Robotique évolutionniste: conception orientée vers le comportement

Stéphane Doncieux

Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique

Habilitation à diriger des recherches

spécialité « Informatique »

par

Stéphane Doncieux

Robotique évolutionniste : conception orientée vers le comportement

Soutenue le 14 décembre 2010 devant le jury composé de:

Pr.	Wolfgang Banzhaf	Memorial University of Newfoundland	(Rapporteur)
Dr.	Raja Chatila	Directeur de recherche LAAS/CNRS	(Examinateur)
Pr.	Patrick Gallinari	UPMC/CNRS	(Invité)
Dr.	Inman Harvey	University of Sussex	(Rapporteur)
Dr.	Jean-Arcady Meyer	Directeur de recherche émérite UPMC/CNRS	(Examinateur)
Pr.	Patrice Perny	UPMC/CNRS	(Examinateur)
Dr.	Marc Schoenauer	Directeur de Recherche INRIA Saclay	(Rapporteur)

REMERCIEMENTS

L'habilitation à diriger des recherches reflète un travail réalisé sur une période assez longue, comprenant ma thèse et s'étalant jusqu'à aujourd'hui. L'essentiel de ces travaux a été réalisé avec des étudiants que j'ai encadré ou en collaboration avec des collègues sans qui tout cela n'aurait pas été possible. Ce travail est donc un travail collectif, plus que la contribution d'une personne isolée. Je m'estime très chanceux d'avoir pu bénéficier de collaborateurs à la fois motivés et efficaces.

Je tiens à remercier Jean-Baptiste Mouret, avec qui j'ai travaillé depuis de nombreuses années, notre collaboration ayant commencé alors que j'étais encore en thèse et qu'il arrivait au laboratoire pour son stage de fin d'étude. Les innombrables discussions et confrontations de points de vue que nous avons pu avoir au fil des ans sont pour beaucoup dans les principaux résultats présentés dans ce manuscrit. Je le remercie également pour sa relecture et ses commentaires sur ce manuscrit. Adrien Angeli, Renaud Barate, Thierry Druot, Benoît Girard, Christophe Grand, Mohamed Hamdaoui, Sylvain Koos, Emmanuel de Margerie, Pascal Martinelli, Laurent Muratet, Tony Pinville, Pierre Sagaut, Mathieu Schmitt, Vincent Padois et Paul Tonelli ont tous contribué à ces travaux tout en créant une ambiance de travail stimulante et conviviale ce dont je ne peux que les remercier. Je tiens aussi à remercier Nicolas Bredèche pour nos nombreuses discussions et ses commentaires sur les premières versions de ce manuscrit.

Le cadre de l'AnimatLab où j'ai commencé ces travaux était très propice pour se former et mener des recherches sereinement. J'ai ainsi grandement bénéficié de la rigueur scientifique, de la confiance et de la bienveillance de Jean-Arcady Meyer et le remercie pour tout ce qu'il m'a apporté. L'arrivée à l'ISIR m'a permis d'être en contact avec Philippe Bidaud, qui a également été un soutien constant et qui m'a ouvert les yeux sur le point de vue d'un roboticien sur les thèmes de recherches qui sont les miens. Je le remercie pour cela et pour sa remarquable ouverture d'esprit.

Je tiens également à remercier Marc Schoenauer, Wolfgang Banzhaf et Inman Harvey d'avoir accepté de se pencher sur mon manuscrit et d'en être les rapporteurs.

Merci enfin à ma chère et tendre Christine. Son ardeur et sa motivation dans son travail, m'ont encouragé à entreprendre la rédaction de ce manuscrit et à la mener jusqu'à son terme. Merci à elle également d'avoir eu le courage de vérifier que ma prose était dans un français convenable.

Villejuif, le 2 octobre 2010.

RÉSUMÉ

Concevoir un contrôleur pour rendre un robot autonome capable d'accomplir une mission donnée dans un environnement non contrôlé est une tâche difficile. Les robots du futur assisteront les humains dans leurs tâches de tous les jours et dans ce but, ils auront à être simples et robustes. Exploiter au mieux leurs caractéristiques est donc un enjeu critique, tant pour des raisons d'efficacité que pour des raisons économiques. La disponibilité d'outils d'aide à la conception de robots proposant automatiquement les comportements les plus simples pour une tâche considérée pourrait donc être une étape critique pour le développement de ces futurs robots. À ce jour, un concepteur de robot doit faire par lui-même, en s'appuyant sur son expérience, le passage d'un cahier des charges donné aux informations qu'utilisent les méthodes de conception robotique – typiquement un ensemble de positions successives qu'un système de planification peut ordonnancer et un système de commande réaliser.

L'objectif des travaux présentés dans ce manuscrit est de concevoir une méthode de conception orientée vers le comportement et dédiée à la robotique mobile et autonome. Cette méthode doit partir des informations disponibles, à savoir une description de la mission à accomplir. Constatant que les animaux ont un niveau d'autonomie élevé qui permettrait à des robots d'accomplir de nombreuses missions, nous avons choisi de focaliser cette étude, dans le cadre de l'approche animat, sur le « mécanisme de conception » à l'œuvre dans la nature : la sélection naturelle. Ce processus algorithmique a l'avantage de ne prendre en compte que le résultat, c'est à dire la capacité à transmettre ses gènes. Considérant que cette capacité résulte d'un comportement potentiellement complexe, nous avons formulé l'hypothèse qu'un algorithme inspiré de la sélection naturelle peut servir de base à une méthode de conception orientée comportement. L'essentiel des travaux présentés concerne la conception de contrôleurs, mais la méthode pourrait s'étendre à la conception de la morphologie du robot et de ses capteurs. Ce manuscrit présente les travaux réalisés depuis ma thèse en 2003.

La conception orientée comportement basée sur des algorithmes évolutionnistes telle que présentée ici, nécessite tout d'abord de disposer d'une simulation fiable et rapide. Dans une première étape préliminaire, des expériences peuvent utiliser des algorithmes évolutionnistes multi-objectifs pour générer des données utiles à un expert pour mieux comprendre le système considéré et identifier des régularités ou des règles de conception spécifiques. Nous avons appliqué une telle méthode à l'étude d'un drone à ailes battantes. À la fin de cette étape, le concepteur propose un prototype de robot et va explorer dans l'espace des comportements pour évaluer ses capacités par rapport à la mission qu'il aura à accomplir. Pour cette seconde étape, nos principales contributions, outre quelques preuves de concept, sont les suivantes :

– nous avons proposé une approche incrémentale multi-objectif pour pallier à certaines limitations des approches incrémentales classiques ;

– nous avons proposé la diversité comportementale pour améliorer la recherche dans l'espace des comportements et ainsi réduire les risques de convergence prématurée ;
– notre étude de la modularité, propriété considérée comme critique pour la synthèse de contrôleurs de complexité croissante, a montré l'importance de prendre en compte *simultanément* le codage et les pressions de sélection, négliger un de ces points conduisant à une chute de performance ; nous avons également proposé un codage incluant les capacités de modularité, répétition et hiérarchie ;
– nous avons enfin proposé une approche pour réduire l'écart entre les comportements obtenus en simulation et sur le robot réel, approche basée sur des expériences sur le robot réel dont on cherche à minimiser le nombre.

Tous les comportements considérés dans ce manuscrit sont essentiellement réactifs, c'est-à-dire que les mouvements ou actions entrepris à un instant donné ne dépendent que des perceptions courantes. L'objectif du projet de recherche présenté est de tendre vers la synthèse de comportements plus cognitifs afin d'aborder des tâches plus complexes. La méthode proposée, développée dans le cadre du projet ANR EvoNeuro consiste à étudier les capacités cognitives identifiées par des chercheurs en neurosciences computationnelles. Des protocoles d'évaluation peuvent ainsi être importés des neurosciences, ainsi que des modèles permettant de comparer les résultats obtenus. Cette approche nécessite de générer des réseaux de neurones structurellement similaires à ceux des neurosciences pour faciliter les comparaison et l'importation des connaissances depuis ce domaine vers la robotique évolutionniste. Nous avons donc développé un codage exploitant les primitives structurelles des modèles de neuroscience que nous avons appliqué en 2010 à deux tâches : la sélection de l'action et une tâche de mémoire de travail. Les perspectives de ces travaux seront d'étudier d'autres capacités, puis d'aborder la question de l'évolution de contrôleurs incluant plusieurs de ces capacités. Nos travaux sur la modularité permettent d'envisager avec optimisme l'étude de cette question. L'objectif à terme est de faire en sorte que la méthode de conception orientée vers le comportement génère des contrôleurs réactifs si nécessaire, mais soit également capable de synthétiser des fonctions de mémoire, de représentation spatiale, de sélection d'action, d'apprentissage, d'analyse de scène, etc. si cela s'avère utile à la résolution de la tâche considérée. La démarche proposée repose sur une interaction étroite avec des chercheurs en neuroscience. Les outils développés sont ainsi mis à leur disposition pour améliorer leur compréhension des mécanismes à l'œuvre dans le cerveau. Ces travaux pourront donc également contribuer, indirectement, à améliorer notre compréhension du cerveau au travers de ces interactions interdisciplinaires.

Un objectif à plus long terme sera de rendre ces méthodes applicables en ligne, de façon à ce qu'un robot puisse, au-delà de sa robustesse et de ses capacités d'apprentissage, s'adapter à des changements drastiques de son environnement.

Remarque : la version anglaise du résumé est plus précise que sa version française de façon à offrir un aperçu détaillé de ce travail, décrit en français dans la suite de ce manuscrit, aux lecteurs non francophones.

SUMMARY

Future robots will assist humans in their every day life. For that purpose, they will have to be simple and robust. Exploiting at best their features will then be a critical issue for efficiency, but also economical reasons. The availability of new tools to assist robot designers by proposing them automatically the most simple behavior required to reach a goal might then be a milestone for future robot development. Today, the discrepancy between the robot mission and the data current robotics methods rely on – typically a set of robot positions that a planing system will schedule and a command system that will realize it – is handled by the robot designer who can rely on its experience only.

The goal of my research is to develop a behavior oriented design methodology dedicated to mobile and autonomous robotics. The sought design methodology should start from what is known, i.e. a description of the mission to be fulfilled. Noticing that animals have a high level of autonomy that would provide to robots the ability to fulfill numerous missions, we have focused our search, in the frame of the animat approach, on the main mechanism that has built and designed living creatures : natural selection. One of the features of natural selection, is that it only takes into account the results, i.e. the ability to transmit the genes. Considering that the selection pressure acts on the results of a creature behavior, our hypothesis in this work is then that an algorithm inspired from evolution may be the basis of a behavior oriented design methodology. Our main focus will be on control system design, but the methodology might extend to morphology and sensors design. In the following, the different work undertaken since my PhD defence in 2003 are briefly summarized and put in the perspective of the behavior oriented design methodology. References to publications I coauthored are given to find more details.

We proposed to classify Evolutionary Robotics work into four different categories Doncieux et al. [2010] :

– parameter optimization
– evolutionary aided design
– online Evolutionary Algorithms
– evolutionary synthesis

Each category contributes differently to the robot design process. Evolutionary aided design is to be used at the very beginning of the process whereas parameter optimization is mostly useful at the very end. Evolutionary synthesis covers a wider part of the process, as it also aims to find a structure, whereas for parameter optimization, it is a choice of the designer. Online Evolutionary Algorithms correspond to an open-ended design process that never stops.

We develop here our contributions that concern these different categories, except online evolution. For parameter optimization, we will show some proof-of-concept experiments. We will then present how evolutionary algorithms, and in particular multi-objective algorithms, can be used in an evolutionary aided design process. Evo-

lutionary synthesis will be considered next, with a proof-of-concept experiment and another experiment showing some of the limitations of current approaches. Solutions to the identified problem and to some other classical problems of evolutionary synthesis are then proposed. Last we will consider the question of the gap between simulation and reality that may prevent solutions generated *in silico* to be of use on real robots.

Evolutionary algorithms are now mature optimization tools which are particularly interesting on problems that feature non linearity, non continuity or non derivability of the function to optimize. This is then adapted to the optimization of some robot features whose impact is measured by the observation of the behavior of the robot, i.e. of its interaction with its environment. Evolutionary algorithms can then directly be used to "optimize" the behavior of robots. We have applied it to find the best suited behavior for some simulated UAVs whose goal was to extract energy from the environment thanks to thermals, slope winds or wind gradients [Doncieux et al., 2007; Barate et al., 2006]. With a thermal in the neighborhood, the evolved controller made a glider turn round, with a large radius to maximize the chance to discover the thermal, and once into a thermal, the behavior changed to a spiral trajectory with a small radius allowing the glider to remain into the thermal and gain altitude. Similarly, 'S' trajectories have been generated while flying near a slope with an up wind and more complex trajectories have been optimized to reproduce albatrosses behavior extracting energy from wind gradients.

The same tools can be used in a different purpose. Evolutionary algorithms are of particular interest in the case of multi-objective problems. In this case, there is no single optimal solution but a set of optimal trade-offs, the Pareto set. This feature of multi-objective problems can be used in order to generate sets of data that can then be analyzed by experts to extract regularities and features of optimal solutions. The Pareto front itself provides information on the values that are within reach for each objective. The relation between each optimized parameter and the objectives can be studied this way. We have applied it to the optimization and analysis of flapping wings robots. Several morphologies with different degrees of freedom have thus been compared, showing, for instance, that a four degrees-of-freedom per wing system is more energy efficient than a three degrees-of-freedom system, which is also much more efficient than a two degrees-of-freedom system [de Margerie et al., 2007; Hamdaoui et al., 2010]. The energy-speed relation, feature of a particular flapping-wings setup, has also been empirically evaluated [Doncieux and Hamdaoui, 2010].

The former experiments were in the classical frame of optimization tools. The algorithms optimized a set of parameters, i.e. the search space was as subset of $\mathbb{R}^n$ or $\mathbb{R}^n$ itself. One of the drawbacks of this approach is that it imposes to choose beforehand the structure of the controller or morphology, thus reducing the space of possible behaviors. Letting the structure evolve may allow the search to start proceeding with simple structures before considering more complex ones. Proof-of-concept experiments on the evolution of a flight controller for a simulated flapping wing aircraft, proved that evolution of neural network structure and parameters can solve a difficult task of flight stabilization. We also proved the limits of a classical incremental approach in which a task is decomposed into simpler tasks solved one at a time. The choice of the solution to keep at one step revealed to be of utmost importance

on the results of the next step, while nothing allowed to forecast it [Mouret et al., 2006]. This has motivated research on a new incremental scheme. A multi-objective approach has then been proposed that avoids any choice during the process : every step in the incremental scheme has its own objective and all objectives are active during a run. When a step is out of reach, all individuals get an equally low fitness value on the corresponding objective that is then implicitly ignored by the selection process. When a step is within reach at a particular generation, individuals performing differently will appear ; they will have different values on the corresponding objective and a selection pressure driving to the resolution of this step will appear. A light seeking task requiring to discover a sequence of actions has been solved with this method [Mouret and Doncieux, 2008a].

The recourse to an incremental scheme is a way to avoid the bootstrap problem, but another approach has been proposed since the very first evolutionary algorithms : keeping a high diversity. While diversity keeping has been carefully explored in genotype, phenotype and even fitness spaces, up to now it had not been explored in the behavior space that is an important feature of Evolutionary Robotics experiments. Actually, taking behavior diversity into account as a separate objective, besides a goal-oriented objective, revealed to be enough to solve a deceptive problem of boolean function synthesis [Mouret and Doncieux, 2009c] and several sequential tasks on a simulated robot [Mouret and Doncieux, 2009b; Doncieux and Mouret, 2010]. An interesting result is that among the different behavior diversity measures tested, a problem independent behavior description relying on the sensorimotor stream generated results competitive with specific behavior descriptions, thus suggesting that a generic objective may advantageously replace a difficult and tricky fitness shaping.

The question of how to represent an evolving structure has also been studied, with a particular focus on modularity. ModNet, A minimalist encoding including modularity and repetition has been proposed and tested on various tasks [Doncieux and Meyer, 2003, 2004b]. Hierarchy, i.e. the ability to recursively compose modules, was absent from this encoding and MENNAG has thus been proposed to include this feature [Mouret and Doncieux, 2008b]. MENNAG description relied on attribute grammars, a formalism adapted to represent several different kinds of encoding, like cellular encoding, for instance, and thus make comparisons easier. Another advantage is that the encoding is completely described by the grammar and new variants can easily be defined and compared. On several tasks, requiring repetition of modules or not, MENNAG revealed to be more efficient than ModNet and NEAT.

Besides the *ability* to evolve modular solutions, remains a question that has hardly been answered : will evolution be able to generate modular solutions ? Put differently, the question might be : what use will be a proto-module if it doesn't exhibit its full functionality ? Biologists actually ask questions like "what use is half a wing ?" and hopefully they find answers to these questions while, in Evolutionary Algorithms, the question was seldom raised that way. This would actually imply to think differently and define selection pressures that would help some functionality to appear in the network before having a chance to be used to fulfill the given mission. This has led us to define an approach named from the biological term "exaptation". In this approach, a modular neural network encoding is considered and objectives are

added, in a multi-objective scheme, to reward a functionality that modules are supposed to implement. A deceptive boolean function revealed to be easily solved this way while generating modular solutions reproducing the structure of the function. Other encodings like NEAT revealed to be less efficient, but more interestingly, using a modular encoding *alone* was much less efficient and similarly using an objective rewarding the sought functionality of the module also decreased performance significantly. To have it work, both the selection pressure *and* the modular encoding needed to be used simultaneously, otherwise the performance drops [Mouret and Doncieux, 2009a]. This work suggests then that, concerning modularity, encoding and selection pressures can't be studied separately and must be considered simultaneously to have a significant gain, what is nowadays rarely done in Evolutionary Robotics.

The last contribution of this work concerns the reality gap : how to generate behaviors that transfer well from a simulated robot to a real one ? Considering that simulation is hard to avoid, because it helps speeding the computations and considering the opportunistic nature of evolution, how to avoid behaviors that would exploit features of the simulation which have no equivalent on the real robot ? Such problems often occur in Evolutionary Robotics. In a physical simulation whose collision management system is not perfect, evolution may find walkers able to travel thousand miles in a second, just by making the robot explode because of carefully chosen collisions. Two different cases have been considered. In both of them, experiments are made on the targeted robot and the goal is to minimize their number. In the first, the simulation was co-evolved with the control, this was applied to a quadrotor helicopter [Koos et al., 2009]. In the second, the simulation was imposed, but the evolution had to maximize a transferability objective, which is an approximation of the real transferability as it can be evaluated considering the transfers on the real robots that have occurred up to now [Koos et al., 2010].

The presented work suggests different ways to include Evolutionary Robotics experiments in a behavior oriented design process. Once a reliable and fast simulation is available, preliminary experiments may be used to gain some insights on a new robot design, as we did on the flapping wing aircraft. Relying on some expert choices at the preliminary step, the behavior based search can begin and propose different ways to reach the goal. Our main contributions to this step are the following :

- the multi-objective incremental scheme, to solve problems of classical incremental approaches ;
- behavioral diversity, used to encourage search in the behavior space ;
- in a modular context, we showed the necessity to consider both encoding *and* selection pressure issues at the same time, forgetting one of the two aspects leading to a significant (and counterintuitive) drop of performance ; we also proposed ModNet encoding, and then MENNAG, which includes features like modularity, repeatability and hierarchy ;
- last, we proposed a solution to the reality gap problem ; it relies on experiments conducted on the real robot that the method tries to minimize.

Up to now, the behaviors which are within reach for this method are mainly reactive. The goal of my research project is to generate controllers with increasing cognitive abilities in order to solve more complex tasks. The proposed methodology,

developed in the frame of the ANR EvoNeuro project consists in studying cognitive abilities considered by neuroscientists. They have indeed described functionalities like working memory, action selection, etc, together with protocols to test them and models that can be used as a reference. A new encoding has been proposed to include structure primitives from computational neuroscience. It has been successfully tested on an action selection task [Mouret et al., 2010] and on working memory [Pinville and Doncieux, 2010]. Future work will consider other cognitive abilities and also the resolution of more complex tasks, mainly inspired from rat experiment, to test the abilities of our method to generate cognitive abilities similar to that of a rat. The objective is to define a tool that would be able to generate a behavior that is robust and adapted to a particular mission. A simple and reactive controller might be generated, if it is enough, but a more cognitive controller – including for instance working memory, map making abilities, action selection, learning, scene analysis, etc. – might be generated, if the task requires it. The questions to be studied concern the evolvability of cognitive or learning systems. Modularity might be a critical issue in this context and exaptation part of an answer. A close collaboration with neuroscientists is mandatory for the proposed approach, and thus our method may also provide new insights to them in their study of the brain. A long term objective is to have the method work online on the robot to make it more resilient and allow it to adapt to completely new situations for which robustness or leaning is not enough.

TABLE DES MATIÈRES

1 INTRODUCTION

MATIÈRES

Les robots mobiles connaissent un essor important et des robots de capacités très variées sont désormais disponibles. Un robot comme le BigDog de Boston Dynamics[1], peut se déplacer sur la neige, la glace, ou en terrain très accidenté. Le robot serpent de Carnegie Mellon peut grimper aux arbres[2] et les robots amphibiens du BioRobotics Lab de l'EPFL peuvent évoluer tant sur la terre ferme que dans l'eau[3]. Tous ces robots peuvent se déplacer d'une position à une autre pour peu que le terrain les séparant respecte les contraintes imposées par la mécanique et les lois de commandes utilisées. Des méthodes de localisation et de cartographie simultanée peuvent leur ajouter la capacité à reconnaître un lieu déjà rencontré et à planifier des trajectoires. Les travaux réalisés en vision et audition artificielle ambitionnent de leur donner une perception de leur environnement aussi adaptée que possible à leurs besoins.

Ces différentes capacités sont indispensables, mais ne couvrent pas l'ensemble des prérequis au développement de robots autonomes, capables d'évoluer dans un environnement non contrôlé, avec des missions qui sont toutes nécessairement très spécifiques. En effet, chacune de ces capacités doit être mise au service d'un système chargé de déterminer le *comportement* du robot, autrement de décider ce qu'il doit faire à chaque instant compte tenu du contexte et des priorités du robot et quelle stratégie adopter pour résoudre un problème donné. Dans un contexte où un opérateur humain peut le déterminer avec précision, des algorithmes de planification peuvent ensuite calculer l'enchaînement d'actions et de mouvements à réaliser. Cependant, ne serait-ce que pour des raisons de coût, il est intéressant de disposer de robots véritablement autonomes et capables d'accomplir cette tâche par eux-même.

L'objet des travaux présentés ici est de contribuer au développement d'un processus de conception qui serait centré sur la notion de comportement et aurait donc pour but de générer un système de contrôle exhibant un comportement souhaité, ou plutôt un comportement ayant un résultat souhaité.

1 http://www.bostondynamics.com/
2 http://www.cs.cmu.edu/~biorobotics/serpentine/serpentine.html
3 http://biorob.epfl.ch/page38261.html

Ce manuscrit résume les travaux réalisés depuis ma thèse, soutenue en 2003. Ils ont été réalisés à l'AnimatLab, dirigé par Jean-Arcady Meyer, qui était également mon directeur de thèse. L'AnimatLab était, sur la période 1999-2007 part intégrante du LIP6 (Laboratoire d'Informatique de Paris 6, UMR 7606). Nous avons ensuite décidé de quitter le LIP6 pour rejoindre l'ISIR (Institut des Systèmes Intelligents et de Robotique, UMR 7222). Ces travaux se sont donc ensuite poursuivis dans ce laboratoire, dirigé par Philippe Bidaud et créé en 2007, et en particulier dans l'équipe de recherche SIMA (Systèmes Intégrés Mobiles et Autonomes), coordonnée par Bruno Gas et moi-même.

1.1 CONTEXTE SCIENTIFIQUE ET MOTIVATIONS

UNE MÉTHODE DE CONCEPTION ORIENTÉE VERS LE COMPORTEMENT Le domaine de l'automatique et de la commande offre des outils permettant à un système de suivre une trajectoire de consigne [Levine, 1996]. Dans le contexte de la robotique autonome, la commande est donc associée à un dispositif de planification permettant de déterminer cette trajectoire désirée [Latombe, 1991].

D'un point de vue opérationnel, l'objectif du concepteur d'un robot est que la tâche que le robot doit prendre en charge soit effectivement résolue et ce par le moyen le plus efficace et économique possible. Les méthodes actuelles de conception en robotique nécessitent que le concepteur fasse un certain nombre de choix quant à la stratégie à adopter pour résoudre le problème. S'il parvient à identifier comment résoudre le problème, alors les outils issus du domaine de la théorie de la commande et de la planification lui permettront de le mettre en œuvre.

Le problème fondamental de ce type d'approche est que la performance finale du système dépendra fortement des choix réalisés en amont par le concepteur. Or rien ne permet de guider ces choix, si ce n'est un état de l'art des robots fonctionnels les plus proches. Une des difficultés rencontrées par un concepteur dans ce cas est qu'il doit quitter son point de vue anthropocentrique pour se mettre à la place du robot et se demander comment ce dernier pourrait résoudre sa tâche compte tenu de ses perceptions et de ses moyens d'action.

Ce changement de point de vue est au cœur de l'approche *animat* [Wilson, 1991; Meyer, 1995; Roitblat, 1995] qui suggère de prendre en compte la biologie et en particulier la façon dont des êtres vivants ont pu résoudre des problèmes similaires [Meyer and Guillot, 2008], élargissant ainsi les sources d'inspiration potentielles. Cette approche a été proposée en réponse à une approche purement symbolique de l'intelligence artificielle, reposant sur la manipulation de connaissances dans un formalisme de représentation permettant le raisonnement et l'inférence de nouvelles connaissances. Une approche symbolique est très attractive de par l'autonomie qu'elle pourrait conférer au robot, cependant elle est très sensible au problème de l'ancrage de symboles. Comment faire une association fiable et reproductible entre des perceptions dans un environnement complexe et les concepts abstraits manipulés par le système de raisonnement ? Cette question est toujours d'actualité, même si plusieurs applications robotiques significatives ont été réalisées avec succès [Hertz-

berg and Chatila, 2008]. L'approche située de l'intelligence artificielle consiste à éluder ce problème en évitant le recours à un modèle abstrait du monde [Brooks, 1991]. Sans modèle interne, le robot perd la capacité à raisonner sur le monde, mais en même temps, il évite le problème de l'adéquation de ce modèle avec l'environnement. Ce type d'approche vise donc des robots moins intelligents mais plus adaptés, voire adaptatifs à des changements de leur environnement, comme le sont tous les êtres vivants, dont la plupart ne sont pas dotés de capacités de raisonnement avancées. L'accent est alors mis sur les comportements, qu'ils soient réactifs ou plus cognitifs [Mataric and Michaud, 2008].

La conception d'un robot à partir de comportements élémentaires reste cependant une tâche délicate. En effet, le comportement résulte de l'interaction du robot avec son environnement et un contrôleur, même très simple, peut provoquer un comportement complexe et difficilement prédictible, en particulier si l'environnement lui-même est complexe. Il serait donc très intéressant de définir un processus de conception partant directement des informations dont dispose le concepteur de robot, autrement dit du cahier des charges portant sur le comportement du robot et ses conséquences. L'intérêt d'une telle approche serait de faciliter la conception en la rendant moins dépendante de l'expérience et des compétences du concepteur. Idéalement, le processus partirait d'une description de la tâche à résoudre et la méthode de conception devrait trouver aussi automatiquement que possible comment la résoudre et mettre en œuvre des solutions.

ADÉQUATION ENTRE LES CAPACITÉS PERCEPTIVES, PHYSIQUES ET LE SYSTÈME DE CONTRÔLE L'interaction du robot avec son environnement repose sur plusieurs éléments. Ses capacités de perception lui permettent de déterminer les actions qu'il va ensuite accomplir, actions qui auront un impact sur l'environnement et donc en particulier sur les prochaines perceptions du robot. Cette boucle est donc composée, du point de vue du robot, de trois éléments fondamentaux : le système de perception, le système de contrôle et la structure physique du robot avec les actionneurs associés. Le comportement du robot dépend fortement de ces trois éléments qui sont donc tous à prendre en compte dans le processus de conception.

Le choix de capteurs adaptés peut ainsi simplifier considérablement la tâche du système de contrôle, comme le montre l'expérience des Didabots de Maris et Boekhorst [Maris and Te Boekhorst, 1996]. Cette expérience, à visée pédagogique, consiste à concevoir des robots dont le but est de regrouper des cubes dans une arène. Les seuls capteurs utilisés sont des capteurs de distance. Les auteurs montrent qu'en donnant le bon écartement aux capteurs, le problème peut être résolu avec un simple contrôleur d'évitement d'obstacles : les cubes rencontrés pendant l'avancement du robot ne sont pas vus, le robot les pousse donc jusqu'à rencontrer un autre bloc sur le côté ou un mur. Il fait alors demi-tour en laissant le ou les cubes poussés sur place. Après un certain temps, il aura donc tendance à créer des tas de cubes. De même, l'exploitation du couplage mécanique-commande, notamment dans le cas des marcheurs passifs [Collins et al., 2005], permet par exemple à une robot humanoïde de marcher en consommant très peu d'énergie.

Cette prise en compte des caractéristiques du robot dans son ensemble s'insère dans l'approche dite des machines morpho-fonctionnelles [Hara and Pfeifer, 2003].

Cependant, aucune méthode de conception ne permet encore de tirer parti de ce type d'approche aussi la conception de robots répondant à ces principes reste un problème difficile. Une méthode de conception à même de prendre en compte simultanément perception, contrôle et morphologie/actionnement doit donc être développée pour bénéficier au mieux des gains de ce type d'approche.

1.2 PROBLÉMATIQUE

La problématique traitée dans la suite de ce document est la conception d'un système robotique sur la base d'une description de son comportement désiré ou plus précisément de l'impact de ce comportement sur l'environnement et le robot. Considérant l'intérêt potentiel de machines morpho-fonctionnelles, la méthode choisie se doit d'être peu contraignante sur l'espace de recherche considéré, ceci afin de pouvoir représenter et concevoir différents types de systèmes de contrôle, mais aussi la morphologie du robot et éventuellement son système de perception[4].

Considérant, dans le cadre de l'approche animat, que les êtres vivants sont des exemples frappants de « machines » morpho-fonctionnelles et qu'un mécanisme de conception orienté vers le comportement est à l'œuvre dans la nature, nous formulons l'hypothèse suivante :

Un mécanisme inspiré de la sélection naturelle des espèces peut servir de base à un processus de conception automatique orienté vers le comportement.

La sélection naturelle des espèces est en effet le principe quasi algorithmique proposé par Darwin pour expliquer l'apparition et l'adaptation des espèces à leur environnement [Darwin, 1859]. Elle a inspiré de nombreux algorithmes depuis les années soixante. Ces algorithmes sont maintenant très matures et utilisés fréquemment dans un cadre d'optimisation. Ce choix a également l'avantage de définir un cadre d'étude mathématiquement très peu contraignant : les fonctions à optimiser n'ont besoin d'être ni linéaires, ni dérivables, ni même continues. En effet, les algorithmes évolutionnistes nécessitent essentiellement la définition des opérateurs génétiques de croisement et de mutation. N'importe quel type de structure peut donc être conçu sur cette base en exploitant les algorithmes évolutionnistes les plus performants, pour peu que des opérateurs efficaces lui soient associés.

Cette hypothèse étant formulée, la définition d'une méthode évolutionniste de conception basée comportement pose la question de sa mise en œuvre :

- comment utiliser ces approches et pour quel usage ?
- quel type de représentation génétique utiliser ?
- comment définir la fonction de fitness guidant le processus de sélection ?
- comment explorer efficacement l'espace des systèmes de contrôle ?

L'objet de ce manuscrit est d'apporter des éléments de réponse à ces questions.

4 nous ne verrons pas ce dernier point dans les travaux présentés ici.

1.3 DÉFINITIONS

OPTIMISATION MONO- ET MULTI-OBJECTIF La notion d'optimisation est cruciale pour les travaux présentés dans ce document. Sa définition formelle [Rao, 2009; Minoux, 2007] est la suivante :

$$\text{Trouver } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \end{Bmatrix} \text{ qui maximise/minimise } f(\mathbf{X})$$

sous les contraintes :

$$\begin{aligned} g_j(\mathbf{X}) &\leqslant 0, \quad j = 1, 2, \ldots, m \\ l_j(\mathbf{X}) &= 0, \quad j = 1, 2, \ldots, p \end{aligned}$$

avec :

- $\mathbf{X}$: vecteur de paramètres, nous utiliserons aussi le terme de *génotype*, issu de la communauté des algorithmes évolutionnistes ;
- $f(.)$: fonction objectif ou fonction de coût, nous utiliserons la plupart du temps l'appellation de fonction de *fitness*, issue de la communauté des algorithmes évolutionnistes ;
- $g_j(.)$: contrainte inégalité ;
- $l_j(.)$: contrainte égalité.

Lorsque $f(.)$ est à valeur dans $\mathbb{R}$ l'optimisation est dite mono-objectif, lorsqu'elle est dans $\mathbb{R}^n$, elle est dite multi-objectif. Dans ce cas, chaque composante de la fitness sera dénommée *objectif*.

Dans les travaux présentés ici, nous n'aurons pas de contraintes ou nous les prendrons en compte directement dans la fonction de fitness en adoptant la méthode des pénalités – consistant à diminuer drastiquement la fitness des individus ne respectant pas les contraintes (dans le cas d'une maximisation) – ou celle de la mort subite – consistant à donner une valeur de fitness arbitrairement basse lorsque les contraintes ne sont pas respectées – [Yeniay, 2005].

ALGORITHMES ÉVOLUTIONNISTES Un algorithme évolutionniste est un algorithme stochastique itératif, générant une solution approchée sur la base d'une recherche en parallèle au moyen d'opérateurs dits *opérateurs génétiques* [Eiben and Smith, 2008]. La recherche opère sur une *population*, i.e. un ensemble, de *solutions potentielles* ou *solutions candidates* que nous appellerons également *individus*. Les opérateurs génétiques sont au nombre de deux : la *mutation* et le *croisement*. La mutation consiste à appliquer une petite modification aléatoire à un individu et le croisement consiste à mélanger plusieurs individus. Chaque solution candidate est évaluée et une sélection s'opère sur la base du résultat de cette évaluation (figure 1).

Ces algorithmes s'inspirent de la théorie de la sélection naturelle des espèces formulée par Darwin [Darwin, 1859]. Les notions importantes dans ce domaine ont ainsi une dénomination issue de la biologie plutôt que des appellations issues du domaine de l'optimisation. De nombreux algorithmes utilisent ces notions de différentes manières, essentiellement pour des raisons historiques, plusieurs communautés ayant travaillé en parallèle à partir des années 1960 sur ce sujet. Les algo-

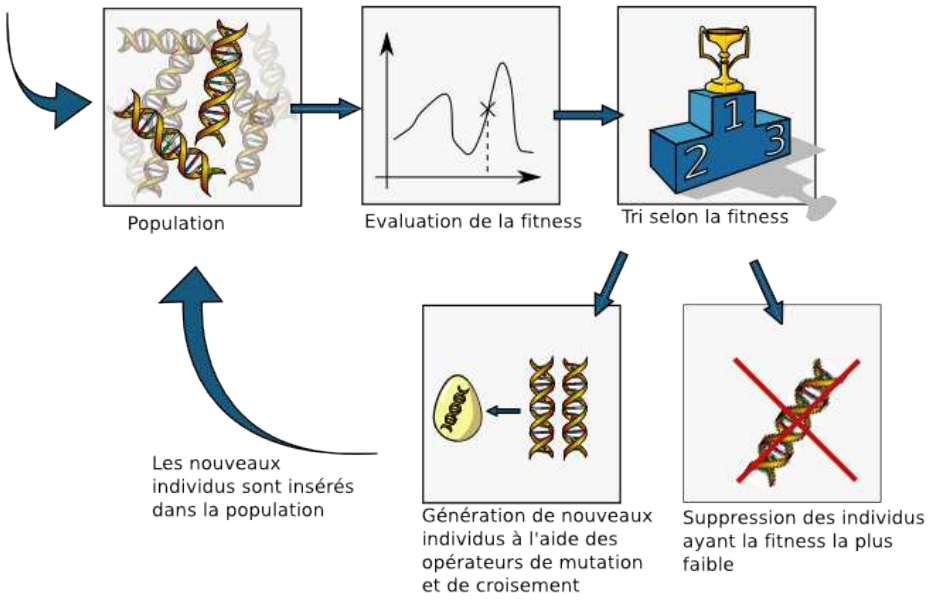


FIG. 1: Le principe général des algorithmes évolutionnistes.

rithmes génétiques [Holland, 1992; Goldberg, 1989], les stratégies d'évolution [Back and Schwefel, 1995; Schwefel and Back, 1995; Schwefel and Rudolph, 1995], la programmation évolutionniste [Fogel et al., 1966] ou encore la programmation génétique [Koza, 1993, 1994] sont les principales variantes et diffèrent essentiellement par l'espace de recherche considéré et par la stratégie de sélection. Par référence à la biologie, dans cette communauté, le terme de *génotype* désigne les paramètres ou structures soumises aux opérateurs génétiques et le *phénotype* désigne les structures à optimiser, autrement dit les structures sur lesquelles porte l'évaluation. Les algorithmes génétiques considèrent pour génotype des chaînes de symboles (le plus souvent dans un alphabet binaire) ; les stratégies d'évolution, un vecteur de paramètres réels ; la programmation évolutionniste, des structures quelconques et la programmation génétique, des programmes informatiques, généralement représentés sous forme d'arbres. Ces différentes familles d'algorithmes ont été unifiées récemment pour donner lieu à l'appellation générique d'algorithmes évolutionnistes (le terme d'algorithme évolutionnaire est également utilisé en français) [de Jong, 2006]. Le principe général de ces algorithmes est repris sur la figure 1.

Les algorithmes évolutionnistes appartiennent à la famille des algorithmes dits méta-heuristiques [Michalewicz and Fogel, 1999], au même titre que le recuit simulé [Kirkpatrick et al., 1983], la recherche tabou [Glover, 1989, 1990] ou encore les algorithmes à colonies de fourmis [Dorigo et al., 1996]. Tous ces algorithmes utilisent une métaphore issue de la biologie ou de la physique pour effectuer une recherche approchée. Ils sont décrits par un certain nombre de principes heuristiques à adapter à un problème particulier, d'où le terme de *méta*-heuristique.

ROBOTIQUE ÉVOLUTIONNISTE Dans ce document, nous utiliserons l'expression *robotique évolutionniste* dans son sens le plus large. Nous qualifierons ainsi du terme de robotique évolutionniste toute expérience consistant à utiliser un algorithme évolutionniste dans le cadre du processus de conception d'un robot. Un robot est un système doté de capteurs lui permettant d'acquérir de l'information sur son environnement et d'effecteurs lui permettant d'agir sur ce dernier. Les travaux présentés dans ce document concerneront essentiellement des robots mobiles. Cependant les approches présentées ne sont pas limitées à ce type de robot. Les phénotypes considérés seront le programme contrôlant le comportement du robot, et parfois également sa morphologie, autrement dit sa structure physique. Nous utiliserons l'appellation de *système de contrôle* ou *contrôleur* pour désigner le programme contrôlant le comportement du robot et non système de commande comme il est d'usage de le faire en robotique et en automatique. Ce choix a pour but de marquer la différence fondamentale de notre approche avec l'automatique dont l'objet, en robotique, est d'assurer que l'état du robot suive une trajectoire cible. Aucune trajectoire cible ne sera connue ici. Certains problèmes pourraient se reformuler de cette façon, mais dans la plupart des cas, cela supposerait de disposer d'une connaissance *a priori* que nous souhaitons découvrir plutôt qu'imposer.

La spécificité de la robotique évolutionniste par rapport aux autres applications des algorithmes évolutionnistes tient tout d'abord dans l'évaluation des performances d'une solution potentielle. Elle repose ici sur l'observation du comportement du robot plongé dans son environnement à partir de conditions initiales données, autrement dit sur l'observation d'un système dynamique fortement non-linéaire : des trajectoires similaires peuvent avoir des performances très différentes, par exemple si l'une évite tout obstacle, mais que l'autre implique des collisions. De même, de légers changements dans le système de contrôle peuvent avoir des conséquences dramatiques pour le robot. Une seconde spécificité tient dans l'espace de recherche considéré. S'il est parfois ramené à des génotypes de type chaîne de symboles, vecteur de nombres réels ou programme informatique, beaucoup de travaux s'intéressent à des génotypes spécifiques, adaptés à la représentation du système de contrôle considéré.

1.4 APERÇU DE LA ROBOTIQUE ÉVOLUTIONNISTE

Le terme de robotique évolutionniste a été introduit pour la première fois dans des travaux remontant à 1993 [Cliff et al., 1993], même s'il ne s'agissait pas là des premiers travaux d'utilisation d'algorithmes évolutionnistes dans un cadre robotique [Parisi et al., 1990; Floreano et al., 1991; Beer and Gallagher, 1992; Husbands and Harvey, 1992]. Conçu comme une nouvelle approche de l'intelligence artificielle, la robotique évolutionniste avait pour but d'éviter le recours à des méthodes de conception analytiques. Ces approches nécessitent en effet une décomposition du problème à résoudre (notamment des parties perception, planification et action) non compatible avec la recherche de solutions s'appuyant sur des comportements émergeant des interactions entre les différents constituants du robot [Husbands et al., 1997].

Après presque vingt ans de recherche, des réseaux de neurones ont été générés pour permettre à un robot à roue d'éviter des obstacles et d'aller se recharger automatiquement dans un environnement simple [Floreano and Mondada, 1996] ; des réseaux de neurones ont été également synthétisés pour piloter des robots marcheurs [Kodjabachian and Meyer, 1997; Hornby et al., 2000] ou volants [Mouret et al., 2006; Shim and Husbands, 2007] ou encore pour contrôler des nuées de robots [Gross et al., 2006; Baele et al., 2009]. Des robots marcheurs ont également été conçus automatiquement (morphologie et contrôle) et fabriqués [Lipson and Pollack, 2000]. Pour une présentation détaillée de la robotique évolutionniste, voir [Nolfi and Floreano, 2000; Floreano and Mattiussi, 2008; Floreano et al., 2008b].

Doncieux et al. [2010] classifient les travaux de robotique évolutionniste en quatre catégories (figure 2) :

- ajustement de paramètres (en simulation ou sur robot réel) ;
- conception aidée par évolution ;
- adaptation en ligne évolutionniste ;
- synthèse évolutionniste.

L'ajustement de paramètres repose sur l'utilisation d'algorithmes évolutionnistes dans un cadre d'optimisation « classique ». Les faibles contraintes mathématiques associées à l'efficacité d'algorithmes comme CMA-ES [Hansen and Ostermeier, 2001] ou NSGA-II en multi-objectif [Deb et al., 2002] en font un outil facilement adaptable à la robotique [Fleming and Purshouse, 2002; Floreano et al., 2008a]. Un état de l'art de ce domaine sera présenté dans la section 2.1.

La conception aidée par évolution intervient à une étape plus en amont dans le processus de conception du robot. Les algorithmes évolutionnistes, toujours utilisés dans un cadre d'optimisation classique, sont utilisés pour aider un expert à mieux comprendre le système robotique qu'il cherche à concevoir. Les paramètres optimaux générés, ainsi que les valeurs des objectifs informent l'expert roboticien sur le cahier des charges à adopter pour la conception mécanique et électronique et l'étude des régularités des solutions peuvent également l'aider à extraire des règles de conception sous-jacentes [Deb and Srinivasan, 2007]. Un état de l'art de ce domaine sera présenté dans la section 2.2.

L'adaptation en ligne évolutionniste consiste à ne pas arrêter le processus de conception évolutionniste, mais à le laisser tourner en permanence de façon à permettre une adaptation à des changements drastiques de l'environnement. Ce type d'approche peut conduire à la conception de robots résilients [Bongard et al., 2006]. Si les travaux présentés dans ce manuscrit peuvent être utilisés dans ce contexte, nous ne présenterons pas ici de travaux dédiés à l'évolution en ligne.

La dernière catégorie est la synthèse évolutionniste dont l'objet est de concevoir des structures. L'espace de recherche considéré n'est alors généralement plus $\mathbb{R}^n$, mais l'espace des réseaux de neurones, par exemple, ou celui des morphologies de robots. Le chapitre 3 est consacré à ce sujet.

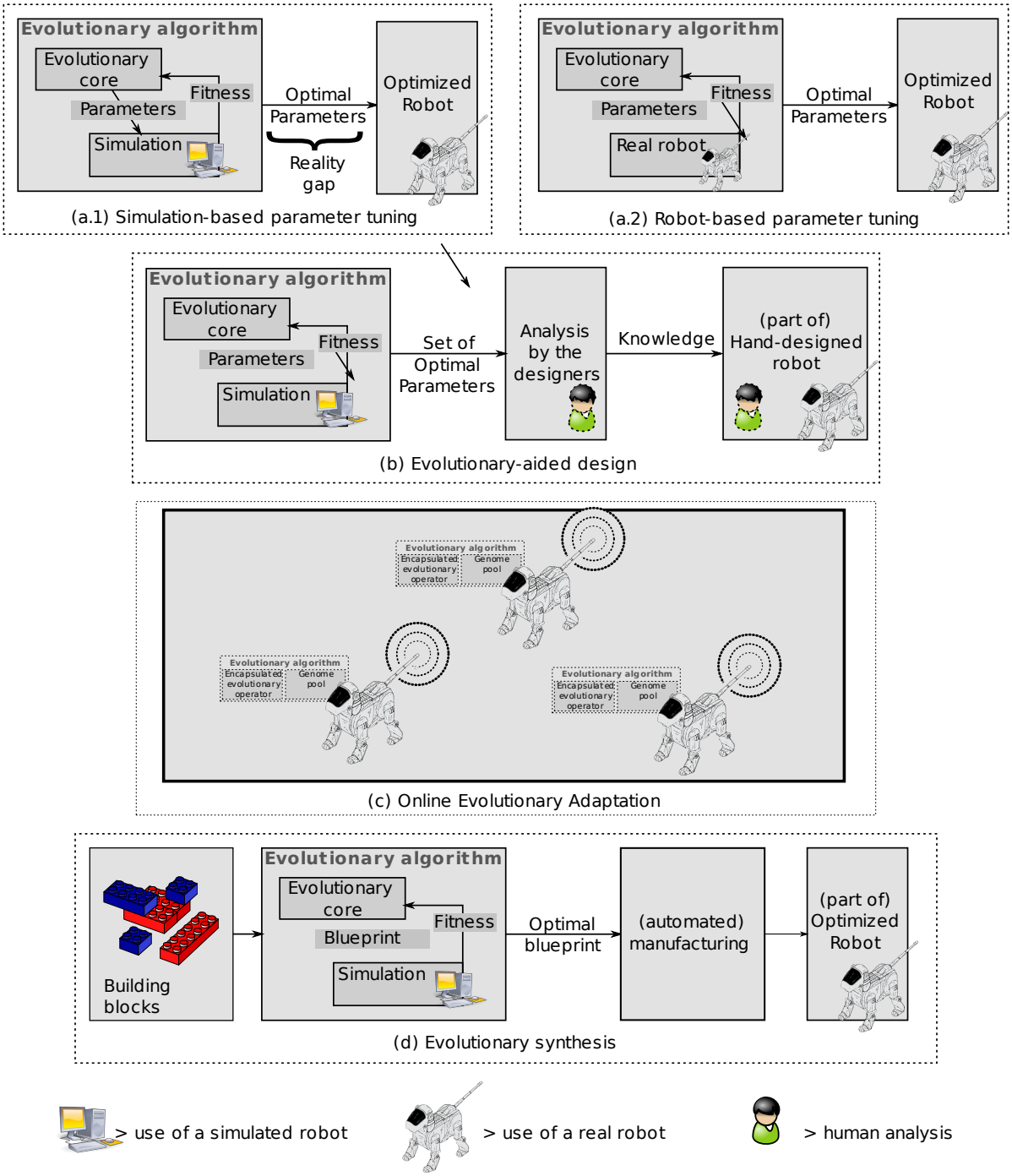


FIG. 2: Vue d'ensemble des différents contextes d'utilisation des algorithmes évolutionnistes dans le cadre d'un processus de conception de robots autonomes.

1.5 GUIDE DE LECTURE DE CE DOCUMENT

Le chapitre 2 présente les travaux que nous avons réalisés et qui restaient dans un cadre d'optimisation classique. Des exemples portant sur le contrôle de robots volants illustreront deux aspects : optimisation de paramètres et conception aidée par évolution.

Le chapitre 3 quitte le cadre formel classique de l'optimisation. Si la notion de fonction de coût ou fitness est conservée, l'espace de recherche considéré est plus vaste. Ce n'est plus $\mathbb{R}^n$, mais l'espace des structures et paramètres possibles pour un graphe orienté (typiquement un réseau de neurones). Après un exemple servant de preuve de concept, seront évoquées des solutions à plusieurs problèmes liés à ce type d'approche, notamment la convergence prématurée ou encore l'évolution de structures modulaires.

Le chapitre 4 se focalise sur les liens entre simulation et réalité et sur les moyens de faciliter le passage de l'une à l'autre.

2 OPTIMISATION ET CONCEPTION ASSISTÉE PAR ÉVOLUTION

MATIÈRES

Ce chapitre présente les travaux réalisés dans un cadre d'optimisation classique. Deux types de travaux reposant sur ces méthodes sont ainsi décrits, tout deux utilisant des algorithmes évolutionnistes. Ils différent sur l'utilisation des résultats. Dans le premier cas, le résultat de l'optimisation marque la fin du processus de conception. Dans le second, le résultat de l'optimisation n'est qu'un résultat intermédiaire dans un processus d'analyse visant une meilleure compréhension d'un système donné. Ce processus peut donner lieu, dans une étape ultérieure, à la conception de nouveaux systèmes qui pourront eux-mêmes être optimisés. Dans les deux cas, l'application reste dans le cadre de la robotique et en particulier, des robots volants.

2.1 OPTIMISATION DU COMPORTEMENT

2.1.1 Introduction

La maximisation ou minimisation d'une fonction de coût sur $\mathbb{R}^n$ peut être utilisée très directement pour concevoir le comportement d'un robot. Il suffit que le système de contrôle du robot et éventuellement sa morphologie soient décrits par un ensemble de paramètres de taille n et que la fonction de coût capture les caractéristiques du comportement souhaité.

Cette utilisation est très fréquente pour sa simplicité de mise en oeuvre. En effet, elle peut se reposer sur des algorithmes d'optimisation standards issus de la communauté des algorithmes évolutionnistes comme les algorithmes génétiques [Goldberg, 1989], les stratégies d'évolution [Back et al., 1991; Schwefel and Rudolph, 1995] ou encore les algorithmes évolutionnistes multi-objectif [Deb, 2001].

Un réseau de neurones de structure fixe permet de générer des comportements très variés simplement en changeant les poids des connexions. De nombreuses expériences ont été réalisées avec une structure choisie et fixée (typiquement un perceptron ou un réseau récurrent complètement connecté), voir par exemple [Wieland, 1991; Beer and Gallagher, 1992; Floreano and Mondada, 1994; Porto et al., 1995; Floreano and Mattiussi, 2001; Reil and Husbands, 2002; Haasdijk et al., 2010].

2.1.2 Évolution de contrôleurs pour exploiter des courants aériens

Dans le cadre du projet ROBUR [Doncieux et al., 2004, 2006], nous nous sommes intéressés à la capacité à exploiter des ascendances de type thermique, des vents remontant le long d'une pente ou encore des gradients de vent, comme savent le faire les oiseaux ou les amateurs de vol-à-voile, parapente ou aéromodélisme (figure 3). Il s'agit ici d'exhiber une trajectoire particulière, dépendante du contexte – ici des mouvements de la masse d'air environnante – pour atteindre un objectif facilement exprimable : maximisation de l'altitude gagnée ou de la durée de vol. Ces travaux sont à situer dans la catégorie « preuve de concept » : ils montrent que le problème considéré est soluble par des approches évolutionnistes.

Si des travaux théoriques ont été menés sur ce sujet, notamment [MacCready, 1958; Cochrane, 1999], ou des observations biologiques [Pennycuick, 1982; Spaar and Bruderer, 1997], peu d'implémentations de contrôleurs permettant d'exhiber ces comportements ont été réalisées. Wharington [1998] propose d'aborder le *soaring* thermique dans un cadre d'apprentissage par renforcement, mais les calculs réalisés sont peu compatibles avec une application embarquée. Allen [2005] propose un contrôleur, également pour le soaring thermique, basé sur des heuristiques permettant d'évaluer la position du centre du thermique et l'a appliqué à un drone réel. Plus récemment, une autre implémentation, exploitant à la fois les travaux de Allen ainsi que ceux de Wharington grâce à des calculs déportés, a permis à un drone de réaliser un vol d'une heure et demi sur une distance de 48km à partir d'un lancement à 140m d'altitude et en exploitant uniquement l'énergie extraite des thermiques de l'environnement [Edwards, 2008].

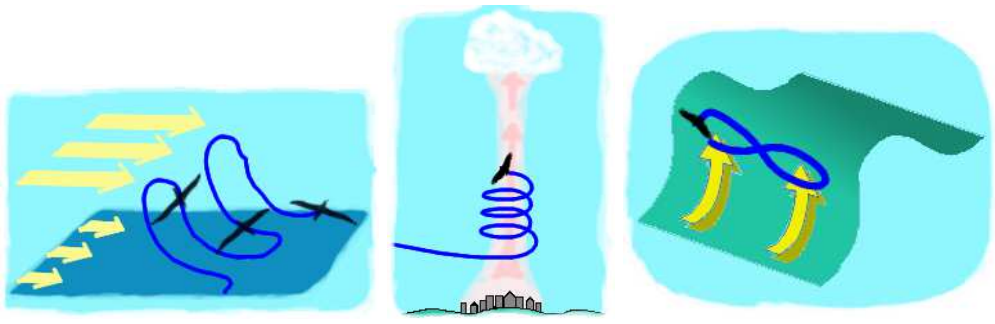


FIG. 3: Les différents types d'exploitation des courants aériens considérés. Gauche : soaring dynamique, milieu : soaring thermique, droite : vol de pente.

SOARING DYNAMIQUE.

Contexte: travaux réalisés pendant le stage de master de Renaud Barate.
Publication(s) associée(s): [Barate et al., 2006]

Les premières expériences réalisées ont pour but de doter un planeur simulé de la capacité à exploiter un gradient de vent [Barate et al., 2006]. Les albatros, par exemple, sont capables de générer une trajectoire en 'S' permettant d'extraire de l'énergie d'un tel gradient et peuvent ainsi voler pendant des jours [Pennycuick, 1982; Sachs, 2005]. Ce comportement se découpe en deux phases :

- une phase ascendante durant laquelle l'oiseau est face au vent et où il prend de l'altitude ;
- une phase descendante durant laquelle l'oiseau est dos au vent et où il plonge pour reprendre de la vitesse.

Cette trajectoire était très difficile à découvrir par une approche automatique, les premiers tests réalisés n'ayant donné aucun résultat probant. Les trajectoires permettant d'extraire de l'énergie d'un gradient de vent doivent être suivies avec une grande précision sans quoi le planeur s'écrase très rapidement. Cette caractéristique rend ce problème très difficile du point de vue de l'apprentissage : la solution cherchée est sur un pic d'efficacité isolé, rien ne pousse donc l'algorithme dans cette direction. Il nous a donc paru pertinent d'utiliser un formalisme qui serait plus facile à comprendre pour un humain, de façon à inclure facilement des connaissances *a priori* dans le processus. Nous avons donc opté pour un contrôleur basé sur des règles floues de type TSK[1] [Takagi and Sugeno, 1985]. Ces contrôleurs contiennent un ensemble de règles de calcul des sorties, exprimées en tant que fonctions linéaires des entrées. À chaque règle est associé un ensemble flou portant sur les entrées et déterminant quand la règle doit être activée. Ce formalisme a l'avantage de ne pas nécessiter d'étape de *déflouification* contrairement à une approche de type Mamdani [Mamdani, 1974] où la prise en compte des entrées comme le calcul des sorties reposent sur des ensembles flous. Les entrées étaient l'altitude du planeur, sa direction relative à celle du vent, son roulis et son tangage ; les sorties étaient les commandes à envoyer aux ailerons, au stabilisateur et à la dérive (figure 4 : gauche).

Le modèle de gradient de vent utilisé est tiré de [Sachs, 2005]. Le modèle de planeur intégre les forces aérodynamiques s'exerçant sur les ailes (constituées de deux panneaux), le stabilisateur et la dérive, le calcul des forces se basant sur les courbes caractéristiques des coefficients de portance et de traînée en fonction de l'angle d'attaque du panneau [Von Mises, 1959; Shevell, 1989]. L'algorithme évolutionniste retenu est un simple algorithme élitiste conservant les vingt individus les meilleurs sur la population de taille cent. Le génotype est le vecteur des paramètres réels des contrôleurs. Les parents du croisement sont choisis par tournoi et chaque paramètre de l'individu résultant est tiré aléatoirement parmi les deux parents. La mutation est de type gaussienne.

Afin d'aider le processus de recherche évolutionniste, nous avons recherché un contrôleur initial d'où pourrait partir l'optimisation. Nous avons donc conçu manuellement un contrôleur de type TSK exhibant approximativement la trajectoire

1 du nom des auteurs Takagi, Sugeno et Kang

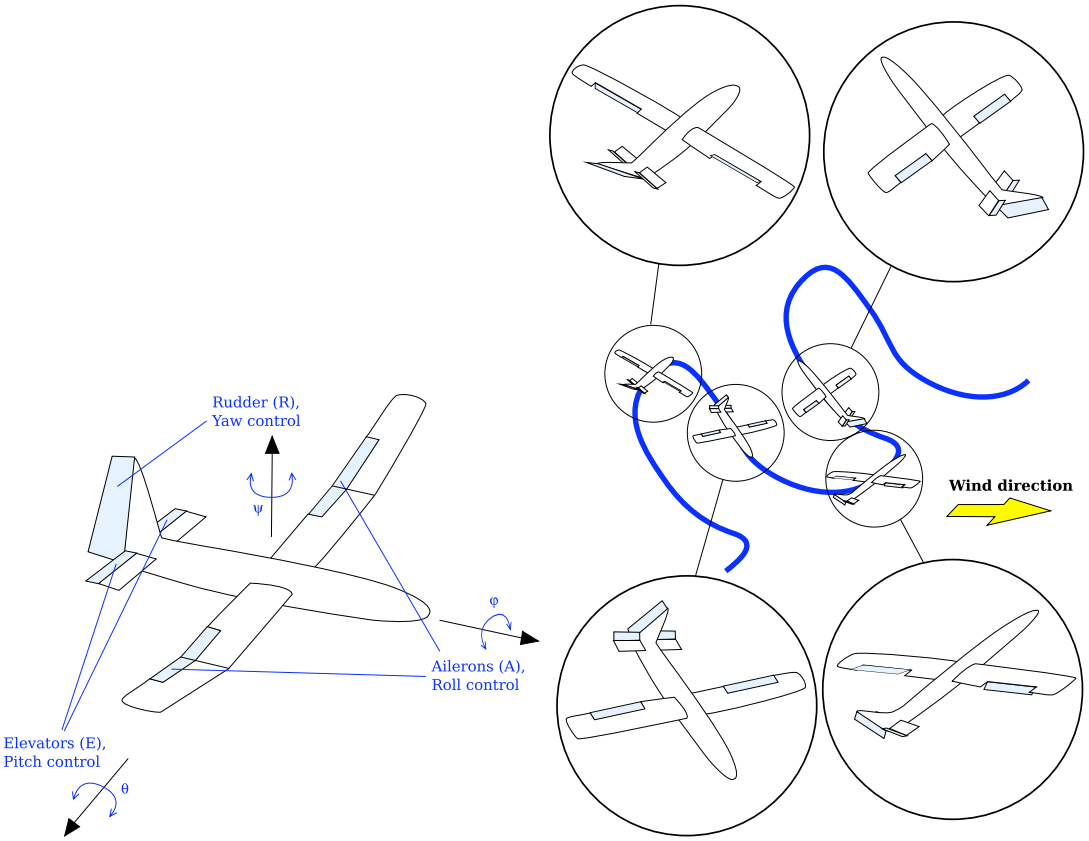


FIG. 4: Gauche : commandes utilisées, droite : trajectoire souhaitée pour le planeur.

représentée sur la figure 4 : droite. Ce contrôleur, composé de deux ensembles flous et de six règles, le tout décrit par vingt-deux paramètres, s'est avéré difficile à régler manuellement. Il parvenait, au mieux, à réaliser quatre cycles pour les vitesses maximales de vent. Plusieurs expériences ont été réalisées en partant de cette solution initiale et en cherchant les meilleurs jeux de paramètres. Dans un premier temps, l'objectif à minimiser était la vitesse du vent nécessaire pour maintenir le planeur pendant mille secondes simulées. Après optimisation, le contrôleur est passé d'un vol de courte durée (suivi d'un crash) avec une vitesse de vent de 20m.s^{-1} à un vol de durée infinie (en simulation) pour une vitesse de vent bien inférieure, à savoir 9.4m.s^{-1}. À titre de comparaison, la vitesse de vent minimale nécessaire à un albatros est estimée à 8.6m.s^{-1}. D'autres aspects ont été étudiés dans le cadre de ces travaux comme les directions possibles relativement à la direction du vent ou encore la robustesse au bruit. Plus de détails sur ces travaux peuvent être trouvés dans [Barate et al., 2006].

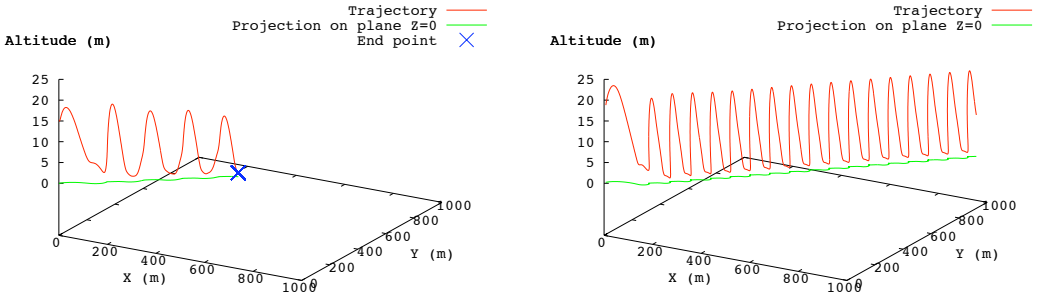


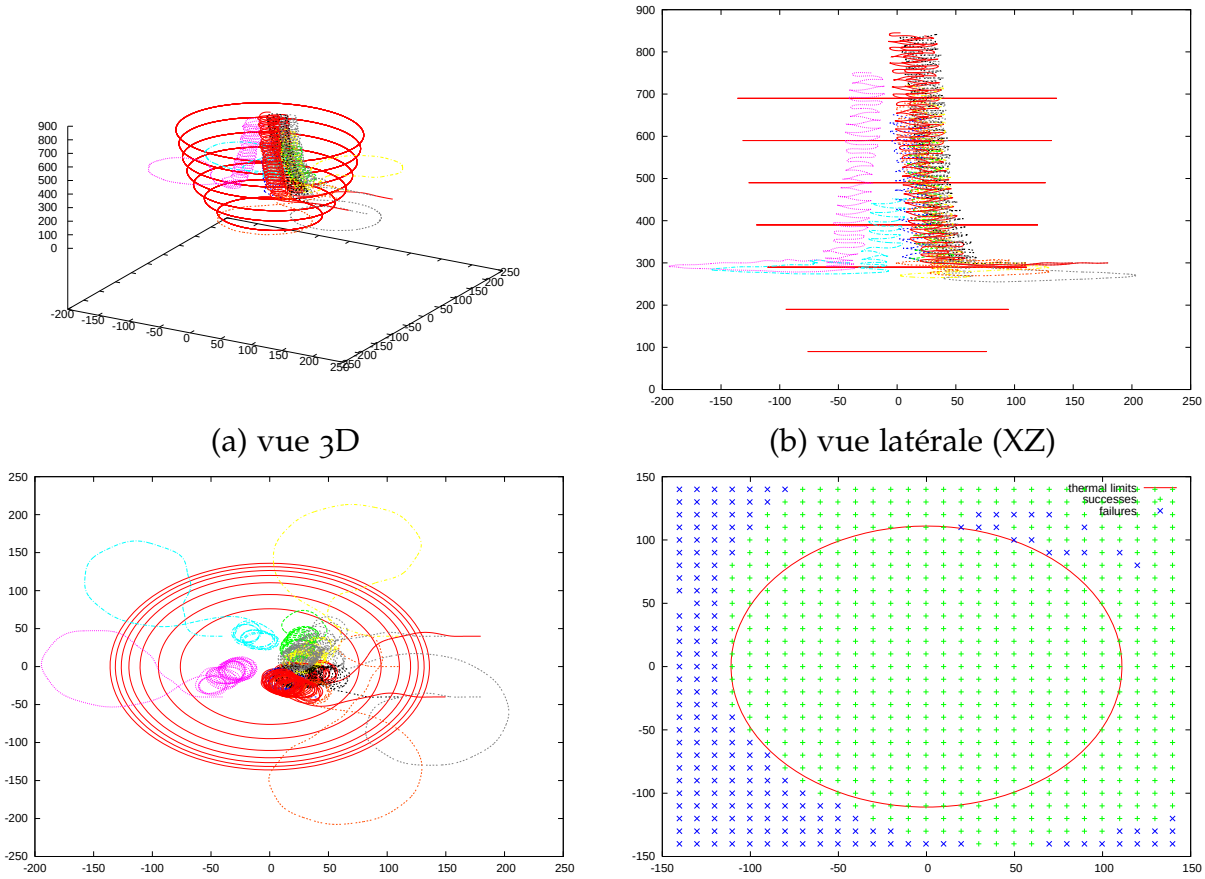
FIG. 5: Gauche : trajectoire générée par le contrôleur initial, droite : trajectoire générée par le contrôleur optimisé par l'algorithme évolutionniste. Le contrôleur optimisé maintient le planeur indéfiniment en l'air et ce pour des vitesses de vent inférieures à celles du contrôleur initial (9.4m.s^{-1} contre 20m.s^{-1})

VOL THERMIQUE ET VOL DE PENTE.

Contexte: travaux initiés pendant les stages de Mathieu Schmitt pour le vol de pente et de Guillaume Tatur pour le vol thermique avant que je ne les reprenne entièrement.
Publication(s) associée(s): [Doncieux et al., 2007]

Ces expériences avaient pour objectif d'étudier l'exploitation d'autres types de courants aériens, à savoir les thermiques, c'est-à-dire les zones d'air chaud entourées par de l'air plus froid et les vents de pente, autrement dit les vents remontant le long de reliefs montagneux. Ces deux phénomènes créent une zone dans laquelle l'air monte. En se maintenant dans ces zones, un drone gagnera de l'altitude sans consommer d'énergie.

La fonction de fitness à maximiser était simplement l'altitude gagnée pendant l'évaluation du drone.



(a) vue 3D (b) vue latérale (XZ)

(c) vue de dessus (XY) (d) Succès & échecs

FIG. 6: (a), (b), (c) Trajectoires générées par un planeur contrôlé par un réseau de neurones optimisés pour le vol thermique et pour onze conditions initiales différentes. Les limites du thermique sont montrées en rouge. (d) Succès et échecs par rapport à la position initiale. Chaque point représente une position initiale (l'altitude étant toujours de 300m). Dans chaque expérience, la vitesse initiale, de 10m.s^{-1}, est orientée vers la gauche. Les coordonnées sont données en mètres.

Contrairement au cas précédent, nous avons opté pour le formalisme des réseaux de neurones, essentiellement parce que nous ne voulions pas donner de contrôleur initial et que dans ce cas, utiliser des contrôleurs de type TSK nécessitait un nombre de paramètres variables si on souhaitait laisser à l'algorithme la possibilité d'ajouter des règles. Les contrôleurs neuronaux utilisés étaient de type perceptron, avec une couche cachée et une structure fixée à l'avance. Les entrées étaient le roulis, le tangage ainsi que la vitesse verticale. Les sorties étaient les commandes à envoyer aux ailerons, à la dérive et au stabilisateur. Dans un premier temps, seule l'altitude moyenne a été utilisée. Cela a donné des comportements très intéressants pour le vol thermique ; cependant, dans le cas du vol de pente, cela n'a généré que des planeurs avançant en ligne droite avec un fort dérapage[2]. Si ce comportement était effectivement optimal sachant que notre pente était d'extension infinie, nous avons ajouté

2 autrement dit avançant « en crabe ».

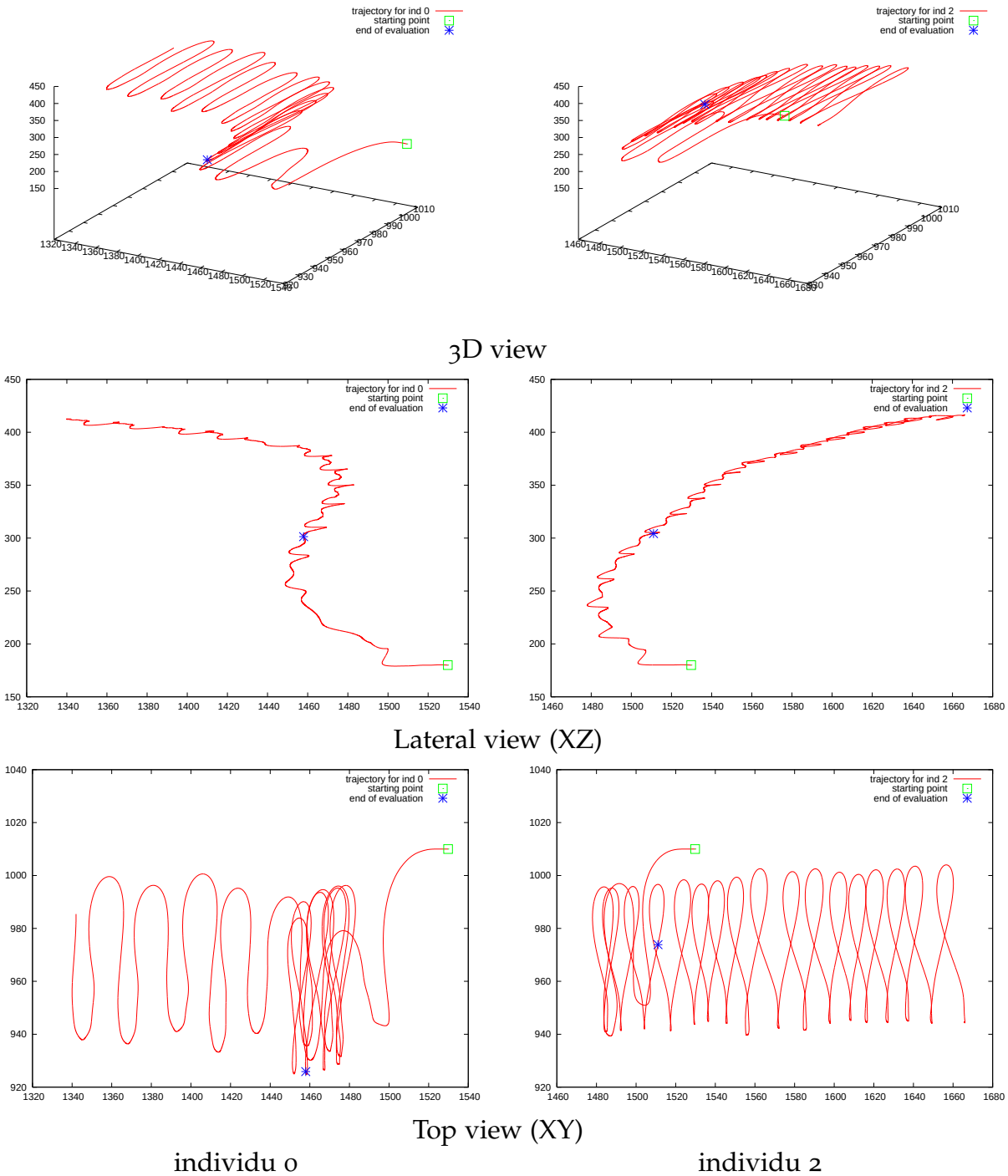


Fig. 7: Trajectoires de planeurs contrôlés par deux réseaux de neurones optimisés pour le vol de pente. La trajectoire est observée sur une durée qui est quatre fois plus longue que pendant l'optimisation (8min vs 2min).

un second objectif dans cette expérience afin de forcer l'algorithme à générer des comportements différents. Cet objectif, à minimiser, visait à encourager le planeur à rester dans une zone d'intérêt donnée.

