'informations a priori sur la tâche à résoudre, étant donné qu'elle ne juge que du degré de réalisation de la tâche et non pas de la manière dont la tâche est réalisée.

Fitness adaptée ("tailored"). Ce type de fonction représente le plus large groupe de fonction de fitness utilisée en ER. Elle combine des éléments issus des fitness comportementales et des fitness agrégées. Un exemple avec une tâche de phototaxie, la fonction de fitness peut contenir un terme qui récompense le contrôleur qui se rapproche de la source de lumière par n'importe quel moyen, sans regarder le comportement des capteurs et des effecteurs. Cela peut être vu comme une fitness ne contenant pas d'a priori, mais elle contient l'information qu'être plus proche du but est un comportement meilleur.

Absence de fitness. Dans certains travaux, aucune fonction de fitness n'est définie explicitement et on recherche les comportements les plus "nouveaux" sur la tâche considérée (Lehman, 2008; Lehman et Stanley, 2010). L'idée principale est de ne pas guider l'optimisation par un objectif de performance défini a priori par le concepteur, mais d'encourager l'exploration de l'espace comportemental afin de trouver des solutions de plus en plus originales, dont certaines se révèlent également performantes. Cette recherche par nouveauté est décrite plus en détails dans la section suivante.

4.2 Objectif d'exploration

Afin de limiter une convergence prématurée de la population vers un minimum local, un algorithme évolutionniste doit posséder une bonne capacité d'exploration

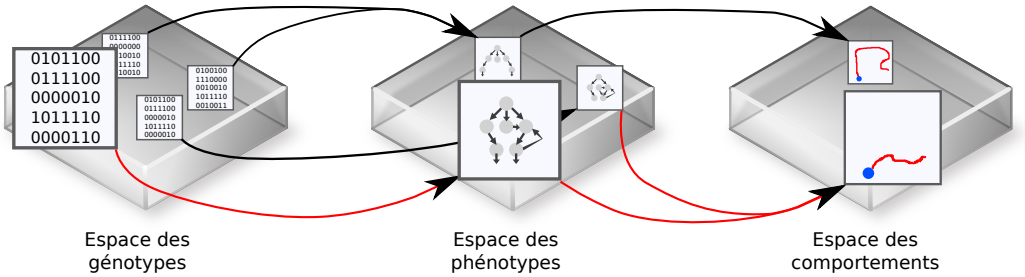


FIGURE 4.1 – Au sein d'un algorithme évolutionniste, les opérateurs génétiques manipulent des représentations de solutions candidates (à gauche) qui sont développées en phénotypes (au milieu), dont la fitness est évaluée. Plusieurs génotypes peuvent correspondre au même phénotype. La relation génotype/phénotype est non-injective. En ER ; les phénotypes (i.e la morphologie du robot ou un réseau de neurones) sont simulés durant un certain nombre de pas de temps et le comportement du robot est observé pour évaluer la fitness (à droite). Plusieurs phénotypes peuvent conduire un robot de la même manière. L'espace phénotype/comportement est aussi non-injectif.

dans l'espace de recherche. Pour cela il est nécessaire de maintenir au sein d'une population une certaine diversité. Cette diversité peut se situer à différents niveaux :

- au niveau de l'espace des génotypes ;
- au niveau de l'espace des fitness ;
- au niveau de l'espace des comportements.

Dans le cas de la robotique évolutionniste, Le comportement est déterminé par le phénotype et résulte de l'interaction du robot avec son environnement (voir figure 4.1 pour une illustration des différents espaces).

Les premiers travaux sur la diversité génotypique viennent de l'observation que les algorithmes évolutionnistes produisent souvent des solutions similaires qui occupent un minimum local. Une première variante, le *fitness sharing* (Goldberg et Richardson, 1987) réduit la *fitness* d'un individu proportionnellement au nombre d'individus de la population qui sont similaires (Rosin et Belew, 1997). Le codage *NEAT* (Stanley et Miikkulainen, 2002) par exemple, protège la diversité entre génotypes en conservant préférentiellement les individus ayant une topologie différente de celles du reste de la population en utilisant ce type de mécanisme. L'idée est de forcer la complexification progressive des comportements des réseaux tout en explorant des topologies variées.

La diversité peut être vue comme un objectif à part entière. Dans ce cas, elle est optimisée via une approche multi-objectifs avec comme premier objectif la fitness initiale et comme second objectif une mesure de diversité (Abbass et Deb, 2003; De Jong et al., 2001). On définit pour cela une métrique $d_g(G_1, G_2)$ qui fournit une distance entre deux génotypes G_1 et G_2 basée sur leur composition. Plus deux génotypes sont différents, plus la distance doit être grande, L'objectif de diversité

devenant ainsi la maximisation de la distance avec les plus n plus proches voisins.

Parfois, les opérateurs génétiques n'agissent pas directement sur les solutions candidates mais sur une représentation sous-jacente. Le codage définit alors une relation génotype/phénotype (figure 4.1 à gauche). Dans ce cas deux solutions proches dans l'espace des génotypes peuvent être très éloignées dans l'espaces des phénotypes, et vice versa. Dans une telle situation, il est souvent plus informatif de calculer la distance dans l'espace des phénotypes. Cependant nous pouvons distinguer trois limitations à l'utilisation d'une distance dans l'espace des phénotypes :

- Le coût : le calcul d'une mesure de similarité générique (la distance d'edite) sur des graphes est NP-difficile (Zhang et al., 1992; Bunke et Shearer, 1998) alors que les techniques de diversité nécessitent de calculer de nombreuses distances à chaque génération ;
- La relation phénotype/comportement est non-linéaire : deux réseaux de neurones proches peuvent mener à des comportements qualitativement différents et ceci à cause de l'interaction avec l'environnement et de l'accumulation des différences pendant l'évaluation de la fitness ;
- la relation phénotype/comportement est non-injective : deux réseaux de neurones structurellement différents peuvent faire que le robot se comporte qualitativement de manière similaire, par exemple le robot peut tourner sur lui-même durant toute l'expérience.

Utiliser un objectif de diversité au niveau de l'espace des fitness (Bui et al., 2005) permet de s'abstraire de ce biais. Cependant selon le degré d'a priori intégré au sein de la fitness, celle-ci ne contient que certains aspects du comportement de l'individu. De ce fait des fitness similaires peuvent correspondre à des comportements très différents, et réciproquement, des fitness différentes peuvent correspondre à des comportements similaires. L'utilisation d'une diversité au niveau de l'espace des fitness n'est donc pas la garantie d'une exploration efficace de l'espace des comportements.

Partant de ce constat, Lehman (2008); Lehman et Stanley (2010) ont proposé une méthode radicalement différente, la recherche de nouveauté. Dans ce cas, la recherche d'individus au comportement différent devient l'unique objectif sans utiliser de fitness classique pour orienter l'exploration dans l'espace de recherche. Cette recherche de nouveauté maximise ainsi uniquement la nouveauté d'un comportement en le comparant à une archive des comportements déjà rencontrés durant le processus d'optimisation. Elle réalise ainsi une recherche dans l'espace des comportements en s'appuyant sur NEAT (Stanley et Miikkulainen, 2002) afin de garantir une augmentation progressive de la complexité des comportements explorés. Cette méthode a montré son efficacité sur différentes tâches :

- Un robot à roue qui doit atteindre un but dans un labyrinthe trompeur (Lehman, 2008; Lehman et Stanley, 2010; Mouret, 2011)
- un robot bipède devant se déplacer le plus rapidement possible (Lehman et Stanley, 2010)
- un robot à roue ramassant des balles (Mouret, 2011)
- un robot devant utiliser une plasticité synaptique pour trouver une récom-

pense au sein d'un T-maze (Risi et al., 2009)

Cependant, nous pouvons distinguer deux limitations quant à l'utilisation de la nouveauté :

- ne pas utiliser de biais dans la recherche d'individus optimaux peut s'avérer problématique lorsque l'espace de recherche est vaste. Pour pallier à cette limitation, l'objectif de nouveauté a depuis été utilisé avec succès en complément d'une fitness classique (Mouret et Doncieux, 2009b; Mouret, 2011) via une approche multiobjectifs.
- avec l'emploi d'une archive, la recherche de nouveauté est un processus plus coûteux qu'un simple objectif de préservation de diversité. En effet, lorsque l'archive des comportements augmente, il devient de plus en plus coûteux de trouver les plus proches voisins. En raison de cette contrainte, un objectif de diversité comportementale non basé sur une archive a été proposé (Mouret et Doncieux, 2009b,a; Doncieux et Mouret, 2010a).

Le principe de diversité comportementale repose sur une mesure de comportement des individus. Cela nécessite d'avoir à disposition un descripteur du comportement des individus permettant de caractériser le degré de similarité de manière efficace. Pour cela deux alternatives sont envisageables :

- emploi de descripteurs adhoc : le concepteur choisit des descripteurs liés au problème à résoudre.
- emploi de descripteurs génériques : les descripteurs ne dépendent pas d'une expérience particulière, mais sont définis de façon systématique sans nécessiter de connaissance supplémentaire. Parmi ceux-ci nous pouvons retenir les descripteurs basés sur la trajectoire, la similarité des individus se mesurant via une distance euclidienne entre des vecteurs qui décrivent le nombre de pas de temps passés dans chaque partie de l'arène. Il est possible aussi d'utiliser une distance de Hamming entre les données binarisées des vecteurs du flux sensorimoteur (Doncieux et Mouret, 2010a).

En pratique, l'utilisation d'une diversité comportementale donne de bons résultats en permettant aux solutions de s'échapper des optima locaux très attractifs de l'espace de fitness (Mouret et Doncieux, 2009b) ou dans le cas de problèmes de *bootstrap* (Mouret et Doncieux, 2008, 2009a), c'est-à-dire quand la fonction de fitness ne fournit pas de gradient. Par ailleurs, certains travaux indiquent qu'optimiser un objectif de diversité est plus efficace qu'une technique de *fitness sharing* sur différents problèmes robotiques (Mouret et Doncieux, 2012).

4.2.1 Synthèse

Nous avons décrit précédemment les grandes catégories de fitness utilisées en ER ainsi que les bénéfices que peuvent apporter l'utilisation d'un objectif favorisant le maintien d'une diversité comportementale au sein des individus. L'objectif de ce chapitre est d'étudier l'influence de ces deux approches sur l'émergence d'une capacité de mémoire. Nous testerons ainsi :

- l'insertion d'a priori dans la fitness : l'objectif dans ce cas est d'intégrer au

sein de la fitness un minimum de connaissance quant à la manière de résoudre la tâche, afin d'obtenir un gradient ;
– l'ajout d'un objectif de diversité comportementale : l'ajout de cet objectif a pour but de favoriser les individus au comportement différent afin de mieux explorer l'espace de recherche.

4.3 Expériences et résultats

Afin de tester l'impact d'insertion d'a priori dans la fitness et l'impact de l'utilisation d'une diversité comportementale en deuxième objectif, nous reprenons les protocoles évoqués dans le chapitre précédent (section 3.4), c'est à dire deux tâches robotiques différentes intégrant le protocole AX-CPT (décrit à la section 3.3) :
– une tâche de pointage utilisant un robot humanoïde, le robot Icub ;
– la tâche de "panneau de signalisation" avec un robot de type Epuck.

Pour les deux tâches, nous utilisons donc le protocole suivant : une séquence de lettres est présentée à l'agent, A ou B, suivi par X ou Y après un délai. Les moteurs sont désactivés durant la présentation des lettres. En fonction de la séquence de lettre le robot doit atteindre une certaine cible : pointer la bonne cible dans le cas de l'iCub ou se diriger du bon côté du labyrinthe dans la tâche du panneau de signalisation.

4.3.1 Protocole expérimental

Pour les deux expériences, le contrôle du robot est assuré par un réseau de neurones. Deux types de contrôleurs seront optimisés :
– Optimisation des paramètres du réseau (*Elman*) : dans ce cas, la topologie du réseau de neurones récurrent de type Elman (Elman, 1990) reste fixe et seuls les paramètres du réseau sont optimisés.
– Optimisation de la topologie et des paramètres du réseau *DNN* : la topologie et les paramètres sont optimisés, et les connexions récurrentes sont autorisées.

Les neurones à optimiser sont des intégrateurs à fuite ayant des propriétés de contractions, des lPDS (Girard et al., 2008) (voir par. 2.2.1.2 pour de plus de détails).

Les différentes configurations utilisent l'algorithme évolutionniste multi-objectif élitiste NSGA-II (Deb, 2001), qui repose sur une sélection par tournoi et une méthode de classement des individus basée sur la relation de dominance (cf. partie 2.1.4.3 pour plus de détails). Ce travail a été implémenté dans le framework Sferes$_{v2}$ (Mouret et Doncieux, 2010).

4.3.2 Fitness

Dans la suite nous utilisons les notations suivantes :
– x : un individu (réseau de neurones)
– T : durée de la simulation (nombre de pas de temps)

- l : nombre de lettres ($l = 4$ pour A, B, X, Y)
- $\mathbf{v}$: vecteur correspondant à une séquence de lettres ($\mathbf{v} \in \{0,1\}^l$) ;
- N_{in} : nombre d'entrées d'un réseau ;
- N_{out} : nombre de sorties d'un réseau ;
- N_{max} : nombre maximum de neurones pour un réseau ($N_{max} = 50$) ;
- $N(x)$: nombre de neurones pour un individu x avec $N(x) \in [N_{in} + N_{out}, N_{max}]$;
- T_{pres} : nombre de pas de temps utilisés lors de la présentation d'une lettre ;
- T_{delay} : nombre de pas de temps utilisés lors du délai entre la présentation de deux lettres, ici $T_{pres} = T_{delay}$;
- T_{end} : nombre de pas de temps à la fin de la séquence de mémoire($T_{end} = 2 * T_{pres} + T_{delay}$) ;
- $\gamma(x, \mathbf{v}, t)_i$: niveau d'activation d'un neurone i au temps t ($t \in [0, T]$) ;
- $\gamma_{end}(x, \mathbf{v})_i$: niveau d'activation d'un neurone i à la fin de la séquence de mémoire (i.e. $t = T_{end}(x, \mathbf{v})$).

Nous comparerons différents types de fonction de fitness afin d'étudier l'impact celle-ci sur les résultats.

Fitness basée sur la trajectoire (Tb). La fitness basée sur la trajectoire récompense les individus sur la base de la distance par rapport à un comportement idéal connu. C'est l'objectif qui requiert le plus de connaissance a priori Cette fitness correspond à la fitness comportementale dans la classification de Nelson et al. (2009)

Expérience 1 : tâche de pointage. Correspond à la minimisation de la différence entre les angles de chaque articulation obtenus via cinématique inverse pour la position cible et les angles pour chaque articulation obtenus par évolution.

$$f(x) = \sum_{i=0}^{L=16} \sum_{j=0}^{M=4} \sum_{k=0}^{N=4} (|\Theta_{ijk} - \Phi_{ijk}(x, T)|) \tag{4.1}$$

Avec L le nombre de contextes, M, le nombre de séquences de lettres différentes, N, le nombre d'articulations, Θ, l'angle obtenu par un individu x à la fin de la simulation ($t = T_{end}$).

Expérience 2 : tâche de panneau de signalisation. Dans ce cas, nous voulons qu'un individu aille tout droit à la vitesse maximum dans le corridor, pour rapidement tourner sur la gauche/droite (suivant la séquence de lettres) à la fin du corridor pour enfin se diriger tout droit jusqu'à atteindre la fin de la branche. Pour cela nous voulons minimiser la fitness suivante :

$$f(x) = \sum_{i=0}^{M=16} \sum_{j=0}^{N=4} \sum_{t=0}^{T_{end}} \left(\begin{cases} \textbf{if } (\mathbf{v}_{ij} = \mathbf{v}_{ax}, p(x,t) = \alpha) \\ \quad 1 - \gamma_0(x, \mathbf{v}_{ij}, t) + \\ \quad 0.75 - \gamma_1(x, \mathbf{v}_{ij}, t) \\ \textbf{if } (\mathbf{v}_{ij} \neq \mathbf{v}_{ax}, p(x,t) = \alpha) \\ \quad 0.75 - \gamma_0(x, \mathbf{v}_{ij}, t) + \\ \quad 1 - \gamma_1(x, \mathbf{v}_{ij}, t) \\ \textbf{otherwise} \\ \quad \sum_{k=0}^{N_{out}} (1 - \gamma_k(x, \mathbf{v}_{ij}, t)) \end{cases} \right) \tag{4.2}$$

Avec $p(x,t)$ la position d'un individu x à t secondes et α la position correspondant à la fin du corridor.

Fitness basée sur la position finale (Ep). Cette fonction de fitness n'est pas liée à une distance par rapport à une trajectoire idéale, mais à la distance entre la position finale du robot et la position finale désirée du robot (position de la main dans expérience avec l'iCub et position du robot dans l'expérience avec le robot) :

$$f(x) = \sum_{i=0}^{L=16} \sum_{j=0}^{M=4} \left(\begin{cases} \textbf{if } (\mathbf{v}_i = \mathbf{v}_{ax}) \\ \quad d_{ij}(p(x, T_{end}), u_{ax}) \\ \textbf{otherwise} \\ \quad d_{ij}(p(x, T_{end}), u_{oth}) \end{cases} \right) \tag{4.3}$$

$d_{ij}(x,y)$ est une distance euclidienne, u_{ax}, la position attendue pour la séquence AX et u_{oth} pour les autres.

Cet objectif correspond à la fonction de fitness adaptée ("Tailored") selon Nelson et al. (2009).

Fitness Discrète (S). Dans ce cas, nous reprenons la fitness utilisée au chapitre précédent.

La fonction de fitness récompense seulement les individus qui se dirigent vers la bonne cible suivant la séquence de lettres. La fitness augmente de 3 pour la séquence AX et de 1 pour les autres séquences, la fitness maximum est donc de 6.

4.3.3 Diversité comportementale

La diversité comportementale est utilisée comme un objectif auxiliaire, cela mène à une *multiobjectivisation* (Knowles et al., 2001; Mouret et Doncieux, 2009a) dans laquelle deux objectifs sont maximisés :

$$\text{Maximize} \begin{cases} F(x) \\ Div(x) \end{cases} \tag{4.4}$$

Où $F(x)$ est la fitness d'un individu x et $Div(x)$ est une mesure de sa diversité.

4.3.4 Nouveauté

Lehman (2008) propose de mesurer la nouveauté $\rho(i)$ d'un individu i en calculant la distance comportementale entre i et ses k voisins les plus proches :

$$\rho(x) = \frac{1}{k}\sum_{j=0}^{k} d_b(x, \mu_j) \tag{4.5}$$

Où k est un paramètre défini à la main. μ_j est le j-ème voisin le plus proche de x en respectant la distance d_b. Les voisins sont calculés en utilisant la population courante et une archive d'individus sélectionnés. Un individu est ajouté dans l'archive si sa nouveauté est au-dessus d'un seuil minimum ρ_{min}.

4.3.5 Mesure du comportement et distance comportementale

Pour les deux expériences, le comportement est basé sur la position d'un individu à la fin de la simulation pour chaque contexte : position du robot dans la tâche du labyrinthe, et position de la main dans l'expérience avec l'iCub.

$$\mathbf{beh}(x) = \sum_{i=0}^{M=16}\sum_{j=0}^{N=4}\left(p(x, T_{end})\right) \tag{4.6}$$

Où $\mathbf{beh}(x)$ est le vecteur des différentes positions d'un individu x à la fin de la simulation pour chaque contexte.

Nous reprenons les conclusion de Bui et al. (2005); Mouret et Doncieux (2012) où l'objectif de diversité comportementale à maximiser $o_{bd}(x)$ est la distance comportementale moyenne par rapport au reste de la population P. Les contrôleurs sont optimisés via le schéma multi-objectif suivant :

$$\text{maximize} \begin{cases} F(x) \\ o_{bd}(x) = \frac{1}{size(P)} \sum_{y \in P} b_{dist}(x, y) \end{cases}$$

avec b_{dist} la distance comportementale utilisée pour comparer les individus et F, la fonction de fitness.

Les différentes configurations utilisés pour les deux expériences sont décrites dans le tableau 4.2. Chacune des approches est répétée 30 fois. Toutes les comparaisons entre ces approches sont effectuées à l'aide de tests de somme de rangs de Wilcoxon (ou test de Mann-Whitney).

4.3.6 Résultats

4.3.6.1 Influence de la fitness

L'ajout d'a priori dans la fitness ne permet pas d'obtenir de meilleurs résultats. Les figures 4.2 et 4.3 montrent les performances des meilleurs individus de chaque runs obtenues par les différentes configurations pour la tâche avec l'iCub et l'epuck. Tous les meilleurs individus de chaque run de chaque configuration sont

TABLE 4.2 – Description des différentes configurations utilisées pour les 2 expériences

# obj.	Nom	Description
1	Tb	Fitness basée sur la trajectoire
2	Tb+Div	Fitness basée sur la traj. avec obj. de diversité
2	Tb+Nov	Fitness basée sur la traj. avec obj. de nouveauté
1	Ep	Fitness basée sur la pos. finale
2	Ep+Div	Fitness basée sur la pos. finale avec obj. de diversité
2	Ep+Nov	Fitness basée sur la pos. finale avec obj. de nouveauté
1	S	Fitness discrète
2	S+Div	Fitness discrète avec obj. de diversité
2	S+Nov	Fitness discrète avec obj. de nouveauté
1	Novelty	Nouveauté seule

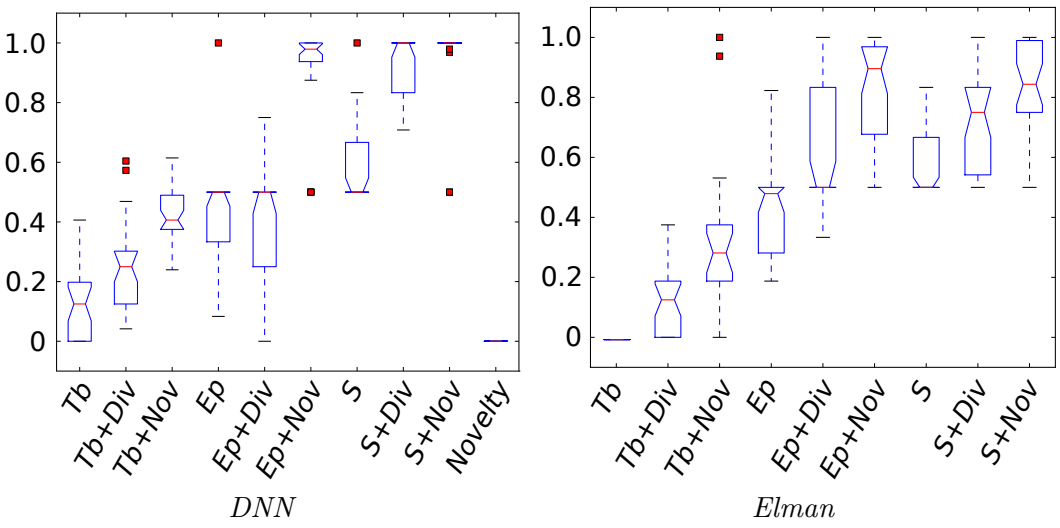


FIGURE 4.2 – Graphique comparant les meilleurs individus de chaque runs des différentes configurations (30 runs par configuration) sur l'expérience sur l'iCub . Ces individus sont comparés sur la fitness *S*.

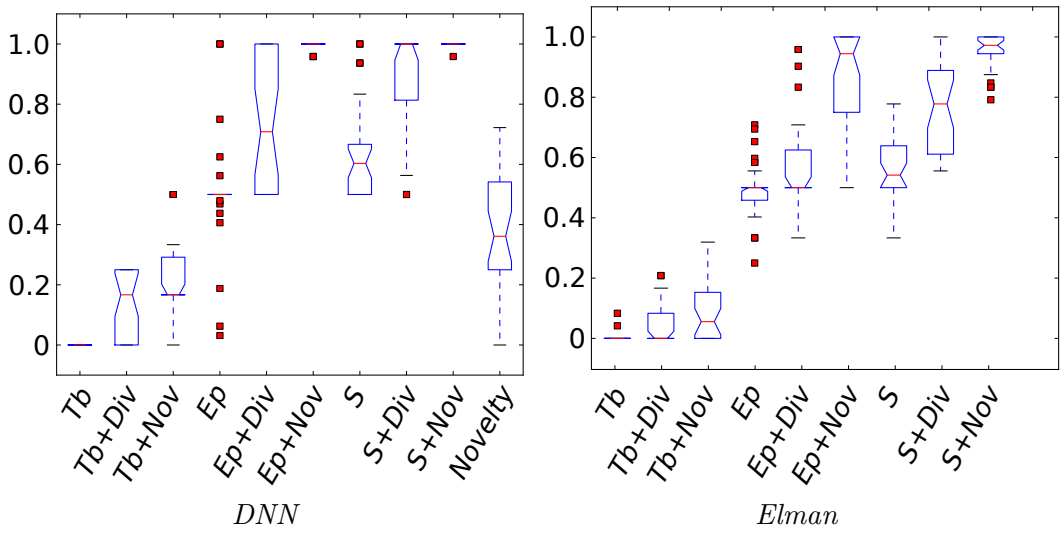


FIGURE 4.3 – Graphique comparant les meilleurs individus de chaque runs des différents configurations (30 runs par configuration) sur l'expérience avec l'epuck. Ces individus sont comparés sur la fitness *S*.

évalués sur la base de la fitness *S* pour comparer leur performance. Les tableaux 4.3 et 4.4 listent les p-values obtenues avec le test de Mann-Whitney sur respectivement l'expérience de pointage et l'expérience "panneau de signalisation".

Les résultats montrent que sur les deux tâches, quel que soit le type de fitness utilisé et sans l'aide d'un deuxième objectif, la plupart des runs de chaque configuration tombent dans un minimum local. Pour les deux tâches la fitness *Tb* obtient significativement (p-values < 0.0004) les résultats les plus bas avec une médiane de 0.15 pour la tâche de pointage et une mediane de 0 pour la tâche de "panneau de signalisation". Avec la fitness *Ep* les résultats sont légèrement meilleurs avec une médiane de 0.5 pour les deux tâches. *S* obtient des résultats significativement meilleurs avec une médiane de 0.5 et le premier quartile entre 0.65 et 0.85 dans l'expérience 1. Dans l'expérience 2 la médiane est légèrement supérieure, et le premier quartile similaire.

Les figures 4.4 et 4.5 nous donnent une information sur le comportement des meilleurs individus de chaque run pour les différentes fitness sans objectifs de diversité *Tb,Ep,S* sur les deux tâches.

La figure 4.5 décrit la position finale de la main du robot pour tous les meilleurs individus de chaque run pour les différentes configurations. Les points rouges représentent toutes les positions pour les séquences AX, les points verts représentant les autres séquences (AY,BX et BY).

Nous pouvons observer qu'avec *Tb* très peu d'individus sont capables d'atteindre la cible AX ou l'autre cible. Il n'y a pas de distinction claire entre la position de la main pour les séquence AX des autres séquences. Avec *Ep* la plupart des individus

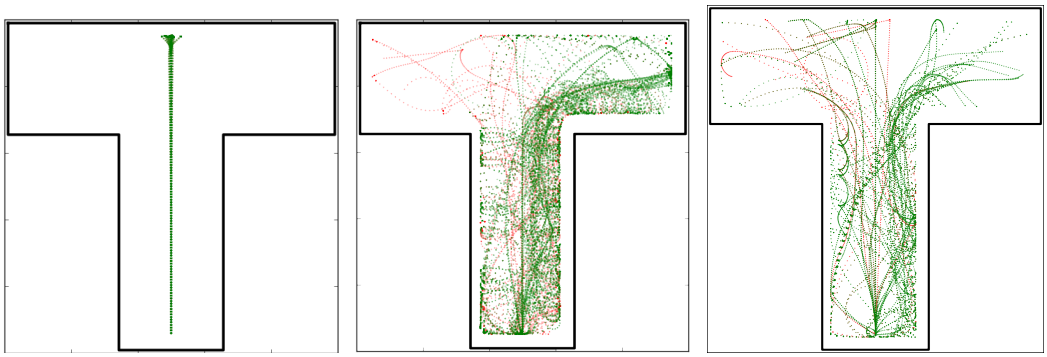


FIGURE 4.4 – Trajectoires obtenus par les meilleurs individus pour les différentes configurations (30 runs par configuration). Les points rouges correspondent aux trajectoires pour les séquences AX. Les points verts pour les autres séquences (AY,BX,BY). *Tb* correspond à la fitness basée sur la trajectoire, *Ep* la fitness basée sur la position finale et *S* la fitness discrète.

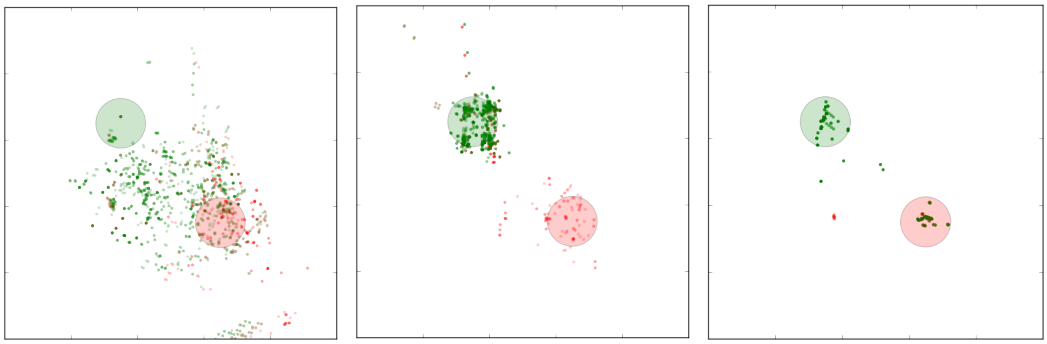


FIGURE 4.5 – iCub : Positions de la main à la fin de la simulation pour les meilleurs individus de chaque configuration utilisant un objectif de diversité *Tb,Ep,S* (30 runs par configuration). Les points rouges correspondent à la position de la main lors de la séquence AX et les points verts pour les autres séquences de lettres (AY,BX,BY).

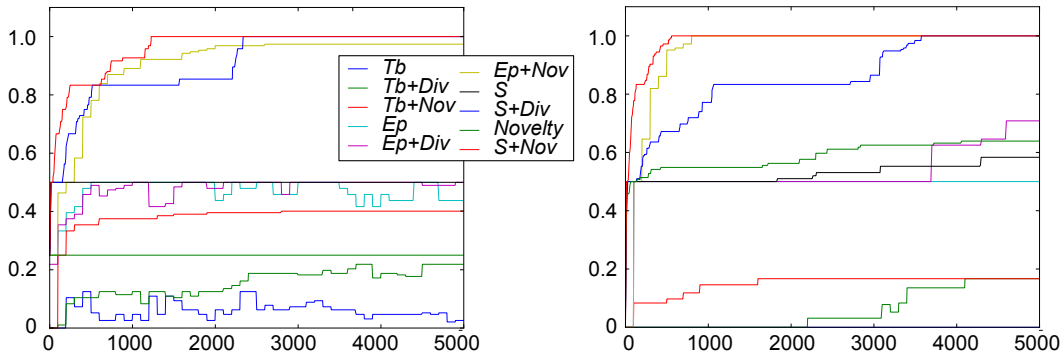


FIGURE 4.6 – Evolution de la médiane des meilleurs individus de chaque génération sur les différentes configurations. Le graphique à gauche représente les résultats pour l'expérience de pointage, à droite les resultats pour l'expérience "panneau de signalisation".

sont capable de minimiser la distance sur seulement une cible qui correspond à la séquence BX,BY ou AY. En effet atteindre la cible AX nécessite de développer une capacité de mémoire. Très peu d'individus atteignent la cible AX. Pour S les résultats sont similaires, la plupart des individus atteignent la même cible quelle que soit la séquence de lettre.

La figure 4.4 montre les trajectoires dans le labyrinthe en T obtenu par les meilleurs individus des différentes configurations : les trajectoires en rouge correspondent aux séquences AX et doivent rejoindre le côté gauche du labyrinthe en T ; les trajectoires vertes correspondent aux autres séquences et doivent atteindre le côté droit du labyrinthe en T.

Ici, la configuration *Tb* a convergé vers seulement des individus capables d'aller tout droit. Avec *Ep*, les individus ont convergé vers une forte tendance à tourner correctement sur la droite pour les séquences AY,BX,BY mais très peu d'individus ont développé la capacité de tourner à gauche pour les séquences AX. Pour S plus d'individus sont capables de tourner à gauche sans qu'il y ait de réelle distinction entre les différentes séquences.

Les graphiques 4.6 comparent l'évolution des meilleurs individus obtenus pour chaque configuration (*Tb, Tb+Div, Tb+Nov, Ep, Ep+Div, Ep+Nov, S, S+Div, S+Nov, Novelty*). Pour cette comparaison, les différents runs de chaque configuration sont évalués sur la fitness S et les graphiques affichent la médiane pour chaque configuration.

Nous pouvons voir pour l'expérience 1 (fig.4.6 à gauche), qu'avec *Tp* et *Ep* la médiane atteint le minimum local à 0.5 très rapidement. Celui-ci correspondant à atteindre la même cible quel que soit la séquence de lettres. Cette fitness est obtenue après environ 500 générations et reste constante même après 5000 générations. Les résultats sont similaires pour l'expérience 2 (fig. 4.6 à droite), ou le minimum local à 0.5 est atteint après seulement 100 générations. Même après 5000 générations,

et ce quelle que soit la fitness, sans objectif de diversité, les résultats ne sont pas satisfaisants.

En observant les résultats obtenus sur les deux tâches, nous pouvons voir que mettre un certain a priori dans la fitness ne donne pas nécessairement de meilleurs résultats. Les meilleurs résultats sont obtenus avec des comportements très différents de ceux attendus. Par exemple dans le cas du labyrinthe en T, les trajectoires des meilleurs individus sont rarement rectilignes, mais plus complexes.

4.3.6.2 Influence de la diversité comportementale

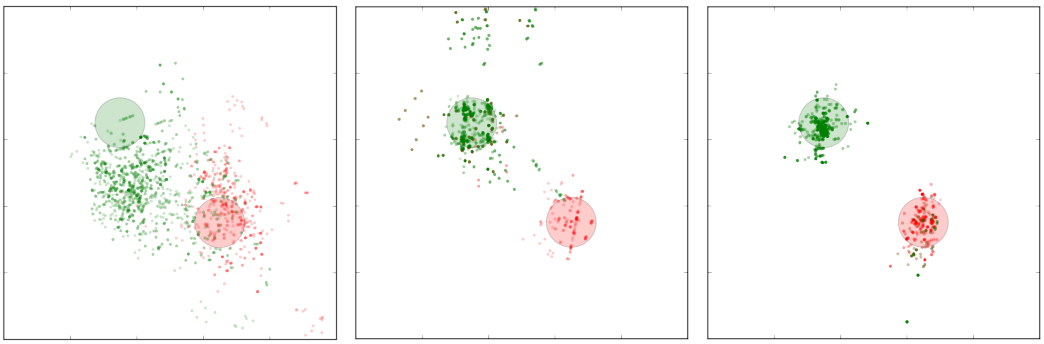
Utiliser un objectif de diversité comportementale augmente les performances sur la fitness. Quelle que soit la fitness, utiliser un deuxième objectif de diversité comportementale augmente les performances. Les figures 4.2 et 4.3 qui représentent les résultats des meilleurs individus de chaque configuration, montrent que l'objectif de diversité a un gros impact sur leur performance. Avec *S+Div*, la médiane atteint la valeur maximale sur les deux tâches. La différence avec *S* est très significative (p-value $< 10^-6$). Avec *Tb+Div*, les performances sont significativement meilleures qu'avec *Tb* (p-value < 0.0003), mais ce n'est pas suffisant pour obtenir des résultats satisfaisants. Pour la fitness *Ep*, utiliser un objectif de diversité ne génère pas nécessairement de meilleurs résultats : sur la tâche de pointage *Ep* et *Ep+Div* réalisent des résultats similaires (p-value $= 0.4756$), alors que sur la tâche avec l'epuck, la médiane augmente de 0.5 à 0.7 (p-value $< 10^-4$).

L'analyse des figures 4.6 qui comparent l'évolution de la performance pour les différentes configurations, nous enseigne différents points. Sur la tâche avec l'iCub, les performances de la configuration *S* suivent deux étapes. La première étape (de la génération 0 à la génération 2500) correspond à un individu capable d'atteindre les deux cibles sans utiliser de mémoire : dans ce cas précis, les individus atteignent la cible de gauche pour les séquences AX et BX et atteigne la cible de droite pour les séquences AY et BY. Dans un deuxième temps, les individus sont capables de mémoriser le stimulus A ou B, pour atteindre la bonne cible correspondant à la séquence BX.

On retrouve un comportement similaire sur la tâche de panneau de signalisation (figure 4.6 à droite) pour *S+Div* ou les deux mêmes étapes sont observées (après 3000 générations). Dans le cas de *Ep+Div*, le graphique montre qu'il faut plus de 3500 générations pour dépasser le minimum local à 0.5.

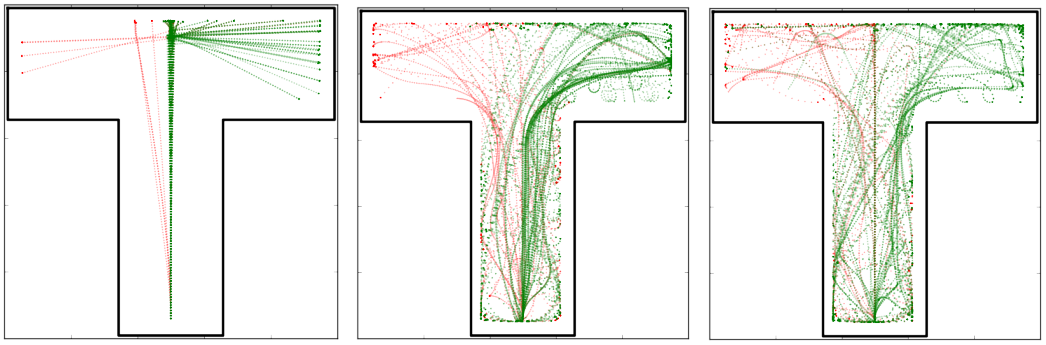
La figure 4.7 montre les positions de la main à la fin de la simulation pour les meilleurs individus utilisant un objectif de diversité avec 30 runs par configuration. Avec *Tb+Div*, l'objectif de diversité permet de distinguer deux aires entre les positions de la main des séquences AX et des autres séquences. Seulement les zones ne sont pas précises et ne correspondent pas au seuil pour chaque cible. Avec à l'objectif de diversité, *S+Div* obtient les meilleurs résultats, très peu de points se retrouvent en dehors des cibles.

La figure 4.8 montre les différentes trajectoires obtenues par les meilleurs individus sur les configurations utilisant un objectif de diversité. On peut distinguer dans



Tb+Div *Ep+Div* *S+Div*

FIGURE 4.7 – Positions de la main à la fin de la simulation pour les meilleurs individus de chaque configuration utilisant un objectif de diversité (30 runs par configuration). Les points rouges correspondent à la position de la main lors de la séquence AX et les points verts pour les autres séquences de lettres (AY,BX,BY).



Tb+Div *Ep+Div* *S+Div*

FIGURE 4.8 – Trajectoires obtenues par les meilleurs individus sur les configurations utilisant un objectif diversité *Tb+Div,Ep+Div,S+Div*. Les points rouges correspondents aux trajectoires pour les séquences AX, les points verts pour les autres séquences (AY,BX,BY).

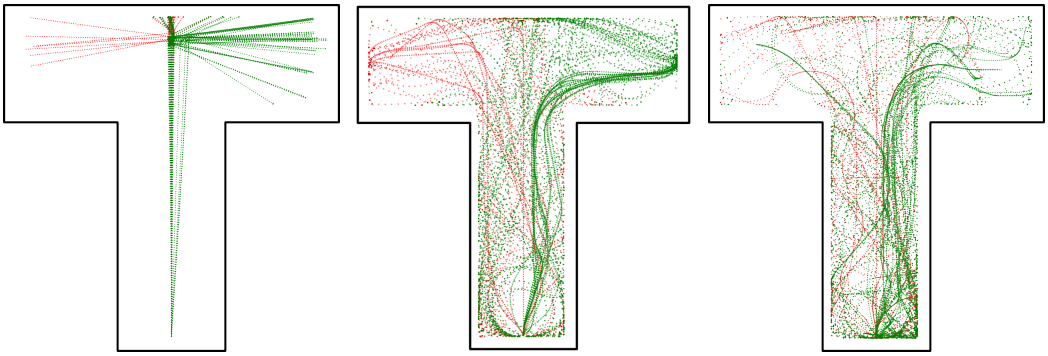


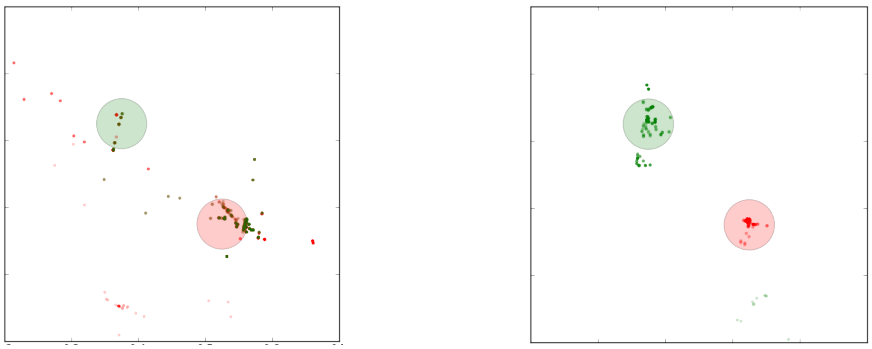
FIGURE 4.9 – Trajectoires obtenues par les meilleurs individus sur les configurations utilisant un objectif de nouveauté *Tb+Nov,Ep+Nov,S+Nov*. Les points rouges correspondent aux trajectoires pour les séquences AX, les points verts pour les autres séquences (AY,BX,BY).

la plupart des cas les trajectoires correspondant à la séquence AX des trajectoires correspondant aux autres séquences et ce quel que soit la configuration (*Tb,Ep* ou *S*).

L'ajout d'un objectif de nouveauté permet d'obtenir de meilleurs résultats qu'un objectif de diversité. Si l'on compare les résultats obtenus par *S+Nov* et *S+Div* sur les deux tâches (fig.4.3 et fig. 4.2), nous pouvons observer que les résultats sont significativement meilleurs avec *S+Nov* avec une p-value = 0.01 sur l'expérience 1 et une p-value de 0.00014 sur l'expérience 2. En effet, sur la tâche de pointage, grâce à l'objectif de diversité, moins de 60% des runs sont capables de produire des individus atteignant la fitness maximum contre environ 90% avec l'objectif de nouveauté. Sur la tâche "panneau de signalisation", la configuration *S+Div* obtient un peu plus de 60% des runs produisent des individus obtenant la fitness maximum contre près de 100% pour la configuration *S+Nov*.

De plus, si l'on observe l'évolution de la fitness pendant le processus d'optimisation (figs. 4.6), on constate qu'avec l'objectif de nouveauté, la médiane atteint la valeur maximale plus rapidement qu'avec l'objectif diversité : à la génération 1500 pour *S+Nov*, à la génération 2500 pour *S+Div* sur la tâche de pointage ; après seulement 900 générations pour *S+Nov* contre seulement à la fin du processus d'évaluation pour *S+Div* sur tâche dans le labyrinthe en T.

Un objectif de nouveauté seul ne permet pas d'obtenir de bonnes performances. Malgré le fait qu'un objectif de nouveauté fournit un bon moyen d'explorer l'espace de recherche, utiliser seulement un objectif de nouveauté, pour une tâche où l'espace de recherche est vaste, n'est pas suffisant. Si nous observons les performances obtenues pour *Novelty* sur la tâche 1 (fig. 4.3 et fig. 4.2), les résultats sont assez similaires (p-value = 0.052) à *Tb+Div* avec un valeur médiane de



Novelty *S+Nov*

FIGURE 4.10 – Positions de la main à la fin de la simulation pour les meilleurs individus des configurations *Novelty* et *S+Nov* (30 runs par configuration). Les points rouges correspondent à la position de la main lors de la séquence AX et les points verts pour les autres séquences de lettres (AY,BX,BY).

0.25. Sur la tâche "panneau de signalisation", les résultats sont meilleurs, avec une médiane de 0.65 similaire aux performances de *Ep* (p-value = 0.43), ce qui reste faible si l'on compare aux performances de *S+Div* et *S+Nov*.

Les performances pour *Novelty* sur la tâche 1 (fig. 4.6) sont constantes durant tout le processus d'optimisation de la génération 0 à 5000, la médiane est de 0.25. Sur la tâche 2 (fig. 4.6 à droite), la performance progresse légèrement durant le run, de 0.5 au départ à 0.65 à la fin de l'optimisation.

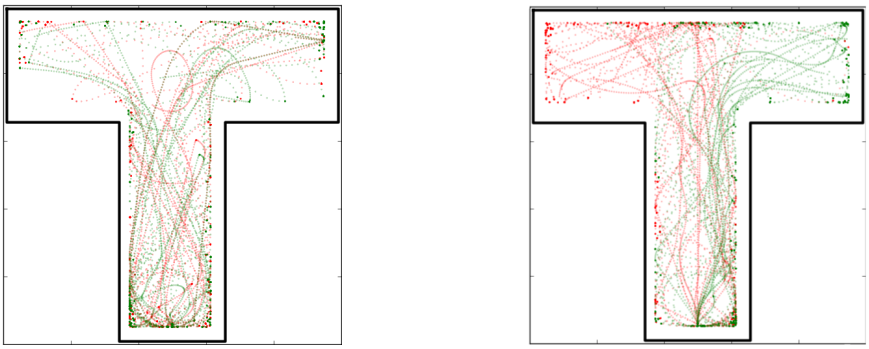
La figure 4.10 affiche les positions de la main à la fin de la simulation pour *Novelty* et *S+Nov* sur la tâche de pointage. On constate que pour *Novelty*, les positions des meilleurs individus sont très disparates, peu d'agents sont capables d'atteindre les cibles correctement.

4.3.6.3 Etude de la capacité de mémoire

Pour évaluer la capacité de mémoire indépendamment des résultats sur la tâche, nous utilisons le test de mémoire décrit à la section 3.4.1.2. Celui-ci détecte la présence d'un neurone qui décharge différemment lors de la séquence AX par rapport aux autres séquences. Afin de vérifier la stabilité de cette mémoire, le délai entre la présentation des lettres est étendu de 50 à 400 pas de temps.

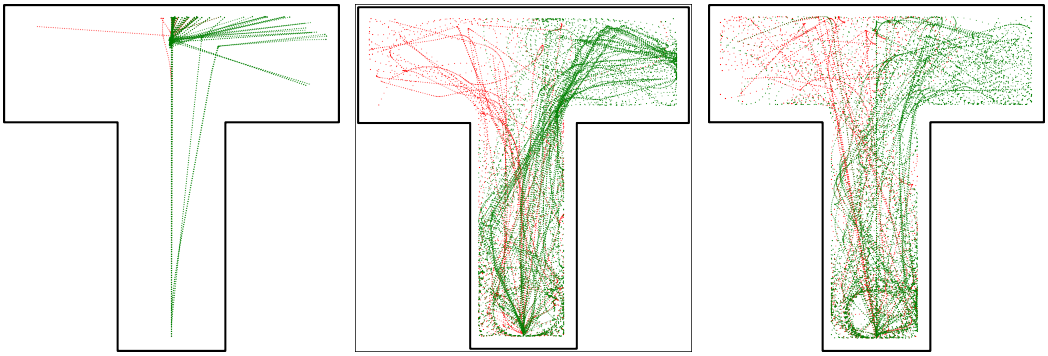
La figure 4.15 montre l'évolution du meilleur individu de chaque génération pour les différentes configurations capables de réussir ce test. Les figures 4.14 et 4.13 montrent les meilleurs individus de chaque run pour les différentes configurations. La première barre représente le pourcentage d'individus ayant la fitness maximum, la seconde représente le pourcentage d'individus ayant à la fois la fitness maximum et la capacité de mémoire.

La diversité comportementale aide à obtenir des individus avec une capacité de mémoire stable. Les figures 4.13 et 4.14 montrent sans surprise que



Novelty *S+Nov*

FIGURE 4.11 – Trajectoires obtenues par les meilleurs individus sur les configurations utilisant un objectif diversité *Tb+Div,Ep+Div,S+Div*. Les points rouges correspondent aux trajectoires pour les séquences AX, les points verts pour les autres séquences (AY,BX,BY).



Tb+Nov *Ep+Nov* *S+Nov*

FIGURE 4.12 – Trajectoires obtenues par les meilleurs individus sur les configurations utilisant un objectif de nouveauté *Tb+Nov,Ep+Nov,S+Nov* et le réseau Elman. Les points rouges correspondents aux trajectoires pour les séquences AX, les points verts pour les autres séquences (AY,BX,BY).

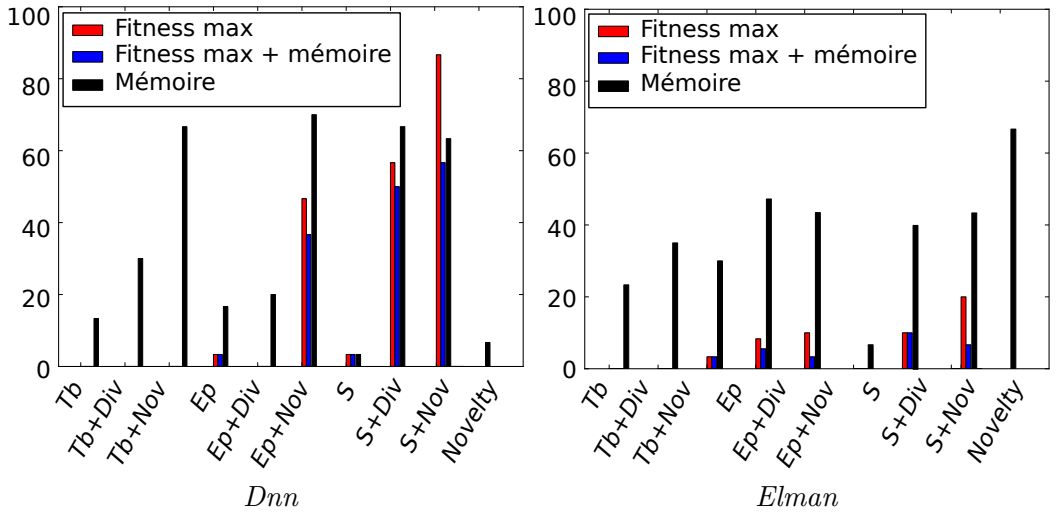


FIGURE 4.13 – Graphiques montrant les meilleurs individus obtenus par les différentes configurations sur l'expérience 1.

S+Nov et *S+Div* sont les configurations qui obtiennent le plus d'individus passant le test de capacité de mémoire : plus de 60% des meilleurs individus pour les deux configurations sur la tâche de pointage. Sur la tâche "panneau de signalisation", les résultats sont meilleurs avec *S+Nov* (50%) qu'avec *S+Div* (30%). Cela montre que l'utilisation d'un objectif de diversité ou de nouveauté améliore de manière significative l'obtention d'individus dotés d'une certaine capacité de mémoire. Alors qu'avec la configuration *S*, sur les deux tâches, les individus ne sont pas capables de développer une capacité de mémoire.

Nous pouvons constater qu'avec un objectif de nouveauté, quel que soit le type de fitness utilisé *Tb, Ep* ou *S*, la proportion d'individus réussissant le test de mémoire demeure élevé (plus de 60%) et ce même si la fitness max n'est jamais atteinte, comme pour *Tb+Nov*.

Sans développer une capacité de mémoire, la fitness maximum qu'un individu peut obtenir est de 0.8. Cela correspond à tourner à gauche pour les séquences AX et BX et tourner à droite pour les séquences AY, BY. Dans ce cas, il n'est pas nécessaire de mémoriser A ou B, c'est seulement la lettre X ou Y qui est prise en compte. Sur la tâche avec l'iCub, nous pouvons voir en comparant 4.15 et 4.6 que le développement d'une capacité de mémoire au sein des individus apparait juste avant que la fitness dépasse 0.8. En effet la médiane atteint la fitness maximum à la génération 2300 juste après que plus de la moitié des individus obtiennent une capacité de mémoire, à la génération 1600. Nous pouvons observer le même fonctionnement sur *S+Nov* : l'augmentation des individus dotés d'une capacité de mémoire (de la génération 0 à la génération 800) précède le moment où la médiane atteint le seuil maximum (à la génération 1000).

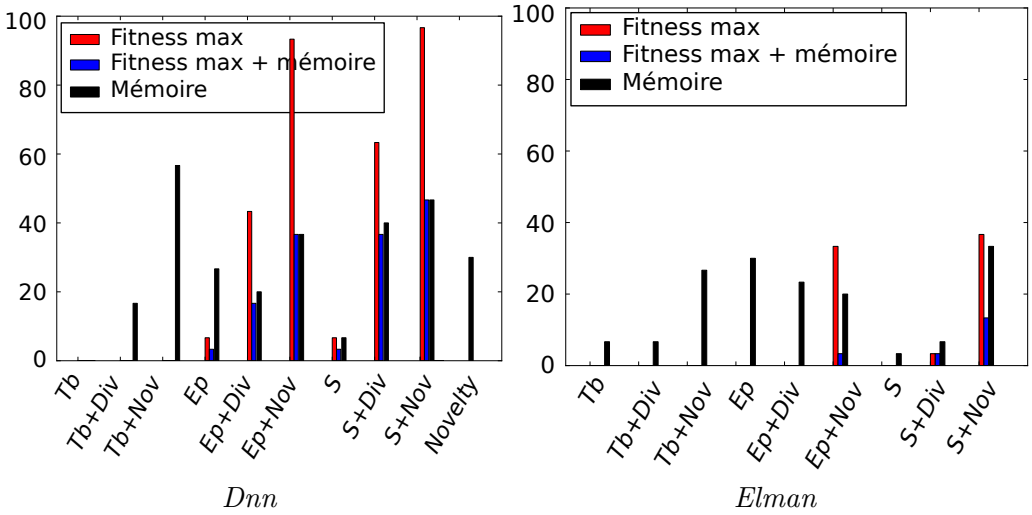


FIGURE 4.14 – Graphiques montrant les meilleurs individus obtenus par les différentes configurations sur l'expérience "panneau de signalisation".

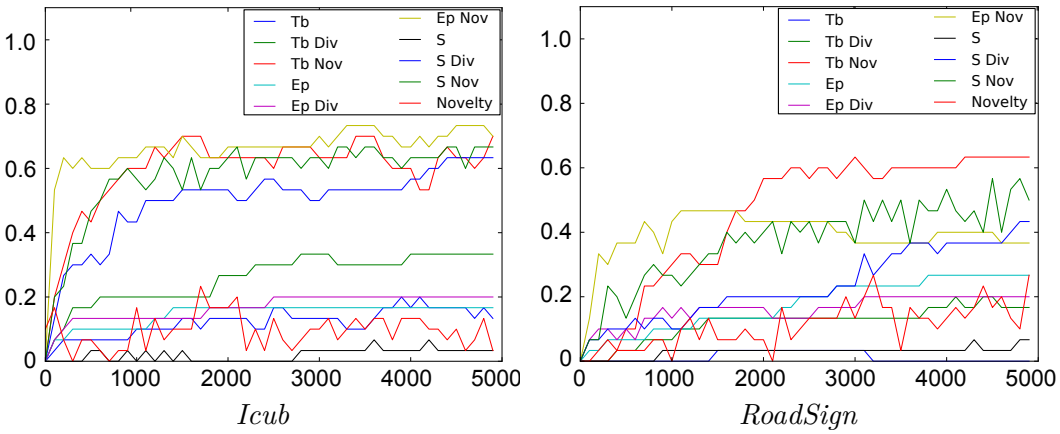
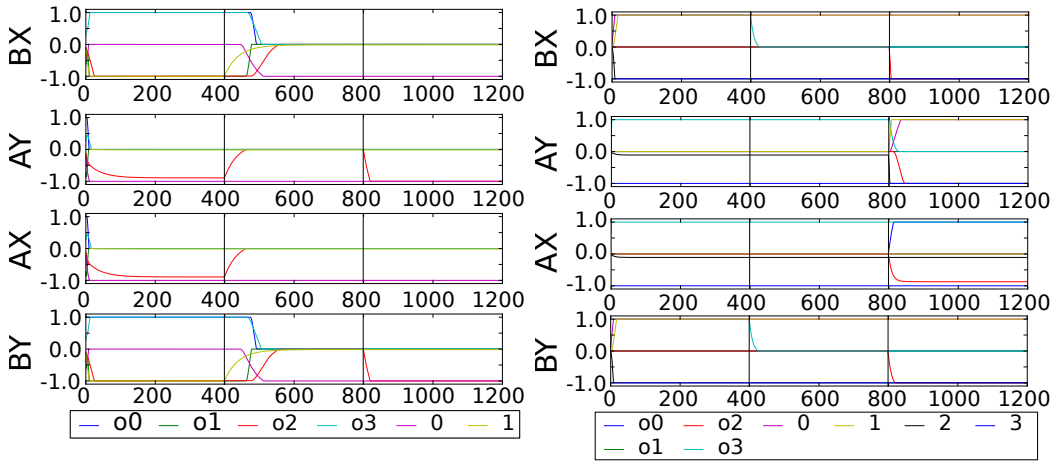


FIGURE 4.15 – Proportion des individus passant le test de mémoire pendant le processus d'optimisation. Toutes les 100 générations, le meilleur individu de chaque génération est testé sur le test de mémoire.

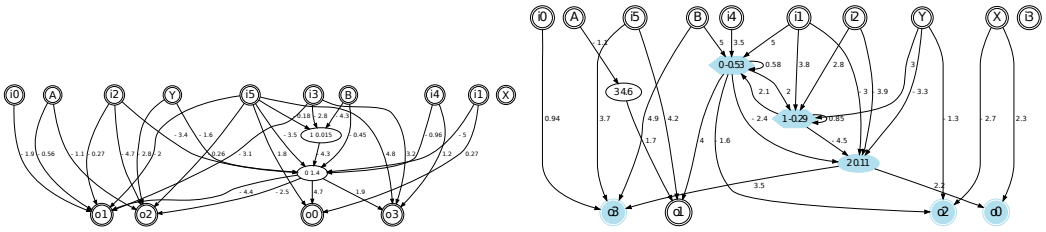


Capacité de mémoire fragile *Capacité de mémoire robuste*

FIGURE 4.16 – Chronogramme de l'activité neuronale durant le test de mémoire. Chaque graphe illustre l'évolution de l'activité neuronale pour chaque séquence AX, AY, BX, BY.

Certains individus qui réussissent la tâche sont dotés d'une capacité de mémoire fragile. Sur les figures 4.13 et 4.14 nous pouvons observer plusieurs individus qui réussissent la tâche sans être capable de passer le test de mémoire. En effet, pour *S+Nov* sur la tâche de pointage, par exemple, 30% des individus obtiennent la fitness maximum, sans réussir le test de mémoire. Sur la tâche "panneau de signalisation" seulement la moitié des meilleurs individus ont passé le test de mémoire pour *S+Nov*.

La figure 4.16 décrit le chronogramme d'activité neuronale pour deux individus réussissant la tâche, lors du test de mémoire : celui à gauche ne passe pas le test de mémoire, et celui de droite le réussit. Grâce à ce graphique, nous pouvons comprendre comment un individu est capable de réussir la tâche sans développer une mémoire robuste. A gauche, le réseau n'est pas robuste à l'extension du délai entre la présentation des deux lettres. Durant la présentation de la première lettre nous pouvons distinguer deux types d'activités entre les séquences A et B. Durant la phase de délai entre la présentation des deux lettres, les activités particulières relatives aux séquences avec la lettre B ne sont maintenues que pendant 50 pas de temps. Juste après ce délai, l'activité neuronale devient similaire à celles des séquences contenant la lettre A. Lors de la présentation de la seconde lettre l'activité neuronale ne permet plus de distinguer les séquence AX et BX. La topologie du réseau de cet individu ne contient pas de neurones possédant une récurrence capable de maintenir en mémoire de manière stable les lettres A ou B. Le réseau de droite possède au contraire deux neurones dotés de connexions récurrentes permettant de maintenir de manière stable la lettre B pendant la période de délai.



Réseau doté d'une mémoire fragile *Réseau doté d'une mémoire robuste*

FIGURE 4.17 – Exemple de topologie de réseaux obtenus sur tâche de pointage.

TABLE 4.3 – P-values obtenues sur la tâche de pointage avec le test de Mann-Whitney.

	Tb+Div	*Tb*	*Ep+Div*	*Ep*	*Novelty*	*S+Nov*	*S+Div*	*S*
Tb+Div	x	0.00038	0.00029	1e-05	0.05258	0.0	0.0	0.0
Tb	0.00038	x	0.0	0.0	0.0	0.0	0.0	0.0
Ep+Div	0.00029	0.0	x	0.4756	0.0007	0.0	0.0	3e-05
Ep	1e-05	0.0	0.4756	x	1e-05	0.0	0.0	1e-05
Novelty	0.05258	0.0	0.0007	1e-05	x	0.0	0.0	0.0
S+Nov	0.0	0.0	0.0	0.0	0.0	x	0.01256	0.0
S+Div	0.0	0.0	0.0	0.0	0.0	0.01256	x	0.0
S	0.0	0.0	3e-05	1e-05	0.0	0.0	0.0	x

TABLE 4.4 – P-values obtenues sur l'expérience "panneau de signalisation" avec le test de Mann-Whitney.

	Tb+Div	*Tb*	*Ep+Div*	*Ep*	*Novelty*	*S+Nov*	*S+Div*	*S*
Tb+Div	x	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tb	0.0	x	0.0	0.0	0.0	0.0	0.0	0.0
Ep+Div	0.0	0.0	x	5e-05	0.06185	2e-05	0.06435	0.06753
Ep	0.0	0.0	5e-05	x	0.00063	0.0	0.0	0.00022
Novelty	0.0	0.0	0.06185	0.00063	x	0.0	0.0	0.43518
S+Nov	0.0	0.0	2e-05	0.0	0.0	x	0.00014	0.0
S+Div	0.0	0.0	0.06435	0.0	0.0	0.00014	x	4e-05
S	0.0	0.0	0.06753	0.00022	0.43518	0.0	4e-05	x

4.4 Conclusion

A travers deux expériences sur robots simulés basées sur le protocole AX-CPT, une tâche de navigation dans un labyrinthe en T et une autre correspondant à une tâche de pointage, nous avons pu observer l'influence de l'insertion d'a priori dans la fonction de fitness et de l'ajout d'un objectif de diversité comportementale.

Sur les deux expériences, l'ajout d'a priori dans la fitness que ce soit en guidant la trajectoire à obtenir (configuration *Tb*) ou en guidant la position finale à obtenir (configuration *Ep*) a donné de moins bons résultats qu'une fitness non guidée (configuration *S*) ne récompensant que le succès ou l'échec de la tâche. En effet ces deux fitness tendent vers des minima locaux dont il est difficile de s'extraire.