Afin de ne pas avoir à choisir de pondération entre cet objectif et l'altitude moyenne, nous avons choisi, dans les deux cas un algorithme évolutionniste multi-objectif, ϵ-MOEA [Deb et al., 2003], pour optimiser les paramètres[3]. Nous avons également ajouté un capteur donnant une information de distance à cet axe. Nous avons utilisé le modèle de thermique de Allen [2006].

Dans les deux cas, à partir d'une fonction de fitness simple, ne décrivant pas le comportement souhaité mais son résultat (maximiser l'altitude gagnée) des contrôleurs efficaces ont été générés. Dans le cas du vol thermique, un comportement de type vol en spirale a été observé (figure 6). Ne connaissant pas la position du thermique, le drone explore son environnement avec une trajectoire en spirale à large rayon. Lorsqu'il entre dans le thermique, son comportement change pour suivre une trajectoire toujours de type spirale, mais à petit rayon. Le comportement observé dans le cas du vol de pente est une trajectoire en 'S', le drone gardant toujours le dos à la pente, comme le font les parapentistes ou les amateurs de vol-à-voile. Plus de détails sur ces travaux peuvent être trouvés dans [Doncieux et al., 2007].

Ces travaux ont été réalisés en simulation uniquement. Leur application sur drones réels poserait probablement la question du passage de la simulation à la réalité, qui sera abordée au chapitre 4. Dans ce cas précis, la méthode la plus raisonnable à adopter consisterait à reproduire le comportement généré par l'algorithme évolutionniste avec des méthodes classiques de commande, comme cela a été fait dans [Hauert et al., 2009], par exemple.

Ces travaux montrent qu'un processus d'optimisation « classique » peut résoudre des problèmes de conception de comportement pour des robots, en particulier pour des robots volants. Leur principale limitation tient dans le choix de la structure initiale. S'il est inapproprié, aucune solution intéressante ne sera générée et le concepteur devra rechercher, à tâtons, des structures plus adaptées. Le chapitre 3 s'intéressera à la question de la synthèse de structure. La section suivante utilise toujours le cadre de l'optimisation, mais cette fois plus en amont dans le processus de conception.

2.2 CONCEPTION ASSISTÉE PAR ÉVOLUTION

2.2.1 Introduction

Dans l'expérience portant sur le vol de pente, deux objectifs ont été utilisés. Afin de ne pas avoir à choisir arbitrairement de pondération entre les deux, nous avons opté pour un algorithme intrinsèquement multi-objectif. Ce choix a d'autres avantages

3 la première expérience est de type mono-objectif, mais ces algorithmes donnent de bons résultats également dans ce cas. Cela nous a permis de plus de limiter les différences entre les deux expériences.

par rapport à des méthodes d'agrégation des objectifs comme la somme pondérée, il permet de s'affranchir des limites liées à la convexité du front[4] et génère, pour une puissance de calcul quasiment équivalente à celle d'une expérience mono-objectif, une approximation du front de Pareto dans son ensemble. Le choix d'une solution peut donc être réalisé *a posteriori* en analysant les résultats.

La génération d'un ensemble de solutions plutôt qu'une seule est une caractéristique essentielle des algorithmes multi-objectif[5]. Ces différentes solutions sont des compromis optimaux. Si l'on considère l'exemple d'un robot devant résoudre une tâche tout en étant rapide et peu consommateur en énergie, le front de Pareto contiendra des solutions rapides, mais consommant beaucoup d'énergie ainsi que des solutions très lentes, mais très économes en énergie avec toutes les solutions intermédiaires. Dans un processus de conception classique comme décrit dans la section précédente, une ou plusieurs solutions sont sélectionnées à partir de critères choisis par le concepteur. Dans cette section, nous allons présenter une utilisation alternative de cette information : plutôt que de n'extraire qu'un ou deux points, l'ensemble (approché) des points Pareto optimaux peut être analysé afin d'en déduire des propriétés du modèle computationnel utilisé pour l'optimisation. Le concepteur ne tire alors plus de l'optimisation des solutions en tant que telles, mais des données à analyser pour améliorer la compréhension de son problème ou plus précisément du modèle computationnel utilisé.

2.2.2 Contexte et état de l'art

La finalité d'un processus d'optimisation est habituellement d'améliorer un processus existant ou de réduire des coûts, cependant de nombreux autres usages sont possibles, notamment dans le cas des algorithmes d'optimisation évolutionnistes étant donné leur versatilité.

Ces algorithmes ont ainsi été directement inclus dans des modèles. Dans un cadre biologique et faisant l'hypothèse que la stratégie suivie par les animaux étudiés optimise certains critères, des modèles incluant une optimisation par des algorithmes évolutionniste ont été conçus pour expliquer des données expérimentales [Schmitz et al., 1997]. De même, l'évolution *in silico* a pu être utilisée pour tester des hypothèses portant sur l'évolution du vivant [de Margerie et al., 2006].

Si dans les utilisations précédemment citées, l'optimisation était partie intégrante du modèle, l'optimisation peut également être utilisée à des fins d'étude de modèles computationnels dont la complexité rend parfois difficile leur appréhension par une approche analytique. Nous nous focaliserons par la suite sur les modèles dits idéalisés [Frigg and Hartmann, 2009], autrement dit les modèles simplifiés permettant de rendre un phénomène plus facile à comprendre et analyser. L'acquisition de nouvelles connaissances dans ce cadre se fait sur la base de la construction du modèle

4 La somme pondérées ne peut trouver de compromis appartenant à une zone non convexe du front de Pareto. Cependant, d'autres fonctions de concaténation comme les métriques pondérées avec la norme infini le peuvent.

5 Il n'y a qu'une exception : dans le cas où tous les objectifs sont linéairement dépendants, il n'y a qu'une seule solution.

et des manipulations qu'il permet [Morgan, 1999]. Plus précisément, ce processus passe par les étapes suivantes [Hughes, 1997] :

- dénotation : construction des liens entre modèle et phénomène observé ;
- démonstration : étude du modèle ;
- interprétation : conversion des conclusions obtenues sur le modèle vers le système étudié.

La construction automatique de modèles à partir de données expérimentales est un domaine largement étudié dans le cadre de l'apprentissage, notamment par des approches de type régression. Si des approches supervisées sont souvent utilisées, des algorithmes évolutionnistes sont également intéressants de par leur capacité à manipuler des types de modèles très variés, notamment des systèmes à base de règles ou des réseaux de neurones ou encore de par leur capacité à préparer les données utilisées pour l'apprentissage [Freitas, 2002]. Dans ce cas notamment, ils peuvent être utilisés pour diriger le choix des données afin de construire un méta-modèle du processus considéré [Knowles and Nakayama, 2008; Nakayama et al., 2003; Jin, 2003], dans le cadre d'une approche similaire à celle de l'apprentissage actif.

Les approches évolutionnistes peuvent cependant intervenir également au niveau de l'étude du modèle. Deb a présenté une approche de ce type dans un contexte d'ingénierie [Deb and Srinivasan, 2007] : l'INNOVIZATION, pour INNOVation + optimiZATION. L'analyse des points générés par un algorithme évolutionniste multi-objectif est utilisée pour trouver des règles de conception sous-jacentes. L'optimisation ne peut en effet, au mieux, que trouver un optimal dans l'espace de recherche considéré. Or si la connaissance *a priori* sur le système à concevoir est imparfaite ou incomplète, l'espace de recherche utilisé peut ne pas contenir l'optimum global. L'INNOVIZATION de Deb vise à améliorer cette connaissance afin de mieux cerner l'optimum global. Le résultat de ce processus peut être un système optimal conçu manuellement selon les règles identifiées ou simplement un nouvel espace de recherche plus adapté, à explorer de nouveau avec un algorithme d'optimisation. L'objectif de ce type d'approche est de générer des données qui pourront être analysées par des experts afin d'en tirer de nouvelles connaissances sur le modèle computationnel considéré. Là où Deb suggère une utilisation essentiellement dans un cadre d'ingénierie, nous en proposons une utilisation plus large, offrant aux scientifiques de nouveaux outils et méthodes pour l'étude de modèles computationnels complexes. Nous avons développé ce type d'approche sur un cas d'étude particulier, celui du vol battu, qui nous a permis d'étudier à quel type de question cette méthode permettait de répondre et également comment il fallait la mettre en œuvre.

2.2.3 Étude du vol battu

L'objectif de l'approche que nous avons développée, approche que l'on peut qualifier d'analyse multi-objectif, est de répondre à une question particulière ou de valider ou invalider une hypothèse sur la base des données résultant d'une optimisation multi-objectif. Au-delà du formalisme multi-objectif, nous avons recherché comment visualiser et interpréter les résultats. Trois études ont été menées autour

de l'étude du vol battu par ce type d'approche. Des analyses de modèles issus des neurosciences computationnelles ont également été menées avec cette méthode dans le cadre du projet ANR EvoNeuro [Liénard et al., 2010].

La première étude sur le vol battu, qui est le résultat du stage post-doctoral d'Emmanuel de Margerie, porte sur l'étude de l'influence de la morphologie et s'arrête à une visualisation des données issues de l'optimisation, qui sont comparées aux données biologiques disponibles. Dans la seconde étude, qui est le résultat de la thèse de Mohamed Hamdaoui, co-encadrée par Pierre Sagaut et moi-même, l'analyse des résultats a été plus poussée avec un recours à des outils algorithmiques de fouille de données. La dernière étude, que j'ai réalisée en vue du workshop EvoDeRob, que j'ai co-organisé en octobre 2009 dans le cadre de la conférence IROS, est une étude plus préliminaire dans laquelle nous avons cherché à matérialiser la relation énergie-vitesse complète plutôt que de choisir a priori quelques vitesses particulières.

Étude de l'influence de la morphologie

> *Contexte: travaux réalisés pendant le stage post-doctoral d'Emmanuel de Margerie encadré par moi-même et réalisé avec l'aide de Jean-Baptiste Mouret*
> *Publication(s) associée(s): [de Margerie et al., 2007]*

Dans cette expérience, les paramètres cinématiques et morphologiques des ailes d'un drone à ailes battantes ont été optimisés par des algorithmes évolutionnistes. Une aile avec les quatre degrés de liberté suivants a été considérée (figure 8) :

- dièdre (DI)
- repliement de la partie extérieure de l'aire (*SW*)
- tangage de la partie interieure de l'aile (SINC)
- tangage de la partie extérieure de l'aile (*WINC*)

Les paramètres cinématiques et morphologiques optimisés sont les suivants :

- fréquence de battement : f, $[1 - 10]\mathrm{Hz}$;
- amplitude du dièdre : amp_{DI}, $[0 - 89]\mathrm{deg}$;
- amplitude de repliement de la partie extérieure de l'aile : amp_{SW}, $[0 - 89]\mathrm{deg}$;
- déphasage du repliement : $deph_{SW}$, $[0 - 500]\%$ de la période ;
- angle de référence du calage : ref_{SINC}, $[-20 - 20]\mathrm{deg}$;
- amplitude du tangage de la première et seconde partie de l'aile : amp_{SINC} , $[0 - -69]\mathrm{deg}$ et amp_{WINC}, $[-20 - 20]\mathrm{deg}$;
- déphasage du tangage de la première et seconde partie de l'aile : $deph_{SINC}$, $deph_{WINC}$, $[0 - 500]\%$ pour les deux ;
- surface alaire $[0,1 - 0,4]\mathrm{m}^2$;
- rapport d'aspect de l'aile $[4,5 - 10]$ (paramètre de forme, rapport entre l'envergure et la corde[6]).

6 la corde est la largeur de l'aile.

Sur la base de ces paramètres, la cinématique des ailes est la suivante :

$$\mathrm{DI} = \mathrm{amp}_{\mathrm{DI}} \sin(2\pi \mathrm{ft}) \tag{2.1}$$

$$SW = \mathrm{amp}_{SW} \left(\frac{1}{2} + \frac{1}{2} \sin(2\pi(\mathrm{ft} + \mathrm{deph}_{SW})) \right) \tag{2.2}$$

$$\mathrm{SINC} = \mathrm{ref}_{\mathrm{SINC}} + \mathrm{amp}_{\mathrm{SINC}} \sin(2\pi(\mathrm{ft} + \mathrm{deph}_{\mathrm{SINC}})) \tag{2.3}$$

$$WINC = \mathrm{ref}_{WINC} + \mathrm{amp}_{WINC} \sin(2\pi(\mathrm{ft} + \mathrm{deph}_{WINC})) \tag{2.4}$$

$$\tag{2.5}$$

Le profil choisi est le Selig 4083 [Selig, 1997], qui a un aspect proche des profils des ailes des oiseaux. Plusieurs expériences ont été conçues. A chaque expérience est associée une vitesse cible particulière parmi les valeurs suivantes : $\{6, 8, 10, 12, 16, 20\}\,\mathrm{m.s}^{-1}$. Des optimisations différentes étant lancées dans chaque cas, des jeux de paramètres spécifiques à chaque vitesse seront donc trouvés. Les objectifs à optimiser sont les suivants :

- distance entre la trajectoire observée du drone et une trajectoire idéale, correspondant à une ligne droite réalisée à la vitesse cible de l'expérience (à minimiser) ;
- puissance mécanique instantanée produite aux différentes articulations (à minimiser).

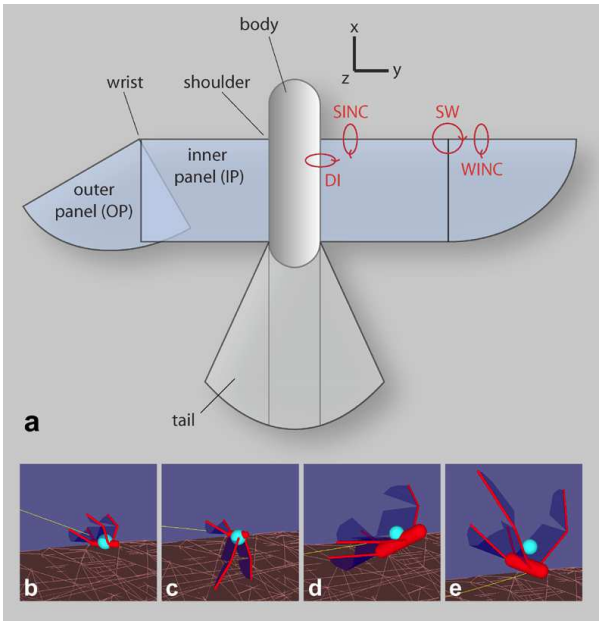


FIG. 8: (a) Morphologie du drone à ailes battantes considéré dans les travaux d'optimisation simultanée de la morphologie et de la cinématique de battement. (b)-(e) : exemples de morphologies.

Les questions posées par cette étude sont les suivantes :

- quelles sont les consommations énergétiques atteignables et pour quels paramètres ?

– quelle est l'influence des degrés de libertés reliant les deux panneaux ?

A des fins de validation et dans l'attente d'essais en soufflerie, les résultats obtenus pour les différents paramètres ont été comparés aux données biologiques lorsque celles-ci étaient disponibles. Les résultats obtenus concernant la consommation énergétique ont été représentés sur la figure 9. Les consommations énergétiques associées aux vitesses supérieures à $10 m.s^{-1}$ sont compatibles avec celles observées chez les oiseaux [Tobalske et al., 2003], alors que celles obtenues pour 8 et surtout $6 m.s^{-1}$ sont irréalistes. Notre hypothèse concernant ces résultats est que le simulateur utilisé [Druot, 2004] ne prend pas en compte certains phénomènes aérodynamiques qui pourraient être cruciaux à ces vitesses. Une autre hypothèse serait que le drone considéré ne dispose pas de degrés de liberté suffisants pour voler efficacement à ces vitesses.

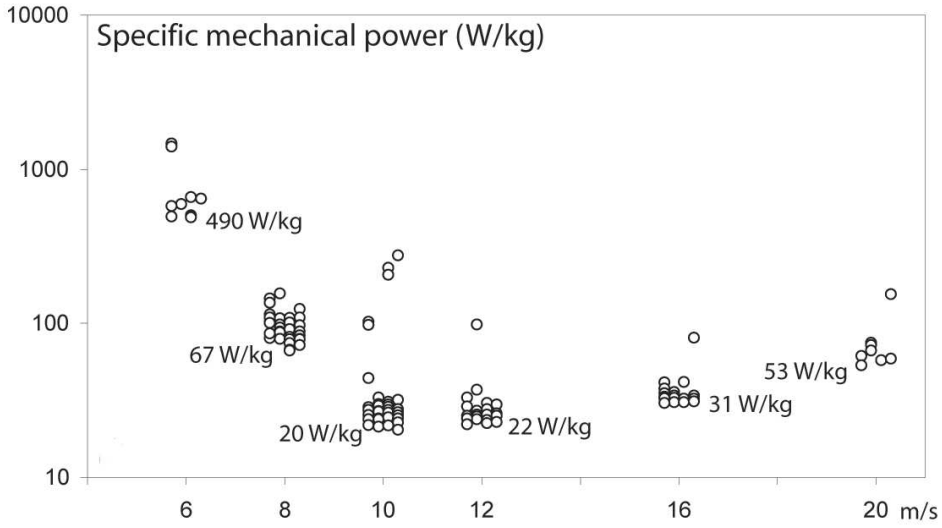


FIG. 9: Puissances mécaniques minimales obtenues par les différentes optimisations réalisées.

Afin de déterminer l'influence des degrés de liberté reliant les deux panneaux des ailes, des optimisations ont été lancées dans lesquelles ces degrés de liberté ont été bloqués, tout autre aspect étant identique par ailleurs. L'impact de ces configurations sur la consommation énergétique a été représenté sur la figure 10. Ces résultats montrent la capacité de ces deux degrés de liberté à réduire la consommation énergétique, l'incidence relative entre les deux panneaux ayant un rôle prépondérant.

Étude de l'efficacité propulsive

Contexte: travaux réalisés dans le cadre de la thèse de Mohamed Hamdaoui, thèse encadrée par Pierre Sagaut et moi-même
Publication(s) associée(s): [Hamdaoui, 2009; Hamdaoui et al., 2010]

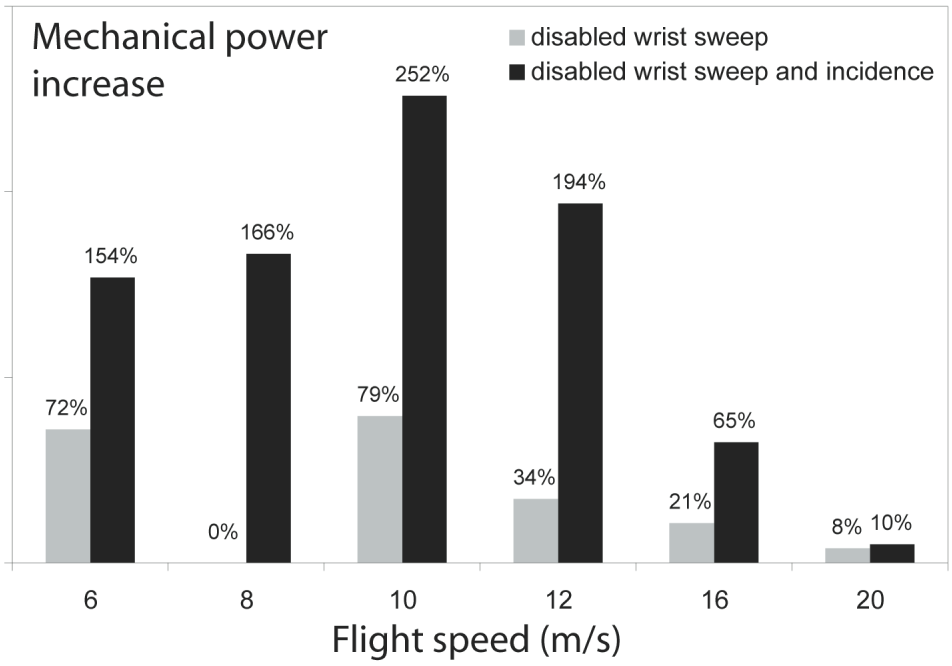


FIG. 10: Augmentation de la puissance mécanique requise lorsque les degrés de libertés reliant les deux panneaux de l'aile sont bloqués : repliement d'aile (wrist sweep) et incidence relative entre les deux panneaux.

Les travaux de thèse de Mohamed Hamdaoui [Hamdaoui, 2009] ont abordé la question de l'optimisation de l'efficacité propulsive η d'un drone à ailes battantes définie comme suit :

$$\eta = \frac{\text{Force Propulsive} \times \text{Vitesse}}{\text{Puissance pour mouvoir les ailes}}$$

Seule la cinématique des ailes a été prise en compte. Les objectifs considérés étaient les suivants :

- efficacité propulsive η (à maximiser)
- différence entre portance et poids (à minimiser)
- norme du moment global (à minimiser).

Deux cas de figure ont été considérés :

- cas 1 : aile monolithique pilotée en dièdre (DI) et tangage (SINC), 7 paramètres à optimiser
- cas 2 : aile constituée de deux panneaux, dièdre (DI) et tangage du panneau intérieur (SINC) et tangage relatif du panneau extérieur (WINC), 9 paramètres à optimiser.

L'objectif de ces travaux, par rapport aux travaux portant sur l'évolution de morphologie, était de pousser plus avant dans l'analyse des données issues du front de Pareto. Une configuration plus simple que dans le cas précédent (pas de repliement) a donc été choisie avec une morphologie fixe. La thèse a ainsi comporté, en plus de la partie aérodynamique et optimisation, une partie sur la fouille de données afin

d'identifier les outils algorithmiques adaptés pour exploiter au mieux les informations disponibles.

Le modèle utilisé pour simuler le comportement du drone à ailes battantes est celui de DeLaurier [1993]. Les ailes sont rectangulaires, dotées d'un profil LPT110A avec une envergue de un mètre. La masse ($0,67$kg) et la surface alaire ($0,15$m^2) sont obtenues via des lois d'échelle pour les oiseaux [Shyy et al., 1999].

Le drone est supposé être à sa vitesse de croisière (la force de traînée est choisie égale à la force de traction). Trois vitesses de croisière sont considérées : 6, 10 et 14m.s^{-1}.

Les questions étudiées sont les suivantes :
- quelle est la vitesse permettant la meilleure efficacité ?
- quelle est la configuration la plus efficace : 1 ou 2 panneaux ?
- quels sont les relations entre les paramètres cinématiques et les objectifs optimisés ?
- quels sont les paramètres cinématiques qui affectent le plus une solution choisie ?

Les analyses réalisées ont reposé essentiellement sur des cartes auto-organisatrices de Kohonen [Kohonen, 2001] et des représentations de type *scatter-plot matrix* [Roudenko, 2004]. Le choix d'une solution particulière utilisait la méthode des diagrammes de niveau [Blasco et al., 2008] et la caractérisation des points choisis sur des arbres de décision [Larose, 2005].

La comparaison des fronts de Pareto générés pour les six différentes expériences (trois vitesses sur deux configurations d'ailes) indique que la vitesse de 14m.s^{-1} est la plus intéressante pour cette configuration de drone (figure 11), car le front associé domine les deux autres sur les objectifs d'efficacité propulsive et de quotient entre portance et poids.

Le cas à deux panneaux s'est révélé meilleur que le cas des ailes à un seul panneau sur m_c^* (quotient entre portance et poids) et C_m^*, confirmant ainsi les résultats précédents.

Les résultats ont montré que, dans le cas 1, la solution optimale choisie est sensible en priorité à la fréquence et à l'angle de référence du panneau. Dans le cas 2, c'est l'amplitude du tangage du panneau intérieur ainsi que son angle de calage qui sont les plus caractéristiques. Ces conclusions ont été tirées de l'étude par des arbres de décision du voisinage du point optimal choisi par rapport au reste du front de Pareto. Cette analyse ne permet donc pas de caractériser une solution optimale par rapport aux autres solutions, mais plutôt une solution optimale par rapport à d'autres solutions optimales.

Par rapport à la validité de la méthode considérée, nous retiendrons que les dépendances trouvées se sont avérées compatibles avec l'état de l'art pour la cas de l'aile à un seul panneau. Le cas de l'aile à deux panneaux a été très peu étudié et les dépendances trouvées dans ce contexte sont donc originales et pourront donner lieu à des expérimentations ultérieures afin d'en valider les prédictions sur un dispositif réel. Les solutions optimales trouvées pour le cas 1 utilisent une puissance massique de $27,6$W/kg, pour un coefficient de trainée de $0,05$ et une fréquence de $3,58$Hz. Pour le cas 2, la puissance massique de la solution retenue est de $25,37$W/kg pour une fréquence de $4,08$Hz. Ces résultats ayant été obtenus en simulation et dans l'attente

de validations expérimentales, il est intéressant, comme pour l'étude précédente, de mettre ces valeurs en perspective des seules données disponibles, à savoir celles des biologistes ayant étudié le vol des oiseaux. Les puissances par unité de masse sont compatibles avec celles des oiseaux qui varient entre 17 et 30W/kg [Tobalske et al., 2003]. La fréquence enfin se rapproche de celle du *Phaeton Lepturus*, à savoir 4, 22Hz, oiseau qui a des caractéristiques morphologiques similaires à celles retenues pour le drone considéré[7].

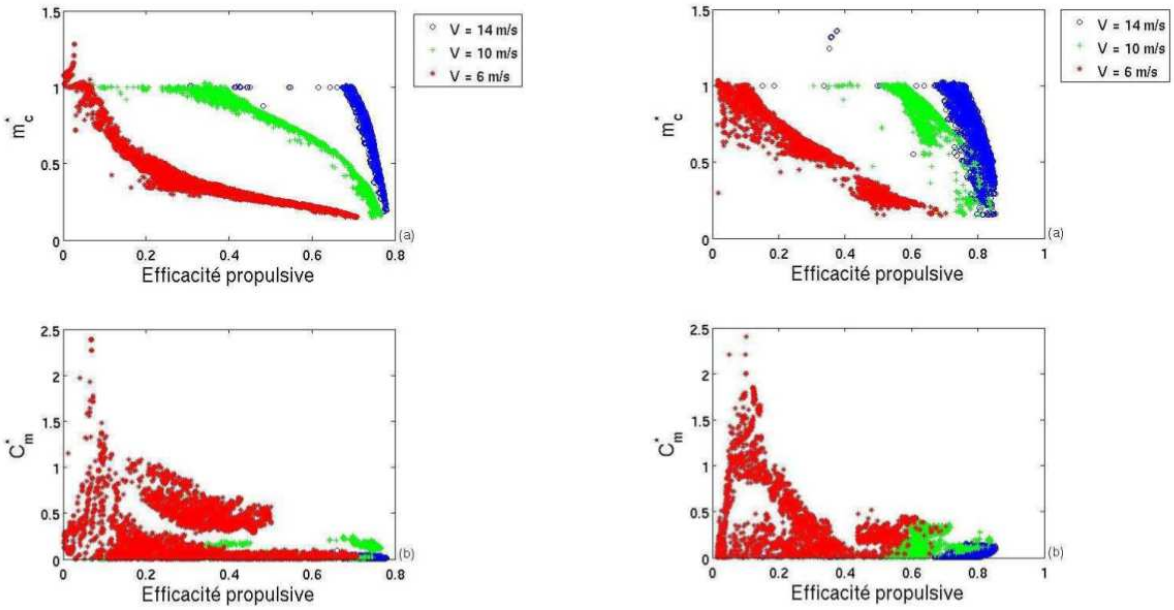


FIG. 11: Comparaison des fronts de Pareto générés pour des vitesses de 6, 10 et 14m.s^{-1} pour la configuration d'ailes à un seul panneau (gauche) et à deux panneaux (droite). m_c^* est le quotient entre portance et poids, C_m^* est le quotient entre C_m et C_{m0} (coefficient de moment à portance nulle).

Étude des caractéristiques énergie-vitesse

> *Contexte: Travaux réalisés par moi-même et complétés avec l'aide de Mohamed Hamdaoui.*
> *Publication(s) associée(s): [Doncieux, 2009; Doncieux and Hamdaoui, 2010]*

Dans les deux études précédentes, un petit nombre de vitesses ont été choisies *a priori*. Afin de mieux matérialiser la dépendance entre les paramètres et la vitesse, d'autres expériences permettant de générer un continuum de vitesses ont été

7 Avec un rapport d'aspect de 6,7 contre 10,06 pour notre drone et une charge alaire est de 43,8 contre 42,85, le Phaeton Lepturus est l'oiseau le plus proche du drone parmi les oiseaux mentionnés dans [Pennycuick, 1990].

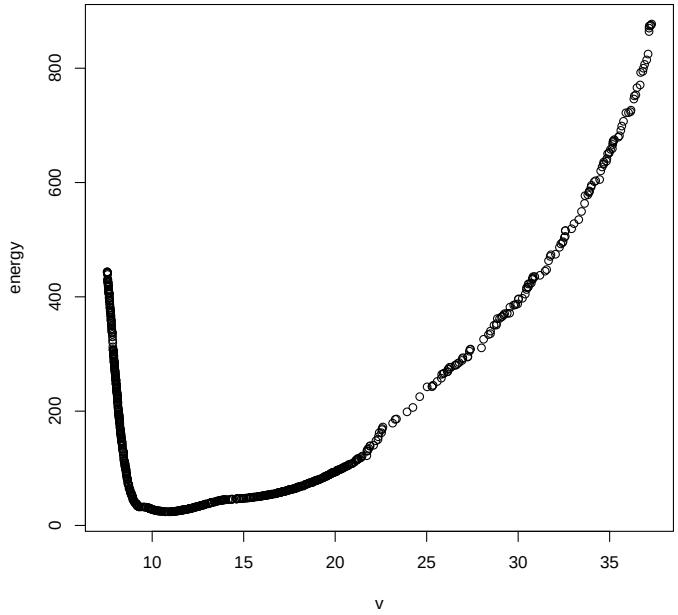


FIG. 12: Points Pareto-optimaux extraits de trois optimisations réalisées pour la minimisation de vitesse et de trois autres réalisées pour la maximisation de vitesse. La fusion entre les fronts issus des deux expériences est réalisée au point de plus faible énergie. Chaque point représente un jeu de paramètres particulier. Abscisses : vitesse en $m.s^{-1}$, ordonnées : énergie instantannée en W.

conçues. Outre la puissance mécanique à minimiser, nous avons donc considéré la vitesse comme objectif à optimiser. Afin de matérialiser des points avec le plus de vitesses différentes possible, deux expériences ont été définies : la première cherchait à minimiser la vitesse et la seconde à la maximiser. Les fronts de Pareto ainsi générés seront fusionnés et matérialiseront ainsi directement la relation énergie-vitesse, relation caractéristique du drone choisi. Celui-ci a une envergure de $1,93m$ pour une masse totale de $1,3kg$ et une surface alaire de $0,407m^2$. La simulation utilisée est celle décrite dans [de Margerie et al., 2007; Druot, 2004] .

La courbe énergie-vitesse ainsi matérialisée est tracée sur la figure 12. Les fronts de Pareto générés par les deux types d'optimisations réalisées (maximisation et minimisation de vitesse) se fusionnent sans discontinuité au point d'énergie minimale, matérialisant donc une relation continue entre énergie et vitesse sur toute la gamme des vitesses atteignables. Le point de consommation énergétique minimale ($24W$) a été atteint pour une vitesse de $10,7m.s^{-1}$. Les vitesses atteignables pour une consommation énergétique inférieure à $100W$ varient entre $8,5$ et $20,4m.s^{-1}$. Les valeurs des paramètres associées à chacun des points optimaux correspondant sont tracées sur la figure 13. Ces courbes peuvent notamment être utilisées pour proposer un modèle réduit du drone ou pour dimensionner un prototype. Nous les avons utilisées pour générer une loi de commande en vitesse en boucle ouverte : les paramètres

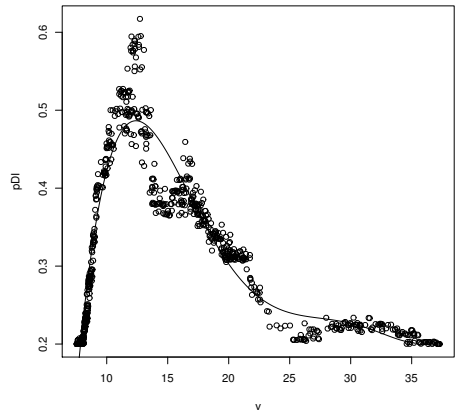
cinématiques appliqués à un instant donné étant ceux de la vitesse souhaitée. Voir [Doncieux, 2009; Doncieux and Hamdaoui, 2010] pour plus de détails.

2.3 DISCUSSION

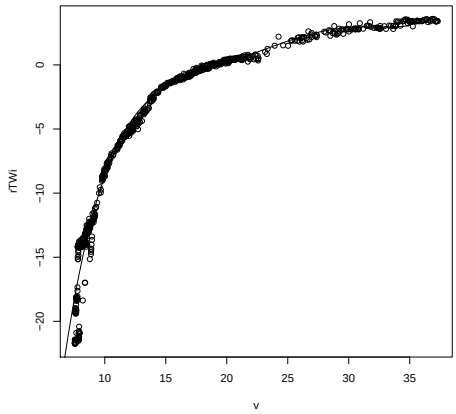
L'analyse multi-objectif constitue une étape intermédiaire, visant à aider un expert à mieux maîtriser le système étudié. Une telle approche ne permet pas une conception automatique directe, mais plutôt une conception itérative dans laquelle l'expertise est construite progressivement. Cette méthode se situe donc dans le cadre d'une approche classique de conception. Elle correspond simplement à un autre usage d'outils algorithmiques largement utilisés par ailleurs.

Est-ce que des algorithmes autres que algorithmes évolutionnistes auraient pu être utilisés ? Est-ce que d'autres algorithmes ou méthodes auraient été plus adaptés ? Est-ce que les performances auraient été meilleures ? Les fonctions de coût optimisées évaluaient la performance d'un robot en interaction avec son environnement, autrement dit elles reposent sur l'observation d'un système dynamique, instable, fortement non-linéaire, ce qui, sans hypothèses simplificatrices, nous oriente vers des méthodes d'optimisation approchées. Les algorithmes de type méta-heuristique sont donc un choix par défaut. Parmi ces méthodes, les algorithmes évolutionnistes ont plusieurs avantages. Leur fonctionnement en parallèle les rend moins sensibles que d'autres algorithmes aux extrema locaux, mais leur principal intérêt tient dans leur maturité et surtout leur efficacité dans un cadre multi-objectif. Cependant, il n'est pas impossible qu'un spécialiste d'une autre méta-heuristique puisse obtenir des résultats similaires. Fondamentalement, si d'autres méta-heuristiques s'avéraient plus efficaces dans un avenir proche (essaims de particules, algorithmes de fourmis ou autre), leur utilisation serait à privilégier, l'approche étant plus ici dans la formulation d'un problème et dans l'exploitation des résultats que dans l'algorithme permettant de passer de l'un à l'autre. Les algorithmes utilisés sont d'ailleurs des algorithmes standards qui n'ont nécessité aucune adaptation particulière.

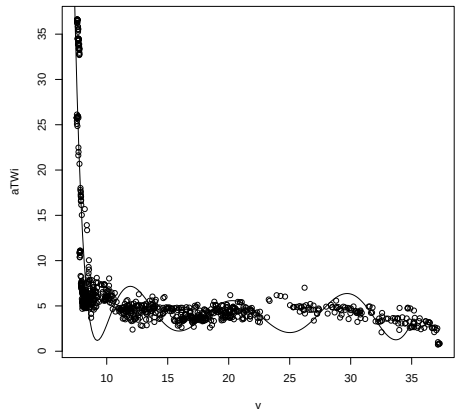
La motivation initiale de ces travaux est de concevoir des contrôleurs pour des robots susceptibles d'exploiter au mieux leurs capacités sensori-motrices tout en tendant vers des aptitudes de plus en plus cognitives. Les expériences sur le soaring montrent que l'on a pu exploiter, avec une structure de contrôle très simple, l'interaction du drone avec son environnement pour générer un comportement complexe. Cependant, les comportements atteignables resteront limités par le choix initial de la structure du neuro-contrôleur ou du système à base de règles. L'absence de récurrence, par exemple, empêche tout comportement nécessitant une forme de mémoire, ce qui limite *de facto* les capacités cognitives accessibles. Une approche pourrait consister à partir d'une structure conséquente, par exemple un réseau de neurones avec plusieurs dizaines de neurones cachés, voire plus, et entièrement connecté. Si une telle approche est envisageable, l'efficacité obtenue sera probablement déce-



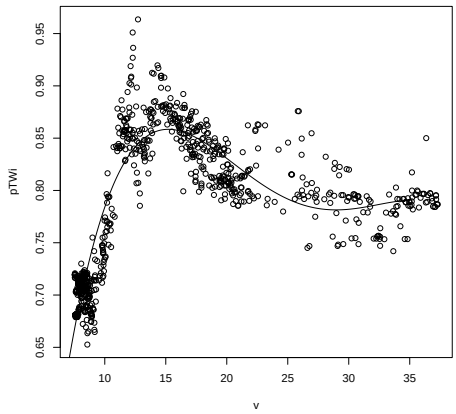
(a) période du dièdre en s



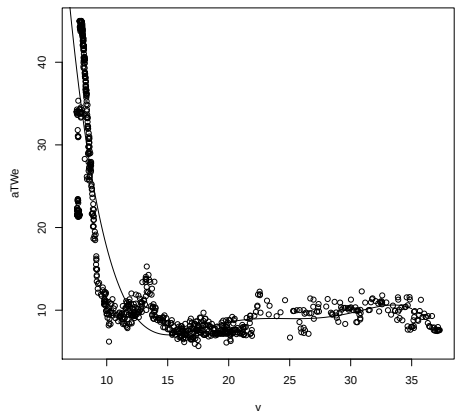
(b) angle de référence du tangage interne en deg



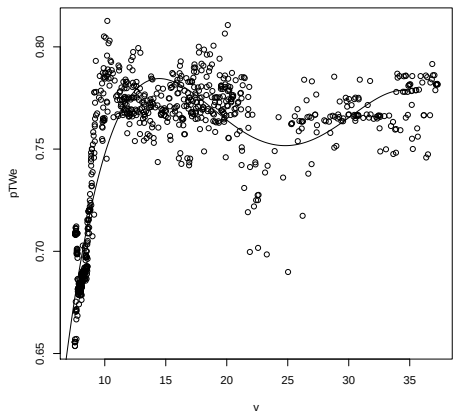
(c) amplitude du tangage interne en deg



(d) déphasage du tangage interne



(e) amplitude du tangage externe en deg



(f) déphasage du tangage externe

FIG. 13: Dépendance empirique de quelque uns des paramètres optimisés à la vitesse. Chaque point représente une solution Pareto optimale.

vante[8]. Des comparaisons ont été réalisées entre un réseau de neurones à la structure conçue par évolution et un réseau de neurones à la structure fixe (la structure choisie étant celle obtenue dans une expérience avec évolution de la structure, c'était donc a priori une structure adaptée) [Stanley and Miikkulainen, 2004]. La synthèse évolutionniste de structure s'est révélée plus efficace que l'évolution des paramètres uniquement, alors que l'espace de recherche dans le premier cas était plus vaste que dans le second. L'hypothèse avancée pour expliquer cette différence est qu'une évolution de structure permet d'explorer des structures de complexité croissante, facilitant ainsi la recherche. Cela nous a conduit à étudier le problème de la synthèse automatique que nous allons aborder dans le chapitre suivant.

8 Cette remarque concerne une optimisation d'un réseau quelconque. D'autres alternatives dans lesquels le réseau respecte certaines contraintes, notamment l'approche des ESN, pourraient se révéler prometteuses, nous y reviendrons.

3 SYNTHÈSE ÉVOLUTIONNISTE

MATIÈRES

3.1 INTRODUCTION

Sims [1994] a présenté, dans une expérience devenue emblématique, l'utilisation de méthodes évolutionnistes pour concevoir des créatures artificielles capables d'accomplir des tâches simples de locomotion, saut, nage ou phototaxie. L'algorithme évolutionniste avait pour tâche de générer la morphologie et le système de contrôle de robots plongés dans un environnement virtuel. Le projet GOLEM est une extension de ces travaux incluant un transfert des robots générés par évolution dans le monde réel [Lipson and Pollack, 2000]. Sans aller jusqu'à la conception de la morphologie, des systèmes de contrôle pour des robots marcheurs hexapodes ou octopodes [Gruau, 1995; Kodjabachian and Meyer, 1997; Filliat et al., 1999] ou encore des robots nageurs [Ijspeert and Kodjabachian, 1999] ont également été conçus par ces méthodes, voire [Meyer et al., 2003] pour une revue. Plus récemment, des applications à la locomotion bipède [Allen and Faloutsos, 2009; Lehman and Stanley, 2010] ont également été considérées. Tous ces travaux ont en commun la recherche d'une structure, que ce soit pour le contrôle du robot uniquement ou également pour sa morphologie. Ils sortent donc du cadre classique de l'optimisation, qui s'intéresse à l'optimisation de vecteurs de paramètres *de taille fixe*. Ici la taille des solutions potentielles est variable, ce que permettent les algorithmes évolutionnistes, à partir du moment où des opérateurs génétiques dédiés et efficaces y sont associés.

Dans le cas de l'optimisation de paramètres, il est raisonnable d'estimer que l'algorithme parviendra à explorer suffisamment pour trouver un extremum local sa-

tisfaisant, si ce n'est l'extremum global. Dans le cas de l'optimisation de structure, l'algorithme évolutionniste doit trouver à la fois une structure satisfaisante *et* les bons paramètres pour en tirer partie. Si une bonne structure est générée, mais avec des paramètres inadaptés, la solution risque d'être rejetée dès qu'elle apparaît. À l'inverse une structure très robuste aux paramètres aura plus de chances d'être conservée et donc ensuite optimisée. Il semble donc probable que la recherche, dans ce type d'espace, soit plus difficile que dans un espace de paramètres. La convergence prématurée vers des extrema locaux non pertinents serait un symptôme de déficience de la recherche, l'algorithme se concentrant sur des structures très simples et insuffisantes pour résoudre la tâche de manière satisfaisante.

Ce commentaire suscite plusieurs remarques sur la recherche dans ce type d'espace :

1. est-il possible de trouver des solutions intéressantes ?
2. si le problème de recherche indiqué ci-dessus se confirme, peut-on le résoudre et si oui, comment ?
3. avec l'objectif de tendre vers des comportements de complexité croissante, quel type de représentation utiliser ?

La première question est un prérequis à ce type d'étude. Si sur une tâche complexe, un algorithme évolutionniste peut proposer des solutions originales, cela pourra motiver d'autres travaux cherchant à étendre la portée et l'efficacité de ces algorithmes. La section 3.2 présente une expérience portant sur le contrôle d'un drone à ailes battantes. Cette section se termine par une expérience montrant les limites des approches incrémentales qui étaient utilisées à cette époque.

La seconde question implique d'étudier les limites de ces approches et de trouver comment les dépasser. Le problème du démarrage, qui entraîne une convergence prématurée vers des solutions trop peu efficaces est abordé dans la section 3.3 sous l'angle de la pression de sélection. Des solutions à ce problème sont proposées qui ne modifient que la fonction de fitness, typiquement en lui ajoutant divers objectifs, de façon à résoudre des problèmes complexes avec un codage direct aussi simple que possible.

La troisième question est celle du passage à l'échelle. Pour des comportements complexes, nécessitant, par exemple, des capacités cognitives de mémorisation, de cartographie, de sélection de l'action ou encore d'analyse de scène – comportements qui sont bien au-delà de ce qui peut être obtenu par une approche évolutionniste à ce jour –, l'étude de la modularité semble une voie prometteuse compte tenu de son omniprésence tant en biologie qu'en ingénierie. Cela pose la question de comment utiliser cette notion dans un codage et éventuellement quelle conséquence cela peut avoir sur la pression de sélection. Ces questions sont abordées dans la section 3.4

3.2 EXPÉRIENCES PRÉLIMINAIRES

L'évolution de la structure et des paramètres d'un système de contrôle est un objectif ambitieux. L'espace de recherche considéré est extrêmement vaste et seule

une toute petite partie peut être explorée en un nombre de générations raisonnable compte tenu des puissances de calcul actuelles. Cette section présente deux expériences préliminaires à finalité différente. La première expérience est une « preuve de concept » ne visant à rien d'autre qu'à montrer que l'approche de synthèse évolutionniste *peut* générer des solutions non triviales pour un problème complexe. La seconde expérience suit une approche incrémentale pour aborder un problème de complexité supérieure. De par ces résultats, cette expérience montre les limites de ce type d'approche, motivant ainsi les travaux réalisés par la suite.

Ces deux expériences s'intéressent au vol battu, dans un premier temps en ligne droite, puis avec la possibilité de faire des virages dans un second temps. Une étude de ce mode de locomotion par une approche de type conception aidée par l'évolution est incluse dans la section 2.2.3. L'objet des expériences présentées dans cette section est de l'aborder du point de vue de la synthèse évolutionniste.

3.2.1 Vol battu en ligne droite

Contexte: travaux réalisés pendant le stage de master de Jean-Baptiste Mouret.
Publication(s) associée(s): [Mouret et al., 2004]

L'objectif de cette première expérience est de générer un contrôleur neuronal permettant à un robot à ailes battantes simulé d'optimiser les objectifs suivants :

- vol à vitesse constante
- vol à altitude constante
- vol stable
- vol économe en énergie

Le robot volant dispose d'ailes composées de trois panneaux, il est simulé à l'aide d'un modèle semi-empirique [Druot, 2004]. Le contrôleur ne dispose d'aucune entrée (commande en boucle ouverte) et pilote les trois degrés de liberté des ailes, à savoir le dièdre, le vrillage de l'aile et la flèche[1]. Le codage modulaire ModNet [Doncieux and Meyer, 2004b] (voir section 3.4.1) a été utilisé avec des neurones de type intégrateurs à fuite afin de permettre l'apparition de générateurs de rythme. L'algorithme évolutionniste multi-objectif MOGA [Fonseca and Fleming, 1993] a été utilisé pour la partie sélection. Le comportement d'une des solutions Pareto-optimales à l'issu de l'évolution est représenté sur le chronogramme 14. Le réseau de neurones correspondant ainsi que le tracé du mouvement des degrés de liberté des ailes sont représentés sur la figure 15. Les mouvements du dièdre et du vrillage s'approchent d'un créneau avec des transitions adoucies, montrant ainsi que des rythmes complexes nécessitant des synchronisations peuvent être générés par ce type d'approche.

Pour plus de détails, voir [Mouret et al., 2004].

1 la commande est symétrique entre les ailes gauche et droite.

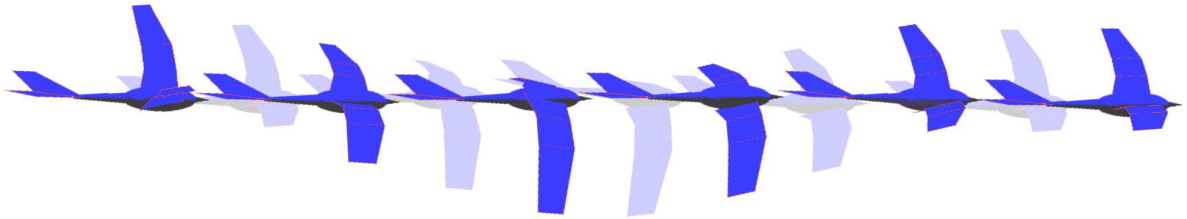


FIG. 14: Exemple de comportement obtenu par l'algorithme évolutionniste avec le codage ModNet.

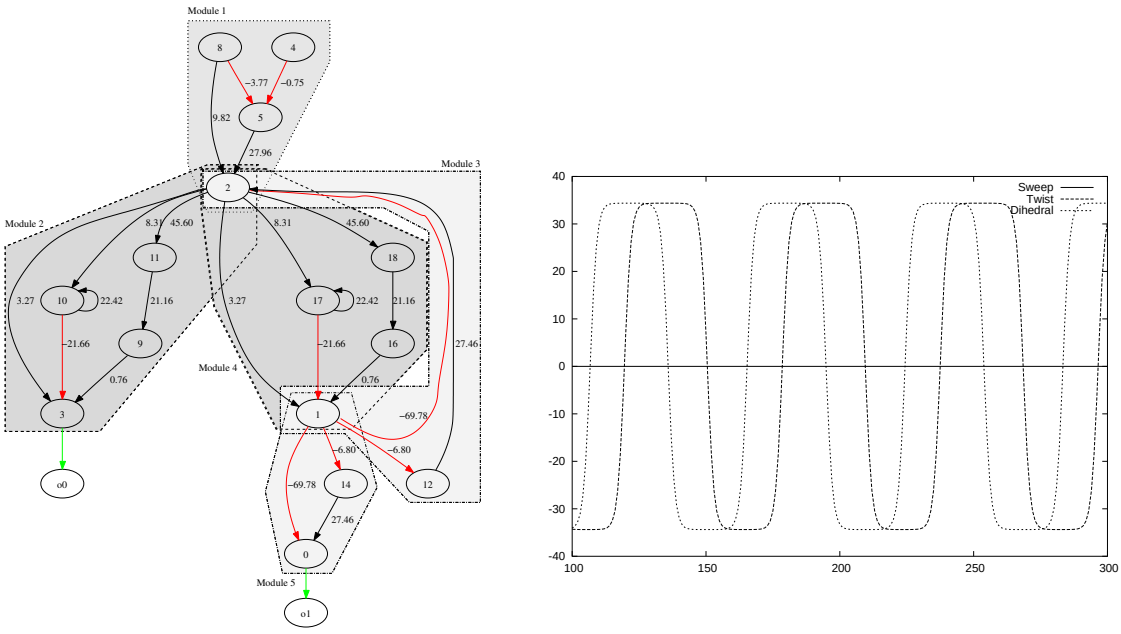


FIG. 15: Exemple de réseau de neurones obtenu par l'algorithme évolutionniste avec le codage ModNet (gauche) et mouvements des ailes correspondant (droite).

3.2.2 Vol en virage

Contexte: travaux réalisés pendant la thèse de Jean-Baptiste Mouret.
Publication(s) associée(s): [Mouret et al., 2006]

Des contrôleurs de vol en ligne droite ayant été générés, une approche incrémentale a été utilisée pour générer un contrôleur de queue permettant des trajectoires plus complexes. Ce type d'approche consiste à décomposer le problème à résoudre en sous-tâches qui sont ensuite résolues les unes après les autres [Harvey et al., 1994; Kodjabachian and Meyer, 1997; Winkeler and Manjunath, 1998; Parker, 2001]. Dans le cas présent, la première sous-tâche est le vol en ligne droite et la seconde le vol en virage, cette sous-tâche nécessitant que la première soit résolue. Compte tenu de l'approche multi-objectif utilisée, plusieurs contrôleurs de vol en ligne droite ont été générés. Après filtrage des solutions Pareto-optimales en supprimant les solutions dont la valeur sur un objectif au moins était jugée trop faible, deux contrôleurs de vol en ligne droite ont été retenus. Ils utilisaient différemment les degrés de liberté disponibles mais avec des performances similaires. Ne pouvant choisir à ce stade entre les deux, deux expériences ont été menées en parallèle. Chacune utilisait une de ces solutions pour le contrôle du vol en ligne droite et un algorithme évolutionniste avait pour tâche de synthétiser un contrôleur de queue permettant un vol en virage (figure 16). Les résultats présentés sur la figure 17 montrent que l'espace exploré dans les deux cas est très différent. Les solutions se basant sur un des deux contrôleurs (figure 17, gauche) dominent nettement celles qui se basent sur l'autre (figure 17, droite), alors que la performance initiale de ces contrôleurs ne le laissait pas présager. Cette constatation montre les limites d'une telle approche incrémentale en robotique évolutionniste, approche très dépendante d'un choix que l'expérimentateur doit faire sans disposer de toutes les données nécessaires.

Pour plus de détails sur cette expérience, voir [Mouret et al., 2006].

3.3 COMMENT EXERCER UNE PRESSION DE SÉLECTION EFFICACE ?

Le recours à une approche incrémentale a pour but de permettre la résolution de problèmes qui se sont avérés insolubles par ailleurs. Le principe général de telles approche est de décomposer le problème et de le résoudre « sous-problème » par « sous-problème » [Harvey et al., 1994; Kodjabachian and Meyer, 1997; Winkeler and Manjunath, 1998; Parker, 2001] ou en augmentant progressivement la complexité [Gomez and Miikkulainen, 1997], se ramenant ainsi à des difficultés abordables.

Une autre méthode fréquemment utilisée consiste à façonner la fonction de fitness (*fitness shaping* en anglais) [Nolfi and Paris, 1995], autrement dit à ajouter des termes visant à lisser le paysage de fitness en récompensant des comportements intermédiaires ou en punissant des comportements non désirés.

Toutes 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. Beaucoup de ces

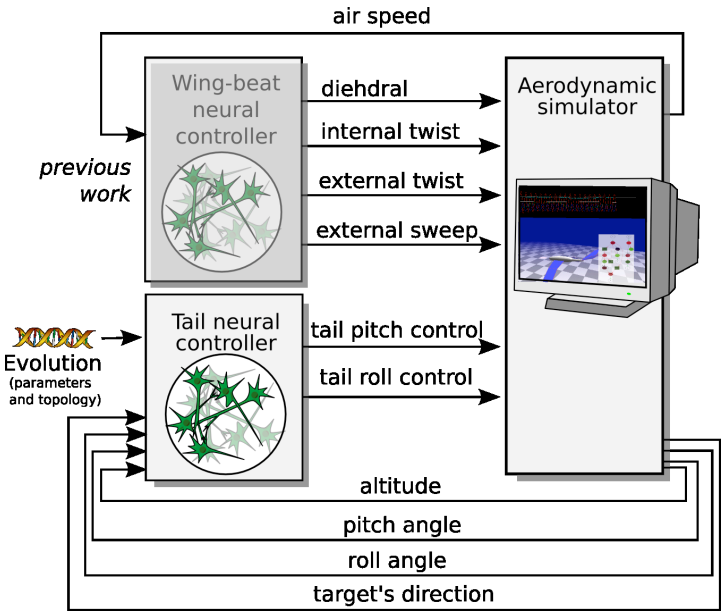


FIG. 16: Vue d'ensemble de l'approche adoptée pour le contrôle de virage d'un robot à ailes battantes.

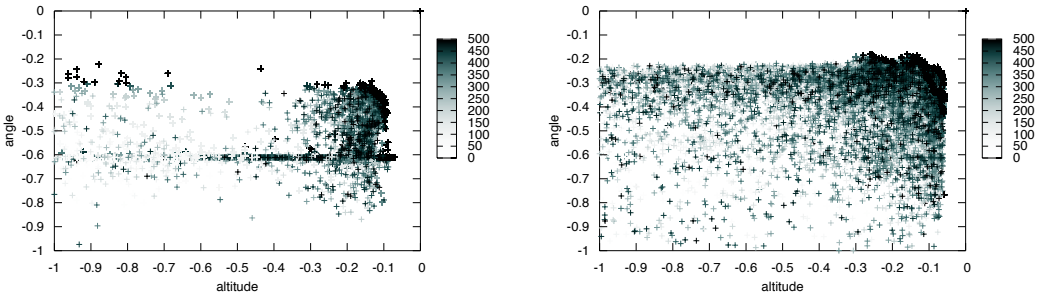


FIG. 17: Espace objectif exploré pour les deux contrôleurs de vol en ligne droite utilisés. Les deux objectifs utilisés récompensent le maintien d'altitude (axe des abscisses) et le maintien de la direction (axe des ordonnées). Le point idéal est en (0,0). Chaque point représente un contrôleur généré pendant le déroulement de l'algorithme évolutionniste. Le niveau de gris représente la génération à laquelle le point a été observé.

travaux consistent à prendre en compte des pressions de sélection multiples et à les activer ou inactiver en fonction du contexte ou encore à les agréger par des sommes pondérées ou des méthodes équivalentes. D'autres approches sont envisageables en exploitant le formalisme et les outils issus du domaine de l'optimisation multi-objectif. L'agrégation d'objectifs basée sur une somme pondérée a de nombreux inconvénients, notamment le choix des paramètres de pondération, mais aussi le fait que toutes les solutions ne sont pas atteignables dans certains cas[2]. L'utilisation du formalisme multi-objectif permet de s'affranchir de ces défauts et peut également ouvrir la voie à des approches incrémentales ne requérant pas l'intervention du concepteur pour le franchissement des différentes étapes, résolvant ainsi le problème évoqué précédemment.

3.3.1 Incrémentalité fondée sur une approche multi-objectif

Contexte: travaux réalisés dans le cadre de la thèse de Jean-Baptiste Mouret
Publication(s) associée(s): [Mouret and Doncieux, 2008a; Mouret, 2008]

Chaque élément à prendre en compte dans une fonction de fitness peut être considéré comme un objectif indépendant. Sur un problème pouvant être abordé par une approche incrémentale, autrement dit un problème que l'on peut décomposer en « sous-problèmes » plus simples, un objectif particulier peut être associé à chaque « sous-problème ». Le principal avantage de ce type d 'approche est de ne plus nécessiter d'intervention du concepteur dès lors que les calculs sont lancés. En effet, dans une approche incrémentale classique, le concepteur doit concevoir une expérience de synthèse ou d'optimisation pour chaque sous-problème. Cela nécessite en particulier de choisir à quel moment arrêter une expérience ; or il ne dispose généralement pas d'une connaissance suffisante du problème pour faire ce choix efficacement. De plus, avant de passer à l'étape suivante, il doit sélectionner parmi les résultats une solution particulière. Les optimisations ou synthèses ultérieures s'appuieront sur cette solution pour la résolution du sous-problème associé. Or là encore, le concepteur ne dispose pas de toutes les informations pour faire un choix efficace (voir la section précédente).

L'utilisation du cadre multi-objectif supprime la nécessité de faire de tels choix. Lorsque la synthèse évolutionniste est lancée, les sous-problèmes les plus simples verront apparaître des individus ayant des performances variées. Une pression de sélection s'exercera donc sur ces individus, dirigeant ainsi les recherches vers la résolution de cette sous-tâche. Un sous-problème difficile verra tous les individus obtenir une note basse similaire. Aucune pression de sélection ne s'exercera donc sur la population. Par contre, dès que ce sous-problème deviendra abordable, par exemple lorsque d'un sous-problème prérequis sera résolu, une pression de sélection commencera à être exercée sur la population. Cette activation de la pression de sélection est complètement automatique et peut se faire à tout moment de l'ex-

2 Si le front de Pareto n'est pas convexe, le changement progressif des pondérations ne permet pas de parcourir tout le front, mais fait passer d'un point extrême à un autre, sautant ainsi des solutions potentiellement intéressantes.

ploration évolutionniste. La recherche, de plus, continuera à s'exercer sur les sous-problèmes prérequis, laissant donc la possibilité de trouver d'autres solutions plus efficaces pour ceux-ci. Les deux inconvénients de l'approche incrémentale classique sont donc supprimés. La dépendance entre les sous-problèmes n'a, de plus, pas besoin d'être spécifiée, l'algorithme les « découvrant » automatiquement.

Une telle approche a été appliquée à la résolution d'une tâché séquentielle d'allumage de lampes [Mouret and Doncieux, 2008a]. Chaque lampe agit comme un interrupteur permettant d'allumer une ou plusieurs autre lampes. Le but de l'expérience est d'allumer une lampe particulière sans savoir quelles lampes il faut allumer ni dans quel ordre il faut le faire (figure 18). Un objectif est associé à l'allumage de chaque lampe : il mesure le temps mis pour allumer la lampe. L'utilisation directe d'une approche multi-objectif permet de résoudre le problème avec l'avantage que si plusieurs séquences permettent d'atteindre l'objectif, toutes seront explorées et la plus courte sera ainsi découverte (figure 18).

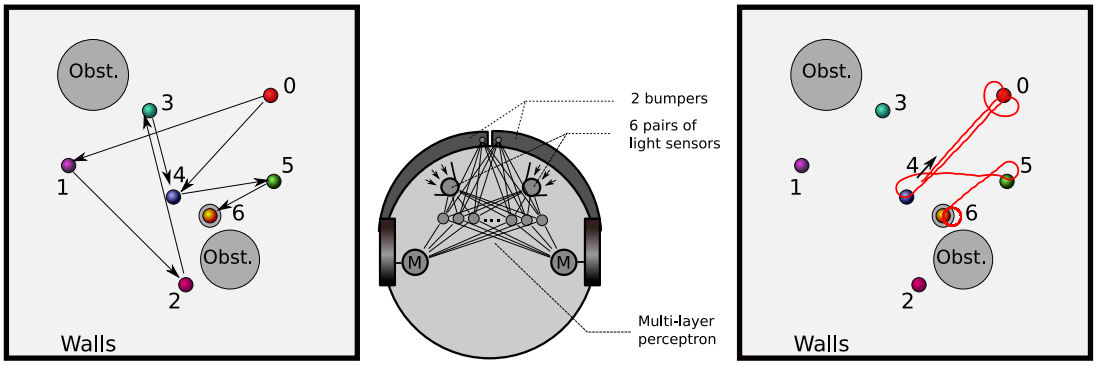


FIG. 18: Gauche : arène du robot avec les lampes-interrupteurs et les obstacles, les flèches indiquent l'ordre dans lequel les lampes doivent être allumées : la lampe 0 agit comme un interrupteur allumant les lampes 1 et 4, la lampe 1 permet d'allumer la lampe 2, etc ; milieu : caractéristiques du robot utilisé ; droite : exemple de trajectoire obtenue, solution choisie parmi celles qui allumaient la lampe 6 le plus vite possible.

La principale limitation de ce type d'approche est dans le faible nombre d'objectifs que les algorithmes multi-objectifs sont susceptibles de manipuler (typiquement 3, même si plus d'objectifs peuvent être utilisés s'ils ne sont pas complètement antagonistes). Intuitivement, une telle approche revient à rechercher simultanément dans des directions nombreuses : chaque objectif est optimisé, mais également chaque combinaison d'objectifs.

3.3.2 Diversité comportementale

Contexte: travaux initiés pendant la thèse de Jean-Baptiste Mouret et poursuivis depuis par Jean-Baptiste et moi-même.
Publication(s) associée(s): [Mouret, 2008; Mouret and Doncieux, 2009b,c; Doncieux and Mouret, 2010]

La difficulté de faire converger des expériences de robotique évolutionniste, difficulté qui a motivé l'utilisation d'approches incrémentales et du façonnage de fitness, peut également être le symptôme d'un problème d'exploration. Si un problème particulier peut être résolu dès lors que la pression de sélection est quelque peu modifiée, cela signifie que la recherche n'était pas efficace avec la version initiale de cette pression de sélection. L'hypothèse généralement admise est que l'espace de recherche est trop vaste et que l'algorithme évolutionniste doit être mis sur la piste des bonnes solutions en ajoutant des connaissances a priori.

En robotique évolutionniste, la fonction de fitness s'intéresse au comportement d'un robot. La recherche doit donc être efficace *dans l'espace des comportements*. Comme pour toute autre métaheuristique, une des difficultés dans la définition d'un algorithme évolutionniste est de trouver un bon équilibre entre l'exploration de l'espace de recherche et l'amélioration des meilleures solutions. Il est connu depuis les premiers algorithmes évolutionnistes que l'efficacité de l'exploration repose le maintien d'une diversité suffisante. De nombreux mécanismes ont ainsi été proposés pour l'assurer : fitness sharing [Goldberg and Richardson, 1987], crowding [Mahfoud, 1997] ou encore maintien multi-objectif de la diversité [de Jong et al., 2001; Toffolo and Benini, 2003; Bui et al., 2005]. L'information sur laquelle se base l'évaluation de la diversité dans ces travaux peut être le génotype ou la fitness.

L'évaluation d'un individu requiert habituellement les étapes suivantes :

– développement du génotype en un phénotype ;
– utilisation du phénotype pour calculer la fitness.

Dans le cas de la robotique évolutionniste, il y a une étape supplémentaire entre le phénotype et la fitness, qui est l'évaluation du comportement du robot. Le comportement est déterminé par le phénotype et résulte de l'interaction du robot avec son environnement. Cette étape complexifie le lien entre le génotype et la fitness. En effet, plusieurs génotypes peuvent correspondre au même comportement (figure 19) : dans le cas de contrôleurs neuronaux, toute partie non connectée aux sorties n'a aucun impact sur le comportement, par exemple. Une hypothèse alternative pour expliquer les problèmes de démarrage et de convergence prématurée en robotique évolutionniste, pourrait donc être que maintenir la diversité au niveau du génotype est insuffisant pour garantir une diversité comportementale. La fitness ne prend en compte que certains aspects du comportement. Elle récompense généralement un comportement adapté mais plusieurs fitness similaires peuvent correspondre à des comportements différents et des fitness très différentes peuvent correspondre à des comportements très proches : dans le cas d'une tâche d'évitement d'obstacles, avec une fitness comptant par exemple le nombre de collisions, une petite modification de la trajectoire peu suffire à changer considérablement la fitness alors que le comportement est finalement très proche. Maintenir la diversité au niveau de la fitness n'est donc pas non plus suffisant pour garantir une bonne exploration de l'espace des comportements.

Cette constatation implique de prendre en compte la diversité à un niveau directement *comportemental* et a conduit à des approches prenant le contre-pied des approches classiques. La recherche de nouveauté de Lehman et Stanley [Lehman and Stanley, 2008, 2010] maximise ainsi *uniquement* la nouveauté d'un comportement en le comparant à une archive des comportements déjà rencontrés. Elle réalise

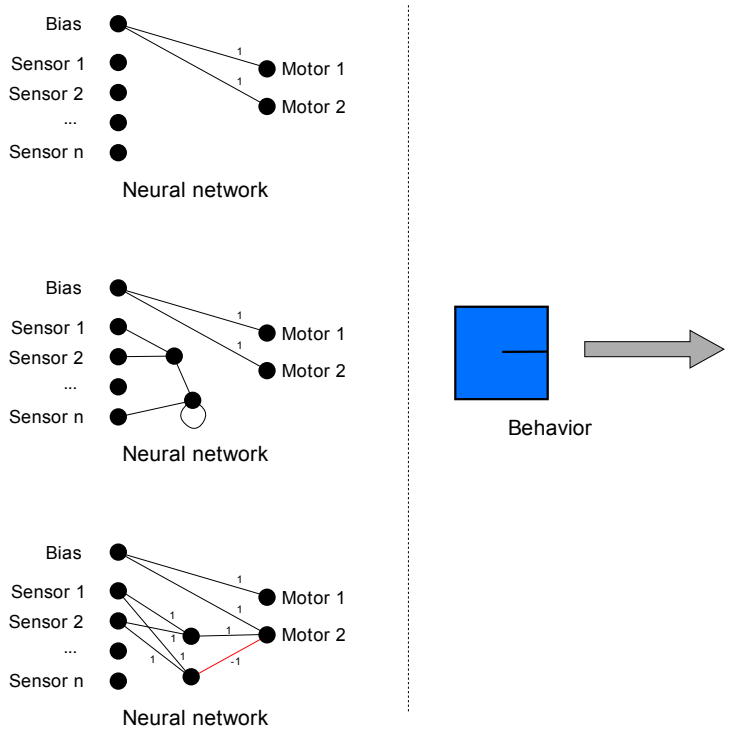


FIG. 19: Exemples de réseaux de neurones différents générant le même comportement.

ainsi une recherche exhaustive dans l'espace des comportements en s'appuyant sur NEAT [Stanley and Miikkulainen, 2002] pour garantir une augmentation progressive de la complexité des comportements explorés. La prise en compte de la diversité à un niveau comportemental n'implique cependant pas de sortir du cadre classique de la synthèse évolutionniste. Il est possible d'utiliser les mêmes fitness que précédemment, en prenant en compte cette diversité comportementale là où elle n'était précédemment prise en compte qu'au niveau de l'espace des génotypes ou de la fitness.

Comment prendre en compte cette diversité ? La méthode la plus populaire est le partage de fitness (fitness sharing) [Goldberg and Richardson, 1987], qui consiste à diviser la fitness associée à un individu par le nombre de solutions similaires dans la population. Cette division est une forme d'agrégation d'objectifs. Le formalisme des algorithmes évolutionnistes multi-objectifs peut donc être utilisé avec la fitness initiale comme premier objectif et la mesure de diversité comme second objectif. Cette approche a été testée sur la base de diversités dans l'espace des génotypes [Abbass and Deb, 2003; de Jong et al., 2001] ou dans l'espace des fitness [Bui et al., 2005] pour arriver à la conclusion que la version multi-objectif du maintien de la diversité était plus efficace.

Les mesures de diversité pénalisent les individus trop similaires au reste de la population et récompensent ceux qui sont les plus différents. Cela implique de pouvoir comparer des comportements et donc de disposer de descripteurs et d'une mesure de similarité dans cet espace. Quels descripteurs utiliser pour cette mesure ? Deux alternatives sont envisageables :

– utiliser des descripteurs *adhoc* : le concepteur choisit des descripteurs liés au problème à résoudre ;
– utiliser des descripteurs génériques : les descripteurs ne dépendent pas d'une expérience particulière, mais sont définis de façon systématique sans nécessiter de connaissance supplémentaire.

Ces alternatives ont été explorées sur des tâches de synthèse de contrôleurs neuronaux pour des tâches variées allant de la synthèse de fonction booléenne [Mouret and Doncieux, 2009c] à la résolution de tâches séquentielles sur un robot simulé [Mouret and Doncieux, 2009b; Doncieux and Mouret, 2010].

La tâche d'allumage séquentiel de lampes évoquée précédemment a ainsi pu être résolue en utilisant deux objectifs [Mouret and Doncieux, 2009b] :

– la fitness évaluant la performance sur la tâche considérée, ici le temps mis à allumer une lampe particulière (à minimiser) ;
– la diversité (à maximiser) ; dans cette expérience, le descripteur de comportement est le vecteur de l'état des lampes (allumée/éteinte) à la fin de l'expérience et la mesure de similarité est une distance Euclidienne entre les vecteurs.