L'ajout d'un objectif de diversité comportementale, que ce soit avec archive, objectif de nouveauté (*Nov*) ou sans archive, objectif de diversité (*Div*), est quant à lui très efficace pour obtenir efficacement des individus réussissant la tâche. Nous avons pu observer que l'objectif de nouveauté permettait d'obtenir de meilleures performances que l'objectif de diversité. Cependant, du fait de l'espace de recherche important, l'utilisation d'un objectif de nouveauté seul, sans aide d'une fitness de haut niveau (*S*) ne permet pas d'obtenir de résultats satisfaisants. L'utilisation de réseaux de neurones récurrents (Elman), censés favoriser l'apparition d'une capacité de mémoire, n'a pas permis d'obtenir de résultats satisfaisants. Les performances obtenues avec ce type de réseau et ceci quelles que soient les configurations utilisées sur les deux expériences, sont bien inférieures à celles obtenues en faisant évoluer la topologie (*DNN*). Au niveau de la capacité de mémoire obtenue au sein des individus, nous avons pu observer que sur les différentes fitness, l'ajout d'un objectif de diversité a permis d'obtenir plus d'individus réussissant le test de mémoire. En revanche l'utilisation de réseau d'Elman a contre-intuitivement donné de moins bons résultats sur ce test.

Nous pouvons donc conclure que l'utilisation d'une fitness discrète de haut niveau combinée avec l'ajout d'un objectif de nouveauté pour optimiser à la fois la topologie et les paramètres du réseau de neurones est la formule donnant les meilleurs résultats.

Il reste cependant différents points à améliorer. En effet, de nombreux agents ont pu réussir les différentes tâches en disposant d'une capacité de mémoire fragile. Pour favoriser l'apparition d'individus dotés de capacité de mémoire robuste, il peut sembler intéressant d'étudier l'influence :

- de l'ajout de différents objectifs auxiliaires dédiés à l'émergence de la mémoire.
- de l'évaluation des individus dans un nombre plus grand de contextes. Cette solution pourrait aider à favoriser l'apparition de mémoire plus robuste. Cependant celle-ci nécessite un grand nombre d'évaluations et est difficilement utilisable dans des temps raisonnables.

Ces deux propositions seront étudiées dans le chapitre suivant.

Chapitre 5

Synthèse de mémoire et objectifs auxiliaires

Objectifs Favoriser l'apparition d'individus dotés d'une mémoire robuste.

Démarche Proposition et étude de l'influence de pressions de sélection :
- méthode *ProGAb*[a] qui récompense les individus dotés d'une capacité de généralisation ;
- méthode *Coher*[b] qui récompense la cohérence des comportements des individus dans des environnements similaires.

Résultats
- L'utilisation de pressions spécifiques a permis de favoriser l'émergence d'agents dotés d'une forme de mémoire robuste ;
- Un lien a pu être mis en évidence entre capacité de généralisation et mémoire.

a. Ce travail a été effectué en collaboration avec S. Koos (Pinville et al., 2011).
b. Ce travail a été effectué en collaboration avec C. Ollion (Ollion et al., 2012).

Sommaire

5.1 Introduction

Les résultats obtenus au chapitre précédent ont montré que la combinaison entre une fitness de haut niveau et d'un objectif de maintien d'une diversité comportementale permettait d'obtenir des individus capables de réaliser une tâche nécessitant une forme de mémoire. Cependant après une caractérisation de la forme de mémoire obtenue au sein des individus, nous avons pu observer que celle-ci pouvait dans certains cas être fragile.

Afin de favoriser l'apparition d'individus dotés d'une mémoire robuste, nous proposons dans ce chapitre l'utilisation de deux pressions de sélection spécifiques :

- une pression favorisant la capacité de généralisation des individus : l'idée sous-jacente dans ce choix est d'hypothétiser une corrélation entre capacité de généralisation d'un individu et capacité de mémoire robuste [1] ;
- une pression favorisant la cohérence des comportements des individus dans des environnements similaires : dans ce cas l'hypothèse serait d'envisager un lien entre des individus dotés d'un comportement proche dans des environnements similaires et une capacité de mémoire robuste.

Pour ce faire, nous proposons deux méthodes différentes :

- méthode *ProGab* qui récompense la capacité de généralisation des individus ;
- méthode *Coher* qui récompense les individus ayant des comportements proches dans des environnements similaires.

5.2 Méthode *ProGab*

5.2.1 Capacité de généralisation

Le plus souvent en robotique évolutionniste durant le processus d'optimisation, un contrôleur est évalué sur plusieurs contextes [2] (Jakobi, 1997, 1998; Berlanga et al., 2002; Ziemke et Thieme, 2002; Barate et Manzanera, 2008; Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012), voire sur un seul contexte (Reynolds, 1993; Miglino et al., 1995; Jakobi et al., 1995; Kondo et al., 1999; Floreano et Mondada, 1998; Floreano et Urzelai, 2001; Urzelai et Floreano, 2001). Le meilleur contrôleur est donc sélectionné sur des comportements spécifiques à quelques contextes précis et il n'y donc aucune garantie qu'il se comporte aussi efficacement sur des contextes différents qu'ils soient proches ou non. Dans de nombreuses applications robotiques, le robot doit gérer de multiples situations sur lesquelles il n'a pas forcément été testé lors de l'optimisation : le robot doit donc faire preuve de capacités de généralisation afin d'assurer une performance élevée dans toutes les situations rencontrées. Prenons par exemple le cas d'un robot devant se déplacer dans une pièce : dans l'idéal, sa performance de navigation ne doit pas dépendre de la taille de la pièce, de sa position initiale ou encore de l'agencement précis des obstacles.

1. Cette hypothèse vaut bien sûr dans le cas d'une tâche nécessitant une forme de mémoire.
2. comme évoqué précédemment, nous définissons ici un contexte comme un ensemble de paramètres tels que la position et l'angle initial du robot, la taille de l'environnement...

L'objectif est donc d'évaluer les contrôleurs sur un nombre représentatif de contextes. Cette évaluation de la capacité de généralisation des individus sera potentiellement plus précise si l'on augmente la taille de cet ensemble de contextes. Dans un processus évolutionniste classique, où chaque contrôleur est évalué sur chacun des contextes, il est évident qu'utiliser un plus grand nombre de contextes va augmenter sensiblement le nombre total d'évaluations : le processus d'optimisation peut en être considérablement ralenti. En pratique, les travaux en robotique évolutionniste qui visent à améliorer les capacités de généralisation des contrôleurs en augmentant directement le nombre de contextes d'évaluation se limitent souvent à moins de 10 contextes (Jakobi, 1998; Berlanga et al., 2002; Ziemke et Thieme, 2002; Barate et Manzanera, 2008; Doncieux et Mouret, 2010b) (voir le tableau 5.1 pour un résumé de différents travaux).

Bien qu'une dizaine de contextes puissent suffire dans les applications présentées dans ces travaux, une telle méthode ne peut pas être utilisée de manière générique si on souhaite maintenant généraliser sur plusieurs centaines de contextes. La question de doter les contrôleurs de capacités de généralisation est donc directement reliée au besoin de limiter le nombre d'évaluations pendant l'optimisation.

Une autre limitation concernant la plupart des travaux de la littérature est que la capacité de généralisation des contrôleurs obtenus est rarement évaluée. Les solutions sont uniquement testées dans les contextes sur lesquels elles ont été optimisées (Jakobi, 1998; Berlanga et al., 2002; Ziemke et Thieme, 2002). Les contrôleurs sélectionnés ont donc des capacités de généralisation apparemment bonnes. Néanmoins, comme ils ont été directement optimisés sur les contextes de tests, on ne peut rien affirmer concernant leurs capacités de généralisation effectives sur de nouveaux contextes.

Ce problème de généralisation peut être interprété comme un cas typique de sur-apprentissage. Un contrôleur qui maximise la performance sur l'ensemble des contextes d'apprentissage risque d'exploiter des spécificités de ces contextes : il pourra ne pas être du tout performant dans des contextes différents où ces spécificités n'apparaissent pas. Autrement dit, une performance optimale sur l'ensemble d'apprentissage ne donnera aucune garantie quant à la performance sur d'autres contextes du contrôleur obtenu.

L'augmentation du nombre de contextes évalués durant la période d'optimisation semble être une solution permettant d'éviter ces problèmes de surapprentissage et garantir une capacité de généralisation importante. Cependant cette approche comporte une limitation importante : l'augmentation du nombre de contextes évalués durant le processus d'optimisation est extrêmement coûteux computationnellement. Du fait de l'utilisation d'algorithmes évolutionnistes qui nécessitent un grand nombre d'évaluations, il demeure difficilement possible de pouvoir évaluer un individu sur une centaine de contextes différents dans un temps raisonnable.

Afin de remédier à cette limitation, nous proposerons dans la section suivante une méthode basée sur une approximation de l'évaluation de la capacité de généralisation d'un individu, la méthode *ProGAb*.

Avant d'introduire les différents travaux effectués sur l'amélioration des capa-

cités de généralisation des contrôleurs, nous allons d'abord proposer une définition formelle de ce que nous entendons par capacité de généralisation.

5.2.2 Formalisation de la capacité de généralisation

Comme le phénomène de généralisation a beaucoup plus été étudié dans le cadre de l'apprentissage supervisé, notre définition s'inspire de la méthodologie standard en apprentissage supervisé définissant trois ensembles de données (Alpaydin, 2004; Gagné et Schoenauer, 2006) :

- un ensemble de contextes d'apprentissage Ω_{train} utilisé pour optimiser les solutions ;
- un ensemble de contextes de validation Ω_{valid} permettant d'estimer la capacité de généralisation de solutions prometteuses sur Ω_{train} ;
- un ensemble de contextes de test Ω_{test} permettant, une fois l'optimisation terminée, d'évaluer la capacité de généralisation sur de nouveaux contextes des meilleures solutions trouvées.

Dans le cadre de la robotique évolutionniste, un contexte ω est une instance de la tâche à résoudre et correspond à un ensemble de valeurs spécifiques pour chacun des paramètres de la tâche : par exemple, les coordonnées initiales du robot, les dimensions de l'environnement, ... Comme à la fois l'ensemble d'apprentissage Ω_{train} et l'ensemble de validation Ω_{valid} sont utilisés pour évaluer les solutions pendant l'optimisation, nous introduisons aussi la notion d'ensemble d'évaluation $\Omega_{eval} = \Omega_{train} \cup \Omega_{valid}$ qui contient tous les contextes pouvant servir à évaluer une solution pendant l'optimisation.

Soit $\mathcal{F}$ la fonction de fitness associée à la tâche et x un individu quelconque de l'espace des contrôleurs $\mathcal{C}$, la capacité de généralisation G de x sur un ensemble de contextes Ω est simplement définie comme la somme des performances de x sur toutes les contextes de Ω :

$$G(x, \Omega) = \sum_{\omega \in \Omega} \mathcal{F}(x, \omega)$$

Un individu optimalement général x doit alors vérifier les deux contraintes ci-dessous :

$$\forall y \in \mathcal{C}, \quad G(y, \Omega_{eval}) \leq G(x, \Omega_{eval}) \tag{C1}$$

$$\frac{G(x, \Omega_{eval})}{|\Omega_{eval}|} \simeq \frac{G(x, \Omega_{test})}{|\Omega_{test}|} \tag{C2}$$

Une solution optimalement générale doit donc se comporter au mieux sur les contextes d'évaluation issus de Ω_{train} et Ω_{valid} (C1) et obtenir des performances similaires sur les contextes de test (C2).

Robustesse d'une solution. Nous considérons la robustesse comme un sous ensemble du concept de généralisation : un individu robuste est capable de généraliser

à des contextes légèrement différents que ceux vus durant le processus d'optimisation.

Selon Jin et Branke (2005), une solution optimale est robuste si :
- elle est insensible à des petites variations des paramètres du génotype ;
- elle est insensible à des petites variations des paramètres de l'environnement ou à des perturbations spécifiques de l'environnement (Kondo et al., 1999).

5.2.3 Promotion de la capacité de généralisation

Nous allons décrire différentes méthodes utilisées en ER afin de favoriser l'apparition d'individus dotés d'une capacité de généralisation.

Ajout de bruit. L'ajout d'un bruit au niveau des capteurs et/ou des effecteurs a été une des premières méthodes utilisée pour favoriser la robustesse (Cliff et al., 1993; Reynolds, 1993; Miglino et al., 1995; Jakobi et al., 1995; Jakobi, 1998).

L'approche proposée par Reynolds (1993) consiste à ajouter du bruit sur les entrées des contrôleurs tout en n'évaluant les solutions que dans un unique contexte. Dans ce travail, le but est d'obtenir des comportements de suivi de couloirs robustes à la variabilité des valeurs indiquées par les capteurs de distance d'un robot simulé. Des approches similaires ont été appliquées au problème du passage de la simulation à la réalité (Miglino et al., 1995; Jakobi et al., 1995) : afin de trouver des contrôleurs efficaces sur le robot physique (qui fait office de contexte de test), du bruit est utilisé pour éviter de prendre en considération les spécificités de la simulation. Les solutions optimisées ne peuvent alors pas exploiter ces spécificités pour obtenir de bonnes performances et développent donc des comportements plus robustes.

Changement de contextes. Dans ce cas, l'environnement est modifié durant l'évaluation. On retrouve notamment cette utilisation dans les travaux de Nolfi et Parisi (1996) où l'environnement alterne à chaque génération entre deux environnements (simulation alternance "jour" et nuit"). Pour l'expérience de Urzelai et Floreano (2001), c'est la position de la lumière qui est changée.

Evaluation dans différents contextes. L'évaluation d'un individu dans différents contextes est une des méthodes les plus usitées en robotique évolutionniste (Jakobi, 1997, 1998; Berlanga et al., 2002; Ziemke et Thieme, 2002; Barate et Manzanera, 2008; Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012). Cette approche permet de se prémunir des risques de surapprentissage. Cependant le nombre de contextes utilisé durant l'évaluation reste le plus souvent limité (le plus souvent < 10) du fait du coût computationnel élevé. La table 5.1 liste un échantillon d'articles avec le nombre de contextes utilisé durant l'expérience.

Incréments environnementaux. Voir section 4.1.1 pour plus de détails. Dans ce cas, l'environnement d'évaluation se complexifie à chaque étape de l'optimisation : le robot doit alors résoudre la tâche considérée dans un environnement simple, puis

TABLE 5.1 – Panel d'articles qui traitent de problèmes de robotique évolutionniste. Les colonnes correspondent aux trois questions suivantes. (1) La capacité de généralisation des contrôleurs est-elle évaluée au cours de l'optimisation ? (2) Si oui, dans combien de contextes d'évaluation ($|\Omega_{eval}|$) ? (3) La capacité de généralisation des solutions est-elle testée après l'optimisation (Ω_{test}).

Papiers	Eval. de la généralisation ?		$\lvert\Omega_{eval}\rvert$	Ω_{test}
	non	oui		
Jakobi et al. (1995)	x		1	x
Miglino et al. (1995)	x		1	
Jakobi (1997)		x	10	x
Kondo et al. (1999)		x	10	x
Di Paolo (2000)	x		1	x
Floreano et Urzelai (2001)	x		1	x
Berlanga et al. (2002)		x	6	x
Ziemke et Thieme (2002)		x	3	
Barate et Manzanera (2008)		x	4	x
Mouret et Doncieux (2012)		x	3	
Lehman et Stanley (2010)	x		1	
Bongard (2011b)	x		1	x

de plus en peu plus réaliste (Miglino et al., 1998; Nakamura et al., 2000; Barlow et Oh, 2006).

Combiner évolution et apprentissage durant la vie de l'individu. Une alternative proposée par Floreano et Mondada (1998) et Floreano et Urzelai (2001) consiste à intégrer des mécanismes d'adaptation au niveau du contrôleur, par exemple en utilisant des réseaux de neurones dits plastiques où à chaque neurone est associé une règle d'apprentissage hebbienne modifiant les différents poids synaptiques du neurone au cours de l'évaluation. De tels réseaux permettent notamment de s'adapter rapidement à des configurations aléatoires de l'environnement dans une tâche de navigation avec un robot mobile Khepera (Urzelai et Floreano, 2001). A chaque génération, la performance d'un réseau est calculée dans un seul contexte. En évaluant les contrôleurs dans un unique contexte, le processus d'optimisation ne peut exploiter aucune information sur la capacité de généralisation des solutions. Il n'y a donc aucune garantie qu'au terme de l'optimisation, les contrôleurs obtenus soient effectivement robustes. Nous pouvons aussi citer les travaux de Floreano et Urzelai (2001); Floreano et Nolfi (2000); Di Paolo (2000); Smith et al. (2002). D'autres travaux inspirés de la fonction de réarrangement dynamique du système nerveux stomatogastrique du homard proposent des mécanismes d'adaptation au sein du contrôleur (Kondo et al., 1999; Eggenberger et al., 2000; Kondo, 2007).

5.2.4 Evaluation de la capacité de généralisation

Le panel d'articles en robotique évolutionniste du tableau 5.1 met en évidence que, même si de nombreux travaux se sont intéressés à la façon d'optimiser des contrôleurs robustes, relativement peu évaluent la capacité de généralisation des contrôleurs pendant l'optimisation. Et dans le cas où la capacité de généralisation

est effectivement évaluée, les contrôleurs sont souvent évalués dans un petit nombre de contextes d'évaluation (moins de 10, cf. table 5.1). Par exemple, dans (Mouret et Doncieux, 2012), des contrôleurs sont optimisés pour une tâche de collecte de balles dans une arène en évaluant chaque individu avec 3 positions initiales fixes. De même dans (Ziemke et Thieme, 2002), des contrôleurs ont été optimisés pour une tâche de navigation dans 6 contextes d'évaluation avec différentes formes de labyrinthe et différentes positions initiales du robot. Dans ces travaux, on cherche des contrôleurs performants sur l'ensemble des contextes d'évaluation, mais les capacités de généralisation des solutions obtenues ne sont pas estimées dans de nouveaux contextes après l'optimisation.

Certains travaux combinent à la fois l'intégration de mécanismes d'adaptation au niveau du contrôleur ainsi que l'utilisation de plusieurs de contextes d'évaluation. C'est le cas pour l'approche de Kondo et al. (1999) basée sur des réseaux de neurones à réarrangement dynamique utilisés pour contrôler un robot mobile Khepera et dont les poids synaptiques peuvent varier en fonction de l'environnement. Les contrôleurs sont optimisés dans 10 contextes d'évaluation avec différentes orientations initiales du robot. Les meilleures solutions obtiennent de bonnes performances quand ils sont testés dans de nouveaux contextes où l'environnement et la dynamique du robot sont modifiés.

D'autres méthodes consistent à utiliser un ensemble de contextes d'évaluation variable pendant l'optimisation. On peut citer notamment le travail de Jakobi (1997, 1998) dont le but initial est de trouver des contrôleurs robustes afin qu'ils se transfèrent efficacement d'un simulateur à un robot physique et soient également robustes sur le robot. Par exemple, dans une tâche de navigation d'un robot mobile Khepera dans un labyrinthe en T (Jakobi, 1997), les contrôleurs sont évalués indépendamment dans 10 simulateurs aux paramètres variables : chaque valeur de paramètre est générée aléatoirement dans un ensemble de valeurs donné. Une alternative est de générer l'ensemble d'évaluation par coévolution (Berlanga et al., 2002), où l'ensemble d'évaluation Ω_{eval} est un sous-ensemble généré à partir d'une liste de tous les contextes possibles. Le meilleur individu trouvé pendant l'optimisation est ensuite testé sur l'ensemble des contextes possibles. Bien qu'il ait une estimation de la capacité de généralisation, on remarquera que l'ensemble de test utilisé est identique à l'ensemble d'évaluation : les capacités de généralisation des solutions sont testées sur des contextes potentiellement utilisés pendant l'évaluation. L'estimation est donc biaisée, alors que les ensembles Ω_{eval} et Ω_{test} devraient être indépendants pour avoir une meilleure idée des capacités de généralisation des contrôleurs obtenus.

Le travail de Barate et Manzanera (2008) se rapproche plus de la formalisation classique de l'apprentissage supervisé. Le but est d'optimiser des contrôleurs évitant des obstacles dans une arène à l'aide de capteurs visuels. Les individus sont évalués sur 4 contextes où la position initiale du robot varie (Ω_{eval}). Les solutions trouvées à la dernière génération sont ensuite évaluées sur 4 nouveaux contextes avec des configurations différentes d'obstacles (Ω_{test}) et la meilleure solution est celle qui obtient les meilleures performances sur ces nouveaux contextes, autrement dit celle qui généralise le mieux. Comme les ensembles d'évaluation et de test sont disjoints,

ce protocole est plus adapté pour effectivement estimer la capacité de généralisation des contrôleurs obtenus. Le désavantage principal d'une telle méthode est que tous les individus sont évalués sur Ω_{eval}, ce qui limite la taille de l'ensemble d'évaluations utilisables sous peine de ralentir drastiquement l'optimisation.

5.2.5 Réduction du nombre d'évaluations

Afin de vérifier la contrainte C1 évoquée précédemment en section 5.2.2, une méthode directe revient à optimiser les solutions directement sur Ω_{eval}. Néanmoins, ceci contraint la taille de l'ensemble d'évaluations, car chaque solution doit être évaluée sur chaque contexte de Ω_{eval}. Ainsi, si l'utilisateur souhaite définir un ensemble d'évaluations de grande taille, l'optimisation est souvent non compatible avec des temps de calcul raisonnables.

De nombreuses méthodes ont été proposées afin de diminuer le nombre d'évaluations d'un algorithme évolutionniste, notamment les procédures de *racing* (Birattari et al., 2002; Heidrich-Meisner et Igel, 2009) et de *early stopping* (Bongard et Hornby, 2010; Bongard, 2011a). La procédure de *racing* (Birattari et al., 2002; Heidrich-Meisner et Igel, 2009), suivant la terminologie introduite par Maron et Moore (1994), désigne une méthode qui trouve une bonne configuration (modèle) à partir d'un ensemble d'alternatives à travers une séquence d'étapes. Lorsque le processus d'optimisation est lancé, s'il y a suffisemment d'éléments qui permettent de déterminer que certaines solutions sont inférieures à au moins une autre, une telle solution est enlevée de l'ensemble et la procédure est répétée jusqu'à ce qu'il n'en reste plus qu'une. L'élimination des solutions inférieures accélère la procédure et permet une évaluation plus fiable sur les évaluations prometteuses. La technique de *early stopping* (Bongard et Hornby, 2010; Bongard, 2011b) est similaire dans le sens où l'évaluation d'une solution est stoppée si continuer l'évaluation ne lui permet pas de ne pas être dominée par une autre solution. Cependant ces différentes méthodes ont seulement été utilisées sur des problèmes simples de renforcement comme la tâche de *mountain car* (Heidrich-Meisner et Igel, 2009). Cette tâche (Sutton et Barto, 1998) consiste à conduire un véhicule sous-puissant en haut d'une route de montagne abrupte, la gravité étant plus forte que le moteur de la voiture. Une deuxième colline se trouve derrière le véhicule qui peut donc s'aider de celle-ci pour prendre de la vitesse.

Une méthode alternative classiquement utilisée pour l'optimisation de fonction de fitness coûteuse en temps de calcul est de construire un modèle de substitution (*surrogate model*) de cette fonction : les solutions sont alors évaluées pour une valeur approchée rapidement calculable avec le modèle et non pas une valeur exacte (Ratle, 1998; Hemker et Sakamoto, 2006; Hemker et al., 2009). Ce type d'approche peut être utilisé pour construire une approximation de la performance des contrôleurs sur l'ensemble Ω_{eval}. Voir (Jin, 2005a) pour une revue sur ce sujet.

5.2.6 Synthèse

Ce bref état de l'art sur la capacité de généralisation nous a permis de mettre en exergue différents points :

- Peu de travaux en ER ont formalisé et évalué une capacité de généralisation ;
- il est nécessaire d'évaluer un individu dans différents contextes, ceci afin de se prémunir des risques de surapprentissage ;
- une évaluation de la capacité de généralisation est très coûteuse computationnellement : les algorithmes évolutionnistes nécessitant un grand nombre d'évaluations, il est difficilement possible d'évaluer, dans un temps raisonnable, un individu dans un grand nombre de contextes ;
- il existe une méthode permettant d'optimiser une fitness coûteuse en temps de calcul : en construisant un modèle de substitution (*surrogate model*) de cette fonction.

Cette synthèse a servi de base de travail à la méthode *ProGAb* décrite dans la section suivante.

5.2.7 Description de la méthode ProGab

L'approche que nous proposons se base sur une méthodologie inspirée de celle de l'apprentissage supervisée. De la même manière, trois ensembles de contextes - d'apprentissage, de validation et de test - vont être définis. Néanmoins, contrairement aux travaux d'apprentissage supervisé qui suggèrent d'utiliser un ensemble d'apprentissage de plus grande taille que l'ensemble de validation (Drucker et al., 1997; Kohavi, 1995; Alpaydin, 2004), nous imposons ici que l'ensemble d'apprentissage soit de petite taille afin de limiter le nombre d'évaluations effectuées au cours du processus d'optimisation.

Chercher à améliorer les capacités de généralisation des contrôleurs met en lumière le conflit suivant : (1) alors que la valeur de performance sur un petit ensemble d'apprentissage Ω_{train} est rapide à calculer, elle peut guider l'optimisation vers des solutions qui sur-apprennent et donc avec de faibles capacités de généralisation ; (2) en revanche, même si évaluer les solutions sur un ensemble de validation Ω_{valid} de grande taille est plus informatif, ce processus est très coûteux en temps de calcul.

Notre méthode est inspirée de l'approche par transférabilité proposée par Koos et al. (2012). Partant du constat qu'être performant sur l'ensemble d'apprentissage Ω_{train} ne garantit pas d'être performant sur l'ensemble de validation Ω_{valid}, l'approche *ProGAb* ("Promoting the Generalisation Ability") propose d'optimiser les solutions via un algorithme évolutionniste multi-objectif où chaque contrôleur est évalué par les objectifs suivants : (1) la performance F sur l'ensemble d'apprentissage Ω_{train} ; (2) la capacité de généralisation $\hat{G}$ calculée sur l'ensemble de validation Ω_{valid} ; (3) un objectif de diversité comportementale pour explorer plus efficacement l'espace de recherche (Trujillo et al., 2008; Gomez, 2009; Mouret et Doncieux, 2012) (cf. partie 4).

De plus, comme évaluer chaque contrôleur sur l'ensemble Ω_{valid} serait trop coû-

teux en temps de calcul, nous proposons que la capacité de généralisation sur Ω_{valid} soit approximée à l'aide d'un modèle de substitution construit et mis à jour pendant le processus d'optimisation en évaluant seulement quelques contrôleurs sur l'ensemble Ω_{valid}.

La méthode ProGAb peut être interprétée comme une technique de multi-objectivisation (voir section 2.1.4). Le but de l'optimisation est en effet de maximiser la performance sur l'ensemble d'évaluation ($\Omega_{eval} = \Omega_{train} \cup \Omega_{valid}$), et la méthode introduit deux objectifs séparés : la performance d'apprentissage sur Ω_{train} et la performance de validation sur Ω_{valid}.

5.2.7.1 Modèle de substitution de la capacité de généralisation

Les modèles de substitution sont classiquement utilisés dans des problèmes d'ingénierie où évaluer une solution est trop long/coûteux : une approximation de la fonction de performance est alors construite pendant l'optimisation et, au lieu d'évaluer les solutions directement sur le système physique testé, elles sont optimisées suivant ce modèle approché de la performance. Nous sommes typiquement dans un cas similaire, le calcul de la capacité de généralisation G^* de chaque contrôleur sur l'ensemble de validation Ω_{valid} implique un très grand nombre d'évaluations. Ainsi, nous allons optimiser une approximation $\hat{G}$ au lieu d'utiliser directement G^*. Utiliser un modèle de substitution revient à définir une structure de modèle et une stratégie de mise à jour pour améliorer l'approximation au cours de l'optimisation. Dans cette étude, nous avons choisi une structure simple d'interpolation par Inverse Distance Weighting (IDW) (Shepard, 1968). Il faut également décider sur quel espace construire le modèle et comme les contrôleurs ne sont tous évalués que sur l'ensemble d'apprentissage, le choix de l'espace d'entrée est limité. Le modèle peut être par exemple construit sur le génotype des solutions ou sur une description de leurs comportements observés sur Ω_{train}. En nous basant sur divers résultats concluant à l'intérêt d'utiliser des descriptions comportementales en robotique évolutionniste (section 4.2), nous avons choisi de construire le modèle de G^* sur le comportement des solutions sur les contextes d'apprentissage de Ω_{train}.

Si on appelle $b(c)$ le comportement correspondant au contrôleur c sur les contextes de Ω_{train}, la capacité de généralisation du contrôleur c est approximée par la valeur $\hat{G}(b(c))$. En supposant qu'une distance comportementale entre individus b_{dist} est définie et que les capacités de généralisation de quelques contrôleurs ont déjà été évaluées sur l'ensemble de validation (au moins 1), il est possible d'interpoler par IDW un modèle de substitution de la capacité de généralisation à partir de ces valeurs.

Soit $\mathcal{C}$ l'ensemble de tous les contrôleurs possibles, soit $\mathcal{C}_{valid}$ l'ensemble des contrôleurs ayant déjà été évalués sur l'ensemble de validation Ω_{valid} et $G^*(c_i)$ la capacité de généralisation exacte correspondant à chaque $c_i \in \mathcal{C}_{valid}$. La valeur prédite par le modèle de substitution $\hat{G}(b(c))$ de la capacité de généralisation pour un contrôleur quelconque c est :

$$\forall c \in \mathcal{C}, \hat{G}(b(c)) = \frac{\sum\limits_{c_i \in \mathcal{C}_{valid}} G^*(c_i)\ b_{dist}(c_i, c)^{-2}}{\sum\limits_{c_i \in \mathcal{C}_{valid}} b_{dist}(c_i, c)^{-2}}.$$

La stratégie de mise à jour du modèle vise à sélectionner les prochains contrôleurs à évaluer sur Ω_{valid} pour améliorer le modèle $\hat{G}$. Nous définissons une stratégie simple qui, à chaque génération, sélectionne l'individu, parmi ceux ayant les plus hautes valeurs de fitness sur l'ensemble d'apprentissage, dont la distance aux contrôleurs déjà évalués de $\mathcal{C}_{valid}$ est maximale[3]. Ceci permet de sélectionner des contrôleurs à la fois prometteurs d'après leur performance sur Ω_{train} et différents de ceux déjà évalués sur Ω_{valid}.

5.2.7.2 Critères d'évaluation

Une fois que le modèle de substitution est défini, chaque contrôleur est évalué par 3 critères à maximiser :

1. la performance F sur l'ensemble d'apprentissage Ω_{train} ;
2. la capacité de généralisation approchée sur l'ensemble de validation Ω_{valid} d'après le modèle de substitution $\hat{G}$;
3. le critère de diversité comportementale.

Ce troisième critère est utilisé afin de maintenir une diversité comportementale au sein de la population (pour plus de détails, voir chapitre 4).

5.2.7.3 Etapes de l'algorithme

Afin d'initialiser le modèle de substitution $\hat{G}$ de la capacité de généralisation, nous supposons qu'un contrôleur c_0 a déjà été évalué sur l'ensemble de validation avant le début de l'optimisation. A chaque génération, l'algorithme suivant est utilisé comme représenté sur la figure 5.1.

A. Evaluation du contrôleur c :
   - A1. calcul de son comportement $b(c)$ et de sa performance $F(c)$ sur l'ensemble d'apprentissage Ω_{train} ;
   - A2. évaluation des 2 autres critères par rapport aux contrôleurs déjà évalués de $\mathcal{C}_{valid}$ sur l'ensemble de validation Ω_{valid} (capacité de généralisation approchée $\hat{G}(b(c))$ sur Ω_{valid} et valeur de diversité comportementale) ;

B. Un contrôleur sélectionné par la stratégie de mise à jour est évalué sur l'ensemble de validation Ω_{valid} et le modèle de substitution est modifié en fonction de sa valeur exacte de capacité de généralisation.

C. Application des opérateurs évolutionnistes et génération de la nouvelle population.

3. Les deux applications considérées dans ce chapitre utilisent des valeurs de fitness discrètes. On sélectionnera donc l'individu à transférer parmi ceux ayant *la* plus haute valeur de fitness.