Les expériences basées uniquement sur le premier objectif n'ont généré aucun contrôleur fonctionnel alors qu'avec l'objectif de diversité, toutes les expériences ont convergé. Si la convergence a été légèrement plus lente que lors des expériences d'incrémentalité multi-objectif (génération moyenne de convergence de 218 avec un écart-type de 75 contre 138 et un écart-type de 38), la performance finale était identique.

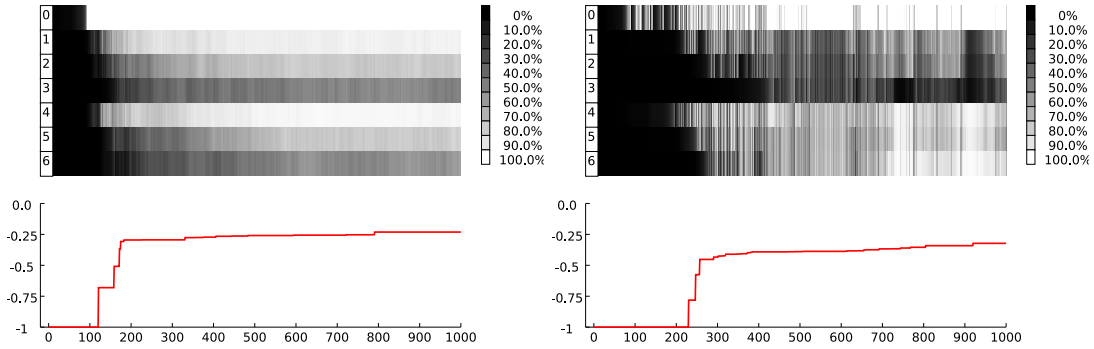


FIG. 20: Haut : tracé de la proportion de la population parvenant à allumer chaque lampe en fonction des générations. Bas : fitness sur l'objectif d'allumage de la lampe désirée (valeur idéale : 0, indiquant un allumage instantané). Gauche : approche incrémentale multi-objectif. Droite : approche basée sur la diversité comportementale.

La façon d'explorer l'espace des comportements s'est révélée être très différente dans les deux cas (figure 20). Dans le cas de l'approche incrémentale multi-objectif, une progression monotone est observée : dès lors qu'une lampe a pu être allumée une fois, la proportion d'individus réussissant à l'allumer ne fait (presque) qu'augmenter. Dans le cas de la diversité comportementale, des cycles peuvent être observés avec des proportions très changeantes au cours du temps. Cela révèle la différence fondamentale entre les deux approches. Dans le premier cas, les pressions de sélec-

tions poussent à allumer le plus de lampes possibles (et ce le plus vite possible) alors que dans le second, elles poussent à considérer toutes les combinaisons de lampes allumées/éteintes possibles. La diversité comportementale pousse donc à examiner plus de situations différentes : il est donc normal que la convergence soit plus lente, mais en contre-partie elle nécessite une connaissance a priori plus faible : la notion de lampe allumée/éteinte est toujours donnée, mais pas le fait qu'il est plus intéressant d'allumer des lampes que de les garder éteintes.

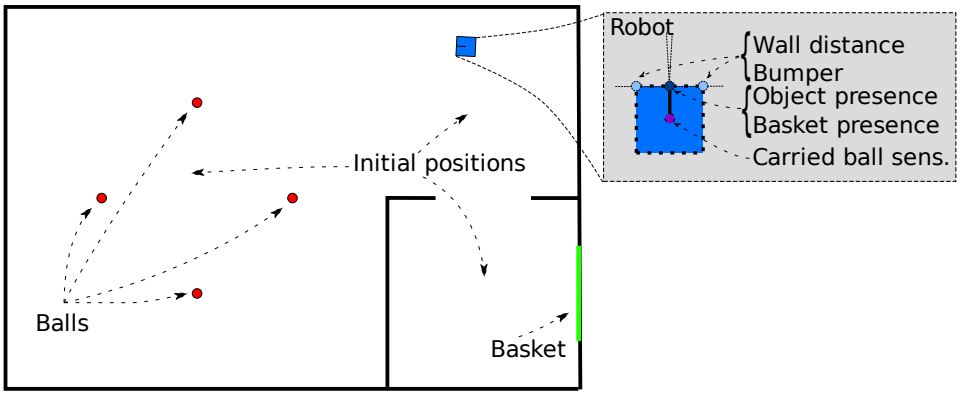


FIG. 21: Vue d'ensemble de l'arène et du robot. Le robot doit ramasser des balles et les déposer dans un panier protégé par des murs. Les capteurs de balle ou de panier ne voient pas à travers les murs. Une balle déposée dans l'arène hors du panier disparaît et ne peut donc être récupérée. L'évaluation repose sur le comptage du nombre de balles ramassées depuis trois positions initiales imposées pendant une durée fixe de 2000 pas de temps.

Une tâche plus complexe a ensuite été considérée (figure 21). Cette tâche consiste à ramasser le plus grand nombre de balles possible pendant une durée donnée et à les déposer dans un panier (une seule balle peut être transportée à la fois). La fitness utilisée est le nombre de balles déposées dans le panier. La difficulté du problème a été augmentée en entourant le panier de murs empêchant ainsi de l'atteindre facilement, mais également de le voir. La tâche consiste donc à ce que le robot détecte une balle, aille vers elle, la ramasse, la conserve, cherche le panier, aille vers lui et dépose la balle au moment où il est en contact avec le panier puis reparte à la recherche d'une autre balle et réitère ce cycle tant que toutes les balles n'ont pas été collectées. L'utilisation du nombre de balles comme seule pression de sélection n'a généré que des individus qui, au mieux, parvenaient à collecter une seule balle dans une ou deux des trois conditions initiales. L'utilisation d'un deuxième objectif avec une valeur aléatoire a donné des résultats encore moins bons.

Plusieurs mesures de diversités ont été testées [Doncieux and Mouret, 2010], qu'elles reposent sur des descripteurs génériques (le flux des perceptions et des ordres moteurs) ou sur des descripteurs adhoc. Toutes les expériences utilisant un objectif de diversité, quel qu'il soit, ont généré des individus capables de ramasser plusieurs balles pendant une seule évaluation, autrement dit capables de réaliser la séquence décrite ci-dessus (figure 22). Parmi les alternatives testées, la plus efficace est celle consistant à mesurer une distance de Hamming entre les flux de perception

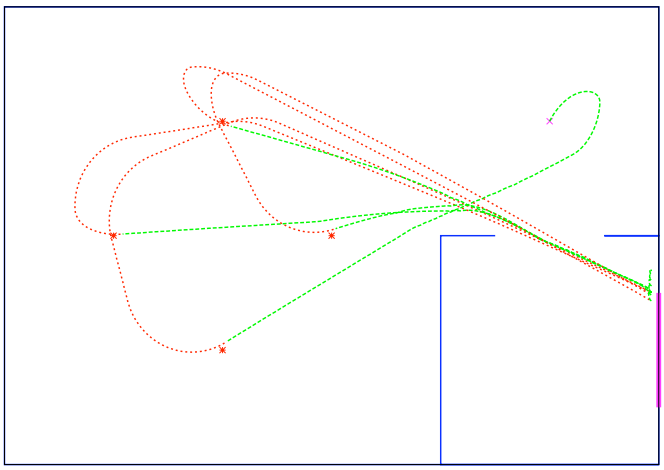


FIG. 22: Exemple de trajectoire générée par un contrôleur obtenu dans une expérience avec diversité comportementale. La position initiale est représentée par une croix, la position des balles avec des étoiles. La trajectoire du robot est représentée en vert avec des traits pointillés longs lorsqu'il ne porte pas de balle et en rouge avec des traits pointillés courts lorsqu'il en porte une.

et d'ordres moteurs binarisés, montrant ainsi que des mesures basées sur des descripteurs génériques sont compétitives, voire parfois plus efficaces que des mesures basées sur des descripteurs adhoc.

Ces résultats tendent à montrer l'importance de la prise en compte de la diversité comportementale dans le processus de recherche. Depuis le travail séminal de Lehman et Stanley sur la nouveauté, de nombreux travaux se sont intéressés à cette question selon différentes perspectives et dans différents contextes :

- étude de mesures de diversité basée sur la trajectoire [Trujillo et al., 2008] ;
- étude de mesures approchées de la complexité de Kolmogorov (entre autres) dans un environnement discret [Gomez, 2009] ;
- utilisation de mesures d'entropie pour faciliter l'évolution ouverte dans un contexte d'évolution embarquée [Delarboulas et al., 2010] ;
- prise en compte dans NEAT [Moriguchi and Honiden, 2010].

Le gain, généralement très significatif, associé à l'utilisation de cette information justifie l'intérêt de mener plus avant ce type de travaux en parallèle de ceux qui se focalisent sur des algorithmes de sélection, des formalismes de contrôleurs ou des codages. Cela correspond à un changement de perspective important. En effet, dans le cadre de l'optimisation, qui a une grande influence sur les travaux en robotique évolutionniste, la fonction de fitness est une donnée du problème, non une variable ajustable. Ces travaux amènent à reconsidérer cette position et à se focaliser sur la définition de la notion de pression de sélection indépendamment d'un problème particulier.

3.4 QUE FAIRE ÉVOLUER ?

Le choix de l'espace de recherche soumis à la recherche évolutionniste est déterminant pour la complexité des comportements atteignables. Les algorithmes d'optimisation évolutionnistes utilisent pour la plupart un génotype de taille et de structure fixes, que ce soit une chaîne de symboles [Goldberg, 1989] ou un vecteur de nombres réels [Back and Schwefel, 1995; Schwefel and Back, 1995]. Une approche simple et intuitive pour synthétiser un contrôleur consiste à considérer un réseau de neurones de taille suffisante et à optimiser ses paramètres, comme par exemple le poids des connexions, comme cela a été présenté dans la section 2.1.2. Le choix du schéma de connexion est dans ce cas déterminant. Un réseau entièrement connecté offre l'avantage de permettre des comportements nécessitant une mémoire ou la génération d'activités cycliques. Les paramètres sont alors simplement les poids des connexions, éventuellement assortis d'un paramètre booléens indiquant si la connexion est présente ou non. En théorie, n'importe quelle structure neuronale pourrait ainsi être générée sur la base d'un génome à structure fixe. Cette approche a été utilisée, par exemple, pour générer des systèmes de contrôle pour des robots Khépéra, pour des tâches d'évitement d'obstacles [Floreano and Mondada, 1994], pour générer des comportement de prédation ou d'échappement [Floreano et al., 1998] ou encore pour exhiber un comportement de recharge automatique [Floreano and Mondada, 1996].

Plusieurs éléments motivent cependant la recherche d'alternatives à cette approche. Les résultats trouvés seront en effet fortement dépendants de la structure choisie. Une structure trop simple ne pourra pas résoudre le problème, alors qu'une structure trop complexe risque de sur-apprendre et d'être donc parfaitement adaptée aux conditions d'évaluation, mais sans généraliser à d'autres contextes.

De plus, l'optimisation des paramètres d'une structure fixe ne permet pas d'exploiter des régularités et de répéter des structures, par exemple, phénomène pourtant courant dans le monde du vivant. La répétition de structures similaires y est très fréquente, et permet d'implémenter des symétries – symétrie morphologique droite-gauche fréquente, par exemple – ou de réutiliser des structures élémentaires comme les colonnes corticales présentes dans le cortex [Sporns, 2002]. Or l'optimisation de paramètres ne permet pas de découvrir de telles dépendances et pour limiter l'épistasie, considérée comme nuisant aux performances des algorithmes évolutionnistes [Ronald, 1997], toute régularité est généralement étudiée au préalable et donnée *a priori*.

Ces constatations ont motivé de nombreux travaux dans lesquels la structure était également à découvrir [Yao, 1999]. Deux types de représentations (ou codages) peuvent être distingués : les codages directs et les codages indirects [Kodjabachian and Meyer, 1995]. Dans les codages directs les opérateurs génétiques manipulent directement la structure en ajoutant ou enlevant neurones et connexions. De cette catégorie, NEAT est actuellement la référence [Stanley and Miikkulainen, 2002]. Ce type de codage ne permet cependant pas de répéter des structures ou de gérer des symétries. Les codages indirects, en renonçant au lien direct entre génotype et phénotype, permettent ce type de propriété en offrant la possibilité de gérer des modules.

3.4.1 Codages modulaires

La modularité d'une solution est un point crucial, tant en biologie qu'en ingénierie. Dans ces deux domaines, une structure n'est pas un assemblage homogènes des primitives les plus élémentaires. Des structures intermédiaires sont définies qui peuvent être réutilisées ou combinées. Ce besoin a été mis en évidence par Simon dans sa parabole des deux horlogers [Simon, 1962]. Le premier construit des montres de telle façon que chaque partie dépend des autres. S'il est interrompu pendant l'assemblage, la montre tombe en morceaux. Le second construit des modules qu'il assemblera ensuite entre eux pour finalement aboutir à la montre finale. S'il est interrompu, l'assemblage en cours échoue, mais les modules ou assemblages de modules déjà réalisés ne sont pas perdus. Simon montre que le premier a beaucoup moins de chances d'aboutir que le second.

La capacité à gérer des modules dans un réseau de neurones a fait l'objet de nombreux travaux [Boers E. et al., 1993; Happel B. and Murre J., 1994; Gruau, 1995; Buessler and Urban, 2003]. Dans un contexte de synthèse par algorithme génétique, plusieurs codages indirects incluant cette capacité ont été proposés : le codage basé sur des *L-systems* de Boers E. et al. [1993], notamment, ou encore le codage cellulaire de Gruau [1992] dès lors que l'on y ajoute la notion d'ADF (*Automatically Defined Functions*), issue de la programmation génétique [Gruau, 1995].

MODNET

Contexte: ModNet a été développé pendant ma thèse et utilisé dans plusieurs travaux suivants
Publication(s) associée(s): [Doncieux, 2003; Doncieux and Meyer, 2003, 2004a, 2005; Mouret et al., 2004, 2006]

ModNet est un codage minimaliste, inspiré des codages directs et intégrant directement la notion de modules [Doncieux, 2003]. Son développement était motivé par la simplicité et la souplesse d'utilisation, qui permettait une inclusion de connaissances *a priori* fournie sous forme de modules de structure donnée[3]. Le génotype est constitué d'une liste de modules modèles, d'une liste de modules, qui sont des copies des modèles et d'un schéma de connexions entre ces modules, donné sous la forme d'une liste de liens (figure 23). Les mutations sont de deux types et peuvent soit modifier les paramètres du réseau, soit changer la structure en modifiant l'un des trois composants du génotype. Le croisement permet d'échanger des modules modèles entre génotypes. Ce codage a été utilisé pour générer des systèmes de contrôle pour un pendule inversé [Doncieux, 2003], pour un dirigeable lenticulaire [Doncieux and Meyer, 2003, 2004a, 2005], un hélicoptère [Doncieux, 2003] ou encore un robot à ailes battantes [Mouret et al., 2004, 2006].

MENNAG

Contexte: Codage développé pendant la thèse de Jean-Baptiste Mouret
Publication(s) associée(s): [Mouret and Doncieux, 2008b]

3 ModNet permet également de générer la structure des modules.

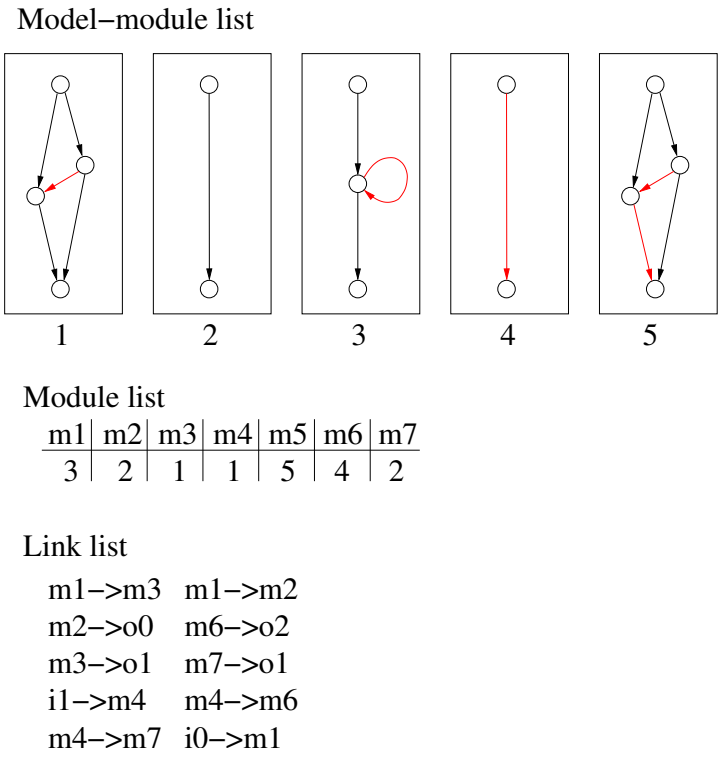


FIG. 23: Codage ModNet. Un génotype est constitué des trois éléments : une liste de modules modèles, une liste de modules, copies des modèles et une liste de liens entre modules.

Aux concepts de module et de répétition précédemment évoqués peut s'ajouter le concept de hiérarchie, correspondant à la composition récursive de modules [Lipson, 2004; de Jong and Thierens, 2004; Hornby, 2005]. Ce concept, totalement absent de ModNet, semble pourtant indispensable à la synthèse de réseaux de neurones générant un comportement de complexité croissante. Cette constatation, associée à l'absence de ce concept dans les autres codages modulaires [Boers E. et al., 1993; Gruau, 1995; Reisinger et al., 2004], a conduit au développement de MENNAG, un nouveau codage incluant cette capacité [Mouret and Doncieux, 2008b].

Ce codage est basé sur des grammaires attribuées [Knuth, 1968], spécifiant à la fois la syntaxe et la sémantique de la représentation. Ce choix, proposé par [Hussain, 2003], est motivé par la volonté de disposer d'une représentation formelle facile à partager et contenant une description complète de toutes les caractéristiques du codage, facilitant ainsi sa ré-implémentation par d'autres auteurs. Au passage, il a aussi pour but de faciliter les modifications des codages et surtout leurs comparaisons, encore trop rares dans la littérature, essentiellement à cause de la difficulté d'implémentation des différents codages existants. Il a en effet été montré qu'un tel

formalisme permettait d'implémenter de nombreux autres codages [Hussain, 2003], notamment le codage cellulaire [Gruau, 1994] et ses variantes[4].

Le principe général de MENNAG est illustré sur la figure 24. Une grammaire attribuée est utilisée pour la génération aléatoire et les opérateurs génétiques de mutation et de croisement. Le réseau de neurones associé à un génotype particulier est obtenu à l'issue d'une étape de développement depuis une cellule initiale, comme pour le codage cellulaire. La spécificité de MENNAG tient dans la définition de la grammaire définissant l'espace de recherche, conçue pour inclure la notion de modules susceptibles d'être répétés, ainsi que la notion de hiérarchie.

Les modules apparaissent au niveau des noeuds intermédiaires de type DIV (pour division). Ces noeuds ont en effet pour fils d'autres DIV ou des CELL (instruction marquant la fin des divisions et le début de la spécialisation vers un neurone) ainsi qu'une liste de connexions. Chaque connexion relie des neurones appartenant uniquement au module auquel elle appartient ; seules des connexions de modules supérieurs peuvent donc relier plusieurs modules entre eux. La hiérarchie découle directement de la structure arborescente du programme de développement : un module, décrit par un noeud DIV et le sous-arbre associé, pourra contenir d'autres noeuds DIV qui seront eux-mêmes des modules contenant d'autres modules, etc. La répétition vient d'une instruction spéciale, CLONE, permettant de cloner un module. Cela donne la grammaire suivante (voir l'article sur MENNAG en annexe pour la version complète incluant la gestion des attributs) :

START ⇒ DIV CONNS
DIV ⇒ DIV CLONES CONNS
DIV ⇒ CLONES DIV CONNS
CLONES ⇒ CLONE CLONES
CLONES ⇒ CLONE
DIV ⇒ DIV DIV CONNS
DIV ⇒ CELL CELL CONNS
CELL ⇒ IN OUT
CLONE ⇒ end
CONNS ⇒ CONN CONNS
CONNS ⇒ end
CONN ⇒ ICONN
CONN ⇒ ICONN
ICONN ⇒ weight node target
IN ⇒ end
IN ⇒ IO in1
IN ⇒ IO in2
OUT ⇒ end
OUT ⇒ IO out1
OUT ⇒ IO out2
IO ⇒ weight target

Cette grammaire a été testée sur un problème simple : le contrôle d'un pendule inversé (avec l'erreur en position du pendule et du chariot uniquement, pas leurs dérivées), et sur un problème plus difficile : le contrôle d'un bras doté de trois articulations commandées, problème permettant de tester la capacité de répétition [Mouret

4 Tous les codages ne sont cependant pas représentables dans ce formalisme – NEAT, notamment, n'inclut pas que la définition du génome et des opérateurs génétiques mais aussi un mécanisme de sélection spécifique [Stanley and Miikkulainen, 2002].

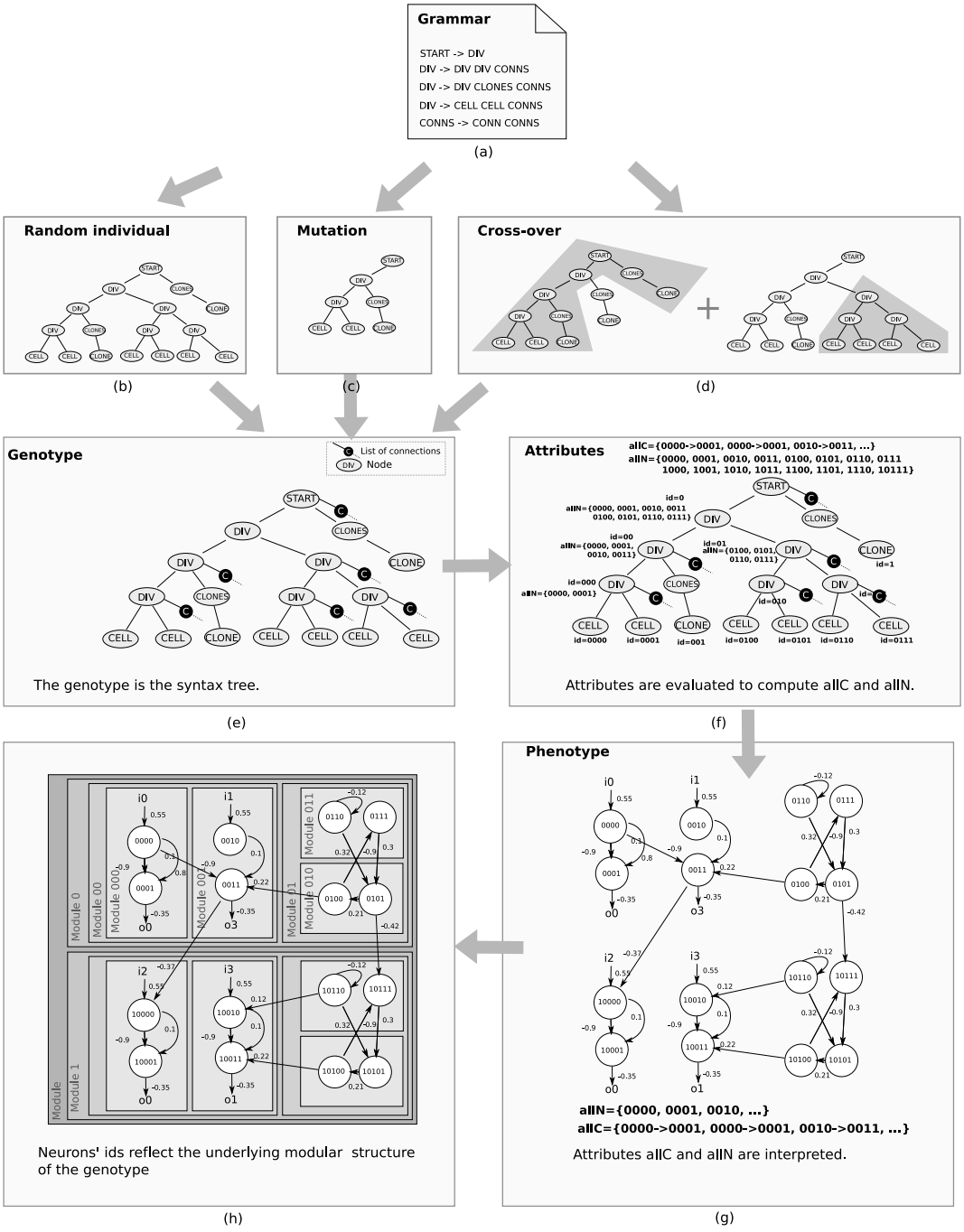
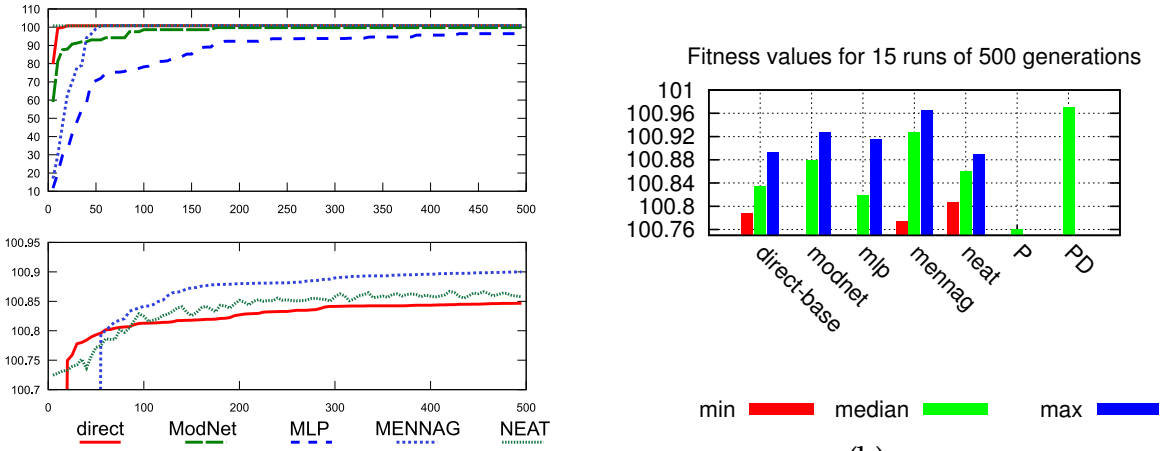


FIG. 24: Codage MENNAG.



(a) (b)

FIG. 25: Contrôle du pendule inversé. (a) Fitness moyenne sur 15 expériences pour les differents codages. Le maximum théorique est de 101. (b)Minimum, médiane et maximum de fitness sur 15 expériences durant 500 générations. Le contrôleur proportionnel (P) évite la chute du pendule, mais n'amortit pas les oscillations. Le correcteur proportionnel dérivé (PD) centre rapidement chariot et pendule. Direct-base est un codage direct simple et mlp, un perceptron multi-couches de structure fixe.

and Doncieux, 2008b]. Dans le cas du pendule (figure 25), la performance maximale obtenue avec MENNAG est très proche de celle d'un correcteur proportionnel-dérivé qui donne les meilleurs résultats pour ce problème. Aucun autre codage n'a atteint de telles performances. Sur les expériences réalisées, NEAT parvient à empêcher le pendule de chuter, mais n'arrive pas à amortir complètement les oscillations dans les mouvements du chariot et du pendule, ces résultats étant proches d'un codage direct simple. ModNet obtient des résultats intermédiaires. Sur l'expérience du contrôle du bras articulé (figure 26) et parmi les codages testés, seul MENNAG parvient à atteindre les performances d'une structure fixe adaptée au problème et de paramètres optimisés (perceptron multi-couches (MLP) ou contrôleurs composés de trois correcteurs proportionnels (P-3)). Des tests ont été réalisés avec plus de degrés de liberté, mais les performances se détériorent rapidement au-delà de trois. Notre hypothèse est que la combinatoire liée au nombre d'entrées-sorties devient trop grande pour permettre une exploration suffisante compte tenu des tailles de population et du nombre de générations considérés.

3.4.2 Exaptation : de l'importance des pressions de sélection

Contexte: travaux réalisés pendant la thèse de Jean-Baptiste Mouret
Publication(s) associée(s): [Mouret, 2008; Mouret and Doncieux, 2009a]

La synthèse artificielle reste limitée à des problèmes extrêmement simples par rapport à l'évolution naturelle, il est donc intéressant de revenir à l'inspiration initiale de ces méthodes pour voir si des points importants n'ont pas été abusivement négligés. Selon les paléontologues, les organismes vivants ont évolué en adaptant

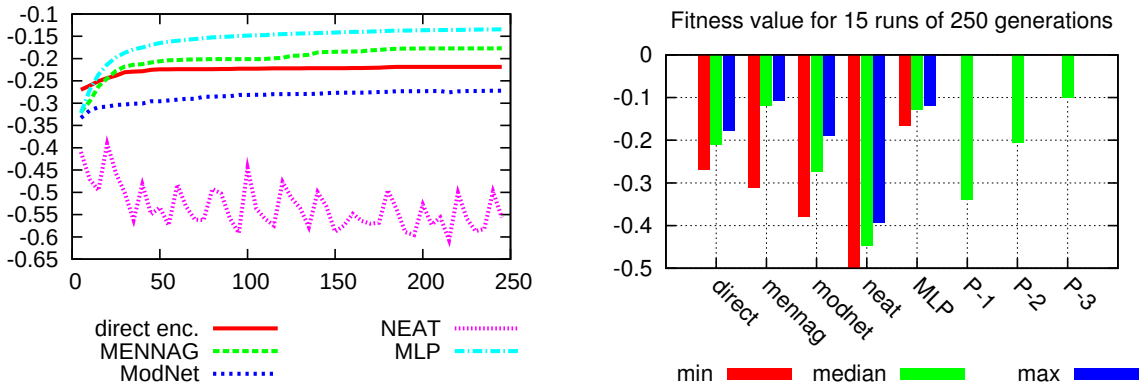


FIG. 26: Contrôle d'un bras articulé (a) Fitness moyenne sur 15 expériences pour les différents codages. Le maximum théorique est 0. (b) Fitness obtenue après 250 générations. P-1 est la fitness obtenue avec un seul contrôleur proportionnel, P-2 avec deux contrôleurs proportionnels, P-3 avec trois.

des caractères dédiés à une fonction particulière à la résolution d'un nouveau problème, un phénomène identifié dès Darwin [Darwin, 1859] et maintenant appelé *exaptation* [Gould and Vrba, 1982]. Les grandes étapes de la vie ont été jalonnées de tels événements, comme par exemple l'apparition des doigts des tetrapodes, initialement adaptés à un environnement aquatique, avant de permettre la conquête de la terre ferme [Coates and Clack, 1990]. Dans un contexte de synthèse artificielle, cela signifie qu'avant de servir à résoudre le problème principal, des modules ou des éléments de solution devraient servir à résoudre d'autres tâches, de façon à les façonner avant de les adapter. Cela nécessite de disposer de pressions de sélection multiples, susceptibles de s'appliquer à différentes parties du génotype. Ce principe ressemble à celui de l'évolution incrémentale [Harvey et al., 1994; Kodjabachian and Meyer, 1997; Urzelai et al., 1998; Mouret et al., 2006; Mouret and Doncieux, 2008a], mais dans ce cas, la pression s'applique à toute la structure, et non à un module du génotype. À l'opposé, de nombreux travaux se sont intéressés à la modularité de la représentation génotypique (voir section précédente), mais sans prendre en compte les pressions de sélection. Peu de travaux se sont intéressés aux liens entre modularité et pressions de sélection ; or, leur importance a été montrée récemment avec la nécessité de l'existence d'un alignement entre modules génotypiques, modules phénotypiques et pressions de sélection [Altenberg, 2005].

S'inspirant des opérateurs identifiés comme responsables de la modularité dans un contexte biologique [Wagner et al., 2005], un codage minimaliste ajoutant à un codage direct la possibilité d'isoler des modules et de les répéter via des opérateurs génétiques dédiés a été défini (figure 27). Ce codage a été testé sur la génération d'une fonction booléenne XOR-AND-XOR, une fonction modulaire par construction et difficile car trompeuse[5]. L'alignement pression de sélection et module génotypique est réalisé grâce à l'ajout d'un objectif supplémentaire récompensant l'existence de modules implémentant une fonction XOR dans le réseau. Les résultats trouvés confirment les travaux théoriques de Altenberg [2005] : l'existence de la

5 un réseau renvoyant « faux » indépendamment des entrées donne 75% des bonnes réponses.

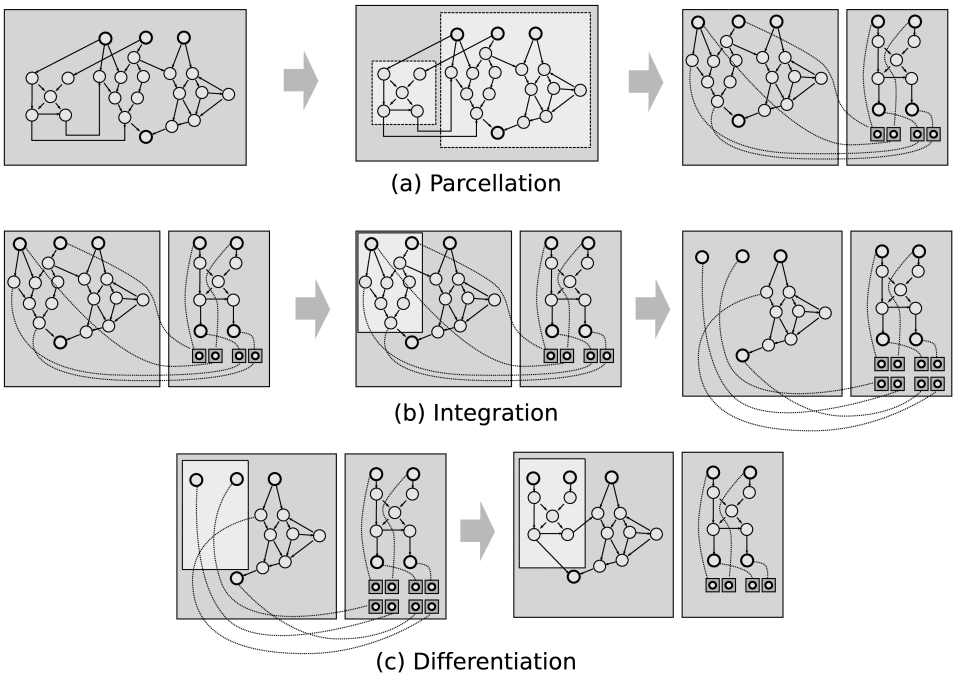


FIG. 27: Les trois principaux opérateurs génétiques du codage modulaire. (a) Parcellation : isolation d'un sous-réseau. (b) Intégration : remplacement d'un sous-réseau par un lien vers un sous-réseau parcellé (ce qui autorise la répétition d'un module) (c) Différentiation : opérateur symétrique de la parcellation, un sous-réseau est replacé dans le réseau principal.

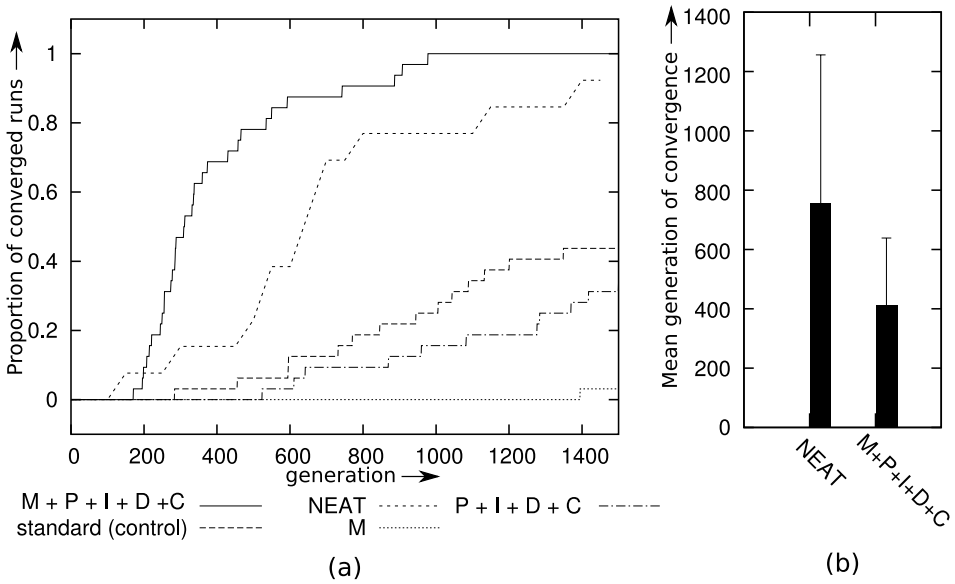


Fig. 28: (a) Proportion d'expériences ayant convergé (sur 32) en fonction des générations. "M+P+I+D+C" correspond aux expériences d'exaptation complète ; "standard" correspond aux expériences de contrôle qui utilisent un codage direct simple ; les expériences "M" correspondent à l'approche multi-objectif sans les opérateurs génétiques dédiés à la modularité ; dans "P+I+D+C", les opérateurs sont utilisés uniquement avec la fonction de fitness récompensant le XOR-AND-XOR. (b) Génération de convergence moyenne et écart-type pour les expériences dont le taux de convergence est supérieur à 0.5).

pression de sélection XOR sans les opérateurs génétiques ou l'utilisation des opérateurs génétiques sans la pression de sélection n'ont convergé que rarement, alors que l'utilisation simultanée des deux aspects permet d'atteindre un taux de convergence de 100% des expériences lancées (figure 28).

De façon intéressante, l'utilisation seule de la pression de sélection vers le XOR ou des opérateurs génétiques permettant de gérer la modularité, a donné des résultats nettement moins bons que l'expérience de contrôle qui utilisait un codage direct simple, montrant ainsi que, dans cette expérience au moins, l'utilisation d'un seul de ces aspects nuit à l'efficacité. Comme les travaux précédents portant sur la diversité comportementale, ces travaux montrent l'importance de la définition des fonctions de fitness et la nécessité de les prendre en compte, même dans des travaux portant pourtant sur le codage.

3.4.3 EvoNeuro : un codage inspiré des neurosciences computationnelles

Contexte: travaux réalisés dans le cadre du projet ANR EvoNeuro avec Jean-Baptiste Mouret, Benoît Girard et dans le cadre de la thèse de Tony Pinville.
Publication(s) associée(s): [Mouret et al., 2010; Pinville and Doncieux, 2010]

Les travaux sur l'exaptation tendent à montrer que concevoir un codage modulaire sans définir de pressions de sélection associées aux modules est voué à l'échec. C'est une remise en cause profonde de l'hypothèse implicite qui avait guidé le développement de ModNet ou MENNAG et qui était que des modules adaptés émergeraient au cours du processus évolutionniste sans besoin d'adapter la pression de sélection. C'est une approche par contre cohérente avec celle des biologistes étudiant l'évolution des espèces et qui, pour chaque partie d'un être vivant, essaient de comprendre quelle pression de sélection a pu la façonner.

Il est donc à prévoir que les problèmes solubles par une approche n'impliquant qu'un codage modulaire et sans pressions de sélection adaptées resteront des problèmes également solubles par une approche non modulaire. À la lumière de ces résultats, les bonnes performances de MENNAG peuvent très bien venir d'une meilleure capacité à explorer l'espace de recherche (indépendamment de toute modularité) et de la relative simplicité des modules nécessaires dans le cas du bras articulé, modules qui étaient ainsi très faciles à découvrir et à propager dès la génération aléatoire.

Dans l'objectif de tendre vers des comportements de plus en plus cognitifs, il est donc nécessaire de définir une nouvelle démarche associant des experts de ce sujet qui seraient à même d'identifier et de décrire les fonctionnalités des modules nécessaires, et donc les pressions de sélection qui devraient y être associées. Les experts en question peuvent se trouver dans le domaine des neurosciences computationnelles, dont l'objet est de définir des modèles numériques du cerveau afin de mieux en comprendre le fonctionnement, notamment pour des vertébrés supérieurs (voire des humains), qui disposent donc des capacités cognitives que nous cherchons à synthétiser.

C'est l'objectif de notre participation au projet ANR EvoNeuro, commencé en décembre 2009, que de faciliter ces échanges en permettant aux neuroscientifiques, en contrepartie, de bénéficier des capacités offertes par l'analyse multi-objectif sur leurs modèles [Liénard et al., 2010].

La démarche développée consiste à choisir une fonctionnalité clairement identifiée dans la communauté des neurosciences et à identifier les modèles l'implémentant. Les protocoles expérimentaux permettant de la tester sont étudiés pour en déduire un protocole d'évaluation ainsi que la fonction de fitness associée. La capacité de l'évolution à synthétiser ce module est ensuite testée en comparant les résultats obtenus aux modèles issus des neurosciences computationnelles. À terme, l'idée sera ensuite de se reposer sur une approche de type exaptation pour synthétiser les réseaux de neurones dotés des capacités cognitives recherchées. Le rôle de l'algorithme évolutionniste sera alors non pas de synthétiser le réseau de neurones *from scratch*, mais de choisir, d'assembler les différents modules proposés et également de les adapter à la tâche robotique considérée.

Les réseaux de neurones qui ont été évoqués jusqu'à présent sont très différents des modèles de neurosciences computationnelles. Les premiers contiennent au maximum quelques dizaines de neurones assemblés dans un réseau ne montrant aucune régularité évidente. Les modèles des neurosciences computationnelles contiennent plusieurs centaines, voire milliers, de neurones regroupés dans des cartes ou champs de neurones [Amari, 1977] et assemblés avec des schémas de projection réguliers. La façon de coder l'information est également très différente dans les deux cas. En robotique évolutionniste, beaucoup de robots utilisés pour des tâches de navigation sont de type Khepera [Mondada et al., 1993]. Pour ces robots, dotés de deux roues motorisées, chaque roue dispose généralement de son neurone moteur [Mondada and Floreano, 1995]. En neurosciences, la direction du mouvement, par exemple, serait plutôt représentée par une population de neurones, chaque neurone représentant une valeur particulière, la direction globale étant la somme de toutes ces contributions [Georgopoulos et al., 1986].

Afin de manipuler des modèles similaires à ceux des neurosciences computationnelles, il est intéressant de générer des réseaux de neurones ayant les mêmes primitives structurelles. Il semble en effet difficile de générer des réseaux aussi réguliers avec un codage direct simple. Cela permet, de plus, de comparer plus facilement les résultats obtenus et de faciliter la synthèse des fonctions identifiées : si elle ne fonctionne pas, cela peut signifier que toutes les primitives structurelles nécessaires ne sont pas présentes et en analysant les modèles des neurosciences, il est possible de proposer des ajouts au codage ; la démarche de conception s'en trouve donc facilitée.

Le codage EvoNeuro (figure 29) est inspiré d'un codage direct simple. Chaque noeud et chaque lien du réseau contient des étiquettes qui sont utilisées pendant le développement pour transformer le graphe en un réseau contenant des cartes neuronales et des schémas de connexion réguliers entre ces cartes. La fonctionnalité de sélection des ganglions de la base [Mouret et al., 2010] ainsi que des mémoires de travail [Pinville and Doncieux, 2010] ont pu être synthétisées avec ce codage. Par rapport à un codage direct, le codage EvoNeuro passait plus facilement à l'échelle grâce à sa capacité à facilement changer la taille des cartes neuronales (figure 30), et, lorsqu'un codage direct donnait également des résultats, les résultats d'EvoNeuro se

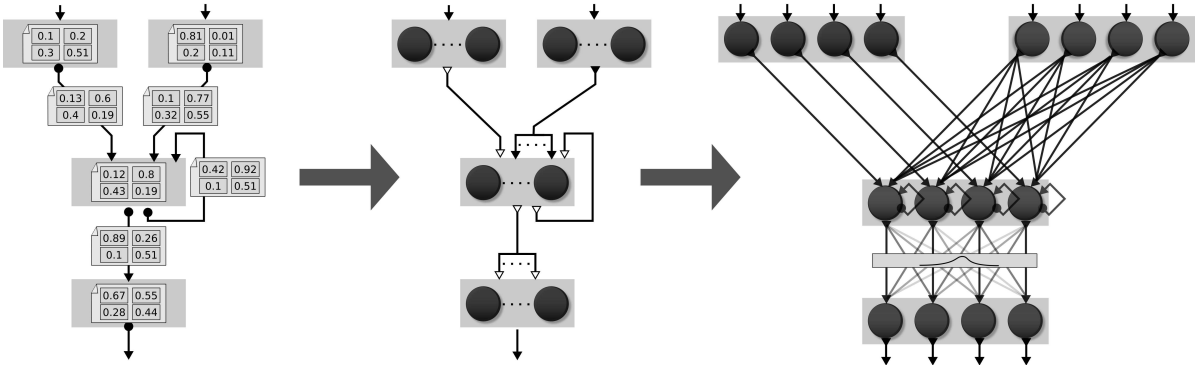


FIG. 29: Vue d'ensemble du processus de développement du codage EvoNeuro. De gauche à droite : (1) le génotype est un graphe étiqueté avec des étiquettes soumises à évolution ; (2) les étiquettes sont interprétées pour générer une description du réseau de neurones inspirée des neurosciences ; (3) pour une taille de carte donnée, ce réseau peut être complètement développé (pour évaluer sa fitness notamment).

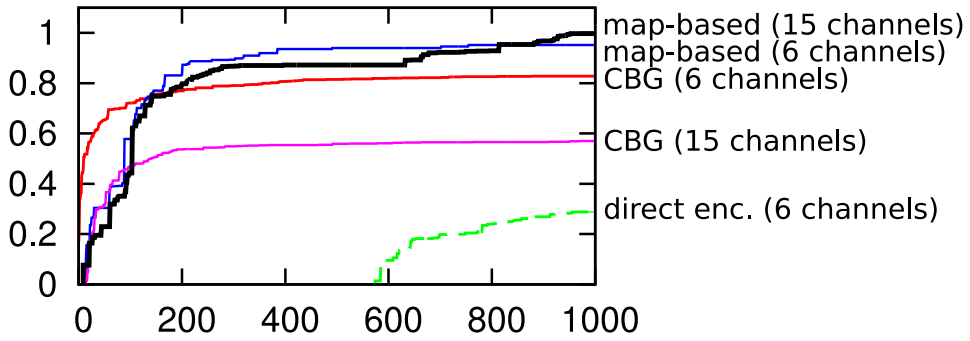


FIG. 30: Fitness médiane par génération pour la tâche de sélection d'action. La fitness maximale est de 1. Les expériences CBG correspondent à l'optimisation des paramètres du modèle des ganglions de la base sur lequel ces travaux se sont appuyés. Le codage direct n'a pas généré de solutions efficaces pour le cas à 15 canaux et n'apparait donc pas sur le graphe. La faible performance du modèle CBG peut s'expliquer par le fait que l'évaluation ne prend pas en compte d'autres fonctionnalités réalisées par ce modèle. CBG respecte également des contraintes biologiques qui n'étaient pas imposées au codage EvoNeuro.

sont avérés plus robustes avec une meilleure capacité de généralisation [Pinville and Doncieux, 2010]. L'étude des capacités de ce codage et des méthodes de comparaison avec des modèles de neurosciences sont en cours.

3.5 DISCUSSION

Les travaux portant sur les pressions de sélection ont été réalisés chronologiquement après les travaux sur ModNet et MENNAG. Ils ont mis en évidence la déficience de l'exploration lorsque l'on ne tenait compte que d'une pression de sélection orientée vers le but à résoudre. Dès lors qu'un objectif poussant à l'exploration dans l'espace des comportements est pris en compte, de bien meilleurs résultats sont obtenus, même avec les codages les plus simples. Les limitations précédemment identifiées, quant au passage à l'échelle en terme de complexité, semblent donc dûes plus à une défaillance d'exploration qu'à un problème de codage, ce qui était l'hypothèse implicite de nombreux travaux en robotique évolutionniste. Divers travaux se sont intéressés à la question de la comparaison des codages, notamment des codages dits directs et des codages dits indirects, pour arriver à des conclusions assez mitigées. Le codage cellulaire de Gruau, codage indirect, s'est révélé plus efficace qu'un codage direct [Gruau et al., 1996] avant que des travaux plus récents ne montrent une meilleure performance de NEAT, un codage direct, par rapport au codage cellulaire [Stanley and Miikkulainen, 2002]. Au regard de nos résultats, il semble que de telles comparaisons soient prématurées. Les bonnes performances de NEAT, qui inclut un schéma de protection de la diversité, pourrait donc être dûes à ce point plus qu'au choix d'un codage direct et à la définition des opérateurs génétiques. Il semble possible qu'un codage indirect associé à un schéma de protection de la diversité similaire à celui de NEAT ou à un de ceux que nous avons proposés obtienne des performances comparables, au moins sur les tâches couramment considérées (contrôle d'un pendule inversé, navigation réactive), dont on peut penser qu'elles ne sont pas d'une complexité suffisante pour qu'un codage indirect prouve ses avantages.

Les méthodes proposées reposant sur la définition de pressions de sélection particulières, ont l'avantage d'être très localisées et donc compatibles avec de nombreux codages ou algorithmes de sélection. Par rapport à un algorithme comme NEAT, qui inclut un codage ainsi qu'un algorithme de sélection particulier, elles ne nécessitent aucune adaptation pour passer de réseaux de neurones à des systèmes à base de règles floues ou à tout autre type de contrôleur. De même, des algorithmes de sélection variés peuvent être utilisés, à partir du moment où ils gèrent des problèmes multi-objectif. Au-delà d'un intérêt pratique, la relative indépendance de cette contribution permet une réutilisation immédiate dans de futurs travaux portant sur le codage, par exemple, ou sur d'autres pressions ou schémas de sélection. Nous avons ainsi utilisé un objectif de diversité comportementale dans la plupart de nos travaux récents. D'une manière générale, ces travaux relèvent d'une démarche dans laquelle nous avons cherché à bien identifier l'apport de la nouvelle approche, et nous avons donc proposé un incrément bien localisé au milieu d'algorithmes issus de l'état de

l'art, plutôt que de proposer un nouveau système complet dont le positionnement est difficile à identifier et les comparaisons délicates[6].

Des alternatives à l'évolution de la structure du réseau de neurones apparaissent actuellement. Les *Echo State Networks* (ESN) [Jaeger, 2001] sont des réseaux récurrents dont les poids normalisés permettent à la couche cachée d'agir comme un réservoir de dynamiques. En apprenant les poids reliant cette couche aux sorties [Hartland and Bredeche, 2007] ou également certains paramètres de ces réseaux [Jiang et al., 2008], il est possible de contrôler le comportement de robots. L'avantage de telles approches est de simplifier la partie évolution et de pouvoir utiliser des algorithmes d'optimisation plus classiques comme CMA-ES [Hansen and Ostermeier, 2001]. Si quelques expériences ont utilisé ce type de réseau leur efficacité, comparativement à celle des structures soumises à évolution, n'a pas encore été évaluée. Si ce type d'approche est concurrent avec des codages directs, il ne permet cependant pas de gérer la modularité des solutions. L'avantage des travaux réalisés sur les pressions de sélection est qu'ils sont complètement indépendants du codage utilisé[7] et qu'ils peuvent donc s'appliquer directement à l'évolution de contrôleurs basés sur des ESN.

HyperNEAT [Stanley et al., 2009; Gauci and Stanley, 2010] est un codage récent permettant d'exploiter des régularités géométriques. Son principe consiste à positionner les neurones dans un substrat géométrique et à synthétiser un réseau de neurones (avec des fonctions de transfert non standard), appelé CPPN pour *Compositional Pattern Producing Network*, qui sera utilisé pour calculer les poids des connexions entre chaque neurone. Pour calculer le poids reliant des neurones n_i et n_j, ce CPPN reçoit en entrée les coordonnées de ces neurones, ainsi qu'un biais, et la valeur de sortie du CPPN est interprétée comme le poids de la connexion reliant n_i à n_j. Comme EvoNeuro, HyperNEAT permet donc de manipuler des cartes de neurones. Il est plus souple qu'EvoNeuro dans la mesure où les schémas de connexion sont moins stéréotypés. Par contre, il impose de définir initialement la structure spatiale et donc le nombre de cartes neuronales. EvoNeuro et HyperNEAT sont en fait relativement complémentaires dans la mesure où EvoNeuro permet de connecter plusieurs cartes entre elles et HyperNEAT pourrait être utilisé pour faire évoluer ces connexions plutôt que de les choisir dans un répertoire prédéfini de schémas de connexions.

6 Un article de journal a été soumis avec des comparaisons très variées sur la façon de prendre en compte, ou pas, la diversité comportementale.

7 L'exaptation nécessite la notion de module dans le codage, mais n'impose pas un codage modulaire particulier.

4 DE LA SIMULATION À LA RÉALITÉ

MATIÈRES

Ce chapitre présente les travaux récents que nous avons réalisés sur le transfert des solutions obtenues en simulation sur des robots réels. Ce sont des travaux encore en cours et moins aboutis que les travaux présentés dans les précédents chapitres. Deux approches différentes sont présentées ; toutes deux reposent sur un nombre aussi réduit que possible de tests sur le robot réel et utilisent l'information ainsi acquise dans le processus de sélection. La première approche, dite de coévolution simulation et contrôle, cherche à apprendre un modèle du robot et de ses interactions avec l'environnement adapté à la tâche pendant la résolution de celle-ci. La seconde approche considère que ce modèle est fixe et va rechercher en priorité les comportements se transférant le plus efficacement sur le robot réel.

4.1 INTRODUCTION

La robotique évolutionniste permet d'optimiser ou de synthétiser des robots et leur système de contrôle sur la base d'un grand nombre d'essais et erreurs. Cette caractéristique rend ces algorithmes susceptibles d'exploiter des particularités du robot et de son interaction avec l'environnement. Si cette caractéristique est généralement souhaitée, de par sa capacité à proposer des solutions simples et astucieuses [Doncieux and Meyer, 2003], elle peut être un handicap dès lors qu'il s'agit de transférer des solutions obtenues en simulation sur un robot réel. En effet, une disparité entre simulation et réalité est quasiment inévitable, d'autant plus que les méthodes évolutionnistes utiliseront en priorité des simulations simplifiées, car plus rapides. Il est donc possible que les caractéristiques exploitées par l'évolution soient propres à la simulation et donc que le comportement obtenu sur le robot réel soit très différent de celui observé en simulation.

L'évolution peut ainsi être réalisée directement sur robot réel [Floreano and Mondada, 1994; Nordin and Banzhaf, 1996], ce qui pose cependant le problème du temps requis pour atteindre la convergence, ainsi que celui des pannes qui peuvent survenir suite à l'usure d'une utilisation intensive du robot avec, régulièrement, des comportements dégénérés et dangereux pour l'intégrité du robot.

La simulation semble difficilement contournable, en particulier pour des comportements complexes nécessitant de nombreuses évaluations. Plusieurs approches ont été proposées pour remédier au problème de la disparité simulation-réalité. Jakobi [1997] propose de définir une simulation minimale, n'incluant que les éléments bien modélisés et entourant les autres d'une enveloppe de bruit afin de faire en sorte que l'évolution ne les utilise pas. La difficulté de ce type d'approche est qu'il est difficile de savoir quelles parties sont bien modélisées et lesquelles sont mal modélisées et que le réglage de l'enveloppe de bruit garante de la robustesse des solutions générées est délicat.

D'autres approches proposent de considérer le passage de la simulation à la réalité comme une variation de l'environnement à laquelle des contrôleurs doivent pouvoir s'adapter en ligne. Hartland and Bredeche [2006] propose d'utiliser un module anticipateur afin de détecter toute disparité et de prendre les mesures correctives adaptées. Floreano and Urzelai [2001] propose de générer un réseau robuste doté de plasticité synaptique. Dans ces deux cas, le comportement recherché doit être relativement similaire en simulation et en réalité pour que les changements soient à la portée d'une adaptation locale.

Nous avons exploré deux approches différentes pour aborder cette question :

1. coévolution simulation - système de contrôle de comportement
2. optimisation de la transférabilité

Ces deux approches se distinguent essentiellement par les hypothèses qui les motivent. Dans le premier cas, il est supposé qu'aucune simulation adaptée n'est disponible ou que la simulation utilisée peut être améliorée, le processus évolutionniste est donc utilisé pour en concevoir une en même temps qu'il explorera les systèmes de contrôle du robot. Dans le second cas, la simulation est supposée non modifiable.

Dans les deux cas, la dégradation de performance pendant le passage de la simulation à la réalité est minimisée en réalisant des expériences sur le robot réel. L'objectif est d'en réaliser le moins possible, car elles sont coûteuses en temps et potentiellement dangereuses pour le dispositif robotique. Les résultats de ces expériences sont utilisés dans le processus évolutionniste qui proposera ultérieurement d'autres expériences à réaliser.

4.2 COÉVOLUTION SIMULATION ET CONTRÔLE

Contexte: Ces travaux ont été réalisés dans le cadre de la thèse de Sylvain Koos
Publication(s) associée(s): [Koos et al., 2009]

Modéliser convenablement un système robotique est une tâche difficile qui peut être abordée par une approche évolutionniste. Nordin and Banzhaf [1997] et Nordin et al. [1998] présentent ainsi l'utilisation de la programmation génétique pour apprendre un modèle d'un robot Khépera avant de choisir l'action la plus appropriée par une recherche exhaustive exploitant les prédictions du modèle. Dans les travaux

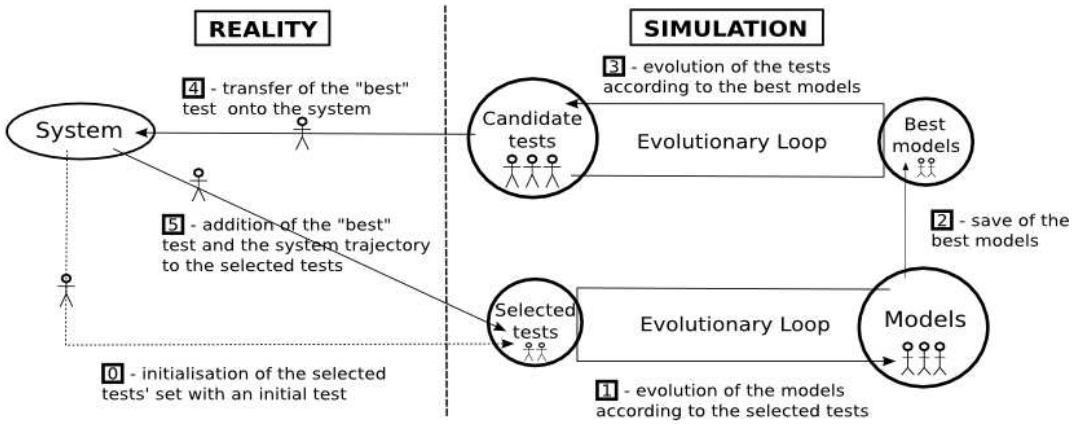


FIG. 31: Vue d'ensemble de l'algorithme de coévolution simulation et contrôle. Un test est constitué d'un contrôleur, d'un état initial et d'un état cible. Un test sélectionné contient également la trajectoire correspondante sur le système réel.

présentés ici, faisant l'hypothèse qu'une telle recherche à chaque pas de temps serait trop coûteuse, nous avons cherché à concevoir simultanément un système de contrôle, le modèle servant dans une boucle évolutionniste de conception comme vu précédemment.

Plutôt que de se reposer sur une approche de type apprentissage supervisé, qui nécessite de disposer d'une grande base d'apprentissage, donc de faire de nombreux tests, une approche évolutionniste a été utilisée ici explicitement pour piloter l'apprentissage du modèle. Le but ne sera pas de générer un modèle complet du robot et de son interaction avec l'environnement, mais plutôt un modèle adapté à la tâche à résoudre, autrement dit un modèle qui ne sera précis que dans le domaine d'évolution du robot. L'algorithme évolutionniste utilisé pour la synthèse ou l'optimisation du contrôleur peut alors également être utilisé pour piloter l'apprentissage du modèle dans une approche de co-évolution. Bongard and Lipson [2005] proposent l'algorithme d'estimation-exploration consistant à faire coévoluer des modèles avec des tests. Les meilleurs modèles sont ceux qui ont un comportement similaire au système cible sur les tests retenus et les meilleurs tests sont les plus discriminants, autrement dit ceux qui parviennent à différencier au mieux les modèles candidats. Ces meilleurs tests sont ensuite transférés sur le système réel afin de disposer de nouvelles données sur lesquelles évaluer les modèles candidats. de Jong and Pollack [2004] proposent la Pareto-coévolution, une approche plus théorique de la coévolution apprenant-test basée sur une évaluation multi-objectif : chaque test donne lieu a un objectif.

L'approche proposée consiste à utiliser la Pareto-coévolution dans le contexte de l'algorithme d'estimation exploration (figure 31). Des modèles sont donc optimisés en même temps que des contrôleurs visant à la fois à améliorer les modèles et à résoudre une tâche donnée. Les modèles visent à prédire au mieux le comportement du système robotique considéré, ils sont évalués pour cela sur la base des tests qui ont été transférés sur ce système avec deux objectifs par test. Ces objectifs cherchent à minimiser la disparité entre les trajectoires obtenues en simulation et sur le système

réel en boucle ouverte pour le premier objectif et en boucle fermée pour le second. Trois objectifs sont utilisés pour évaluer les tests. Le premier objectif est la capacité d'un test à discriminer les modèles candidats. Le second cherche à minimiser la variance des résultats observés, dès lors que l'on change légèrement les paramètres du test. Cet objectif permet d'éviter que les tests ne se concentrent sur les instabilités du système modélisé. Le troisième et dernier objectif mesure la qualité du contrôle.

Cette approche a été appliquée à la modélisation-commande d'un drone de type quadricoptère dont la dynamique a été modélisée à l'aide de la programmation génétique linéaire [Heywood and Zincir-Heywood, 2000]. Si les résultats obtenus sont prometteurs, une utilisation de cette approche sur des tâches plus complexes nécessitera d'aborder la question du nombre d'objectifs utilisés. En effet, l'évaluation des modèles requiert deux objectifs par test, le nombre d'objectifs augmente donc très rapidement, alors que les performances des algorithmes évolutionnistes multi-objectif se détériorent au-delà de trois objectifs.

4.3 OPTIMISATION DE LA TRANSFÉRABILITÉ

Contexte: Ces travaux ont été réalisés dans le cadre de la thèse de Sylvain Koos
Publication(s) associée(s): [Koos et al., 2010]

La seconde approche proposée considère la simulation comme une donnée non modifiable. En conséquence, elle va chercher à générer des solutions qui se transfèrent bien en évitant donc les comportements les moins robustes au passage simulation-réalité. L'idée consiste donc à évaluer la transférabilité, définie comme étant la différence entre le comportement du robot en simulation et en réalité, tout en minimisant le nombre de transferts sur le robot réel. Cette transférabilité sera donc approchée pour la plupart des individus sur la base des transferts déjà réalisés.

L'algorithme développé repose sur l'évaluation d'une diversité comportementale $d(x)$, définie comme suit :

$$d(x) = \min_{y \in C_T} b_{dist}(x, y) \tag{4.1}$$

avec C_T l'ensemble des individus déjà transférés sur le système réel, $b_{dist}(x, y) = ||\mathbf{b}^{(x)} - \mathbf{b}^{(y)}||^2$, $\mathbf{b}^{(x)}$ étant un vecteur de valeurs caractéristiques du comportement.

La disparité entre la simulation et la réalité est alors évaluée de la façon suivante :

$$\hat{D}(x) = \frac{1}{N} \sum_{y \in C_T} \frac{D^*(y)}{d(x,\ y)} \tag{4.2}$$

avec $N = \sum_{y \in C_T} \frac{1}{d(x,\ y)}$, C_T étant l'ensemble des contrôleurs transférés sur le système réel pour lesquels la disparité exacte $D^*(y)$ est donc connue.

Trois objectifs sont à maximiser :

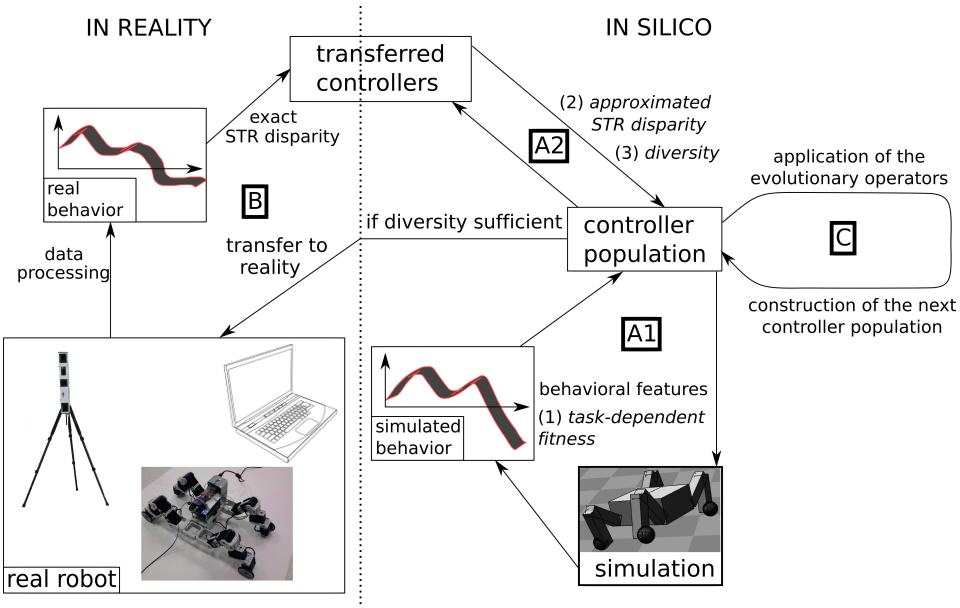


FIG. 32: Schéma de l'algorithme d'optimisation de la transférabilité.

– la fitness liée à la tâche à résoudre
– la disparité approchée entre simulation et réalité $\hat{D}(x)$
– la diversité $d(x)$

L'algorithme est le suivant (figure 32) :

A. évaluation des contrôleurs :
   A1. calcul de la fitness dépendant de la tâche et des vecteurs de caractéristiques comportementales ;
   A2. évaluation des deux autres objectifs relativement aux contrôleurs déjà transférés ;
B. si certains contrôleurs ont une diversité dépassant un seuil donné, un parmi eux est transféré sur le système réel ;
C. application des opérateurs génétiques et construction de la population de la génération suivante.

L'algorithme a été testé sur une tâche de locomotion pour un robot construit en Bioloïdes et mimant le robot Hylos (figure 33). Les résultats sont présentés sur la figure 34. Les expériences avec minimisation de la disparité ont généré des contrôleurs se transférant plus efficacement. La distance parcourue sur le robot réel est en moyenne plus grande pour ces expériences.

4.4 DISCUSSION

Dans les approches proposées, des expériences sont menées sur le robot réel pendant le processus de recherche évolutionniste afin de le diriger vers les solutions

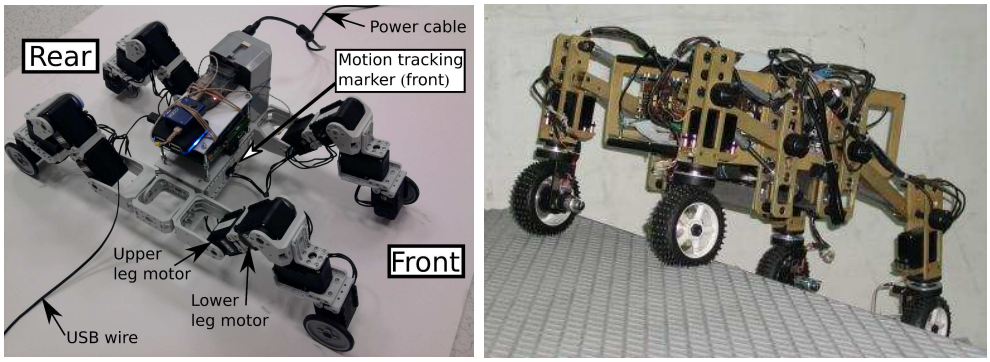


FIG. 33: Gauche : le robot Bioloïde utilisé pour les expériences de transfert simulation-réalité. Droite : le robot Hylos dont s'inspire le robot Bioloïde.

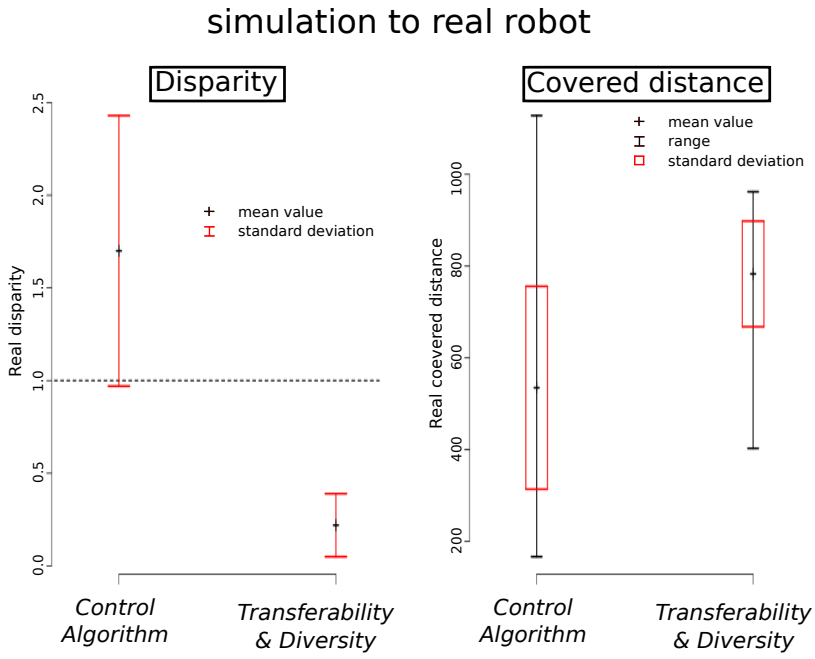


FIG. 34: Résultat des expériences d'optimisation du contrôleur de marche dans l'expérience de contrôle (fitness : distance parcourue) et dans l'expérience de minimisation de la disparité (fitness : 3 objectifs, distance parcourue, diversité, transférabilité). Gauche : disparité réelle mesurée à l'issue des expériences (erreur moyenne au carré normalisée) ; droite : distance parcourue.

les plus intéressantes, et surtout d'éviter des solutions exploitant des caractéristiques propres à la simulation. Ce type d'approche peut également s'appliquer dans d'autres contextes. Le principe est que l'on dispose d'une évaluation très coûteuse – celle qui est réalisée sur le robot réel – et d'une évaluation peu coûteuse – celle qui est réalisée en simulation. Le grand nombre de tests à réaliser implique de disposer d'une évaluation rapide des solutions potentielles. Les travaux présentés pourraient alors se transposer facilement à une optimisation du temps avant convergence si on considère d'un côté une simulation rapide, mais peu précise, et d'un autre une simulation lente, mais très précise. De même, il est possible de considérer plusieurs versions d'un robot : une version robuste simplifiée sur laquelle les évaluations peuvent opérer en toute sécurité et la version finale du robot, plus efficace, mais plus fragile. Il serait alors intéressant de minimiser le nombre d'évaluations sur le robot fragile tout en garantissant que le comportement obtenu sera fonctionnel sur ce dernier.