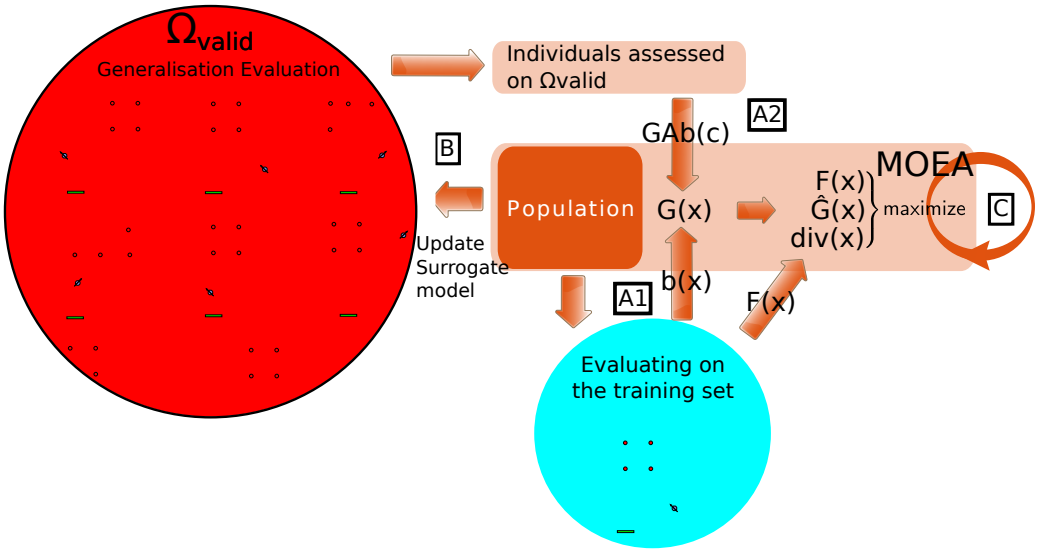


FIGURE 5.1 – Principe de l'approche *ProGAb* – A1. Pour chaque contrôleur c de la population, son comportement $b(c)$ et sa fitness $F(c)$ sont évalués sur l'ensemble d'apprentissage Ω_{train}. A2. Le comportement d'un contrôleur donné permet de calculer la valeur de sa capacité de généralisation prédite par le modèle de substitution $\hat{G}$, ainsi que sa valeur de diversité. B. Suivant la stratégie de mise à jour du modèle, un individu c_s de la population est sélectionné puis évalué sur l'ensemble de validation Ω_{valid}. Ce contrôleur est ajouté à l'ensemble $\mathcal{C}_{valid}$ des contrôleurs déjà testés et sa valeur exacte $G^*(c_s)$ de capacité de généralisation sur Ω_{valid} permet de mettre à jour le modèle de substitution $\hat{G}$ par interpolation. C. Les opérateurs évolutionnistes sont ensuite appliqués aux contrôleurs de la population et un processus de sélection permet de construire la population suivante.

Une fois le nombre de générations prévu écoulé, le meilleur contrôleur trouvé par l'optimisation est celui dont la valeur de capacité de généralisation est la plus élevée.

5.2.8 Expériences et résultats

L'approche *ProGAb* a été testée sur deux expériences :

- une tâche de ramassage de balles dans une arène (Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012), où le robot doit effectuer une série de sous-tâches (naviguer, trouver une balle, ramasser la balle, atteindre la corbeille, relâcher la balle, ...) pour obtenir une fitness non nulle ;
- le dispositif expérimental simple de navigation dans un labyrinthe en T pour un robot mobile simulé utilisé dans les chapitres précédents (voir 3.4 pour plus de détails) où la capacité de mémoire sera testée.

Nous allons maintenant présenter les différentes approches de référence auxquelles nous nous proposons de comparer l'approche *ProGAb*, avant de détailler pour chacune de ces deux applications son dispositif expérimental et les résultats obtenus. Dans chacune des applications étudiées, nous n'utilisons qu'*un unique contexte d'apprentissage* ($|\Omega_{train}| = 1$) pour l'approche *ProGAb*. Ceci signifie notamment que le modèle de substitution de la capacité de généralisation ne prend en entrée que les comportements des solutions sur ce contexte d'apprentissage spécifique.

L'objectif de ces deux expériences est de vérifier l'efficacité de la méthode *ProGAb* au niveau sur la capacité de généralisation des individus obtenus.

5.2.8.1 Protocole expérimental

Approches de référence. La méthode *ProGAb* est comparée à 4 approches de référence sur deux applications robotiques en simulation, détaillées ici de la plus simple à la plus sophistiquée. Toutes les approches partagent les mêmes ensembles de validation et de test. Les approches de référence diffèrent entre elles par l'ensemble d'apprentissage utilisé.

L'approche *Train_only* est basée sur un ensemble d'apprentissage ne contenant qu'un unique contexte $|\Omega_{train}| = 1$. Le contexte d'apprentissage choisi est le même que celui utilisé pour la méthode *ProGAb*. Les résultats de cette approche permettront notamment de déterminer si optimiser sur ce contexte précis permet de généraliser à d'autres contextes.

L'approche *Eval_all* optimise directement la fitness de chaque contrôleur sur l'ensemble de tous les contextes d'évaluation $\Omega_{valid} \subset \Omega_{train}$. En contrepartie du nombre important d'évaluations faites par individu, l'optimisation comptera moins de générations que pour les autres approches.

Les deux approches suivantes sont inspirées de l'article Panait (2003) : *Fixed-Random-Initial* (*FRI*) et *Random-Per-Generation* (*RPG*). Pour l'approche *FRI*, l'ensemble d'apprentissage contient n contextes ($|\Omega_{train}| > 1$) qui sont choisis aléatoirement dans l'ensemble de validation Ω_{valid} au début de l'optimisation. L'ensemble d'apprentissage reste ensuite fixe.

Pour la dernière approche appelée *RPG*, l'ensemble d'apprentissage contient également n contextes et il est généré aléatoirement à partir de Ω_{valid} à chaque génération. Cette approche est similaire à la méthodologie de Jakobi (1998) à ceci près que les contrôleurs sont toujours évalués sur un même ensemble d'apprentissage pendant une génération donnée.

Pour chacune de ces approches, les contrôleurs sont évalués par deux objectifs à maximiser : leur fitness moyenne sur les contextes d'apprentissage Ω_{train} et un objectif de diversité. Pour ces différentes expériences de contrôle, l'utilisation d'un critère de diversité suppose qu'une distance comportementale a été définie pour comparer les individus. La valeur de diversité est alors simplement calculée comme la moyenne des distances entre l'individu considéré et le reste de la population. Ainsi, soit $\mathcal{P}$ la population de contrôleurs de taille $|\mathcal{P}|$, soit c le contrôleur évalué et soit b_{dist} la distance comportementale définie entre individus, la valeur de diversité

$div(c)$ associée à c s'obtient comme suit :

$$div(c) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} b_{dist}(c,\ p)$$

Enfin, bien que ces différentes approches mettent en jeu des ensembles d'apprentissage de tailles différentes, nous avons fait en sorte de conserver un budget d'évaluations constant en jouant sur le nombre de générations de l'optimisation. Pour les 4 approches de référence, tous les contrôleurs de la population P étant évalués sur l'ensemble Ω_{train} pendant n_g, le nombre total d'évaluations E_{ctrl} pendant l'optimisation est simplement :

$$E_{ctrl} = |\mathcal{P}| * n_g * |\Omega_{train}| \tag{5.1}$$

Dans le cas de l'approche *ProGAb*, un contrôleur de la population est également évalué sur l'ensemble de validation Ω_{valid} à chaque génération. Dans ce cas, le nombre total d'évaluations E_{ProGAb} se calcule comme suit :

$$E_{ProGAb} = |\mathcal{P}| * n_g * |\Omega_{train}| + n_g * |\Omega_{valid}| \tag{5.2}$$

Les valeurs de paramètres utilisées, ainsi que le budget d'évaluations choisi pour chacune des tâches, sont répertoriées dans les tables 5.6 et 5.3.

Contrôleurs optimisés. Le contrôle du robot dans les deux applications est assuré par un réseau de neurones. Nous utilisons le codage *DNN* décrit précédemment où à la fois la structure et les paramètres des réseaux sont optimisés. Nous reprenons les paramètres décrit à la section 3.2.2.

Distance comportementale. Dans les deux applications, le comportement du robot est décrit par la séquence temporelle binarisée de ses valeurs de capteurs et d'effecteurs. Pour l'obtenir, les valeurs de chaque capteur (dans l'intervalle $[0, 1]$) sont transformées en 0 quand elles sont inférieures à 0.5 et en 1 sinon. Les effecteurs pouvant prendre des valeurs négatives, le seuillage se fait autour de 0. La distance comportementale entre individus b_{dist} est alors définie comme la distance de Hamming entre deux comportements. Cette distance générique renvoie le nombre de bits qui diffèrent entre les séquences temporelles. Des résultats intéressants ont été obtenus avec cette description comportementale sur la tâche de collecte de balles, où l'utilisation de cette distance comportementale comme critère de diversité permet d'obtenir les contrôleurs les plus efficaces (Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012). Pour toutes les approches, le calcul de distance comportementale se fait sur le comportement global observé sur l'ensemble d'apprentissage.

Paramètres Les cinq approches utilisent l'algorithme évolutionniste multi-objectif NSGA-II (Deb et al., 2002), qui repose sur une sélection élitiste par tournoi et une méthode de classement des individus basée sur la relation de dominance de

Pareto (cf. section 2.1.4.3 pour plus de détails). Ce travail a été implémenté dans le framework Sferes$_{v2}$ (Mouret et Doncieux, 2009a). Le code source est disponible sur `http://www.isir.fr/evorob_db`.

Chacune des approches est répétée 20 fois avec un budget total d'environ 1.2 millions d'évaluations pour la tâche de ramassage de balles et d'environ 3.8 millions d'évaluations sur la tâche de panneau de signalisation. Les budgets exacts, ainsi que les différentes valeurs de paramètres utilisées sont indiqués dans la table 5.3 et 5.6 pour respectivement l'expérience 1 et 2. A noter que pour les approches *FRI* et *RPG*, l'ensemble d'évaluation compte 4 contextes sur la tâche ramassage de balles et 10 sur la tâche panneau de signalisation. Les paramètres liés à l'encodage des réseaux de neurones utilisés pour le contrôle sont répertoriés dans la table 5.4 et 5.7.

5.2.8.2 Experience 1 : ramassage de balle

La tâche de collecte de balles implémentée ici (cf. figure 5.2) est principalement basée sur le dispositif expérimental décrit dans (Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012). Un robot doit naviguer dans une arène, trouver quatre balles, et les amener une par une dans une corbeille. Pour résoudre cette tâche, le robot doit pouvoir naviguer dans l'arène et modifier son comportement quand il détecte une balle. Plus précisément, on peut distinguer plusieurs sous-tâches à résoudre : naviguer pour trouver une balle, récupérer cette balle, naviguer pour trouver la corbeille, poser la balle dans la corbeille, naviguer pour trouver une deuxième balle, ... La fonction de fitness utilisée ne récompense pas le contrôleur évalué pour ces différentes étapes : la valeur de la fitness correspond au nombre de balles dans la corbeille à la fin de la simulation. Une telle fonction de fitness rend la tâche d'autant plus complexe à résoudre par un processus d'optimisation.

L'agent considéré est un robot à deux roues possédant 10 capteurs : 3 capteurs de distance situés à l'avant, 2 capteurs de contact binaires, 2 capteurs pour détecter les balles [4], 2 capteurs par tranche pour détecter la corbeille[4] et un capteur binaire activé seulement si le robot est en train de porter une balle. Le robot possède également 3 effecteurs : les deux roues et un moteur de récupération de balle. La valeur associée à ce dernier moteur doit être supérieure à 0.5 pour ramasser une balle et la conserver, et inférieure à 0.5 pour la relâcher. Si le robot détient déjà une balle, il ne peut pas en ramasser d'autres.

La fonction de fitness compte le nombre de balles dans la corbeille à la fin des 3000 pas de temps de l'expérience. La totalité des expériences s'effectue dans un simulateur 2D sans friction ni glissement.

Les paramètres définissant un contexte ω pour cette application sont les suivants : 1) la taille de l'arène ; 2) l'orientation initiale du robot ; 3) les coordonnées initiales du robot (abscisse et ordonnée) ; 4) le positionnement initial des 4 balles. Il y a 6 positions de balles possibles comme indiqué sur le schéma 5.2, soit 15 posi-

4. L'un des capteurs détecte les éléments situés à gauche du robot, l'autre détecte les éléments situés à droite. Cf. figure 5.2 pour une illustration du champ de détection.

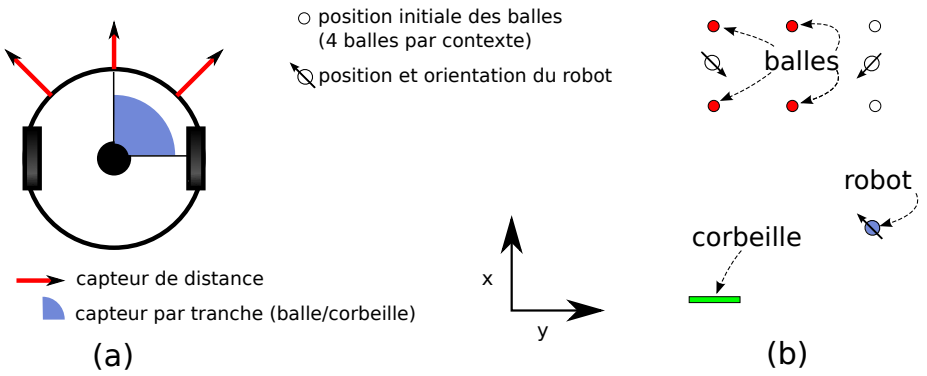


FIGURE 5.2 – (a) Présentation de la tâche de collecte de balles. En plus de ses 3 capteurs de distance, le robot est équipé de 2 capteurs détectant les balles et de 2 capteurs détectant la corbeille. Le champ de détection des capteurs droits est indiqué sur la figure. Celui des capteurs gauches est symétrique. Un réseau de neurones contrôle les vitesses des deux roues motorisées, ainsi qu'un moteur de ramassage de balles. (b) Arène où le robot évolue. Les 6 positions possibles des balles et quelques positions et orientations initiales du robot sont schématisées.

tionnements possibles. Les valeurs des différents paramètres sont répertoriées dans la table 5.2.

5.2.8.3 Résultats

L'ensemble des résultats obtenus avec les 5 approches est indiqué sur le graphe 5.3. Pour chaque méthode sont indiquées les performances des meilleurs individus trouvés (20 répétitions) sur les ensembles d'apprentissage, de validation et de test. La figure 5.4 indique l'évolution de la capacité de généralisation de ces contrôleurs sur l'ensemble Ω_{valid} au cours de l'optimisation. Toutes les comparaisons entre les approches sont effectuées à l'aide de tests de somme de rangs de Wilcoxon (ou test de Mann-Whitney).

L'approche *Train_only* ne trouve pas toujours des contrôleurs optimaux sur le contexte d'apprentissage et les 4 balles sont collectées dans seulement 13 des 20 optimisations. Ce résultat est cohérent avec d'autres travaux effectués sur ce dispositif expérimental (Doncieux et Mouret, 2010b; Mouret et Doncieux, 2012).

Les approches *Train_only* et *Eval_all* correspondent à des capacités de généralisation faibles, avec respectivement 0.63 et 1.00 balles collectées en moyenne sur l'ensemble des contextes de validation (sur 4 balles). Globalement, les performances des approches *Eval_all*, *FRI* et *RPG* ne sont pas très significativement différentes (p-values > 0.07), avec respectivement 1.00, 1.70 et 2.03 balles collectées en moyenne sur l'ensemble des contextes de Ω_{valid}.

Seule l'approche *ProGAb* obtient des résultats significativement meilleurs avec des contrôleurs pouvant ramasser en moyenne 2.63 sur les contextes de validation (p-values < 0.018). Il s'avère que la performance sur Ω_{test} est également bonne

TABLE 5.2 – Paramètres des ensembles de contextes pour l'application ramassage de balles. Ω_{train} correspond au contexte d'apprentissage utilisé dans les approches *ProGAb* et *Train_only*. Les coordonnées initiales sont exprimées en pourcentage de la taille de l'arène. Se reporter à la figure 5.2 pour l'orientation.

Paramètres	Ω_{train}	$\lvert\Omega_{valid}\rvert = 30$	$\lvert\Omega_{test}\rvert = 600$
taille de l'arène (mm)	600	600	650
orientation initiale (°)	0	0	[-180, -90, 45, 135]
abscisse initiale	0.75	[0.25, 0.75]	[0.25, 0.75]
ordonnée initiale	0.75	0.75	[0.25, 0.40, 0.50, 0.60, 0.75]
nb. de config. des balles	1	15	15

TABLE 5.3 – Paramètres utilisés dans l'application ramassage de balles ($\lvert\Omega_{valid}\rvert = 30$) afin d'assurer un nombre total d'évaluations de $1.2 \cdot 10^6$ par optimisation (cf. équations (5.1) et (5.2)).

Approches	$\lvert\mathcal{P}\rvert$	n_g	$\lvert\Omega_{train}\rvert$	nombre total d'évaluations
Train_only	200	6000	1	$1.20 \cdot 10^6$
Eval_all	”	200	30	$1.20 \cdot 10^6$
FRI	”	1500	4	$1.20 \cdot 10^6$
RPG	”	1500	4	$1.20 \cdot 10^6$
ProGAb	”	5500	1	$1.27 \cdot 10^6$

TABLE 5.4 – Paramètres liés à l'encodage des réseaux de neurones pour l'application II.

Paramètres	Valeurs
nb. min./max. de neurones	10 / 30
nb. min/max. de connexions	50 / 250
prob. d'ajouter un neurone	0.15
prob. d'enlever un neurone	0.05
prob. d'ajouter une connexion	0.05
prob. d'enlever une connexion	0.05
prob. de modifier un poids/biais	0.15

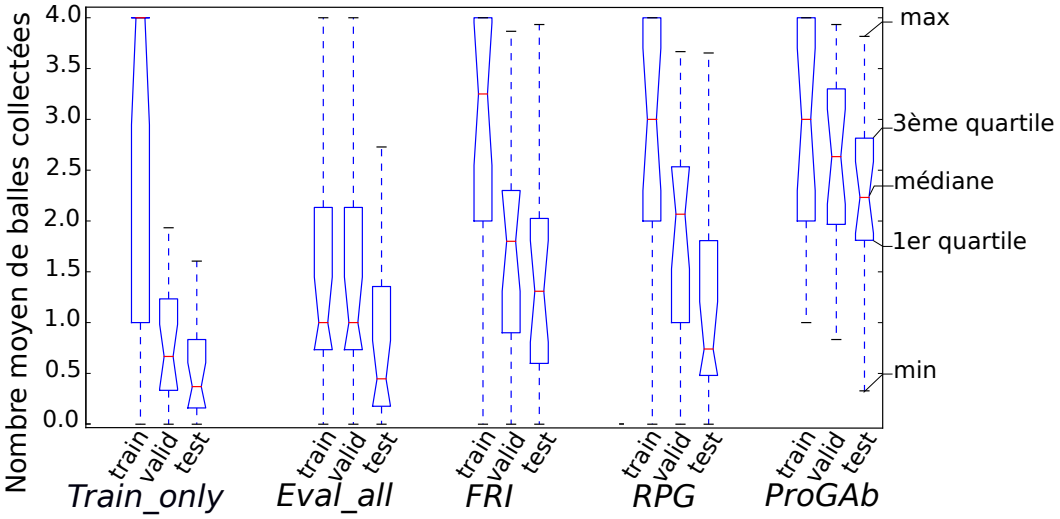


FIGURE 5.3 – Nombre moyen de balles collectées par contexte sur les ensembles Ω_{train}, Ω_{valid}, Ω_{test} par les meilleurs contrôleurs trouvés par chaque méthode sur la tâche de collecte de balle (20 répétitions par approche). La méthode *ProGAb* obtient les meilleures capacités de généralisation d'après un test de Mann-Whitney (p-values < 0.018)

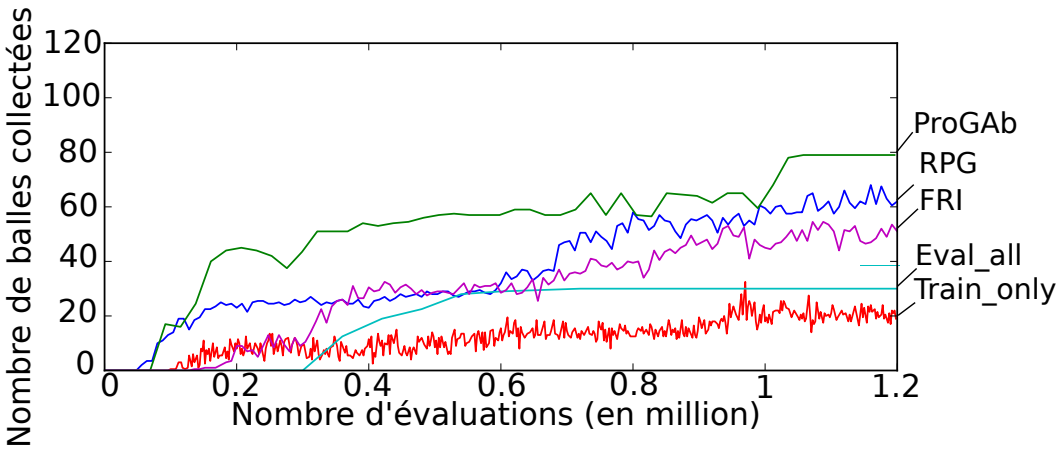


FIGURE 5.4 – Evolution de la capacité de généralisation sur l'ensemble Ω_{valid} des meilleurs contrôleurs obtenus avec chaque approche (valeur médiane sur les 20 répétitions). Pour obtenir ce graphe, les différents contrôleurs sont évalués sur Ω_{valid} toutes les 10^4 évaluations (hors optimisation). L'ordonnée représente le nombre total de balles collectées sur tous les contextes de Ω_{valid} : 30 contextes comptant chacun 4 balles, d'où 120 balles collectées au maximum.

TABLE 5.5 – Paramètres des ensembles de contextes pour l'application I. Ω_{train} correspond au contexte d'apprentissage utilisé dans les approches *ProGAb* et *Train_only*. La position initiale sans décalage correspond à un point situé au milieu du premier corridor du labyrinthe (coordonnées [50, 300]). Se reporter à la figure 3.13 pour l'orientation.

Paramètres	Ω_{train}	$\lvert\Omega_{valid}\rvert = 180$	$\lvert\Omega_{test}\rvert = 180$
taille du lab. (mm)	600	[500, 550, 600, 650]	[530, 580, 630, 640]
orientation init. (°)	0	[-30, -15, 0, 15, 30]	[-23, -18, 11, 18, 23]
décalage en x (mm)	0	[-20, 0, 20]	[-10, 10, 30]
décalage en y (mm)	0	[-40, 0, 40]	[-20, 0, 20]

TABLE 5.6 – Paramètres utilisés dans l'application panneau de signalisation ($\lvert\Omega_{valid}\rvert = 180$) afin d'assurer un nombre total d'évaluations d'environ $3.8 \cdot 10^6$ par optimisation (cf. équations (5.1) et (5.2)).

Approches	$\lvert\mathcal{P}\rvert$	n_g	$\lvert\Omega_{train}\rvert$	nombre total d'évaluations
Train_only	200	19000	1	$3.80 \cdot 10^6$
Eval_all	”	110	180	$3.96 \cdot 10^6$
FRI	”	1900	10	$3.80 \cdot 10^6$
RPG	”	1900	10	$3.80 \cdot 10^6$
ProGAb	”	10000	1	$3.80 \cdot 10^6$

avec une moyenne de balles collectées de 2.18 par contexte. La différence n'est pas significative (p-value = 0.107). En comparaison, les performances de *RPG*, la deuxième meilleure approche sur cette application, sont de 2.07 balles collectées en moyenne sur Ω_{valid} et de seulement 0.65 balles collectées en moyenne sur Ω_{test} (dégradation significative : p-value = $6.5 \cdot 10^{-3}$).

Pour cette application sur une tâche complexe de collecte de balles dans une arène, l'approche *ProGAb* semble également être une méthode performante pour améliorer les capacités de généralisation des contrôleurs.

5.2.8.4 Expérience 2 : "Panneau de signalisation"

Le protocole est décrit dans la section 3.4.3.

Afin de définir les ensembles d'apprentissage, de validation et de test, nous définissons 3 paramètres variables par contexte : 1) la taille du labyrinthe ; 2) l'orientation initiale du robot ; 3) les coordonnées initiales du robot (abscisse et ordonnée). Les différentes valeurs utilisées pour définir les ensembles Ω_{train}, Ω_{valid} et Ω_{test} sont répertoriées dans la table 5.5.

TABLE 5.7 – Paramètres liés à l'encodage des réseaux de neurones pour l'application panneau de signalisation.

Paramètres	Valeurs
nb. min./max. de neurones	3 / 20
nb. min/max. de connexions	5 / 50
prob. d'ajouter un neurone	0.05
prob. d'enlever un neurone	0.05
prob. d'ajouter une connexion	0.05
prob. d'enlever une connexion	0.05
prob. de modifier un poids/biais	0.05

5.2.8.5 Tests post-optimisation

Test de robustesse de la mémoire. Comme décrit dans le chapitre précédent, nous considérons qu'un réseau est doté d'une mémoire robuste si au moins un neurone interne respecte les points suivants :

- après la présentation des lettres, le neurone a une sortie différente pour les séquence de lettres AX des autres séquences de lettres (BX,BY,AY).
- la mémoire n'est pas affectée par la durée de présentation des lettres. Alors que durant l'évolution la durée de présentation était de 50 pas de temps pour chaque lettre, l'activité du réseau est testée — après le processus d'optimisation — avec une durée de 400 pas de temps. L'objectif est de détecter les réseaux qui sont basés sur une dynamique qui leur permettent d'avoir une activité différente après exactement 50 pas de temps, mais ne fonctionneront pas avec une durée différente.

Test de capacité de généralisation. Durant le processus d'optimisation, le robot est testé dans 12 contextes différents pour chaque séquence de lettre, et la fitness maximale est atteinte seulement si l'individu est capable de réussir la tâche sur tous les contextes. Après le processus d'optimisation, les meilleurs contrôleurs sont testés sur 180 contextes non vus précédemment. Les 180 nouveaux contextes incluent différentes tailles de cartes, de positions et d'orientations de départ du robot. Un contrôleur est considéré comme ayant une bonne capacité de généralisation si celui-ci réussit la tâche sur au moins 60 des nouveaux contextes.

5.2.8.6 Résultats

L'ensemble des résultats obtenus avec les 5 approches est indiqué sur le graphe 5.5. Pour chaque méthode sont indiquées les performances des meilleurs individus trouvés (20 répétitions) sur les ensembles d'apprentissage, de validation et de test. La figure 5.6 indique l'évolution de la capacité de généralisation de ces contrôleurs sur l'ensemble Ω_{valid} au cours de l'optimisation. Toutes les comparaisons

TABLE 5.8 – Nombre d'optimisations N où le meilleur contrôleur de la population résout plus de 150 des 180 contextes de l'ensemble de validation Ω_{valid}. Le nombre d'évaluations nécessaire pour obtenir un tel individu est indiqué : les valeurs de médianes, moyennes et écart-types sont calculés sur les N optimisations concernées. L'approche utilisée a un impact significatif sur le nombre N obtenu d'après un test exact de Fisher (p-value $= 2.63 \cdot 10^{-8}$).

Approche	Nb. d'optimisations (maximum 20)	Nombre d'évaluations (10^6)		
		médiane	moyenne	écart-type
Train_only	0	·	·	·
Eval_all	1	2.59	2.59	0
FRI	6	1.75	1.86	0.95
RPG	12	1.21	1.64	1.12
ProGAb	14	0.42	0.88	0.93

entre les approches sont effectuées à l'aide de tests de somme de rangs de Wilcoxon (ou test de Mann-Whitney).

Dans cette tâche, il est relativement simple de maximiser la fitness sur un seul contexte. Si on s'intéresse en particulier aux approches *Train_only* et *ProGAb*, la valeur maximale de fitness en médiane est effectivement rapidement atteinte sur l'unique contexte d'apprentissage, au bout de respectivement 55000 et 71000 évaluations. L'approche *Train_only* n'obtient pas de contrôleurs capables de généraliser sur l'ensemble de validation avec un nombre médian de contextes réussis sur Ω_{valid} de 3 sur 180. Ce résultat montre que même sur une tâche simple à résoudre dans un contexte particulier, atteindre la performance maximale sur ce contexte spécifique ne donne pas de garantie de performance sur d'autres contextes.

Sur cette application de navigation, il n'était pas nécessaire d'effectuer un si grand nombre d'évaluations : pour l'ensemble des approches, la capacité de généralisation des contrôleurs trouvés se stabilise au bout de 2 millions d'évaluations sans augmentation ultérieure significative (p-values > 0.35). Malgré cela, dans 18 cas sur 20, l'approche *Eval_all* n'est pas capable de trouver des contrôleurs capables de résoudre la tâche ne serait-ce que sur un seul contexte. Pour cette application, il serait peut-être nécessaire d'allouer un budget d'évaluations plus important. Avec le budget d'évaluation accordé, l'approche *Eval_all* ne dispose, en effet que de 110 générations.

Comparé à l'approche *Train_only* où un seul contexte d'évaluation est utilisé, les approches *FRI* et *RPG* évaluant chaque contrôleur sur 10 contextes différents se comportent significativement mieux (p-values $< 10^{-3}$), avec des nombres médians respectifs de contextes réussis de 80 et 144 sur 180. Par contre, générer l'ensemble d'évaluation aléatoirement à chaque génération comme le préconise l'approche *RPG* n'a pas d'impact déterminant par rapport à un ensemble de validation fixe utilisé avec *FRI* (p-value $= 0.148$ au bout de 3.8 millions d'évaluations).

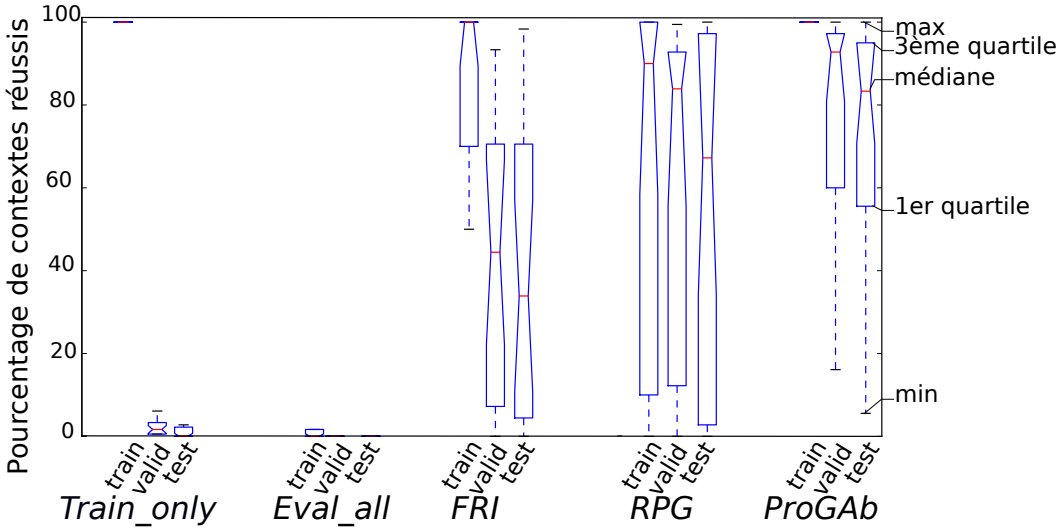


FIGURE 5.5 – Pourcentage de contextes des ensembles Ω_{train}, Ω_{valid} et Ω_{test} réussis par les meilleurs contrôleurs obtenus avec chaque approche pour la tâche de navigation dans un labyrinthe en T (20 répétitions par approche). La méthode *ProGAb* obtient les meilleures capacités de généralisation d'après un test de Welch (p-values < 0.039).