Un point est fondamental pour les approches proposées : comment évaluer la différence entre deux comportements ? Cette distance comportementale sert à évaluer quantitativement la différence entre le comportement d'une solution dans la simulation et sur le système réel, mais également pour évaluer la distance entre le comportement d'une solution potentielle et d'une autre solution qui a été transférée, cette distance intervenant dans l'évaluation de la valeur approchée de la transférabilité. Il est intéressant de noter que ce besoin se présente également dans un contexte très différent, celui de la diversité comportementale présentée précédemment (voir section 3.3.2). Cette question de la définition d'un comportement et de comparaisons quantitatives entre comportements se retrouve donc au centre de travaux très divers et méritera donc une attention accrue dans les années à venir.

5 DISCUSSION

MATIÈRES

Différentes utilisations des algorithmes évolutionnistes dans un contexte de conception orientée vers le comportement ont été présentées, chacune séparément et indépendamment du reste du processus. À quoi pourrait ressembler ce processus de conception complet ? Nous aborderons cette question avant de discuter de la portée de la méthode proposée et des applications envisageables. Nous terminerons avec une mise en perspective de la robotique évolutionniste considérée dans un cadre plus ouvert que celui d'une conception ayant un début et une fin clairement identifiée. Sans changement fondamental de l'approche, cela permettrait une adaptation à des changements drastiques et imprévus de l'environnement sans intervention extérieure, capacité présente dans le monde du vivant du fait de l'évolution naturelle.

5.1 PROCESSUS DE CONCEPTION BASÉ COMPORTEMENT

Les différents travaux présentés dans ce manuscrit peuvent s'insérer dans un processus de conception de robots mobiles et autonomes centré sur le comportement. Cette section présente un exemple typique des différentes étapes d'un tel processus.

5.1.1 Se doter d'un simulateur fiable et rapide

Le principal prérequis est de disposer d'un simulateur fiable et aussi rapide que possible. Il doit permettre de réaliser de nombreux tests préliminaires sur des fonctions de fitness ou le type de contrôleur choisi, sans risquer d'endommager un robot. Lorsque tous les éléments ont bien été validés, le simulateur est encore utilisé afin d'accélérer la convergence.

5.1.2 Explorer et améliorer la morphologie

Sur la base d'une morphologie donnée (éventuellement paramétrée), des algorithmes évolutionnistes peuvent être utilisés au début du processus de conception pour explorer les capacités du robot dans une démarche de type « conception aidée par évolution » (chapitre 2.2). Ce sera particulièrement intéressant, lorsque les outils à disposition des ingénieurs ne permettent pas de prédire avec précision les performances du robot. Plusieurs itérations peuvent être réalisées pendant lesquelles la morphologie du robot peut être améliorée sur la base des informations obtenues à l'itération précédente.

5.1.3 Rechercher un comportement

Le comportement souhaité peut ensuite être développé, après avoir affiné la morphologie ou l'espace de recherche ouvert à exploration, pendant à l'étape d'exploration préliminaire. Cette étape peut se baser sur une optimisation évolutionniste (chapitre 2) ou sur une synthèse évolutionniste (chapitre 3).

5.1.4 Application sur le robot réel

Si le comportement doit être directement appliqué sur le robot, il est important de prendre des mesures permettant d'assurer que le comportement obtenu en simulation sera tout aussi efficace sur le système réel. Cela peut se faire par les méthodes proposées au chapitre 4.

Une alternative consiste à étudier les comportements générés et à synthétiser des contrôleurs les exhibant par des méthodes plus traditionnelles. C'est un moyen de bénéficier des garanties de stabilité de telles méthodes, par exemple, tout en exploitant la créativité du processus évolutionniste.

5.2 PORTÉE DE LA MÉTHODE PROPOSÉE

En pratique, la plupart des comportements obtenus par évolution jusqu'à présent sont essentiellement de type réactifs. Le comportement est choisi sur la base des perceptions courantes sans prendre en compte d'historique quel qu'il soit ou sans anticipation. Le comportement de collecte de balle est, à ma connaissance, un des comportements les plus compliqués, si ce n'est le plus compliqué, obtenu par évolution sans recourir à une décomposition du problème. Pourquoi cette limite sur les comportements presque exclusivement réactifs ? Le passage à des comportements plus cognitifs pose un certain nombre de problèmes, certains étant d'ordre pratique, d'autres étant d'ordre plus théorique.

D'un point de vue pratique, évaluer un comportement réactif est simple : il suffit de plonger le robot dans différents contextes et d'observer son comportement. Le robot ne déterminant son comportement que sur la base de ses perceptions instantanées, l'évaluation sera ainsi représentative de ce que le robot sera capable de

faire dans des circonstances similaires. Dès lors qu'une forme de mémoire est nécessaire, l'évaluation s'en trouve considérablement compliquée. Il faut en effet prévoir beaucoup plus d'expériences pour s'assurer d'une capacité de généralisation acceptable : toutes les combinaisons temporelles possibles des événements pertinents sont susceptibles de mener à un comportement différent, alors que précédemment, il suffisait d'observer la réponse à chaque événement indépendamment de l'historique. Le temps passé à l'évaluation d'une solution candidate sera donc nécessairement plus long pour les tâches plus cognitives. Le risque en gardant des évaluations courtes, est que l'évolution trouve des contrôleurs réactifs qui, sur les conditions d'évaluation considérées, donne de bonnes performances, mais ne généralisent pas de façon satisfaisante.

D'un point de vue plus théorique, on peut se poser la question de l'évolvabilité d'un contrôleur cognitif. Comment pourrait apparaître un système de mémorisation ou d'anticipation un tant soit peu sophistiqué s'il n'a aucun intérêt tant que son fonctionnement n'est pas parfait ? Comment la recherche évolutionniste pourrait façonner un tel module pour le rendre opérationnel, si rien dans la fonction de fitness ne l'encourage à le faire ? A quoi pourrait servir un proto-module de mémoire ? Dans le cas d'un système réactif, on peut faire l'hypothèse que de résoudre partiellement le problème entraîne la recherche dans la bonne voie. Lorsque l'on cherche à obtenir via évolution des capacités plus cognitives, il est très possible que cette hypothèse ne tienne plus. Si les performances se dégradent très vite dès lors que l'on s'écarte de la solution recherchée, la solution ne peut être trouvée que par chance, lorsqu'une mutation ou un croisement génère exactement cette solution. La probabilité de convergence est alors très faible. Dans ce cas, s'il existe des solutions réactives, elles seront trouvées en priorité et attireront donc la recherche dans une voie de garage. L'exaptation a été proposée pour résoudre ce problème au chapitre 3.4.2. La recherche de nouveauté pourrait être une solution alternative [Lehman and Stanley, 2010]. Il est en tout cas fort probable que la solution de ce problème comprenne une étude des pressions de sélection.

5.3 APPLICATIONS

La conception de robots mobiles et autonomes et plus particulièrement la conception du système gérant leur comportement, est le domaine d'application qui a motivé ces travaux. L'utilisation de l'optimisation de paramètres dans ce contexte est fréquente, voire par exemple [Fleming and Purshouse, 2002] pour une vue d'ensemble. La conception aidée par évolution est encore rarement mise en œuvre et des études approfondies doivent encore être menées pour affiner l'approche d'analyse multi-objectif proposée et identifier clairement les cas d'utilisation et les réponses que cette approche peut apporter. La synthèse évolutionniste est encore peu appliquée. La raison tient à la complexité de mise en œuvre de ces méthodes, ainsi que le gain pas toujours significatif dès lors que le concepteur a eu recours à une fonction de fitness incluant de nombreuses connaissances a priori. Les nouvelles approches

utilisant la nouveauté et la diversité comportementale pourraient changer cet état de fait.

D'autres utilisations qu'en robotique sont envisageables. La biologie est un domaine qui pourrait ainsi bénéficier de ces travaux. Il a déjà été mis en avant que la robotique évolutionniste pouvait être un outil pour étudier des comportements minimalement cognitifs [Harvey et al., 2005]. Par ailleurs, des outils d'optimisation ont déjà été utilisés pour valider ou invalider des hypothèses concernant le comportement d'animaux et répondre à des questions comme celle de savoir quel critère ils optimisent ou combien de critères ils optimisent simultanément [Schmitz et al., 1997]. Plusieurs expériences sont lancées avec des critères différents et les résultats qui correspondent le mieux à la réalité sont réputés confirmer la prépondérance du ou des critères correspondants. Globalement, le schéma suivi pour ce type de travaux consiste à s'intéresser à un processus dans lequel il est raisonnable de penser qu'une forme d'optimisation est à l'œuvre – ce qui est souvent vérifié du fait de la sélection naturelle – et ensuite de proposer un modèle. Ce modèle est soumis à optimisation et les résultats sont comparés aux observations afin de le valider. Disposer de méthodes efficaces d'optimisation de comportement ouvre de nouvelles perspectives à ce type de démonstration en permettant d'étudier quel type de comportement différents critères permettent de générer via une synthèse de contrôleurs. Une autre possibilité serait de partir d'un codage particulier, incluant ou interdisant certains modules ou connexions connus par les neuroscientifiques afin d'étudier, de manière plus exhaustive qu'une exploration manuelle, les capacités qu'offrent ces différents éléments.

Une telle utilisation pourrait avoir un impact significatif en contribuant à la compréhension des processus ayant conduit aux comportements des êtres vivants tel que l'on peut les observer aujourd'hui. Beaucoup de travail nécessite cependant d'être entrepris pour qu'une telle utilisation soit véritablement envisageable. Les méthodes doivent être simples d'utilisation pour que des biologistes se les approprient. Ils doivent être à même de les intégrer dans leur raisonnement et de tirer des conclusions significatives et incontestables sur le plan méthodologique.

5.4 VERS DES ROBOTS RÉSILIENTS

Afin d'évoluer dans notre environnement, nous devons constamment nous adapter à celui-ci. Certaines adaptations sont drastiques, lorsque nous nous cassons une jambe, par exemple, nous sommes capables d'apprendre à marcher avec des béquilles. Les robots actuels sont loin de disposer de telles capacités, pourtant leur utilisation pratique le nécessitera. En effet, si un ingénieur ou un technicien doit suivre le robot à chacun de ses pas de façon à le réparer en cas de besoin, le déploiement de robots sera coûteux et se fera donc à petite échelle, uniquement pour des applications de défense, à très fort impact économique ou dans des zones facilement accessibles. Le déploiement de robots sur des planètes lointaines, par exemple, implique une robustesse conséquente du robot. Plus il sera à même de se tirer de situations délicates par lui-même, plus il sera possible d'étendre le champ d'action

et d'exploration de ces robots, actuellement limités par le temps de latence dans les échanges entre la Terre et la planète considérée. Le besoin est donc fort de disposer de robots véritablement adaptatifs, capables de changer leur comportement pour répondre à des modifications importantes de leur environnement ou de leur propre morphologie. La conception de robots *résilients* pourrait s'appuyer sur une méthode automatique de conception orientée vers le comportement, pour peu que cette méthode puisse être appliquée en ligne, pendant le fonctionnement du robot.

Un moyen pour atteindre cet objectif, serait de maintenir un modèle interne du monde, constamment adapté aux observations réalisées, et à l'utiliser en permanence pour apprendre comment se mouvoir ou résoudre une tâche donnée, comme proposé dans [Bongard et al., 2006]. Une alternative pourrait être de considérer une nuée de robots et de s'affranchir de modèle interne. La nuée de robots serait soumise à une évolution embarquée, très proche de la sélection naturelle [Watson et al., 2002; Bredeche and Montanier, 2010].

Ce type d'application de la robotique évolutionniste semble très prometteur compte tenu de l'importante capacité d'adaptation qu'elle confèrerait aux robots concernés, même si de nombreux travaux restent à entreprendre pour atteindre cet objectif.

6 PROJET DE RECHERCHE

MATIÈRES

Si le domaine de l'automatique s'intéresse à la commande de robot sur la base du suivi d'une consigne, aucune approche ne s'est imposée pour l'instant en ingénierie dès lors qu'il s'agit d'aller au-delà et de concevoir le comportement d'un robot, voire de le doter de capacités cognitives. L'ambition des travaux présentés ici est de développer des méthodes de conception automatisées ou semi-automatisées permettant de générer des contrôleurs (et éventuellement aussi des morphologies) sur la base d'une description de ce qu'ils doivent réaliser. Partant de comportements actuellement réactifs, l'objectif est, de plus, de tendre vers des comportements de plus en plus cognitifs.

6.1 ÉTUDE DE LA NOTION DE COMPORTEMENT

Les travaux présentés dans ce manuscrit ont illustré à plusieurs reprises l'importance de la notion de comportement. L'évaluation quantitative de la similarité entre comportements a ouvert plusieurs possibilités :

- pousser à la diversité comportementale pour éviter le problème du démarrage et de la convergence prématurée ;
- réduire la disparité simulation-réalité.

L'inefficacité de la recherche mise en évidence par les travaux portant sur la diversité rend également intéressant le développement de critères permettant de vérifier quantitativement à quel point l'algorithme a pu explorer, voire à quel point il est toujours en train d'explorer des comportements différents. Disposer de ce type d'information serait précieux à plusieurs titres.

Il peut enrichir des comparaisons entre approches. En effet, actuellement la comparaison entre approches repose sur l'étude des performances obtenues sur un ou plusieurs problèmes. L'approche résolvant le problème le plus efficacement ou le plus rapidement est réputée comme étant la plus intéressante. Ce raisonnement peut cependant conduire à des conclusions trop hâtives, notamment dans le cas où une des approches comparées est biaisée vers une catégorie de solutions particulières. Par exemple, un codage peut être biaisé pour explorer en priorité des réseaux non récurrents. Il ne sera alors pas surprenant qu'il soit plus performant sur des pro-

blèmes ne nécessitant pas de récurrences, qu'un codage qui n'aurait pas ce biais, indépendamment de tout autre aspect algorithmique. Une comparaison ne prenant pas en compte ces biais ne serait donc pas juste. Or, il n'est pas toujours évident de les remarquer et, dès lors qu'ils sont connus, il n'est pas non plus aisé de les prendre en compte. Si un critère permettant d'évaluer la capacité à explorer dans l'espace des comportements est connu, il pourra entrer dans les critères pris en compte pendant la comparaison pour la rendre plus juste. Cela pourrait aussi suggérer d'autres bancs de tests qui seraient des problèmes plus ouverts, avec une sélection basée sur la recherche de nouveauté par exemple, pour lesquels seraient étudiés, tant qualitativement que quantitativement, les types de comportements générés indépendamment d'une tâche particulière.

De plus, ne prendre en compte que la performance finale revient à agréger deux aspects bien distincts : la capacité d'exploration et la capacité d'exploitation, autrement dit l'amélioration des solutions déjà trouvées. Disposer d'un critère évaluant la capacité d'exploration permettrait d'évaluer plus précisément les avantages d'une approche particulière et peut encourager le développement d'approches hybrides mêlant les qualités de plusieurs approches distinctes. Ceci ne peut se faire que si ces qualités peuvent être identifiées.

La capacité à explorer l'espace des comportements peut également se révéler utile dès lors que l'on aborde des problèmes hors de portée des algorithmes actuels. Dans ce cas, comment progresser ? A partir du moment où le problème n'est pas résolu, il est difficile de comparer des approches, de savoir laquelle est la plus prometteuse. De plus, il est également difficile d'évaluer l'impact d'une modification si cela ne permet pas non plus de résoudre le problème. L'approche actuellement préconisée consiste à décomposer le problème et à adopter une approche incrémentale. Cependant, sous l'hypothèse que la non résolution du problème est dûe à une défaillance de l'exploration, l'utilisation d'un critère d'exploration pourrait guider des travaux de recherche sur des approches candidates à résoudre directement le problème dans son ensemble. Dans le cas où aucune approche ne parvient à résoudre le problème, la plus intéressante serait alors celle qui explore le mieux.

L'étude de la notion de comportement pourrait donc avoir un double impact sur la robotique évolutionniste et les méthodes de recherche associées :

- permettre des comparaisons plus informées donc plus rigoureuses ;
- proposer une approche pour aborder des problèmes actuellement insolubles.

Ces travaux devront trouver comment avoir une vue synthétique de l'espace de recherche exploré et surtout comment comparer l'exploration réalisée lors d'expériences différentes. Des méthodes de fouille de données pourraient être utilisées dans ce but, mais il reste à identifier précisément les besoins afin de trouver les outils algorithmiques les plus adaptés, s'ils existent, ou d'en développer de nouveau en s'inspirant de l'existant si cela s'avère nécessaire.

6.2 WHY NOT THE WHOLE IGUANA ?

Dans les travaux de robotique évolutionniste réalisés jusqu'à présent, peu des fonctionnalités utiles à la « survie » d'un robot autonome ont été considérées simultanément. L'objectif de synthétiser des systèmes de contrôle de plus en plus cognitifs n'est donc clairement pas encore atteint, sans parler de la prise en compte des aspects perception et morphologie. Ceux-ci nécessitent une interaction étroite avec des spécialistes de ces domaines afin de définir les primitives sur lesquelles pourra se baser une exploration évolutionniste. S'il est difficile de prévoir à ce jour la direction que prendraient de tels travaux, il est certain que l'ISIR en général et l'équipe SIMA en particulier offrent un cadre privilégié à l'étude de cette question compte tenu de la présence d'experts en perception et en robotique/mécatronique.

Si l'on se concentre sur le contrôle de comportement, les travaux présentés dans ce manuscrit mettent en lumière une approche susceptible de mener, si ce n'est à l'Iguane complet cher à Dennett [1978], au moins au « cerveau » qui pourrait être le sien. Une telle approche nécessite d'apporter une réponse à deux questions :

- quelle représentation génétique utiliser ?
- quelle méthodologie pour garantir, autant que possible, le progrès des travaux de recherche envisagés ?

6.2.1 Quelle représentation ?

Dans la perspective d'intégrer à terme toutes les capacités requises pour rendre un robot autonome dans une approche de conception de type évolutionniste et dans le cadre du projet ROBUR, j'ai participé à des travaux portant sur l'évitement d'obstacles par flux optique [Muratet et al., 2004, 2005; Doncieux and Angeli, 2007] ou encore la cartographie et localisation simultannée [Angeli et al., 2006, 2008a,b, 2009]. Ces différents travaux exploitent un formalisme très différent de celui des réseaux de neurones utilisé précédemment. S'il est très probable qu'une approche évolutionniste peut optimiser les paramètres de tels systèmes, dans une perspective de synthèse évolutionniste d'un système cognitif complet, il faut disposer d'une représentation permettant de générer, si besoin est, un tel système de cartographie ou de perception en plus des autres fonctionnalités requises pour la survie du robot. La définition de cette représentation reste une question ouverte.

À l'opposé, les êtres vivants sont capables de résoudre des tâches de navigation complexes et certains animaux disposent de capacités cognitives qui rendrait des robots véritablement autonomes et à même d'accomplir une mission plus efficacement qu'aujourd'hui. Il se trouve, de plus, que les systèmes nerveux de ces animaux peuvent se décrire dans un formalisme relativement unifié : celui des réseaux de neurones. Si les modèles issus des neurosciences computationnelles sont bien plus complexes que les réseaux de neurones considérés en robotique évolutionniste, les premiers résultats obtenus avec le codage EvoNeuro montrent qu'il est possible d'inclure de nouvelles primitives structurelles dans des codages de réseaux de neurones et donc de s'approcher de ces modèles. L'objectif d'un tel codage, à terme, est de disposer d'une expressivité suffisante pour couvrir les différentes fonctionnalités im-

pliquées dans la navigation ou dans des tâches plus cognitives. Le but sera de savoir que de générer un cerveau simplifié doté de capacités similaires à celles d'un vertébré est *possible* avec ce formalisme, ce qui est une première étape importante. Cela laisse par contre ouverte la question de savoir comment générer un tel cerveau.

6.2.2 Quelle méthodologie ?

Les travaux sur l'exaptation ont montré la difficulté à résoudre un problème pourtant simple, sans prendre en compte les fonctionnalités internes du système de contrôle dans la pression de sélection. Il est très probable que d'autres méthodes permettent de résoudre ce problème ; néanmoins, ces travaux pointent du doigt la difficulté d'une approche directe où l'on cherche à obtenir un réseau résolvant une tâche complexe. Que faire si l'évolution ne génère pas de solutions intéressantes ? La diversité comportementale et la nouveauté sont des approches à explorer pour améliorer la recherche, mais permettront-elles de passer à l'échelle pour atteindre des comportements véritablement cognitifs ? Rien ne permet de l'affirmer à ce jour. Disposer de critères d'évaluation de la capacité d'exploration dans l'espace des comportements peut également aider, mais il n'est pas certain non plus que ce soit suffisant.

Le codage EvoNeuro (section 3.4.3) permet de générer des modèles similaires à ceux des neurosciences computationnelles, des comparaisons entre les modèles générés par évolution et les modèles conçus par les neuroscientifiques seront donc possibles. De plus, les différentes fonctionnalités identifiées par les chercheurs en neurosciences (sélection de l'action, mémoire de travail, etc.) ainsi que leurs protocoles pourront être directement réutilisés. Les premiers résultats obtenus sur des fonctionnalités de sélection de l'action [Mouret et al., 2010] et de mémoire de travail [Pinville and Doncieux, 2010] sont très encourageants.

Au-delà de l'étude d'une fonctionnalité séparée se posera ensuite la question de l'intégration de ces différentes fonctionnalités, intégration qui pourrait reposer sur l'exaptation présentée précédemment (section 3.4.2). Le principal défaut de cette méthode est qu'il nécessite de disposer d'une connaissance importante sur toutes les fonctionnalités intermédiaires utiles à la résolution d'un problème particulier. Cette constatation vient à l'encontre des objectifs initiaux de disposer d'une méthode de conception automatique nécessitant une faible connaissance du problème considéré. Cette limitation pourrait être contournée si la méthode d'exaptation se révèle capable de converger même lorsque de nombreuses pressions de sélection sont données, dont certaines qui ne seraient pas du tout utiles pour la tâche considérée. Des connaissances *a priori* seraient toujours nécessaires. Cependant, il serait alors possible de définir un répertoire de fonctionnalités qui serait utilisé dans tous les cas, même si toutes ne seraient pas systématiquement utiles. Atteindre ce but avec l'approche d'exaptation proposée nécessiterait de disposer d'algorithmes multi-objectif efficaces même avec un grand nombre d'objectifs, ce qui est encore un problème ouvert faisant l'objet d'intenses recherches. Une telle approche nécessitera de toute façon d'explorer beaucoup plus de solutions que ce qui est actuellement réalisé. Cela pourrait être obtenu en augmentant drastiquement la taille des populations considé-

rées et le nombre de générations, mais il est également possible que cela nécessite des travaux de recherche spécifiques pour conserver une recherche efficace sur une si grande échelle.

BIBLIOGRAPHIE

Abbass, H. A. and Deb, K. (2003). Searching under multi-evolutionary pressures. *Proceedings of the Fourth Conference on Evolutionary Multi-Criterion Optimization, Spain*. (Cité page 40.)

Allen, B. and Faloutsos, P. (2009). Complex networks of simple neurons for bipedal locomotion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4457–4462. IEEE. (Cité page 31.)

Allen, M. J. (2005). Autonomous Soaring for Improved Endurance of a Small Uninhabited Air Vehicle. In *Meeting Paper AIAA-2005-1025, Research Engineering, NASA Dryden Flight Research Center*. (Cité page 12.)

Allen, M. J. (2006). Updraft Model for Development of Autonomous Soaring Uninhabited Air Vehicles. *44 th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1–19. (Cité page 18.)

Altenberg, L. (2005). Modularity in Evolution : Some Low-Level Questions. In Callebaut, W. and Rasskin-Gutman, D., editors, *Modularity : Understanding the Development and Evolution of Natural Complex Systems*. MIT Press. (Cité page 50.)

Amari, S.-i. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27(2) :77–87. (Cité page 54.)

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2006). 2D Simultaneous Localization And Mapping for Micro Aerial Vehicles. In *Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference*. (Cité page 75.)

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008a). Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. *IEEE Transactions on Robotics*, 24(5) :1027–1037. (Cité page 75.)

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008b). Real-Time Visual Loop-Closure Detection. In *in the proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. (Cité page 75.)

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2009). Visual topological SLAM and global localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*. (Cité page 75.)

Back, T., Hoffmeister, F., and Schwefel, H. (1991). A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. (Cité page 11.)

Back, T. and Schwefel, H. P. (1995). Evolution Strategies I : Variants and their Computational Implementation. In *Genetic Algorithms in Engineering and Computer Science, Proc.˜First Short Course EUROGEN-95*. Wiley. (Cité aux pages 6 et 44.)

Baele, G., Bredeche, N., Haasdijk, E., Maere, S., Michiels, N., Van˜de peer, Y., Schmickl, T., Schwarzer, C., and Thenius, R. (2009). Open-ended On-board Evolutionary Robotics for Robot Swarms. In *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*. (Cité page 8.)

Barate, R., Doncieux, S., and Meyer, J.-A. (2006). Design of a bio-inspired controller for dynamic soaring in a simulated {UAV}. *Bioinspiration & Biomimetics*, 1 :76–88. (Cité aux pages vi, 13 et 15.)

Beer, R. D. and Gallagher, J. C. (1992). Evolving Dynamical Neural Networks for Adaptive Behavior. *Adaptive Behavior*, 1(1) :91–122. (Cité aux pages 7 et 12.)

Blasco, X., Herrero, J., Sanchis, J., and Martinez, M. (2008). A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization. *Information Sciences*, 178(20) :3908–3924. (Cité page 25.)

Boers E., J. W., Kuiper, H., Happel, B. L. M., and Springhuizen-Kuiper, I. G. (1993). Designing modular artificial neural networks. Technical report, Rijksuniversiteit te Leiden. (Cité aux pages 45 et 46.)

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient Machines Through Continuous Self-Modeling. *Science*, 314(5802) :1118–1121. (Cité aux pages 8 et 71.)

Bongard, J. C. and Lipson, H. (2005). Nonlinear System Identification Using Coevolution of Models and Tests. *Evolutionary Computation, IEEE Transactions on*, 9(4) :361–384. (Cité page 61.)

Bredeche, N. and Montanier, J. (2010). Environment-driven Embodied Evolution in a Population of Autonomous Agents. *Parallel Problem Solving from Nature–PPSN XI*, PPSN 6239 :290—-299. (Cité page 71.)

Brooks, R. A. (1991). Intelligence without Representation. *Artificial Intelligence*, 47 :139—-159. (Cité page 3.)

Buessler, J. L. and Urban, J. P. (2003). *Modular neural architectures for robotics*, pages 261—-298. Springer Verlag. (Cité page 45.)

Bui, L. T., Abbass, H. A., and Branke, J. (2005). Multiobjective optimization for dynamic environments. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 3 :2349–2356 Vol. 3. (Cité aux pages 39 et 40.)

Cliff, D., Harvey, I., and Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behavior*, 2 :73—-110. (Cité page 7.)

Coates, M. I. and Clack, J. A. (1990). Polydactyly in the earliest known tetrapod limbs. *Nature*, 347(6288) :66–69. (Cité page 50.)

Cochrane, J. H. (1999). MacCready Theory with Uncertain Lift and Limited Altitude. *Technical Soaring*, 23 :88—-96. (Cité page 12.)

Collins, S., Ruina, A., Tedrake, R., and Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science (New York, N.Y.)*, 307(5712) :1082–5. (Cité page 3.)

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London : John Murray. (Cité aux pages 4, 5 et 50.)

de Jong, E. D. and Pollack, J. B. (2004). Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2) :159—-192. (Cité page 61.)

de Jong, E. D. and Thierens, D. (2004). Exploiting modularity, hierarchy, and repetition in variable-length problems. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, pages 1030–1041. (Cité page 46.)

de Jong, E. D., Watson, R. A., and Pollack, J. B. (2001). Reducing bloat and promoting diversity using multi-objective methods. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. (Cité aux pages 39 et 40.)

de Jong, K. A. (2006). *Evolutionary Computation : A Unified Approach*. MIT Press. (Cité page 6.)

de Margerie, E., Mouret, J.-B., Doncieux, S., and Meyer, J.-A. (2007). Artificial evolution of the morphology and kinematics in a flapping-wing mini {UAV}. *Bioinspir. Biomim.*, 2 :65–82. (Cité aux pages vi, 21 et 27.)

de Margerie, E., Tafforeau, P., and Rakotomanana, L. (2006). In silico evolution of functional morphology : A test on bone tissue biomechanics. *Journal of the Royal Society, Interface / the Royal Society*, 3(10) :679–87. (Cité page 19.)

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley. (Cité page 11.)

Deb, K., Mohan, M., and Mishra, S. (2003). A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. *KanGAL report*. (Cité page 18.)

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197. (Cité page 8.)

Deb, K. and Srinivasan, A. (2007). INNOVIZATION : Discovery of Innovative Design Principles Through Multiobjective Evolutionary Optimization. *Multiobjective Problem Solving from Nature : From Concepts to Applications*, page 243. (Cité aux pages 8 et 20.)

Delarboulas, P., Schoenauer, M., and Sebag, M. (2010). Open-Ended Evolutionary Robotics : an Information Theoretic Approach. In *PPSN XI, Krakow, Poland*. (Cité page 43.)

DeLaurier, J. D. (1993). An aerodynamic model for flapping-wing flight. *Aeronautical Journal*, 97(964) :125–130. (Cité page 25.)

Dennett, D. C. (1978). Why not the whole iguana ? *Behavioural and Brain Sciences*, 1 :103—-104. (Cité page 75.)

Doncieux, S. (2003). *{É}volution de contrôleurs neuronaux pour animats volants : méthodologie et applications*. PhD thesis, LIP6/AnimatLab, Université Pierre et Marie Curie, Paris, France. (Cité page 45.)

Doncieux, S. (2009). Evolutionary algorithms as exploration and analysis helper tools, application to a flapping wing aircraft. In *Proceedings of IROS Workshop "Exploring New Horizons in the Evolutionary Design of Robots"*. (Cité aux pages 26 et 28.)

Doncieux, S. and Angeli, A. (2007). Navigation des drones par flux optique. *Objets volants miniatures : modélisation et commande embarquée*, pages 305–328. (Cité page 75.)

Doncieux, S. and Hamdaoui, M. (2010). Evolutionary Algorithms to Analyse and Design a Controller for a Flapping Wings Aircraft. In Doncieux, S., Bredeche, N., and Mouret, J.-B., editors, *New Horizons in Evolutionary Robotics : post-proceedings of the 2009 EvoDeRob workshop*. Studies in Computational Intelligence, Springer. (Cité aux pages vi, 26 et 28.)

Doncieux, S. and Meyer, J.-A. (2003). Evolving Neural Networks for the Control of a Lenticular Blimp. In Raidl, G. R., Cagnoni, S., Romero Cardalda, J. J., Corne, D. W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C. G., Marchiori, E., Meyer, J.-A., and Middendorf, M., editors, *Applications of Evolutionary Computing, EvoWorkshops2003 : Evo{BIO}, Evo{COP}, Evo{IASP}, Evo{MUSART}, Evo{ROB}, Evo{STIM}*. Springer Verlag. (Cité aux pages vii, 45 et 59.)

Doncieux, S. and Meyer, J.-A. (2004a). Evolution of neurocontrollers for complex systems : alternatives to the incremental approach. In *Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004)*. (Cité page 45.)

Doncieux, S. and Meyer, J.-A. (2004b). Evolving Modular Neural Networks to Solve Challenging Control Problems. In *Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*. (Cité aux pages vii et 33.)

Doncieux, S. and Meyer, J.-A. (2005). Evolving PID-like neurocontrollers for nonlinear control problems. *Control and intelligent systems*, 33(1) :55–62. (Cité page 45.)

Doncieux, S. and Mouret, J.-B. (2010). Behavioral diversity measures for Evolutionary Robotics. In *IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)*. (Cité aux pages vii, 38, 41 et 42.)

Doncieux, S., Mouret, J.-B., Angeli, A., Barate, R., Meyer, J.-A., and De˜Margerie, E. (2006). Building an Artificial Bird : Goals and Accomplishments of the {ROBUR} Project. In *Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference*. (Cité page 12.)

Doncieux, S., Mouret, J.-B., Bredeche, N., and Padois, V. (2010). Evolutionary Robotics : Exploring New Horizons. In Doncieux, S., Bredeche, N., and Mouret, J.-B., editors, *New Horizons in Evolutionary Robotics : post-proceedings of the 2009 EvoDeRob workshop*. Studies in Computational Intelligence. Springer-Verlag. (Cité aux pages v et 8.)

Doncieux, S., Mouret, J.-B., and Meyer, J.-A. (2007). Soaring behaviors in UAVs : 'animat' design methodology and current results. In *7th European Micro Air Vehicle Conference (MAV07)*, Toulouse. (Cité aux pages vi, 15 et 18.)

Doncieux, S., Mouret, J. B., Muratet, L., and Meyer, J.-A. (2004). The {ROBUR} project : towards an autonomous flapping-wing animat. In *Proceedings of the Journées MicroDrones*, Toulouse. (Cité page 12.)

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system : optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 26(1) :29–41. (Cité page 6.)

Druot, T. (2004). Technical report on the implementation and validation of a flight mechanics simulator for flapping articulated wings. (Cité aux pages 23, 27 et 33.)

Edwards, D. J. (2008). Implementation Details and Flight Test Results of an Autonomous Soaring Controller. *Report from North Carolina State University, Raleigh, NC, 27604*. (Cité page 12.)

Eiben, A. E. and Smith, J. E. (2008). *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer. (Cité page 5.)

Filliat, D., Kodjabachian, J., and Meyer, J.-A. (1999). Evolution of neural controllers for locomotion and obstacle-avoidance in a 6-legged robot. *Connection Science*, 11 :223—-240. (Cité page 31.)

Fleming, P. J. and Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering : a survey. *Control Engineering Practice*, 10(11) :1223–1241. (Cité aux pages 8 et 69.)

Floreano, D., Dürr, P., and Mattiussi, C. (2008a). Neuroevolution : from architectures to learning. *Evolutionary Intelligence*, 1(1) :47–62. (Cité page 8.)

Floreano, D., Husbands, P., and Nolfi, S. (2008b). Evolutionary Robotics. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 1423–1451, Berlin, Heidelberg. Springer Berlin Heidelberg. (Cité page 8.)

Floreano, D. and Mattiussi, C. (2001). Evolution of Spiking Neural Controllers for Autonomous Vision-Based Robots. *Evolutionary Robotics. From Intelligent Robotics to Artificial Life*, 2217 :38–61. (Cité page 12.)

Floreano, D. and Mattiussi, C. (2008). *Bio-{I}nspired {A}rtificial {I}ntelligence : {T}heories, {M}ethods, and {T}echnologies*. Intelligent Robotics and Autonomous Agents. MIT Press. (Cité page 8.)

Floreano, D., Miglino, O., and Parisi, D. (1991). Emergent Complex Behaviors in Ecosystems of Neural Networks. In Caianiello, E., editor, *Parallel Architectures and Neural Networks*, Singapore. World Scientific Press. (Cité page 7.)

Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent : genetic evolution of a neural-network driven robot. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3 : from animals to animats 3*, pages 421–430. (Cité aux pages 12, 44 et 59.)

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*. (Cité aux pages 8 et 44.)

Floreano, D., Nolfi, S., and Mondada, F. (1998). Competitive Co-Evolutionary Robotics : From Theory to Practice. In Pfeifer, R., Blumberg, B., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats : Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*. MIT Press. (Cité page 44.)

Floreano, D. and Urzelai, J. (2001). Evolution of plastic control networks. *Autonomous Robots*, 11(3) :311–317. (Cité page 60.)

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons Inc, New York - London - Sydney. (Cité page 6.)

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization : formulation, discussion and generalization. In *Proceedings of the fourth International Conference on Evolutionary Programming*, pages 416—-423. (Cité page 33.)

Freitas, A. A. (2002). *Data Mining and Knowledge Discovery With Evolutionary Algorithms*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K. (Cité page 20.)

Frigg, R. and Hartmann, S. (2009). Models in science. *Stanford Encyclopedia of Philosophy*. (Cité page 19.)

Gauci, J. and Stanley, K. O. (2010). Autonomous Evolution of Topographic Regularities in Artificial Neural Networks. *Neural Computation*, 22(7) :1860—-1898. (Cité page 57.)

Georgopoulos, a., Schwartz, a., and Kettner, R. (1986). Neuronal population coding of movement direction. *Science*, 233(4771) :1416–1419. (Cité page 54.)

Glover, F. (1989). Tabu Search–Part I. *INFORMS Journal on Computing*, 1(3) :190–206. (Cité page 6.)

Glover, F. (1990). Tabu Search–Part II. *INFORMS Journal on Computing*, 2(1) :4–32. (Cité page 6.)

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Educational Publishers Inc. (Cité aux pages 6, 11 et 44.)

Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 41–49. (Cité aux pages 39 et 40.)

Gomez, F. and Miikkulainen, R. (1997). Incremental Evolution of Complex General Behavior. *Adaptive Behavior*, 5(3-4) :317–342. (Cité page 35.)

Gomez, F. J. (2009). Sustaining diversity using behavioral information distance. In *GECCO'09 : Proceedings of the 11th annual conference on Genetic and Evolutionary Computation*, pages 113–120. ACM. (Cité page 43.)

Gould, S. J. and Vrba, E. S. (1982). Exaptation ; a missing term in the science of form. *Paleobiology*, 8(1) :4–15. (Cité page 50.)

Gross, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous Self-Assembly in Swarm-Bots. *IEEE Transactions on Robotics*, 22(6) :1115–1130. (Cité page 8.)

Gruau, F. (1992). Cellular Encoding of Genetic Neural Networks. Technical Report 9221, Laboratoire de l'informatique du Parallélisme, Ecole Normale Supérieure de Lyon. (Cité page 45.)

Gruau, F. (1994). *Synthèse de Réseaux de Neurones par Codage Cellulaire et Algorithmes Génétiques*. PhD thesis, ENS Lyon, Université Lyon I. (Cité page 47.)

Gruau, F. (1995). Automatic Definition of Modular Neural Networks. *Adaptive Behaviour*, 3(2) :151–183. (Cité aux pages 31, 45 et 46.)

Gruau, F., Whitley, D., and Pyeatt, L. (1996). A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks. In John Koza, R., David Goldberg, E., David Fogel, B., and Rick Riolo, L., editors, *Genetic Programming 1996 : Proceedings of the First Annual Conference*, pages 81–89, Stanford University, CA, USA. MIT Press. (Cité page 56.)

Haasdijk, E., Eiben, A., and Karafotias, G. (2010). On-line evolution of robot controllers by an encapsulated evolution strategy. In *WCCI 2010 IEEE World Congress on Computational Intelligence, IEEE CEC*, pages 3547—-3553. (Cité page 12.)

Hamdaoui, M. (2009). *Optimisation multicritères de l'efficacité propulsive de mini-drones biomimétiques à ailes battantes par algorithmes évolutionnaires*. PhD thesis, Université Pierre et Marie Curie. (Cité aux pages 23 et 24.)

Hamdaoui, M., Chaskalovic, J., Doncieux, S., and Sagaut, P. (2010). Using Multiobjective Evolutionary Algorithms and Data Mining Methods to Optimize Bird-like UAVs' Kinematics. *Journal of Aircraft*, 47(5) :1504—-1516. (Cité aux pages vi et 23.)

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2) :159–195. (Cité aux pages 8 et 57.)

Happel B., L. M. and Murre J., M. J. (1994). The Design and Evolution of Modular Neural Network Architectures. *Neural Networks*, 7 :985–1004. (Cité page 45.)

Hara, F. and Pfeifer, R. (2003). *Morpho-Functional Machines : The New Species : Designing Embodied Intelligence*. Springer-Verlag Tokyo, Inc. (Cité page 3.)

Hartland, C. and Bredeche, N. (2006). Evolutionary Robotics, Anticipation and the Reality Gap. In *Proc. of ROBIO'06*, pages 1640–1645. (Cité page 60.)

Hartland, C. and Bredeche, N. (2007). Using echo state networks for robot navigation behavior acquisition. *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 201–206. (Cité page 57.)

Harvey, I., Di Paolo, E. A., Tuci, E., Wood, R., and Quinn, M. (2005). Evolutionary robotics : A new scientific tool for studying cognition. *Artificial Life*, 11(1–2) :79—-98. (Cité page 70.)

Harvey, I., Husbands, P., and Cliff, D. (1994). Seeing the light : artificial evolution ; real vision. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S., editors, *From Animals to Animats 3, Proceedings of the third international conference on Simulation of Adaptive Behavior*. MIT Press/Bradford Books. (Cité aux pages 35 et 50.)

Hauert, S., Zufferey, J. C., and Floreano, D. (2009). Reverse-engineering of {A}rtificially {E}volved {C}ontrollers for {S}warms of {R}obots. In *{IEEE} {C}ongress on {E}volutionary {C}omputation*. (Cité page 18.)

Hertzberg, J. and Chatila, R. (2008). AI Reasoning Methods for Robotics. *Handbook of Robotics*, pages 207–223. (Cité page 2.)

Heywood, M. J. and Zincir-Heywood, A. N. (2000). Page-based linear genetic programming. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 5, pages 3823—-3828. (Cité page 62.)

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems : An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. (1st edition 1975). MIT Press. (Cité page 6.)

Hornby, G. S. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of GECCO'2005*, pages 1729–1736. (Cité page 46.)

Hornby, G. S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., and Fujita, M. (2000). Evolving Robust Gaits with AIBO. In *IEEE International Conference on Robotics and Automation*, pages 3040–3045. (Cité page 8.)

Hughes, R. I. G. (1997). Models and Representation. *Philisophy of science*, 64(Supplement Proceedings of the 1996 Biennial Meetings of the Philosophy of Science Association. Part II : Symposia Papers) :325—-336. (Cité page 20.)

Husbands, P. and Harvey, I. (1992). Evolution versus design : Controlling autonomous robots. In *Integrating Perception, Planning and Action, Proceedings of 3rd Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. (Cité page 7.)

Husbands, P., Harvey, I., Cliff, D., and Miller, G. (1997). Artificial evolution : a new path for artificial intelligence ? *Brain and cognition*, 34(1) :130–59. (Cité page 7.)

Hussain, T. S. (2003). *Attribute Grammar Encoding of the Structure and Behaviour of Artificial Neural Networks*. PhD thesis, Queen's University. (Cité aux pages 46 et 47.)

Ijspeert, A. J. and Kodjabachian, J. (1999). Evolution and Development of a Central Pattern Generator for the Swimming of a Lamprey. *Artificial Life*, 5(3) :247—-269. (Cité page 31.)

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. (Cité page 57.)

Jakobi, N. (1997). Evolutionary Robotics and the Radical Envelop of Noise Hypothesis. *Adaptive Behavior*, 6(1) :131–174. (Cité page 60.)

Jiang, F., Berry, H., and Schoenauer, M. (2008). Supervised and evolutionary learning of echo state networks. In *Parallel Problem Solving from Nature–PPSN X*, pages 215–224. Springer. (Cité page 57.)

Jin, Y. (2003). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1) :3–12. (Cité page 20.)

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science (New York, N.Y.)*, 220(4598) :671–680. (Cité page 6.)

Knowles, J. and Nakayama, H. (2008). Meta-modeling in multiobjective optimization. *Multiobjective Optimization, LNCS*, 5252 :245–284. (Cité page 20.)

Knuth, D. E. (1968). Semantics of context-free languages. *Theory of Computing Systems*, 2(2) :127–145. (Cité page 46.)

Kodjabachian, J. and Meyer, J. (1995). Evolution and Development of Control Architectures in Animats. *Robotics and Autonomous Systems*, 16(2-4) :161–182. (Cité page 44.)

Kodjabachian, J. and 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 :796–812. (Cité aux pages 8, 31, 35 et 50.)

Kohonen, T. (2001). *Self-Organizing Maps*, volume 1. Springer, Berlin, Germany, 3rd edition. (Cité page 25.)

Koos, S., Mouret, J.-B., and Doncieux, S. (2009). Automatic system identification based on coevolution of models and tests. In *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*. (Cité aux pages viii et 60.)

Koos, S., Mouret, J.-B., and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers. In *GECCO'10 : Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher*. (Cité aux pages viii et 62.)

Koza, J. R. (1993). *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press. (Cité page 6.)

Koza, J. R. (1994). *Genetic Programming II : Automatic Discovery of Reusable Programs*. MIT Press. (Cité page 6.)

Larose, D. T. (2005). *Data Mining Methods And Models*. John Wiley & Sons Inc. (Cité page 25.)

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers. (Cité page 2.)

Lehman, J. and Stanley, K. O. (2008). Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Artificial Life*, volume 11. (Cité page 39.)

Lehman, J. and Stanley, K. O. (2010). Abandoning Objectives : Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, page (à paraître). (Cité aux pages 31, 39 et 69.)

Levine, W. S. (1996). *The Control Handbook*. CRC Press Inc. (Cité page 2.)

Liénard, J., Guillot, A., and Girard, B. (2010). Multi-Objective Evolutionary Algorithms to Investigate Neurocomputational Issues : The Case Study of Basal Ganglia Models. In Meyer, J.-A., Guillot, A., and Hallam, J., editors, *From animals to animats 11*, LNAI. Springer. (Cité aux pages 21 et 54.)

Lipson, H. (2004). Principles of Modularity, Regularity, and Hierarchy for Scalable Systems. *Genetic and Evolutionary Computation Conference (GECCO'04) Workshop on Modularity, regularity and Hierarchy*. (Cité page 46.)

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic life forms. *Nature*, 406(31) :974–978. (Cité aux pages 8 et 31.)

MacCready, P. B. (1958). Optimum airspeed selector. *Soaring magazine*, pages 10—-11. (Cité page 12.)

Mahfoud, S. W. (1997). Niching methods. In *Handbook of Evolutionary Computation*, pages 87—-92. Taylor & Francis. (Cité page 39.)

Mamdani, E. H. (1974). Application of Fuzzy Algorithms for Control of Simple Dynamic Plants. In *Institution of Electrical Engineers*. (Cité page 13.)

Maris, M. and Te Boekhorst, R. (1996). Exploiting Physical Constraints : Heap Formation through Behavioral Error in a Group of Robots. In *In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS-96*, pages 1655—-1660. (Cité page 3.)

Mataric, M. J. and Michaud, F. (2008). *Behavior-Based Systems*, pages 891–909. Springer. (Cité page 3.)

Meyer, J.-A. (1995). The Animat Approach to Cognitive Science. In Roitblat, H. L. and Meyer, J.-A., editors, *Comparative Approaches to Cognitive Science*, pages 27—-44. The MIT Press. (Cité page 2.)

Meyer, J.-A., Doncieux, S., Filliat, D., and Guillot, A. (2003). Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots. In *Biologically inspired robot behavior engineering*, pages 1—-43. (Cité page 31.)

Meyer, J.-A. and Guillot, A. (2008). Biologically-inspired robots. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*. Springer-Verlag. (Cité page 2.)

Michalewicz, Z. and Fogel, D. B. (1999). *How to Solve It : Modern Heuristics*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K. (Cité page 6.)

Minoux, M. (2007). *Programmation mathématique : Théorie et algorithmes*. Tec & Doc Lavoisier. (Cité page 5.)

Mondada, F. and Floreano, D. (1995). Evolution of neural control structures : some experiments on mobile robots. *Robotics and Autonomous Systems*, 16(2-4) :183–195. (Cité page 54.)

Mondada, F., Franzi, E., and Ienne, P. (1993). Mobile robot miniaturisation : A tool for investigation in control algorithms . In *Proceedings of the third International Symposium on Experimental Robots*, pages 501—-513. Springer Verlag. (Cité page 54.)

Morgan, M. S. (1999). Learning from models. *Models as mediators : perspective on natural and social science.*, pages 347—-388. (Cité page 20.)

Moriguchi, H. and Honiden, S. (2010). Sustaining Behavioral Diversity in NEAT. In *GECCO'10 : Proceedings of the 12th annual conference on Genetic and Evolutionary Computation*, pages 611–618. ACM. (Cité page 43.)

Mouret, J.-B. (2008). *Pressions sélectives multiples pour l'évolution de réseaux de neurones destinés à la robotique*. PhD thesis, Université Pierre et Marie Curie (UPMC). (Cité aux pages 37, 38 et 49.)

Mouret, J.-B. and Doncieux, S. (2008a). Incremental Evolution of Animats' Behaviors as a Multi-objective Optimization. In *From Animals to Animats 10*, volume 5040, pages 210–219. Springer. (Cité aux pages vii, 37, 38 et 50.)

Mouret, J.-B. and Doncieux, S. (2008b). MENNAG : a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. *Evolutionary Intelligence*, 1 :187–207. (Cité aux pages vii, 45, 46 et 47.)

Mouret, J.-B. and Doncieux, S. (2009a). Evolving modular neural-networks through exaptation. In *IEEE Congress on Evolutionary Computation (CEC'09)*. (Cité aux pages viii et 49.)

Mouret, J.-B. and Doncieux, S. (2009b). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*. (Cité aux pages vii, 38 et 41.)

Mouret, J.-B. and Doncieux, S. (2009c). Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. In *GECCO'09 : Proceedings of the 11th annual conference on Genetic and evolutionary computation*. ACM. (Cité aux pages vii, 38 et 41.)

Mouret, J.-B., Doncieux, S., and Girard, B. (2010). Importing the Computational Neuroscience Toolbox into Neuro-Evolution—Application to Basal Ganglia. In *GECCO'10 : Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. (Cité aux pages ix, 53, 54 et 76.)

Mouret, J.-B., Doncieux, S., and Meyer, J. A. (2006). Incremental Evolution of Target-Following Neuro-controllers for Flapping-Wing Animats. In *From Animals to Animats : Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*, pages 606–618, Rome, Italy. (Cité aux pages vii, 8, 35, 45 et 50.)

Mouret, J. B., Doncieux, S., Muratet, L., Druot, T., and Meyer, J.-A. (2004). Evolution of neuro-controllers for flapping-wing animats. In *Proceedings of the Journées MicroDrones*, Toulouse. (Cité aux pages 33 et 45.)

Muratet, L., Doncieux, S., and and J.A. Meyer (2004). A biomimetic reactive navigation system using the optical flow for a rotary-wing UAV in urban environment. In *Proceedings of ISR2004, Paris*. (Cité page 75.)

Muratet, L., Doncieux, S., Brière, Y., and Meyer, J.-A. (2005). A Contribution to Vision-Based Autonomous Helicopter Flight in Urban Environments. *Robotics and Autonomous Systems*, 50(4) :195–209. (Cité page 75.)

Nakayama, H., Arakawa, M., and Washino, K. (2003). *Optimization for Black-box Objective Functions*, pages 185—-210. World Scientific Pub Co Inc. (Cité page 20.)

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics : The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press Bradford Books. (Cité page 8.)

Nolfi, S. and Paris, D. (1995). Evolving non-Trivial Behaviors on Real Robots : an Autonomous Robot that Picks up Objects. *Topics in Artificial Intelligence : 4th Conference of the Italian Association for Artificial Intelligence, AI* IA'95, Florence, Italy, October 11-13, 1995 : Proceedings*. (Cité page 35.)

Nordin, P. and Banzhaf, W. (1996). An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behavior*, 5(2) :107—-140. (Cité page 59.)

Nordin, P. and Banzhaf, W. (1997). Real time control of a Khepera robot using genetic programming. *Control and Cybernetics*, 26(3) :533—-561. (Cité page 60.)

Nordin, P., Banzhaf, W., and Brameier, M. (1998). Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, 25 :105—-116. (Cité page 60.)

Parisi, D., Cecconi, F., and Nolfi, S. (1990). Econets : Neural networks that learn in an environment. *Network : Computation in Neural Systems*, 1(2) :149–168. (Cité page 7.)

Parker, G. B. (2001). The Incremental Evolution of Gaits for Hexapod Robots. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1114–1121. (Cité page 35.)

Pennycuick, C. J. (1982). The flight of petrels and albatrosses ({P}rocellariiformes), observed in {S}outh {G}eorgia and its vicinity. *Phil. Trans. R. Soc. Lond. B*, 300 :75–106. (Cité aux pages 12 et 13.)

Pennycuick, C. J. (1990). Predicting Wingbeat Frequency and Wavelength of Birds. *J. Exp. Biol.*, 150(1) :171—-185. (Cité page 26.)

Pinville, T. and Doncieux, S. (2010). Automatic Synthesis of Working Memory Neural Networks with Neuroevolution Methods. In *Cinquième conférence française de Neurosciences Computationnelles (Neurocomp'10)*. (Cité aux pages ix, 53, 54, 56 et 76.)

Porto, V. W., Fogel, D. B., and Fogel, L. J. (1995). Alternative neural network training methods. *IEEE Expert Intelligent Systems And Their Applications*, 10(3) :16—-22. (Cité page 12.)

Rao, S. S. (2009). *Engineering Optimization : Theory and Practice*. John Wiley & Sons Ltd. (Cité page 5.)

Reil, T. and Husbands, P. (2002). Evolution of Central Pattern Generators for Bipedal Walking in a Real-Time Physics Environment. *IEEE Transactions on Evolutionary Computation*, 6(2) :159–168. (Cité page 12.)

Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2004). Evolving reusable neural modules. In *GECCO'04 : Proceedings of annual conference on Genetic and evolutionary computation*. Springer. (Cité page 46.)

Roitblat, H. L. (1995). Comparative Approaches to Cognitive Science. In Roitblat, H. L. and Meyer, J.-A., editors, *Comparative Approaches to Cognitive Science*, pages 13—-26. The MIT Press. (Cité page 2.)

Ronald, S. (1997). Robust encodings in genetic algorithms : a survey of encoding issues. *Evolutionary Computation, 1997., IEEE International Conference on*, pages 43–48. (Cité page 44.)

Roudenko, O. (2004). *Application des algorithmes évolutionnaires aux problèmes d'optimisation multi objectif avec contraintes*. PhD thesis. (Cité page 25.)

Sachs, G. (2005). Minimum shear wind strength required for dynamic soaring of albatrosses. *Ibis*, 147(1) :1–10. (Cité page 13.)

Schmitz, O. J., Cohon, J. L., Rothley, K. D., and Beckerman, A. P. (1997). Reconciling variability and optimal behaviour using multiple criteria in optimization models. *Evolutionary Ecology*, 12(1) :73–94. (Cité aux pages 19 et 70.)

Schwefel, H. and Rudolph, G. (1995). Contemporary Evolution Strategies. *Advances in artificial life*, pages 891–907. (Cité aux pages 6 et 11.)

Schwefel, H. P. and Back, T. (1995). Evolution Strategies II : Theoretical Aspects. In *Genetic Algorithms iEngineering and Computer Science, Proc. First Short Course EUROGEN-95*. Wiley. (Cité aux pages 6 et 44.)

Selig, M. (1997). Summary of low speed airfoil data - Vol 2. (Cité page 22.)

Shevell, R. S. (1989). *Fundamentals of flight*, chapter 3, pages 51–59. Prentice Hall, second edition. (Cité page 13.)

Shim, Y. and Husbands, P. (2007). Feathered Flyer : Integrating Morphological Computation and Sensory Reflexes into a Physically Simulated Flapping-Wing Robot for Robust Flight Manoeuvre. *Lecture Notes in Computer Science*, 4648 :756. (Cité page 8.)

Shyy, W., Berg, M., and Ljungqvist, D. (1999). Flapping and flexible wings for biological and micro air vehicles. *Progress in Aerospace Sciences*, 35 :455–505. (Cité page 25.)

Simon, H. A. (1962). The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106(6) :467–482. (Cité page 45.)

Sims, K. (1994). Evolving virtual creatures. In *SIGGRAPH '94 : Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM. (Cité page 31.)

Spaar, R. and Bruderer, B. (1997). Migration by flapping or soaring : flight strategies of {M}arsh, {M}ontagu's and {P}allid {H}arriers in southern {I}srael. *The Condor*, 99(2) :458–469. (Cité page 12.)

Sporns, O. (2002). Network analysis, complexity, and brain function. *Complexity*, 8(1) :56–60. (Cité page 44.)

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*. (Cité aux pages 40, 44, 47 et 56.)

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, 15(2) :185—-212. (Cité page 57.)

Stanley, K. O. and Miikkulainen, R. (2004). Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research*, 21 :63–100. (Cité page 30.)

Takagi, T. and Sugeno, M. (1985). Fuzzy Identification of Systems and its application to Modelling and Control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1). (Cité page 13.)

Tobalske, B. W., Hedrick, T. L., Dial, K. P., and Biewener, A. A. (2003). Comparative power curves in bird flight. *Nature*, 421(6921) :363–6. (Cité aux pages 23 et 26.)

Toffolo, A. and Benini, E. (2003). Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation*. (Cité page 39.)

Trujillo, L., Olague, G., Lutton, E., and De Vega, F. F. (2008). Behavior-based speciation for evolutionary robotics. In *GECCO'08 : Proceedings of the 10th annual conference on Genetic and Evolutionary Computation*, pages 297–298, New York, New York, USA. ACM. (Cité page 43.)

Urzelai, J., Floreano, D., Dorigo, M., and Colombetti, M. (1998). Incremental Robot Shaping. *Connection Science Journal*, 10(384) :341–360. (Cité page 50.)

Von Mises, R. (1959). *Theory of flight*, chapter 7, pages 139–169. Dover Publications, Inc. (Cité page 13.)

Wagner, G. P., Mezey, J., and Calabretta, R. (2005). Natural Selection and the Origin of Modules. In Callebaut, W. and Rasskin-Gutman, D., editors, *Modularity : Understanding the Development and Evolution of Natural Complex Systems*. MIT Press. (Cité page 50.)

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution : Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1) :1–18. (Cité page 71.)

Wharington, J. (1998). *Autonomous Control of Soaring Aircraft by Reinforcement Learning*. PhD thesis, Royal Melbourne Institute of Technology. (Cité page 12.)

Wieland, A. P. (1991). Evolving neural network controllers for unstable systems. *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, 2. (Cité page 12.)

Wilson, S. W. (1991). The Animat Path to AI. In Meyer, J.-A. and Wilson, S. W., editors, *From animals to animats : Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 15—-21. The MIT Press. (Cité page 2.)

Winkeler, J. F. and Manjunath, B. S. (1998). Incremental evolution in genetic programming. *Genetic Programming*, pages 403–411. (Cité page 35.)

Yao, X. (1999). Evolving artificial neural networks. In *Proceedings of the IEEE*, pages 1423–1447. (Cité page 44.)

Yeniay, O. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1) :45–56. (Cité page 5.)

Annexe

A SÉLECTION D'ARTICLES

A.1 MENNAG : A MODULAR, REGULAR AND HIERARCHICAL ENCODING FOR NEURAL-NETWORKS BASED ON ATTRIBUTE GRAMMARS

Mouret, J.B. and Doncieux, S. (2008). *MENNAG : a modular, regular and hierar- chical encoding for neural-networks based on attribute grammars*. **Evolutionary Intelligence**. Vol 1 No 3 Pages 187–207.

Travaux réalisés dans le cadre de la thèse de Jean-Baptiste Mouret dont j'étais encadrant scientifique.

Noname manuscript No.
(will be inserted by the editor)

Jean-Baptiste Mouret · Stéphane Doncieux

MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars

the date of receipt and acceptance should be inserted later

Abstract Recent work in the evolutionary computation field suggests that the implementation of the principles of modularity (functional localization of functions), repetition (multiple use of the same sub-structure) and hierarchy (recursive composition of sub-structures) could improve the evolvability of complex systems. The generation of neural networks through evolutionary algorithms should in particular benefit from an adapted use of these notions. We have consequently developed MENNAG (Modular Encoding for Neural Networks based on Attribute Grammars), a new encoding designed to generate the structure of neural networks and parameters with evolutionary algorithms, while explicitly enabling these three above-mentioned principles. We expressed this encoding in the formalism of attribute grammars in order to facilitate understanding and future modifications. It has been tested on two preliminary benchmark problems: cart-pole control and robotic arm control, the latter being specifically designed to evaluate the repetition capabilities of an encoding. We compared MENNAG to a direct encoding, ModNet, NEAT, a multi-layer perceptron with a fixed structure and to reference controllers. Results show that MENNAG performs better than comparable encodings on both problems, suggesting a promising potential for future applications.

Keywords Modular neural-networks · Evolutionary algorithms · Evolutionary robotics · Attribute grammars

1 Introduction

Engineers and software developers make great efforts to subdivide their creations into sub-entities which can be reused and combined. The similarity between this approach and biological evolution was famously emphasized by Simon (Simon, 1962) with the parable of the two watchmakers. The first craftsman constructed watches so that each part depended on the others. As a result, if he happened to be interrupted during assembly, the watches fell to pieces. The second watchmaker adopted a modular method by dividing the watch into several sub-parts. Simon then shows that the first one's chances of completing a watch are very slim indeed when compared to the second one's, so demonstrating that nature would have followed a similar path. Since the publication of this seminal paper, modularity has been widely recognized as an important feature of both living organisms

Jean-Baptiste Mouret · Stéphane Doncieux
Université Pierre et Marie Curie - Paris 6, FRE 2507, ISIR, 4 place Jussieu, F-75005 Paris, France
Tel.: (+33) 144278747
Fax: (+33) 144278809
E-mail: jean-baptiste.mouret@isir.fr

(Hartwell et al, 1999; Wagner and Altenberg, 1996) and artifacts. Many authors suggest that a modular design could increase both the evolvability and scalability of evolved systems (Wagner and Altenberg, 1996; Watson, 2005). In neuropsychology, double dissociation studies (Teuber, 1955), in which task deficits are mapped to brain damages, largely suggest a modular organization of animals' brain. This hypothesis have been recently refined using functional magnetic resonance imagery (fMRI, Huettel et al (2004); Anderson (2007)) to, for instance, analyze the object-vision pathway (de Beeck et al, 2008).

Recent papers in the evolutionary computation field (Lipson, 2004; de Jong and Thierens, 2004; Hornby, 2005) have clarified the *modularity* concept by associating it with *regularity* (sometimes called repetition) and *hierarchy*. Modularity is defined as the structural localization of function that enables a group of elements to be handled as a unit. In an evolutionary context, this is related to the building block hypothesis (Holland, 1975). Regularity is the repetition of similar sub-parts in a structure, making it more compressible. Hierarchy is the recursive composition of sub-structures. Current engineering practices, biological observations (Hartwell et al, 1999; Wagner and Altenberg, 1996; Anderson, 2007) and recent evolutionary algorithm experiments (Hornby, 2005) support the hypothesis that these three features are required in order to evolve complex systems.

There is no doubt that neural networks constitute such complex systems. They have proved their efficiency in many situations but are confronted to scalability and evolvability problems which could be addressed with a modular approach. Evolutionary robotics provides numerous examples of such situations, for instance when several legs have to be moved in a similar and synchronized manner (Kodjabachian and Meyer, 1997), or when the wings of an artificial bird have to be carefully controlled (Mouret et al, 2006). In this work as in many other instances, complex neuro-controllers with unknown optimal topologies are sought.

Though many papers have described methods to evolve modular neural networks (Gruau, 1995; Doncieux and Meyer, 2004a; Gallinari, 1998; Buessler and Urban, 2002; Boers et al, 1993; Reisinger et al, 2004), it would seem that none of them proposes a process exhibiting the three previously mentioned features for arbitrary networks. In this article, we describe a new neural network encoding scheme called MENNAG (Modular Encoding for Neural Network based on Attribute Grammars) that allows genetic operators to manipulate modules, repeat them, and hierarchically combine them. While we have focused our work primarily on the evolution of neural networks, the encoding could be easily adapted to evolve any graph structure. Modules are explicitly modeled in the genotype, noticeably because the emergence of modularity is a large debate (Bullinaria, 2007; Lipson et al, 2002; Kashtan and Alon, 2005) as it does not emerge in many evolutionary simulations (Bowers and Bullinaria, 2005).

As previously mentioned, many papers describe methods to evolve the topology of neural networks. Unfortunately, the most efficient of these systems involve a large and complex source code (often not available) as well as many implementation details neither not described nor justified in the papers. This complexity prevents researchers to easily re-implement other encodings to confirm the presented results. Moreover, it makes difficult the exploration and evaluation of variants of the encodings to distinguish its crucial features from the small refinements. In a few words, it is currently difficult to base future work on the literature because of the lack of a solid basis. Hussain (Hussain, 2003) made a step forward to the establishment of a formalism for neural network encodings by demonstrating how some published encodings can be formally expressed using attribute grammars (Knuth, 1968), a formalism designed to specify both the syntax and the semantics of programming languages. Following Hussain's proposition, the encoding we propose in this article is based on abstract syntax trees, similarly to (Gruau, 1995; Luke and Spector, 1996; Kodjabachian and Meyer, 1997), which are described and interpreted using an attribute grammar.

We assessed the performance of the proposed encoding by comparing results with three current methods from literature (Stanley and Miikkulainen, 2002; Doncieux and Meyer, 2004a; Wieland, 1991); on the classic cartpole problem (Wieland, 1991; Pasemann and Dieckmann, 1997; Stanley and Miikkulainen, 2002; Geva and Sitte, 1993; Doncieux and Meyer, 2005; Michel et al, 1997; Gomez et al, 2006; Igel, 2003), and on the control of a planar robotic arm, which requires repetition of the same structure. To our knowledge, only a few papers have attempted to benchmark the

available methods to evolve neural networks (Grönroos, 1999; Siddiqi, 1998; Gruau et al, 1996; Stanley and Miikkulainen, 2002; Igel, 2003).

This paper is organized in seven parts. First, we briefly review the literature concerning the evolution of neural networks. Next, we introduce the new encoding scheme followed by the associated operators. The fourth part describes the results of each of the compared methods on the cartpole problem. Then results concerning the robotic arm problem are presented. A short discussion and some ideas about future work xconclude this paper.

2 Evolving neural networks

As pointed out by Yao (Yao, 1999), evolution can be involved in the design of a neural network at three different levels: synaptic weights choice (or more generally parameters choice), topology design and learning rules selection. In this paper, we are primarily interested in methods that evolve both synaptic weights and neural network structure, without any restriction on the reachable topologies. Two kinds of encoding strategies have been explored (Kodjabachian and Meyer, 1995): direct and indirect encodings. With a direct encoding, there exists a bijection between genotype and phenotype so that each neuron and each connection appears independently in the genome. An indirect encoding is a more compact representation; for instance, using development rules. More recently a new kind of encoding has been proposed: implicit encodings. These encodings are based on an analogy with gene regulation networks. The genome is a sequence of characters where specific tokens delimit neurons' description Mattiussi and Floreano (2004, 2007). These encodings seem to facilitate the evolvability of neural networks Reisinger and Miikkulainen (2007), but their ability to generate modular networks has not been studied yet. Yao (Yao, 1999) and Cantu-Paz and Kamath (Cantu-Paz and Kamath, 2005) provide a large overview of the combination of neural networks and evolutionary algorithms in a broader context.

Many systems have been suggested to directly encode a neural network. Binary strings have been used to describe a connectivity matrix (Miller et al, 1989) or variable-length lists of neurons and connections (Cliff et al, 1992). Some other authors have used a sequence of parameters (Pasemann and Dieckmann, 1997). Mutation of these genomes can easily be defined in most cases, but cross-over is seldom used due to the complexity of combining graphs efficiently. NEAT (Stanley and Miikkulainen, 2002) offers an original solution by the introduction of innovation numbers, markers uniquely attributed when a new connection is created during the evolutionary process. These integers are employed to exchange similar connections - because historically derived from the same mutation and consequently supposed to share a common functionality - during the cross-over. Such numbers avoid the need to run complicated graph-matching algorithms. Moreover, these numbers can be used to compute a distance that is exploited to create ecological niches (Deb and Goldberg, 1989), aimed at preserving diversity. Although these direct encodings gave good results on simple problems, they cannot generate hierarchical regular and modular neural networks.

During biological development, the complex process that links DNA code to phenotype allows the reuse of the same gene in many contexts, and many pleiotropic effects. This compactness makes it possible to describe complex individuals with comparatively little information. Indirect encodings try to broadly copy this principle by using various generative representation schemes. The graph-generating grammar, introduced by Kitano (Kitano, 1990), used rewriting rules on the connectivity matrix. This research was pursued by Gruau who defined the cellular encoding (Gruau, 1995) in which a chromosome is a development program syntactically specified by a context-free grammar. Instructions in this language are neuron-centered: they divide neurons in different manners, like a mitosis, and create connections between them. The representation in the form of an abstract syntax tree (AST) allows us to employ the genetic operators defined for genetic programming (Koza, 1992), such as, for instance, sub-tree swapping. Drawing inspiration once again from genetic programming, Gruau added some primitive automatically defined functions (ADF). Nonetheless, as highlighted in (Luke and Spector, 1996), swapping two sub-trees in cellular encodings is not the equivalent of swapping two sub-networks in the phenotype, because the instructions are execution-order dependent. Moreover, Gruau does not provide any elegant way to

specify synaptic weights. This led Luke and Spector to define a similarly-inspired encoding, but edge-based instead of node-based. The same approach has been followed in (Hornby and Pollack, 2002), in which the authors explored body-brain evolution by combining L-Systems (Lindenmayer, 1968) and edge encoding. At the same time, SGOCE (Kodjabachian and Meyer, 1997) extended cellular encoding by situating cells in a geometric substrate and in two or three dimensions, thus enabling development instructions to take account of the relative positions of neurons in the substrate.

The concept of geometric substrate has also been exploited in (Cangelosi et al, 1994) where an axon growth process was proposed. The genotype is made up of a list of cells, each one of them containing parameters such as the length of the axon or the development angle. Vaario (Fukutani and Vaario, 1997) relied on another development scheme on a plane by explicitly taking into account the environment. In this model, each cell observes its neighborhood to execute one or several production rules. More recently, (D'Ambrosio and Stanley, 2007; Gauci and Stanley, 2007) exploited NEAT to evolve *Compositional Pattern Producing Networks* (CPNNs), a development abstraction able to represent repetitive patterns in a Cartesian space. These CPNNs are used to compute the connectivity and the synaptic weights between neurons situated on a plane.

The tree structure and the substrate concept are implicitly modular but, to our knowledge, no experiments have been carried out to evaluate the modularity of the networks obtained using the encodings described above. Repetition has been explored by Gruau (Gruau, 1995) using a predefined number of sub-structures. It seems that with these encodings, hierarchy has never been studied explicitly.

An explicit repetition mechanism could be applied in order to efficiently reuse a sub-network or to exchange it during the cross-over, as proposed in the ModNet encoding (Doncieux and Meyer, 2004b). A chromosome is made up of a list of model-modules, a list of modules and a list of links between those modules. The list of modules is constituted of model-modules, directly encoded, which can thus be used several times in the same network. Modules have only one input and one output. The cross-over is defined as an exchange of model-modules between chromosomes. Modular NEAT (Reisinger et al, 2004) constitutes another recent attempt to explicitly use modules. Modules are sub-networks encoded with NEAT that evolve symbiotically with a population of blueprints which specify how to combine them. Since modules cannot contain other modules, hierarchy is not present and, as a consequence, the complexity of modules is arbitrarily chosen (a few neurons in both cases). So far, ModNet has been successfully used to evolve controllers for cart-poles, lenticular blimps (Doncieux and Meyer, 2004b), and flapping-wing animats (Mouret et al, 2006). ModularNEAT has been tested on a simplified board game, demonstrating substantially better performance than NEAT.