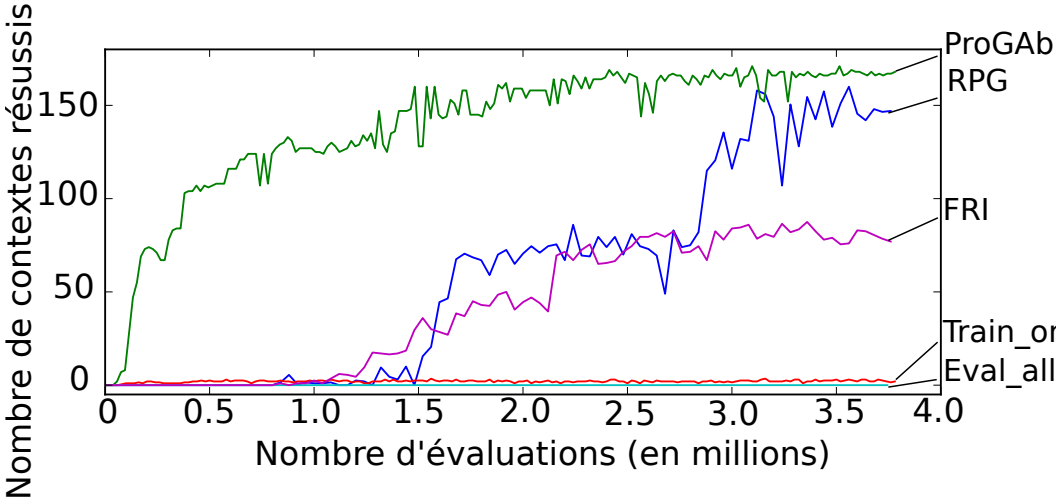


FIGURE 5.6 – Evolution de la capacité de généralisation sur l'ensemble Ω_{valid} des meilleurs contrôleurs obtenus avec chaque approche (valeur médiane sur les 20 répétitions) pour la tâche de navigation dans un labyrinthe en T. Pour obtenir ce graphe, les différents contrôleurs sont évalués sur Ω_{valid} toutes les 10^4 évaluations (hors optimisation). L'ordonnée représente le nombre de contextes réussis (i.e. où le contrôleur a obtenu la fitness maximale de 6) : 180 contextes au maximum.

L'approche *ProGAb* trouve les contrôleurs aux meilleures capacités de généralisation, avec un nombre médian de contextes réussis de 170 sur les 180 de l'ensemble de validation, ce qui est significativement meilleur que les 4 approches de référence (p-values $< 4 \cdot 10^{-2}$). D'autre part, si on se réfère à la figure 5.6, la capacité de généralisation sur Ω_{valid} augmente plus rapidement avec l'approche *ProGAb*. Pour étudier ce point plus en détail, nous avons sélectionné pour chaque approche les optimisations où le meilleur contrôleur résout plus de 150 contextes de validation (cf. table 5.8). Il s'avère que la méthode *ProGAb* trouve de tels individus deux à trois fois plus rapidement que la méthode *RPG*, avec des nombres d'évaluations médians respectifs de $0.42 \cdot 10^6$ et de $1.21 \cdot 10^6$. Les autres approches n'obtiennent d'aussi bonnes capacités de généralisation que dans moins de 30% des optimisations.

Un dernier point important à mettre en avant concerne la performance sur l'ensemble de test Ω_{test}. En effet, comme nous l'avons indiqué précédemment, même si maximiser la capacité de généralisation sur l'ensemble de validation est déterminant, les contextes de Ω_{valid} font partie des contextes utilisés pour l'évaluation (sauf pour l'approche *Train_only*). Il est nécessaire, pour prouver que les contrôleurs obtenus ont bien des capacités de généralisation intéressantes, de tester les contrôleurs obtenus sur un ensemble de test disjoint de Ω_{valid}. Il s'avère que l'approche *ProGAb* obtient également les meilleurs résultats sur Ω_{test}, avec un nombre médian de contextes réussis de 150 sur 180, c'est-à-dire sans dégradation significative par rapport aux performances sur l'ensemble de validation (p-value = 0.45).

L'approche *ProGAb* permet donc d'améliorer significativement les capacités de généralisation des contrôleurs trouvés sur cette application à la navigation d'un robot mobile simulé dans un labyrinthe en T, et plus rapidement que des approches classiques.

Influence sur la capacité de mémoire. La figure 5.7 affiche la proportion des meilleurs individus de chaque runs qui répondent aux critères suivant : la barre rouge correspond au pourcentage d'individus atteignant la fitness maximale ; la barre noire ceux ayant une mémoire robuste ; la barre bleue indiquant ceux dotés à la fois de la fitness maximale d'une mémoire robuste.

Nous pouvons constater qu'avec *S*, peu d'individus à la mémoire robuste se développent. Seulement 2 individus sur 30 ont été capables de faire émerger ce type de mémoire. L'ajout d'un objectif d'exploration comportementale, configurations *S+Div* et *S+Nov*, augmente significativement la proportion d'individus possédant ce type de mémoire, avec respectivement 25% et 45% d'individus dotés de mémoire robuste. Cependant un nombre important d'individus obtient la fitness maximale en développant une mémoire fragile. Avec *ProGAb* près de 60% des individus possèdent cette caractéristique. Cette pression de sélection favorisant une capacité de généralisation est plus efficace que les objectifs d'exploration (Diversité et Nouveauté). Cela renforce l'idée d'un lien entre capacité de généralisation et mémoire robuste.

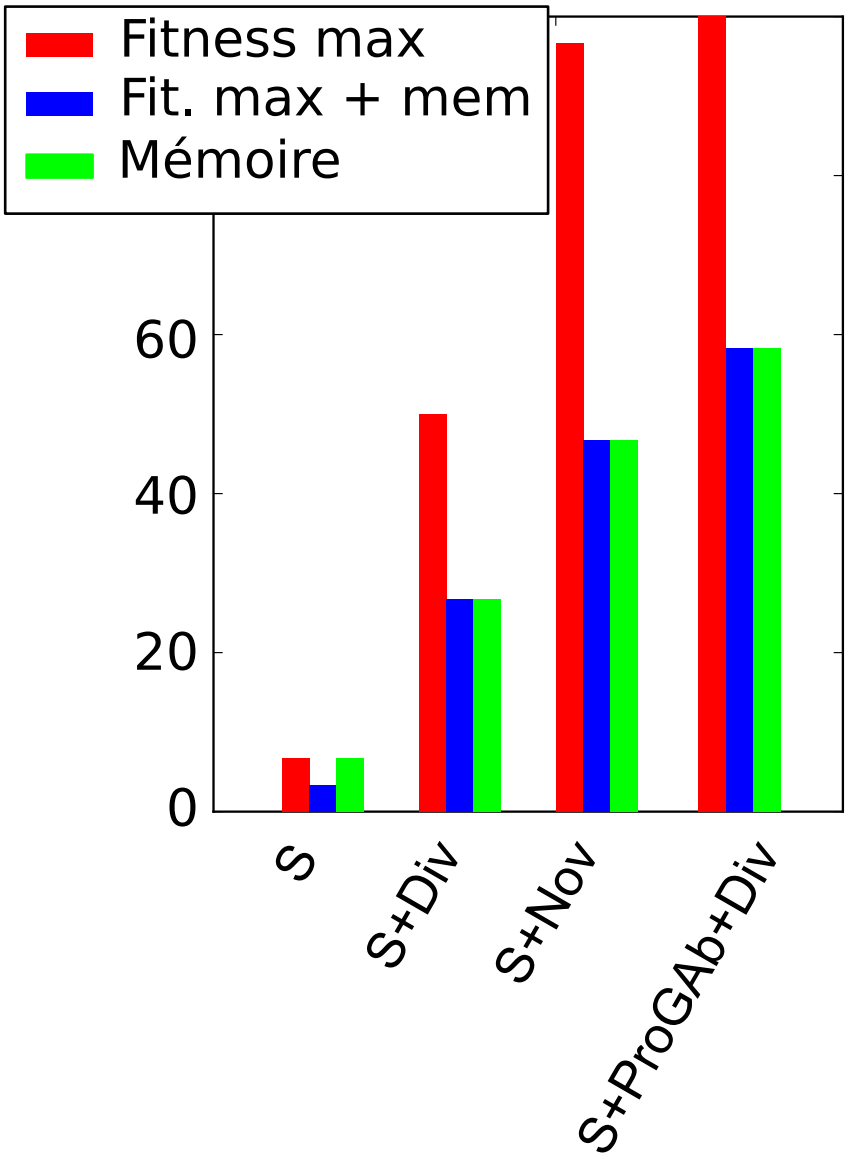


FIGURE 5.7 – Proportion des meilleurs individus de chaque runs qui répondent aux critères suivants : (1) atteint la fitness maximale ; (2) doté d'une mémoire robuste ; (3) atteint la fitness maximale et est doté d'une mémoire robuste.

5.3 Méthode *Coher*

Afin de survivre dans son environnement, un agent doit avoir un comportement qui n'est pas perturbé par le bruit ou n'importe quel autre distracteurs. Son comportement est supposé être relativement inchangé lorsqu'il est testé sur des situations similaires. Au lieu d'évaluer les individus dans différents environnements afin d'évaluer leur capacité de généralisation, nous proposons ici de formuler ce problème comme un problème de cohérence de comportement dans différents contextes. L'objectif est de proposer un objectif ayant pour but de récompenser explicitement la cohérence des comportements. Son principe est de définir différents ensembles de contextes et de comparer le comportement du système optimisé dans chacun d'eux. La fonction de fitness ainsi définie a pour but de récompenser les individus qui exhibent le comportement attendu. Même si le comportement exact à générer n'est pas connu, l'expérimentateur peut savoir dans quels contextes il s'attend à avoir un comportement similaire et dans quels contextes il s'attend à avoir des comportements différents. Par exemple, un agent qui se déplace dans son environnement devrait être peu sensible au bruit ou à d'autres distracteurs et se comporter de manière similaire dans des contextes proches.

L'approche *Coher* est basée sur l'évaluation d'une cohérence de comportements dans différents scénarios. Un scénario étant défini comme un sous-ensemble d'un contexte d'évaluation. Par exemple, dans le cas de notre tâche AX-CPT, nous avons un contexte d'évaluation qui est défini par : une taille d'environnement spécifique, une position de départ de l'agent, un angle de départ, quatre séquences de lettres évaluées. Dans ce cas, un scénario peut, par exemple, être défini pour chaque séquence de lettres. On aurait donc quatre scénarios (un pour AX, BY, BX, AY) pour chaque contexte.

5.3.1 Formalisation

La méthode *Coher* s'appuie sur la simulation d'un individu sur un ensemble S de comme montré en figure 5.8. La cohérence de l'individu est ensuite évaluée en comparant les comportements (sorties). Nous notons $o_i(t)$ la sortie simulée du scénario i après t pas de temps. En fonction du problème considéré, l'expérimentateur définit différents scénarios et différentes contraintes de cohérences entre les scénarios. Deux opérateurs de contraintes peuvent être définis de la manière suivante :

- sortie du scénario i est différent de la sortie du scénario j : $o_i(t) \neq o_j(t)$
- sortie du scénario i est exactement la même que la sortie du scénario j : $o_i(t) = o_j(t)$

Avec comme calcul pour les contraintes :

- $o_i(t) \neq o_j(t) : f_t = d(o_i(t), o_j(t))$
- $o_i(t) = o_j(t) : f_t = 1 - d(o_i(t), o_j(t))$

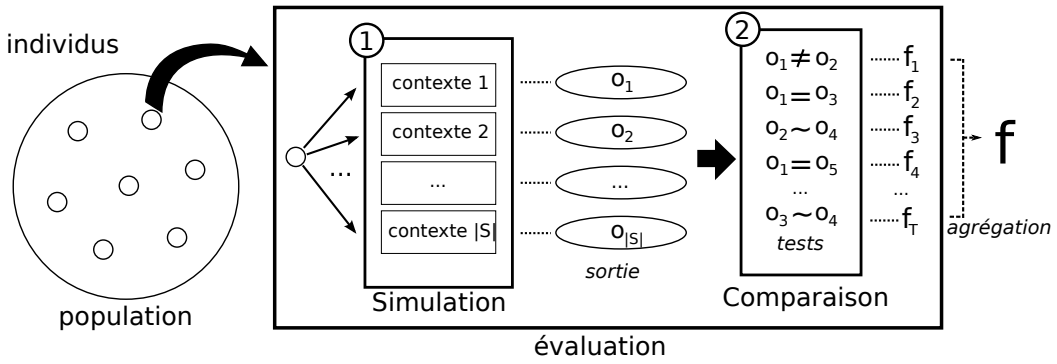


FIGURE 5.8 – Détail de l'évaluation d'un individu via l'objectif basé sur la cohérence. 1) Un individu (ici un réseau de neurones avec des neurones internes $n_1 n_2, ...$) est simulé dans différents scénarios prédéfinis. Durant cette simulation, le comportement de chaque neurone interne est stocké. 2) les comportements internes sont ensuite comparés entre les différents contextes et évalué en fonction des critères de cohérence attendus. Il en résulte une valeur de fitness partielle f_i. Ensuite les fitness partielles sont agrégées au sein de l'évaluation finale f.

La distance d est la différence entre les comportements de sortie. d est spécifique à chaque expérience. L'évaluation finale de la qualité d'un individu est ensuite calculée en agrégeant les différents termes :

$$f(x) = \frac{1}{T} \sum_{t}^{T} f_t \tag{5.3}$$

La méthode *Coher* peut se résumer en trois étapes :
- définir une collection de différents scénarios ;
- définir une contrainte entre les sorties des scénarios ;
- définir la manière de comparer les sorties avec une distance d.

Bien que la définition d'une collection de différents scénarios nécessite un minimum de connaissance de la tâche, le comportement exact n'a pas besoin d'être connu.

5.3.2 Expériences et résultats

5.3.2.1 Contexte expérimental

L'approche *Coher* a été testée sur le dispositif expérimental simple de navigation dans un labyrinthe en T pour un robot mobile simulé utilisé précédemment, tous les détails de l'expérience sont décrits à la section 3.4.3.

Cette approche sera comparée aux pressions de sélection évoquées précédemment :

– les objectifs d'explorations (diversité et nouveauté) ;
– l'approche *ProGAb*.

Description des scénarios Pour la tâche AX-CPT nous savons que l'agent doit exhiber deux comportements différents : l'un lors de la séquence AX, l'autre lors de la présentation des séquences BX, AY, BY. Comme précédemment, un individu est évalué dans 12 contextes différents.

Nous considérons donc les scénarios suivant :
– 12 scénarios correspondant à la séquence AX (un pour chaque contexte)
– 36 scénarios correspondant aux séquences BX, AY, BY. Cet ensemble est noté S.

Pour réaliser la tâche, l'agent doit posséder une représentation interne de la lettre A ou B. Cette représentation interne doit avoir un comportement similaire pour les entrées AY, BX ou BY, et différent pour l'entrée AX. Le comportement est calculé après la présentation des lettres, ce qui signifie que les neurones d'entrées des différentes lettres ne sont plus actifs. L'existence d'une différence entre les deux types de séquences devrait refléter l'émergence d'une forme de mémoire.

Pour chaque individu, le comportement (sortie) de chaque neurone interne est stocké. Un individu a N neurones internes — N peut varier d'un individu à l'autre.

L'objectif est de sélectionner les individus qui ont *au moins* un neurone interne qui représente l'information.

$o_s^i(t)$ est la sortie du ième neurone interne dans le scénario s au pas de temps t, après la présentation des lettres ($t > 150$). le but de l'objectif de cohérence est d'encourager les individus à suivre les règles suivantes :

$$\forall s \in S,\, o_{AX}^i(t) \neq o_s^i(t)$$

$$\forall s, s' \in S,\, o_s^i(t) = o_{s'}^i(t)$$

Pour chaque neurone interne i, deux fitness partielles f_1^i et f_2^i sont calculées. Elles mesurent de quelle manière les neurones internes respectent les deux règles précédentes :

$$f_1^i = \frac{1}{|S|} \sum_{s \in S} \frac{1}{T} \sum_{t}^{T} \frac{|o_{AX}^i(t) - o_s^i(t)|}{2}$$

$$f_2^i = 1 - \Big[\frac{1}{|S|^2 - |S|} \sum_{s2 \in S} \sum_{s \neq s2} \frac{1}{T} \sum_{t}^{T} \frac{|o_s^i(t) - o_{s2}^i(t)}{2} | \Big]$$

Ensuite la fitness de chaque neurone interne est calculée de la manière suivante :

$$f^i = f_1^i + f_2^i$$

Afin de sélectionner les individus qui ont *au moins* un neurone interne qui représente l'information. La fitness finale est calculée comme le maximum de toutes les fitness internes f^i :

$$f = \max_{0 \leq i < N} f^i$$

TABLE 5.9 – Résumé des différentes configurations utilisées

	Configuration	Description
1	**S**	Fitness discrète
2	**S+Div**	Fitness discrète + diversité
3	**S+Nov**	Fitness discrète + Nouveauté
4	**S+Coh**	Fitness discrète + obj. cohérence
5	**S+Coh+Div**	Fitness discrète + obj. cohérence + Diversité
6	**S+Coh+Nov**	Fitness discrète + obj. cohérence + Nouveauté
7	**S+ProGAb**	Fitness discrète + obj. ProGAb

La fitness f compare les quatre séquences de lettres évaluées sur le même contexte. L'objectif de cohérence correspond à la moyenne des 12 fitness ainsi définies (une pour chaque contexte).

5.3.2.2 Résumé des différentes configurations

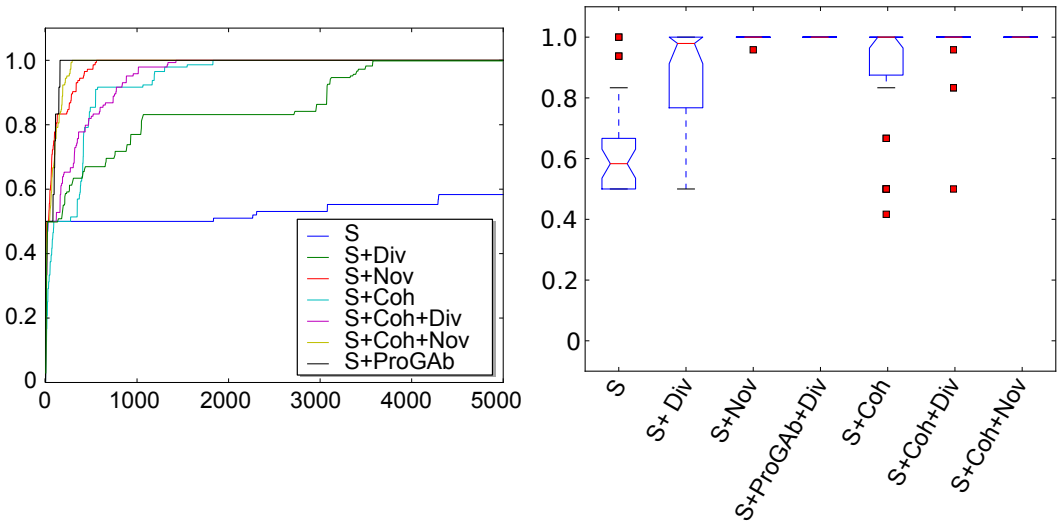
Durant les expériences, nous utiliserons les objectifs ci-dessous :
- **S** : Fitness discrète ;
- **Div** : objectif de diversité comportementale ;
- **Nov** : objectif de recherche de nouveauté ;
- **Coh** : objectif de cohérence introduit dans cette section ;
- **ProGab** : objectif ProGab décrit dans la section précédente.

Afin de tester l'influence de chaque objectif, les expériences sont lancées avec les différentes configurations décrites dans le tableau 5.9. L'algorithme multi-objectif NSGA-II (Deb, 2001) est utilisé et chacune des configurations est lancée 30 fois.

5.3.2.3 Résultats

La figure 5.9 affiche les différents résultats obtenus sur l'objectif principal pour toutes les configurations. Le graphique de gauche montre l'évolution de la fitness durant le processus d'évolution, chaque courbe représentant la médiane des meilleurs individus des 30 runs de chaque configuration. Les boîtes à moustache à droite représentent les performances des meilleurs individus de chaque run pour toutes les configurations. Les figures montrent que la fitness simple sans objectif auxiliaire obtient des résultats peu satisfaisants. On peut voir sur le graphique de gauche un plateau à $f = 0.5$, ce qui correspond aux contrôleurs qui se dirigent toujours du même côté du labyrinthe.

Nous pouvons constater que l'ajout un objectif auxiliaire permet d'obtenir une vitesse de convergence rapide : la fitness maximale est obtenue avec moins de 500 générations pour les configurations *S+ProGAb*, *S+Coh+Nov* et *S+Nov*, moins de 2000 pour *S+Coh+Div* et *S+Coh*. Le graphique de droite 5.9 nous permet d'observer l'efficacité de l'approche *ProGab* et *Coh* : avec *ProGab* et *S+Coh+Nov* tous les runs



(a) *(b)*

FIGURE 5.9 – Résultats obtenus sur l'objectif principal pour les différentes configurations : (a) évolution de la fitness (valeur médiane pour les 30 runs de chaque configuration) ; (b) performance des meilleurs individus de chaque run pour toutes les configurations.

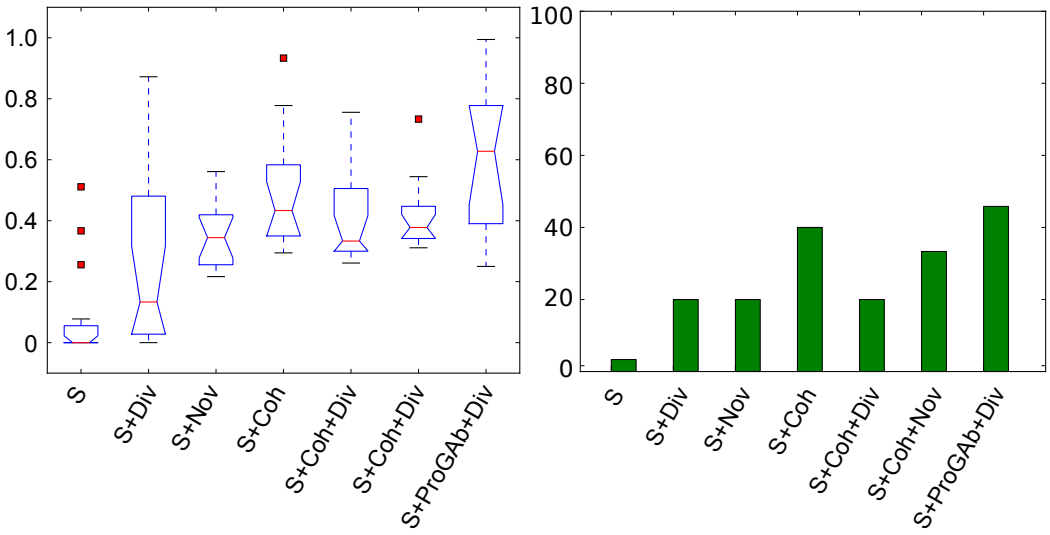
TABLE 5.10 – P-values pour la capacité de généralisation

	S	S DIV	S NOV	S COH	S COH DIV	S COH NOV	S PROGAB
S	x	0.00346	0.00013	2e-05	6e-05	4e-05	2e-05
S DIV	0.00346	x	0.15965	0.02817	0.11441	0.10661	0.0052
S NOV	0.00013	0.15965	x	0.02319	0.24674	0.10282	0.0025
S COH	2e-05	0.02817	0.02319	x	0.04444	0.19171	0.08207
S COH DIV	6e-05	0.11441	0.24674	0.04444	x	0.14499	0.00557
S COH NOV	4e-05	0.10661	0.10282	0.19171	0.14499	x	0.01404
S PROGAB	2e-05	0.0052	0.0025	0.08207	0.00557	0.01404	x

atteignent la fitness maximale à la fin du processus d'optimisation. Ces résultats sont significativement meilleurs qu'à la fin de *S* et *S+Div*.

Influence sur la capacité de généralisation. La figure 5.11 montre les résultats du test de capacité de généralisation post-optimisation effectué sur les meilleurs individus de chaque runs pour toutes les configurations. Les boîtes à moustache indiquent la capacité de généralisation des 15 meilleurs runs de chaque configuration, la valeur correspond au nombre de contextes normalisé dans lesquels l'agent réussit la tâche. Le graphique de droite indique le pourcentage d'individus réussissant au moins 60 des 180 contextes du test de généralisation. Le tableau 5.10 affiche les p-values correspondantes.

Comme attendu, en utilisant la fitness seule, les individus obtenus possèdent



(a) *(b)*

FIGURE 5.10 – Evaluation de la capacité de généralisation des meilleurs individus pour les différentes configurations : (a) Capacité de généralisation des 15 meilleurs runs de chaque configuration, la valeur correspond au nombre de contextes normalisé dans lesquels l'agent réussit la tâche ; (b) Pourcentage d'individus réussissant au moins 60 des 180 contextes.

une très faible capacité de généralisation : seul un individu sur 30 est capable de réussir la tâche sur plus de 60 contextes. Nous pouvons constater une capacité de généralisation significativement plus importante avec l'utilisation d'un objectif auxiliaire comme *Coher* ou *ProGAb* :

Influence sur la capacité de mémoire. La figure 5.11 affiche la proportion des meilleurs individus de chaque runs qui répondent aux critères suivant : la barre rouge correspond au pourcentage d'individus atteignant la fitness maximale ; la barre noire ceux ayant une mémoire robuste ; la barre indiquant ceux dotés à la fois de la fitness maximale d'une mémoire robuste.

Nous pouvons constater qu'avec *S*, peu d'individus à la mémoire robuste se développent. Seulement 2 individus sur 30 ont été capables de faire émerger ce type de mémoire. L'ajout d'un objectif d'exploration comportementale, configurations *S+Div* et *S+Nov*, augmente significativement la proportion d'individus possédant ce type de mémoire, avec respectivement 25% et 45% d'individus dotés de mémoire robuste. Cependant un nombre important d'individus obtient la fitness maximale en développant une mémoire fragile. L'utilisation d'un objectif *Coher* permet d'augmenter significativement la proportion d'individus réussissant la tâche avec une mémoire robuste avec près de la totalité des individus dotés de ce type de mémoire. Dans ce cas l'impact de l'ajout d'une diversité comportementale est peu significatif. Nous pouvons aussi observer que récompenser la capacité de généralisation des individus a un impact significatif sur la proportion d'individus à mémoire robuste.

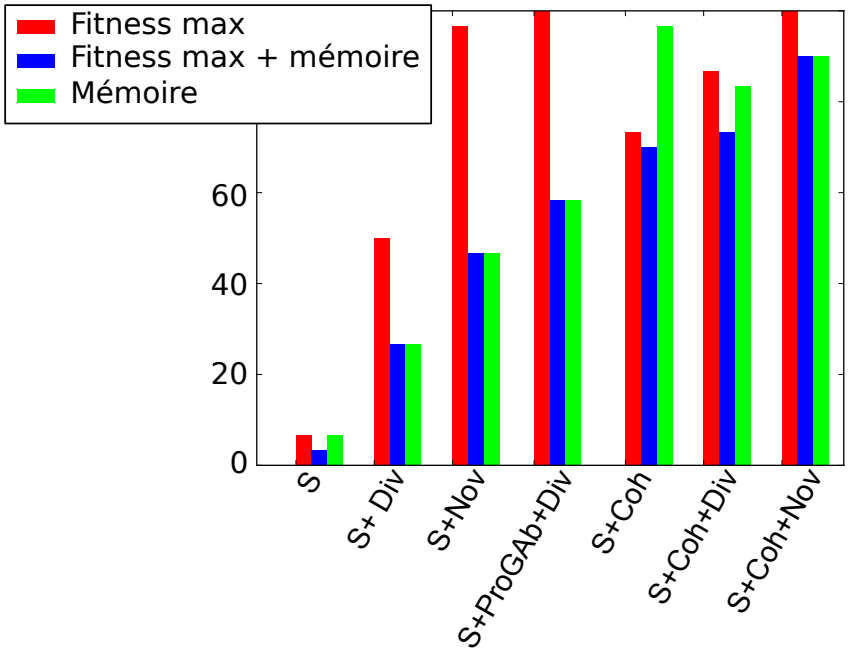


FIGURE 5.11 – Proportion des meilleurs individus de chaque runs qui répondent aux critères suivant : (1) atteint la fitness maximale ; (2) doté d'une mémoire robuste ; (3) atteint la fitness maximale et est doté d'une mémoire robuste.

5.4 Conclusion

Afin de favoriser l'émergence d'une forme de mémoire robuste et d'une capacité de généralisation, nous avons proposé au sein de ce chapitre deux approches différentes, la méthode *ProGAb* et la méthode *Coher*.

ProGAb s'inspire des méthodes d'apprentissage automatique afin d'évaluer la capacité de généralisation. Afin de se prémunir du coût computationnel dû à l'évaluation dans de multiples contextes, *ProGAb* s'appuie sur la construction d'un modèle de substitution (*Surrogate Model*) pour évaluer cette capacité de généralisation. Cette approche a permis d'obtenir des contrôleurs aux capacités de généralisation élevées tout en assurant un nombre total d'évaluations faible via l'usage d'un modèle de substitution, ainsi qu'un taux élevé d'individus dotés d'une mémoire robuste. Ceci valide notamment l'idée de construire le modèle de substitution de la capacité de généralisation à partir du comportement des solutions sur un unique contexte d'apprentissage.

Coher, quant à elle, se base sur la cohérence des comportements des individus en les comparant dans différentes situations et récompense (ou pénalise selon les cas) les comportement similaires. Cette approche a montré qu'il n'était pas forcément nécessaire d'intégrer un fort a priori pour obtenir un taux de mémoire robuste élevé au sein des individus.

Chapitre 6

Discussion et perspectives

6.1 Contributions

L'objectif de cette thèse est d'étudier la synthèse automatique d'architectures de contrôle pour robot capables de réaliser des tâches nécessitant le développement d'une forme de mémoire interne. Cela nous a permis de mettre en exergue différents points liés à l'émergence de la mémoire : le caractère trompeur, l'importance de l'utilisation des pressions de sélection et l'importance de l'utilisation d'une fitness de haut niveau.

6.1.1 L'émergence de mémoire, un problème trompeur

Dans le chapitre 3, nous avons tout d'abord montré la difficulté de mettre en place un protocole expérimental garantissant la nécessité d'une forme de mémoire interne afin de résoudre la tâche. Ceci a été illustré par l'intermédiaire de la tâche de "panneau de signalisation" où en exploitant l'environnement afin d'externaliser une forme de mémoire, des robots au comportement réactif ont pu résoudre la tâche.

A partir d'un bref état de l'art sur les tâches de mémoire utilisées en neurosciences et des modèles computationnels associés, nous avons pu mettre en place un protocole basé sur une tâche AX-CPT impliquant la mémoire de travail. L'utilisation de ce protocole particulier a permis de faire émerger une forme de mémoire interne mais avec un taux de convergence très faible.

Afin d'améliorer le taux de convergence, nous nous sommes intéressés à deux points :

– l'influence de différentes pressions de sélection ;
– l'influence de l'utilisation d'a priori au sein de la fitness.

Notre hypothèse de départ a été validée. Synthétiser une forme de mémoire interne au sein d'une architecture de contrôle est un problème trompeur.

6.1.2 Influence des pressions de sélection

Dans la section 4.2, nous avons pu vérifier l'influence de l'utilisation d'une diversité comportementale sur l'émergence d'une capacité de mémoire. L'influence de cette pression de sélection est statistiquement significative sur les différentes expériences.

Dans le chapitre 5, nous avons pu tester l'impact de deux autres pressions de sélection : une pression de sélection sur la capacité de généralisation ; une pression de sélection sur la cohérence des comportements. Dans les deux cas ces pressions

ont aidé à l'émergence d'une capacité de mémoire plus robuste. Une relation entre capacité de généralisation et capacité de mémoire robuste a pu être établie.