The implementation complexity of these methods seems to have limited empirical comparisons and the number of problems on which they have been evaluated. (Grönroos, 1999) compared a direct encoding, the graph-generating grammar (Kitano, 1990), to the substrate-based encoding proposed in (Nolfi and Parisi, 1998). Working on the encoder problem and a classification task, the author's conclusions were similar to Kitano's: that the best performances on the encoder problem were achieved when using the graph grammar. However, he obtained the best results for the classification task with a direct encoding. Furthermore,(Siddiqi, 1998) concluded that a properly set up direct encoding can be as efficient as Kitano's encoding on the encoder. Cellular encoding has been compared to direct encoding with a fixed topology in relation to the cartpole problem (see section 4) and its variants. Both encodings led to working results but cellular encoding required less knowledge from the user. In (Stanley and Miikkulainen, 2002), the number of evaluations required by NEAT has been compared to the numerical results published in (Gruau, 1995). The authors concluded that NEAT required about 25 times fewer evaluations than cellular encoding to obtain similar results. Recently, Gomez et al. (Gomez et al, 2006) compared several encoding schemes for neural networks on variants of the cartpole problem. The best results were obtained with topology-fixed methods, then by NEAT, cellular encoding ranking last.

This lack of precise comparison stems from the difficulty to implement most neural network encodings. The reasons are at least twofold: the complexity of the algorithms, which often have many parameters and an incompletely described inner working, especially in short articles, and

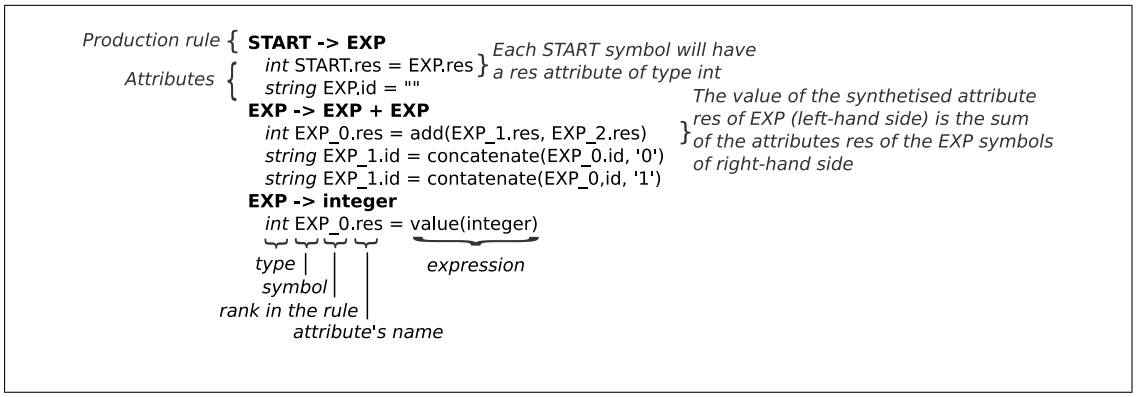


Fig. 1 Example of an attribute grammar which defines a language to add integers. The attributes *res* are synthesized, *id* are inherited. The result of the evaluation of programs is the value START.res. *id* attributes are only shown to illustrate inherited attributes behavior.

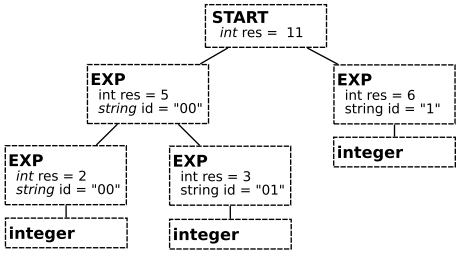


Fig. 2 Example of an abstract syntax tree for the expression "2 + 3 + 6", using the language defined by the attribute grammar of figure 1. Attributes are computed for each node, resulting in START.res = 11.

the difficulty of implementation. Both problems highlight the need for a common formalism that might lead to generic tools, making the implementation and testing of proposed algorithms more straightforward. The formalism introduced by Hussain (Hussain, 2003), based on attribute grammars (Knuth, 1968), may fulfill this need. This extension of context-free grammars adds the capability to formally specify the semantics of a language. Each symbol is linked to a set of attributes and each production rule to a set of evaluation rules of these attributes (figures 1 and 2). Attributes are divided into two distinct sets: *synthesized attributes*, which are the results of the update rules (and are computed from the terminals to the root symbol), and *inherited attributes*, whose value is issued from the parent node (and are consequently computed from the root to the terminals). Many algorithms have been developed to evaluate these attributes (Paakki, 1995).

Individuals generation and development involves the three following steps:

- If the individual has to be randomly generated (at the beginning of the evolutionary process), the grammar makes it possible to generate a syntactically correct development program (a tree); if the individual is the result of a cross-over, the grammar is used to choose the crossing point without breaking the syntax.
- Once the tree has been generated, the attributes are evaluated.
- The list of connections and the list of neurons are two synthesized attributes of the start symbol used to build the neural network.

The operators used in genetic programming can be used to evolve programs described by an attribute grammar.

Hussain showed that a large class of neural network encodings can be formally expressed with these grammars, noticeably cellular encoding and its variants. Nonetheless, some encodings such

as, for instance, NEAT, would probably be hard to express with such formalism because of their dependency on the evolutionary algorithm.

We implemented this in SFERES[1](Landau et al, 2002), a software framework dedicated to evolutionary robotics experiments.

3 MENNAG, a modular encoding for neural networks

Let us first list our objectives in creating a new encoding scheme:

1. to encode all topologies (closure property);
2. to be at least as efficient as the current encoding schemes, particularly on problems of evolutionary robotics;
3. to enable modularity, hierarchy and regularity in neural-networks, in the hope that these properties will improve evolvability and be able to cope with more complex problems than do current encodings;
4. to implement an efficient way of exchanging sub-networks during cross-over;
5. to be as formal as possible in order to allow easy re-implementation by other researchers and facilitate the exploration of variants.

ModNet and ModularNEAT enable modularity and repetition but are not formally specified and do not work with hierarchy of modules. Moreover, ModNet is limited to modules with only one input and one output. Cellular encoding and edge-encoding can be good inspiration sources, because they can be formalized using a context-free grammar, but their efficiency to allow repetition, modularity and regularity has not been demonstrated. The need to formalize and the possibility of exploiting tree-based genetic operators suggest using the framework described by Hussain. It is possible, with a correctly designed attribute-grammar, to ensure that exchanging a sub-tree corresponds to exchanging a sub-network.

Consequently, in this work we defined a new encoding scheme based on an attribute grammar that explicitly enables modularity, hierarchy and regularity. We called the new encoding MENNAG (Modular Encoding for Neural Networks based on Attribute Grammars). Moreover, we had to create a new mutation operator, less destructive than the one used by Hussain.

Figure 3 outlines the different steps from MENNAG's attribute grammar to a modular, repetitive and hierarchical neural network. The genetic operators and the random generation of individuals rely on the MENNAG grammar to create syntactically correct parse trees (genotypes). The attributes are then evaluated to create a list of neurons and a list of connections, which is subsequently translated to a neural network. The following section details how the designed grammar defines individuals with the desired properties.

3.1 The mechanisms of the grammar

In this section, we present the main mechanisms used in our attribute grammar. The complete grammar can be consulted in the appendix. As most of the inner workings of the encoding are described by the grammar (except the mutation and crossover operators), it appears quite complex at first glance, which is why we have chosen to present it step by step.

Neurons

The main structure of MENNAG's grammar relies on the non-terminals DIV and CELL, which respectively create a new module (DIV) and a new neuron (CELL), similarly to the Cellular Encoding. Each DIV level in the tree defines a module. These modules can be manipulated (exchanged, for instance) as independent units. An identifier, coded as a binary string, is given to each DIV

[1] http://sferes.lip6.fr

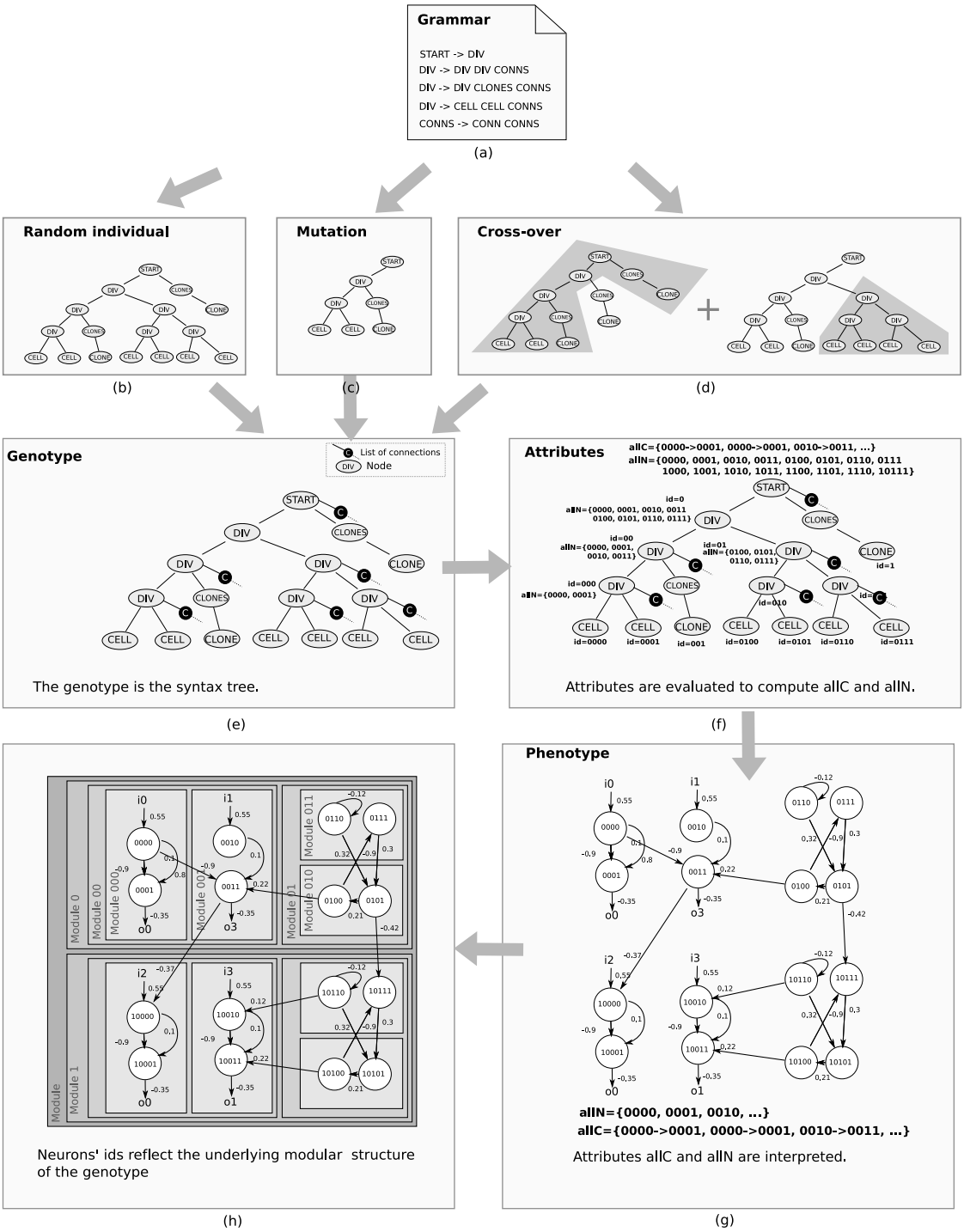
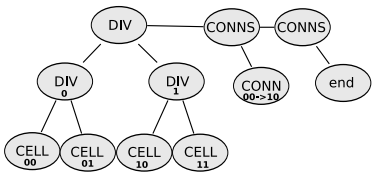


Fig. 3 The defined grammar (a) is used by genetic operators ((b),(c),(d)) to create new syntactically correct genotypes, defined as parse trees (e). Attributes are attached to each node of the tree, according to the grammar. The attributes START.allC (all connections) and START.allN (all neurons) are then evaluated (f). These attributes are easily transformed to a neural network (g) by scanning the two lists. Last, a modular view of the phenotype can be built using neurons' identifiers because they reflect the modular and hierarchical structure of the parse tree. Network (h) exhibits most of the presented features. Module 000 is repeated to define module 00. These modules are combined with two other modules to create the module 0. This module is itself repeated. A connection belonging to module 00 links a neuron from module 000 and one from module 001. Similarly, a connection created at the top level links neuron 0011 from module 0 to neuron 10000 from module 1. I/O are randomly swapped during each clone operation.

DIV ⇒ DIV DIV CONNS
string CONNS.divId = DIV.id
string DIV_1.id = concatenate(DIV_0.id, "0")
string DIV_2.id = concatenate(DIV_0.id, "1")
set DIV.allC = union(DIV_1.allC, DIV_2.allC, CONNS.allC)
set DIV.allNodes = union(DIV_1.allNodes, DIV_2.allNodes)
DIV ⇒ CELL CELL
set DIV.allNodes = union(CELL_0.node, CELL_1.node)
string CELL_0.id = concatenate(DIV.id, "0")
string CELL_1.id = concatenate(DIV.id, "1")
CONNS ⇒ CONN CONNS
set CONNS.allC = union(CONNS_1.allC, CONN.allC)
CONNS ⇒ end
set CONNS.allC = createset()
CONN ⇒ end

(a)



(b)

Fig. 4 (a) The main structure of the MENNAG grammar relies on a binary tree made of DIV instructions (modules) as node and of CELL instructions (neurons) as leaves. Functions used to update attributes are described in appendix C. The attributes of the CONN symbols are detailed in figure 5. (b) Example of a syntax tree which follows the grammar (a).

CONN ⇒ end
evofloat weight.value = evolved
bitstring targetNode.value = evolved
bitstring sourceNode.value = evolved
string CONN.targetNode = concatenate(CONN.targetNodeBaseId, target.value)
string CONN.sourceNode = concatenate(CONN.sourceNodeBaseId, node.value)
set CONN.allC = createset(createhash(name="sourceNode", value=CONN.sourceNode, name="targetNode", value=CONN.targetNode, name="weight", value=weight.value)

Fig. 5 Each connection links two neurons, each identified by a bitstring which can mutate. To ensure that the source and target neurons are in the same sub-tree as the list of connections, the identifier of the DIV is concatenated at the beginning of the identifier of each neuron. Weights are encoded as real numbers (*evofloat*), able to mutate.

and CELL as a function of the path within the tree linking it to the root node[2]. These identifiers are unique, and enable us to reference a module or a cell in order to create connections or duplicate a given module. The corresponding set of rules is depicted in figure 4. A sample syntax tree is shown in figure 4 (b).

Connections

Connections are made up of the identifiers of the neurons they connect and, contrary to Cellular Encoding, they have their own creation rules. The identifiers are represented as bitstrings, seen as constants by the attributes during the syntax tree interpretation but able to mutate during evolution, thanks to a bit-flip operator. To promote modularity, each DIV is associated with a list of connections that can only connect neurons created in the sub-trees of the DIV[3]. Synaptic weight

[2] For each DIV instruction, "0" is concatenated to its identifier to obtain that of its left son and "1" for its right one.

[3] To obtain this behavior, the identifier of the DIV is passed down to the CONN symbol. The DIV identifier is then concatenated to the evolved bit-strings representing source and target neurons, thus guaranteeing that these connected neurons belong to one the two sub-trees of the DIV.

is an attribute of each connection, mutated using Gaussian mutation. A connection is created using the rule in figure 5.

An overall direction can be given to connections to make networks more feed-forward. To that aim, it is easy to define rules that produce connections, for example, from the left sub-tree to the right. The rules are similar to the ones used for standard connections but "0" is added to the identifiers of the source neuron before concatenating it with the DIV's identifiers; and "1" to those of the target neuron. Thus, the source neuron is always in the left sub-tree and the target in the right, so creating an implicit direction in the computation flow from neurons created in the left part of the tree toward neurons created in the right. We have defined such a rule in our grammar, together with a simple rule without this constraint and tune selection probabilities (see section 3.2) so that these rules may control the aspect of the networks (see the complete grammar in appendix for more details).

Inputs/Outputs

We investigated two different ways to define input/output (I/O) connections: by creating a list of I/O attached to the root node, or by linking them to the leaves (i.e. to the neurons). In the first case, it is easy to ensure that each I/O is connected since the list is handled globally. But I/O don't belong to any module, and so cannot be exchanged during crossover with the sub-graphs to which they are linked. Moreover, this approach prevents them from being correctly handled when a module is duplicated by a clone, as we will see later. On the other hand, if the I/O are attached to the neurons, the total number of used I/O is not available locally, which prevents the control of how I/O are used in the network. One possible solution might be to choose each I/O to be connected from a stack[4]. This approach has often been used in genetic programming. Gruau, for instance, used it to manage connection weights. The problem is that a mutation or a crossover has many side effects, and may in particular imply changes in network parts not described by the modified sub-tree. In our case, this would mean that exchanging sub-trees would not correspond to exchanging sub-graphs. We then decided to link I/O connections to neurons, but without using a stack. The control of I/O connections is ensured by global constraints as we will see in section 3.2.

To define I/O connections, we introduced the following production rules:

CELL ⇒ IN OUT
IN ⇒ end
IN ⇒ IO in_1
...
IN ⇒ IO in_k
OUT ⇒ end
OUT ⇒ IO out_1
...
OUT ⇒ IO out_l

for a network with k inputs and l outputs, in_i and out_i refering to input number i and, resp. output number i.

CLONE

The last important part of MENNAG grammar deals with the non-terminal CLONE, designed to duplicate sub-networks. A CLONE copies a module (DIV), which can contain itself one or more CLONE. We add the rules:

[4] the first connection to input is connected to the first input, etc.

DIV ⇒ DIV CLONES CONNS
DIV ⇒ CLONES DIV CONNS
CLONES ⇒ CLONE CLONES
CLONES ⇒ CLONE
CLONE ⇒ end

The principle of the CLONE consists in copying the list of neurons and the list of connections of the DIV, while changing the indices.

Two main issues must be resolved if we are to write the attribute updates corresponding to these rules: to ascribe new identifiers to the cloned neurons, while preserving the structure of the sub-graph and handling the I/O connections contained in the cloned module. A solution to the first issue consists in concatenating "1" (or "0" if the CLONE appears before the DIV) before identifiers of neurons and before target and source neuron bitstrings. Connections and neurons, once cloned, can thus be added to the synthesized attributes allC and allN of the DIV from the left-hand side of the production rule. These cloned connections and neurons are handled in the same way as any "standard" (non-cloned) ones.

Handling I/O connections means finding them among the connections to be cloned and swapping them with "equivalent" ones. Imagine that the same treatment (for instance, computing a derivative) might be applied to a set of I/O pairs. For instance, input 1 (I_1) should be derived to compute output 1 (O_1), input 2 (I_2) to compute output 2 (O_2), and so on. If the evolutionary process managed to create a module able to compute the derivative of input 1, it ought to be able to duplicate this module and replace I_1 by I_2. To obtain this behavior, we must find a way to evolve a permutation of the I/O for each copy.

These permutations are encoded as a vector of real numbers. If k is the rank of a real number in the vector (unsorted) and i its rank in the same vector in ascending order, then k should be permuted with i. For instance, in the vector $[0.5; 0.1; 0.7]$, the rank of 0.1 is 1, the one of 0.5 is 2 and the one of 0.7 is 3. This vector could then be associated with $[2; 1; 3]$ which is translated as "I_1 is replaced by I_2, I_2 by I_1 and I_3 is not changed". This simple way to encode a permutation allows us to exploit the genetic operators designed for real numbers optimization (Deb, 2001). A vector attribute of evolved floats has thus been added to manage permutations of inputs and another one to manage permutation of outputs in the CLONE symbol (see detailed grammar in appendix).

The complete grammar is made up of the following production rules (see appendix for the attributes):

START ⇒ DIV CONNS
DIV ⇒ DIV CLONES CONNS
DIV ⇒ CLONES DIV CONNS
CLONES ⇒ CLONE CLONES
CLONES ⇒ CLONE
DIV ⇒ DIV DIV CONNS
DIV ⇒ CELL CELL CONNS
CELL ⇒ IN OUT
CLONE ⇒ end
CONNS ⇒ CONN CONNS
CONNS ⇒ end
CONN ⇒ ICONN
CONN ⇒ ICONN
ICONN ⇒ weight node target
IN ⇒ end
IN ⇒ IO in1
IN ⇒ IO in2
OUT ⇒ end
OUT ⇒ IO out1
OUT ⇒ IO out2

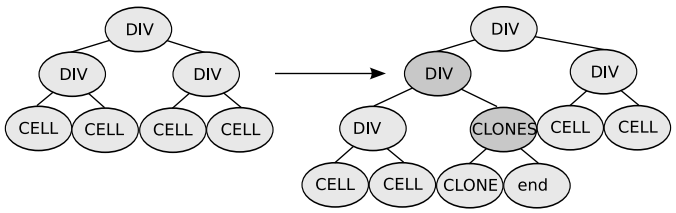


Fig. 6 Example of an insertion mutation.

IO ⇒ weight target

3.2 Genetic operators

Hussain (Hussain, 2003) and Gruau (Gruau, 1995) suggested the use of genetic programming operators (Koza, 1992). However, we found that substantial improvements could be obtained by slightly modifying the initialization process and the mutation operators.

Initial population A selection probability is assigned to each production rule, enclosed in brackets within the grammar: for instance: [0.1] DIV ⇒ DIV DIV. In order to randomly create individuals, the basic algorithm begins with a starting symbol (START in our grammar) and creates a list of the applicable rules, similarly to the traditional GROW algorithm used in genetic programming (Koza, 1992). The selection probability is then used to bias the choice of one production rule from the list of applicable rules. This process is run for each freshly created symbol until each syntax tree's branch ends with a terminal symbol.

Managing the properties of the trees created with this procedure is difficult. For instance, it can be very useful to create initial individuals with a specified number of neurons or with a particular mean number of connections. Furthermore, our local management of I/O connections makes it difficult to control how inputs and outputs are used. Though some theoretical analyses of the grammar and more powerful algorithms (García-Arnau et al, 2007) may provide more elegant solutions, we chose a simpler method in this work: trees are generated using the procedure described above and those that do not follow certain constraints are removed. In our work, we limited the number of neurons and the number of connections by forcing them to remain within a given range, (see appendix for typical chosen values). These constraints are fast to compute once the attributes have been evaluated. Noticeably, they do not require fitness evaluation.

Mutation Traditional genetic programming mutation relies on variants of the GROW algorithm applied to a sub-tree: a node is randomly chosen and the corresponding sub-tree is re-generated using the same algorithm as is used to create new individuals. This modification of the syntax tree is not translated into small changes of the neural-network. An ideal mutation should at least allow to add/remove a connection; and add/remove a module clone. On the tree side, the insertion of a module cloning symbol may be obtained by adding a DIV level using the rule DIV ⇒ DIV CLONES, for instance. In this example, let us add an identifier to each symbol to make clearer the correspondence before and after insertion. DIV_1 denotes the DIV we would like to clone. DIV_0, the DIV symbol on the left hand side, would be the father of DIV_1. A new DIV, DIV_3, would be inserted and DIV_1 would replace the DIV on the right hand side of the inserted rule. The CLONES sub-tree would be generated using the GROW-like algorithm (figure 6). Corresponding grammar-rules are:

Before insertion:

...
DIV$_0$ ⇒ **DIV**$_1$ **DIV**$_2$
...

After insertion:

...
DIV$_0$ ⇒ **DIV**$_3$ **DIV**$_2$
DIV$_3$ ⇒ **DIV**$_1$ **CLONES**
CLONES ⇒ **...**
...

By generalizing this idea, if an execution order-independent language is used then symbols can be safely added in the tree when there exists a recursive production rule to generate them. Such symbols include CONNS (CONNS ⇒ CONN CONNS) and DIV (DIV ⇒ DIV CLONES and DIV ⇒ DIV DIV).

A new mutation for the attribute grammar-specified genomes based on this insertion concept has been designed. To control its behavior precisely, two new numbers (i_1 and i_2) are enclosed in the brackets preceding production rules. They specify respectively the probability of selecting nodes generated by this rule as insertion points and the probability to extend the tree using this rule. Here is an example of a full description of a production rule and the associated selection probabilities: [0.01,i_1=0.5,i_2=0.5] DIV ⇒ DIV CLONES CONNS.

The corresponding mutation operator is implemented by executing the following procedure:

- randomly select an insertion rule r_1 used in the individual by biasing the choice using i_1;
- randomly select a recursive rule r_2 which will be used to extend the tree; the left hand side symbol of r_2 must belong to the right hand side symbols of r_1; the probability i_2 is used to bias the choice;
- among the nodes generated using r_1, randomly select a node n_1 with the same symbol as on the left hand side of r_2;
- create a new node n_2 with the same symbol as n_1;
- replace n_1 by n_2 in the list of sons of the father of n_1;
- use r_2 to randomly generate the sons of n_2;
- replace one of the sons of n_2 by n_1, in a compatible place.

A symmetrical mutation, which removes a node instead of adding one, can easily be derived from the same principles. This has been implemented in MENNAG. Moreover, the traditional genetic programming mutation can still be used with a low probability to create new sub-trees.

4 Cart-pole experiment

This encoding was first benchmarked on the classical cart-pole problem (sometimes referred as pole balancing or inverted pendulum). This task has been chosen to check that MENNAG can solve a simple non-modular control task at least as efficiently as other encodings.

A pole, attached to a cart by a hinge, has to be balanced by moving the cart (figure 7). The goal is to maintain the pole in a vertical position while keeping the cart at the center of the track. This problem has been used to evaluate numerous encoding schemes for neuro-controllers (Wieland, 1991; Pasemann and Dieckmann, 1997; Stanley and Miikkulainen, 2002; Geva and Sitte, 1993; Doncieux and Meyer, 2005; Gomez et al, 2006; Igel, 2003) because it is representative of a wide class of control problems involving unstable systems. Contrary to some previous work, we

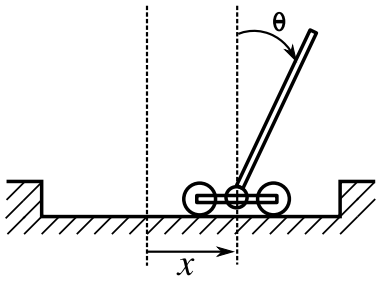


Fig. 7 Cart-pole problem. The pole has to be maintained in a vertical position while keeping the cart as close as possible to the center of the track.

don't consider the cart-pole problem as solved if the pole doesn't fall but only if the controller is able to keep the pole vertically without oscillating, as a classic proportionnal-derivative (PD) controller does. This prevents this task to be solved with a single neuron (Widrow, 1987) and makes it more challenging. For instance, the path between the input and the outputs should be as short as possible to react as fast as possible.

If the angle and the position of the cart are the only inputs, the controllers have to derive the input signals to avoid oscillating behaviors. Automatically discovering a neural-network able to perform such computation is the main challenge of this problem. In the following experiments, we did not provide the networks with velocities: pole angle and cart positions were the only inputs. The networks have a single output, which controls cart acceleration[5]. Although modular properties are not mandatory to solve the problem, it has been shown that they may help by allowing the propagation of a derivative module (Doncieux and Meyer, 2004b).

Equations enabling the simulation of an inverted pendulum are widely available (e.g., (Wieland, 1991; Pasemann and Dieckmann, 1997; Doncieux, 2003)).

These experiments were carried out with the fitness described in (Doncieux and Meyer, 2004b). Let us first define the mean normalized error for the angle θ and the position x during the duration T:

$$e_\theta(g) = \frac{1}{T}\sum_{t=1}^{T} e_\theta(t,g)$$

$$e_x(g) = \frac{1}{T}\sum_{t=1}^{T} e_x(t,g)$$

where $e_\theta(t,g)$ and $e_x(t,g)$ denote the normalized errors at step t.

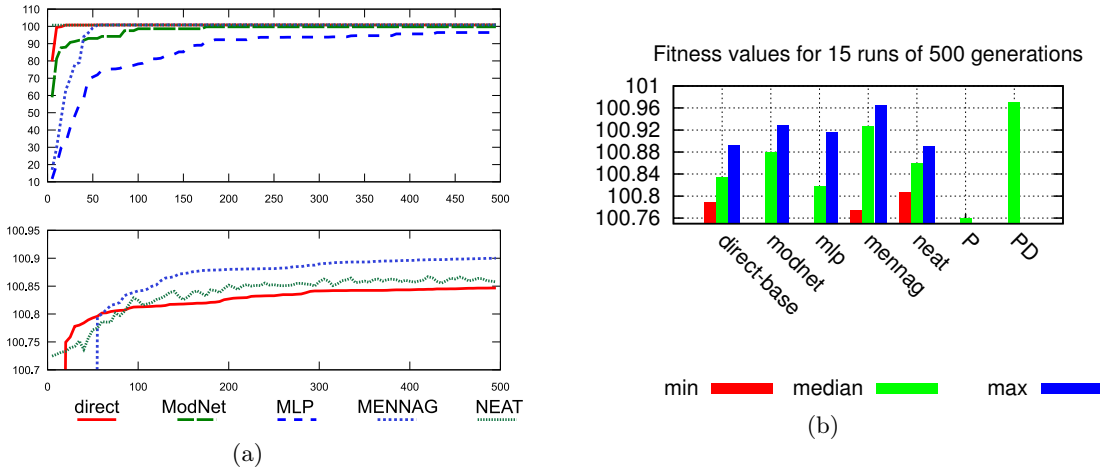
The fitness is the sum of two terms, a decimal and an integer:

$$f(g) = p(g) + \frac{1}{2}((1 - e_\theta(g)) + (1 - e_x(g)))$$

where $p(g)$ denotes the percentage of evaluation time the pendulum spent before going beyond the boundaries (± 0.2 rad and ± 2 m). This fitness, which varies between 0 and 101, ensures that controllers that maintain the cart-pole within the boundaries have a better fitness than the others, whatever the sum of errors may be.

Many encodings have been tested on the double cartpole problem (Wieland, 1991; Stanley and Miikkulainen, 2002), in which two poles are attached to the same cart. We did not benchmark MENNAG using this task because bootstraping the evolutionary algorithm is especially difficult, making the task as much a problem of random generation and selection pressure than a problem of encoding. Moreover, our first tests with published encodings have shown that in many cases the evolutionary process was not able to fully solve the simple cart-pole task, being only able to prevent the pole from falling.

[5] Neuron output belongs to the $[-1, 1]$ interval and is maped to a force ranging between -10N and 10N.



(a) (b)

Fig. 8 (a) Mean fitness over 15 runs for the different encodings. (b)Min, median and max fitness values over 15 runs of 500 generations, for each encoding. No initial module was given to ModNet. P reference controller prevents the pole from falling, but cannot reduce the oscillations in the pole angle or in the cart position. PD reference controller quickly centers the cart with the pole in an upward position. All differences are statistically significant ($p < 0.05$, Wilcoxon-Mann-Whitney test ; see appendix A), except between MENNAG and ModNet ($p = 0.074$).

Results We ran a set of 15 experiments for each encoding using the fitness previously described, a population of 100 individuals and a standard steady-state evolutionary algorithm. Figure 8 shows the obtained fitness with the MENNAG encoding, ModNet, a direct encoding similar to (Pasemann and Dieckmann, 1997) and NEAT, using the original source code, (see appendix D for detailed parameters used in this comparison. For reference, we added the fitness obtained using classic P and PD controllers tuned using a basic evolutionary search and those obtained with a simple multi-layer perceptron with one hidden layer (3 hidden neurons, 18 weights). An implementation of the cellular encoding based on the work of Hussain (Hussain, 2003) was set up but we did not manage to get solutions with an efficiency at least similar to published results.

For all the encodings, the synaptic weights were encoded by a real number with Gaussian mutation.

Results are plotted on figure 8. NEAT obtains working controllers, with a fitness greater than 100, at the first generation. This is not surprising since it starts with a topology capable of emulating a P controller. While some MENNAG runs get good results in a few generations, about 60 generations are needed to obtain a fitness greater than 100 for all the runs. The most surprising result is the low performance of the multi-layer perceptron (MLP) : despite the smaller search space, more generations are needed to achieve working controllers. This highlights an interesting case where evolving a topology is more efficient than using a fixed one. One of the main reasons may lie in the incremental path followed by evolution while evolving topologies. As a simple proportional link is sufficient to prevent the pole from falling, the individuals of the first generations use very simple networks with a few weights to tune. Evolution then improves behavior by adding neurons and connections. The multi-layer perceptron, on the contrary, starts with a complex and poorly-adapted topology with 18 weights to optimize simultaneously.

The results obtained with NEAT may be surprising given previously published results (Stanley and Miikkulainen, 2002). Although we tried to find the best parameters, we cannot claim that it would be impossible to improve the efficiency of NEAT on the investigated problem by using a different set of parameters. In (Stanley and Miikkulainen, 2002), the authors investigated the cart-pole problem and their findings suggested than NEAT was very efficient in solving it. However, they did not focus their attention on fine control; they concentrated on the cartpole remaining in the boundaries and consequently “solved” the problem. In our benchmarks, NEAT led to working controllers in all runs, confirming the published results, but the more finely observed behavior was

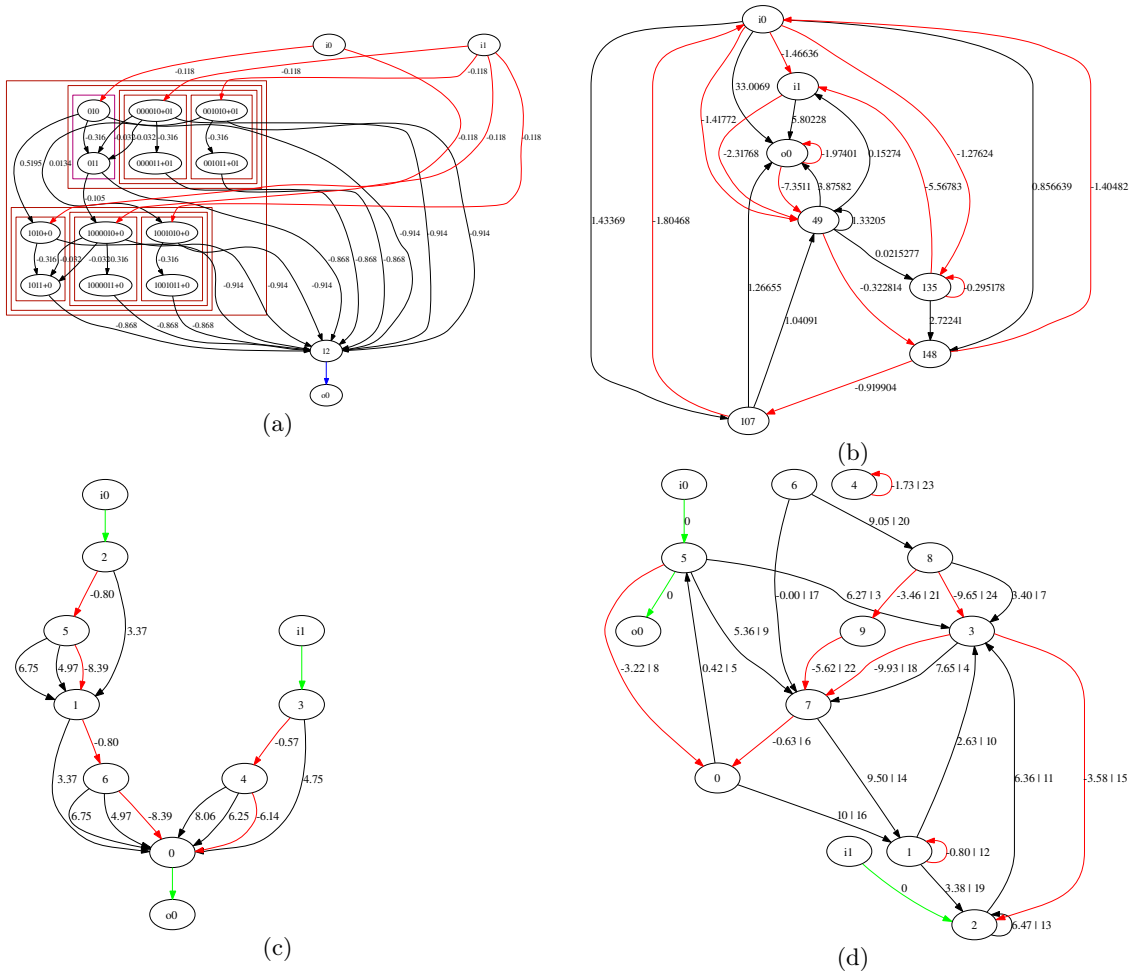


Fig. 9 Best neural networks after 1500 generations with the compared encodings, for the cart-pole problem. i_0 is the angle input and i_1 the position input. o_0 is the network output. (a) MENNAG (fitness: 100.967); (b) NEAT (fitness:100.89); (c) ModNet (fitness: 100.93); (d) Direct encoding (fitness: 100.899).

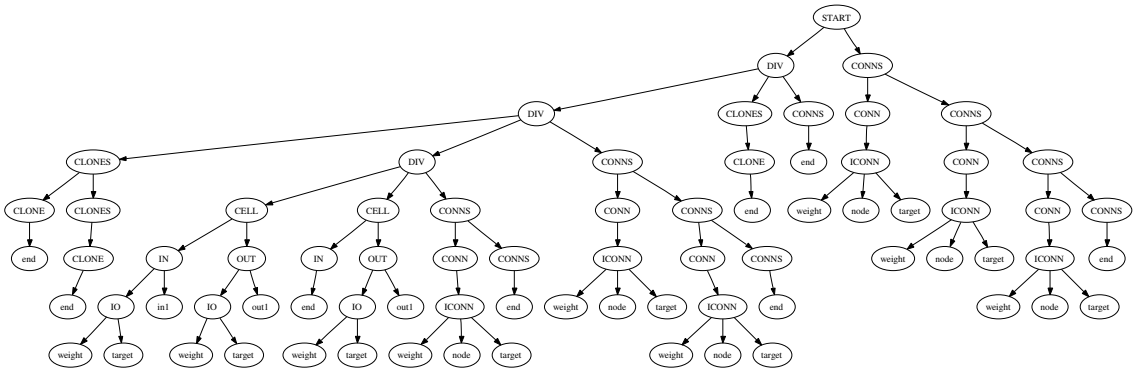


Fig. 10 Syntax tree that generates the neural-network of figure 9(a).

Fig. 11 Typical behaviors of the position x of the cart and of the value of ϑ for a 100.858 fitness (a) and a 100.965 fitness (b). At $t = 0$, the poles starts from $\vartheta = 0.01$ rad and $x = 0.5$ m. The controller has to drive the pole to $\vartheta = 0$ rad and $x = 0$ m. Controller (a) avoids the pole from falling during this evaluation, but the increasing amplitude suggests an imminent fall.

not as good as that obtained with a PD controller or MENNAG. We tried to use more generations but it did not improve the results.

All the encodings led to better fitness than the simple P controller but only MENNAG managed to produce at least one neural network as good as a PD controller, which prevents the pole from oscillating. Overall, MENNAG led to significantly better results than the other methods. ModNet found some good controllers but certain runs failed even to generate a good P corrector.

Figure 9 depicts the best neural network obtained with each encoding. The genotype corresponding to the neural network obtained with MENNAG is seen in figure 10. Although this structure exploits the modular and hierarchical features of MENNAG, it has not created "derivative" modules and cloned them from one input to the other. Instead, it built the whole structure by cloning a simple module six times and by adding connections between the cloned modules.

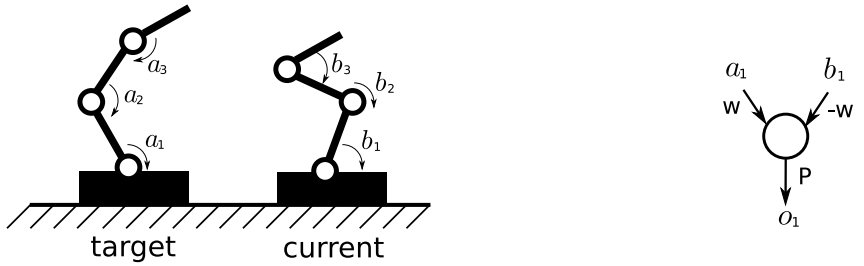
MENNAG was able to generate several neuro-controllers as efficient as a PD-controller and more than half of the runs lead to better results than the best individuals obtained using the other encodings. Although the difference in fitness between the different solutions is small (a few hundredths), the qualitative difference on the behavior is not negligible: 100.858 corresponds to an oscillating behavior with an increasing amplitude, while 100.965 corresponds to a fast control without any oscillations at all (figure 11). At any rate, the reasons for this difference are unclear and require further study. Although we tried to tune the different encodings to obtain the best possible results, it is possible that some slight modifications reduce the gap between the solutions. The best neural networks generated with MENNAG use modularity, hierarchy and regularity, but it remains to be proved whether these features are effectively critical for this application, or whether the good performances can be explained by other factors, like connections management, for instance.

We will now focus on another experiment in which the modular aspect of MENNAG should be even more decisive and more easily observed.

5 Robotic arm experiment

One of the main features of MENNAG is its ability to exploit the same sub-structure several times to solve similar sub-tasks. To evaluate the efficiency of this behavior, we designed a toy-problem based on the control of a simple robotic arm with three degrees of freedom (figure 12 (a)) and simulated with the widely used dynamics library ODE[6]. The evolved neural networks have to drive each motor to a target position $T = (a_1, a_2, a_3)$ while knowing only T and its current position $P = (b_1, b_2, b_3)$. The motivation in the choice of this problem is twofold. First, the benefit of a repetitive encoding is obvious given the similarity between the degrees of freedoms. MENNAG should easily find good solutions whereas direct encodings are expected to be less efficient. Second,

[6] http://www.ode.org



(a) (b)

Fig. 12 (a) Robotic arm with three degrees of freedom. The controllers have to move each hinge to reach the target position as fast as possible, without oscillation. The robotic arm is simulated using a realistic dynamic simulator. (b) Example of a typical P controller for one degree of freedom.

this problem is a simplified instance of many robotics tasks in which several degrees of freedom have to be controlled using similar strategies. Legged robots, in which each leg looks like the other ones, and industrial robot arms are simple examples.

The robotic arm problem can easily be solved by computing the difference between a_i and b_i ($i \in [1;3]$) to obtain the angular position error. This error can then be multiplied by a proportional factor to create a basic proportional controller (figure 12 (b)). The same sub-network can be used to control each degree of freedom, making the problem completely decomposable. It should be noted that a network combining three times the structure of 12 (b) should efficiently solve the problem. Such a network is a sparsely connected multi-layer perceptron and, consequently, a multi-layer perceptron should be able to control the arm.

This task is surprisingly hard for evolved neuro-controllers. For each degree of freedom, the algorithm has to:

- choose the right pair of inputs to connect to the right output;
- choose exactly opposite weights to compute a difference;
- find a proportional coefficient.

Most evolutionary algorithms for neural networks could easily find such a solution for one degree of freedom. However, if they are unable to repeat the structure previously found, they would have to find it three times over in the same network. The ability to manipulate modules in MENNAG should help to discover the 'difference' structure once and go on to clone it twice.

The chosen fitness is the normalized accumulation of errors between a_i and b_i, additioned for each degree of freedom:

$$F(g) = -\frac{1}{3T}\sum_{i=1}^{i=3}\sum_{t=0}^{t=T}|a_i(t) - b_i(t)|$$

where $i = 1, 2, 3$ and T denotes the number of time-steps. To ensure that the obtained controllers are able to generalize, five different sets of target position are used during each evaluation procedure.

Results We ran a set of 15 experiments using the fitness described above, a population of 200 individuals and a standard steady-state evolutionary algorithm. 250 generations and the same parameters were used as in the former experiment, except concerning the number of I/Os.

Results are shown on figure 13. Among the investigated encodings, only MENNAG managed to reach the performance of a simple controller where each degree of freedom is connected to a P corrector. Moreover, the median values for the 15 runs show that these performances are obtained most of the time. The direct encoding and ModNet found 2-DOFs controllers while NEAT found no good P controller.

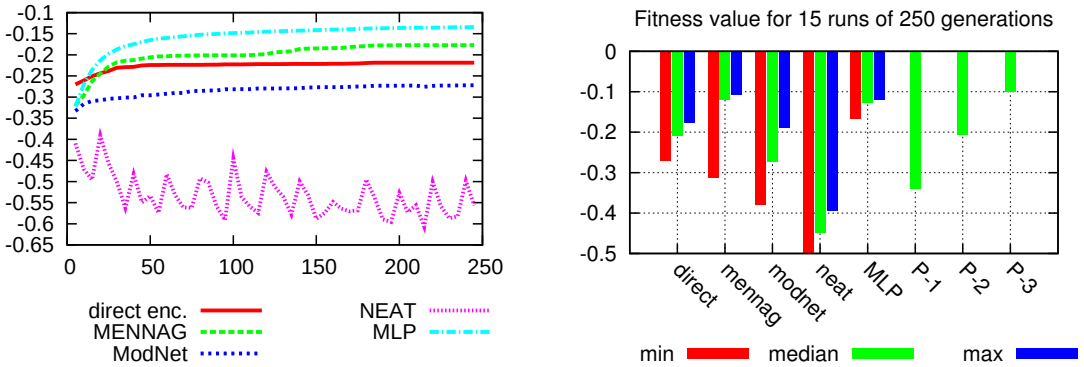


Fig. 13 (a) Mean fitness over 15 runs for the robotic arm experiment and the different encodings. (b) Fitness obtained after 250 generations. P-1 is the fitness obtained using one P controller, P-2 using two P controllers, P-3 using three P-controllers. All differences are statistically significant except between MENNAG and the MLP ($p \leq 0.02$, Wilcoxon Mann-Whitney test; see appendix A.2).

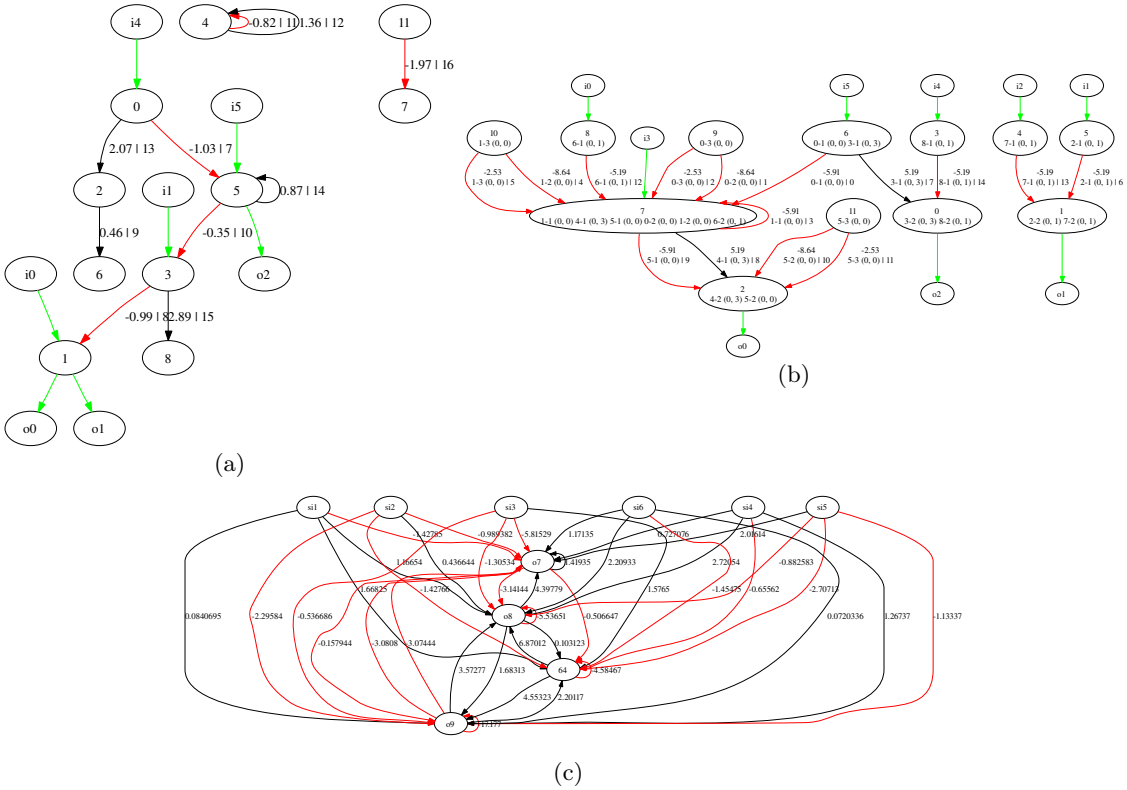


Fig. 14 Neural networks obtained with the different encoding on the robotic arm problem. (a) Direct encoding (fitness: −0.161155); (b) ModNet (fitness: −0.188119); (c) NEAT (fitness: −0.395931); .

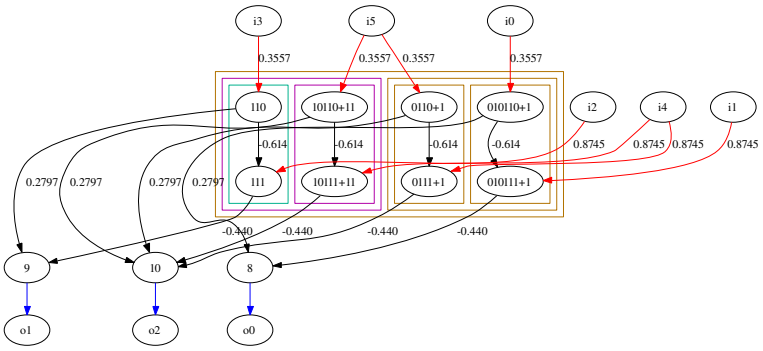


Fig. 15 Example of a 3-DOFs controller evolved using MENNAG with a fitness of −0.102298. Identifiers with a "+" symbols are used to track cloned modules. For instance, the identifier "10110+11" means that this neuron is part of a cloned version of the module "11". In this network, a P-like corrector connecting i2, i3 and o1 (module 11x, on the left side) has been cloned to link i5, i4 and o3 (module 1011x). These two modules have then been duplicated at level 1 to obtain the right side of the neural network.

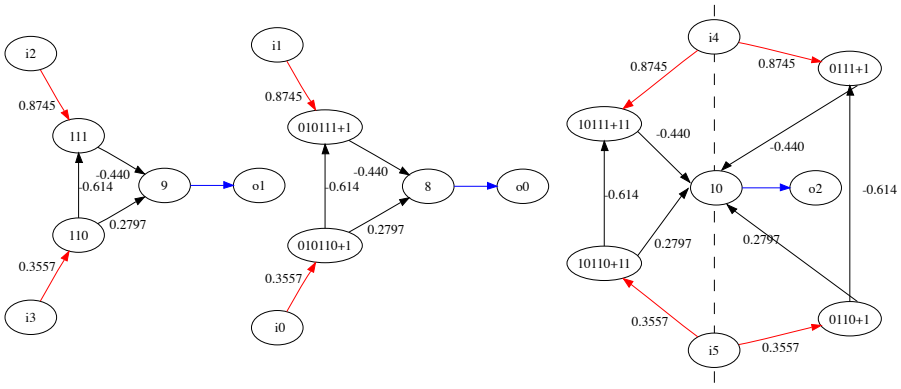


Fig. 16 Re-arrangement of the neural network depicted on figure 15 to obtain a better view of the repeated sub-structures. The right neural network has a vertical axis of symmetry. The two left networks differ only by their I/Os.

Some preliminary tests have been conducted with more degrees of freedom to understand how the proposed method scales up. While some good controllers have been found with four DOFs using MENNAG, the performance quickly decreases with more DOFs because of the combinatorial problem of selecting the good I/Os for each module. More details about this topic are available in the discussion section.

The evolution of the weights of a simple multi-layer perceptron with one hidden layer gave very good results and is given as a reference. While this performance may be surprising, it must be emphasized that the search space is much smaller when topologies are not explored. A brief analysis of the robotic arm easily led us to conclude that it should be efficiently controllable by a feed-forward neural network. However, by fixing the topology, we added a lot of information that was not available to the methods that were seeking efficient topologies.

Figure 14 is a sample of the best neural networks obtained with each encoding.

Figures 15, 17 and 16 detail one of the best controllers, obtained with MENNAG. A P-like structure has been found then repeated once using a CLONE instruction, demonstrating the usefulness of MENNAG's repetition instructions. This module is then duplicated using another CLONE instruction, at a higher level. In consequence, the same sub-structure is repeated four times (figure 16). Most successful networks observed did not succeed in obtaining precisely three

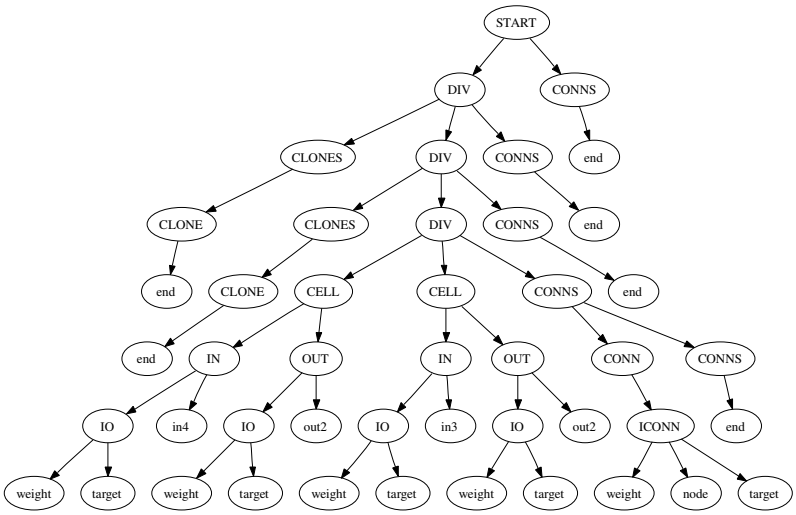


Fig. 17 Syntax tree (genome) that generates the neural network of figure 15 and 16.

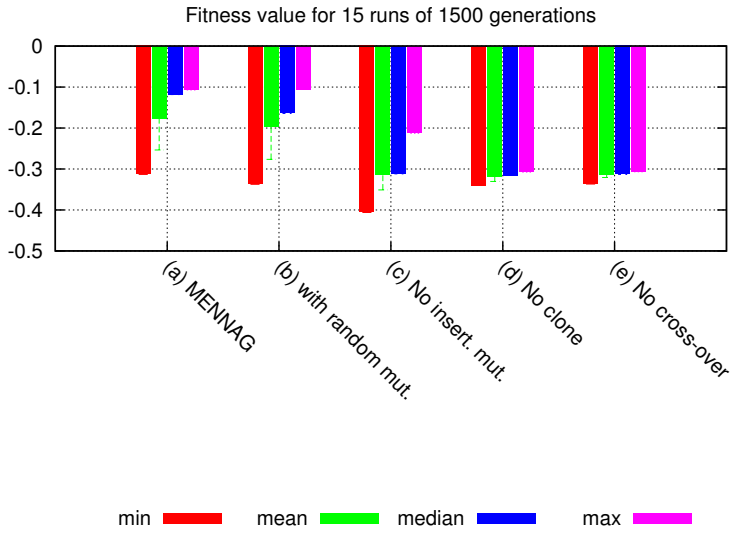


Fig. 18 Fitness obtained for the robotic arm experiment with different setups. (a) MENNAG; (b) MENNAG without the "random tree" mutation (c) MENNAG without the new insertion mutation; (d) MENNAG without the CLONE instruction; (e) MENNAG without cross-over. Differences between (a) and (b) are not statistically significant (Wilcoxon Mann-Whitney test; see appendix **??**) and differences between MENNAG and the other tests are significant ($p < 10^{-4}$).

instances of a P module but relied on a hierarchy of clones to obtain four instances. This behavior could possibly be improved by a finer tuning of the selection and insertion probabilities, but we opted to keep the same parameters for the two problems.

Lesion experiments In order to evaluate the contribution of each component of the proposed system to the overall performance, we performed robotic arm experiments while successively removing each key part of MENNAG: CLONE instruction, insertion mutation and cross-over. Moreover, we

tried to add the "random tree" mutation often used in genetic programming. This approach can be assimilated to the lesion experiments executed by biologists to understand living systems.

Results are plotted on figure 18. Best fitnesses were obtained with all the operators enabled excepting the random tree mutation. This mutation was used with a low probability, so we did not expect it to have a strong influence on the results. Nevertheless, it does not seem to have improved them. Designed to maintain a diversity in the population, this operator may still be useful in other problems. The three degrees of freedom cannot be controlled if the tree's main components – insert mutation, CLONE and cross-over – are not used. If we consider the structure of the problem under investigation, the need for a CLONE operation seems obvious. The insertion mutation operator allows us to clone a module that has already been optimized, or at least efficiently tuned by evolution. CLONE and this operator are then complementary and reach their full potential when used simultaneously. Contrary to some other encodings that don't rely on cross-over, this operator proved to be useful in the proposed encoding. To know whether functional modules were exchanged, or whether cross-over was only used as a basic exploration method would require further investigations.

6 Discussion

This paper aims at two main contributions: proposing an efficient modular, regular and hierarchical encoding for neural network and expressing it using an appropriate and flexible formalism. Both ideas lead to good results but underline some difficulties.

MENNAG performed better than the comparable encodings we tested on the two simple tasks we investigated. More benchmarks are needed to gain a better view of what makes MENNAG more efficient. Modularity and the repetition features of MENNAG were demonstrated. Hierarchy was enabled in the presented experiments, although it is unclear whether it improved the evolvability of the neural networks. As pointed out in (Watson, 2005), designing a toy problem that would be both modular, hierarchical and repetitive is a difficult task which requires a deep understanding of modularity in complex systems. Moreover, some other kinds of modularity not easily available in MENNAG, such as repetition with variation and symmetry (Stanley, 2006), may be required to evolve complex systems. The evolutionary robotics field could provide some interesting tasks as many of the envisioned situations imply showing different behaviors that intuitively require at least functional modularity in order to build up systems of increasing complexity while exploiting previously generated modules.

The proposed encoding used the attribute grammar formalism, proposed in the context of neural network evolution by Hussain (Hussain, 2003). This formalism interacts nicely with tree-based genotype and consequently with modular evolution, since trees are almost naturally modular and hierarchical. By choosing to use attribute grammars, our main goal was to ground future works on a solid basis by enabling an easy implementation (and future re-implementations) and to quickly explore a wide range of variants, for instance to benchmark different I/O handling strategies. Furthermore, the genetic operators are completely decoupled from the neural network problems, making possible to study them separately on simpler setups. The development of MENNAG proved us that experimenting with different alternatives was possible by only slightly altering the grammar, that is compact enough to be easily manipulated. This leaded to a faster development of the encoding. Moreover, as an illustration, we recently tried to re-implement MENNAG in a new attribute grammar system. Once the generic attribute grammar evaluator and the genetic operators were set-up – both of them being not specific to neural networks –, implementing and testing MENNAG was only a matter of hours. As a conclusion, we expect to be able to conduct many future work with a minimal implementation effort and a rigorous framework.

Nevertheless, this work brings to light some limitations of the attribute grammar paradigm for neural network encoding which should be pointed out. First, tree-based genotypes are easily expressed but many other genotypes could probably be used and could interact less nicely with attribute grammars. For instance, NEAT is very hard to express in this formalism due to its dependency to a niching algorithm and its custom cross-over operator. Another significant drawback

is the computational cost of the attribute evaluation process. Noticeably, it can substantially slow down the evolutionary process when big sets are copied at each node of the trees. Nonetheless, quickly evaluating an attribute grammar is a well studied process and it should be possible to implement faster systems than our prototypes.

7 Future work

An important point of this work concerns its possible generalization. Thanks to the formalization, it is easy to evolve other structures than neural networks. To use MENNAG on other weighted directed graphs, one only has to change the interpretation of the allNodes and allConns lists. Moreover, many variants, for instance to evolve undirected graphs, can be designed by slightly modifying the presented grammar. Such uses will be studied in future work as well as the test of MENNAG on different neural-network problems.

The main technical issue revealed by the obtained results is the difficulty of correctly connecting I/Os when the dimension of the problem rises. In the robotic arm problem, there exists $6! \times 3! = 4320$ different ways to link these I/O to a same, potentially optimal, structure. If one degree of freedom is added, the solution has to be found among more than one million different combinations. In consequence the main difficulty soon appears to lie in a good I/O choice rather than in exploring efficient modular structures. The proposed encoding has been designed to explore structures, so it will not be efficient enough to handle control problems with a high number of I/O. As a result, although modularity, hierarchy and repetition can lower the complexity of the search in cases of a high number of I/Os, for instance by using the same sub-network several times, they are not sufficient to handle an arbitrary number of I/Os.

Two approaches seem conceivable to cope with this central scalability issue. It has been suggested by some authors (Kodjabachian and Meyer, 1997; Cangelosi et al, 1994; Fukutani and Vaario, 1997; D'Ambrosio and Stanley, 2007) that using a geometric substrate to localize neurons can reduce the difficulty of the search by spatially gathering I/O that should work together. By defining the localization of neurons, users add knowledge and, consequently, face a trade-off between user-constraints and exploration. If users have almost no a priori information about the structure of the solutions, this substrate approach could be very difficult to use. Another approach could be to use development to link network building to I/O creation. This would imply evolving both morphology and control, as in (Sims, 1994).

8 Conclusion

In this work, we have presented MENNAG, a new encoding scheme designed to evolve neural-networks while exploiting modularity, hierarchy and repetition. It has been formalized by using an attribute grammar whose mechanisms have been described in detail. Compared to previous work on modular encodings, such as ModNet or ModularNEAT, MENNAG adds essentially hierarchy and formalization.

The performance of the new encoding has been compared to a direct encoding, NEAT, ModNet and a multi-layer perceptron in two simple control problems. MENNAG led to better fitness than the other encodings in both cases. These encouraging results call for further studies of this new encoding, and further evaluations on new problems. Modules hierarchy, for instance, ought to be more extensively tested.

References

Anderson JR (2007) How Can the Human Mind Occur in the Physical Universe?, Oxford University Press, chap The Modular Organization of the Mind, pp 45–91

de Beeck H, Haushofer J, Kanwisher N, et al (2008) Interpreting fMRI data: maps, modules and dimensions. Nature Reviews Neuroscience 9:123–135

Boers JW E, Kuiper H, Happel BLM, Springhuizen-Kuiper IG (1993) Designing modular artificial neural networks. Tech. rep.

Bowers C, Bullinaria J (2005) Embryological modelling of the evolution of neural architecture. In: A Cangelosi GB, Borisyuk R (eds) Modeling Language, Cognition and Action, World Scientific, Singapore, pp 375–284

Buessler JL, Urban JP (2002) Biologically Inspired Robot Behavior Engineering, Springer Verlag, chap Modular Neural Architectures for Robotics

Bullinaria J (2007) Understanding the Emergence of Modularity in Neural Systems. Cognitive Science 31(4):673–69,523

Cangelosi A, Parisi D, Nolfi S (1994) Cell division and migration in a 'genotype' for neural networks. Network: Computation in Neural Systems 5(4):497–515

Cantu-Paz E, Kamath C (2005) An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. Systems, Man and Cybernetics, Part B, IEEE Transactions on 35(5):915–927

Cliff D, Harvey I, Husbands P (1992) Incremental evolution of neural network architectures for adaptive behaviour. Tech. Rep. Cognitive Science Research Paper CSRP256, Brighton BN1 9QH, England, UK

D'Ambrosio D, Stanley K (2007) A novel generative encoding for exploiting neural network sensor and output geometry. Proceedings of the 9th annual conference on Genetic and evolutionary computation pp 974–981

Deb K (2001) Multi-objectives optimization using evolutionnary algorithms. Wiley

Deb K, Goldberg DE (1989) An investigation of niche and species formation in genetic function optimization. In: Proceedings of the third international conference on Genetic algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 42–50

Doncieux S (2003) Évolution de contrôleurs neuronaux pour animats volants : méthodologie et applications. PhD thesis, LIP6/AnimatLab, Université Pierre et Marie Curie, Paris, France

Doncieux S, Meyer JA (2004a) Evolution of neurocontrollers for complex systems: alternatives to the incremental approach. In: Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004)

Doncieux S, Meyer JA (2004b) Evolving modular neural networks to solve challenging control problems. In: Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)

Doncieux S, Meyer JA (2005) Evolving PID-like neurocontrollers for non-linear control problems. International Journal of Control and Intelligent Systems (IJCIS) Special Issue on nonlinear adaptive PID control 33(1):55–62

Fukutani I, Vaario J (1997) The effect of environment to genetic growth. In: International Symposium on System Life, July 21-22, 1997, Tokyo, Japan, pp 227–232

Gallinari P (1998) Modular neural net systems, training of. The handbook of brain theory and neural networks table of contents pp 582–585

García-Arnau M, Manrique D, Ríos J, Rodríguez-Patón A (2007) Initialization method for grammar-guided genetic programming. Knowledge-Based Systems 20(2):127–133

Gauci J, Stanley K (2007) Generating large-scale neural networks through discovering geometric regularities. Proceedings of the 9th annual conference on Genetic and evolutionary computation pp 997–1004

Geva S, Sitte J (1993) A cartpole experiment benchmark for trainable controllers. Control Systems Magazine, IEEE 13(5):40–51

Gomez FJ, Schmidhuber J, Miikkulainen R (2006) Efficient non-linear control through neuroevolution. In: Proceedings of the European Conference on Machine Learning (ECML-06, Berlin), pp 654–662

Grönroos M (1999) A Comparison of Some Methods for Evolving Neural Networks. Proceedings of the Genetic and Evolutionary Computation Conference 2:1442

Gruau F (1995) Automatic definition of modular neural networks. Adaptive Behaviour 3(2):151–183

Gruau F, Whitley D, Pyeatt L (1996) A comparison between cellular encoding and direct encoding for genetic neural networks. In: John Koza R, David Goldberg E, David Fogel B, Rick Riolo L (eds) Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press, Stanford University, CA, USA, pp 81–89

Hartwell L, Hopfield J, Leibler S, Murray A (1999) From molecular to modular cell biology. Nature 402(6761):C47–C52

Holland JH (1975) Adaptation in Natural and Artificial Systems. MI: University of Michigan Press

Hornby G (2005) Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. Proceedings of the 2005 conference on Genetic and evolutionary computation pp 1729–1736

Hornby G, Pollack J (2002) Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8(3):223–246

Huettel S, Song A, McCarthy G (2004) Functional magnetic resonance imaging. Sinauer Associates Sunderland, Mass

Hussain T (2003) Attribute Grammar Encoding of the Structure and Behaviour of Artificial Neural Networks. PhD thesis, Queen's University

Igel C (2003) Neuroevolution for reinforcement learning using evolution strategies. In: The 2003 Congress on Evolutionary Computation, CEC'03., IEEE Press, vol 4, pp 2588–2595

de Jong E, Thierens D (2004) Exploiting modularity, hierarchy, and repetition in variable-length problems. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04 pp 1030–1041

Kashtan N, Alon U (2005) Spontaneous evolution of modularity and network motifs. Proceedings of the National Academy of Sciences 102(39):13,773–13,778

Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Systems 4:461–476

Knuth D (1968) Semantics of context-free languages. Theory of Computing Systems 2(2):127–145

Kodjabachian J, Meyer JA (1995) Evolution and development of control architectures in animats. Robotics and Autonomous Systems 16:161–182

Kodjabachian J, Meyer JA (1997) Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. IEEE Transactions on Neural Networks 9:796–812

Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press

Landau S, Doncieux S, Drogoul A, Meyer JA (2002) SFERES: un framework pour la conception de systèmes multi-agents adaptatifs. Technique et Science Informatiques 21(4):427–446

Lindenmayer A (1968) Mathematical models for cellular interaction in development, parts i and ii. Journal of theoretical biology 18(18):280–315

Lipson H (2004) Principles of Modularity, Regularity, and Hierarchy for Scalable Systems. Genetic and Evolutionary Computation Conference (GECCO'04) Workshop on Modularity, regularity and Hierarchy

Lipson H, Pollack J, Suh N (2002) On the origin of modular variation. Evolution 56(8):1549–1556

Luke S, Spector L (1996) Evolving graphs and networks with edge encoding: Preliminary report. Late Breaking Papers at the Genetic Programming 1996 Conference pp 117–124

Mattiussi C, Floreano D (2004) Evolution of analog networks using local string alignment on highly reorganizable genomes. Evolvable Hardware, 2004 Proceedings 2004 NASA/DoD Conference on pp 30–37

Mattiussi C, Floreano D (2007) Analog Genetic Encoding for the Evolution of Circuits and Networks. IEEE Transactions on Evolutionary Computation 11:596–607

Michel O, Clergue M, Collard P (1997) Artificial neurogenesis: Applications to the cart-pole problem and to an autonomous mobile robot. International Journal on Artificial Intelligence Tools

Miller GF, Todd PM, Hedge SU (1989) Designing neural networks using genetic algorithms. In: Proceedings of the Third International Conference on Artificial Intelligence, Morgan Kaufmann,

pp 762–767
Mouret JB, Doncieux S, Meyer JA (2006) Incremental evolution of target-following neurocontrollers for flapping-wing animats. In: Nolfi S, Baldassare G, Calabretta R, Hallam J, Marocco D, Meyer JA, Miglino O, Parisi D (eds) From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB), Rome, Italy, pp 606–618
Nolfi S, Parisi D (1998) "Genotypes" for neural networks. The handbook of brain theory and neural networks pp 431–434
Paakki J (1995) Attribute grammar paradigms: a high-level methodology in language implementation. ACM Computing Surveys (CSUR) 27(2):196–255
Pasemann F, Dieckmann U (1997) Evolved neurocontrollers for pole-balancing. Biological and Artificial Computation: From Neuroscience to Technology, Proceedings IWANN 97:1279–1287
Reisinger J, Miikkulainen R (2007) Acquiring evolvability through adaptive representations. Proceedings of the 9th annual conference on Genetic and evolutionary computation pp 1045–1052
Reisinger J, Stanley K, Miikkulainen R (2004) Evolving reusable neural modules. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) pp 69–81
Siddiqi A (1998) Comparison of matrix rewriting versus direct encoding for evolving neural networks. The 1998 IEEE International Conference on Evolutionary Computation, ICEC'98 pp 392–397
Simon H (1962) The Architecture of Complexity. Proceedings of the American Philosophical Society 106(6):467–482
Sims K (1994) Evolving 3d morphology and behavior by competition. In: Brooks RA, Maes P (eds) Proceedings of the Fourth International Workshop on Artificial Life, The MIT Press, pp 28–39
Stanley K, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2)(99-127)
Stanley KO (2006) Comparing artificial phenotypes with natural biological patterns. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program, New York, NY
Teuber H (1955) Physiological Psychology. Annual Review of Psychology 6(1):267–296
Wagner G, Altenberg L (1996) Complex adaptations and the evolution of evolvability. Evolution 50(3):967–976
Watson R (2005) Modular interdependency in complex dynamical systems. Artificial Life 11(4):445–457
Widrow B (1987) The original adaptive neural net broom-balancer. International Symposium on Circuits and Systems pp 351–357
Wieland A (1991) Evolving neural network controllers for unstable systems. Neural Networks, 1991, IJCNN-91-Seattle International Joint Conference on 2
Yao X (1999) Evolving artificial neural networks. Proceedings of the IEEE 87(9):1423–1447

A Results of the Wilcoxon-Mann-Whitley test

A.1 Cartpole (two tails p-values)

	Direct enc.	ModNet	MLP	MENNAG	NEAT
Direct enc.	1.000	0.077	0.171	0.011	0.237
ModNet	0.077	1.000	0.025	0.074	0.206
MLP	0.171	0.025	1.000	0.013	0.101
MENNAG	0.011	0.074	0.001	1.000	0.040
NEAT	0.237	0.206	0.101	0.040	1.000

A.2 Robotic arm (two tails p-values)

	MENNAG	ModNet	Direct enc.	NEAT	MLP
MENNAG	1.000	0.002	0.019	$5 \cdot 10^{-6}$	0.395
ModNet	0.002	1.000	0.002	$7 \cdot 10^{-6}$	$4.6 \cdot 10^{-6}$
Direct	0.019	0.002	1.000	$5 \cdot 10^{-6}$	$3.07 \cdot 10^{-6}$
NEAT	$5 \cdot 10^{-6}$	$7 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	1.000	$5 \cdot 10^{-6}$
MLP	0.395	$5 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	1.000

A.3 Lesion experiments

	No clone	No cross-over	MENNAG (std)	No insert. mut.	Random mut.
No clone	1.000	0.712	$7 \cdot 10^{-5}$	0.315	$2 \cdot 10^{-5}$
No cross-over	0.719	1.000	$5 \cdot 10^{-5}$	0.130	$3 \cdot 10^{-5}$
MENNAG (std)	$7 \cdot 10^{-5}$	$6 \cdot 10^{-5}$	1.000	$10 \cdot 10^{-5}$	0.898
No insert. mut.	0.315	0.130	$10 \cdot 10^{-5}$	1.000	$3 \cdot 10^{-5}$
Random mut.	$2 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	0.898	$3 \cdot 10^{-5}$	1.000

B Complete attribute grammar

```
   // allN is the list of nodes
// allC the list of connections
// allI the list of input connections
// allO the list of output connections
[1.0,i1=0.5] START ⇒ DIV CONNS
    string DIV.id = ""
    set START.allN = DIV.allN
    string CONNS.divId = DIV.id
    set START.allC = union(set=DIV.allC, set=CONNS.allC)
    set START.allI = DIV.allI
    set START.allO = DIV.allO
// n being the depth of the generated tree, this probability decreases when the depth increases.
[1/(n*n)] DIV ⇒ DIV DIV CONNS
    string CONNS.divId = DIV.id
    string DIV_1.id = concatenate(DIV_0.id, "0")
    string DIV_2.id = concatenate(DIV_0.id, "1")
    set DIV.allC = union(set=DIV_1.allC, set=DIV_2.allC, set=CONNS.allC)
    set DIV.allN = union(set=DIV_1.allN, set=DIV_2.allN)
    set DIV.allI = union(set=DIV_1.allI, set=DIV_2.allI)
    set DIV.allO = union(set=DIV_1.allO, set=DIV_2.allO)
//Similarly, this probability increases when the depth of the generated tree increases.
[1.0-1.0/(n*n)] DIV ⇒ CELL CELL CONNS
    string CONNS.divId = DIV.id
    string CELL_0.divId = DIV.id
    string CELL_1.divId = DIV.id
    string CELL_0.id = concatenate(DIV.id, "0")
    string CELL_1.id = concatenate(DIV.id, "1")
    set DIV.allN = union(set=CELL_0.node, set=CELL_1.node)
    set DIV.allC = CONNS.allC
    set DIV.allI = union(set=CELL_0.allI, set=CELL_1.allI)
    set DIV.allO = union(set=CELL_0.allO, set=CELL_1.allO)
// Implementation of the creation of neuron. The spec could be changed to create another kind of neuron. A
potential input and output connections are associated to each neuron (cell).
[1.0] CELL ⇒ IN OUT
    int CELL.spec = 0
```

set CELL.node = createset(createhash(name="id", value=CELL.id, name="spec", value=CELL.spec))
set CELL.allI = IN.allI
set CELL.allO = OUT.allO
string IN.divId = CELL.divId
string OUT.divId = CELL.divId
string IN.node = CELL.id
string OUT.node = CELL.id
// i1=probability to select this rule as r_1
// i2=probability to select this rule as r_2
[0.01,i2=0.5,i1=0.5] DIV ⇒ DIV CLONES CONNS
string DIV_1.id = concatenate(DIV_0.id, "0")
string CLONES.id = concatenate(DIV_0.id, "1")
string CONNS.divId = DIV.id
set CLONES.divN = DIV_1.allN
set CLONES.divC = DIV_1.allC
set CLONES.divI = DIV_1.allI
set CLONES.divO = DIV_1.allO
set DIV.tmp = addattribute(data=CLONES.allN, attname="cloneId", value=DIV_1.id)
set DIV.allN = union(set=DIV_1.allN, set=DIV.tmp)
set DIV.allC = union(set=CONNS.allC, set=DIV_1.allC, set=CLONES.allC)
set DIV.allI = union(set=DIV_1.allI, set=CLONES.allI)
set DIV.allO = union(set=DIV_1.allO, set=CLONES.allO)
[0.01,i2=0.5,i1=1.5] DIV ⇒ CLONES DIV CONNS
string DIV_1.id = concatenate(DIV_0.id, "1")
string CLONES.id = concatenate(DIV_0.id, "0")
string CONNS.divId = DIV.id
set CLONES.divN = DIV_1.allN
set CLONES.divC = DIV_1.allC
set CLONES.divI = DIV_1.allI
set CLONES.divO = DIV_1.allO
set DIV.tmp = addattribute(data=CLONES.allN, attname="cloneId", value=DIV_1.id)
set DIV.allN = union(set=DIV_1.allN, set=DIV.tmp)
set DIV.allC = union(set=CONNS.allC, set=DIV_1.allC, set=CLONES.allC)
set DIV.allI = union(set=DIV_1.allI, set=CLONES.allI)
set DIV.allO = union(set=DIV_1.allO, set=CLONES.allO)
// list of clones
[0.01,i2=0.5] CLONES ⇒ CLONE CLONES
string CLONE.id = concatenate(CLONES_0.id, "0")
string CLONES_1.id = concatenate(CLONES_0.id, "1")
set CLONE.divN = CLONES.divN
set CLONE.divC = CLONES.divC
set CLONE.divI = CLONES.divI
set CLONE.divO = CLONES.divO
set CLONES_1.divN = CLONES.divN
set CLONES_1.divC = CLONES.divC
set CLONES_1.divI = CLONES.divI
set CLONES_1.divO = CLONES.divO
set CLONES.allN = union(set=CLONE.allN, set=CLONES_1.allN)
set CLONES.allC = union(set=CLONE.allC, set=CLONES_1.allC)
set CLONES.allI = union(set=CLONE.allI, set=CLONES_1.allI)
set CLONES.allO = union(set=CLONE.allO, set=CLONES_1.allO)
// End of the list of clones
[1.0] CLONES ⇒ CLONE
string CLONE.id = CLONES.id
set CLONE.divN = CLONES.divN
set CLONE.divC = CLONES.divC
set CLONE.divI = CLONES.divI
set CLONE.divO = CLONES.divO
set CLONES.allN = CLONE.allN

set CLONES.allC = CLONE.allC
set CLONES.allI = CLONE.allI
set CLONES.allO = CLONE.allO
// Implementation of the clone
// permi is the vector of permutation for inputs, permo the one for outputs. 2 is the number of inputs and 1 the number of outputs. These numbers have to changed if you add new I/Os
// The CLONE.id is concatenated to each id found in the cloned DIV. I/Os are permutated using permo and permi.
[1.0] CLONE ⇒ end
evofloat[2] CLONE.permi = **evolved**
evofloat[1] CLONE.permo = **evolved**
set CLONE.allN = prependtoattributes(data=CLONE.divN, attname="id", value=CLONE.id)
set CLONE.tmp = prependtoattributes(data=CLONE.divC, attname="target", value=CLONE.id)
set CLONE.allC = prependtoattributes(data=CLONE.tmp, attname="node", value=CLONE.id)
set CLONE.tmpii = prependtoattributes(data=CLONE.divI, attname="node", value=CLONE.id)
set CLONE.tmpi = prependtoattributes(data=CLONE.tmpii, attname="target", value=CLONE.id)
set CLONE.allI = map(data=CLONE.tmpi, pushedname="data", foncteur=setattribute(value=aspermutation(data=CLONE.permi, index=getattribute(data=∴data, attname="spec")), attname="spec"))
set CLONE.tmpoo = prependtoattributes(data=CLONE.divO, attname="target", value=CLONE.id)
set CLONE.tmpo = prependtoattributes(data=CLONE.tmpoo, attname="node", value=CLONE.id)
set CLONE.allO = map(data=CLONE.tmpo, pushedname="data", foncteur=setattribute(value=aspermutation(data=CLONE.permo, index=getattribute(data=∴data, attname="spec")), attname="spec"))
[0.5] CONNS ⇒ CONN CONNS
string CONN.divId = CONNS.divId
string CONNS_1.divId = CONNS.divId
set CONNS.allC = union(set=CONNS_1.allC, set=CONN.allC)
[0.5] CONNS ⇒ end
set CONNS.allC = createset()
// This variant of CONN creates a feed-forward connection by forcing the source of the connection to be in the left branch of the tree and the target to be in right branch. This is implemented by modifying the Ids.
[0.9] CONN ⇒ ICONN
set CONN.allC = ICONN.allC
string ICONN.targetBaseId = concatenate(CONN.divId, "1")
string ICONN.nodeBaseId = concatenate(CONN.divId, "0")
// This is the basic (not feed-forward) connection
[0.1] CONN ⇒ ICONN
set CONN.allC = ICONN.allC
string ICONN.targetBaseId = CONN.divId
string ICONN.nodeBaseId = CONN.divId
// Implementation of a connection.
// A connection is made of a source node, a target, a weight and a specification (spec). The spec could be changed to create different kind of connections.
[1.0] ICONN ⇒ weight node target
evofloat weight.value = **evolved**
bit*string* target.value = **evolved**
bit*string* node.value = **evolved**
float weight.float = numericalvalue(weight.value)
string ICONN.target = concatenate(ICONN.targetBaseId, target.value)
string ICONN.node = concatenate(ICONN.nodeBaseId, node.value)
set ICONN.allC = createset(createhash(name="node", value=ICONN.node, name="target", value=ICONN.target, name="weight", value=weight.float, name="spec", value=0, name="type", value="internal"))
// An empty IN connection (no connection to sensors for this neuron)
[0.5] IN ⇒ end
set IN.allI = createset()
// Connection to the first sensor.
[0.3,p2=0.1] IN ⇒ IO in1
int IO.spec = 0
set IN.allI = IO.allC

```
        string IO.divId = IN.divId
        string IO.node = IN.node
        string IO.type = "in"
  // Connection to the second sensor.
  [0.0,p2=0.1] IN ⇒ IO in2
        int IO.spec = 1
        set IN.allI = IO.allC
        string IO.divId = IN.divId
        string IO.node = IN.node
        string IO.type = "in"
  [0.5] OUT ⇒ end
        set OUT.allO = createset()
  // Connection to the first actuator
  [0.3,p2=0.1] OUT ⇒ IO out1
        int IO.spec = 0
        set OUT.allO = IO.allC
        string IO.divId = OUT.divId
        string IO.node = OUT.node
        string IO.type = "out"
  // Add here new in and out if needed
  // Implementation of IO connections
  // The source node is the node passed down when the cell has been created;
  [1.0] IO ⇒ weight target
        evofloat weight.value = evolved
        bitstring target.value = evolved
        string IO.target = IO.node
        float weight.float = numericalvalue(weight.value)
        set IO.allC = createset(createhash(name="node", value=IO.node, name="target", value=IO.target,
  name="weight", value=weight.float, name="spec", value=IO.spec, name="type", value=IO.type))
```

C Defined types and functions

Mennag uses a short list of types and functions to handle attributes. Here is a short reference.

Seven basic types are used in the MENNAG grammar:

- string: character string
- bitstring: string of bits which can mutate (bit flipping)
- float: real number
- evofloat: real number which can mutate (Gaussian mutation)
- int: integer
- set: a simple set which can handle any type
- hash: a simple hash table

A vector of these types can be defined by adding the size enclosed in brackets. For instance, “evofloat[10]” defines a vector of 10 evofloat.

Hash tables are used to emulate a structured type (struct in C). For instance, a neuron is “created” using “createhash(name="id", value=CELL.id, name="spec", value=CELL.spec)”.

Some simple functions are built-in:

- union: merge two sets
- concatenate: concatenate two strings
- createset: create an empty set
- createhash: create an empty hash table
- numericalvalue: convert the argument to a real number
- prependtoattributes: concatenate a string in front of all the specified entries in all the hash tables contained in a set
- addattribute: add an entry to every hash tables contained in a set
- map: apply a functor to every elements of a set
- aspermutation: compute the permutation corresponding to a vector a real numbers

D Experimental setup

D.1 MENNAG

- cross rate: 0.3
- min. number of neurons (random generation): 2
- max. number of neurons (random generation): 10
- min. number of connections (random generation): 2
- min. number of connections (random generation): 12
- mutation rate (random tree): 0.05
- insertion mutation rate: 0.2
- mutation delete rate: 0.1

D.2 NEAT

- starting network (cart-pole): each input directly connected to the output
- parameter file: p2nv.ne (available in NEAT's source code)

D.3 ModNet

- max. number of modules: 10
- cross rate: 0.6
- no predefined module in the initial module pool
- add model module rate: 0.01
- insert module rate: 0.01
- delete model module rate: 0.005
- mutation module rate: 0.05

D.4 Direct encoding

- max. number of neurons: 12
- min. number of neurons: 2
- max. number of connections: 10
- min. number of connections: 2
- add neuron rate: 0.1
- delete neuron rate: 0.1
- add connection rate: 0.25
- delete connection rate: 0.25
- change connection rate: 0.25

A.2 ARTIFICIAL EVOLUTION OF THE MORPHOLOGY AND KINEMATICS IN A FLAPPING–WING MINI UAV

de Margerie, E., Mouret, J.-B., Doncieux, S., and Meyer, J.-A. (2007). *Artificial evolution of the morphology and kinematics in a flapping-wing mini UAV*. **Bioinspir. Biomim.**, 2 :65-82.

Travaux réalisés dans le cadre du stage post-doctoral d'Emmanuel de Margerie dont j'étais l'encadrant scientifique.

Artificial evolution of the morphology and kinematics in a flapping-wing mini UAV.

Running title: *Evolution of a flapping-wing UAV*

Emmanuel de Margerie, Jean-Baptiste Mouret, Stéphane Doncieux and Jean-Arcady Meyer
SIMA dpt, ISIR, Université Pierre et Marie Curie
104 av. Pdt Kennedy, 75016 Paris, FRANCE.

corresponding author: edemargerie@gmail.com

Abstract
Birds demontrate that Flapping-wing flight (FWF) is a versatile flight mode, compatible with hovering, forward flight, and gliding to save energy. This extended flight domain would be especially useful on mini UAVs. However, design is challenging because aerodynamic efficiency is conditionned by complex movements of the wings, and because many interactions exist between morphological (wing area, aspect ratio) and kinematic parameters (flapping frequency, stroke amplitude, wing unfolding). Here we used Artificial Evolution to optimise these morpho-kinematic features on a simulated 1 kg UAV, equipped with wings articulated at the shoulder and wrist. Flight tests were conducted in a dedicated steady aerodynamics simulator. Parameters generating horizontal flight for minimal mechanical power were retained. Results showed that flight at medium speed (10-12 m/s) can be obtained for reasonnable mechanical power (20 W/kg), while flight at higher speed (16-20 m/s) implied increased power (30-50 W/kg). Flight at low speed (6-8 m/s) necessitated unrealistic power levels (70-500 W/kg), probably because our simulator neglected unsteady aerodynamics. The underlying adaptation of morphology and kinematics to varying flight speed were compared to available biological data on the flight of birds.

Keywords
unmanned aerial vehicle, flapping-wing flight, artificial evolution, morphology, kinematics.

1. Introduction

In order to achieve sustained horizontal flight, human flying machines usually rely on a fixed wing and a powered propeller such as an airplane, or on powered rotating wings like in helicopters. On the other hand, birds, bats, and insects - i.e., actively flying animals - use flapping-wing flight (FWF) to produce the lift and thrust forces needed for forward flight. The reasons why biological and technical worlds have retained different solutions may be of both historical (e.g., contingency) and structural (e.g., material constraints) nature. In particular, it appears that:

- Continually, rotating mechanical joints - on which propellers and rotors are based in human technology - do not exist in animals at the macroscopic, morphological level. Skeletal joints would belong to the category of rotating joints, but the dependence on muscles and tendons as force and torque effectors limit their angular rotation. Hence only *reciprocating* movements between skeleton elements are possible in animals. As a consequence of the historical, contingent constraint of inheriting a muscle-based activation system, propellers or rotating wings for active flight are beyond the reachable phenotypes of extant animals.

- Articulated, moving wings as necessitated by FWF are hard to design for human flight. At a man-lifting scale, with usual aeronautical materials such as wood, steel, aluminium or even newer composite materials, flapping-wing flight is a tremendous aerodynamic, mechanical and structural challenge for current technology. Although several "ornithopters" have been constructed in the last 100 years, even the most recent designs (DeLaurier 1999) remain marginally efficient compared to classical fixed-wing or rotor designs. As a result, the great potential of FWF demonstrated by animals in terms of speed range or manoeuvrability, though attractive, remains beyond the achievable goals of today's human aerial transportation prospect.

Although FWF seems rather impractical at a man-lifting scale, which is also illustrated by the fact that flying animals rarely exceed 10 kg in mass, the recent technological field of unmanned aerial vehicles (UAVs) may find in FWF solutions to challenging flight dynamics problems. As small size is a determining factor, research efforts on FWF are mostly focused on micro-UAVs (insect to small bird-sized: 1-100g) and mini-UAVs (medium to large bird-sized: 0.1-10kg). At these sizes, FWF has the potential to allow unique flight dynamics abilities, as demonstrated by the performance of flying animals :

- FWF is versatile. Depending on specific size and weight, flying animals can hover, fly forward at varying speeds, and glide or soar to save energy. Many species can perform all three flight regimes, as exemplified by the European kestrel, *F. tinnunculus.* These regimes are selected according to daily activities like foraging, observation or migration. These species merge both helicopter and airplane-like abilities into a single, extended flight domain.
- Active articulated wings and asymmetrical flapping provide very high manoeuvrability, especially useful in obstructed spaces, as demonstrated by perching birds that fly among tree branches for example.
- Reciprocating wing movements allow flying animals to use favourable *unsteady aerodynamics,* at least during hovering and slow flight (Norberg 1990). For example, one well-known unsteady effects is the "delayed stall", which can increase the airfoil maximal lift up to 50% when the wing's angle of attack is suddenly increased (Vogel 1994).

Drawing inspiration from natural flyers, one of the main interests in transposing FWF to small UAVs is the ability to obtain an extended flight-mode range. Namely, a hovering and very slow flight appears useful for exploration and observation in obstructed or urban areas, while a medium to high speed horizontal flight at low energetic cost, which is not well achieved by helicopters, is necessary to cover large distances. Moreover, the ability to soar in ascending air currents is a supplementary key-feature for energy saving. Although the perspective of implementing these three flight modes on a single UAV is very attractive, the main drawback in using FWF, as mentioned above, is the high difficulty of designing a mechanically and aerodynamically functional flapping wing, because of the large number of interacting parameters like morphological characteristics, degrees of freedom and kinematic data associated with the wing's parts.

In an attempt to overstep these difficulties, we used Artificial Evolution (AE) to explore the range of functional wing morphologies and kinematics for a simulated bird-like mini UAV. AE is a "trial and error"

optimisation method inspired from Darwinian natural selection. It may call upon several numerical optimisation procedures such as “Genetic Algorithms”, “Evolution Strategies”, and others (Goldberg 1989), which are used in engineering, artificial intelligence and biology to solve complex problems (for an AE application to biological morphology, see de Margerie et al 2005). Compared to classical knowledge-driven engineering methods, AE has the main advantage that it does not need a comprehensive high-level knowledge of the considered problem. In our case, only the basic laws of aerodynamics need to be implemented in a simulator, to make the generation of various FWF solutions possible. By selecting the best among such randomly generated solutions, by randomly recombining and mutating them, and by testing the “offspring” solutions again, AE can generate satisfactory optimised solutions to intricate problems, with minimal initial knowledge. This can be useful for solving the FWF problem because one only needs to know rather general aerodynamic laws to test any AE-produced wing morphology, or any flapping movement, in a flight simulator.

The present work is part of the ROBUR project (Doncieux *et al* 2004) that aims at designing an outdoor bird-like mini UAV of 1-2m wingspan, with adaptive locomotion modes and abilities required for a true decisional autonomy like obstacle avoidance (Muratet *et al* 2005), localization and mapping (Angeli *et al* 2006) and energy management (Barate *et al* 2006) . Here we focus on wing morphology and movements and use AE to find morphological and kinematic parameters providing flapping flight at minimal energetic cost. These parameters mainly include the wing area, the wing aspect ratio, the flapping frequency, the stroke amplitude and the angles of attack. As a first stepping stone, before later considering turning, ascending or acrobatic flight, we concentrated the present work on forward horizontal flight at varying speeds (6-20 $m.s^{-1}$), for an approximately 1 kg bird-like UAV.

A few previous works have used AE to optimize FWF, but with notable differences relatively to the present work. Salles and Schiele (2004), for instance, optimised the movement of a small rigid wing inspired from an hawkmoth's wing, manipulated at low Reynolds number by a robotic arm, using a genetic algorithm. Van Breugel and Lipson (2005) used an evolutionary algorithm to optimise the lift produced by a simulated 50-310g four-wing ornithopter. Although not using AE, Rakotomamonjy *et al* (2004) used an optimisation algorithm (non linear programming) on a neural network controlling the kinematics of a simulated 30g micro-UAV, in order to maximize the lift forces. Beyond differences in the UAV’s mass, our work differs from these previous studies by the fact that we simultaneously optimize the kinematics and the wing morphology (size and shape) of our UAV. To our knowledge, the work of Shim *et al* (2004) is the only other study evolving both the morphology and kinematics of bird- or bat-sized FWF UAVs. However, their optimisation process does not consider energy consumption - their fitness criterion being a sum of flight speed and hovering time - and accordingly embeds their study in a different perspective tied to artificial life and virtual worlds and then less realistic and applicable to real UAVs than ours.

2. Methods

2.1. UAV morphology and kinematics

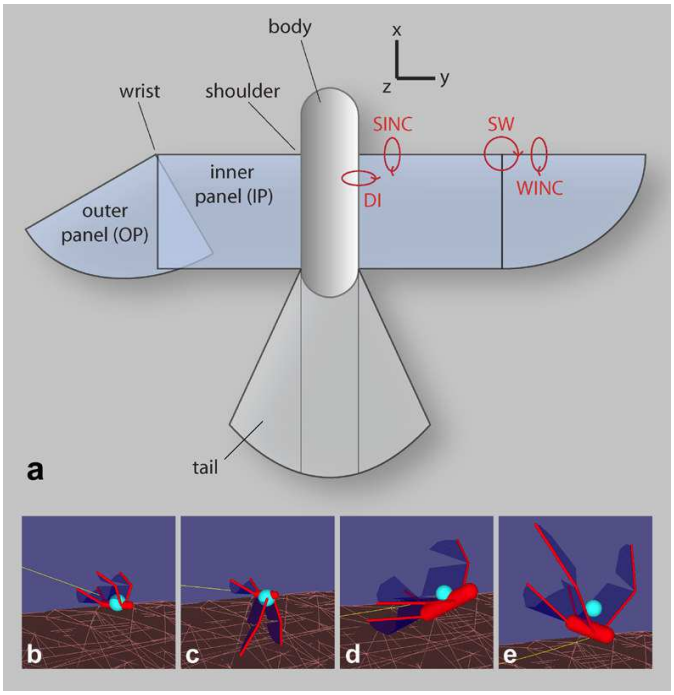


Figure 1 : UAV Morphology. **a**: Wing panels and their degrees of freedom (DOFs) ; dihedral (DI), sweep (SW), shoulder incidence (SINC) and wrist incidence (WINC). **b**, **c**, **d**, **e**: four possible morphologies, for extreme values of the wing area (a_w) and the wing aspect ratio (λ_w). UAVs are displayed in flight, with random angular values on the four DOFs. The light-blue sphere indicates the position of the UAV's center of gravity. The yellow line is the trajectory of the UAV's body. **b**: $a_w = 0.1$ m^2, $\lambda_w = 4.5$, wingspan = 0.67 m. **c**: $a_w = 0.1$ m^2, $\lambda_w = 10$, wingspan = 1 m. **d**: $a_w = 0.4$ m^2, $\lambda_w = 4.5$, wingspan = 1.34 m. **e**: $a_w = 0.4$ m^2, $\lambda_w = 10$, wingspan = 2 m.

Freely inspired from bird morphology, our simulated UAV had two symmetrical wings and a central tail, as described on figure 1. Each wing comprised an inner rectangular and an outer elliptic panel of equal spans, named IP and OP, respectively. Each wing had 4 degrees of freedom (DOF): rotation was possible in dihedral (x) and incidence (y) at the "shoulder" joint, i.e. between body and IP, while rotations in sweep (z) and incidence (y) were allowed at the "wrist" joint, i.e. between IP and OP. Rotation in sweep allowed the UAV to possibly retract its wings during the flapping stroke. A sweep angle implied that IP and OP partly overlap, hence entailing a decrease in the aerodynamically efficient wing area. This overlap interacted freely with incidence rotation at the wrist, i.e. potential collisions between panels were avoided by continuously modifying the panels' shape at their joint. As dihedral movements only happened at the shoulder and sweep movements at the wrist, these two DOFs will be referred to as "dihedral" (DI) and "sweep" (SW). Likewise, the two incidence DOFs will be distinguished with the terms "shoulder incidence" (SINC) and "wrist incidence" (WINC).

The size, shape and movements of the wings were determined by 12 parameters which were allowed to vary - within the bracketed limits given below - during the AE optimisation process. These parameters constituted the "genome" of our UAV:

1 – The wing area (a_w) [0.1 - 0.4m^2] was the "size" parameter, i.e. the sum of areas of both wings fully extended (no sweep).

2 – The wing aspect ratio (λ_w) [4.5 – 10] was the "shape" parameter, expressing the ratio of wing span - both

wings included - to wing chord.

3 – The flapping frequency (f) [1 – 10 Hz] was common to all DOFs - shoulder dihedral and incidence, wrist sweep and incidence - which were all controlled through sinusoidal functions.

4 – The dihedral amplitude (amp_{di}) [0 – 89°] was the amount of x angular oscillation between body and wing. Upward and downward rotation ranges of the wings were symmetric (e.g. 89° upward and 89° downward, zero-centered). The max value of 89° was chosen to prevent geometrical and numerical singularities at 90°.

5 – The sweep amplitude (amp_{sw}) [0 – 89°] was the maximal z angle between IP and OP leading edges. Contrary to dihedral, this range holds only backward, i.e. OP could not rotate frontward to IP in a "negative sweep" configuration.

6 – The sweep offset (off_{sw}) [0 – 500% of sinus period] determined the periodic offset between dihedral and sweep movements. A range of 500%, instead of the theoretically sufficient 100%, allowed this parameter to possibly evolve near to a 100% value without encountering any boundary.

7 – The shoulder incidence reference (ref_{sinc}) [-20 – 20°] is the default y angle between body and IP, around which the incidence sinusoidal oscillation occurred.

8 – The shoulder incidence amplitude (amp_{sinc}) [0 – 69°] is the amount of y angular oscillation between body and IP.

9 – The shoulder incidence offset (off_{sinc}) [0 – 500%] is the periodic offset between dihedral and shoulder incidence movements.

10 – The wrist incidence reference (ref_{winc}) [-20 – 20°] is the default y angle between IP and OP.

11 – The wrist incidence amplitude (amp_{winc}) [0 – 69°] is the amount of y angular oscillation between IP and OP.

12 – The wrist incidence offset (off_{winc}) [0 – 500 %] is the periodic offset between dihedral and wrist incidence movements.

Based on the preceding parameters, the time-dependent kinematic laws of the angular variations in DOFs were:

$$DI = amp_{di}.\sin(2.\pi.f.t) \quad (1)$$

$$SW = amp_{sw}.(\frac{1}{2} + \frac{1}{2}.\sin(2.\pi.(f.t + off_{sw}))) \quad (2)$$

$$SINC = ref_{sinc} + amp_{sinc}.\sin(2.\pi.(f.t + off_{sinc})) \quad (3)$$

$$WINC = ref_{winc} + amp_{winc}.\sin(2.\pi.(f.t + off_{winc})) \quad (4)$$

with t being the simulated time in seconds.

The remaining features of the UAV may be inferred either from the evolving parameters just mentioned, or from deliberate constraints such as:
The span (b_w) and inner chord (c_w) of each wing were direct geometrical outcomes of the wing area and aspect ratio:

$$b_w = \frac{\sqrt{\lambda_w . a_w}}{2} \quad (5)$$

$$c_w = \frac{a_w}{b_w . (1 + \frac{\pi}{4})} \quad (6)$$

The body of the UAV was a cylinder with rounded tips (figure 1). Its length (l_b) was proportional to the wing chord, and its radius (r_b) was a function of the wing area, such that the body cross-section was proportional to the wing area:

$$l_b = \frac{4}{3} . c_w \quad (7)$$

$$r_b = \frac{\sqrt{a_w}}{10} \quad (8)$$

The tail area (a_t) was proportional to the wing area. The tail parts extending laterally beyond body sides were raised at 45° around x axis to provide some lateral stability to the UAV through a V-shaped tail surface:

$$a_t = \frac{a_w}{2} \quad (9)$$

Tail position relative to the body was not allowed to change and remained constant along all experiments. A control of the tail should further increase our artificial bird stability, but we preferred not to include one, to rely on passive stability as much as possible.

The masses of the UAV elements were determined as follows:
Considering that the wing mass would represent a significant part of the total mass in a real FWF UAV, and considering that we aim at designing an approximately 1 kg flyer, the body mass (m_b) was set to:

$$m_b = 0.5\, kg \quad (10)$$

The wing mass (m_w, for both wings) depended on the wing area through an isometric relationship inspired from Greenewalt (1975). The tail mass (m_t) was estimated through a similar relationship:

$$m_w = 2 . (a_w)^{1.5} \quad (11)$$

$$m_t = (a_t)^{1.5} \quad (12)$$

The masses of wings and tail were concentrated in their respective leading edges, and uniformly distributed along the span. The mass distribution within the body was such that the whole UAV's center of gravity (CG) was located at about 25% of wing chord when the wings were fully extended.

The chosen ranges and laws of variation of our parameters can now be compared to biological data.
Starting from the hard constraint that the body of our UAV had a 0.5 kg mass, we found several species with similar masses in the data of Magnan (1922, in Greenewalt, 1962). These species, among which are Hooded Crow (*Corvus cornix*), Shearwater (*Calonectris diomedea*) or Marsh-Harrier (*Circus aeruginosus*), have wing areas ranging from 0.13 to 0.23 m^2, except some "duck-model" fast-flying species like Shoveler (*Anas clypeata*) or Coot (*Fulica atra*), which have wing areas as low as 0.06 m^2. Choosing a range of (0.1 – 0.4)

for a_w covered the whole biological size range with the same mass, except the smallest, highly loaded duck-styled flyers, which can be considered as high-speed flight specialists, differing from our versatile UAV objective. As a direct outcome of the wing area, the wing loading of our UAV (ratio of total mass to wing area) could range from 2.6 for maximal a_w to 5.7 $kg.m^{-2}$ for minimal a_w. According to Greenewalt (1975, p.16), this range covers the natural wing loadings of all sampled groups, including raptors [*Falconiformes*], owls [*Strigiformes*], herons [*Ardeidae*] and bats of comparable masses, except the highly loaded ducks and shorebirds. In the latter groups, wing loading usually attains 6 – 10 $kg.m^{-2}$, while Auks [*Alcidae*] can attain even higher wing loadings, up to 24 $kg.m^{-2}$.
Concerning wing masses, relationship (11) was inspired from isometric biological data, but yielded slightly higher UAV wing masses compared to the above mentioned bird species, especially at higher wing areas. For example, we predicted $m_w = 0.22$ kg for a 0.23 m^2 wing area, whereas wings of a Marsh-Harrier are 0.14 kg according to Magnan. We chose to be rather conservative regarding this issue, as real articulated UAV wings will probably not equal natural performances regarding weight saving. We were also conservative when assuming a uniform wing mass distribution along the span of our UAV, rather than a decrease in mass toward the wing tips as it is observed in birds, which reduces the inertial power required for flapping flight (Van den Berg and Rayner 1995).
According to Norberg (1990, p.173), the aspect ratio of birds weighting 0.75 kg - a rough estimation of the mean total mass of our UAV including body, wing and tail - averages 7.7. When individual species are considered in Magnan's data, λ ranges from 6 for Hooded Crow to 9 for Shearwater. We allowed AE to search the optimal λ between 4.5 and 10. The lower boundary is imposed by our wing geometry for consistent panel overlap. The upper boundary at 10 was chosen to avoid generating high-λ virtual morphologies that would probably not be stiff or strong enough in reality. This precaution was necessary, as our mechanical model did not take structural resistance into account (see below).
Last, the range of frequency we allowed (1-10 Hz) is similar to biological data corresponding to the same mass (2-10Hz; Norberg 1990 p.177).

Drawing inspiration from birds for setting the variation ranges of the morphology and kinematics of our UAV is deliberate. It does not warrant that the optimal FWF UAV will be found within these limits. It is definitely possible that a more efficient UAV design virtually exists beyond what nature has ever explored (especially using artificial construction materials). However as one observes natural flyers' skills and versatility, it is obvious that an UAV exhibiting at least some of these features would already be a large step forward compared to current humanly-designed flying machines. In a broader perspective, our deliberately bio-mimetic strategy falls within the scope of the "animat approach", which aims at *designing simulated animals or real robots whose structure and functionalities are inspired from current biological knowledge, in the hope that they will exhibit at least some of the versatile capacities of real animals*. Rather than direct inspiration consisting of copying the morphology and movements of a single bird species, here we launch artificial evolution within a biologically-informed search space, in order to obtain results that can be compared *a posteriori* to biological data for validation and analysis, e.g. evolutionary trends and comparative adaptations. Interestingly, some of the results obtained in simulation, in a fully controlled environment (contrary to real world experiments) may, in turn, appear useful to biologists to disentangle the complex biological adaptations such as those involved in FWF.

2.2. Flight simulation
To test the flight characteristics of the multiple FWF solutions generated by the AE algorithm in terms of wing size, shape and movements, a flight simulator using the air speed vector at local points of the wing and tail was needed to compute the generated aerodynamic forces. This vector was a composition of both the UAV speed and the speed induced by the wing stroke. Moreover, possibly high local angles of incidence and lateral drift due to sweep had to be accounted for.
We used a model specifically designed for flapping articulated wings (FMFAW, *Flight Mechanics for Flapping Articulated Wings*), which has been described elsewhere (Druot 2004). This semi-empirical, quasi-steady-aerodynamics model considered that a wing was divided in a number of rigid flat quadrangular wing elements (WEL). In the present work, we divided each wing's IP into 3 coplanar WELs, and each OP into 6 coplanar WELs. At each time step (0.005 sec in the present work) and for each WEL, the model estimated the local incident airspeed, and computed 3 cumulative aerodynamic forces: the leading edge lift, the parachute drag and the friction drag, as decribed in more details by Druot (2004) . As the size and shape of our wings could vary, it was necessary that the model took the aspect ratio and the Reynolds number into account. The wing aspect ratio (λ_w) was accounted for by classical induced lift and drag formulae:

$$Cl_\lambda = Cl.\frac{\lambda_w}{\lambda_w + 2} \quad (13)$$

$$Cd_\lambda = Cd + \frac{Cl^2}{\pi.\lambda_w} \quad (14)$$

with Cl_λ, Cd_λ being the lift and drag coefficients corrected for aspect ratio, and Cl, Cd uncorrected coefficients, i.e. at infinite aspect ratio. λ_w was common to all WELs, such that the whole wing's aspect ratio affected the performance of each WEL, whereas the wing movement effects on the effective aspect ratio were neglected for simplicity. Moreover, no particular loss of aerodynamic efficiency was assumed at the body/IP and IP/OP interfaces because it was uneasy to determine *a priori* what mechanical solution would be adopted for the real UAV's shoulder and wrist, and because we did not want the optimisation process to depend on such matters initially.
The Reynolds number (Re) was assumed to have an effect on the friction drag coefficient in FMFAW (Cd_f). According to Norberg (1990), we assumed a dependence on $Re^{0.5}$ for laminar flow, and on $Re^{0.2}$ for turbulent flow:

$$Cd_f = \max(8\mathrm{Re}^{-0.5}; 0.2\mathrm{Re}^{-0.2}) \quad (15)$$

Contrary to the aspect ratio, Re (and thus Cd_f) was considered as a "local" variable, computed at each time step and for each WEL individually.
As a result of (15), the transition from laminar to turbulent flow happened in our model around Re 2.10^5. As for other parameters in FMFAW, e.g., those setting the dependence of the lift coefficient on the angle of incidence, proportionality coefficients in (15) were chosen to have the closest possible fit with experimental data for a particular airfoil. We chose the Selig 4083 airfoil, which is a 8% thick, under-cambered airfoil designed for providing high lift and lift/drag ratio at low Reynolds number (6.10^4 – 2.10^5). Its performances were measured experimentally in a wind tunnel for Re 6.10^4-3.10^5 (Selig 1997). Figure 2 compares these experimental curves with the outcome of FMFAW as used in the present work. Beyond providing useful experimental data to calibrate our model, we chose the Se 4083 for its affinities with wing airfoils in birds, in terms of shape, thickness, camber, maximum lift coefficient and Reynolds number range (Withers 1981).

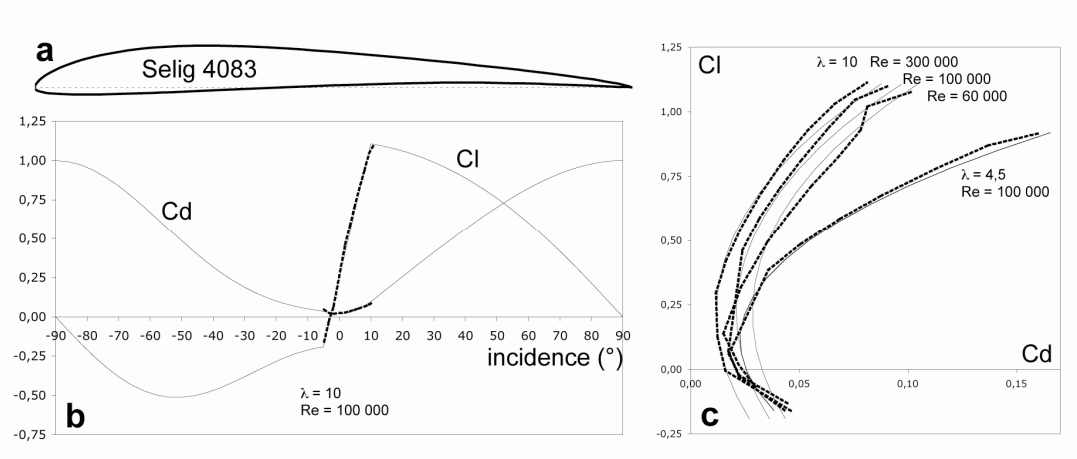


Figure 2 : Lift and drag coefficients as computed by the *FMFAW* model (thin lines), **compared to experimental measures on the Selig 4083 airfoil** (corrected for aspect ratio effect, dotted lines). **a**: outline of the Selig 4083 airfoil. **b**: Lift (Cl) and Drag (Cd) coefficients of wing surface elements, as a function of angle of incidence. Although experimental data are only available in the "common" incidence range (-5 to 10°), *FMFAW* estimates values for the whole 180° incidence range. **c**: effects of Reynolds number (Re) and aspect ratio (λ) variations on the polar lift-drag curve.

Concerning the UAV tail, aerodynamic forces were calculated similarly to the wings, but assuming a symmetrical Naca 009 airfoil, and a 1.0 value for aspect ratio. Finally, the body of the UAV was assumed to produce drag only, with a drag coefficient of 0.3 indexed on its frontal area (Norberg 1990, p.165).

Although the fit between FMFAW aerodynamic forces and experimental data at varying Reynolds number seems rather satisfactory (figure 2), FMFAW remains based on steady aerodynamics, and thus does not compute unsteady aerodynamics effects, nor interactions between UAV's parts. Thus the quantitative results of simulations must be interpreted with enough caution, especially at low flight speed, where unsteady effects, interactions and flight in disturbed air grow in importance. For this reason, we optimised our UAV using FMFAW for flight speeds ranging from 6 to 20 $m.s^{-1}$, but not for lower flight speeds, nor for hovering flight.

Aside FMFAW and flight mechanics, we used the Open Dynamics Engine (ODE, Smith 2006) to simulate the relative movements of each part of our articulated UAV and compute its flight trajectory. Body and wing parts, considered as not deformable solids, were attached using joints having the same DOFs as in figure 1. Sinusoidal angular movement of these 4 DOFS were obtained by producing enough torque at the joints to follow precisely the desired kinematic curve as dictated by the UAV's genome. High torques, possibly up to unrealistic values, were allowed to be produced at some joints if this was necessary to follow the "genetic" kinematics against possibly strong external forces, i.e. weight and aerodynamic forces on wing panels. It was the role of evolution to find an adequate, realistic wing morphology and movement that minimized the torques at joints, and hence decreased the required mechanical power, while achieving forward flight at a given speed.

2.3. Evaluation, Fitness and Evolutionary Algorithm

First, we briefly remind the general principles of Artificial Evolution, and how AE draws inspiration from natural random variation and Darwinian selection.

Each potential solution generated by AE is called an "individual", as its characters are dictated by a genome (a chain of 12 floating point numbers in the present work), and expressed into a phenotype (specific morphology and kinematics) interacting with a simulated world (flight simulation). The "fitness" of each individual relatively to the chosen problem (forward horizontal flight) is measured during its lifetime (the duration of an evaluation), and determines its breeding success, i.e. the chance that its genome will be selected for creating a new individual or "offspring" at the next evolution step. When an individual possesses a high fitness and is selected for offspring production, its genome is copied, crossed with the genome of another selected individual, randomly mutated, and then expressed into the offspring's phenotype, which is evaluated in turn.

In the present case, the fitness of each individual was assessed through a standardized test flight: the UAV was launched forward at an initial 300m height, with a given initial horizontal speed which was constant for all individuals within an evolutionary "run" (e.g. 10 $m.s^{-1}$). The genetically determined kinematics of the individual were symmetrically applied at joints of both wings since the first time step of evaluation and for 10 seconds - i.e. 2000 time steps - during which the UAV flew freely: no particular constraint was applied to its trajectory to "help" it achieve a stable horizontal flight. At each time step of the evaluation, two variables were recorded in order to quantify the fitness of the individual:

- the distance (D) between the UAV's body and the ideal "reference" trajectory, i.e. an horizontal path at the initial launch speed (e.g. 10 $m.s^{-1}$).
- the instantaneous mechanical power (P) produced at the wings' joints (shoulders and wrists). This variable was computed as the sum, for all 4 joints, of the scalar product between the instantaneous torque (τ) and the instantaneous rotational speed (ω).

$$P = \sum_{i=1}^{4} \left| \vec{\tau}_i . \vec{\omega}_i \right| \quad (16)$$

with i referring to each individual joint.

Note that the power was counted positive regardless of the sign of the scalar product. This means that the torques produced to accelerate the instantaneous joint rotation and the torques used to slow

down the rotation were assumed to have equivalent energetic costs. We chose this conservative hypothesis which maximized the power consumption, *a priori* assuming that the real UAV would probably not have an elastic energy storage capacity. Referring to assumptions in the biological literature about the power consumption of birds, some authors consider the acceleration only and ignore the decelerating power, although some other recommend to add both, as we did (Van den Berg and Rayner 1995).

At the end of the evaluation flight, the fitness of the individual was determined by two separate criterions, based on D and P respectively: the maximal value attained by D during the 10 sec - which quantified how far from its reference horizontal path the UAV's trajectory diverged -, and the mean absorbed P - which measured the mechanical power cost of the achieved flapping movement, given the individual's size and shape:

$$Fitness = \left[-\max(D); -mean(P)\right] \quad (17)$$

Both fitness parameters were minus signed, because selection in the Evolutionary Algorithm we used favors high fitness values, and because we aimed at reducing both the trajectory divergence and the power consumption.

The Evolutionary Algorithm we used (*epsilon-MOEA*; Deb *et al* 2005) is a multi-objective algorithm, that takes into account the two just-mentioned fitness criterions simultaneously without merging them into one single fitness value as many other algorithms proceed. As an outcome, not a single but several individuals are considered as the "bests". The selection scheme is based on the concept of "domination": within the population, the best individuals are those which have fitness values such that no other individual has higher values on both fitness criterions. The individual is then declared "non-dominated" (see figure 3). By this rule, many individuals in the population can be non-dominated, representing locally optimal compromises between fitness criterions. These individuals are called "Pareto-optimal" solutions, and constitute the most favoured individuals for offspring production. In *epsilon-MOEA*, Pareto-optimal individuals are placed in what is called an "elite" group, from which one of the two parents implied in each offspring production is systematically chosen at random. The other parent is chosen among the population (see Deb *et al* 2005 for details). An important particularity of *epsilon-MOEA*, compared to some other multi-objective algorithms, is that individual in the elite must differ from each other by some fitness increment: 0.1m in trajectory divergence and 1.0 W in power consumption, in the present case. In other words, the "Pareto front" of the population is interval-sampled. It prevents too much similarity between favoured genitor individuals within the elite, which often causes the premature convergence of evolution toward a local optimum.

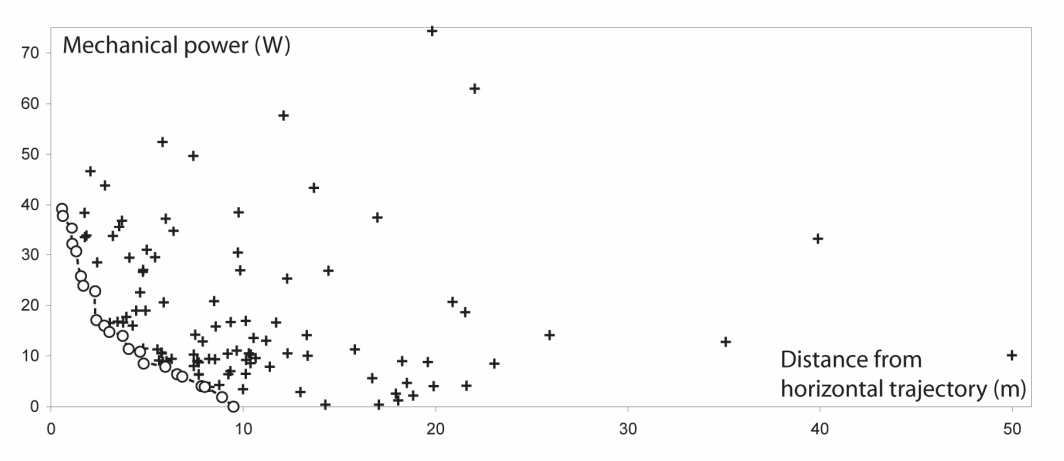


Figure 3: Population and fitness values. Plot of the two fitness criterions (absolute values) for the whole population, at an intermediary step of evolution (example of a 10 m/s run after 10000 generation). Each cross represents an individual, i.e. the phenotypic performance of a given genome (a combination of 12 parameter values). The "best" individuals (i.e. "non-dominated individuals" or "Pareto front", see text) are those in the

bottom left corner (i.e. lower power consumption and lower departure from reference trajectory), represented as white dots. The next generation will consist of crossing the genome of one individual of the elite (white dot) with one individual of the remaining population (cross). If the offspring is better than both its parents, it will appear closer to the bottom left corner, hence making the Pareto front progress towards better performance.

The sequence of an evolutionary run was as follows:
1 – 2000 individuals with randomly generated genomes were created and individually tested.
2 – the best individuals, in the sense of the multi-objective fitness just exposed, were retained to constitute a first elite group, and the 100 next individuals, only dominated by the elite, constituted the root population for evolution. Other individuals were discarded.
3 – an individual of the elite was randomly chosen for mating with another individual drawn from the population.
4 - the two genomes were crossed to produce an offspring genome: each of the 12 parameters in the offspring genome was randomly chosen from one or the other parent.
5 – the offspring genome was mutated, with a probability of 30% for each parameter: the corresponding value was randomly increased or decreased by some amount. This amount was randomly drawn from a normal distribution of mean 0 and variance 4% of the parameter authorized range.
6 – the offspring genome was expressed into a phenotype tested into the flight simulator, and its fitness values were measured.
7 – If its fitness values made the offspring a non-dominated individual compared to individuals currently in the population and the elite, it joined the elite group. Otherwise, it was only placed in the "regular" population, with the condition that it was able to replace a relatively worse individual. Otherwise, the offspring was discarded.
8 – steps 3 to 7 were repeated 50 000 times. In the following, we will refer to such a cycle as a *generation.*
At the end of the evolutionary run, the performance, as well as the morphological and kinematic parameters, of individuals in the final elite group were scrutinized.

In order to assess the influence of the flight speed on the evolutionary adaptation of wing size, shape, and movement, we conducted separate evolutionary runs with 6, 8, 10, 12, 16 and 20 $m.s^{-1}$ initial horizontal speeds. In all runs, only the flight speed was changed: all individuals had a 500 g body mass, and the same possible range of variation for other parameters. As AE is a stochastic optimisation method implying many random draws in initial genome generation, as well as in the crossover and mutation processes, we felt necessary to launch 4 duplicate runs per flight speed value in order to estimate how much the resulting adapted morphologies and kinematics converged (or diverged towards different local optima). Hence a total of 24 independent evolutionary runs was conducted initially, each one representing 52000 test flights. As a whole, this represented a total of 3500 virtual flight hours, and approximately 1500 hours of computation on standard personal computers (2 Ghz processor with 512 Mb of RAM). In a second stage, some supplementary runs were launched for further analysis (see results and discussion).

3. Results and Discussion

3.1. Progressive emergence of forward horizontal flight

Figure 4 shows the progression of the Pareto front ("elite") of the population through successive generations for one of the evolutionary runs aiming at a horizontal flight speed of 10 $m.s^{-1}$. The randomly generated front of the initial population already contained different compromise solutions to the horizontal FWF problem. On the right-hand part of the front were located individuals consuming little or no power, but sensibly departing from the horizontal trajectory. These individuals consumed no power because they usually did not move their wings at all, or only through passive movements, i.e. caused by external forces only, and they may be assimilated to "gliders". As a consequence, they systematically lose height during the test flight, as quantified by their score on the distance to the reference trajectory criterion: 15-20 m departure from the horizontal 10 $m.s^{-1}$ trajectory at the end of the 10 s. flight. On the left-hand part of the initial front were individuals flying closer to an horizontal path (4 m departure), but at the cost of high power consumption, up to 200 W of mechanical power in that run. Such high power values were usually due to large wing flapping, at high frequency and with non-optimal incidence angles. Between these two extreme solutions were a few other non-dominated individuals with intermediate performances: better than pure gliders on trajectory departure, and better than the most active flappers on power consumption. After the first 500 generations, the Pareto front progressed significantly, meaning that some offspring were better than their parents and replaced them in the elite group. This was true for all types of solutions: gliders lose less height (12m) and active flappers consumed less power (120w). Better intermediate solutions were also found. Similarly, after 1000 offspring generations, the front progressed further, especially for intermediate solutions that became more numerous. Later in the evolution, after 10000 generations, the best glider attained a height loss reduced around 10 m, hence a glide ratio of approximately 10, since the reference trajectory for 10 s at 10 $m.s^{-1}$ was a 100 m horizontal path. On the other hand, the departure from horizontal trajectory was reduced to 0.6 m, for 40 W consumed. During the last 40000 offspring generations, the evolutionary algorithm processed much slower, with the generation of individuals decreasing the power consumption on the "active flapper", on the left-hand side of the Pareto front (mechanical power finally dropped to 25 W), and little or no progress on the "glider", on the right-hand side of the front, attaining limits of the airfoil's lift/drag ratio.

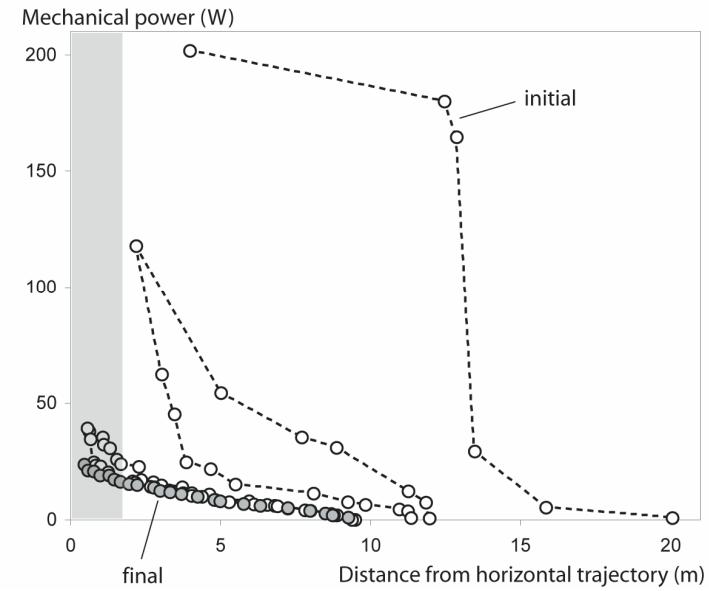


Figure 4: Progression of the elite population through 50000 generations. For the same evolutionary run as in figure 3 (10 m/s flight), the Pareto front of the population is plotted at 0 (initial random individuals), 500, 1000, 10000, 20000 and 50000 generations. Within the final elite group (gray dots), only individuals departing less than 2m from perfect horizontal flight (gray area) are retained for further analysis.

Evolutionary runs at other reference flight speeds (6, 8, 12, 16 and 20 $m.s^{-1}$) displayed the same trends, though with varying power consumption values. There was an early emergence of glider solutions, which satisfied only one of the fitness criterions, and a more progressive evolution of active flappers and

intermediate solutions toward lower power consumption and lower departure from horizontal flight.
At the end of each evolutionary run, after 50000 generations, we retained a few relevant individuals for further analysis of their morphological and kinematic parameters. We were only interested in horizontal flight, and not in glider optimisation, despite the fact that evolution possibly used gliders as parents of active flappers, taking advantage of the multi-objective optimisation scheme. Therefore, we tolerated a maximal departure of 2 m from horizontal flight over the 10s flight (see figure 4). Only individuals satisfying this *a posteriori* constraint were analysed later in the study. Whether this relative tolerance on trajectory departure represents a significant bias on power consumption can be evaluated by considering that, in the worst case, a 2 m height loss for a 1 kg UAV represents 20 J of lost potential energy during 10 s, hence a power saving of the order of magnitude of a few Watts, which remains tolerable compared to the optimised power levels attained at varying flight speeds (see par. 3.2.1). Furthermore, small errors on altitude might also be due to the lack of pitch closed-loop control, a situation that implies a very accurate parameter tuning that could disappear in a closed-loop control system. All evolutionary runs, at all tested flight speeds, yielded individuals satisfying this constraint after 50000 generations, though in variable number: fewer individuals succeeded in performing a sub-horizontal flight at the lowest (6 $m.s^{-1}$) and highest (20 $m.s^{-1}$) flight speed, compared to results obtained with intermediate speeds.

3.2. Morpho-kinematic adaptation to varying flight speeds : comparative analysis
Figures 5 presents the results of the 24 evolutionary runs we launched, i.e. 4 duplicate runs for 6 flight speeds. We plotted the performance and parameters of the horizontal active flappers at the end of evolution. As an outcome of the stochastic optimisation process, variability between duplicate runs for the same flight speed existed, by varying amounts, depending on the variable considered. However, for most variables, there was a significant convergence between duplicate runs when compared to differences between runs at different flight speeds. In other words, the variability was low enough to identify and discuss comparative adaptations:
3.2.1. - Power consumption (figure 5a, b).
The lowest mechanical power for horizontal FWF of our 500g-bodied UAV, plus the mass of wings and tail (which depended on individual morphology), approximated 15 W, and was achieved at intermediate flight speeds (10, 12 $m.s^{-1}$). At high speeds, i.e., 16 and 20 $m.s^{-1}$, the minimal power to sustain flight increased to almost 20 and 30 W, respectively. On the other hand, at low speeds, the minimal power attained by the most efficient individuals reached much higher values of 60 W at 8 $m.s^{-1}$ and 400 W at 6 $m.s^{-1}$. These results suggest a general U-shaped curve for power consumption across flight speeds, which is consistent with biological models and measurements of FWF power (Rayner, 1999). However, it should be noted that the curves in figure 5a are not directly comparable to biological power/speed curves, because biologists estimate power consumption at varying speeds for a same species, for which only kinematics vary, through the flapping behaviour of a given bird. On the other hand, we present results of (evolutionary) adaptation of both the morphology and kinematics to a given flight speed. Beyond the general U-shape trend, comparisons of absolute power values with biological literature is tempting but somewhat hazardous, as our power values depend on many model parameters that are inevitably only partly bio-mimetic, for example airfoil characteristics. Moreover, empirical measurements of mechanical power in bird species have been conducted on smaller bird species, because of wind tunnel size constraints. With these restrictions in mind, we still can refer to the results of Dial *et al* (1997), who measured a minimum 9 $W.kg^{-1}$ power (per kilogram of bird mass) in the magpie (*Pica pica*). Tobalske *et al* (2003) measured a minimum of 17 $W.kg^{-1}$ in the cockatiel (*Nymphicus hollandicus*), and 31 $W.kg^{-1}$ in the ringed-turtled dove (*Streptopelia risoria*). Our minimal value of 20 $W.kg^{-1}$ after mass correction (figure 5b) falls among those biological values. However, maximal measured mechanical power consumption is 54 $W.kg^{-1}$ in the dove (Tobalske *et al* 2003), a value that many of our optimised individuals exceed by a large amount, especially at low speed. Though the mass-specific power limit in birds is probably above the above-mentioned values, as 80 $W.kg^{-1}$ have been documented in some species during take-off (Askew *et al* 2001), it is clear that the power values we obtain at 6 $m.s^{-1}$ are unrealistic at 400W or 500 $W.kg^{-1}$. Technological considerations suggest that the mechanical power produced on a real UAV prototype would hardly exceed 200W. This rather optimistic figure is obtained assuming a single, 150 g state-of-the-art electric motor (e.g. ModelMotors 2820/8), absorbing 400 W of electric power, and a global 50% efficiency for the whole flapping mechanism. Two main hypotheses can explain the unreasonable energetic levels we obtain for slow flapping flight :

(i) Our UAV's DOFs are too restrictive compared to those of real birds, and constrain the possible kinematics so much that slow flight cannot be performed efficiently.
(ii) Our flight mechanics model does not take unsteady aerodynamics into account, and thus our

candidate individuals cannot use effects such as delayed stall to increase airfoil performance (Vogel 1994).

Since we obtain extreme power values only at low flight speed, and since unsteady aerodynamics are known to grow in relative importance at low flight speed, the second hypothesis is theoretically well grounded. As for the first hypothesis, it will be discussed later during the analysis of morphological and kinematic parameters (par. 3.2.5).

3.2.2. - Wing area (a_w) adaptation (figure 5c, d).

The optimal wing area emerging from evolution depended greatly on the flight speed: the general trend was that a_w decreased with an increasing flight speed. As the mass of the UAV's body remained 0.5 kg, the adaptation of wing area implied an increase of wing loading for higher flight speeds, a well known relationship for all flying objects, as there is a physical proportionality relationship between the natural flight speed and the square root of wing loading (e.g. Norberg 1990). At 6-8 $m.s^{-1}$, the mean a_w was near 0.3 m^2. At 10 and 12 $m.s^{-1}$, a_w decreased to approx. 0.2 and 0.15 m^2 respectively, i.e., to values approaching the natural wing areas of 500g-bodied birds such as Marsh-Harrier, Shearwater or Hooded Crow (Greenewalt 1962). At 16 and 20 $m.s^{-1}$, evolution converged to the minimum allowed wing area value of 0.1 m^2, indicating that selection strongly favoured highly loaded individuals at these high speeds, mimicking a "duck-like" adaptation. It should be noted that optimal individuals at the end of our evolutionary runs are necessarily "specialists" of the flight speed at which they were selected, which is different in natural bird species, whose characters (e.g. wing loading) probably result from selective compromises over the whole flight speed range they practice. Whether wing areas values selected here at a given flight speed would remain functional at other flight speeds, by changing the wing movement only, is an issue dealt with later (par. 3.5).

3.2.3. – Wing aspect ratio (λ_w) adaptation (figure5e, f).

Evolution yielded high aspect ratio values at almost all tested flight speeds. The maximal ratio value of 10 was reached by most individuals at 6, 8, 10, 12 and 16 $m.s^{-1}$. It is only at 20 $m.s^{-1}$ that an optimal aspect ratio averaging 8.5 was obtained. A λ_w lower than 7 was never retained at the end of the evolutionary runs. High aspect ratios have the beneficial effect of decreasing the induced drag (eq.14), and hence the forward thrust force that must be generated by flapping. However, for a given wing area, wings with a high aspect ratio have a lower chord and thus experience lower Reynolds number, which increases the airfoil friction drag (eq.15). Optimal aspect ratio should theoretically result from a compromise between these contradictory effects. The optimal value therefore depends on the flight speed, the wing area, and on the sensibility of the airfoil performance to Reynolds number. Moreover, in FWF, aspect ratio has other implications: more power will be needed to accelerate/decelerate a high aspect ratio wing, which has higher inertia around the shoulder joint. On the other hand, the flapping frequency would possibly be reduced with a wing with a high aspect ratio, as wing tip velocity induced by flapping will be increased by higher wingspans, thus generating higher thrust forces. In the present case, with the specific characteristics of our UAV in terms of mass, area, airfoil and flight speeds, it appears that the optimal λ_w is around 10 or more, which is above values observed for similarly-sized birds (7.7 on average; Norberg 1990). This difference has two main possible origins:

(i) our simulator did not take structural resistance into account, whereas a bird's fitness strongly depends on maintaining the integrity of its wing structure with a reasonable safety factor. In other words, depending on the material used, it is possible that high aspect ratio wings would bend or break during flapping at a high frequency. This is a first selective pressure towards low aspect ratio wings that is lacking in our study.

(ii) most importantly, we only selected our UAVs for forward flight. Thus no selective pressure was placed on manoeuvrability, on flight in obstructed areas (vegetation), or simply on wing folding for walks on the ground, which are factors that all favour the selection of lower aspect ratios in birds, at the expense of a slightly lower aerodynamic efficiency (Norberg 1990, 2002). Considering these limitations compared to natural conditions, it is not surprising that simulated evolution converged towards what can be considered as high aspect ratio "open space flyers", somewhat analogous to marine bird species that are almost 100% occupied at flying, like albatrosses and other *Diomedeidae* or *Procellaridae*.

3.2.4. – Flapping frequency (f) and Dihedral amplitude (amp_{di}) adaptation (figure 5g, h).

At intermediate speeds (10 and 12 $m.s^{-1}$), the flapping stroke frequency was about 3 Hz. This value is in the lower biological range (2-10 Hz at this mass, Norberg 1990), which is not surprising given the long, seabird-like wings of our UAV: with this morphology, sufficient thrust can be generated with a low flapping frequency (Norberg, 1990, p.177). As a point of comparison, the Kelp gull (*Larus dominicanus*) has a "natural" flapping frequency of 3.5 Hz (Pennycuick 1996), for mass and area (0.89 kg and 0.23 m^2) characteristics comparable to those of our intermediate speed UAV. However, this species has a slightly lower aspect ratio of approximately 7.5.

Slow and fast flight both implied higher frequency values (closer to 5 Hz on average), which contribute to explain the observed increase in power consumption at those speeds. Concerning the stroke amplitude, there was a less clear adaptive trend, with most individuals presenting a dihedral amplitude in the 25-45° range, with some increase at the lowest and highest flight speeds, contributing further to the increase in mechanical power. As a whole it seems that variations in both the stroke frequency and amplitude were implied in the adaptation of flapping kinematics to flight speed. However, both variables did not vary similarly: for example, only frequency increased from 12 to 16 $m.s^{-1}$ (with a slight decrease in stroke amplitude), whereas only amplitude increased from 16 to 20 $m.s^{-1}$, suggesting that these variables exhibit a rather complex adaptive landscape. These trends are unfortunately not easily comparable to intraspecific biological kinematic data because, in the present case, the wing area varied between flight speeds. Nevertheless, it should be mentioned that biological data show that flapping frequency depends on flight speed in some species, while it remains fairly constant in others (Tobalske and Dial 1996, Park *et al* 2001).

3.2.5. – Sweep amplitude (amp_{sw}) and offset (off_{sw}) adaptation (figure 5i, j).

The potential usefulness - or worthlessness - of an articulated wing in a FWF UAV is an interesting issue that has not been directly addressed previously. Of course birds and bats have elbow and wrist and use these DOFs in flight, with an amount depending on species and flight speed (Tobalske and Dial 1996, Park *et al* 2001, Tobalske *et al* 2003). From an adaptationist, functionalist point of view, this suggests that an articulated wing may be aerodynamically useful. On the other hand, other factors constrain the presence of wing articulations in birds and bats : first of all, the heredity of a vertebrate limb organisation plan is an historical, contingent constraint that questions the purely functional necessity of an articulated wing. For example, insect can fly efficiently without articulated wings. Moreover, an articulated wing may be beneficial to other functional aspects than forward flight, e.g., to increased manoeuvrability, and the simple biological necessity to fold wings on the ground represent a potentially strong natural selective pressure that is not necessarily relevant for an UAV. Hence it was interesting to test whether the presence of a wrist in our simulated wing was used by evolution for purely forward flight, and to quantify its possible beneficial effects.

Figure 5i shows that the sweep of the outer panel was used by almost all optimised individuals, at all flight speeds. Between 8 and 20 $m.s^{-1}$, the amplitude of the sweep did not vary much in a consistent manner, and averaged 25°. A different pattern appeared at 6 $m.s^{-1}$, as most individuals at this lowest speed used a much higher amount of sweep, attaining 60 to 80°. Fig 5j shows how the sweep was synchronised with the dihedral : the sweep (SW) tended to have a 0-25 % period offset compared to the dihedral (DI) which, according to equations (1) and (2), shows that a maximal sweep angle (i.e. adducted wing tips, minimal wingspan) was attained in the second half of the upstroke, whereas a zero sweep angle (i.e. fully extended wing) was attained in the second half of the downstroke. This is close to what is observed in birds, for which it has been usually reported that the maximal wingspan occurs at mid-downstroke, and minimal wingspan at mid-upstroke, which corresponds to a 25% value for off_{sw}.The amount of wing retraction was globally less than in real birds: given our UAV morphology (figure 1), the ratio of minimal to maximal wingspan was 0.95 for 25° of sweep, 0.75 for 60° and 0.58 for 80°. Birds for which this same "span ratio" variable has been measured in flight exhibit much lower values, usually below 0.5 (Tobalske and Dial 1996, Park *et al* 2001, Tobalske *et al* 2003). Moreover, most of these species (e.g. Barn Swallow [*Hirundo rustica*], Pigeon [*Columba livia*], Cockatiel [*Nymphicus hollandicus*]) have a tendency to more retract their wings at a higher speed, which is not observed in the present case. As already noted, this discrepancy can partly result from the fact that our UAV's size changed between flight speeds, hence the wing area adaptation need not be achieved through partial wing folding as in real birds. It is also important to note that the bird species thus investigated are far from the "seabird" morphotype, for which data on span ratio is lacking. We speculate from personal observations that wing adduction in gulls and akin species is less pronounced than in pigeons for example. There are also structural reasons why our UAV retracts its wings rather modestly compared to birds. First, the wrist in our UAV cannot be adducted, as only the wingtips can. This constraint *de facto* limits the span ratio to a minimal value of 0.5. Second, as wrists cannot move forward or backward relative to the body, we considered the possibility that a strong sweep of the external panel would separate the lift center of the wing from the center of gravity of the UAV, and hence cause pitch torques and instability issues. However, this hypothesis is partly refuted by the fact that individuals at 6 $m.s^{-1}$ succeed in using up to 83° of sweep. It remains that the limitations we put on wing retraction, suggested by anticipated constraints on prototype construction, might indeed partly cause the very high power consumptions we obtained at low speed. It could indeed prevent our UAV from exploring some of the wing movements a bird can achieve, which are especially refined at low speeds, as illustrated and discussed in the biological literature mentioned herein. Although the wing retraction possibilities of our UAV were modest compared to those of birds and

bats, it remains that the wrist sweep was almost systematically used, thus suggesting that it allowed the generation of more efficient aerodynamic forces. To test this idea further, we quantified the power gained from wrist movements (see par. 3.4).

3.2.6. – Shoulder and wrist incidence rotations (figure 5k, l, m, n, o, p).

The shoulder incidence position (SINC) determines the angle between IP and body, whereas the wrist incidence position (WINC) determines the angle between OP and IP. Hence the angle between OP and body results from the sum of SINC and WINC. Plots of reference angles versus flight speed (ref_{sinc}, ref_{winc}, figure 5k and 5l) show that IP tended to have a higher angle of incidence at low speed, but not OP: negative angles at the wrist tended to compensate the positive angles at the shoulder. Concerning the variation of the angle of incidence, its amplitude (amp_{sinc}, amp_{winc}, figure 5m, 5n) tended to increase with lower flight speeds, for both IP and OP, which is in agreement with kinematic data in birds which associate higher variations in the wing angle of attack with slow flights (Heddrick *et al* 2002). It is noteworthy that SINC and WINC values being relative angles between body and wing panels, they provide only an indirect information on the aerodynamic angle of attack, which depends on the incident air speed induced by the flapping stroke, and on possible changes in the body tilt angle (see par. 3.2.7). The offset between the incidence and dihedral oscillation (off_{sinc}, off_{winc}, figure 5o, 5p) converged to values approaching 25% on average. In other words, a maximal positive incidence was attained at mid-upstroke, while a minimal incidence (often negative angles between body and wing panels) was attained at mid-downstroke. As a direct outcome, this pattern tend to maintain the wing airfoil at low angles of attack throughout the stroke, thus maximizing the lift/drag ratio (figure 2).