6.1.3 Influence de l'utilisation d'a priori au sein de la fitness

Au sein du chapitre 4, nous avons montré l'importance de garder une fitness de haut niveau afin de résoudre la tâche. En effet, mettre de l'a priori peut introduire des biais conduisant à des minima locaux : l'ajout d'a priori dans la fitness que ce soit en guidant la trajectoire à obtenir ou en guidant la position finale à obtenir ont donné de moins bons résultats qu'une fitness non guidée ne récompensant que le succès ou l'échec de la tâche. Les deux fitness incorporant de l'a priori tendent vers des minima locaux.

6.2 Perspectives

Les contributions évoquées précédemment soulèvent un certain nombre de questions pour les expériences futures. Nous en détaillerons quatre :

- Comment caractériser le comportement des individus obtenus par évolution ?
- Au delà de la mémoire, les pressions de sélection décrites précédemment peuvent-elles permettre l'émergence de modèles internes au sein des agents ?
- L'émergence de capacités cognitives plus complexes repose-t-elle sur l'élaboration de pressions de sélection nouvelles et/ou sur l'utilisation d'un codage indirect ?

6.2.1 Caractérisation du comportement des individus obtenus par évolution

La difficulté dans la mise en place d'un protocole permettant de faire émerger une forme de mémoire interne pose un problème plus général : il est difficile de déterminer si un agent réalise la tâche de la manière attendue. En effet, les solutions obtenues par évolution ont une forte propension à exploiter les failles d'un protocole expérimental. Cela motive le fait de mettre en place des outils permettant d'évaluer la complexité des comportements individus obtenus. En étendant le premier travail de caractérisation d'individus effectué dans la section 3.2.1, nous pouvons par exemple distinguer les catégories d'individus suivants :

- Robot "aveugle" : robot dans lequel l'information sensorimotrice venant de l'environnement externe manque ou ne joue aucun rôle pour déterminer les actions motrices du robot ;
- Robot réactif : robot qui ne possède pas d'état interne et dans lequel l'action motrice courante est seulement fonction de l'état courant des capteurs (Nolfi, 2001) ;
- Robot asymptotiquement réactif : robot dont le comportement devient réactif après n pas de temps ;

- Robot doté d'une capacité de mémoire interne : robot qui possède au moins une représentation interne pouvant co-déterminer au moment opportun avec les états courants des capteurs, les actions motrices de l'agent ;
- Robot doté d'une capacité de mémoire externe : robot qui, au lieu d'utiliser une mémoire interne, utilise l'environnement ou plutôt sa position par rapport à l'environnement comme mémoire externe.

Or, nous avons à disposition peu d'outils permettant d'évaluer la complexité des comportements des individus obtenus par évolution. Les mesures naïves telles l'évaluation de la complexité d'un réseau en fonction de sa taille ne sont pas efficaces : il n'existe en effet, pas de lien direct entre la taille des réseaux et la complexité des comportements. De plus, des architectures de contrôles réactives peuvent exhiber un comportement ayant une apparence complexe. Nous avons notamment utilisé dans la section 3.2.1, le calcul des valeurs propres des matrices de connexions pour déterminer la présence ou non d'une forme de mémoire interne. De leur coté, Marstaller et al. (2012) ont défini une mesure de représentation R, en terme de théorie de l'information comme l'entropie partagée entre les états de l'environnement et les états internes du "cerveau", étant donnés les états des capteurs.

Les objectifs de ces outils de catégorisation et d'évaluation de la complexité sont multiples. D'une part, comme beaucoup d'outils de mesure, nous pouvons envisager de les intégrer en tant que pression de sélection. Dans le cas de l'optimisation évolutionniste multi-objectif, nous pouvons citer l'exemple de l'indicateur d'hypervolume vu à la section 2.1.4.4. Cet indicateur a été originellement proposé et employé comme mesure de performance pour comparer différents MOEA (Zitzler et Thiele, 1999) et par la suite été intégré dans le processus d'optimisation (Knowles, 2002).

Dans une vision à plus long terme, cette capacité à caractériser des individus pourrait servir dans d'autres domaines. Dans les neurosciences expérimentales, par exemple, l'élaboration de protocoles expérimentaux est un processus extrêmement long, fastidieux et souvent effectué de manière empirique. La problèmatique est la suivante : sur une expérience où l'on veut tester la capacité de mémoire d'un animal, il faut tout d'abord s'assurer que celle-ci est bien nécessaire dans le protocole mis en place. Il est en effet difficile de déterminer à l'avance les différentes méthodes possibles permettant de résoudre une certaine tâche. La tâche du *roadsign* en est un exemple : requièrant en théorie une forme de mémoire, elle peut être résolue de manière réactive. Elle doit donc être utilisée avec précaution comme protocole de validation de présence ou déficit de mémoire. De plus, mettre en place des expériences sur des animaux en neurosciences expérimentales est très coûteux en temps, il peut être intéressant de disposer d'un outil de validation rapide. Pour ce faire, nous pourrions envisager l'utilisation de méthodes évolutionnistes comme aide à l'élaboration et processus de validation de protocoles expérimentaux. En effet, en analysant le comportement des individus obtenus par évolution sur ces protocoles grâce aux outils de catégorisation définis précédemment, il serait possible de déterminer si ceux-ci impliquent nécessairement de posséder une forme de mémoire interne.

6.2.2 Vers l'émergence de modèles internes ?

Selon Craik (1943), les modèles internes seraient des représentations supposées de la réalité (ou de l'imagination) au sein du cerveau. Ils permettraient non seulement de se représenter l'environnement externe, mais aussi les actions possibles de l'individu lui-même. Posséder une forme de représentation spatiale par exemple, permet de développer une capacité de discriminer différents lieux et de se localiser. Ces différentes compétences sont nécessaires pour résoudre de nombreux problèmes de navigation (Trullier et al., 1997).

Viser à l'émergence de modèles internes au sein d'un agent peut être vu comme une extension du travail effectué au sein de cette thèse. Un des objectifs pourrait être de déterminer si les différentes pressions de sélection évoquées précédemment restent applicables sur ce domaine.

Différents travaux tentent de faire émerger des modèles internes (Nolfi, 2005; Gigliotta et Nolfi, 2008; Gigliotta et al., 2010, 2011) sur des problèmes de catégorisation des perceptions dans lesquels un agent doit lui même découvrir des catégories abstraites. Cependant, ces approches ont plutôt tendance à se baser sur des réseaux ad hoc contenant explicitement des neurones dédiés à cette représentation interne ainsi qu'une fitness récompensant explicitement la présence de cette représentation. L'utilisation des pressions de sélection décrites précédemment peut être une solution afin de faire émerger des modèles internes sans utilisation de réseaux et de fitness ad hoc.

D'autres pressions de sélection peuvent être envisagées afin d'aider à l'émergence de modèles internes au sein des individus. Nous avons vu précédemment qu'une des clés permettant l'émergence d'une mémoire interne reposait sur l'utilisation d'un objectif d'exploration (diversité comportementale, nouveauté). S'inspirer des recherches faites en génétique des populations peut fournir des pistes sur la façon de maintenir une certaine diversité d'individus au sein d'une population. En effet, cette discipline s'intéresse principalement à la variabilité d'origine génétique présente dans les populations, cette variabilité étant désignée sous le nom de polymorphisme.

Par exemple, un lien a été montré entre polymorphisme et pressions de sélection (Powell, 1971). Le polymorphisme est maximum dans les populations qui ont été soumises aux variations écologiques les plus nombreuses et les plus violentes et minimum dans celles qui ont vécu dans la stabilité. D'une manière générale, les variations violentes et fréquentes entretiennent un polymorphisme élevé alors que des conditions sévères mais permanentes tendent à l'appauvrissement génétique. Il pourrait donc être intéressant de tester des variations brutales de l'environnement pendant le processus d'optimisation, afin de déterminer l'impact sur la diversité au sein de la population.

D'autre part, les mécanismes qui assurent le polymorphisme sont différents chez les êtres supérieurs et chez les organismes inférieurs (Ruffié, 1986). Au niveau des bactéries ce sont les mutations qui jouent un rôle dans le maintien du polymorphisme, alors qu'au niveau des organismes supérieurs le polymorphisme est principalement assuré par la recombinaison génétique. Nous pouvons donc émettre l'hy-

pothèse que l'opérateur de croisement est nécessaire, à terme, pour élaborer des structures plus complexes.

Or, dans de nombreuses expériences abordant l'évolution de topologie l'opérateur de croisement n'est pas présent (Angeline et al., 1994). La permutation n'est pas considérée comme seulement inefficace mais aussi susceptible de produire des individus moins performants (Yao et Islam, 2008), ceci à cause du *problème de permutation* (Angeline et al., 1994; Belew et al., 1990; Hancock, 1992; Haflidason et Neville, 2009). Celui-ci est dû au fait qu'un même réseau peut être représenté dans un codage génétique par de nombreuses codifications différentes. Différentes opérateurs de croisement ont été proposées afin de résoudre ce problème (Thierens, 1996; Stanley et Miikkulainen, 2002; García-Pedrajas et al., 2006).

6.2.3 Emergence de capacités cognitives plus complexes : Pression de sélection sur la structure et/ou utilisation d'un codage indirect ?

De nombreuses expériences ont montré les limitations de l'utilisation d'un codage direct (Mouret et al., 2010; Pinville et Doncieux, 2010; Tonelli et Mouret, 2011a,b; Tonelli, 2012). Nous pouvons notamment citer les problèmes de passages à l'échelle (Mouret et al., 2010) : un codage direct éprouvera des difficultés à traiter des problèmes avec de nombreuses entrées et sorties. Il est de la même manière difficile d'obtenir des réseaux dotés de régularités avec un codage direct (Tonelli et Mouret, 2011a,b; Tonelli, 2012). Un codage direct possède une tendance à privilégier des réseaux très peu polyvalents et spécifiques à une tâche (Pinville et Doncieux, 2010).

En comparant les réseaux obtenus par neuroévolution avec un codage direct à des modèles de réseaux de neurones issus des neurosciences, nous avons pu constater qu'il existait très peu de similarités entre les deux (Mouret et al., 2010; Pinville et Doncieux, 2010). D'un côté en neuroévolution, les réseaux sont souvent composés de quelques dizaines de neurones avec une connectivité non régulière, alors qu'au niveau des modèles issus des neurosciences de nombreux modèles sont constitués de plusieurs centaines de neurones bien organisés. L'observation de différents modèles computationnels (Gurney et al., 2001a,b; Girard et al., 2008; Trappenberg et al., 2001; Vitay et al., 2005; Rougier et Vitay, 2006; Rougier, 2006), montre que ces réseaux, au lieu de représenter les neurones un à un, utilisent différents blocs de construction. Le cerveau ayant un grand nombre de régularités.

Parmi ces blocs fréquemment utilisés nous retrouvons l'emploi de cartes de neurones en lieu et place de neurones individuels. Ces cartes sont définies comme des grilles de neurones identiques organisés spatialement. De nombreux modèles emploient seulement des cartes avec la même dimension, arbitrairement fixée aux dimensions des entrées. Une telle structure est largement présente dans le neocortex (Kaas, 1997). Afin de connecter les cartes de neurones entre elles, différents schémas de connexion existent :

– 1-1 : chaque neurone d'une carte M_1 est relié au neurone correspondant d'une

carte M_2 avec des poids de connexion identiques sur chaque connexion ;
- 1-all avec poids constant : chaque neurone d'une carte M_1 est relié à tous les neurones d'une carte M_2 avec un poids de connexion constant ;
- 1-all suivant une distribution gaussienne : identique au schéma de connexion précédent, mais dans ce cas les poids de connexions suivent une distribution gaussienne.

D'autres domaines de recherche s'intéressent spécifiquement à l'étude des structures des réseaux biologiques. La connectomique (Sporns et al., 2005; Bullmore et Sporns, 2009; Rubinov et Sporns, 2010; Hagmann et al., 2010), par exemple, qui étudie l'ensemble des connexions neuronales du cerveau ou la biologie des systèmes (Alon, 2006), qui a comme objectif de caractériser les différents composants élémentaires d'un système biologique et leurs interactions. L'analyse effectuée dans ces différents domaines, en grande partie basée sur la théorie des graphes, a permis de déterminer la présence de certaines structures récurrentes à différentes échelles. Nous citerons à titre d'illustration deux de ces structures : les réseaux petit-monde et les motifs.

Les propriétés d'un réseau petit monde ("small-world") (Watts et Strogatz, 1998) sont les suivantes : la distance moyenne entre deux nœuds est proportionnelle au logarithme du nombre de nœuds ; un grand nombre de structures sont proches de cliques, c'est-à-dire que les voisins d'un sommet donné seront souvent connectés entre eux. On retrouve cette caractéristique dans de nombreux types de réseaux tel que dans le réseau du C. Elegans (Watts et Strogatz, 1998), dans le cerveau, au niveau de la formation réticulée (Humphries et al., 2006). Certains papiers tendent à montrer qu'un réseau de neurones petit monde peut exhiber une forme de mémoire à court terme. Roxin et al. (2004) ont développé un modèle bistable, propriété importante pour la mémoire.

L'analyse de certains types de réseaux tels que les réseaux régulateurs de gènes, réseaux de neurones, ont permis de détecter certains blocs de construction spécifiques. Ces sous-réseaux sont appelés motifs (Kashtan et Alon, 2005; Milo et al., 2002; Alon, 2007, 2006). On les retrouvent plus fréquemment, et de manière statistiquement significative, dans de nombreux réseaux biologiques par rapport à des réseaux générés aléatoirement [1]. Les premiers motifs ont été découverts au niveau du réseau régulateur de gènes de la bactérie E. coli. (Milo et al., 2002).

6.2.4 Quelles approches possibles ?

Nous avons décrit précédemment différentes structures présentes au sein de réseaux biologiques à différentes échelles. L'apparition de telles structures au sein des réseaux obtenus par neuroévolution est souhaitable afin d'espérer l'émergence de capacités cognitives plus complexes. Pour ce faire, il existe plusieurs méthodes :

1. On peut trouver différents outils permettant d'analyser les réseaux et de détecter ces sous motifs, tels que Fanmod http://theinf1.informatik.uni-jena.de/~wernicke/motifs/, Mavisto http://mavisto.ipk-gatersleben.de/, Cytoscape http://www.cytoscape.org/, Neat : http://rsat.ulb.ac.be/rsat/index_neat.html

une première approche est de mettre en place un codage indirect contenant spécifiquement ce type de structures, une autre est de pousser à l'émergence de ces structures à l'aide de pressions de sélection spécifiques.

Des premiers travaux ont été effectués sur la mise en place d'un codage indirect basé sur l'utilisation de blocs de construction issu des neurosciences : le codage Evoneuro (Mouret et al., 2010; Pinville et Doncieux, 2010; Tonelli et Mouret, 2011b; Tonelli, 2012; Tonelli et Mouret, 2011a) utilise des cartes de neurones et les schémas de connexions décrits précédemment. Ce codage fait évoluer un graphe qui est ensuite utilisé pour générer le réseau de neurones. Chaque génotype est donc un graphe orienté où chaque nœud et chaque lien possède un nombre prédéfini de paramètres modifiables par évolution. Contrairement au codage direct, le graphe n'est pas utilisé tel quel comme réseau de neurones mais est utilisé comme patron pour créer une structure neuronale plus complexe. Lors de la création du phénotype à partir du génotype, chaque nœud est transformé en carte de neurones ou en neurone seul en fonction d'un paramètre interne et chaque connexion est transformée en schéma de connexion (1-1, 1-all, 1-all gaussien).

Les premiers résultats ont permis de montrer différentes caractéristiques d'EvoNeuro :

- Passage à l'échelle (*Scalabilité*) : contrairement à un codage direct classique, Evoneuro est capable de réaliser des tâches de *winner-takes-all* avec un grand nombre d'entrées et de sorties (Mouret et al., 2010) ;
- Adaptation : la régularité présente dans les réseaux obtenus par Evoneuro leur permettent de s'adapter efficacement à des situations sur lesquelles ils n'ont jamais été testés (Pinville et Doncieux, 2010; Tonelli et Mouret, 2011b; Tonelli, 2012; Tonelli et Mouret, 2011a).

Ces premiers éléments nous confortent dans l'idée d'intégrer d'autres blocs de construction issus des neurosciences au sein d'un codage indirect du type Evoneuro.

D'autre part il est possible de pousser à l'émergence de ces structures via des pressions de sélection spécifiques. Il existe cependant une limitation dans l'utilisation d'un grand nombre de pressions de sélection : une approche Pareto optimale est difficilement utilisable avec un nombre important d'objectifs. En effet, lorsque le nombre de dimensions augmente, la probabilité de trouver une solution qui domine une autre solution est très faible. Il y a donc un risque de problème de bootstrap si l'on augmente fortement le nombre de pressions de sélection. De plus, la plupart des algorithmes multi-objectifs tel que NSGA-II (voir section 2.1.4.3 pour plus de détails) sont peu efficaces avec plus de trois objectifs. Une possibilité peut être d'utiliser d'autres relations de domination que celle de Pareto. La k-dominance (Chan et al., 2006), par exemple, relaxe l'idée de dominance : un point p k-domine un autre point q si il y a $k(\leq d)$ dimensions sur lesquelles p est meilleur ou égal à q et meilleur sur au moins une de ces k dimensions (avec d le nombre total de dimensions).

6.2.5 Conclusions

L'utilisation de différentes pressions de sélections présentées dans ce manuscrit, apporte une première solution permettant de faire face à la difficulté d'obtenir via évolution des contrôleurs autres que réactifs.

A partir des résultats obtenus, diverses extensions de ce travail ont été discutées, comme l'utilisation de ces pressions de sélection afin d'aider à l'émergence de modèles internes au sein d'un contrôleur. Il semble néanmoins nécessaire de définir de nouvelles pressions de sélection afin de permettre l'émergence de capacités cognitives plus complexes. Quelques pistes ont été évoquées précédemment : s'inspirer de travaux issus de la génétique des populations, définir des pressions visant à l'émergence de structures spécifiques au sein du contrôleur ou encore affiner la mise en place d'un codage indirect utilisant des blocs de construction.

Chapitre 7

Conclusion

L'objectif de cette thèse est de faire émerger de manière automatique des architectures de contrôle pour robots capables de réaliser des tâches nécessitant le développement d'une forme de mémoire interne. Cet objectif est atteint et nous a permis de mettre en exergue différents points liés à l'émergence de la mémoire : le caractère trompeur, l'importance de l'utilisation des pressions de sélection, et l'importance de l'utilisation d'une fitness de haut niveau.

Le chapitre 2 présente un état de l'art sur la robotique évolutionniste. Il contient une présentation des différents algorithmes évolutionnistes les plus usités à ce jour dans ce domaine, un aperçu des différents codages permettant de représenter et de faire évoluer les réseaux de neurones artificiels ainsi qu'un panel des différents types de tâches abordées en évolution.

Dans le chapitre 3, nous avons tout d'abord montré la difficulté de mettre en place un protocole expérimental garantissant la nécessité d'une forme de mémoire interne afin de résoudre la tâche. Ceci a été illustré par l'intermédiaire de la tâche de "panneau de signalisation" où en exploitant l'environnement afin d'externaliser une forme de mémoire, des robots au comportement réactif ont pu résoudre la tâche. A partir d'un bref état de l'art sur les tâches de mémoire utilisées en neurosciences et des modèles computationnels associés, nous avons pu mettre en place un protocole basé sur une tâche AX-CPT impliquant la mémoire de travail. L'utilisation de ce protocole particulier a permis de faire émerger une forme de mémoire interne mais avec un taux de convergence très faible. Afin d'améliorer le taux de convergence, nous sommes intéressés à deux points :

– l'influence de différentes pressions de sélection ;
– l'influence de l'utilisation d'a priori au sein de la fitness.

Notre hypothèse de départ a été validée. Synthétiser une forme de mémoire interne au sein d'une architecture de contrôle est un problème trompeur.

Au sein du chapitre 4, nous avons montré l'importance de garder une fitness de haut niveau afin de résoudre la tâche. En effet, mettre de l'a priori peut introduire des biais conduisant à des minima locaux : l'ajout d'a priori dans la fitness que ce soit en guidant la trajectoire à obtenir ou en guidant la position finale à obtenir ont donné de moins bons résultats qu'une fitness non guidée ne récompensant que le succès ou l'échec de la tâche. Les deux fitness incorporant de l'a priori tendent vers des minima locaux. Nous avons vérifier l'influence de l'utilisation d'une diversité comportementale sur l'émergence d'une capacité de mémoire. L'influence de cette pression de sélection est statistiquement significative sur les différentes expériences.

Dans le chapitre 5, nous avons testé l'impact de deux autres pressions de sé-

lection : une pression de sélection sur la capacité de généralisation ; une pression de sélection sur la cohérence des comportements. Dans les deux cas ces pressions ont aidé a l'émergence d'une capacité de mémoire plus robuste. Une relation entre capacité de généralisation et capacité de mémoire robuste a pu être établie.

Le chapitre 6 expose différentes questions scientifiques posées par les résultats des chapitres précédents : la nécessité d'avoir à disposition des outils de catégorisation mesurant la complexité des individus obtenus par évolution ; la possibilité de faire émerger des modèles internes au sein des contrôleurs en utilisant les pressions de sélections décrites précédemment. Enfin diverses pistes sont abordées afin de favoriser l'émergence de comportements plus complexes au sein des robots. Nous évoquons notamment le dilemme entre l'utilisation au sein d'un codage indirect de blocs de constructions issus de différents domaines tels que les neurosciences et le fait de pousser à l'émergence de ces blocs de constructions via des pressions de sélection spécifiques.

Bibliographie

Abbass, H. et Deb, K. (2003). Searching under multi-evolutionary pressures. In *Evolutionary Multi-criterion Optimization*, page 67. Springer. (Cité en page 79.)

Abeles, M. (1991). *Corticonics : Neural circuits of the cerebral cortex*. Cambridge Univ Pr. (Cité en page 48.)

Alon, U. (2006). *An introduction to systems biology : design principles of biological circuits*, volume 10. Chapman & Hall/CRC. (Cité en page 138.)

Alon, U. (2007). Network motifs : theory and experimental approaches. *Nature reviews. Genetics*, 8(6) :450–61. (Cité en page 138.)

Alpaydin, E. (2004). *Introduction to machine learning*. MIT Press. (Cité en pages 104 et 109.)

Altenberg, L. (2004). Modularity in Evolution : Some Low-Level Questions. In *Modularity, Understanding the Development and Evolution of Natural Complex Systems*, pages 1–32. (Cité en page 35.)

Angeline, P. J., Saunders, G. M., et Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *Neural Networks, IEEE Transactions on*, 5(1) :54–65. (Cité en page 137.)

Antonelo, E., Schrauwen, B., Dutoit, X., Stroobandt, D., et Nuttin, M. (2007). Event detection and localization in mobile robot navigation using reservoir computing. *Artificial Neural Networks–ICANN 2007*, pages 660–669. (Cité en page 32.)

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman and Hall, London. (Cité en page 44.)

Ashcraft, M. H. (1994). *Human memory and cognition*. HarperCollins College Publishers, New York, NY, USA. (Cité en page 44.)

Atkinson, R. C. et Shiffrin, R. M. (1968). Human memory : A proposed system and its control processes. *The psychology of learning and motivation : Advances in research and theory*, 2 :89–195. (Cité en page 45.)

Atkinson, R. C. et Shiffrin, R. M. (1971). *The control processes of short-term memory*. Institute for Mathematical Studies in the Social Sciences, Stanford University. (Cité en pages 45 et 46.)

Azevedo, F. A. C., Carvalho, L. R. B., Grinberg, L. T., Farfel, J. M., Ferretti, R. E. L., Leite, R. E. P., Jacob Filho, W., Lent, R., et Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain

an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5) :532–541. (Cité en page 27.)

Bäck, T., Hoffmeister, F., et Schwefel, H. P. (1991). A survey of evolution strategies. *Proceedings of the 4th International Conference on Genetic ALgorithms and their Applications*. (Cité en page 15.)

Baddeley, A. (1992). Working memory. *Science*, 255(5044) :556–559. (Cité en page 7.)

Baddeley, A. (2000). The episodic buffer : a new component of working memory ? *Trends in Cognitive Sciences*, 4(11) :417–423. (Cité en pages 47 et 48.)

Baddeley, A. D. (1997). *Human memory : Theory and practice (Revised edition)*. Psychology Press, Hove, UK. (Cité en page 44.)

Baddeley, A. D., Hitch, G. J., et Others (1974). Working memory. *The psychology of learning and motivation*, 8 :47–89. (Cité en pages 46 et 47.)

Bader, J. et Zitzler, E. (2011). HypE : An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*. (Cité en pages 25 et 26.)

Baele, G., Bredeche, N., Haasdijk, E., Maere, S., Michiels, N., de Peer, Y., Schmickl, T., Schwarzer, C., et Thenius, R. (2009). Open-ended on-board evolutionary robotics for robot swarms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1123–1130. IEEE. (Cité en page 7.)

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA. L. Erlbaum Associates Inc. (Cité en page 16.)

Bakker, B. (2002). Reinforcement learning with long short-term memory. *Advances in neural information processing systems*. (Cité en page 50.)

Balkenius, C. (1995). *Natural intelligence in artificial creatures*. [Kognitionsforskning], Univ. (Cité en page 8.)

Banzhaf, W., Nordin, P., et Olmer, M. (1997). Generating Adaptive Behavior using Function Regression within Genetic Programming and a Real Robot. *Evolutionary Computation*, pages 1–13. (Cité en page 38.)

Barate, R. et Manzanera, A. (2008). Generalization performance of vision based controllers for mobile robots evolved with genetic programming. *Proc. of GECCO*, pages 1331–1339. (Cité en pages 102, 103, 105, 106 et 107.)

Barberá, H. M. et Skarmeta, A. G. (2002). A framework for defining and learning fuzzy behaviors for autonomous mobile robots. *International Journal of Intelligent Systems*, 17(1) :1–20. (Cité en page 6.)

Bares, J. et Wettergreen, D. (1999). Dante II : Technical description, results and lessons learned. *International Journal of Robotics Research*, 18 :621–649. (Cité en page 3.)

Barlow, G., Mattos, L., Grant, E., et Oh, C. (2005). Transference of Evolved Unmanned Aerial Vehicle Controllers to a Wheeled Mobile Robot. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2087–2092. (Cité en page 38.)

Barlow, G. et Oh, C. K. (2006). Robustness analysis of genetic programming controllers for unmanned aerial vehicles. *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*, page 135. (Cité en pages 78 et 106.)

Beer, R. D. (1995). On the Dynamics of Small Continuous-Time Recurrent Neural Networks. *Adaptive Behavior*, 3(4) :469–509. (Cité en page 32.)

Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From animals to animats*. (Cité en pages 8, 40 et 52.)

Beer, R. D. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4) :209–243. (Cité en page 40.)

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18(12) :3009–3051. (Cité en page 32.)

Beer, R. D. et Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adapt. Behav.*, 1(1). (Cité en page 50.)

Bekele, E. G. et Nicklow, J. W. (2007). Multi-objective automatic calibration of SWAT using NSGA-II. *Journal of Hydrology*, 341(3-4) :165–176. (Cité en page 27.)

Belew, R. K., McInerney, J., et Schraudolph, N. N. (1990). Evolving networks : Using the genetic algorithm with connectionist learning. In *In*. Citeseer. (Cité en page 137.)

Benkirane, R., Morin, E., Prigogine, I., et Varela, F. (2002). *La complexit{é}, vertiges et promesses : 18 histoires de sciences*. Le pommier Paris. (Cité en page 6.)

Bergfeldt, N. et Linaker, F. (2002). Self-organized modulation of a neural robot controller. In *Proceedings of the International Joint Conference on Neural Networks, Hawaii*, volume pages, pages 495–500. Citeseer. (Cité en page 40.)

Berlanga, A., Sanchis, A., Isasi, P., et Molina, J. (2002). Neural Network Controller against Environment : A Coevolutive approach to Generalize Robot Navigation Behavior. *Journal of Intelligent and Robotic Systems*, pages 139–166. (Cité en pages 102, 103, 105, 106 et 107.)

Beume, N., Naujoks, B., et Emmerich, M. (2007). SMS-EMOA : Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3) :1653–1669. (Cité en pages 25 et 26.)

Beume, N. et Rudolph, G. (2006). Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. In *Proceedings of the Second IASTED Conference on Computational Intelligence (CI 2006)*, pages 231—-236. (Cité en page 25.)

Birattari, M., Stutzle, T., Paquete, L., et Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *GECCO*, volume 2, pages 11–18. Citeseer. (Cité en page 108.)

Blynel, J. et Floreano, D. (2003). Exploring the T-maze : Evolving learning-like robot behaviors using CTRNNs. *Lecture notes in computer science*, pages 593–604. (Cité en page 40.)

Boeing, A., Hanham, S., et Braunl, T. (2004). Evolving autonomous biped control from simulation to reality. *on Autonomous Robots*, 445(6) :440–445. (Cité en page 38.)

Bongard, J. (2011a). Innocent until proven guilty : Reducing robot shaping from polynomial to linear time. *Evolutionary Computation, IEEE Transactions on*, (99) :1–15. (Cité en pages 78 et 108.)

Bongard, J. (2011b). Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 2010 :1234–1239. (Cité en pages 32, 106 et 108.)

Bongard, J. et Hornby, G. S. (2010). Guarding against premature convergence while accelerating evolutionary search. *GECCO*. (Cité en page 108.)

Bongard, J. et Pfeifer, R. (2002). Relating Neural Network Performance to Morphological Differences in Embodied Agents. In *Proceedings of the Sixth International Conference on Cognitive and Neural Systems*, Boston. (Cité en page 32.)

Branke, J. (1995). Evolutionary algorithms for neural network design and training. In *In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications*, pages 1–21. (Cité en page 33.)

Braver, T. S. et Barch, D. M. (2002). A theory of cognitive control, aging cognition, and neuromodulation. *Neurosci. biobehav. r.*, 26(7) :809–817. (Cité en page 62.)

Braver, T. S., Cohen, J. D., et Servan-Schreiber, D. (1995). A computational model of prefrontal cortex function. *Nips*, page141148 :141–148. (Cité en pages 62, 63 et 64.)

Braver, T. S. T., Barch, D. M. D., et Cohen, J. D. (1999). Cognition and control in schizophrenia : a computational model of dopamine and prefrontal function. *Biol. psychiat.*, 46(3) :312–328. (Cité en page 62.)

Brette, R. (2003). Modèles Impulsionnels de Réseaux de Neurones Biologiques. (Cité en page 30.)

Brindle, A. (1981). *Genetic algorithms for function optimization*. PhD thesis, University of Alberta, Department of Computer Science. (Cité en page 16.)

Brockhoff, D. et Zitzler, E. (2007). Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. *Evolutionary Computation, 2007. CEC*, pages 2086–2093. (Cité en page 25.)

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*. (Cité en page 2.)

Brooks, R. A. (1991). Intelligence without representation. (Cité en pages 2 et 5.)

Buccafusco, J. J. (2008). *Methods of Behavior Analysis in Neuroscience, Second Edition (New Frontiers in Neuroscience)*. CRC Press. (Cité en page 60.)

Buhmann, M. D. (2003). *Radial basis functions : theory and implementations*, volume 12. Cambridge university press. (Cité en page 29.)

Bui, L. T., Abbass, H., et Branke, J. (2005). Multiobjective optimization for dynamic environments. *In proc. of IEEE-CEC'05*, pages 2349–2356. (Cité en pages 80 et 85.)

Bullmore, E. et Sporns, O. (2009). Complex brain networks : graph theoretical analysis of structural and functional systems. *Nature reviews. Neuroscience*, 10(3) :186–98. (Cité en page 138.)

Bunke, H. et Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3) :255–259. (Cité en page 80.)

Capi, G. et Doya, K. (2005). Evolution of recurrent neural controllers using an extended parallel genetic algorithm. *Robotics and Autonomous Systems*, 52 :148–159. (Cité en pages 38 et 41.)

Chan, C. Y., Jagadish, H. V., Tan, K. L., Tung, A. K. H., et Zhang, Z. (2006). Finding k-dominant skylines in high dimensional space. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 503–514. ACM. (Cité en page 139.)