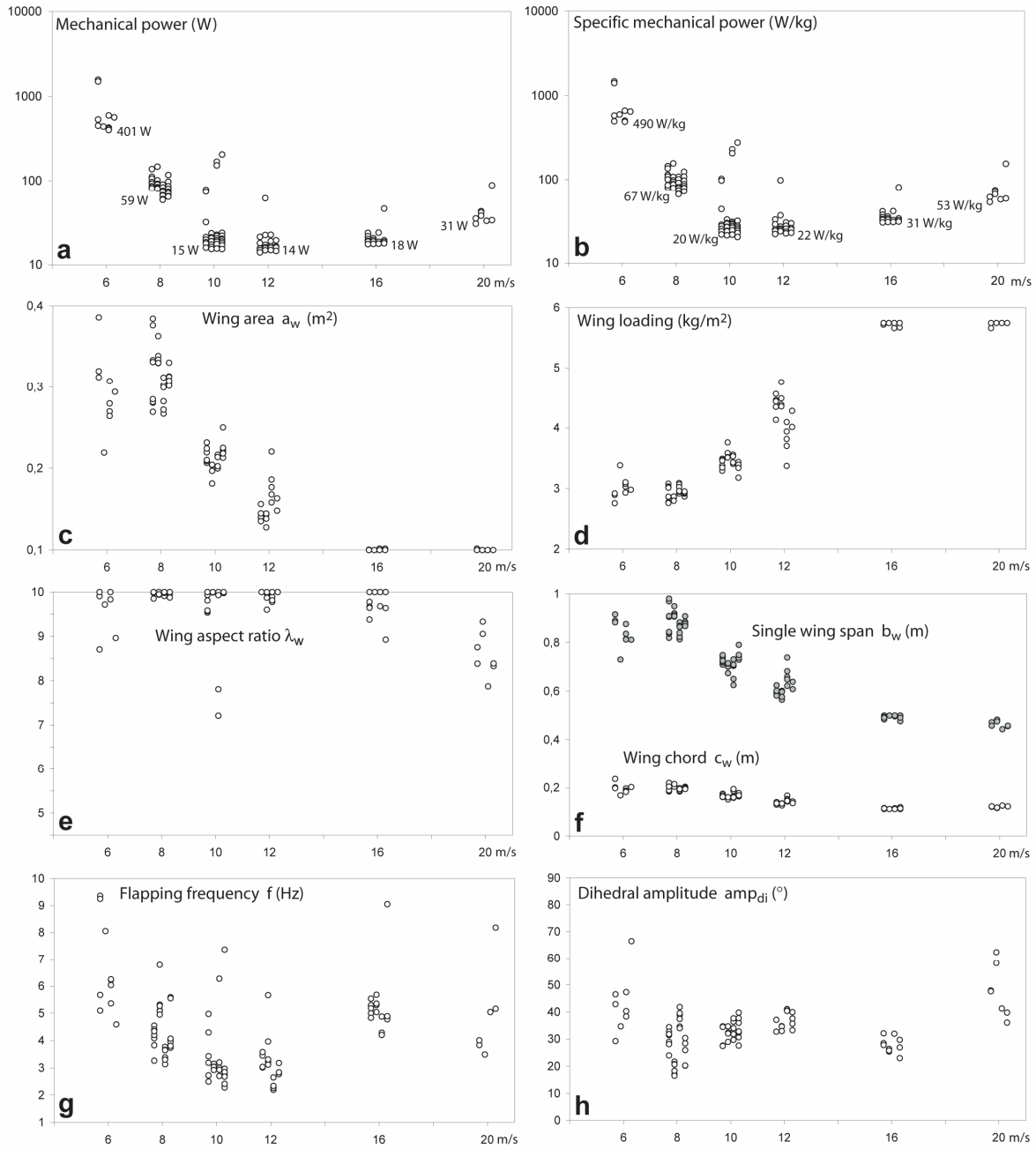


Figure 5: Results of morpho-kinematic adaptation to varying flight speed. The characters of the best individuals resulting from 4 duplicate evolutionary runs at each flight speed (6, 8, 10, 12, 16 and 20 m/s) are plotted (individuals generated during the same run are aligned vertically). For all genomic parameters, the vertical range of the graph equals the authorised search space during evolution.

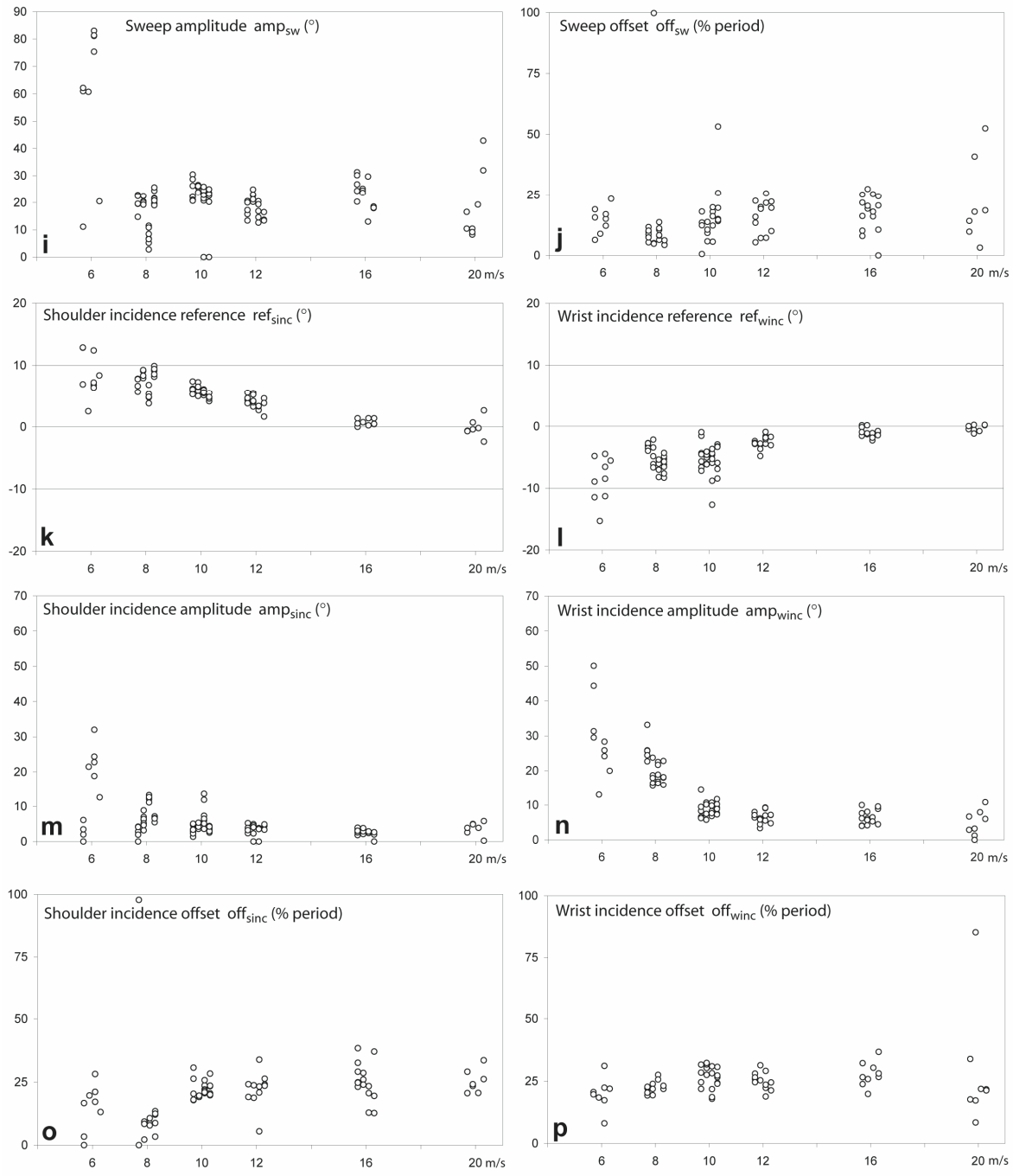


Figure 5 (continued).

3.2.7. – Analysis of aerodynamic forces in representative individuals.
To investigate the respective aerodynamic role of inner/outer panels during downstroke and upstroke throughout the flight speed range, we plotted the aerodynamic forces generated by IP and OP along the flight path for one individual. We chose the most power-saving individual at each flight speed. As a consequence of the dual-objective optimisation scheme, these individuals do not necessarily have the best performances in terms of horizontality of flight. On the contrary, turns out that the best individuals in terms of flight horizontality achieved their flight at the expense of power consumptions that were an order of magnitude higher, with much more variability between duplicate runs, than those of the six individuals selected here - clearly suggesting that they were much less aerodynamically efficient, and hence less interesting and representative of aerodynamic optimisation. As a complementary illustration, we also produced in-flight motion videos of these six individuals (available on http://animatlab.lip6.fr). Numerical values of all parameters for these individuals are presented in table 1.

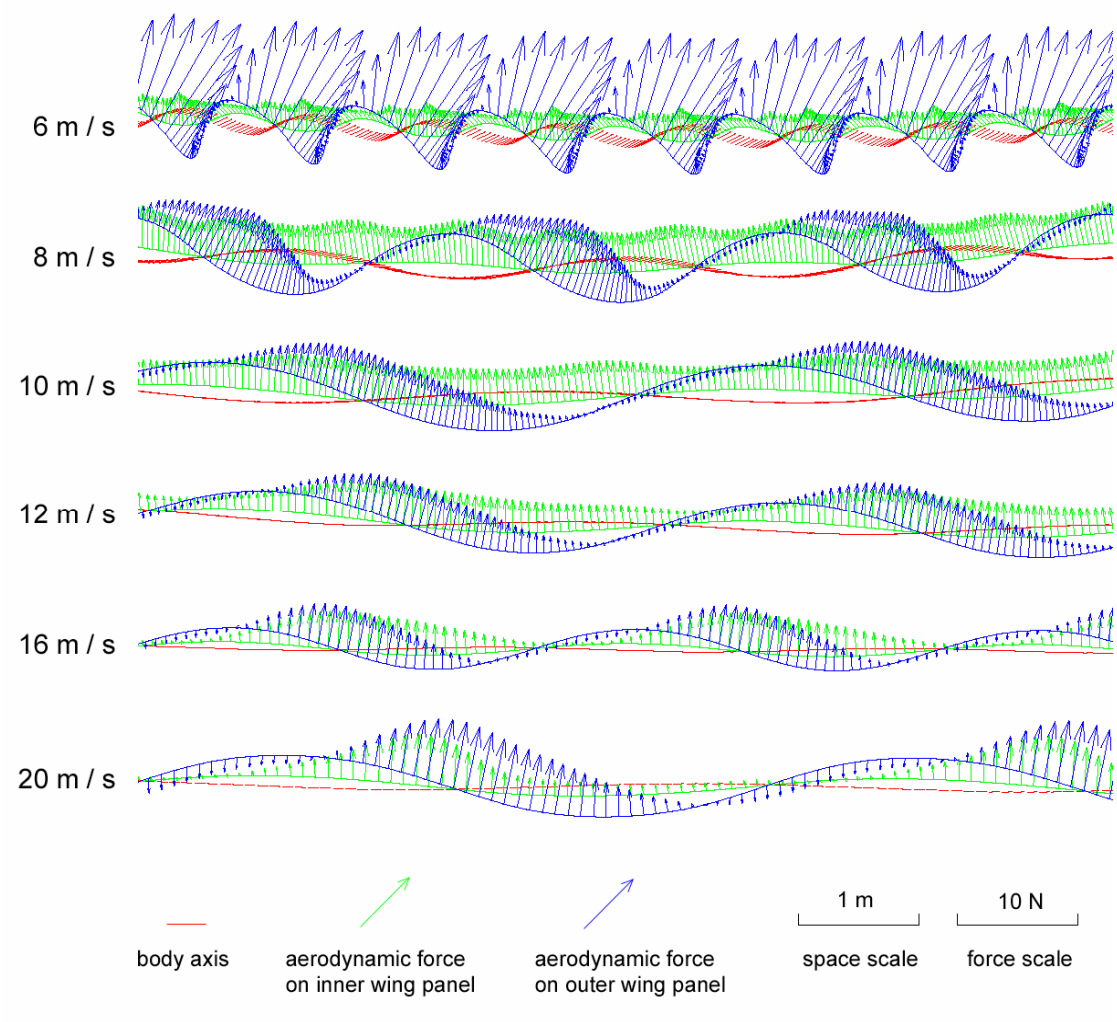


Figure 6: Aerodynamic forces on the wing panels of six optimised UAVs. The force vectors are summed over the "wing elements" (WELs, see "Flight Simulation" section) constituting a single (inner or outer) panel. The vectors' origins are on a line representing the trajectory of each panel's center of area. Tail and fuse forces are not represented for clarity.

Table 1: Parameters of the most power-saving individual at each flight speed.

Flight speed (m/s)	**6**	**8**	**10**	**12**	**16**	**20**
Fitness						
mean P (W)	401	59	15	14	18	31
max D (m)	1.11	1.95	1.98	1.77	1.80	1.48
Genome						
a_w (m^2)	0.264	0.299	0.225	0.145	0.100	0.100
λ_w	10.0	10.0	10.0	10.0	9.6	8.4
f (Hz)	6.25	3.30	2.41	3.03	4.83	4.01
amp_{di} (°)	40.3	34.0	32.8	32.7	27.9	47.7
amp_{sw} (°)	75.4	10.8	25.0	20.0	30.2	16.6
off_{sw} (%)	17.0	11.2	14.6	13.5	25.1	9.7
ref_{sinc} (°)	6.4	4.8	4.8	4.6	1.4	-0.7
amp_{sinc} (°)	18.8	11.1	4.3	3.2	3.1	3.9
off_{sinc} (%)	21.1	7.8	20.0	24.1	32.6	20.6
ref_{winc} (°)	-11.3	-5.4	-3.3	-2.8	-0.9	0.0
amp_{winc} (°)	24.2	16.3	7.3	7.1	6.2	6.7
off_{winc} (%)	22.3	25.6	27.2	24.5	23.7	33.9
Other information						
b_w (m)	0.81	0.87	0.75	0.60	0.49	0.46
c_w (m)	0.18	0.19	0.17	0.13	0.11	0.12
r_b (m)	0.05	0.05	0.05	0.04	0.03	0.03
total span (m)	1.68	1.78	1.55	1.24	1.01	0.95
m_b (kg)	0.500	0.500	0.500	0.500	0.500	0.500
m_w (kg)	0.272	0.327	0.213	0.110	0.063	0.063
m_t (kg)	0.048	0.058	0.038	0.020	0.011	0.011
total mass (kg)	0.820	0.885	0.751	0.630	0.574	0.574
wing loading (kg/m^2)	3.1	3.0	3.3	4.3	5.7	5.7
specific power (w/kg)	490	67	20	22	31	53

A first remark concerns the trajectory and position of the body. It followed an oscillating path, ascending during downstroke and descending during downstroke. More interestingly, the body axis took a significantly tilted position at low speed (up to approx. 30° at 6 $m.s^{-1}$), a tendency observed and measured with comparable amounts in birds (Tobalske and Dial 1996, Tobalske *et al* 2003).
Considering the forces generated by OP it appeared that the force generation was almost fully concentrated during the downstroke, regardless of the flight speed. These downstroke OP forces had both vertical (upward) and horizontal (forward) components, showing that OP had both a lifting and a propulsive function. At the beginning of the upstroke, weak lifting forces were also produced, but shifted to weak downward forces later in the upstroke. As a whole, OP was almost inactive during the upstroke at all flight speeds. Interestingly, at 6 $m.s^{-1}$, this asymmetric OP force pattern between downstroke and upstroke was achieved through a drastic variation in the relative airspeed : the simultaneous effects of a high wrist sweep - causing backward retraction of the outer wing part during upstroke - and the body tilt angle produced an almost zero OP horizontal speed during upstroke, while the same panel was greatly accelerated during downstroke. At higher flight speed, such a velocity difference was not observed with comparable amounts, thus suggesting that the absence of OP upstroke forces was mainly caused by placing the OP airfoil at a non-lifting angle of

attack.
Concerning IP, the force generation exhibited a different pattern. Forces were more evenly distributed among down- and upstroke, and included a lift (upward) and a drag (backward) component. This showed that IP, contrary to OP, did not usually participate to the propulsion of the UAV. This IP force pattern changed somewhat at the highest speeds, with a weakening of upstroke forces, with relatively stronger forces generated during the downstroke, and with a slight forward component of the generated force at mid-downstroke. As a whole, this IP force pattern at high speed was closer to the previously described OP force pattern.
The comparison of the mean force amplitude over the full stroke on OP and IP showed a clear OP domination at 6 $m.s^{-1}$, and a more even repartition over the wing span at other speeds. This is explained by the fact that the airflow is dominated by wing flapping at slow flight speeds (Hedrick *et al* 2002), and hence depends on the distance from the articulated dihedral joint.
Concerning the down-/upstroke repartition for the wing as a whole, more lifting forces and all propulsive forces were generated at the downstroke, at all flight speeds. This was mainly due to OP generating forces during downstroke only. However, according to what has been described above, this downstroke domination tend to be less obvious at intermediate speeds (8-12 $m.s^{-1}$) as lift produced by IP during upstroke took relatively higher importance. This is globally convergent with the results of Hedrick *et al* (2002) in the Dove and Cockatiel: these birds appear to have a more continuous lift generation at intermediate speeds (Hedrick *et al* 2002, Tobalske *et al* 2003). These significant variations in force generation modes across the flight speed range suggest that the limited kinematics of our UAV compared to real birds, which prevent very adducted upstrokes (e.g. "feathered" upstroke, Tobalske and Dial 1996), still leave room for efficient adaptation in the aerodynamic flight regime.

3.3. Flight stability and robustness of the "open-loop" kinematic control
Aside from pure energetic performance, the applicability of our optimised flapping flight kinematics to a real UAV prototype depends on its ability to generate a stable flight. Our UAV's wing movement control was truly "open loop" in the present work, i.e. the UAV had no information on its flight variables and was not able to change its kinematics to correct its flight trajectory. For this reason, we anticipated that the UAV's trajectory, even after kinematic optimisation, would necessarily diverge from horizontal flight after a short time, presumably because of pitch instability. Therefore, we *a priori* considered the optimised kinematics produced here as basic wing motion laws that would necessitate a supplementary, higher level "closed-loop" controller to achieve flight stability in our UAV, like the ones described in Mouret *et al* (submitted).
To verify this presumption, we extended to 60 seconds the flying time of the optimised individuals (table 1), instead of the 10 s. of regular evaluation flight, in order to study the type of instability that might thus occur. Results are presented as flight trajectories in figure 7. Surprisingly, the longitudinal (pitch axis) stability was much better than expected, as any strong vertical trajectory divergence after the first 10s of evaluation time was never observed. Instead, despite the symmetric flight kinematics, we observed some lateral instability, i.e. the UAV progressively engaged into a descending spiral turn. This spiral occurred after a longer time at high flight speed, individuals flying at 16 and 20 $m.s^{-1}$ being able to fly for 60 s. without being affected. Even during these relatively longer forward flights, pitch stability was observed throughout, the sub-horizontal trajectory being maintained for the whole minute (i.e. 920 and 1200m distances, respectively). Moreover, a few supplementary flight tests, with variable initial speeds on the same individual, demonstrated that some individuals were able to passively correct large flight speeds discrepancies. For example, the individual optimised for 20 $m.s^{-1}$ and launched at 6 $m.s^{-1}$ was able to return to its horizontal 20 $m.s^{-1}$ flight within a minute (figure 7). Unfortunately, the opposite test of launching the individual optimised for 6 $m.s^{-1}$ at 20 $m.s^{-1}$ was unsuccessful, as lateral instability soon occurred (figure 7).

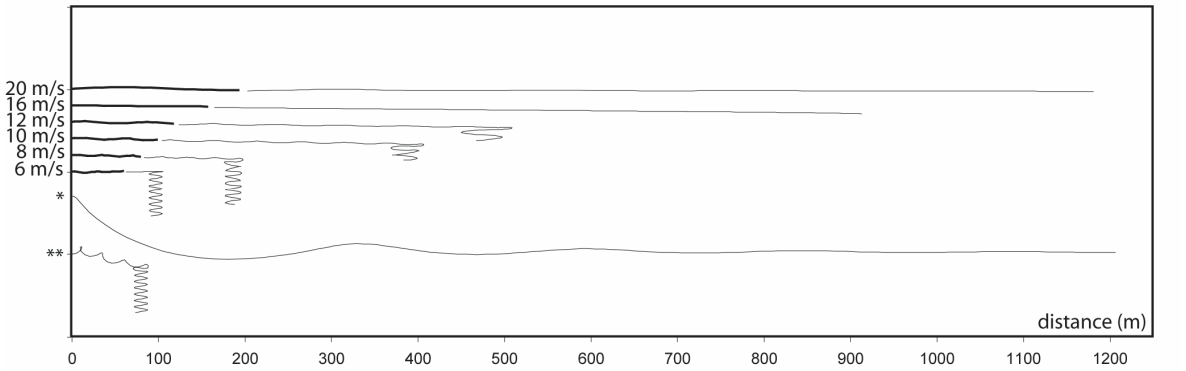


Figure 7: Flight trajectories (side views) of optimised UAVs during extended 60s. flights. Thick lines represent the 10s. initial evaluation flight.Thin lines represent the 50 s. flight prolongation, and demonstrate good longitudinal stability, but lesser lateral stability at low speed (spiral dive) of open-looped controlled FWF UAVs (see text). * : trajectory of the individual optimised for 20 m/s, launched at a lower speed (6 m/s) : horizontal flight at 20 m/s is recovered. ** : trajectory of the individual optimised for 6 m/s, launched at a higher speed (20 m/s) : horizontal flight is not achieved, due to lateral instability.

It is probable that our initial choice of placing the CG at 25% of wing chord, where lift forces apply, and of providing the morphology with a large tail helped the UAV to achieve pitch stability, at least in gliding flight. However, it remains that most non-optimised flapping kinematics during evolutionary exploration had as a first consequence to destabilise the UAV and to place it on an erratic flight trajectory. In this perspective, it is an interesting result that optimised UAVs were finally able to achieve a reasonable amount of passive pitch stability in flight. This suggests that the necessarily superimposed closed-loop controller mentioned above will eventually have relatively little corrective work to do to provide long-term longitudinal - and lateral – stability.

3.4. Effect of wrist lock : Usefulness of an articulated wing

Previously exposed results show that the wrist sweep was used by almost all optimised individuals. At 6 $m.s^{-1}$, the sweep was used at its maximum, and force plots (figure 6) suggest that it allowed to accelerate the wing tip during downstroke. However, the role of a relatively lower amount of sweep at higher speed was less obvious, and thus could appear *a priori* less important for flight performance. To test further the usefulness of a functional wrist, which would imply a more complex UAV prototype structure, we launched additional evolutionary runs in the same original conditions except that the wrist DOFs were disabled. We launched 2 runs at each flight speed with the wrist sweep locked (SW = 0°, i.e. wing fully extended), and 2 more runs at each flight speed with the wrist sweep and the wrist incidence locked (SW = 0° and WINC = 0°, i.e. wing fully extended and same incidence for IP and OP). After 50000 generations for each of these 24 evolutionary runs, we compared the power consumption for the best horizontal flappers, with the same original 2 m tolerance in the departure from reference trajectory.

Results (figure 8) show that disabling the wrist sweep implied a 8-79% increase in power consumption, depending on flight speed, except at 8 $m.s^{-1}$ where no power cost was observed. This surprising pattern at 8 $m.s^{-1}$ is partly explained by the fact that the original individual at 8 $m.s^{-1}$ used 11° of wrist sweep only, which affords the sweep-locked individual the possibility of attaining very similar kinematics. At other flight speeds, where the original sweep amplitude was 17 – 75°, locking the wrist sweep caused evolution to attain more costly kinematics, even when the original sweep amplitude was modest and its aerodynamic role unclear. For example, a 79% power increase at 12 $m.s^{-1}$ resulted from preventing the original 25° sweep amplitude. The sweep's beneficial effect tend to be less important at high speed. Globally, these results suggest that a functional wrist is really useful even in pure forward flight, at most flight speeds. Locking both the wrist sweep and the incidence rotations caused dramatic power increases at most speeds (10-252 %). Stressing further the crucial role of an articulated wing for FWF, this finding demonstrates that much power can be saved by allowing different incidence angles between rigid wing panels. Such variation in the incidence angle along the wing span is well known in birds and bats (Norberg 1990, p.118), though achieved through a flexible wing surface composed of feathers or membrane respectively, rather than through articulated rigid panels. It seems likely that we would get lower power consumption if we allowed a continuous twisting of our UAV wing, but this possibility generates additional technical complications if one

wants to maintain an efficient airfoil (e.g. DeLaurier 1999).

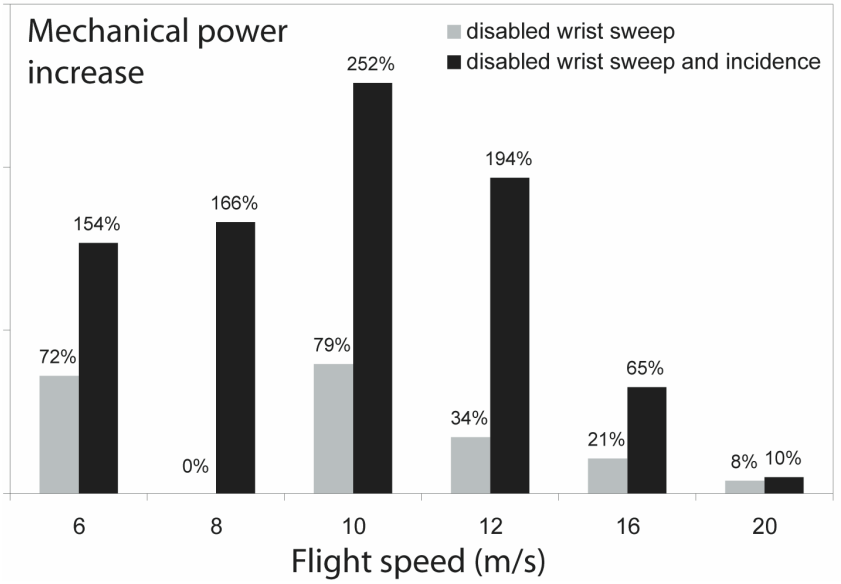


Figure 8: Effect of wrist lock on power consumption.

3.5. Evolution at fixed size and shape : Readaptation of kinematics to different flight speeds

As already explained, evolving the wing size, the shape and the kinematics at a given flight speed probably tends to produce "specialist" individuals, i.e. whose characters are well optimised for the considered flight speed, but non optimal at any other regime. To evaluate the degree of specialisation of the optimised individuals we previously obtained, we let the kinematics of some individuals evolve again at different flight speeds, while keeping the initial morphology (i.e. wing area and aspect ratio) unchanged. We performed 12 additional evolutionary runs: the morphology of the most power-saving individual at 6 $m.s^{-1}$ (A) was used to re-evolve the kinematics at 10 and 16 $m.s^{-1}$ (2 duplicate runs per flight speed). Similarly, the best morphology adapted for 10 $m.s^{-1}$ (B) was tested at 6 and 16 $m.s^{-1}$, while new kinematics for the best morphology at 16 $m.s^{-1}$ (C) were evolved at 6 and 10 $m.s^{-1}$. Results in terms of minimal power consumption are presented in figure 9.

A first finding was that each of the three morphologies remained the most power-saving solutions at their original flight speeds, which comforted the idea that separately evolving the kinematics on a constrained morphology has indeed a cost, as compared with the original choice of simultaneously evolving the wing morphology and kinematics. However, the performances of B at 6 $m.s^{-1}$ and A at 10 $m.s^{-1}$ were only a few watts above the original values. This is not surprising given that A and B had rather close wing area values (0.264 and 0.225 m2 respectively) and hence similar wing loadings (3.10 and 3.34 kg.m-2). Evolving the kinematics for A or B at 16 $m.s^{-1}$, as compared to the high-speed specialist (C, 0.1 m2 wing area), had a more obvious consequence since necessary power was approximately doubled. Aerodynamically, this cost is caused by the high drag forces exerted on a large wing at a high speed, as compared to a smaller wing. Biologically, this same energetic cost explains why gulls do not fly forward as fast as ducks. Most interesting was the fact that C was able to fly at 10 $m.s^{-1}$ with a 5-fold increased cost compared to B, and simply unable to fly at 6 $m.s^{-1}$: evolution failed to find any kinematics generating an horizontal flight. The best individual in the two runs lost 8 m of height in 10 s. This result expresses the difficulty to generate much lift with a small wing, and can be biologically illustrated by the tendency of ducks and similar species to refrain from flying slowly - though they can occasionally do it. As a whole, these tests showed that all individuals resulting from our optimisation runs had not attained the same degree of specialisation. Namely, whereas morphologies evolved initially for low to medium speed flight could adapt their kinematics to fly faster, the opposite was not true : high speed morphs could loose the ability to fly at lower flight speeds. Technically, this suggests that one would better choose A or B morphologies for a FWF UAV prototype rather than C, if the versatility in terms of flight speed range is an objective.

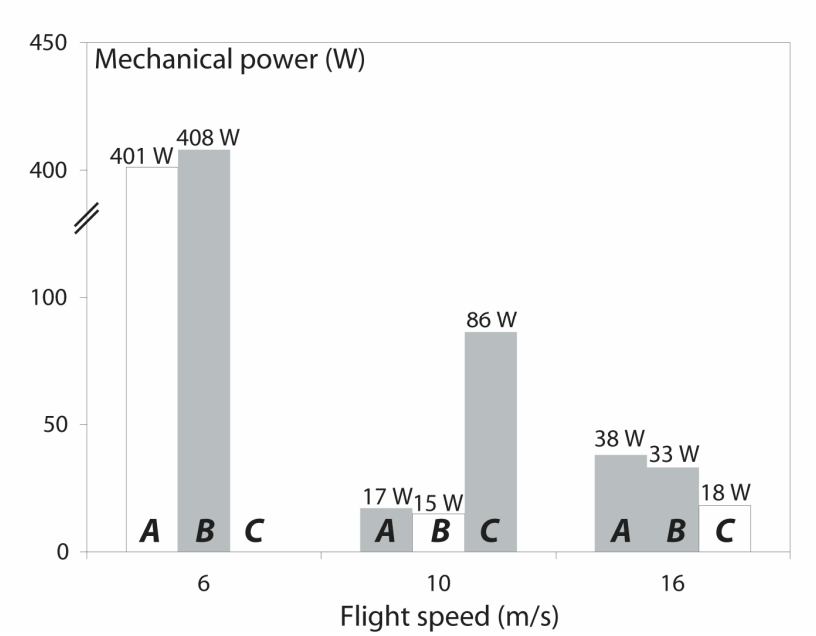


Figure 9: Re-adaptation of kinematics to different flight speeds. Three individual morphologies (A, B, C), initially optimised for different flight speed (6, 10 and 16 m/s respectively, white bars), were used to re-evolve new kinematics (without changing the morphology) at other speeds (gray bars). No experiment converged with the C morphology at 6m/s. See text for analysis of the consequences on power consumption.

4. Conclusions and perspectives

The proximal aim of the present work was to find optimal morphologies and kinematics for a flapping, articulated-wing mini-UAV. By using Artificial Evolution on a bird-like parameterised morphology, we were able to find morphological parameters and flapping stroke kinematics that achieve forward flight at medium speed (10-12 $m.s^{-1}$) for an estimated mechanical cost comparable to that of real birds: approximately 15W or 20 $W.kg^{-1}$. These parameters correspond to a 0.2 m^2, high aspect ratio wing, flapping around 3 Hz, looking like a gull in many aspects. Even more indicative is the fact that evolution also yielded angular variation laws for each articulation of this UAV (shoulder and wrist) capable of producing efficient lift and thrust forces throughout the flapping stroke with the chosen wing airfoil.

In a short-term perspective, we plan to test this morphology and its associated kinematics on a real UAV prototype, secured on a robotic arm (research in progress in the ROBUR project). We expect of course that real energy levels will differ somewhat from simulation predictions, and that some fine-tuning of kinematics will be necessary to correct for the unescapable simulation approximations.

Beyond medium-speed steady flight, the main interest of flapping wing flight for an UAV is the potential to vary its flight speed to a large amount, and to achieve special flight modes such as hovering or gliding. Though we could not cover the whole flight domain extensively in the present study, we already gained valuable information on the energetic consequences of speed variation. Flying at high speed (16-20 $m.s^{-1}$) could be achieved at rather low cost (20-30 W) in a small UAV (0.1m^2), but with a loss of flight abilities at lower speeds. On the other hand, we observed that it was possible to re-adapt the kinematics of a larger UAV (0.2 - 0.3 m^2), initially optimised for lower speed flight, to fly at high speed for a reasonable power cost (30 - 40W at 16 $m.s^{-1}$). This second strategy hence seems more promising in the perspective of a versatile real UAV.

Flying at low speed (6-8 $m.s^{-1}$) appeared very costly, and unrealistic power consumption were attained at 6 $m.s^{-1}$. Though the steady aerodynamics that were used in our simulation probably underestimated the lift production in slow flight, it is possible that our articulated morphology, as it stands, is inadequate for very slow or hovering flights, at least for a 1 kg UAV prototype. Even birds of this size hardly achieve hovering for more than a few seconds. At another end of the flight domain, the fact that our UAV was able to achieve gliding flight with a decent sinking speed (around -1 $m.s^{-1}$) is encouraging and suggests that bio-mimetic energy-saving behaviours, through soaring in ascending air, are within reach of our UAV's abilities.

Another original result brought by this work concerns the usefulness of an articulated wing for flapping flight: even without considering turning or manoeuvrability issues, we already learned that the movement of the wrist may have a strong influence on energy saving in forward flight, especially at medium-speed flight: compared to monolithic wings, allowing sweep and incidence rotations of no more than 20-25° at mid-span can drastically reduce the power consumption of a FWF UAV, provided that these supplementary movements are well synchronised with the flapping stroke. In other words, this finding suggests that implementing an articulated wing on a UAV prototype might be worth the implied technical complications.

Finally, from a methodological point of view, we mainly relied on biological data to validate/analyse our results. This was facilitated by the fact that we initially constrained the evolutionary search space to bio-mimetic morphologies and kinematics, in agreement with the rationale of the animat approach. Although more efficient flapping-wing machines may exist outside these boundaries, we are convinced that discovering and analysing such non-biomimetic FWF solutions would probably have been much more difficult, as we could not have relied on zoological records as an helpful documented referential for a preliminary validation of our results before experiments on real prototypes. Conversely, some of our simulation results should be useful to biologists. Although being not as reliable as real-world measures, Artificial Evolution coupled with environment simulation may be considered as a very valuable, fully controlled experimental framework serving to test the effect of all sorts of constraints - physical, historical or developmental - on the course and outcome of adaptation of all sorts of biological characters. Such an evolutionary modelling approach should be especially useful for functional morphologists and biomechanicists (de Margerie *et al* 2005). As illustrated by the present work on FWF, many experiments that would be impossible in a wind tunnel may be relatively easily reproduced in simulation, and yield to original results, such as:

- "flapping wing flight without the brain": our flight experiments with purely open-looped kinematics demonstrate that flapping a wing efficiently does not necessarily compromise the passive gliding-flight stability, nor does a short forward flight necessarily require any stabilizing neural control. This kind of result, possibly reinforced by experiments on variable tail areas for example, could provide useful information on the degree of active stabilisation the extant birds need to achieve in flight, and also produce valuable arguments on the morphological and neurological prerequisites for the emergence of flight in vertebrates.
- "flapping wing flight with a stiff wing": we were able to independently lock some degrees of freedom in our morphology, which is hardly conceivable on a real bird in a wind tunnel, and to evaluate the consequences on the energetic cost of flapping flight after letting the UAV readapt its kinematics. It was thus demonstrated that even a limited sweep movement of the outer wing part can save a significant part of power consumption, and that the ability to vary the incidence angle along wing span is a crucial feature of FWF at this size.

We are convinced that, as well as roboticists may draw helpful inspiration from zoological records and Darwinian evolution (Meyer and Guillot, in press), biologists interested in adaptation at a high integration level, such as organismal biologists or ecologists, can find new lines of evidence by using Artificial Evolution on modelled organisms. Although the structure of our study - notably the type of Artificial Evolution experiments that were conducted and the way the results were analysed - was primarily aimed at helping designing an UAV, and not at yielding biologically relevant findings, we hope the present work still participates at demonstrating these reciprocating interests.

Acknowledgement
The authors thank T. Druot for helpfull advice on using and improving the FMFAW flight simulator. This study was partly funded by the post-doctoral fellowship of the french “Direction Generale de l'Armement” (DGA) for E. de Margerie.

Reference

Angeli A, Filliat D, Doncieux S and Meyer J-A 2006 2D simultaneous localization and mapping for micro aerial vehicles. In *Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference*

Askew G N, Marsh R L and Ellington C P 2001 The mechanical power output of the flight muscles of blue-breasted quail (Coturnix chinensis) during take-off. *J. Exp. Biol.* **204** 3601-3619.

Barate R, Doncieux S, and Meyer J-A 2006 Design of a bio-inspired controller for dynamic soaring in a simulated UAV. *Bioinspiration & Biomimetics* **1** 76-88

de Margerie E, Tafforeau P and Rakotomanana L 2006 In silico evolution of functional morphology: a test on bone tissue biomechanics. *J. R. Soc. Interface* **3** 679-687

Deb K, Mohan M and Mishra S 2005 Evaluating the epsilon-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation* **13** 501-525

DeLaurier J 1999 The development and testing of a full-scale piloted ornithopter. *Canadian Aeronautics and Space Journal* **45** 72-82

Dial K P, Biewener A A , Tobalske B W and Warrick D R 1997 Mechanical power output of bird flight. *Nature* **390** 67-70

Doncieux S, Mouret J, Muratet L, and Meyer J-A 2004 The ROBUR project: towards an autonomous flapping-wing animat. *In Proceedings of the Journées MicroDrones, Toulouse*

Druot T 2004 Technical report on the implementation and validation of a flight mechanics simulator for flapping articulated wings. Available on http://animatlab.lip6.fr

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

Greenewalt C H 1962 Dimensional relationships for flying animals. *Smithson. Misc. Collect.* **144** 1-46

Greenewalt C H 1975 The flight of birds. *Trans. American Philosophical Society* **65** 1-67

Hedrick T L, Tobalske B W and Biewener A A 2002 Estimates of circulation and gait change based on three-dimensional kinematic analysis of flight in cockatiels (*Nymphicus hollandicus*) and ringed turtle-doves (*Streptopelia risoria*). *J. Exp. Biol.* **205** 1389-1409

Meyer J-A and Guillot A In press *Biologically-inspired robots*. In Siciliano B and Khatib O editors, *Handbook of Robotics*. Springer-Verlag

Mouret J-B, Doncieux S, Druot T and Meyer J-A (submitted). Evolution of closed-loop neuro-controllers for flapping-wing animats

Muratet L, Doncieux S, Brière Y and Meyer J-A 2005 A contribution to vision-based autonomous helicopter flight in urban environments. *Robotics and Autonomous Systems*, **50** 195-209

Norberg U M 1990 *Vertebrate flight*. Springer, Berlin

Norberg U M 2002 Structure, form and function of flight in engineering and the living world. *J. Morphol.* **252** 52-81

Park K J, Rosen M and Hedenström A 2001 Flight kinematics of the barn swallow (*Hirundo rustica*) over a wide range of speeds in a wind tunnel. *J. Exp. Biol.* **204** 2741-2750

Pennycuick C J 1996 Wingbeat frequency of birds in steady cruising flight: New data and improved predictions. *J. Exp. Biol.* **199** 1613-1618

Rakotomamonjy T, Le Moing T and Ouladsine M 2004 Simulation model of a flapping wing micro air vehicle. *EMAV 2004 Conference, Braunschweig, Germany*

Rayner J M V 1999 Estimating power curves of flying vertebrates. *J. Exp. Biol.* **202** 3449-3461

Salles R and Schiele A 2004 Analysis of flapping-wing robots for planetary exploration. An evolutionary approach. *Proceedings of the 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation ASTRA 2004*

Selig M 1997 *Summary of low speed airfoil data – Vol. 2*. SoarTech Aero Publications, Virginia Beach, VA, USA. www.ae.uiuc.edu/m-selig/uiuc_lsat.html

Shim S Y, Kim S J and Kim CH 2004 Evolving flying creatures with path-following behaviour. *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*

Smith R 2006. Open Dynamics Engine - www.ode.org

Tobalske B W and Dial K P 1996 Flight kinematics of blacked-billed magpies and pigeons over a wide range of speeds. *J. Exp. Biol.* **199** 263-280

Tobalske B W, Hedrick T L, Dial K P and Biewener A A 2003 Comparative power curves in bird flight. *Nature* **421** 363-366

van Breugel F and Lipson H 2005 Evolving buildable flapping ornithopters. *GECCO 2005*

Van den Berg C and Rayner J M V 1995 The moment of inertia of bird wings and the inertial power requirement for flapping flight. *J. Exp. Biol.* **198** 1655-1664

Vogel S 1994 *Life in moving fluids*. Princeton, NJ: Princeton University Press

Withers P C 1981 An aerodynamic analysis of bird wings as fixed airfoils. *J. Exp. Biol.* **90** 143-162

A.3 DESIGN OF A BIO-INSPIRED CONTROLLER FOR DYNAMIC SOARING IN A SIMULATED UAV

Barate, R., Doncieux, S., and Meyer, J.-A. (2006). *Design of a bio-inspired controller for dynamic soaring in a simulated UAV*. **Bioinspir. Biomim.**, 1 :76-88.

Travaux réalisés dans le cadre du stage de master de Renaud Barate dont j'étais l'encadrant scientifique.

INSTITUTE OF PHYSICS PUBLISHING BIOINSPIRATION & BIOMIMETICS
Bioinsp. Biomim. **1** (2006) 1–13 UNCORRECTED PROOF

Design of a bio-inspired controller for dynamic soaring in a simulated UAV

Renaud Barate, Stéphane Doncieux and Jean-Arcady Meyer

Université Pierre et Marie Curie—Paris 6, UMR 7606, AnimatLab/LIP6, 8 rue du Capitaine Scott, Paris 75015, France

E-mail: Renaud.Barate@ensta.fr, Stephane.Doncieux@lip6.fr and Jean-Arcady.Meyer@lip6.fr

Received 16 June 2006
Accepted for publication 6 October 2006
Published DD MMM 2006
Online at stacks.iop.org/BB/1/1

Abstract
This paper is inspired by the way birds like albatrosses are able to exploit wind gradients at the surface of oceans for staying aloft during very long periods while minimizing their energy expenditure. The corresponding behaviour has been partially reproduced here via a set of Takagi–Sugeno–Kang (TSK) fuzzy rules controlling a simulated glider. First, such rules were hand-designed. Then, the rules were optimized with an evolutionary algorithm that improved their efficiency at coping with challenging conditions. Finally, the robustness properties of the generated controller were assessed with a view to its applicability to a real platform.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Although designing unmanned aerial vehicles (UAV) represents a great challenge for engineers because such platforms have to face complex and dynamic environments, numerous research efforts have been devoted to this endeavour and it has now become relatively easy to build autonomous flying robots, even with off-the-shelf components [1]. But the decisional autonomy of these platforms is still limited to a capacity to follow a given trajectory defined by a set of GPS waypoints. For lots of applications, a higher degree of autonomy is required, that would make it possible to give a robot watching over forest fires, for instance, a simple order like 'stay over this area'. This would entail providing the robot with a map of its environment and with the capacity of self-localizing within this map, but would avoid the necessity of pre-defining a series of waypoints to be followed. More importantly, such an approach would afford the robot the possibility of autonomously and opportunistically choosing its trajectory, so as to spare its energy consumption, a major issue for current UAV technology. Indeed, small-scale fixed-wing UAV—with wingspans less than 2 m—currently exhibit an energy autonomy of 1 or 2 h because they are committed to permanent use of their motors. If one could find a way to exploit the environment more efficiently, this autonomy could be widely expanded. To this end, nature is a good source of inspiration as lots of energy-saving strategies are exploited by birds.

Indeed, some birds are able to fly for very long periods. Albatrosses, for instance, can spend days in the air without touching the ground. They can reach such a performance because they are able to remain aloft without flapping their wings, mainly by exploiting wind gradients, thus saving precious energy. Likewise, birds of prey like eagles gain altitude by turning inside thermals, and mountain birds like jackdaws exploit slope-winds to fly without flapping their wings. All these behaviours have been observed by biologists [2–5] and exploited by humans to efficiently pilot gliders [6]. However, in the latter case, the glider is permanently controlled by the human pilot, and the implementation of energy-saving behaviours in autonomous flying robots has scarcely been attempted yet, with the notable exception of a recent work [7] on the exploitation of thermals. The corresponding robot flies until a thermal is found, and then remains inside it by turning round, thus gaining altitude.

This strategy is only one of those that can be deduced from animal behaviours. It belongs to the *static soaring* category because it exploits the air flowing upwards: to gain altitude, the robot 'just' needs to remain inside the thermal. Another strategy, that belongs to the *dynamic soaring* category, exploits wind gradients. Here, there is no upward flow but a horizontal wind, the speed of which varies with altitude, as may be

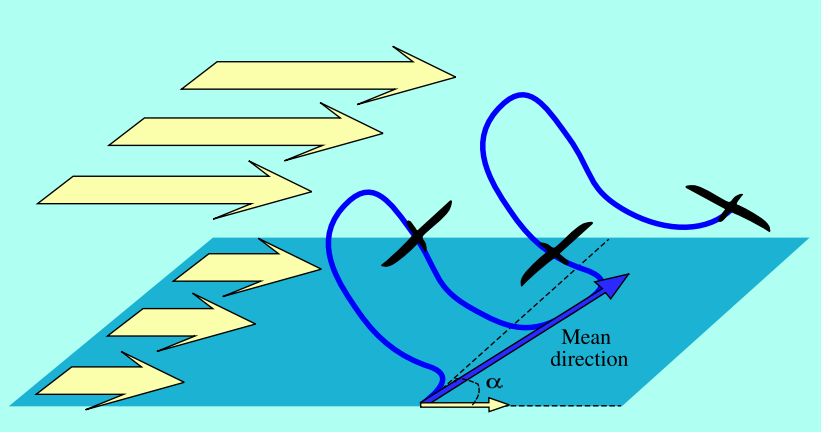


Figure 1. Trajectory used by the albatross for dynamic soaring. At the highest point of the trajectory, the albatros turns back to the wind and starts diving to gain speed. At the lowest point, the albatros sharply turns to face the wind and exploits the speed gained during the dive to gain altitude. At the end of this phase, the bird has reached the same altitude as that of its starting point. On average, the birds fly at a certain angle α with respect to the wind.

the case above the ocean or above a mountain slope. To exploit such circumstances and save energy, a bird or a glider must follow a cyclic trajectory, starting from a relatively high altitude and flying down, wind in the back. Before hitting the water or the ground, it must turn back sharply to face the wind, in order to exploit the speed attained during the dive and so gain altitude. When its relative speed becomes too slow, it must turn back again, wind in the back and start another cycle. When this strategy is tightly followed, the bird or glider extracts energy from the wind gradient, and is thus able to reach the same altitude it started from (figure 1).

Dynamic soaring has been studied from theoretical and empirical perspectives [8–10]. Experienced pilots have recently proved to be able to exploit dynamic soaring principles on a small-scale glider [8]. Open loop controllers exhibiting an optimal trajectory have been designed in simulation [11]. Preliminary work on closed loop control of this behaviour revealed the difficulty of their implementation [12].

The objective of this research effort is to design such a closed loop controller. Experiments reported here have been done in simulation only, but with a view to assess the difficulty of implementation on a real platform.

This work is part of the ROBUR project that aims at building an artificial bird [13] within the framework of the *animat approach*, which draws inspiration from animal behaviour for the design of artificial autonomous agents able to 'survive' or fulfil their mission in changing environments [14]. Ultimately, this project should produce a flapping-wing UAV able to take off, to land, to explore its environment and to manage its energy resources by exploiting aerology conditions or by getting back to a refilling station when required. The work reported here is the first step in this long journey.

2. Material and methods

2.1. Flight model

In order to test different controllers designed for dynamic soaring, we developed a specific flight simulator with a realistic aerodynamic model. This simulator is based on a perception-action control loop with a time step of 40 ms, which means that 25 times per second the controller provides orders to the glider's actuators, according to the current state of its sensors. Wings are made of two panels with different dihedral angles—i.e. 2° for the internal panel and 6° for the external—to improve the stability of the glider.

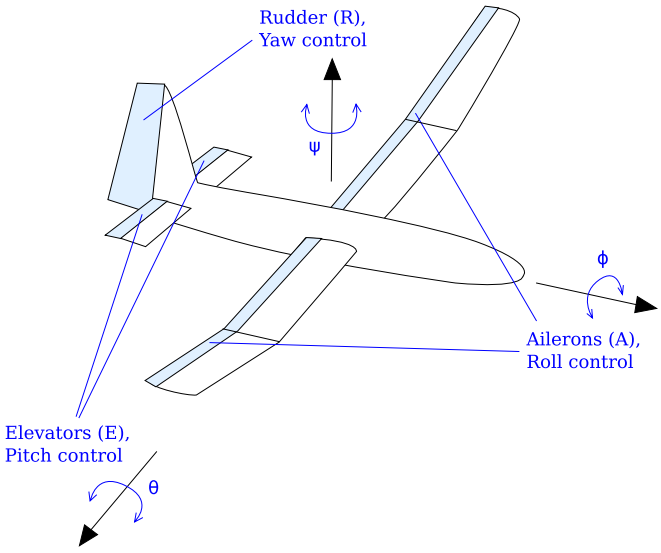


Figure 2. The different controls of the simulated glider: ailerons, elevators and rudder.

The aerodynamic model integrates the forces exerted on the two panels that constitute a wing, and on the two perpendicular surfaces that characterize the tail (figure 2). These forces depend on the characteristic curves that describe how the lift (CZ) and drag (CX) coefficients vary with a panel's angle of attack [15]. The aerodynamic force exerted on each panel is calculated according to the following formula [16]:

$$\vec{F_A} = \begin{pmatrix} -CX \\ 0 \\ -CZ \end{pmatrix} \frac{\rho S {V_A}^2}{2}$$

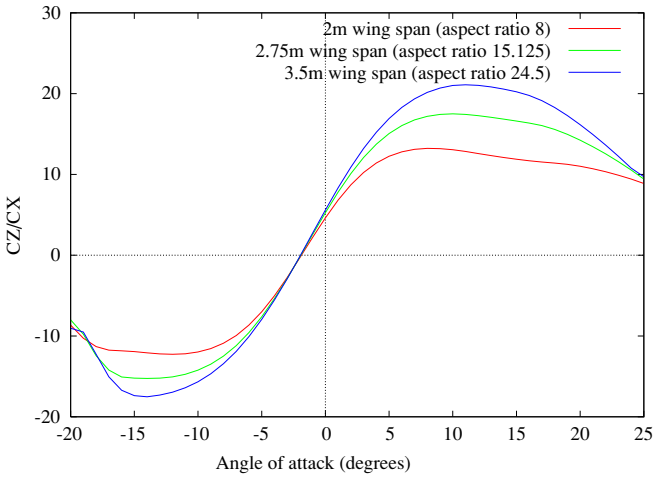


Figure 3. Variation of the lift/drag factor (CZ/CX) according to the angle of attack for the whole glider. Aspect ratios are taken into account in the calculation of the characteristic curves of each panel. The total wing area is held constant at 0.5 m^2, no sideslip effect is present here.

$\vec{F_A}$ being the aerodynamic force exerted on a given panel, ρ the air density, S the area of the panel and V_A the airspeed.

Aerodynamic forces on the body part of the glider are neglected here. The independent calculations of the forces on each panel allow the model to take implicitly into account some aerodynamic effects such as induced drag, environmental vortices and wind gradients. Indeed, in a wind gradient, the airspeed of each panel will be different, and the resulting aerodynamic forces and momentums will reflect these differences. The characteristic curves of the panels are calculated to take into account the panel's aspect ratio. Figure 3 shows the resulting lift/drag ratio when the independent forces are integrated to calculate the aerodynamic force applied on the whole glider. However, it should be noted that the turbulences and vortices generated by the glider's geometry are not taken explicitly into account. Aerodynamic ground effect is also not modelled here. Although this choice can be discussed, we consider that the short period spent near the ground, and the high bank angle at this point of the trajectory, allow us to safely ignore this effect.

These aerodynamic forces are then integrated using classical solid body equations to determine the global linear and angular accelerations of the glider. Its weight is added to these forces and the system's state (position and velocity) is updated at each time step.

In this simplified aerodynamic model, the controls of the glider are assumed to modify the angle of attack of its panels. Each swivelling control surface is thus characterized by an efficiency coefficient k_{si} with $i = 0$ for the ailerons, $i = 1$ for the elevator and $i = 2$ for the rudder. This coefficient is roughly equivalent to the proportion of the panel's surface it occupies. We chose:

- $k_{s0} = -0.12$ (it has the same value for the internal and the external parts of a wing)
- $k_{s1} = -0.15$
- $k_{s2} = -0.5$.

The amplitude of the controller's action on a particular control surface is represented by a value d_i ($d_0 = A$ corresponds to the aileron command, $d_1 = E$ is the elevator command and $d_2 = R$ is the rudder command, as explained later on) comprised between -1 and $+1$, which is assumed to modify the angle of attack α_{Ri} of the corresponding panel according to the formula: $\alpha_{Mi} = \alpha_{Ri} + d_i * k_{si}$, where α_{Mi} is the modified value.

Hopefully, this procedure provides a realistic enough simulation to make it possible to re-use the controllers thus generated on a real motorized glider. Be that as it may, although this simulation is based on a motorized glider geometry, the corresponding propulsion capacities are never used, and only the wind serves to move the UAV. Its physical characteristics are accordingly set to:

- Mass: 3.6 kg
- Length: 1.5 m
- Wing span: 2.75 m (variable in the experiment described in subsection 3.2)
- Total wing area: 0.5 m^2 (70% for the internal panel, 30% for the external panel)
- Wings sweep angle: 0°
- Wings dihedral angle: 2° for the internal panel and 6° for the external panel
- Wings twist angle: 2.5°.

More details about the mass characteristics of the glider are presented in appendix A.

2.2. Wind model

The vertical wind gradient that may be used for dynamic soaring above the ocean stretches over a thin layer (a few meters) in which the wind is slowed down by the interactions with the water surface. Observations made for winds of low to medium speed indicate that this gradient may be represented as a logarithmic function [17]. More precisely, we used the function described in [9] which defines the wind speed V_W at altitude h as:

$$V_W = V_{W_{\mathrm{REF}}} \frac{\log(h/H_0)}{\log(H_{\mathrm{REF}}/H_0)}$$

where H_{REF} is the reference altitude, i.e., 10 m in this application. H_0 is the altitude at which the wind speed equals zero, here it is set to 0.03 m. $V_{W_{\mathrm{REF}}}$ is the wind speed at reference altitude. Its default value is 20 m s^{-1}. This speed will occasionally be changed in some experiments described later.

The wind gradient described by this function is presented in figure 4.

2.3. Sensors and controls

The sensory system of birds has not been thoroughly studied yet, but some relevant observations have been made [18]. Briefly, birds possess the same kind of visual receptors as humans (rods and cones) but they can discriminate colours better, see ultra-violet light, and perceive light polarization. They also have magnetic receptors, acting like compasses.

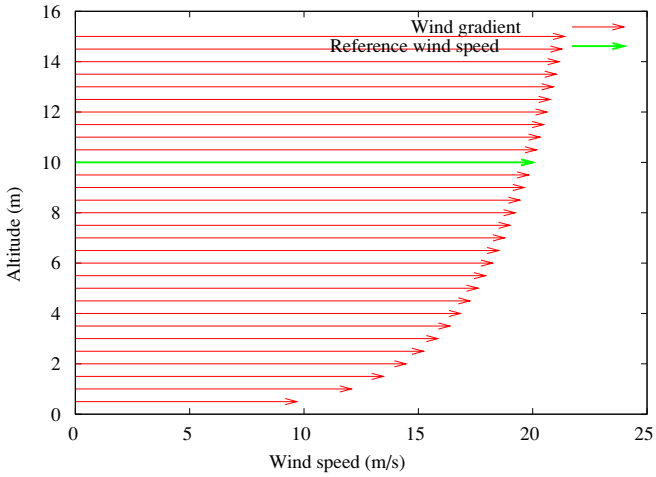


Figure 4. Wind-speed gradient used in the model.

Their tactile perception allows them to precisely sense pressure, pressure change and pressure change acceleration, thus affording important information about air speed and air acceleration.

Nevertheless, because a long-term objective of this work is to apply it to a real glider, sensors as complex as those of birds were not used here. Instead, we relied on more classical sensors to get the data necessary for flight control: an altimeter to get the glider's altitude, and an inertial measurement unit to get orientation angles. The variables measured by these sensors were:

- z, glider's altitude
- ψ, glider's heading with respect to the wind
- θ, glider's pitch angle with respect to an horizontal plane
- φ, glider's bank angle with respect to an horizontal plane.

We first considered that these sensors were perfect, in the sense that they were expected to provide exact values in real time. Later on in this paper, the influence of noisy sensors on the generated behaviour will be studied.

To control the glider, classical effectors were used, i.e., a pair of ailerons, a pair of elevators and a rudder (figure 2). As explained above, the corresponding variables were:

- A, aileron command comprised between -1 (maximum bank to the right) and +1 (maximum bank to the left);
- E, elevator command comprised between -1 (maximum upward) and +1 (maximum downward);
- R, rudder command comprised between -1 (maximum on the right) and +1 (maximum on the left).

2.4. TSK fuzzy rules

Controllers based on fuzzy rules have been successfully used for the control of several UAV systems [19, 20]. Their main advantage is to call upon explicit rules, unlike neural networks whose inner workings may be difficult to decipher. They are particularly well suited for dealing with incomplete or uncertain data. The corresponding rules can be hand-designed by a human, or automatically generated by an evolutionary algorithm.

The principle of such fuzzy controllers consists in designing a set of rules that associate a particular action to a given sensory state. Such rules are made up with a condition part and an action part. The former may refer to fuzzy sets associated with each input, e.g., 'IF z is low AND $\dot{z}$ is greatly negative'. The sum of the membership values of each input variable gives the strength of the rule. The latter may be fuzzy or not. Rules with fuzzy actions are called Mamdani rules [21], and sometimes require a tricky defuzzification procedure to generate the orders to be send to the system's actuators. Instead, in this work, we used Takagi-Sugeno-Kang (TSK) rules [22] for which the consequent of a rule is not a fuzzy set but a linear combination of the input variables. This method allows faster calculations than classical defuzzyfication methods. Thus, a typical TSK fuzzy rule would be for example 'IF z is low AND $\dot{z}$ is greatly negative THEN $E = 0.25 + 0.5z - 0.25\dot{z}$'.

It is also simple to make TSK fuzzy rules evolve via an evolutionary algorithm, as demonstrated, for instance, with the control of a real helicopter [23]. Fuzzy set limits and rule coefficients being simple real numbers, it is easy to define the corresponding cross-over and mutation operators, a point that will be touched on later in this section.

2.5. The controller's bio-inspired design

In a first stage, we hand-designed a reference controller with rules inspired by observations of real albatrosses. This way, we checked that a TSK controller is well suited for this problem and we set a base for further evolutionary improvements.

In his 1982 observations [2], Pennycuick described the dynamic soaring behaviour in these terms:

> A bird soaring across wind, along a wave, would from time to time turn into wind and pull up sharply, then turn at the top of the climb and glide off again across wind. The turns initiating and completing this manoeuvre were often abrupt and very steeply banked, up to 70°. This behaviour was seen in all the albatross species, and also in the giant petrel and white-chinned petrel.

Consequently, we decided to decompose the dynamic soaring behaviour of our controller into two phases:

- an ascending phase in which the bird faces the wind and gains altitude,
- a descending phase in which the bird is back to the wind and dives to gain speed.

More precisely, we wrote two sets of rules: one for turning towards the wind origin when the glider is low (ascending phase), and one for turning back to the wind when it is high (descending phase). The corresponding fuzzy sets were defined by ramp-type membership functions (figure 5).

The turns defined by these two sets of rules need to be very sharp, therefore they were triggered using the elevators with a very high bank angle. The rudder was used to prevent the glider from plunging right down to the sea. As for the ailerons, they were used to maintain the bank angle. With

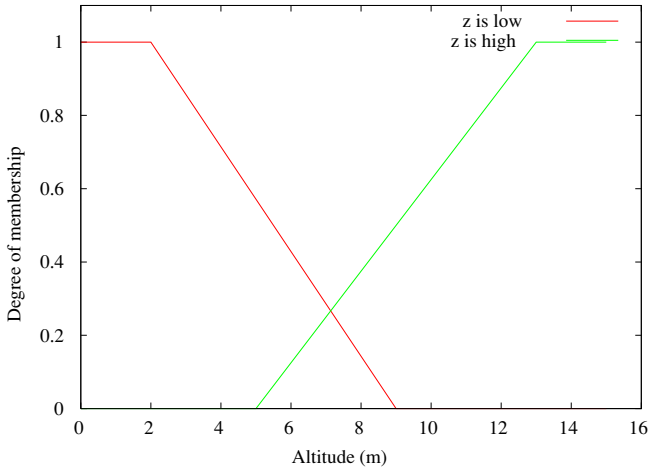


Figure 5. Fuzzy sets defined on variable z (altitude). Two sets are defined, respectively for high and low altitudes, the corresponding degree of membership progressively changing according to the glider's altitude. In other words, when the latter is lesser than 5 m, the system's altitude has more chances to be classified as 'low' than as 'high'.

these principles, we wrote the first rules for the reference controller. Then we empirically improved the values defining the fuzzy sets, as well as other parameters of the rules, until we obtained an improved and coherent behaviour. Here are the details of the three hand-designed rules for turning back to the wind:

- When the glider is high, the elevators must be raised in order to trigger a sharp descent, but they also have to control the pitch angle to prevent the glider from diving too quickly. They also must get lowered at the end of the turn, when the glider is almost back to the wind. Therefore, the first rule is 'If z is high, $E = -0.6 + 0.01\theta + 0.006(\psi + 180)$'.
- When the glider is high, it turns sharply to the right[1]. The rudder must then be put to the left in order to prevent the glider from diving due to the high bank angle[2]. The corresponding R value depends on the bank angle itself, and on the heading in order to trigger diving when the turn ends. Therefore, the second rule is 'If z is high, $R = -1.0 + 0.006(\psi + 180) + 0.01\varphi$'.
- When the glider is high, we use the ailerons to increase the bank angle for the turn to the right. The value for the ailerons depends on the bank angle because we want to stabilize it, and on the heading because we want to decrease the bank angle at the end of the turn. Therefore, the third rule is 'If z is high, $A = -1.0 + 0.006(\psi + 180) + 0.007\varphi$'.

The three rules for turning towards the wind rely on the same principles:

- If z is low, $E = -0.8 + 0.02\theta + 0.005(\psi + 180)$;
- If z is low, $R = 0.006(\psi + 180) + 0.01\varphi$;
- If z is low, $A = 0.006(\psi + 180) + 0.007\varphi$.

[1] We arbitrarily chose this direction, with the consequence that the glider moved from north-west to south-east.

[2] In this configuration, the rudder is almost horizontal and acts like the elevators in the glider's nominal position.

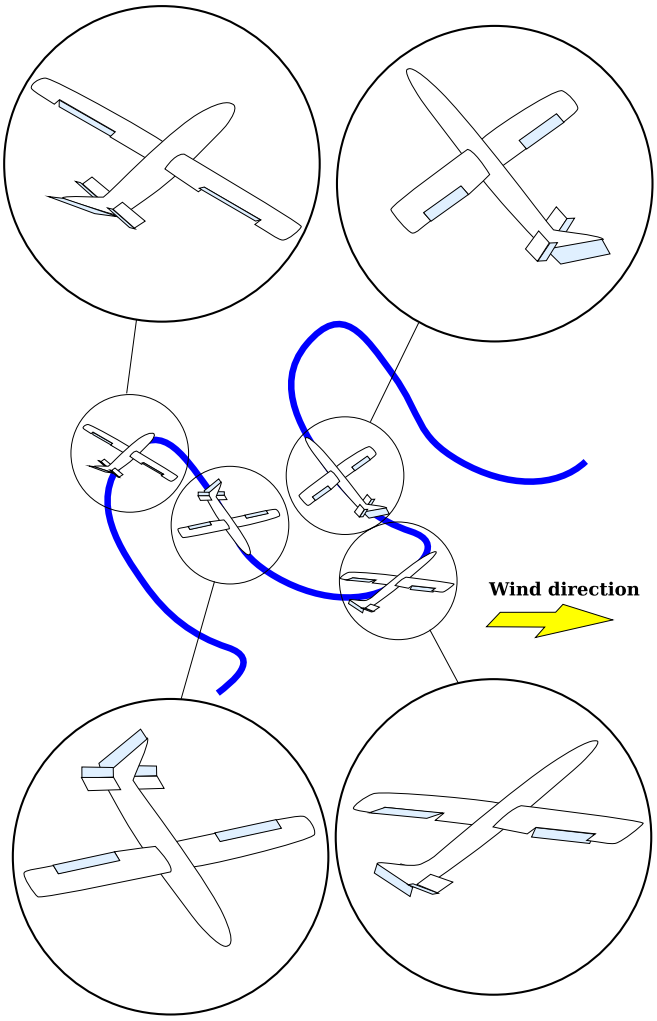


Figure 6. Glider's trajectory and position of its control surfaces in each phase. See the text for details.

The positions of the control surfaces in each phase of the trajectory are shown in figure 6. Here we assume that the wind blows from north to south. It should be mentioned that the rules just described allow dynamic soaring in one direction only, i.e., from the north-west to the south-east. To prompt flights from north-east to south-west, we just have to replace ψ by $(360 - \psi)$, φ by $-\varphi$ and to change the sign of the rules for R and A. We will describe in section 3.3 some experiments we made to evolve controllers able to fly in different directions.

In fact, it appeared that this hand-designed reference controller exhibited the kind of behaviour we were interested in, but it was not really efficient. More specifically, it worked only when the wind speed was high enough, and it failed to keep the glider aloft beyond four complete cycles. Nevertheless it constituted a good starting point for its improvement via an evolutionary algorithm, as described in the following section.

2.6. Evolutionary optimization

The goal of the evolutionary algorithm described here was to generate a more efficient and more robust soaring behaviour.

This algorithm was initialized with the hand-designed rules of the reference controller just described.

The genome of each controller was represented by a vector of real variables, each number corresponding to a rule parameter, or to a boundary of the altitude fuzzy set. Each generation of controllers but the first contained 100 individuals, and the experiments involved 100 generations. The first generation contained 1000 individuals in order to increase the number of interesting controllers at the beginning of the evolutionary run, and thus to accelerate the convergence of the algorithm.

In each generation, the 20 best individuals were kept, while the other 80 individuals were deleted (980 in the first generation) and replaced by 80 new ones. The fitness criteria that were used depended on the experiments that were performed, and will be described in the next section. Each creation of a new individual was done in two steps: first a cross-over, then a mutation.

For the cross-over, two individuals were randomly selected from among the survivors of the previous generation. The probability of selection depended on individual rankings, individuals with higher fitnesses being chosen more often than those with lower fitnesses. Each gene of a new individual was then inherited from one of its parents with an equal probability.

Mutations consisted of slightly changing the value of each gene. The mutation of a gene value γ was defined by:

$$\gamma_{t+1} = \gamma_t + \mathcal{N}(0, \mu * \tau)$$

where the function $\mathcal{N}(0, \mu*\tau)$ represents a normal distribution of zero mean and standard deviation $\mu * \tau$. μ is a mutation factor defined for each gene, and τ is an attenuation factor τ which decreases the amplitude of the mutations in the last generations. Methods for automatically adapting the mutation rate exist [24, 25], but the use of an attenuation factor, as done here, leads to quicker convergence than such alternative procedures.

The mutation factor μ was defined by:

- $\mu = 0.1$ for genes corresponding to rule parameters,
- $\mu = 1.0$ for genes corresponding to altitude fuzzy set boundaries.

The attenuation factor τ has been determined empirically. For generation g it was defined by equation:

$$\tau = \exp\left(\frac{-g}{20}\right).$$

3. Experiments and results

3.1. Low wind-speed conditions

The first series of experiments consisted in using the evolutionary algorithm just described to generate controllers able to secure a dynamic soaring behaviour under lower wind-speed conditions than those required by the reference controller. To limit simulation time, we assessed the fitness of an individual by the minimal wind speed that maintained it aloft for 1000 s, and tried to minimize this value.

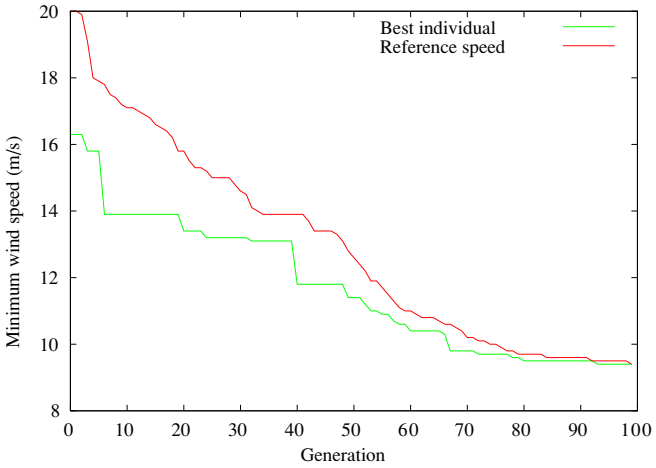


Figure 7. Evolution of the minimal wind speed that secures dynamic soaring in the simulated glider.

For each individual, this process started with the nominal wind speed to which the reference controller was adapted—i.e., 20 m s^{-1}—and proceeded with a dichotomous search. Individuals that could not fly for the whole of the evaluation period, even with the reference wind speed, were ranked according to how long they stayed aloft.

For each generation, if the necessary wind speed for the twentieth and less adapted individual was less than the current reference wind speed, the corresponding value became the new reference speed to be used for next generations. This procedure greatly accelerated the computations.

Figure 7 shows the results obtained along 100 generations. It appears that the best individual thus generated is able to exhibit a dynamic soaring behaviour with a wind speed of only 9.4 m s^{-1}. By comparison, the minimum wind speed needed by the albatross seems to be about 8.6 m s^{-1} [9].

It is interesting to compare the trajectory used by the reference controller with the one used by the best evolved controller (figures 8 and 9). The first loses altitude in each cycle and ends up in the water after only four cycles. The second keeps a constant altitude at the top of each cycle and rapidly follows a periodical trajectory, although the wind speed is twice lower.

Table 1 summarizes the differences between the hand-designed controller and the evolved one. These differences are neither negligible nor reflecting drastic changes. They suggest that the design of dynamic soaring controllers, like those considered here at least, is a difficult task, as small changes in the corresponding parameters may cause a glider to operate perfectly well, or to crash after a few cycles only. This remark probably explains why we never succeeded in evolving such controllers from scratch, and why they proved to be sensitive to turbulences and sensory noise, as mentioned in section 3.4.

3.2. Wing aspect ratio

The goal of this series of experiments was to evaluate the morphology of a real glider on which the dynamic soaring

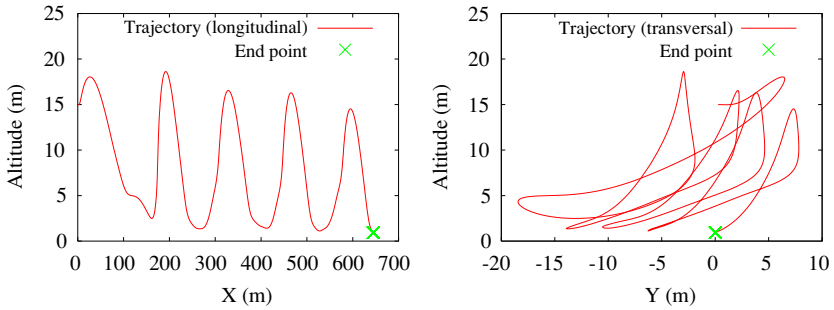


Figure 8. Longitudinal and transversal projections of the glider's trajectory with the reference controller (wind speed: 20 m s^{-1}). The flight is not regular and ends after four cycles at the 'End point'.