Charnes, A. et Cooper, W. W. (1957). Management models and industrial applications of linear programming. *Management Science*, 4(1) :38–91. (Cité en page 19.)

Charnes, A. et Cooper, W. W. (1961). *Management Models and Industrial Applications of Linear Programming, vol. 1*. John Wiley, New York, New York, USA. (Cité en page 18.)

Chemova, S. et Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2562–2567. (Cité en page 38.)

Clark, A. (1997). *Being there : Putting brain, body, and world together again*. MIT press. (Cité en pages 40 et 51.)

Cliff, D., Husbands, P., Harvey, I., et Others (1993). Evolving visually guided robots. *From animals to animats*, 2 :374–383. (Cité en page 105.)

Clune, J., Beckmann, B. E., Ofria, C., et Pennock, R. T. (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2764–2771. IEEE. (Cité en page 36.)

Clune, J., Mouret, J.-B., et Lipson, H. (2012). The evolutionary origins of modularity. *arXiv preprint arXiv :1207.2743*, pages 1–17. (Cité en page 35.)

Coello, C. A. (2006). Evolutionary multi-objective optimization : a historical view of the field. *Computational Intelligence Magazine, IEEE*, 1(1) :28–36. (Cité en page 27.)

Coello, C. A., Christiansen, A. D., et Aguirre, A. H. (1995). Multiobjective Design Optimization of Counterweight Balancing of a Robot Arm using Genetic Algorithms. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, pages 20—-23, Washington, DC, USA. IEEE Computer Society. (Cité en page 18.)

Coello, C. A. et Lamont, G. B. (2004). *Applications of multi-objective evolutionary algorithms*, volume 1. World Scientific Publishing Company Incorporated. (Cité en pages 13 et 27.)

Cohen, N. J. et Squire, L. R. (1980). Preserved learning and retention of pattern-analyzing skill in amnesia : Dissociation of knowing how and knowing that. *Science*, 210(4466) :207–210. (Cité en page 46.)

Corne, D., Jerram, N. R., Knowles, J. D., Oates, M. J., et J, M. (2001). PESA-II : Region-based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001*, pages 283–290. Morgan Kaufmann Publishers. (Cité en pages 22 et 26.)

Corne, D. W., Knowles, J. D., et Oates, M. J. (2000). The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, UK. Springer-Verlag. (Cité en pages 21 et 26.)

Cowan, N. (2001). The magical number 4 in short-term memory : A reconsideration of mental storage capacity. *Behavioral and brain sciences*, 24(01) :87–114. (Cité en page 46.)

Craik, K. J. W. (1943). *The nature of explanation*. Cambridge University Press. (Cité en page 136.)

Cramer, N. L. (1985). A Representation for the Adaptive Generation of Simple Sequential Programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187, Hillsdale, NJ, USA. L. Erlbaum Associates Inc. (Cité en page 13.)

Crevier, D. (1993). *AI : the tumultuous history of the search for artificial intelligence*. Basic Books, Inc., New York, NY, USA. (Cité en page 2.)

Darwin, C. (1859). *Origin of Species*. (Cité en page 12.)

Dayan, P. et Abbott, L. F. (2005). *Theoretical Neuroscience : Computational and Mathematical Modeling of Neural Systems*. The MIT Press. (Cité en page 29.)

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, Ann Arbor, MI, USA. (Cité en page 13.)

De Jong, K. A., Watson, R., et Pollack, J. (2001). Reducing bloat and promoting diversity using multi-objective methods. *GECCO*. (Cité en page 79.)

De Loor, P., Manac'h, K., et Tisseau, J. (2009). Enaction-Based Artificial Intelligence : Toward Co-evolution with Humans in the Loop. *Minds Mach.*, 19(3) :319–343. (Cité en page 5.)

De Nardi, R., Togelius, J., Holland, O. E., et Lucas, S. M. (2006). Evolution of Neural Networks for Helicopter Control : Why Modularity Matters. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1799–1806. (Cité en page 77.)

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons. (Cité en pages 53, 55, 82 et 128.)

Deb, K., Mohan, M., et Mishra, S. (2005). Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, 13(4) :501–525. (Cité en pages 23 et 24.)

Deb, K., Pratap, A., Agarwal, S., et Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2) :182–197. (Cité en pages 22, 26 et 114.)

Dehaene, S., Changeux, J. P., et Alexander, G. E. (1989). A simple model of prefrontal cortex function in delayed-response tasks. *J. Cognitive Neurosci.*, 1(3) :244–261. (Cité en pages 61 et 63.)

Demb, J. B., Desmond, J. E., Wagner, A. D., Vaidya, C. J., Glover, G. H., Gabrieli, J. D. E., et Others (1995). Semantic encoding and retrieval in the left inferior prefrontal cortex : a functional MRI study of task difficulty and process specificity. *Journal of Neuroscience*, 15(9) :5870–5878. (Cité en page 47.)

Di Paolo, E. (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In *From Animals to Animats 6. Proc. 6th Int. Conf. on Simulation of Adaptive Behavior, Paris, France*. (Cité en page 106.)

di Pellegrino, G. et Wise, S. P. (1993). Visuospatial versus visuomotor activity in the premotor and prefrontal cortex of a primate. *J. Neurosci.*, 13(3) :1227–1243. (Cité en page 61.)

Doncieux, S. et Meyer, J.-A. (2003). Evolving neural networks for the control of a lenticular blimp. In *Applications of Evolutionary Computing, EvoWorkshops2003 : EvoBIO, EvoCOP, EvoIASP, Evo- MUSART, EvoROB, EvoSTIM*. Springer. (Cité en page 34.)

Doncieux, S. et Mouret, J.-B. (2010a). Behavioral diversity measures for evolutionary robotics. In *Proc. of IEEE CEC*, volume 2010, pages 1–8. (Cité en pages 7, 53 et 81.)

Doncieux, S. et Mouret, J.-B. (2010b). Behavioral diversity measures for Evolutionary Robotics. In *Proceedings of IEEE-CEC'10*, pages 1303–1310. (Cité en pages 39, 102, 103, 105, 112, 114, 115 et 116.)

Doncieux, S., Mouret, J.-B., et Bredeche, N. (2009). Exploring New Horizons in Evolutionary Design of Robots. In *IROS Workshop*, pages 5–12. (Cité en page 53.)

Doncieux, S., Mouret, J.-B., Bredeche, N., et Padois, V. (2011). *Evolutionary Robotics : Exploring New Horizons*, pages 3–25. Springer. (Cité en page 37.)

Drchal, J., Kapral, O., Koutník, J., et Šnorek, M. (2009). Combining Multiple Inputs in HyperNEAT Mobile Agent Controller. In *Proceedings of the 19th International Conference on Artificial Neural Networks*, pages 775–783, Berlin. Springer. (Cité en page 36.)

Droulez, J. et Berthoz, A. (1991). A neural network model of sensoritopic maps with predictive short-term memory properties. *Proc. Natl. Acad. Sci. U.S.A.*, 88(21) :9653–9657. (Cité en pages 61 et 63.)

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., et Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, pages 155–161. (Cité en page 109.)

Duckstein, L. (1981). Multiobjective optimization in structural design : The model choice problem. Technical report, DTIC Document. (Cité en page 19.)

Durstewitz, D., Seamans, J. K., et Sejnowski, T. J. (2000). Neurocomputational models of working memory. *Nat. neurosci.*, 3 Suppl :1184–1191. (Cité en page 47.)

Earon, E. J. P. et Barfoot, T. D. (2000). From the Sea to the Sidewalk : The Evolution of Hexapod Walking Gaits by a Genetic Algorithm. *Evolvable Systems : From Biology to Hardware*, pages 51–60. (Cité en page 38.)

Ebner, M. et Zell, A. (1998). Evolving a behavior-based control architecture - From simulations to the real world. In *GECCO*, pages 13—-17. (Cité en page 38.)

Eggenberger, P., Ishiguro, A., et Tokura, S. (2000). Toward Seamless Transfer from Simulated to Real Worlds : A Dynamically—Rearranging Neural Network Approach. *Advances in Robot*, pages 1–18. (Cité en page 106.)

Eguíluz, V. M., Chialvo, D. R., Cecchi, G. a., Baliki, M., et Apkarian, a. V. (2005). Scale-Free Brain Functional Networks. *Physical Review Letters*, 94(1) :1–4. (Cité en page 35.)

Eiben, A. E. et Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35(1) :35–50. (Cité en page 15.)

Eiben, A. E. et Smith, J. E. (2003). *Introduction to evolutionary computing*. Springer. (Cité en pages 6 et 11.)

Elman, J. (1990). Finding structure in time. *Cognitive science*, 14(2) :179–211. (Cité en pages 31, 33, 49, 50, 53, 65 et 82.)

Everson, R. et Fieldsend, J. (2002). Full elite sets for multi-objective optimisation. *Adaptive Computing in*, (Acdm) :1–12. (Cité en page 25.)

Filliat, D., Kodjabachian, J., et Meyer, J.-A. (1999). Incremental evolution of neural controllers for navigation in a 6-legged robot. *on Artificial Life and Robots*. (Cité en page 38.)

Floreano, D. (1994). Automatic creation of an autonomous agent : Genetic evolution of a neural-network driven robot. *From animals to animats*. (Cité en page 32.)

Floreano, D. et Mattiussi, C. (2008). *Bio-inspired artificial intelligence : Theories, methods, and technologies*. (Cité en pages 28 et 29.)

Floreano, D. et Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *Systems, Man, and Cybernetics, Part*, 26(3) :396–407. (Cité en pages 6, 7, 38 et 39.)

Floreano, D. et Mondada, F. (1998). Evolutionary neurocontrollers for autonomous mobile robots. *Neural networks : the official journal of the International Neural Network Society*, 11(7-8) :1461–1478. (Cité en pages 102 et 106.)

Floreano, D. et Nolfi, S. (2000). *Evolutionary Robotics : The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Book. (Cité en pages 6, 37 et 106.)

Floreano, D. et Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural networks : the official journal of the International Neural Network Society*, 13(4-5) :431–43. (Cité en page 38.)

Floreano, D. et Urzelai, J. (2001). Evolution of plastic control networks. *Auton. Robot.*, 11(3) :311–317. (Cité en pages 102 et 106.)

Fogel, D. B. (1991). *System Identification through Simulated Evolution : A Machine Learning Approach to Modeling*. Ginn Press. (Cité en page 13.)

Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4 :14–19. (Cité en page 13.)

Fonseca, C. M. et Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization : Formulation : Discussion and Generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (Cité en pages 20 et 26.)

Fonseca, C. M., Paquete, L., et López-Ibáñez, M. (2006). An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *Congress on Evolutionary Computation (CEC 2006)*, pages 1157 —-1163. (Cité en page 25.)

Fourman, M. P. (1985). Compaction of Symbolic Layout Using Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 141–153, Hillsdale, NJ, USA. L. Erlbaum Associates Inc. (Cité en page 20.)

Fox, R., Lehmkuhle, S. W., et Westendorf, D. H. (1976). Falcon visual acuity. *Science*. (Cité en page 6.)

Fraser, A. S. (1957). Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction. *Australian Journal of Biological Sciences*, 10 :484–491. (Cité en page 13.)

Fukushima, K. (1975). Cognitron : a self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4) :121–136. (Cité en page 31.)

Funahashi, S., Bruce, C. J., et Goldman-Rakic, P. S. (1989). Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of Neurophysiology*, 61(2) :331–349. (Cité en page 47.)

Fuster, J. M. (1984). Behavioral electrophysiology of the prefrontal cortex. *Trends in Neurosciences*, 7(11) :408–414. (Cité en page 61.)

Fuster, J. M. (2008). *The prefrontal cortex*. Academic Press. (Cité en page 47.)

Gagné, C. et Schoenauer, M. (2006). Genetic programming, validation sets, and parsimony pressure. *Genetic Programming*. (Cité en page 104.)

Gallagher, J. (1999). Evolution and analysis of dynamical neural networks for agents integrating vision, locomotion, and short-term memory. In *Proceedings of the Genetic and Evolutionary*. (Cité en pages 32 et 40.)

Gallagher, J., Beer, R. D., Espenschied, K., et Quinn, R. (1996). Application of evolved locomotion controllers to a hexapod robot. *Robotics and Autonomous Systems*, 19 :95–103. (Cité en page 50.)

García-Pedrajas, N., Ortiz-Boyer, D., et Hervás-Martínez, C. (2006). An alternative approach for neural network evolution with a genetic algorithm : Crossover by combinatorial optimization. *Neural Networks*, 19(4) :514–528. (Cité en page 137.)

Gershenson, C. (2004). Cognitive paradigms : which one is the best ? *Cognitive Systems Research*, 5(2) :135–156. (Cité en page 5.)

Gerstner, W. et Kistler, W. (2002). *Spiking Neuron Models : An Introduction*. Cambridge University Press, New York, NY, USA. (Cité en page 30.)

Gigliotta, O. et Nolfi, S. (2008). On the coupling between agent internal and agent/environmental dynamics : Development of spatial representations in evolving autonomous robots. *Adaptive Behavior*, 16(2-3) :148–165. (Cité en page 136.)

Gigliotta, O., Pezzulo, G., et Nolfi, S. (2010). Emergence of an internal model in evolving robots subjected to sensory deprivation. *From Animals to Animats 11*, pages 575–586. (Cité en page 136.)

Gigliotta, O., Pezzulo, G., et Nolfi, S. (2011). Evolution of a predictive internal model in an embodied and situated agent. *Theory in biosciences*, 130(4) :259–276. (Cité en page 136.)

Girard, B., Tabareau, N., Pham, Q. C., Berthoz, A., et Slotine, J. J. (2008). Where neuroscience and dynamic system theory meet autonomous robotics : a contracting basal ganglia model for action selection. *Neural Networks*, 21(4) :628–641. (Cité en pages 29, 36, 53, 82 et 137.)

Gisiger, T., Kerszberg, M., et Changeux, J. P. (2005). Acquisition and Performance of Delayed-response Tasks : a Neural Network Model. *Cereb. Cortex*, 15(5) :489–506. (Cité en pages 61 et 63.)

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. (Cité en pages 8, 13 et 20.)

Goldberg, D. E. et Richardson, J. (1987). Genetic algorithms with sharing for multi-modal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. (Cité en page 79.)

Goldman-Rakic, P. S. (1987). Circuitry of primate prefrontal cortex and regulation of behavior by representational memory. *Comprehensive Physiology*, pages 373–417. (Cité en page 47.)

Gomez, F. (2005). Co-evolving recurrent neurons learn deep memory POMDPs. *Proc. of GECCO 2005*, pages 491–498. (Cité en page 50.)

Gomez, F. et Miikkulainen, R. (1996). Incremental Evolution of Complex General Behavior. Technical Report AI96-248. (Cité en page 77.)

Gomez, F. et Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4) :317–342. (Cité en page 76.)

Gomez, F. J. (2009). Sustaining diversity using behavioral information distance. In *Proceedings of GECCO'09*, pages 113–120. ACM. (Cité en page 109.)

Gomi, T. et Ide, K. (1998). Evolution of gaits of a legged robot. *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)*, pages 159–164. (Cité en page 38.)

Graf, P. et Schacter, D. L. (1985). Implicit and explicit memory for new associations in normal and amnesic subjects. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 11(3) :501. (Cité en page 46.)

Gro\csan, C. et Oltean, M. (2004). Improving the performance of evolutionary algorithms for the multiobjective 0/1 knapsack probl em using \varepsilon-dominance. *Computational Science-ICCS 2004*, pages 674–677. (Cité en page 23.)

Groß, R., Bonani, M., Mondada, F., et Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *Robotics, IEEE Transactions on*, 22(6) :1115–1130. (Cité en page 7.)

Gruau, F. (1993). Genetic synthesis of modular neural networks. In *Proceedings of the 5th International Conference on Genetic Algorithms*. (Cité en page 33.)

Gruau, F. (1994). Automatic definition of modular neural networks. *Adaptive Behavior*, pages 1–44. (Cité en pages 35 et 36.)

Gruau, F., Whitley, D., et Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 81–89. MIT Press. (Cité en page 33.)

Gurney, K., Prescott, T. J., et Redgrave, P. (2001a). A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biological cybernetics*, 84(6) :401–410. (Cité en page 137.)

Gurney, K., Prescott, T. J., et Redgrave, P. (2001b). A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. *Biological cybernetics*, 84(6) :411–423. (Cité en page 137.)

Haflidason, S. et Neville, R. (2009). On the significance of the permutation problem in neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 787–794. ACM. (Cité en page 137.)

Hagmann, P., Cammoun, L., Gigandet, X., Gerhard, S., Ellen Grant, P., Wedeen, V., Meuli, R., Thiran, J.-P., Honey, C. J., et Sporns, O. (2010). MR connectomics : Principles and challenges. *Journal of neuroscience methods*. (Cité en page 138.)

Haldane, J. B. S. (1932). The causes of evolution. (Cité en page 12.)

Hamdi, F. A. et Whitteridge, D. (1954). The representation of the retina on the optic tectum of the pigeon. *Experimental Physiology*, 39(2) :111–119. (Cité en page 6.)

Hancock, P. J. B. (1992). Genetic algorithms and permutation problems : A comparison of recombination operators for neural net structure specification. In *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pages 108–122. IEEE. (Cité en page 137.)

Hansen, N., Auger, A., Finck, S., et Ros, R. (2010). Real-parameter black-box optimization benchmarking 2010 : Experimental setup. (Cité en pages 17 et 26.)

Hansen, N. et Koumoutsakos, P. (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11 :1–18. (Cité en pages 16 et 26.)

Hansen, N. et Ostermeier, a. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2) :159–95. (Cité en pages 16 et 26.)

Hansen, T. F. (2003). Is modularity necessary for evolvability ? Remarks on the relationship between pleiotropy and evolvability. *Bio Systems*, 69(2-3) :83–94. (Cité en page 35.)

Harnad, S. (1990). The symbol grounding problem. *Physica D : Nonlinear Phenomena*, 42(1) :335–346. (Cité en page 5.)

Hartland, C., Bredeche, N., et Sebag, M. (2009). Memory-enhanced evolutionary robotics : the echo state network approach. In *In proc. of IEEE-CEC'09*, number section IV, pages 2788–2795. (Cité en pages 32 et 41.)

Hartwell, L. H., Hopfield, J. J., Leibler, S., et Murray, A. W. (1999). From molecular to modular cell biology. *Nature*, 402(6761 Suppl) :C47—-52. (Cité en page 35.)

Harvey, I. et Husbands, P. (1992). Evolutionary robotics. In *Proceedings of IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pages 1–4. (Cité en pages 6 et 37.)

Harvey, I., Husbands, P., et Cliff, D. (1994). *Seeing The Light : Artificial Evolution A Real Vision*. Number April. School of Cognitive and Computing Sciences, University of Sussex. (Cité en pages 38, 50, 76 et 77.)

Harvey, I., Husbands, P., Cliff, D., et Thompson, A. (1997). Evolutionary robotics : the Sussex approach. *Robotics and autonomous systems*, 20 :205–224. (Cité en page 37.)

Haykin, S. (1998). *Neural Networks : A Comprehensive Foundation, 2nd edition*. Prentice Hall PTR. (Cité en page 28.)

Hebb, D. (1949). *The Organization of Behavior : A Neuropsychological Theory*. Psychology Press. (Cité en page 47.)

Heidrich-Meisner, V. et Igel, C. (2009). Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8. (Cité en page 108.)

Hemker, T. et Sakamoto, H. (2006). Hardware-in-the-Loop Optimization of the Walking Speed of a Humanoid Robot. In *Proc of CLAWAR 2006*, pages 614–623. (Cité en page 108.)

Hemker, T., Stelzer, M., Von Stryk, O., et Sakamoto, H. (2009). Efficient walking speed optimization of a humanoid robot. *The International Journal of Robotics Research*, 28(2) :303. (Cité en page 108.)

Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8) :1735–1780. (Cité en page 49.)

Hodgkin, A. L. et Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4) :500–544. (Cité en page 30.)

Hoffmann, F. et Pfister, G. (1996). Evolutionary learning of a fuzzy control rule base for an autonomous vehicle. *Management of Uncertainty in Knowledge-Based*. (Cité en pages 6 et 38.)

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, volume Ann Arbor. University of Michigan Press. (Cité en page 13.)

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8) :2554–8. (Cité en pages 31 et 48.)

Horn, J., Nafpliotis, N., et Goldberg, D. (1994). A niched Pareto genetic algorithm for multiobjective optmization. In *First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Volume 1*. (Cité en page 21.)

Hornby, G. S. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1729–1736. AcM. (Cité en page 35.)

Hornby, G. S. et Lipson, H. (2001). Evolution of generative design systems for modular physical robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, pages 4146—-4151. (Cité en pages 35 et 38.)

Hornby, G. S., Lohn, J., et Linden, D. S. (2011). Computer-automated evolution of an x-band antenna for nasa's space technology 5 mission. *Evolutionary Computation*, (x). (Cité en page 13.)

Hornby, G. S. et Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3) :223–246. (Cité en pages 35 et 36.)

Hornby, G. S., Takamura, S., Yamamoto, T., et Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Transactions on Robotics*, 21(3) :402–410. (Cité en page 38.)

Hornby, G. S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., et Fujita, M. (2000). Evolving robust gaits with AIBO. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 3040–3045. IEEE. (Cité en page 7.)

Houdé, O., Kayser, D., Koenig, O., Proust, J., et Rastier, F. (1998). *Vocabulaire de sciences cognitives*. Presses Univ. de France. (Cité en page 44.)

Huband, S., Hingston, P., et While, L. (2003). An evolution strategy with probabilistic mutation for multi-objective optimisation. *Evolutionary*. (Cité en page 25.)

Hume, D. (1739). *Traité de la nature humaine*. (Cité en page 44.)

Humphries, M. D., Gurney, K., et Prescott, T. J. (2006). The brainstem reticular formation is a small-world, not scale-free, network. *Proceedings. Biological sciences / The Royal Society*, 273(1585) :503–11. (Cité en page 138.)

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive science*, 19(3) :265–288. (Cité en page 8.)

Igel, C., Hansen, N., et Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1) :1–28. (Cité en pages 25 et 26.)

Ijiri, Y. (1965). *Management goals and accounting for control*. North-Holland Publishing Company. (Cité en page 19.)

Ishiguro, A., Tokura, S., Kondo, T., Uchikawa, Y., et Eggenberger, P. (1999). Reduction of the gap between simulated and real environments in evolutionary robotics : a dynamically-rearranging neural network approach. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 3, pages 239–244. IEEE. (Cité en page 38.)

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Networks*, pages 1569–1572. (Cité en page 30.)

Jaeger, H. (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik. (Cité en pages 32, 53 et 54.)

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2) :325–368. (Cité en pages 33, 40, 102, 105, 106 et 107.)

Jakobi, N. (1998). Running across the reality gap : Octopod locomotion evolved in a minimal simulation. *Evolutionary Robotics*. (Cité en pages 38, 77, 102, 103, 105, 107 et 113.)

Jakobi, N., Husbands, P., et Harvey, I. (1995). Noise and the reality gap : The use of simulation in evolutionary robotics. *Advances in Artificial Life*, pages 704–720. (Cité en pages 102, 105 et 106.)

Jansen, T. et Wegener, I. (1999). On the analysis of evolutionary algorithms - a proof that crossover really can help. *Algorithms-ESA'99*, page 700. (Cité en page 15.)

Jensen, M. (2003). Reducing the run-time complexity of multiobjective EAs : The NSGA-II and other algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(5) :503–515. (Cité en pages 23 et 26.)

Jin, Y. (2005a). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1) :3–12. (Cité en page 108.)

Jin, Y. (2005b). Evolutionary Optimization in Uncertain Environments. *IEEE Transactions on Evolutionary Computation*, 2610(1) :264–317. (Cité en page 37.)

Jin, Y. et Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3) :303–317. (Cité en page 105.)

Jin, Y., Olhofer, M., et Sendhoff, B. (2001). Dynamic weighted aggregation for evolutionary multi-objective optimization : Why does it work and how. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049. (Cité en page 18.)

Joel, D., Weiner, I., et Feldon, J. (1997). Electrolytic lesions of the medial prefrontal cortex in rats disrupt performance on an analog of the Wisconsin Card Sorting Test, but do not disrupt latent inhibition : implications for animal models of schizophrenia. *Behavioural Brain Research*, 85(2) :187–201. (Cité en page 41.)

Jordan, M. I. (1986). Serial order : A parallel distributed processing approach. *Advances in Connectionist Theory Speech*, 121(ICS-8604) :471–495. (Cité en pages 31, 49 et 50.)

Jun, J. K., Miller, P., Hernandez, A., Zainos, A., Lemus, L., Brody, C., et Romo, R. (2010). Heterogenous population coding of a short-term memory and decision task. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30(3) :916–29. (Cité en page 61.)

Kaas, J. H. (1997). Topographic maps are fundamental to sensory processing. *Brain research bulletin*, 44(2) :107–112. (Cité en page 137.)

Kamio, S. et Iba, H. (2005). Adaptation Technique for Integrating Genetic Programming and Reinforcement Learning for Real Robots. *IEEE Transactions on Evolutionary Computation*, 9(3) :318–333. (Cité en page 38.)

Kashtan, N. et Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39) :13773–8. (Cité en pages 35 et 138.)

Kaski, S., Honkela, T., Lagus, K., et Kohonen, T. (1998). WEBSOM–self-organizing maps of document collections. *Neurocomputing*, 21(1) :101–117. (Cité en page 31.)

Kim, D. (2004). Evolving internal memory for T-maze tasks in noisy environments. *Connection Science*, 16(3) :183–210. (Cité en page 40.)

Kim, I. Y. et De Weck, O. L. (2006). Adaptive weighted sum method for multiobjective optimization : a new method for Pareto front generation. *Structural and Multidisciplinary Optimization*, 31(2) :105–116. (Cité en page 18.)

Knowles, J. D. (2002). *Local-search and hybrid evolutionary algorithms for Pareto optimization*. PhD thesis. (Cité en pages 24, 25, 26 et 135.)

Knowles, J. D. et Corne, D. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary computation*, 8 :149—-172. (Cité en pages 21 et 26.)

Knowles, J. D., Watson, R., et Corne, D. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *EMO'01 : Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 269–283. Springer. (Cité en page 84.)

Kodjabachian, J. et Meyer, J.-A. (1997). Evolution and development of Neural Networks Controlling Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks*, 9(5) :796–812. (Cité en pages 7, 76 et 77.)

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. (Cité en page 109.)

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1) :59–69. (Cité en page 31.)

Kohonen, T. (2001). *Self-organizing maps*, volume 30. Springer Verlag. (Cité en page 31.)

Kondo, T. (2007). Evolutionary design and behavior analysis of neuromodulatory neural networks for mobile robots control. *Applied soft computing*, 7(1) :189–202. (Cité en page 106.)

Kondo, T., Ishiguro, A., Tokura, S., Uchikawa, Y., et Eggenberger, P. (1999). Realization of robust controllers in evolutionary robotics : a dynamically-rearranging neural network approach. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, pages 366–373. (Cité en pages 102, 105, 106 et 107.)

Koos, S., Mouret, J.-B., et Doncieux, S. (2012). The Transferability Approach : Crossing the Reality Gap in Evolutionary Robotics. *IEEE Transactions on Evolutionary Computation*, 1 :1. (Cité en pages 40 et 109.)

Kowaliw, T., Grogono, P., et Kharma, N. (2007). Environment as a spatial constraint on the growth of structural form. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, pages 1037–1044, New York, New York, USA. ACM Press. (Cité en page 35.)

Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, volume 229. MIT Press. (Cité en pages 6 et 13.)

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., et Lanza, G. (2003). *Genetic programming IV : Routine human-competitive machine intelligence*. Springer. (Cité en page 13.)

Lahanas, M., Baltas, D., et Zamboglou, N. (2003). A hybrid evolutionary algorithm for multi-objective anatomy-based dose optimization in high-dose-rate brachytherapy. *Physics in medicine and biology*, 48 :399. (Cité en page 27.)

Lahanas, M., Karouzakis, K., Giannouli, S., Mould, R., et Baltas, D. (2004). Inverse planning in brachytherapy : Radium to high dose rate 192 iridium afterloading. *Nowotwory Journal of Oncology*, 54(6) :547–554. (Cité en page 27.)

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traites comme une polarization. *J Physiol (Paris)*, pages 622–635. (Cité en page 30.)

Laumanns, M., Thiele, L., Deb, K., et Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3) :263–82. (Cité en pages 23, 24 et 26.)

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press. (Cité en page 4.)

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4) :541–551. (Cité en page 31.)

Lee, W., Hallam, J., et Lund, H. H. (1997). Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. *Evolutionary Computation, 1997*. (Cité en page 38.)

Lehman, J. (2008). Exploiting open-endedness to solve problems through the search for novelty. *Proceedings of Alife XI*, (Alife Xi) :71–80. (Cité en pages 78, 80 et 85.)

Lehman, J. et Stanley, K. O. (2010). Abandoning Objectives : Evolution through the Search for Novelty Alone. *Evolutionary computation*, 19(2) :189–223. (Cité en pages 78, 80 et 106.)

Linaker, F. et Jacobsson, H. (2001). Mobile robot learning of delayed response tasks through event extraction : A solution to the road sign problem and beyond. In *International Joint Conference On Artificial Intelligence*, volume 17, pages 777–782. (Cité en page 40.)

Lisman, J. E., Fellous, J. M., Wang, X. J., et Others (1998). A role for NMDA-receptor channels in working memory. *Nat Neurosci*, 1(4) :273–275. (Cité en page 48.)

Liu, J. (1999). Learning coordinated maneuvers in complex environments : a sumo experiment. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, pages 343–349. (Cité en page 38.)

Lohn, J. D., Hornby, G. S., et Linden, D. S. (2004). An evolved antenna for deployment on nasa's space technology 5 mission. In *Genetic Programming Theory and Practice Workshop*. (Cité en page 13.)

Lorenz, K. (2003). *The foundations of ethology*. Springer. (Cité en page 8.)

Lund, H. H. et Miglino, O. (1996). From simulated to real robots. In *Proceedings of IEEE International Conference on Evolutionary Computation*. (Cité en pages 38 et 39.)

Maass, W. (1997). Networks of spiking neurons : the third generation of neural network models. *Neural networks*. (Cité en page 30.)

Machens, C. K. et Brody, C. (2008). Design of continuous attractor networks with monotonic tuning using a symmetry principle. *Neural Computation*, 20(2) :452–485. (Cité en page 48.)

Machens, C. K., Romo, R., et Brody, C. (2005). Flexible control of mutual inhibition : a neural model of two-interval discrimination. *Science*, 307(5712) :1121. (Cité en pages 61 et 63.)

Macinnes, I. et Paolo, E. (2003). Crawling Out of the Simulation : Evolving Real Robot Morphologies Using Cheap, Reusable Modules. *Neuroscience*. (Cité en page 38.)

Maes, J. H. R., Bouwman, B. M., et Vossen, J. M. H. (2001). Effects of d-amphetamine on the performance of rats in an animal analogue of the A-X Continuous Performance Test. *Journal of Psychopharmacology*, 15(1) :23–28. (Cité en page 64.)

Maniadakis, M. et Tani, J. (2009). Acquiring Rules for Rules : Neuro-Dynamical Systems Account for Meta-Cognition. *Adaptive Behavior*, 17(1) :58–80. (Cité en page 41.)

Maniadakis, M., Trahanias, P., et Tani, J. (2010). Self-organized executive control functions. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. (Cité en pages 33 et 41.)

Marder, E., Abbott, L. F., Turrigiano, G. G., Liu, Z., et Golowasch, J. (1996). Memory from the dynamics of intrinsic membrane currents. *Proceedings of the National Academy of Sciences*, 93(24) :13481. (Cité en page 48.)

Maron, O. et Moore, A. W. (1994). Hoeffding races : Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263. (Cité en page 108.)

Marstaller, L., Hintze, A., et Adami, C. (2012). Cognitive systems evolve complex representations for adaptive behavior. *arXiv preprint arXiv :1206.5771*. (Cité en page 135.)

Mataric, M. et Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and autonomous systems*, 19 :67–83. (Cité en page 37.)

Matellán, V., Fernandez, C., et Molina, J. (1998). Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*. (Cité en page 38.)

Mattiussi, C. (2005). *Evolutionary Synthesis of Analog Networks*. PhD thesis, École Polytechnique Fédérale de Lausanne. (Cité en pages 35 et 36.)

Mattiussi, C. et Floreano, D. (2007). Analog Genetic Encoding for the Evolution of Circuits and Networks. *IEEE Transactions on Evolutionary Computation*, 11(5) :596–607. (Cité en page 36.)

Mayr, E. (1942). *Systematics and the origin of species, from the viewpoint of a zoologist*. Harvard Univ Pr. (Cité en page 12.)

McCallum, A. (1996). Learning to use selective attention and short-term memory in sequential tasks. *From animals to animats*. (Cité en page 50.)

McCarthy, J., Minsky, M. L., Rochester, N., et Shannon, C. E. (1955). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI Magazine*, 27(4) :12. (Cité en page 5.)

McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4) :115–133. (Cité en pages 5, 28 et 48.)

Meeden, L. (1995). An Incremental Approach to Developing Intelligent Neural Network Controllers for Robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 26 :474–485. (Cité en page 41.)

Meeden, L. et Kumar, D. (1998). Trends in evolutionary robotics. *Soft computing for intelligent robotic*. (Cité en page 37.)

Mendel, G. (1865). Experiments in plant hybridization (1865). In *Read at the meetings of February 8th, and March 8th*. (Cité en page 12.)

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., et Montesano, L. (2010). The iCub humanoid robot : an open-systems platform for research in cognitive development. *Neural networks : the official journal of the International Neural Network Society*, 23(8-9) :1125–34. (Cité en page 66.)

Metta, G., Sandini, G., Vernon, D., Natale, L., et Nori, F. (2008). The iCub humanoid robot. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems - PerMIS '08*, page 50, New York, New York, USA. ACM Press. (Cité en page 3.)

Meyer, J.-A. et Guillot, A. (1990). Simulation of adaptive behavior in animats : review and prospect. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 2–14, Cambridge, MA, USA. MIT Press. (Cité en page 2.)

Meyer, J.-A., Husbands, P., et Harvey, I. (1998). Evolutionary robotics : A survey of applications and problems. *Evolutionary Robotics*, pages 1–22. (Cité en pages 6 et 37.)

Michalewicz, Z. (1994). *Genetic Algorithms Plus Data Structures Equals Evolution Programs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition. (Cité en page 13.)

Michalewicz, Z. (2004). *How to solve it : modern heuristics*. (Cité en page 12.)

Miglino, O., Denaro, D., Tascini, G., et Parisi, D. (1998). Detour Behavior in Evolving Robots : Are Internal Representations Necessary ? In *Evolutionary Robotics*, pages 59—-70. (Cité en pages 38, 78 et 106.)

Miglino, O., Lund, H. H., et Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial life*, 2(4) :417–434. (Cité en pages 102, 105 et 106.)

Miglino, O., Nafasi, K., et Taylor, C. E. (1994). Selection for wandering behavior in a small robot. *Artif. Life*, 2(1) :101–116. (Cité en page 33.)

Miller, B. L. et Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Evolutionary Computation*, 4 :113—-131. (Cité en page 16.)

Miller, E. K., Erickson, C. A., et Desimone, R. (1996). Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *J. Neurosci.*, 16(16) :5154–67. (Cité en page 47.)

Miller, G. a. (1956). The magical number seven, plus or minus two : some limits on our capacity for processing information. 1956. *Psychological review*, 101(2) :343–52. (Cité en page 46.)

Miller, P. et Wang, X. J. (2006). Inhibitory control by an integral feedback signal in prefrontal cortex : a model of discrimination between sequential stimuli. Supporting Information. *Proc. Natl. Acad. Sci. U.S.A.*, 103(1) :201–6. (Cité en pages 61 et 63.)

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., et Alon, U. (2002). Network motifs : simple building blocks of complex networks. *Science (New York, N.Y.)*, 298(5594) :824–7. (Cité en page 138.)

Minami, T. et Inui, T. (2001). A Neural Network Model of Working Memory. *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, pages 126–133. (Cité en page 63.)

Mitchell, J. et Zipser, D. (2003). Sequential memory-guided saccades and target selection. *Vision Res.*, 43(25) :2669–2695. (Cité en pages 61 et 63.)

Mitchell, T. et Lin, L. (1993). Reinforcement Learning With Hidden States. In *Proceedings of the second international conference on From animals to animats*, pages 271—-281. (Cité en page 50.)

Miyake, A. et Shah, P. (1999). Models of working memory. *Models of working memory : Mechanisms of active maintenance and executive control*, pages 1—-27. (Cité en page 47.)

Moody, S. L., Wise, S. P., Pellegrino, G., et Zipser, D. (1998). model that accounts for activity in primate frontal cortex during a delayed matching-to-sample task. *The Journal of Neuroscience*, 18(1) :399–410. (Cité en pages 61 et 63.)

Moriguchi, H. et Honiden, S. (2011). Sustaining Behavioral Diversity in NEAT. In *Proceedings of the 12th Annual conference on Genetic and evolutionary computation - GECCO '11*. (Cité en page 34.)

Moss, C. F. et Sinha, S. R. (2003). Neurobiology of echolocation in bats. *Current opinion in Neurobiology*, 13(6) :751–758. (Cité en page 6.)

Mouret, J.-B. (2008). *Pressions sélectives multiples pour l'évolution de réseaux de neurones destinés à la robotique*. Thèse de doctorat, Université Pierre et Marie Curie (UPMC), 4 place jussieu, 75005 PARIS. (Cité en page 35.)

Mouret, J.-B. (2011). Novelty-based Multiobjectivization. In *New Horizons in Evolutionary Robotics : Extended Contributions from the 2009 EvoDeRob Workshop*, volume 341 of *Studies in Computational Intelligence*, pages 139–154. Springer. (Cité en pages 80 et 81.)

Mouret, J.-B. et Doncieux, S. (2006). Incremental evolution of target-following neuro-controllers for flapping-wing animats. In *From Animals to Animats 9*, pages 606–618. (Cité en pages 7 et 34.)

Mouret, J.-B. et Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. *From Animals to Animats 10*, pages 210–219. (Cité en pages 34, 35 et 81.)

Mouret, J.-B. et Doncieux, S. (2009a). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*. (Cité en pages 53, 81, 84 et 115.)

Mouret, J.-B. et Doncieux, S. (2009b). Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, pages 627–635. (Cité en pages 53 et 81.)

Mouret, J.-B. et Doncieux, S. (2010). Sferes v2 : Evolvin ' in the Multi-Core World. *Proc. of CEC 2010*, (2). (Cité en pages 55 et 82.)

Mouret, J.-B. et Doncieux, S. (2012). Encouraging Behavioral Diversity in Evolutionary Robotics : an Empirical Study. *Evolutionary Computation*. (Cité en pages 7, 15, 34, 39, 78, 81, 85, 102, 105, 106, 107, 109, 112, 114, 115 et 116.)

Mouret, J. B., Doncieux, S., et Girard, B. (2010). Importing the Computational Neuroscience Toolbox into Neuro-Evolution — Application to Basal Ganglia Track : Generative and Developmental Systems. *Neuroscience*. (Cité en pages 36, 137 et 139.)

Mucientes, M., Alcalá-Fdez, J., Alcalá, R., et Casillas, J. (2010). A case study for learning behaviors in mobile robotics by evolutionary fuzzy systems. *Expert Systems with Applications*, 37(2) :1471–1493. (Cité en page 77.)

Murphy, R. R. (2004). Human-robot interaction in rescue robotics. *IEEE Systems, Man and Cybernetics Part C : Applications and Reviews, special issue on Human-Robot Interaction*. (Cité en page 3.)

Nakamura, H., Ishiguro, A., et Uchikawa, Y. (2000). Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00*, pages 158–165. (Cité en pages 38, 78 et 106.)

Nandasana, A. D., Ray, A. K., et Gupta, S. K. (2003). Applications of the non-dominated sorting genetic algorithm (NSGA) in chemical reaction engineering. *International Journal of Chemical Reactor Engineering*. (Cité en page 27.)

Nehmzow, U. (2002). Physically embedded genetic algorithm learning in multi-robot scenarios : The PEGA algorithm. *Lund University Cognitive Studies*. (Cité en page 38.)

Nelson, A. L. (2004). Maze exploration behaviors using an integrated evolutionary robotics environment. *Robotics and Autonomous Systems*, 46(3) :159–173. (Cité en page 38.)

Nelson, A. L., Barlow, G., et Doi Tsidis, L. (2009). Fitness functions in evolutionary robotics : A survey and analysis. *Robotics and Autonomous Systems*, 57(4) :345–370. (Cité en pages 7, 37, 38, 39, 51, 52, 76, 77, 83 et 84.)

Nelson, A. L. et Grant, E. (2007). Aggregate selection in evolutionary robotics. In *Mobile Robots : The Evolutionary Approach*, volume 50, pages 63–87. (Cité en page 37.)

Nilsson, N. J. (1984). Shakey the Robot. (Cité en page 2.)

Nof, S. Y. (1999). *Handbook of Industrial Robotics*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition. (Cité en page 1.)

Nolfi, S. (1996). Evolving non-Trivial Behaviors on Real Robots : a garbage collecting robot. *Robotics and Autonomous Systems*. (Cité en page 38.)

Nolfi, S. (2001). Evolving robots able to integrate sensory-motor information over time. *Theory in Biosciences*, 120(3-4) :287–310. (Cité en pages 8, 50, 51, 52 et 134.)

Nolfi, S. (2002a). Evolving robots able to self-localize in the environment : the importance of viewing cognition as the result of processes occurring at different time-scales. *Connection Science*, 14(3) :231–244. (Cité en page 40.)

Nolfi, S. (2002b). Power and the limits of reactive agents. *Neurocomputing*, 42(1) :119–145. (Cité en pages 7 et 8.)

Nolfi, S. (2005). Categories formation in self-organizing embodied agents. *Handbook of categorization in cognitive science*, pages 869—-889. (Cité en page 136.)

Nolfi, S., Floreano, D., Miglino, O., et Mondada, F. (1994). How to evolve autonomous robots : Different approaches in evolutionary robotics. *Artificial Life IV*. (Cité en page 50.)

Nolfi, S. et Parisi, D. (1995). Evolving non-Trivial Behaviors on Real Robots : an Autonomous Robot that Picks up Objects. *Topics in Artificial Intelligence*. (Cité en pages 76 et 77.)

Nolfi, S. et Parisi, D. (1996). Learning to adapt to changing environments in evolving neural networks. Technical Report 1, Department of Neural Systems and Artificial Life, Rome. (Cité en page 105.)

Nolfi, S. et Tani, J. (1999). Extracting regularities in space and time through a cascade of prediction networks : The case of a mobile robot navigating in a structured environment. *Connection Science*, 11(2) :125–148. (Cité en page 41.)

Nordin, P. (1998). Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, 25(1-2) :105–116. (Cité en page 38.)

Okura, M., Matsumoto, A., Ikeda, H., et Murase, K. (2003). Artificial Evolution of FPGA that controls a Miniature Mobile Robot Khepera. *Artificial Intelligence*, pages 2521–2526. (Cité en page 38.)

Ollion, C., Pinville, T., et Doncieux, S. (2012). With a little help from selection pressures : evolution of memory in robot controllers. In *Proc. Alife XIII*, pages 1–8. (Cité en page 101.)

Olton, D. S. et Samuelson, R. J. (1976). Remembrance of places passed : Spatial memory in rats. *Journal of Experimental Psychology : Animal Behavior Processes*, 2(2) :97–116. (Cité en page 60.)

O'Regan, J. K. (1992). Solving the "real" mysteries of visual perception : the world as an outside memory. *Canadian Journal of Psychology*, 46(3) :461. (Cité en page 8.)

O'Regan, J. K., Noë, A., et Others (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5) :939–972. (Cité en page 8.)

O'Reilly, R. C. et Frank, M. J. (2006). Making working memory work : a computational model of learning in the prefrontal cortex and basal ganglia. *Neural comput.*, 18(2) :283–328. (Cité en page 63.)

Paine, R. (2005). How hierarchical control self-organizes in artificial adaptive systems. *Adaptive Behavior*. (Cité en page 33.)

Panait, L. (2003). Methods for evolving robust programs. *Proc. of GECCO 2003*, pages 213–221. (Cité en page 113.)

Pareto, V. (1897). *Cours d'Economie Politique*, volume 2. F. Rouge & Cie., Lausanne, Switzerland. (Cité en page 17.)

Parker, G. (2001). The incremental evolution of gaits for hexapod robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1114–1121. (Cité en page 76.)

Parker, G. et Georgescu, R. (2006). Using cyclic genetic algorithms to evolve multiloop control programs. *Mechatronics and Automation,*. (Cité en pages 38 et 39.)

Pasemann, F., Steinmetz, U., Hülse, M., et B (2001). Evolving brain structures for robot control. *Bio-Inspired Applications of Connectionism*, pages 410—-417. (Cité en page 38.)

Pattacini, U., Nori, F., Natale, L., Metta, G., et Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1668–1674. (Cité en page 66.)

Perlstein, W. M., Dixit, N. K., Carter, C. S., Noll, D. C., et Cohen, J. D. (2003). Prefrontal cortex dysfunction mediates deficits in working memory and prepotent responding in schizophrenia. *Biological psychiatry*, 53(1) :25–38. (Cité en page 64.)

Pfeifer, R. et Bongard, J. C. (2006). *How the body shapes the way we think : a new view of intelligence*. MIT press. (Cité en pages 4 et 5.)

Pfeifer, R., Iida, F., et Bongard, J. (2003). New robotics : design principles for intelligent systems. *Artificial life*, 11(1-2) :99–120. (Cité en page 37.)

Pfeifer, R. et Scheier, C. (2001). *Understanding Intelligence*. MIT Press. (Cité en page 5.)

Piaget, J. et Inhelder, B. (1968). *Mémoire et intelligence*. Presses universitaires de France. (Cité en page 7.)

Pinville, T. et Doncieux, S. (2010). Automatic synthesis of working memory neural networks with neuroevolution methods. *Neurocomp 2010*. (Cité en pages 137 et 139.)

Pinville, T., Koos, S., Mouret, J.-B., et Doncieux, S. (2011). How to Promote Generalisation in Evolutionary Robotics : the ProGAb Approach. *Proc. of GECCO'11*. (Cité en page 101.)

Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning*, 7(2-3) :227–252. (Cité en pages 33 et 49.)

Powell, J. R. (1971). Genetic polymorphism in varied environments. *Science*, 174(4013) :1035–. (Cité en page 136.)

Pratihar, D. K. (2003). Evolutionary robotics — A review. *Sadhana*, 28(6) :999–1009. (Cité en page 37.)

Prusinkiewicz, P. et Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer-Verlag. (Cité en page 35.)

Pylyshyn, Z. W. (1984). *Computation and cognition : toward a foundation for cognitive science*. Massachusetts Institute of Technology, Cambridge, MA, USA. (Cité en page 5.)

Quinn, M., Smith, L., Mayley, G., et Husbands, P. (2001). Evolving Team Behaviour for Real Robots. In *EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics : The Legacy of W. Grey Walter, WGW*. (Cité en page 38.)

Ratle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. *Parallel Problem Solving from Nature—PPSN V*. (Cité en page 108.)

Ray, T. (2004). Applications of multi-objective evolutionary algorithms in engineering design. *Applications of multi-objective evolutionary algorithms*, 1 :29. (Cité en page 27.)

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. In *Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans.* Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants. (Cité en page 13.)

Rechenberg, I. (1973). *Evolutionsstrategie – Optimierung technisher Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, GER. (Cité en pages 13 et 21.)

Reed, P., Kollat, J. B., et Devireddy, V. K. (2007). Using interactive archives in evolutionary multiobjective optimization : A case study for long-term groundwater monitoring design. *Environmental Modelling & Software*, 22(5) :683–692. (Cité en page 27.)

Reed, S. K. (2011). *Cognition théories et applications*. De Boeck. (Cité en pages 4 et 7.)

Reynolds, C. W. (1993). An Evolved, Vision-Based Model of Obstacle Avoidance Behavior. In *Artificial Life III Proceedings*. (Cité en pages 102 et 105.)

Risi, S. (2011). Enhancing ES-HyperNEAT to Evolve More Complex Regular Neural Networks. In *Proceedings of the 2011 anual conference on Genetic and evolutionary computation : GECCO'11*. (Cité en page 36.)

Risi, S. et Stanley, K. O. (2010). Indirectly Encoding Neural Plasticity as a Pattern of Local Rules. *SAB*, pages 533–543. (Cité en page 36.)

Risi, S., Vanderbleek, S. D., Hughes, C. E., et Stanley, K. O. (2009). How novelty search escapes the deceptive trap of learning to learn. *GECCO*. (Cité en page 81.)

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(1958). (Cité en pages 5 et 31.)

Rosin, C. D. et Belew, R. K. (1997). New Methods for Competitive Coevolution. *Evolutionary Computation, IEEE Transactions on*, 5(1) :1—-29. (Cité en page 79.)

Rosvold, H. E., Mirsky, A. F., Sarason, I., Bransome Jr., E. D., et Beck, L. H. (1956). A continuous performance test of brain damage. *J. Consult. Clin. Psych.*, 20(5) :343–350. (Cité en page 61.)

Rougier, N. P. (2006). Dynamic neural field with local inhibition. *Biological cybernetics*, 94(3) :169–179. (Cité en page 137.)

Rougier, N. P. et Vitay, J. (2006). Emergence of attention within a neural population. *Neural Networks*, 19(5) :573–581. (Cité en page 137.)

Roxin, A., Riecke, H., et Solla, S. A. (2004). Self-sustained activity in a small-world network of excitable neurons. *Physical review letters*, 92(19) :198101. (Cité en page 138.)

Rubinov, M. et Sporns, O. (2010). Complex network measures of brain connectivity : Uses and interpretations. *NeuroImage*, 52(3) :1059–1069. (Cité en page 138.)

Ruffié, J. (1986). *Traité du vivant. Tome I.* Flammarion. (Cité en page 136.)

Rumelhart, D. E., Hinton, G. E., et Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088) :533–536. (Cité en page 31.)

Rumelhart, D. E., McClelland, J. L., Group, P. D. P. R., et Others (1988). *Parallel distributed processing*, volume 1. MIT press Cambridge. (Cité en page 5.)

Rumelhart, D. E. et Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive science*, 9(1) :75–112. (Cité en page 31.)

Rylatt, R. M. et Czarnecki, C. A. (2000). Embedding Connectionist Autonomous Agents in Time : The 'Road Sign Problem'. *Neural Processing Letters*, 12(2) :145–158. (Cité en pages 40, 51 et 69.)

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., et Fujimura, K. (2002). The intelligent ASIMO : system overview and integration. In *IEEERSJ International Conference on Intelligent Robots and System*, volume 3, pages 2478–2483. Ieee. (Cité en page 3.)

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University. (Cité en pages 19 et 20.)

Schraft, R. D. et Wanner, M.-C. (1993). The Aircraft Cleaning Robot "Skywash". *Industrial Robot : An International Journal*, 20(6) :21—-24. (Cité en page 3.)

Schultz, A., Grefenstette, J., et Adams, W. (1996). Robo-shepherd : Learning complex robotic behaviors, Robotics and Manufacturing : Recent Trends in Research and Application, Volume 6. *World*. (Cité en pages 38 et 39.)

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA. (Cité en page 13.)

Seok, H., Lee, K., et Zhang, B. (2000). An on-line learning method for object-locating robots using genetic programming on evolvable hardware. *Symposium on Artificial Life and Robotics*. (Cité en pages 38 et 39.)

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proc. of the ACM national conference*. (Cité en page 110.)

Shim, Y. S. et Husbands, P. (2007). Feathered flyer : integrating morphological computation and sensory reflexes into a physically simulated flapping-wing robot for robust flight manoeuvre. *Advances in Artificial Life*, pages 756–765. (Cité en page 7.)

Singh, H., Roman, C., Pizarro, O., et Eustice, R. (2005). Advances in high resolution imaging from underwater vehicles. *International Symposium of Robotics Research*. (Cité en page 3.)

Slocum, A. et Downey, D. C. (2000). Further experiments in the evolution of minimally cognitive behavior : From perceiving affordances to selective attention. *From animals to animats*. (Cité en pages 32 et 40.)

Smith, T., Husbands, P., Philippides, A., et O'Shea, M. (2002). Neuronal plasticity and temporal adaptivity : GasNet robot control networks. *Adaptive Behavior*, 10(3-4) :161. (Cité en page 106.)

Sporns, O., Tononi, G., et Kötter, R. (2005). The human connectome : A structural description of the human brain. *PLoS computational biology*, 1(4) :e42. (Cité en page 138.)

Sprinkhuizen-kuyper, I., Kortmann, R., et Postman, E. (2000). Fitness functions for evolving box-pushing behaviour. In *Proceedings of the Twelfth Belgium–Netherlands Artificial Intelligence Conference*, pages 275—-282. (Cité en page 38.)

Srinivas, N. et Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3) :221–248. (Cité en page 20.)

Stanfill, C. et Waltz, D. (1987). Toward memory-based reasoning. *Communications of the ACM*, 29(12) :1213–1228. (Cité en page 7.)

Stanley, K. O. (2006). Exploiting regularity without development. *Proceedings of the AAAI Fall Symposium on Developmental Systems.* (Cité en page 36.)

Stanley, K. O. (2007). Compositional pattern producing networks : A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2) :131–162. (Cité en page 36.)

Stanley, K. O., Ambrosio, D. D., et Gauci, J. (2009). A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks. *Artif. Life*, 15(2) :1–39. (Cité en page 36.)

Stanley, K. O. et Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2) :99–127. (Cité en pages 15, 33, 34, 53, 79, 80 et 137.)

Stanley, K. O. et Miikkulainen, R. (2003). Evolving adaptive neural networks with and without adaptive synapses. *GECCO*. (Cité en page 34.)

Suchorzewski, M. (2011). A novel generative encoding for evolving modular, regular and scalable networks. *GECCO'11*, pages 1523–1530. (Cité en page 36.)

Sutton, R. et Barto, A. (1998). *Reinforcement learning : An introduction*. Cambridge Univ Press. (Cité en page 108.)

Tani, J. et Nolfi, S. (1999). Learning to perceive the world as articulated : an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7) :1131–1141. (Cité en page 41.)

Teo, J. et Abbass, H. (2005). Multiobjectivity and complexity in embodied cognition. *IEEE Transactions on Evolutionary Computation*, 9(4) :337–360. (Cité en page 37.)

Thierens, D. (1996). Non-redundant genetic coding of neural networks. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 571–575. IEEE. (Cité en page 137.)

Thompson, A. (1995). Evolving electronic robot controllers that exploit hardware resources. *Advances in Artificial Life*. (Cité en page 38.)

Thrun, S., Burgard, W., et Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press. (Cité en page 4.)

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Stang, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., et Mahoney, P. (2007). The Robot that Won the DARPA Grand Challenge. *The 2005 DARPA Grand Challenge*, pages 1—-43. (Cité en page 3.)

Tinbergen, N. (1951). The study of instinct. (Cité en page 8.)

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4) :189–208. (Cité en page 41.)

Tonelli, P. (2012). *Relations entre plasticité synaptique et régularité des codages en neuro-évolution*. Thèse de doctorat, Université Pierre et Marie Curie, 4 place Jussieu, 75005 Paris. (Cité en pages 36, 137 et 139.)

Tonelli, P. et Mouret, J.-B. (2011a). On the Relationships between Synaptic Plasticity and Generative Systems. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 1531–1538. (Cité en pages 137 et 139.)

Tonelli, P. et Mouret, J.-B. (2011b). Using a Map-Based Encoding to Evolve Plastic Neural Networks. In *Proceedings of IEEE Symposium Series on Computational Intelligence*, pages 9–16. (Cité en pages 137 et 139.)

Trappenberg, T. P., Dorris, M. C., Munoz, D. P., et Klein, R. M. (2001). A model of saccade initiation based on the competitive integration of exogenous and endogenous signals in the superior colliculus. *Journal of Cognitive Neuroscience*, 13(2) :256–271. (Cité en page 137.)

Trianni, V. et Dorigo, M. (2006). Self-organisation and communication in groups of simulated and physical robots. *Biological cybernetics*, 95(3) :213–31. (Cité en page 38.)

Trujillo, L., Olague, G., Lutton, E., et de Vega, F. (2008). Discovering several robot behaviors through speciation. *Applications of Evolutionary Computing*, pages 164–174. (Cité en page 109.)

Trullier, O., Wiener, S. I., Berthoz, A., et Meyer, J. A. (1997). Biologically based artificial navigation systems : Review and prospects. *Progress in neurobiology*, 51(5) :483–544. (Cité en page 136.)

Tulving, E. (1972). Episodic and Semantic Memory1. *Organization of memory*, pages 381–402. (Cité en page 46.)

Ulbricht, C. (1996). Handling time-warped sequences with neural networks. In *From animals to animats*, pages 180–192. The MIT Press. (Cité en pages 32, 40 et 51.)

Urzelai, J. et Floreano, D. (2001). Evolution of adaptive synapses : robots with fast adaptive behavior in new environments. *Evolutionary computation*, 9(4) :495–524. (Cité en pages 102, 105 et 106.)

Urzelai, J., Floreano, D., Dorigo, M., et Colombetti, M. (1998). Incremental Robot Shaping. *Connection Science Journal*, 10(384) :341–360. (Cité en page 77.)

Van Dartel, M., Sprinkhuizen-Kuyper, I., Postma, E., et Van Den Herik, J. (2005). Reactive agents and perceptual ambiguity. *Adaptive Behavior*, 13(3) :227–242. (Cité en page 8.)

Varela, F. J., Thompson, E., et Rosch, E. (1991). *The embodied mind : Cognitive science and human experience.* (Cité en page 5.)

Variano, E. A., McCoy, J. H., et Lipson, H. (2004). Networks, dynamics, and modularity. *Physical review letters*, 92(18) :188701. (Cité en page 35.)

Vera, A. et Simon, H. A. (1993). Situation action : A symbolic interpretation. *Cognitive science*, 17 :7–48. (Cité en page 44.)

Vickerstaff, R. et Di Paolo, E. (2005a). An evolved agent performing efficient path integration based homing and search. *Advances in Artificial Life*, pages 221–230. (Cité en page 39.)

Vickerstaff, R. J. et Di Paolo, E. A. (2005b). Evolving neural models of path integration. *Journal of Experimental Biology*, 208(17) :3349–3366. (Cité en page 39.)

Vitay, J., Rougier, N. P., et Alexandre, F. (2005). A distributed model of spatial visual attention. In *Biomimetic Neural Learning for Intelligent Robots*, pages 54–72. Springer. (Cité en page 137.)

Wagner, T., Beume, N., et Naujoks, B. (2007). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proceedings of the 4th international conference on Evolutionary multi-criterion optimization*, EMO'07, pages 742–756, Berlin, Heidelberg. Springer-Verlag. (Cité en page 26.)

Walker, J., Garrett, S., et Wilson, M. S. (2003). Evolving Controllers for Real Robots : A Survey of the Literature. *Adaptive Behavior*, 11(3) :179–203. (Cité en pages 37 et 38.)

Walter, W. (1950). An imitation of life. *Scientific American*. (Cité en page 2.)

Watson, R. (2002). Embodied Evolution : Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1) :1–18. (Cité en page 38.)

Watts, D. J. et Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684) :440–2. (Cité en page 138.)

Whitehead, S. D. et Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1) :45–83. (Cité en pages 8 et 50.)

Whiteson, S., Stone, P., Stanley, K. O., Miikkulainen, R., et Kohl, N. (2005). Automatic feature selection in neuroevolution. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1225–1232, New York, NY, USA. ACM. (Cité en page 34.)

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4) :625–636. (Cité en page 5.)

Wilson, R. et Keil, F. (2001). *The MIT encyclopedia of the cognitive sciences*. MIT Press. (Cité en page 4.)

Wilson, S. W. (1991). The animat path to AI. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 15–21, Cambridge, MA, USA. MIT Press. (Cité en page 2.)

Winkeler, J. F. et Manjunath, B. S. (1998). Incremental evolution in genetic programming. *Genetic Programming*, pages 403–411. (Cité en page 76.)

Wolff, K. et Nordin, P. (2001). Evolution of efficient gait with an autonomous biped robot using visual feedback. *-RAS International Conference on Humanoid Robots,*. (Cité en page 38.)

Woolley, B. G. et Stanley, K. O. (2011). Evolving a single scalable controller for an octopus arm with a variable number of segments. *Parallel Problem Solving from Nature–PPSN XI*, pages 270–279. (Cité en page 36.)

Yacoub, M., Badran, F., et Thiria, S. (2001). A topological hierarchical clustering : Application to ocean color classification. *Artificial Neural Networks—ICANN 2001*, pages 492–499. (Cité en page 31.)

Yamauchi, B. et Beer, R. D. (1996). Spatial learning for navigation in dynamic environments. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 26(3) :496–505. (Cité en page 32.)

Yang, Q. et Ding, S. (2007). Novel algorithm to calculate hypervolume indicator of Pareto approximation set. *ICIC 2007*. (Cité en pages 25 et 26.)

Yao, X. (1993). Evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4(3) :203–222. (Cité en page 33.)

Yao, X. et Islam, M. M. (2008). Evolving artificial neural network ensembles. *Computational Intelligence Magazine, IEEE*, 3(1) :31–42. (Cité en page 137.)

Zhang, K., Statman, R., et Shasha, D. (1992). On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3) :133–139. (Cité en page 80.)

Ziegler, J. et Banzhaf, W. (2001). Evolving control metabolisms for a robot. *Artificial life*, 7(2) :171–90. (Cité en page 38.)

Ziemke, T. (1996). Towards adaptive behaviour system integration using connectionist infinite state automata. In *From animals to animats 4*. (Cité en pages 33 et 41.)

Ziemke, T. (1999). Remembering how to behave : Recurrent neural networks for adaptive robot behavior. *Recurrent neural networks : Design and applications*, pages 355–389. (Cité en page 33.)

Ziemke, T. (2001). Remembering how to behave : Recurrent neural networks for adaptive robot behavior. *Recurrent neural networks : Design and applications*, pages 355–389. (Cité en page 49.)

Ziemke, T., Bergfeldt, N., Buason, G., Susi, T., et Svensson, H. (2004). Evolving cognitive scaffolding and environment adaptation : a new research direction for evolutionary robotics. *Connection Science*, 16(4) :339–350. (Cité en page 40.)

Ziemke, T. et Thieme, M. (2002). Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adapt. Behav.*, 10(3-4) :185–199. (Cité en pages 33, 40, 50, 51, 69, 102, 103, 105, 106 et 107.)

Zipser, D. (1991). Recurrent network model of the neural mechanism of short-term active memory. *Neural Computation*, 3(2) :179–193. (Cité en pages 60 et 63.)

Zipser, D., Kehoe, B., Littlewort, G., et Fuster, J. M. (1993). A spiking network model of short-term active memory. *J. Neurosci.*, 13(8) :3406. (Cité en pages 60 et 63.)

Zitzler, E. et Künzli, S. (2004). Indicator-based selection in multiobjective search. *Parallel Problem Solving from Nature-PPSN VIII*, (i) :1–11. (Cité en page 25.)

Zitzler, E., Laumanns, M., et Thiele, L. (2001). SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. *Computer Engineering*, TIK-Report(103) :1–21. (Cité en pages 21 et 26.)

Zitzler, E. et Thiele, L. (1999). Multiobjective evolutionary algorithms : A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE*, 3(4) :257–271. (Cité en pages 24, 25, 26 et 135.)

Zufferey, J.-c., Floreano, D., Leeuwen, M. V., et Merenda, T. (2002). Evolving Vision-Based Flying Robots. *Evolutionary Computation*, pages 592–600. (Cité en page 38.)

Zykov, V., Bongard, J., et Lipson, H. (2004). Evolving Dynamic Gaits on a Physical Robot. In *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO*. (Cité en page 38.)