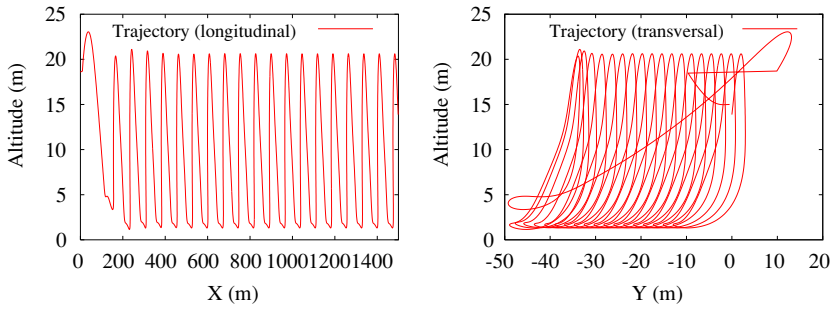


Figure 9. Longitudinal and transversal projections of the glider's trajectory with the best evolved controller (wind speed: 10 m s^{-1}). The flight quickly becomes cyclic and the altitude at the top of each cycle remains constant.

Table 1. Comparison between hand-designed and automatically-evolved rules.

	Hand-designed controller	Evolved controller
Fuzzy sets		
z is high, Ramp bounds	$5.0 \rightarrow 13.0$	$3.095\,691 \rightarrow 11.452\,055$
z is low, Ramp bounds	$2.0 \rightarrow 9.0$	$1.857\,289 \rightarrow 8.453\,340$
Rules		
z is high, $E =$	$-0.6 + 0.006(\psi + 180) + 0.01\theta$	$-0.599\,127 + 0.004\,392(\psi + 180) + 0.006\,740\theta$
z is high, $G =$	$-1.0 + 0.006(\psi + 180) + 0.01\varphi$	$-1.056\,806 + 0.002\,087(\psi + 180) + 0.008\,521\varphi$
z is high, $A =$	$-1.0 + 0.006(\psi + 180) + 0.007\varphi$	$-0.831\,355 + 0.003\,803(\psi + 180) + 0.014\,658\varphi$
z is low, $E =$	$-0.8 + 0.005(\psi + 180) + 0.02\theta$	$-1.288\,160 + 0.006\,382(\psi + 180) + 0.020\,173\theta$
z is low, $G =$	$0.0 + 0.006(\psi + 180) + 0.01\varphi$	$0.000\,793 + 0.011\,099(\psi + 180) + 0.006\,601\varphi$
z is low, $A =$	$0.0 + 0.006(\psi + 180) + 0.007\varphi$	$-0.002\,391 + 0.009\,162(\psi + 180) + 0.012\,029\varphi$

behaviour studied here could be implemented. To this end, we compared results obtained by changing the glider's wing aspect ratio AR_W, defined as the ratio of the wing span S_W and the chord C_W for a rectangular wing, or as the ratio of the squared wing span and the wing area A_W for a wing of any shape (figure 10):

$$AR_W = \frac{S_W}{C_W} = \frac{S_W * S_W}{C_W * S_W} = \frac{S_W^2}{A_W}.$$

In these experiments, we increased the wing aspect ratio by increasing the wing span and decreasing the wing chord. The wing area and mass were held constant all the while. Again, an evolutionary approach was used to seek the lowest wind speed compatible with dynamic soaring for a variety of aspect ratios. The corresponding results are shown in figure 11.

These results show that dynamic soaring can be efficiently performed for aspect ratios varying between 12 and 21. As a comparison, the aspect ratio of an albatross' wing roughly equals 18 [9]. The existence of a lower limit can be related to the fact that a low aspect ratio decreases the lift, and increases the drag of the wing. This explains why gliders and migratory birds have wings with high aspect ratios. As for the existence of an upper limit, it has to do with the nature of the trajectory used for dynamic soaring. Indeed, when the glider is near the water, it makes a sharp turn on the wing to face the wind. The bank angle is then very high, which means that a glider with a 2 m wing span won't be able to get much lower than 1 m before hitting the water. The wind gradient being more important in the first meters above water, a high aspect ratio can prevent efficient dynamic soaring, as it will prevent the glider from going low enough to take advantage from the steepest part of the gradient[3].

[3] As our evolved controller brings the glider down to 1.32 m, one of its wings is approximately 0.15 m above the water during the sharp turn (wing length of 1.375 and bank angle of 58°).

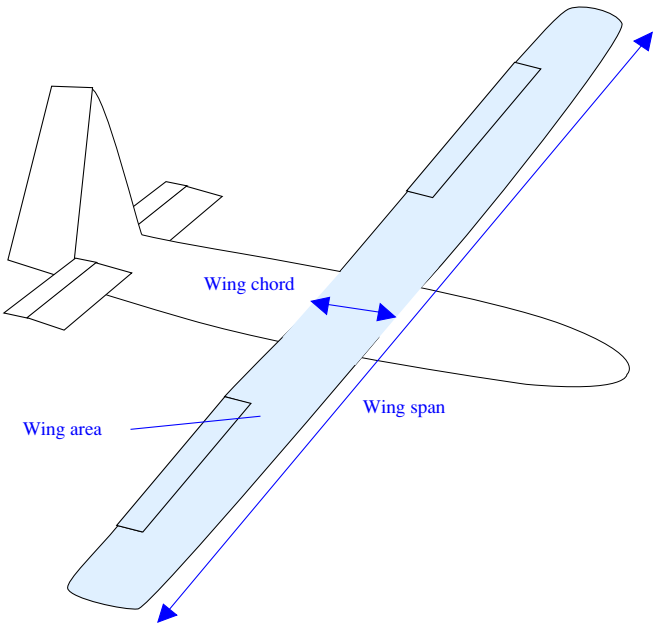


Figure 10. Characterization of the wing span, the wing chord, and the wing area that serve to calculate the wing's aspect ratio.

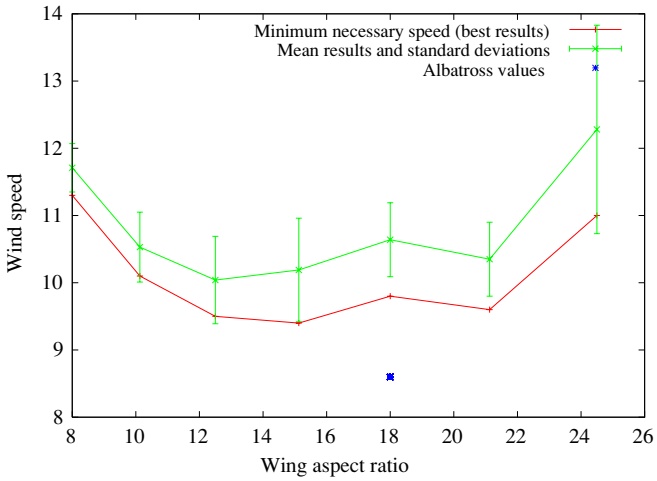


Figure 11. Minimum necessary wind speed as a function of the wing aspect ratio. Efficient dynamic soaring, i.e., under slow wind-speed conditions, can be performed for wing aspect ratios varying between 12 and 21.

3.3. Soaring direction

In the experiments just described, the goal for the glider was merely to stay aloft for the longest time, and the average direction of the glider was not taken into account in the fitness. In fact, the reference controller generates a trajectory whose overall direction is oriented from north-west to south-east with a wind blowing from the north. Here, we wanted to assess the possibility of evolving controllers able to move the glider in different directions, against the wind for instance.

To this end, we changed the fitness function in order to select individuals that flew in a given direction. More precisely, the corresponding fitness function was the difference between the glider's mean heading and the target's heading,

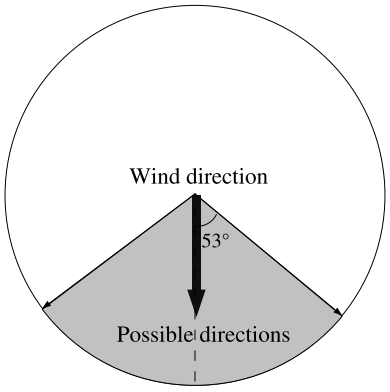


Figure 12. The range of average directions towards which a glider can fly using the controllers evolved in this work.

the goal being to minimize this function. The heading variable ψ being defined as the angle between the wind and the glider's heading, the fitness function $f(ind)$ was set to $f(ind) = \text{abs}(\psi_{\text{mean}} - \psi_{\text{target}})$. For each generation, we selected the individuals whose headings were the closest to the target's one.

Only two experiments were made in order to determine the full range of directions that could be achieved with the kind of controllers studied here. The first one sought controllers generating flights against-the-wind (target heading $= 180°$), while the second one sought controllers generating back-to-the-wind flights (target heading $= 0°$). These experiments made it possible to determine the minimum and maximum angles relative to the wind that delimit the directions in which the glider can fly.

Results obtained during the first experiment indicate that the maximum angle between the glider's heading and the wind's origin was $53°$. In the second experiment, the minimum angle was $8°$.

Inasmuch as symmetrical flying-angle values may be attained by making appropriate changes in a controller's fuzzy rules, it appears that angles between 8 and $-8°$ can be reached by adequately switching the desired direction between these two values[4]. It may be concluded from this section that the full range of flying directions afforded by the variety of controller investigated in this work is $[-53°; 53°]$ (figure 12).

To appreciate the significance of these results, we thoroughly examined the flying behaviours generated by three evolved controllers: the one corresponding to a mean heading of $53°$—the maximum value, the one corresponding to a mean heading of $8°$—the minimum value and the one corresponding to a mean value of $30°$—arbitrarily chosen in the preceding interval. The details of the rules that correspond to these three controllers are given in appendix B.

It thus appears that these controllers implement different strategies. In particular, the $53°$ controller oscillates less than the others (figures 13(*b*)–(*d*)) and remains approximately between 0 and 10 m whereas other controllers reach far higher altitudes (figure 13(*a*)). The movement direction imposed by the $53°$ controller is almost always further from the wind than

[4] If the glider spends 50% of its trajectory oriented towards $8°$ and 50% oriented towards $-8°$, the glider direction will be $0°$ on average.

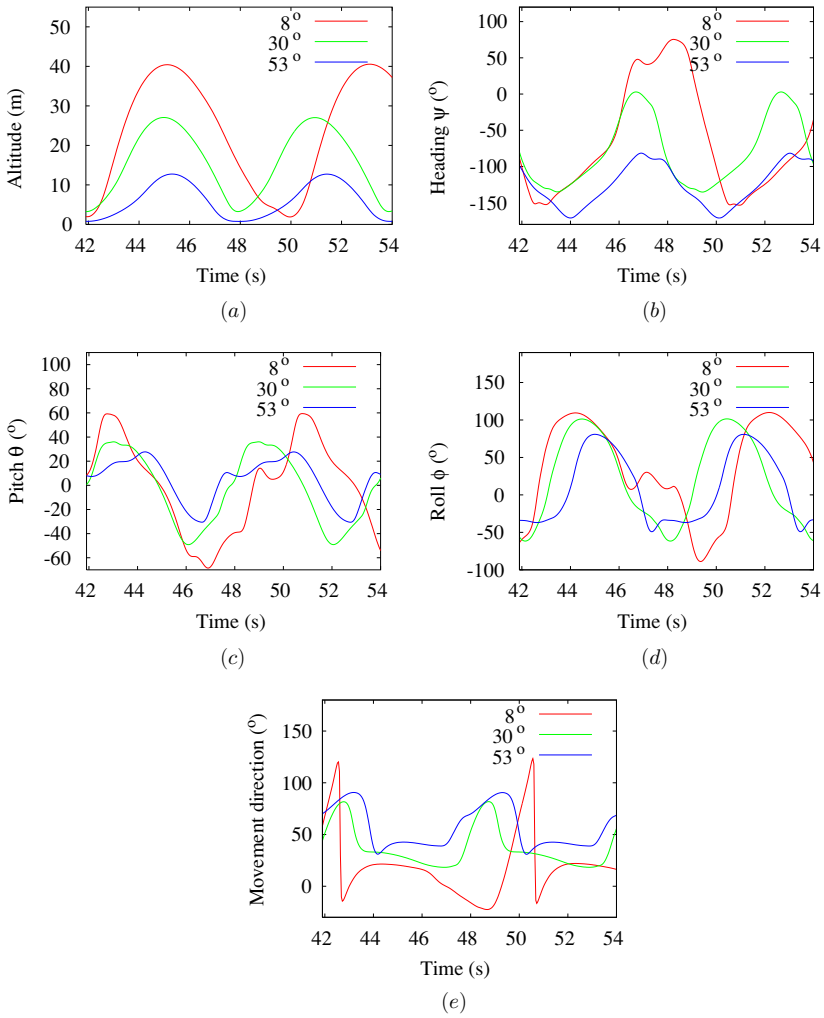


Figure 13. Altitude (*a*), orientation angles (heading (*b*), pitch (*c*) and roll (*d*)) and movement direction (*e*) of a glider flying towards three different mean directions : 8°, 30° and 53° relative to the wind. Data for 8° and 30° have been respectively shifted by 2.4 s and 1.5 s, to synchronize the beginning of each cycle.

those corresponding to the other controllers, even reaching a peak at 90.5° (figure 13(*e*)). The movement direction angle is greater in the lower part of the cycle than in the higher part for all three controllers. This is particularly true with the 53° controller that makes the glider fly lower than with the two other controllers: this way, it avoids being carried by the wind which is stronger at higher altitudes.

With respect to the two other controllers, it also appears that the '53°' one generates greater forces serving to accelerate along the lateral axis during the period when the altitude is high (table 2). Such a strategy increases the lateral speed of the glider during the first cycles, with the consequence that it also increases the drag during this critical phase that makes it possible to gain kinetic energy (figures 14 and 15). As a result, the farther the glider is relatively to the wind direction, the more energy it looses. According to our results, 53° seems to be the greatest angle for which an equilibrium can be found, at least in the experimental setup used here.

These results stress the limits of our approach focussed on dynamic soaring only. Indeed, the glider is unable to follow a trajectory that is, on average, directed against the wind, a performance of which albatrosses are perfectly capable. This is due to the fact that these birds may use other means than dynamic soaring for flying without self-propulsion. For instance, Pennycuick observed albatrosses flying without flapping their wings with a very low wind, or even with no wind at all [2] and, because dynamic soaring is impossible under these conditions, he suggested that another source of energy is used by these birds:

Table 2. Average value of projection on the *Y*-axis of the sum of aerodynamic forces in the absolute frame. These values correspond to the first 60 s of a test flight and accordingly mostly characterize the transient phase. At steady state, the average value of Fy_{abs} is null, as the lateral speed is constant.

	53°	30°	8°
Average Fy_{abs} (N)	0.69	0.34	−0.08
Average altitude z_{avg} (m)	6.9	15.5	19.6
Average Fy_{abs} (N)			
for $z > z_{\text{avg}}$	14.9	3.98	−3.79
for $z < z_{\text{avg}}$	−12.19	−3.6	2.9

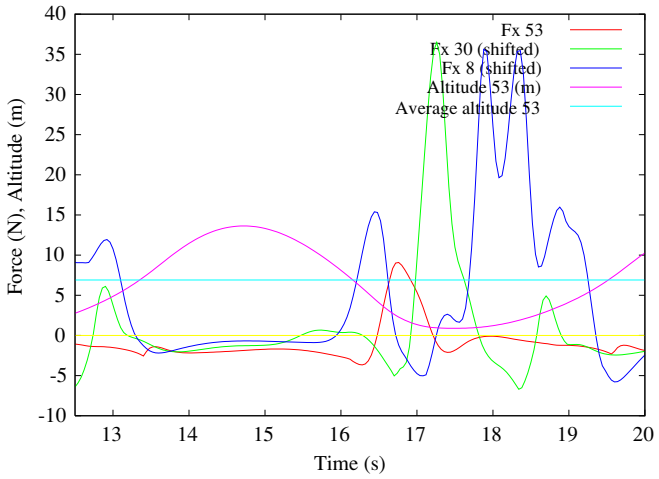


Figure 14. Resultants of aerodynamic forces projected on the longitudinal axis of the glider (F_x). F_x is lesser with the 53° controller than with the two others. In particular, it is negative and, most of the time, smaller than the others when z is greater than z_{avg}.

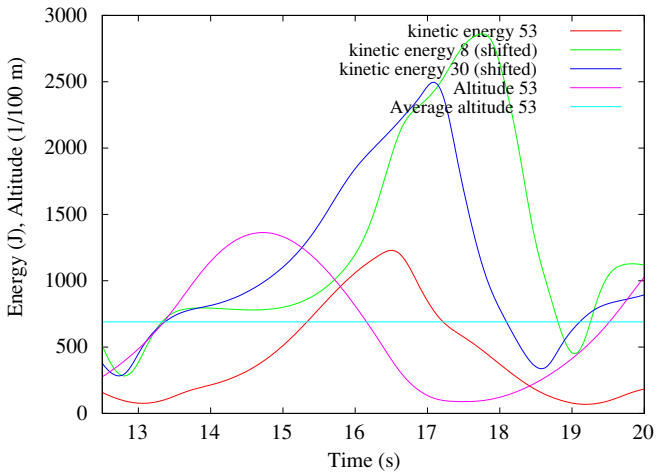


Figure 15. Kinetic energy variations for controllers generating flights in different directions. 6.9 m being the average value of altitude corresponding to the 53°-controlled glider, the period when altitude exceeds this value corresponds to the acceleration phase of the glider. This acceleration is lesser for the 53° controller than for the other two. Kinetic energies corresponding to the 30° and 8° controllers are shifted in time by resp. 1.1 s and 1.8 s, in order to synchronize the three cycles.

> The observations suggest [. . .] that in practice most of the energy for windward pull-ups comes from slope lift [. . .] and relatively little from the wind gradient. The wind gradient may supply relatively more energy in downwind flight, but no direct observations were obtained to support this.

It would therefore be necessary to model such slope currents along the waves to be able to reproduce the trajectories of albatrosses more precisely.

3.4. Control robustness

Other experiments were run to test the robustness of the controllers developed here. More precisely, we evaluated their robustness to variations under initial conditions, to wind turbulences, and to sensory noise. It should be noted that these experiments were not designed to assess a controller's robustness to particular conditions, which would require a much more realistic environmental model, but rather to evaluate the genericity of the evolved controllers. We only look for an answer to the question 'Are these controllers able to cope with slight changes in the environment?'. Answers to the questions 'Are these controllers able to cope with a real environment?' and 'Are these controllers able to adjust their behaviour depending on the environment?' are, of course, related to this study, but they will be investigated thoroughly in a future work.

Concerning robustness to initial conditions, we evolved controllers as we did in the first experiment, but starting with a wider range of altitude, heading and relative speed conditions. These conditions were set at random in the following ranges:

$$\begin{cases} -135^\circ - \delta * 10^\circ < \psi < -135^\circ + \delta * 10^\circ \\ 15\text{ m} - \delta * 0.5\text{ m} < h < 15\text{ m} + \delta * 0.5\text{ m} \\ 20\text{ m s}^{-1} - \delta * 0.5\text{ m s}^{-1} < V < 20\text{ m s}^{-1} \\ \quad + \delta * 0.5\text{ m s}^{-1}. \end{cases}$$

If ten consecutive runs were successful—i.e. if the glider could remain aloft for the whole experiment—we considered the individual as adapted to the variation rate δ.

The results that were obtained indicate that the best evolved controllers were able to stand a variation rate of 9, which means they could maintain dynamic soaring with initial conditions in the ranges:

$$\begin{cases} -225^\circ < \psi < -45^\circ \\ 10,5\text{ m} < h < 19,5\text{ m} \\ 15,5\text{ m s}^{-1} < V < 24,5\text{ m s}^{-1}. \end{cases}$$

Concerning robustness to turbulences, the fitness function was the maximum amplitude of wind-speed perturbations tolerable to the controller. These perturbations were represented as a noise vector added to the reference wind vector. At each time step, the noise vector amplitude was modified by a random value between −0.04 and +0.04 m s^{-1} and its heading was modified by a random value between −0.6 and +0.6° (corresponding to a maximum change rate of 1 m/s/s and 15°/s). This noise vector is added to the reference wind vector. Winds at other altitudes are then computed as stated in section 2.2, so the noise is less important at lower altitudes where the wind is slower.

Controllers able to keep the glider aloft for ten consecutive experiments with a given maximum turbulence amplitude were considered adapted to this noise level. Results indicate that the best evolved controllers were able to stand a maximum turbulence amplitude of 1.7 m s^{-1} for a reference wind set at 20 m s^{-1}.

Finally, to assess the robustness to sensory noise, we did not use evolution. Instead, we simply tested the controller evolved in the experiment described in subsection 3.1, using a reference wind speed of 10 m s^{-1} and adding to the sensors a Gaussian noise with a zero mean and a standard deviation that was systematically varied. The results presented here correspond to the maximum standard deviation value

for which ten consecutive experiments were successful, in the sense that the glider remained aloft for more than 1000 s:

- Maximum standard deviation for noise on altimeter: 0.09 m
- Maximum standard deviation for noise on ψ: 1°
- Maximum standard deviation for noise on θ: 1°
- Maximum standard deviation for noise on ϕ: 1.5°.

Although such results seem to indicate that robust control requires a sensory precision that exceeds that of current off-the-shelf devices, it is still possible that this conclusion will have to be amended if more robust controllers are sought through improved evolutionary approaches. For instance, one could select individual able to simultaneously cope with a low and turbulent wind, on the one hand, and with noisy sensors, on the other hand. Likewise, it is still possible that more robust controllers will be generated using rules better adapted to sensory noise and environmental conditions than those that were used here.

4. Discussion

The main advantage of the dynamic soaring controllers described in this paper is the simplicity of their design and optimization: a first controller has been hand-designed and then optimized by evolutionary algorithms. The trajectories these controllers generate are very similar to those exhibited by albatrosses above the ocean, thus demonstrating that such a natural behaviour can be implemented with few simple fuzzy-rules.

However, if they are able to keep a glider aloft under low wind conditions similar to those an albatross can cope with during dynamic soaring, these controllers cannot get energy from other aerological conditions, like slope wind, which this bird is able to exploit for flying without flapping its wings. These conditions allow it to fly against the wind for example, a behaviour that the controllers studied here are unable to exhibit. To reach a performance similar to that of an albatross, several other controllers should be designed, for instance a series of low-level controllers able to exploit other aerological conditions, and a high-level controller able to switch between different behaviours, i.e. from static soaring to dynamic soaring.

Despite the fact that the apparent lack of robustness of the controllers studied here to noisy sensors, or to wind turbulence, could probably be improved with appropriate evolutionary approaches, controlling dynamic soaring is intrinsically a difficult problem because the corresponding trajectories must be very precise. In particular, when a glider flies at a few centimetres above the ocean's surface, small errors in sensors, or small wind turbulences, may be enough to make it touch the water. To help avoid such a fatal issue, robust filtering methods could be used, as well as avoidance strategies based on optical flow that have already proved to be efficient in similar contexts [26]. Likewise, other rules could be tried that would hopefully correct the trajectory more often and more precisely.

It is also interesting to note that, if the wing aspect ratio is kept small enough, the parameters of the controllers obtained after evolution show only small differences compared to those of the reference controller. This however has a great impact on the performance of the glider, thus suggesting that the generated trajectories are very sensitive to these parameters. This renders the evolution of such a controller from scratch extremely difficult, as this process will get a substantial reward only when it is near the optimal solution. Indeed, very few solutions have a high fitness, the majority of them, especially those of the first generations, being characterized by low fitness values that cannot guide evolution. It is highly probable that, even with other sorts of controllers—e.g., neural networks—or with a different set of fuzzy rules, the same bootstrapping difficulty would be experienced because of the required precision in the generated trajectories.

5. Conclusion

Although dynamic soaring is a complex behaviour that needs to be very precisely controlled, the results that have been presented here demonstrate that it may be generated by a set of simple fuzzy rules bootstrapped by an educated guess. These results also suggest that the implementation of such rules on a real glider capitalizing on simple off-the-shelf sensors is possible, provided the corresponding platform is not used under too challenging conditions, i.e., with a too low and turbulent wind, or with too noisy sensors, and provided trajectories against the wind are not sought. Some hints for relaxing these constraints have been given in the text.

Acknowledgments

This work was supported by a BQR grant from Pierre and Marie Curie University (Paris 6). The authors wish to thank Thierry Druot for the development of the flight simulator and for his help with the analysis of the huge amount of data produced by the simulation. They also express their gratitude to the reviewers for their help in improving the manuscript.

Appendix A. Mass characteristics of the glider

Table A1 summarizes the mass characteristics of the glider. The solid body equations are calculated according to these values.

Appendix B. Control rules evolved for different flight directions

Table B1 specifies the rules for controllers evolved for different directions of flight relative to the wind.

Table A1. Mass characteristics of the simulated glider. Element frames have the same orientation as the glider frame, but they are translated by the position vector.

Element	Position (m) Glider frame	Mass (kg)	Centre of gravity (m) Element frame	Inertial matrix Element frame
Right wing (int.)	$\begin{pmatrix} -0.402\,083 \\ 0.419\,744 \\ -0.014\,6578 \end{pmatrix}$	0.6	$\begin{pmatrix} -0.052\,0833 \\ -9.730\,64 \times 10^{-18} \\ -0.018\,75 \end{pmatrix}$	$\begin{pmatrix} 0.058\,6406 & 2.619\,41 \times 10^{-21} & -0.000\,976\,563 \\ 2.619\,41 \times 10^{-21} & 0.003\,895\,61 & 1.012\,66 \times 10^{-19} \\ -0.000\,976\,563 & 1.012\,66 \times 10^{-19} & 0.062\,3956 \end{pmatrix}$
Right wing (ext.)	$\begin{pmatrix} -0.357\,44 \\ 1.062\,26 \\ -0.052\,73 \end{pmatrix}$	0.2	$\begin{pmatrix} -0.038\,5076 \\ -0.019\,6473 \\ -0.015\,557 \end{pmatrix}$	$\begin{pmatrix} 0.021\,8375 & -0.002\,178\,37 & -0.000\,621\,278 \\ -0.002\,178\,37 & 0.002\,273\,67 & -0.000\,640\,059 \\ -0.000\,621\,278 & -0.000\,640\,059 & 0.024\,0123 \end{pmatrix}$
Left wing (int.)	$\begin{pmatrix} -0.402\,083 \\ -0.419\,744 \\ -0.014\,6578 \end{pmatrix}$	0.6	$\begin{pmatrix} -0.052\,0833 \\ -9.730\,64 \times 10^{-18} \\ -0.018\,75 \end{pmatrix}$	$\begin{pmatrix} 0.058\,6406 & 2.619\,41 \times 10^{-21} & -0.000\,976\,563 \\ 2.619\,41 \times 10^{-21} & 0.003\,895\,61 & 1.012\,66 \times 10^{-19} \\ -0.000\,976\,563 & 1.012\,66 \times 10^{-19} & 0.062\,3956 \end{pmatrix}$
Left wing (ext.)	$\begin{pmatrix} -0.357\,44 \\ -1.062\,26 \\ -0.052\,73 \end{pmatrix}$	0.2	$\begin{pmatrix} -0.038\,5076 \\ 0.019\,6473 \\ -0.015\,557 \end{pmatrix}$	$\begin{pmatrix} 0.021\,8375 & 0.002\,178\,37 & -0.000\,621\,278 \\ 0.002\,178\,37 & 0.002\,273\,67 & 0.000\,640\,059 \\ -0.000\,621\,278 & 0.000\,640\,059 & 0.024\,0123 \end{pmatrix}$
Tail (horiz.)	$\begin{pmatrix} -1.501\,67 \\ 0 \\ 0 \end{pmatrix}$	0.1	$\begin{pmatrix} -0.041\,6667 \\ 5.511\,39 \times 10^{-17} \\ -0.006\,25 \end{pmatrix}$	$\begin{pmatrix} 0.029\,8906 & -4.572\,76 \times 10^{-19} & -0.000\,260\,417 \\ -4.572\,76 \times 10^{-19} & 0.002\,456 & 3.011\,45 \times 10^{-20} \\ -0.000\,260\,417 & 3.011\,45 \times 10^{-20} & 0.032\,331 \end{pmatrix}$
Tail (vert.)	$\begin{pmatrix} -1.31 \\ 0 \\ -0.125 \end{pmatrix}$	0.1	$\begin{pmatrix} -0.052\,441 \\ -0.020\,3419 \\ -0.006\,453\,42 \end{pmatrix}$	$\begin{pmatrix} 0.005\,276\,51 & -0.001\,649\,27 & -0.000\,344\,249 \\ -0.001\,649\,27 & 0.003\,985\,02 & -0.000\,179\,818 \\ -0.000\,344\,249 & -0.000\,179\,818 & 0.009\,244\,67 \end{pmatrix}$
Body	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	0.6	$\begin{pmatrix} -0.5225 \\ -7.080\,19 \times 10^{-18} \\ -0.096\,875 \end{pmatrix}$	$\begin{pmatrix} 0.003\,683\,59 & 1.754\,73 \times 10^{-18} & -0.044\,2188 \\ 1.754\,73 \times 10^{-18} & 0.318\,62 & 9.099\,95 \times 10^{-19} \\ -0.044\,2188 & 9.099\,95 \times 10^{-19} & 0.317\,811 \end{pmatrix}$
Propulsion	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	0.3	$\begin{pmatrix} -0.04 \\ -8.881\,12 \times 10^{-19} \\ -0.038\,8889 \end{pmatrix}$	$\begin{pmatrix} 0.000\,259\,259 & 7.080\,46 \times 10^{-20} & -0.001\,555\,56 \\ 7.080\,46 \times 10^{-20} & 0.001\,897\,67 & -1.480\,89 \times 10^{-21} \\ -0.001\,555\,56 & -1.480\,89 \times 10^{-21} & 0.001\,897\,67 \end{pmatrix}$
Technical load	$\begin{pmatrix} -0.15 \\ 0 \\ 0 \end{pmatrix}$	0.9	$\begin{pmatrix} -0.1 \\ -2.699\,81 \times 10^{-18} \\ -0.058\,3333 \end{pmatrix}$	$\begin{pmatrix} 0.000\,583\,333 & 1.144 \times 10^{-19} & -0.005\,833\,33 \\ 1.144 \times 10^{-19} & 0.011\,355 & -3.533\,34 \times 10^{-20} \\ -0.005\,833\,33 & -3.533\,34 \times 10^{-20} & 0.011\,355 \end{pmatrix}$
Whole glider	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	3.6	$\begin{pmatrix} -0.429\,336 \\ -0.000\,179\,262 \\ -0.054\,6229 \end{pmatrix}$	$\begin{pmatrix} 0.752\,483 & -0.000\,164\,927 & -0.078\,8293 \\ -3.442\,49 \times 10^{-05} & 1.000\,59 & 0.000\,924\,467 \\ -0.085\,4511 & -0.000\,398\,502 & 1.741\,49 \end{pmatrix}$

Table B1. Rules of three controllers generating flights in different directions relative to the wind. Relatively small parameter changes entail quite different behaviours, as described in section 3.3.

	Manual controller	Controller for 8°
Fuzzy sets		
z is high, Ramp bounds	$5.0 \rightarrow 13.0$	$0.654\,010 \rightarrow 9.801\,226$
z is low, Ramp bounds	$2.0 \rightarrow 9.0$	$2.523\,452 \rightarrow 12.672\,154$
Rules		
z is high, $E =$	$-0.6 + 0.006(\psi + 180) + 0.01\theta$	$-0.595\,588 + 0.003\,959(\psi + 180) + 0.008\,705\theta$
z is high, $G =$	$-1.0 + 0.006(\psi + 180) + 0.01\varphi$	$-2.334\,902 + 0.000\,483(\psi + 180) + 0.006\,289\varphi$
z is high, $A =$	$-1.0 + 0.006(\psi + 180) + 0.007\varphi$	$-1.149\,783 + 0.005\,537(\psi + 180) + 0.012\,901\varphi$
z is low, $E =$	$-0.8 + 0.005(\psi + 180) + 0.02\theta$	$-1.730\,784 + 0.000\,289(\psi + 180) + 0.016\,000\theta$
z is low, $G =$	$0.0 + 0.006(\psi + 180) + 0.01\varphi$	$0.074\,608 + 0.007\,348(\psi + 180) + 0.009\,041\varphi$
z is low, $A =$	$0.0 + 0.006(\psi + 180) + 0.007\varphi$	$0.001\,825 + 0.010\,158(\psi + 180) + 0.007\,893\varphi$
	Controller for 30°	Controller for 53°
Fuzzy sets		
z is high, Ramp bounds	$3.983\,407 \rightarrow 19.475\,286$	$3.036\,179 \rightarrow 9.232\,050$
z is low, Ramp bounds	$1.514\,679 \rightarrow 9.579\,008$	$2.521\,118 \rightarrow 7.275\,177$
Rules		
z is high, $E =$	$-0.595\,588 + 0.004\,847(\psi + 180) + 0.008\,689\theta$	$-0.598\,565 + 0.011\,557(\psi + 180) + 0.012\,849\theta$
z is high, $G =$	$-1.748\,078 + 0.004\,470(\psi + 180) + 0.009\,806\varphi$	$-0.886\,981 + 0.006\,102(\psi + 180) + 0.007\,703\varphi$
z is high, $A =$	$-0.986\,498 + 0.009\,999(\psi + 180) + 0.005\,711\varphi$	$-1.135\,517 + 0.014\,454(\psi + 180) + 0.009\,268\varphi$
z is low, $E =$	$-0.748\,618 + -0.001\,014(\psi + 180) + 0.018\,740\theta$	$-0.736\,545 + 0.003\,373(\psi + 180) + 0.025\,652\theta$
z is low, $G =$	$-0.004\,074 + 0.005\,250(\psi + 180) + 0.008\,782\varphi$	$-0.000\,016 + 0.006\,232(\psi + 180) + 0.010\,888\varphi$
z is low, $A =$	$-0.000\,089 + 0.006\,423(\psi + 180) + 0.007\,393\varphi$	$0.000\,988 + 0.013\,196(\psi + 180) + 0.014\,100\varphi$

References

[1] http ://www.nongnu.org/paparazzi/

[2] Pennycuick C J 1982 The flight of petrels and albatrosses (Procellariiformes), observed in South Georgia and its vicinity *Phil. Trans. R. Soc.* B **300** 75–106

[3] Spaar R and Bruderer B 1997 Migration by flapping or soaring: flight strategies of Marsh, Montagu's and Pallid Harriers in southern Israel *The Condor* **99** 458–69

Q1 [4] Smith N G, Goldstein D L and Bartholomew G A 1986 Is long-distance migration possible for soaring hawks using only stored fat? *Auk* **103** 607–11

[5] Hedenström A, Rosén M, Åkesson S and Spina F 1999 Flight performance during hunting excursions in Eleonora's falcon *falco eleonorae J. Exp. Biol.* **202** 2029–39

[6] Parle J 2004 Preliminary dynamic soaring research using a radio control glider 42nd AIAA Sciences Meeting and Exhibit (Reno, Nevada, 5–8 January 2004) AIAA Paper 2004-132

[7] Allen M J 2005 Autonomous soaring for improved endurance of a small uninhabited air vehicle *43nd AIAA Sciences Meeting and Exhibit (Reno, Nevada, 10–13 January 2005)* AIAA Paper 2005-1025

[8] Boslough M B E 2002 Autonomous dynamic soaring platform for distributed mobile sensor arrays *Technical Report,* Sandia National Laboratories

[9] Sachs G 2005 Minimum shear wind strength required for dynamic soaring of albatrosses *Ibis* **147** 1–10

[10] Davies M 2004 An exploratory analysis of dynamic soaring trajectories in shear layer *Technical Report,* Stanford University

[11] Wharington J 1998 Autonomous control of soaring aircraft by reinforcement learning *PhD Thesis* Royal Melbourne Institute of Technology, Melbourne, Australia

[12] Wharington J 2004 Heuristic control of dynamic soaring *Proc. 5th Asian Control Conf.*

[13] Doncieux S, Mouret J B, Muratet L and Meyer J-A 2004 The ROBUR project: towards an autonomous flapping-wing animat *Proc. Journées MicroDrones, (Toulouse, 2004)*

[14] Meyer J A and Guillot A 1991 Simulation of adaptive behavior in animats: Review and prospect *Proc. First Int. Conf. on Simulation of Adaptive Behavior Meyer and Wilson* (Cambridge, MA: MIT Press)

[15] Von Mises R 1959 *Theory of Flight* (New York: Dover) chapter 7 pp 139–69

[16] Shevell R S 1989 *Fundamentals of Flight* 2nd edn (Englewood Cliffs, NJ: Prentice Hall) chapter 3 pp 51–9

[17] Ruggles K W 1970 The vertical mean wind profile over the ocean for light to moderate winds *J. Appl. Meteorol.* **9** 389–95

[18] Beason R C 2003 Through a bird's eye—exploring avian sensory perception Bird Strike Committee Canada

[19] Hoffmann F, Koo T J and Shakernia O 1998 Evolutionary design of a helicopter autopilot *3rd On-Line World Conf. on Soft Computing (WSC 3)*

[20] Nikolos I K, Doitsidis L, Christopoulos V N and Tsourveloudis N 2003 Roll control of unmanned aerial vehicles using fuzzy logic *WSEAS Trans. Syst.* **4** 1039–47

[21] Mamdani E H 1974 Application of fuzzy algorithms for control of simple dynamic plants *Institution Electrical Engineers* Q2

[22] Takagi T and Sugeno M 1985 Fuzzy identification of systems and its application to modelling and control *IEEE Trans. Syst. Man Cybern.* **15** Q3

[23] Shim H, Koo T, Hoffmann F and Sastry S 1998 A comprehensive study on control design of autonomous helicopter

[24] Bäck T and Schwefel H P 1995 Evolution strategies I: variants and their computational implementation *Genetic Algorithms in Engineering and Computer Science, Proc. First Short Course EUROGEN-95*

[25] Schwefel H P and Bäck T 1995 Evolution strategies II: theoretical aspects Genetic Algorithms in Engineering and Computer Science, Proc. First Short Course EUROGEN-95

[26] Muratet L, Doncieux S, Brière Y and Meyer J-A 2005 A contribution to vision-based autonomous helicopter flight in urban environments *Robot. Auton. Syst.* **50** 195–209

Queries

(1) Author: Please provide complete names of editors Meyer and Wilson in reference [14].
(2) Author: Please update references [21], [23].
(3) Author: Please provide page number in reference [22].

A.4 EVOLVING MODULAR NEURAL-NETWORKS THROUGH EXAPTATION

Mouret,J.-B. and Doncieux,S.(2009). *Evolving modular neural-networks through exaptation*. **IEEE Congress on Evolutionary Computation, 2009 (CEC 2009).**

Travaux réalisés dans le cadre de la thèse de Jean-Baptiste Mouret dont j'étais encadrant scientifique.

Evolving modular neural-networks through exaptation

Jean-Baptiste Mouret Stéphane Doncieux

Abstract— **Despite their success as optimization methods, evolutionary algorithms face many difficulties to design artifacts with complex structures. According to paleontologists, living organisms evolved by opportunistically co-opting characters adapted to a function to solve new problems, a phenomenon called *exaptation*. In this paper, we draw the hypotheses (1) that exaptation requires the presence of multiple selection pressures, (2) that Pareto-based multi-objective evolutionary algorithms (MOEA) can create such pressures and (3) that the modularity of the genotype is a key to enable exaptation. To explore these hypotheses, we designed an evolutionary process to find the structure and the parameters of neural networks to compute a Boolean function with a modular structure. We then analyzed the role of each component using a Shapley value analysis.**

Our results show that: (1) the proposed method is efficient to evolve neural networks to solve this task; (2) genotypic modules and multiple selections gradients needed to be aligned to converge faster than the control experiments. This prominent role of multiple selection pressures contradicts the basic assumption that underlies most published modular methods for the evolution of neural networks, in which only the modularity of the genotype is considered.

I. Introduction

Finding the parameters and the topology of neural networks is a complex task that has been successfully tackled with evolutionary algorithms many times (e.g. [1]–[5]). Although these methods easily lead to good parameters for many systems, they turn out to be inefficient to evolve artifacts with arbitrary complex structures. In other words, current evolutionary algorithms are widely successful optimization methods but are bad engineers. This result is surprising with regard to the complexity reached by living organisms using "natural" evolution.

These difficulties echo the main criticisms addressed by early – and current – opponents to Darwin. While it is easy to admit that natural selection can maintain and improve a given trait, how do complex characters emerge at first? Put differently, how natural selection could favor the intermediate structures required to solve complex tasks? Darwin wrote a lengthy answer to this challenge in the last edition of *The Origin of Species* [6] in which he highlighted that a structure does not have to be employed for the same purpose during the whole evolution of a species. Consequently, structures might originate from an adaptation to a function and then be opportunistically co-opted to solve a new problem. Darwin termed this concept *preadaptation*, but modern evolutionary biologists use the less connoted word *exaptation* [7]. According to current paleontology, exaptation explains, in particular, crucial steps of the evolution of life such as the appearance of the digits of tetrapods, initially adapted to an aquatic environment [8], or first bones, useful for the storage of inorganic ions.

Despite the prominent role of exaptation in the evolution of complex organisms, this phenomenon seems to be, if not absent, at least rare in artificial evolution. Complex life forms are subject to many selection pressures such as the performance of their vision system, the ability to move faster, the variety of food they can eat, etc. As a consequence, evolution can improve organisms with respect to many selection gradients and the obtained adaptations can later be employed to solve problems raised by another selection gradient, resulting in an exaptation. Classical artificial evolution defines only one selection gradient, the fitness function, thus preventing the evolutionary process to exploit exaptation to lead to complex artifacts.

Recognizing this problem, several researchers (e.g. [9]–[12]) proposed to replace the fitness function by an implicit evaluation scheme in which individuals are situated in an artificial world. Inhabitants of these simulated environments compete for limited resources while trying to meet sexual partners to spread their genotype. These studies provide many insights about the dynamics of evolutionary systems and can lead to complex "artificial life forms"; however, they are not designed to solve concrete problems, such as creating a neuro-controller for a robot to clean a room or finding the optimal structure of a bridge. In this paper, we draw the hypothesis that Pareto-based multiobjective evolutionary algorithms (MOEA, see [13]) can explicitly provide multiple selection gradients, thus enabling exaptation to evolve complex and potentially useful artifacts.

The existence of a selection pressure may not be sufficient to improve a particular trait: if a gene has many pleiotropic effects[1], it may be impossible to improve one character without altering some other ones; consequently, the more independent the genes coding for unrelated traits, the more efficiently a selection gradient can be followed. In other words, the *genotype-phenotype map* – how genotype variations are reflected in the phenotype – should be *modular* (see [14]). This intuition is confirmed by a recent theoretical study [15] that highlights that genotypic modules, phenotypic modules and selection gradients should be "aligned" to fully benefit from the advantages of modularity for the evolvability of complex systems. Moreover, modularity makes easier the repetition of a functional sub-part in an organism. Such a duplicated module can then be employed for a new function without damaging its use in the original context. The impor-

Jean-Baptiste Mouret and Stéphane Doncieux are with the Université Pierre et Marie Curie (UPMC) - Paris 6, Institut des Systèmes Intelligents et de Robotique (ISIR), CNRS UMR 7222, F-75005, Paris, France; e-mail: {mouret,doncieux}@isir.fr

[1]Pleiotropy occurs when a single gene influences multiple phenotypic traits.

tance of modularity to evolve complex systems and especially neural networks has been widely recognized ([1], [16]–[19]) but, to our knowledge, neural-network modules have never been explicitly linked to selection gradients.

In this article, we propose an evolutionary process designed to exploit exaptation and modularity to synthesize complex systems. Our aim is twofold: (1) investigating the links between evolution, modularity and selection gradients and (2) introducing a scalable method to evolve complex systems. We illustrate our ideas by evolving neural-networks to compute a complex and modular Boolean function, a problem previously employed in another study about modularity [20].

The remainder of this paper is organized into four parts. Section II is an overview of the related previous work. Section III describes our approach, from the definition of a modular genotype-phenotype map to the use of MOEAs to create multiple selection gradients. Section IV details our experiment and analyzes the results using the Shapley value. The last section is a brief discussion.

II. Previous Work

Only a few papers explicitly deal with exaptation and preadaptation in the evolution of artificial systems [11], [21]–[23]. De Oliveira [11] studies an artificial ecosystem of two-dimensional cellular automata designed to make difficult adaptations by natural selection. Exaptation was possible by exploiting a sequence of non-adaptations caused by genetic drift. Studying the evolution of neural networks, Miglino et al. [22] noticed that some changes of the neural networks were non-adaptive – and consequently only selected by chance – but were later required to increase the fitness. Miglino et al. conclude that this phenomenon was an evidence of artificial preadaptation. While these two papers report observations of preadaptation in an artificial context, Graham and Oppacher [23] proposed a process designed to exploit exaptation to improve existing evolutionary algorithms. Tackling a toy problem, the authors designed four fitness functions of increasing difficulty and associated them to four niches. Individuals of each niche were evaluated using the corresponding fitness but they were allowed to migrate between niches at the end of each generation. At the beginning of their experiments, individuals were viable only in the simplest niche but, after a few hundred generations, they managed to migrate from niches to niches to finally solve the most complex problem. These results demonstrate the power of exaptation to solve difficult tasks by providing different fitness gradients.

Using intermediate steps to evolve solutions for complex problems is often referred to as incremental evolution [4], [24]–[26]. In these works, the task is split into stages of increasing complexity. Candidate solutions are first selected using a first fitness function and, once a convergence criterion is reached, the experimenter changes the fitness to a more complex one. Despite its practical successes, this manual approach has several drawbacks (see [27]) such as the need to find a good switch criterion, the necessity to follow designer's ordering of sub-problems, the inability to explore multiple hypotheses and the absence of explicit selection pressures on the previous sub-tasks.

Mouret et al. [27] show that these drawbacks can be efficiently overcome by defining a multiobjective optimization problem in which each objective corresponds to a sub-task and by then solving it using a Pareto-based MOEA (see [13]). At the beginning of the process, individuals obtain a non-minimal fitness only for the simplest objectives. As they improve with respect to these objectives, they will reach the minimal performance for the sub-tasks required to try the more complex ones. As a consequence of the Pareto sort, individuals that obtain a non-minimal fitness on a previously unreachable sub-task will be non-dominated – and consequently selected – but best individuals for the simpler sub-tasks will be non-dominated too. Thus, the evolutionary process will automatically switch to complex tasks as soon as possible, while maintaining a selection pressure on previous tasks and while not depending on any a priori ordering of sub-tasks. Moreover, a part of the population may dominate with respect to one of the intermediate sub-tasks while another part can improve along another selection gradient; the process can thus simultaneously explore different evolutionary paths. Each selection gradient in this case relies on complete solutions features and has no direct impact on sub-parts of it. This differs from natural evolution, in which particular evolutionary pressures can shape specific organs. This difficulty will be addressed in the present paper by introducing modules in candidate solutions.

The evolution of modular systems and especially modular neural networks has drawn considerable attention in the last few years because of the intuitive hypothesis that they are more evolvable than monolithic ones. Most papers about the evolution of modular neural networks deal with the design of modular encodings with different approaches. For instance, Gruau [1], Hornby and Pollack [28] and Mouret and Doncieux [29] derive techniques from evolutionary programming to exploit the intrinsic modularity of tree-based representations; Doncieux and Meyer [30] and Reisinger et al. [19] use simpler representations based on a list of modules, directly encoded, and a blueprint that specifies how modules are connected. In these papers, the authors implicitly assume that modular systems will be favored by the evolutionary process. Consequently, they do not propose any explicit link between the selection pressures and the modules.

Nevertheless, this hypothesis is questionable. As noticed in [31], monolithic neural networks are often more efficient and easier to obtain than their modular counterpart. The observation that living organisms are highly modular, and that it explains at least a part of their success, raises the question: how modularity did emerge in nature? Wagner et al. [32] review many proposed explanations and conclude that the direct selection for evolvability, i.e. the implicit hypothesis behind modular encodings, is one of the less likely ones. Recent papers in theoretical biology emphasize the role of

the selection pressure to explain the modular organization of organisms. In particular, Lipson et al. [33] and Kashtan and Alon [20] employ numerical models to show that rapid changes of evolutionary pressures induce modular structures. In another theoretical study, Altenberg [15] concludes: "My main proposal is that the evolutionary advantages that have been attributed to modularity do not derive from modularity *per se*. Rather, they require that there be an 'alignment' between the spaces of phenotypic variation, and the selection gradients that are available to the organism."

III. Method

A. Modular genotype-phenotype map

Definitions of what a module is are multiple and rather vague. In this work, we chose to call a module a subset of a system containing several entities functionally integrated and largely independent of the entities that constitute the other modules. In the context of neural-networks, the interactions between neurons are defined by the connections and the synaptic weights; so we can consider the network as a directed graph and try to find highly connected clusters. This problem has a long history in graph theory due to its usefulness in sociology – to analyze groups of people – and in biology – notably to analyze gene regulatory networks. Leicht and Newman [34] recently proposed a robust and efficient approach to detect modules by optimizing a measure (simply called "modularity") over the possible divisions of a network. Broadly, Leicht and Newman [34] state that, in a good division, the number of edges between groups is smaller than expected, i.e. in most cases, smaller than the mean number of edges in a random network of the same size. Good divisions regarding this measure can be computed using a spectral analysis method that demands $O(n^2)$ time, where n is the number of vertices in the network.

Such sub-graphs of a neural-network constitute phenotypic modules. Genotypic modules are more straightforward to define since one has only to check that genetic operators only affect sub-parts of the genotype. Using the approach introduced in [34], phenotypic modules can be extracted from any network and many modular genotypes can be designed. Consequently, the important point for the evolvability of complex systems is not only the existence of modules in the genotype and in the phenotype; it is how genotypic modules relate to phenotypic ones.

This relation, called the genotype-phenotype map, is said to be modular if genotypic modules are developed into phenotypic modules[2]. The neural network modules emerge from the topology of the neural network. As a consequence, a small change in the neural network, for instance the addition of a connection induced by a mutation, can considerably modify the optimal division. This makes it difficult to define a modular genotype-phenotype map: depending on the neural network encoding, it can be difficult to guarantee that a genotypic module will lead to a meaningful phenotypic module. In a biological context, Wagner et al. [32] explain that only two processes can lead to a modular genotype-phenotype map: parcellation, which consists in a differential suppression of pleiotropic effects between groups of characters, and integration, which corresponds to the selective acquisition of pleiotropy among characters from the same group. How could these two concepts be transposed to neural networks? Removing pleiotropic effects can be interpreted as making the contour of phenotypic modules more resistant to mutations. Parcellation can consequently be the isolation of the part of the genome coding for a module, extracted using the approach described in [34]. Thus a phenotypic module would remain stable during the evolution. Integration can be seen as the replacement of a module by a copy of another one, thus providing a repetition mechanism.

Trying to implement these operations as mutations in the simplest genotype possible, we start by considering a typical graph-based direct encoding in which two kinds of mutations are possible:

- structural mutation: addition/removal of a neuron or a connection;
- parametric mutation: change of a randomly chosen

[2]In real organisms, many levels of modules are nested and so the modularity of genotype-phenotype map can be difficult to express clearly.

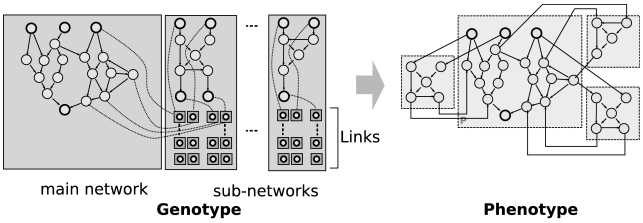


Fig. 1. An individual's genotype is made of a main network and one or several sub-networks. Each sub-network is associated with a list of links that connect the sub-networks to the main network. If several lists of links exist for a same sub-network, the latter is added several times to the main network. The neural network on the right is an example of a phenotype that could have been encoded with the genotype on the left.

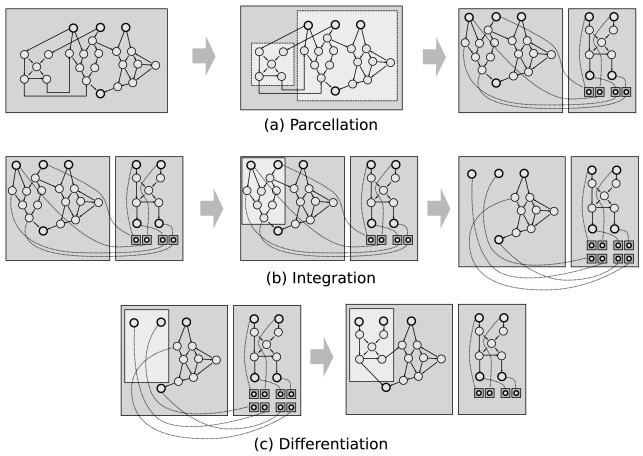


Fig. 2. The three main mutation operators. (a) Parcellation: isolation of a sub-network. (b) Integration: replacing a sub-network by a link to a parcellated one (this allows the repetition of a module) (c) Differentiation: putting a sub-network back in the main network.

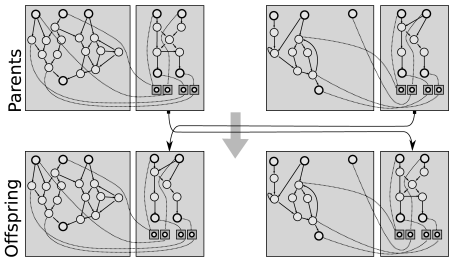


Fig. 3. Principle of the cross-over operator. If both parents have a parcellated module, it is exchanged to create their offspring. Otherwise, the each children is a copy of one of the parent (mutation operators are then applied).

synaptic weight or a neuronal bias; we use here a change in a set of 9 possible values (see appendix).

To easily add integration and parcellation as mutation operators, we then defined our neural network encoding as a main network – encoded using the previously described direct encoding – and a list of sub-networks, each of them associated with links towards the main network (figure 1). Before applying any operator, the main network is divided into modules using [34] and their inputs and outputs are identified. Inputs of a module are either a connection between an extra-module neuron to an intra-module one, or an input of the main network. Outputs are defined similarly. The specification of each modular operator is now straightforward. The parcellation acts in three steps (figure 2(a)):

- removal of a module from the main network;
- addition to the sub-network list;
- addition of the links to the sub-network's links list.

The removed module can be, either randomly selected or deterministically chosen as the best module regarding one of the sub-tasks. In this work, this second approach is used. The integration requires also three steps (figure 2(b)):

- random selection of a module m_1 from the sub-network list;
- random selection of a module m_2 of the main network with the same number of inputs and outputs as m_1;
- removal of the module m_2 from the main network;
- addition of the links to the sub-network links list.

To counterbalance the effects of these two operators, we added an operator named "differentiation" that puts back a sub-network into the main network and removes the corresponding links in the list (figure 2(c)). This operator undoes parcellation and integration and it can be used by the evolutionary process to create a variant of a repeated module. Each operator is assigned a mutation rate (between 0.05 and 0.2, see appendix).

A simple cross-over operator is implemented by exchanging two parcellated modules (figure 3).

Note that the proposed encoding aims at being simple and abstractly related to biology in order to investigate the links between selection gradients, genotypic modules and selection gradients. Our main goal is therefore not to introduce a novel encoding for a neural network but is to provide a tool to study modularity. Other modular encodings for neural networks (e.g [17], [19], [30]) might be used but their structure makes them harder to link to biological literature.

B. Selection gradients

Following the work of Altenberg [15], we "align" phenotypic modules and selection gradients, which means that each phenotypic module should be linked to a fitness function. To that aim, we propose to first define fitness functions able to evaluate sub-networks for potentially useful sub-functions. For each individual, the main network is divided into modules using the approach described in [34]. Each such module and each parcellated module are then evaluated according to each objective. We use the best value obtained by a module for each objective as the fitness of the individual with respect to this objective. Each of these fitness functions constitutes an objective of a multiobjective problem. The main fitness, which rewards the ability to solve the problem we are interested in, is added as another objective.

Using a Pareto-based MOEA to optimize this multiobjective problem has several consequences. An individual with an inefficient overall behavior will be considered as Pareto-optimal if it contains at least a good module for a sub-task. If the main task requires such an efficient module to build more complex solutions, it will be gradually improved thanks to the selection pressure inducted by the corresponding objective. This module can be later co-opted at any time. Moreover, the same module can be repeated (it may be useful elsewhere) or propagated in the population by cross-over.

Another important consequence of the use of a MOEA is that, as shown in [27], solving each sub-task is not mandatory. If the evolutionary process finds better solutions without using modules, these individuals will dominate with regard to the main objective and will consequently be selected. Different hypotheses about the usefulness of each sub-task can therefore be investigated at once: some individuals will be good at one sub-task while some other ones will contain useful modules for other sub-tasks; the evolutionary process will opportunistically modify them, without any a priori knowledge about the ordering of sub-tasks or about their utility.

C. Shapley value

When designing and evaluating an evolutionary process, it is important to understand what makes it efficient. A particular genetic operator may be mandatory to lead to working results; it may also only change the convergence speed or even have no influence on the results. Furthermore, the critical components may be combinations of some particular genetic operators, in which all of them are required to bring improvements but each one is useless alone. In the context of this paper, we would like to be able to estimate the usefulness of each modular operator (parcellation, integration, differentiation and cross-over) and to evaluate what brings the multiobjective approach.

Evaluating the contribution of each such component is equivalent to finding a fair allocation of gains, e.g. the fitness,

between players in a coalitional game, a problem solved in game theory by the Shapley value [35]. The Shapley value is defined from the marginal contribution of a player i to a coalition S, where $i \notin S$ and $v(S)$ is a value function (the fitness or any other relevant measure of efficiency):

$$\Delta_i(S) = v(S \cup \{i\}) - v(S)$$

The Shapley value is then defined by the payoff γ_i of each player $i \in N$:

$$\gamma_i = \frac{1}{|N|!} \sum_{r \in R} \Delta_i(S_i(r))$$

where R is the set of all $|N|!$ orderings of N and $S_i(r)$ is the set of players preceding i in the ordering r.

Once the efficiency of the 2^n configurations has been evaluated, the Shapley value can be computed as a summation of $v(s)$ for all the configurations, properly weighted by the number of possible orderings of the elements (see [36]):

$$\gamma_i = \underbrace{\frac{1}{|N|!} \sum_{S \subset N, i \in S} v(s) \cdot (|S|-1)! \cdot (|N|-|S|)!}_{\text{configurations with i}} - \underbrace{\frac{1}{|N|!} \sum_{S \subset N, i \notin S} v(S) \cdot (|S|)! \cdot (|N|-|S|-1)!}_{\text{configurations without i}}$$

IV. Experiment

A. Experimental setup: $\big[(a \oplus b) \wedge (c \oplus d)\big]$

We evaluated the proposed approach on the evolution of neural networks to compute the Boolean function $\big[(a \oplus b) \wedge (c \oplus d)\big]$, where a, b, c and d are Boolean values and $\oplus$ denotes the exclusive "or" operator (xor). This function, previously used in [20] to study the evolution of modular NAND networks, has a clear modular structure: it is made of two "xor" functions, each of them requiring at least one hidden neuron, linked by a simpler logical "and". The truth table of $\big[(a \oplus b) \wedge (c \oplus d)\big]$ shows that a simple neural network that returns "false" for any input would have a 75% success rate, a good score at the beginning of the evolutionary process. As a consequence, these degenerated neural networks could easily invade the population whereas they do not constitute a good starting point. This makes the typical single-objective fitness for this function very deceptive. The $\big[(a \oplus b) \wedge (c \oplus d)\big]$ function therefore constitutes a simple illustration of the situations in which exaptation could be useful: the typical fitness does not provide useful enough search gradients and we can suggest a helpful sub-function (xor).

We used the sum of errors for each possible set of inputs as the main fitness (expressed in a fitness maximization scheme):

$$F_x = 1 - \frac{1}{16} \sum_{i=1}^{16} |o_i - d_i|$$

where o_i is the output of the neural network for the input set i and d_i the desired output. Each neural network is simulated during 100 time-steps. Since we do not constrain the structure of the neural networks, nothing prevents them from oscillating. To avoid such behaviors, we attribute an arbitrary low fitness if the output is not constant during the last 10 time-steps.

We then defined a second objective F_m that rewards the efficiency of a "xor" module in an individual:

$$F_m = \max_{m \in M} F_{xor}(m)$$

where M is the set of all modules. Considering that a module m is compatible if it has 2 inputs and 1 output, F_{xor} is the sum of errors for the module m with respect to the function "xor":

$$F_{xor}(m) = \begin{cases} 1 - \frac{1}{4} \sum_{i=1}^{4} |o_i - d_i| & \text{if m is compatible} \\ -1 & \text{otherwise} \end{cases}$$

The NSGA-II algorithm was employed with a population of 400 individuals and the parameters described in appendix. We launched five sets of experiments, each of them made of 32 runs:

- exaptation: parcellation (P), integration (I), differentation (D), cross-over (C) and multiple selection pressures (M);
- M: multiple selection pressures only, i.e. no genotypic modules;
- P+I+D+C: genotypic modules only;
- standard: direct encoding without genotypic modules and without selection pressures;
- NEAT, a popular neuro-evolution approach [3].

We didn't compare our results with a modular encoding for neural network because (1) they are very complex to implement and to tune and (2) our main focus is on the benefits of linking selection gradients to modules.

B. Results

Figure 4(a) shows the proportion of converged runs ($F_x > 0.9$) as a function of generation, for each of the three investigated approaches. Less than half of the control runs converged within 5000 generations. This result agrees with [20], in which the authors report that only 72% of their control runs converged in less than 10^5 generations. NEAT leads to substantially better results since almost all the runs converged in 1500 generations and more than half of them in only 500 generations. This confirms the published results in which NEAT outperforms direct encodings (e.g. [3]). Nevertheless, the exaptation-based approach converged faster than NEAT. The 32 experiments converged in less than 1000 generations and half of them in about 300 generations. Figure 4(b) corroborates this observation: on average, 400 generations are required with the exaptation-based approach while NEAT need 700 generations[3].

Surprisingly, not only did the approach based only on the genotypic modularity (P+I+D+C) not improve the convergence rate over the control experiments, but also it deteriorated it slightly. The runs that employed only the multiple

[3]This difference is statistically significant ($p < 0.003$).

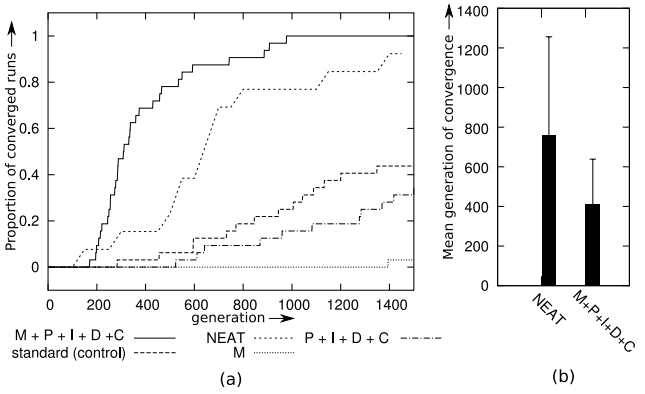


Fig. 4. (a) Proportion of converged runs ($F_x > 0.9$) as a function of generation. "M+P+I+D+C" denotes the full exaptation experiments, "standard" denotes the control experiments, which use a simple direct encoding (with the same parameters as the one used by the exaptation experiment) and one objective; the experiments "M" correspond to the multiobjective approach with the modular genetic operators disabled; in "P+I+D+C", the modular operators are used with the main fitness only. (b) Mean generation of convergence and standard deviation (the other experiments are omitted because only a few runs converged in less than 5000 generations). The differences between these two sets of experiments are statistically significant ($p < 0.003$).

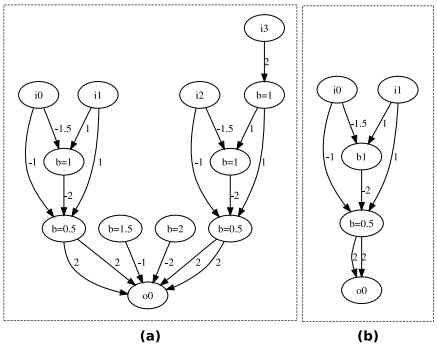


Fig. 5. (a) Typical neural network obtained with the proposed approach. (b) The parcellated module used in (a).

pressures scheme (M) obtained even worse results: only one run converged in less than 1500 generations. Given that our exaptation experiment used the combination of P, I, D, C and M, these results demonstrate that, at least in this experiment, both a modular encoding *and* multiple selection gradients are required to improve the efficiency of the evolutionary process. A simple "waste of resources" may explain the results obtained with the "M" experiment: a part of the population is used to maintain many Pareto-optimal candidate solutions with a low main fitness; if this scheme does not improve the evolutionary process, it is broadly equivalent to reducing the population size, hence possibly deteriorating the convergence rate over the control experiment. Further investigations are needed to fully understand this phenomenon.

A typical neural network obtained with the exaptation approach is drawn on figure 5. It Is clearly made of two repeated modules, each of them computing a "xor" function.

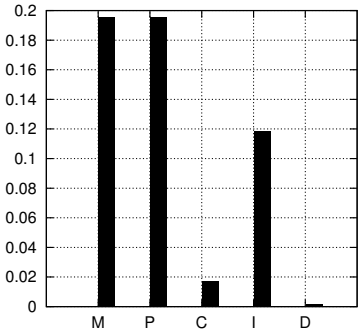


Fig. 6. Shapley values for each component. M: Multiple selection gradients; P: Parcellation; C: Cross-over; I: Integration; D: differentiation.

These modules are two instances of a parcellated module, which obtained an optimal fitness according to F_{xor}.

C. Shapley value analysis

To draw a better picture of what makes the proposed approach efficient, we employed the Shapley value analysis described in section III-C. We investigated the role of the use of two selection gradients (M), of the parcellation (P), of the cross-over (C), of the integration (I) and of the differentiation (D). The efficiency of $2^5 = 32$ variants must therefore be evaluated.

Several choices are available for the value function, which should reflect the efficiency of a variant. We chose to focus our work on the convergence rate because we are primarily interested in getting as much successful experiments as possible. In some configurations, some components of our system can substantially decrease the convergence rate. For instance, we found that using the two selection gradients without parcellation lead to a 3% success rate whereas the control experiment has a 44% rate (section IV-B). If this component is required to obtain good scores in other configurations, it should have a high Shapley value but this value can be substantially lowered by these bad configurations, hiding the interesting analysis. We are interested in understanding which components *improve* over the control experiments and not about the fact that they can lower the scores in some cases. We consequently designed the following value function:

$$v(S) = \max(v_c(\emptyset), v_c(S))$$

where $v_c(S)$ is the convergence rate for the configuration S and $v_c(\emptyset)$ denotes the convergence rate of the control experiment.

We launched 32 runs of each variant for 1500 generations with the same parameters as in section IV-B. The resulting Shapley values are displayed on figure 6.

The two highest values are obtained by the multiple selection gradients (M) and the parcellation operator (P). This means that these two components are *critical* to get the highest convergence rates. The fact that their values are very

close[4] confirms that using only one of them is not sufficient to improve the efficiency of a variant; both of them are required. The parcellation operator links genotypic modules to phenotypic modules and selections gradients are linked to phenotypic modules by the multiobjective approach (M). As a consequence, the obtained Shapley values highlights the need for the alignment suggested by the theoretical work of [15]: genotypic modularity (P) is useless if it is not linked to selection gradients. This conclusion contradicts the underlying hypothesis of current modular encodings for neural networks, which assumes that providing a modular genetic encoding is enough to improve the efficiency of the evolutionary process.

The next highest value is obtained by the integration operator, which is trivially useful for the $\big[(a \oplus b) \wedge (c \oplus d)\big]$function. This Shapley value is substantially lower than the two previous ones, showing that the key-point in the evolution of our modular neural networks is not the repetition mechanism. The cross-over only slightly improves the results and the differentiation has almost no effect.

V. Discussion

In the artificial evolution paradigm, researchers often try to design a universal encoding which would allow to easily solve any problem by only specifying a simple and high-level fitness function. Hence, they put complex mechanisms in the encoding and try to use as little knowledge as possible in the fitness function. This approach is surprisingly disconnected from evolutionary biology, in which most descriptions of the evolution of living organisms primarily rely on selection pressures. When Darwin introduced his theory, nothing was known about the genome; but this didn't prevent him and his successors from successfully explaining the essence of the origin of most species.

The results presented in this paper indicate that the use of a single fitness function might be an over-simplification of the natural evolutionary process, which could prevent the evolution of complex artifacts. In particular, the Shapley value analysis shows that multiple selection pressures are required to efficiently evolve neural networks for the investigated problem. This result was expected as the presence of multiple selection gradients is the key to enable exaptation, hence the evolution of complex designs. However, a more interesting result of this analysis is that multiple selection gradients are useless by themselves. Both modularity and selections pressures are required: if one of them is omitted, the evolutionary process is less efficient than the control experiment. This leads to a new evolutionary scheme centered on evolutionary pressures and genome modularity.

Since we add more knowledge in the fitness function, this approach might move our research away from a mythical "universal problem solver". However, all the published methods to evolve complex systems rely on biases that could constrain them to specific problems, since we do not know any universally good bias. NEAT, for instance, begins the evolutionary process using only one topology, typically a feed-forward neural network without hidden nodes. This requires from the experimenter the implicit knowledge that such a network is a good starting point. Putting biases in the selections gradients possesses at least the advantage of being explicit, and therefore could allow a better analysis. Compared to incremental evolution methods, the approach presented in this paper puts fewer constraints on the search because intermediate steps are not mandatory. We only suggest potentially useful steps; the process is then free to use or to ignore them.

Having highlighted the need for multiple selection gradients and modular genomes, we can wonder if the selected genome and MOEAs are the best tools for these tasks. The main difference between the proposed encoding and other modular encodings lies in the idea that genotypic modules should develop to phenotypic modules. However, applying a selection pressure directly on the sub-network associated with a genotypic module might be enough to facilitate the emergence of phenotypic modules. Hence, more elaborated modular encodings (e. g. [29]) could be used as long as the inputs and outputs of each module can be determined.

The use of a classical Pareto-based MOEA to introduce multiple selection gradients allows us to rely on a well established set of efficient algorithms. However, recent theoretical [37] and empirical [38] studies demonstrate that such evolutionary processes were not well suited to optimize more than three antagonistic objectives. By employing an objective for each sub-function, the proposed process will easily require more than three objectives and consequently could put Pareto-based MOEA out of their limits. Nevertheless, if our starting hypothesis is valid, the objectives will not be antagonistic: the exaptation process should exploit individuals with a good fitness on one objective to build individuals for the more difficult objectives. Therefore, some individuals could dominate on most objectives. However, the dominance relation puts all objectives at the same level whereas we are mainly interested in the main task. Some experiments with modified domination criteria may therefore reveal themselves more efficient.

VI. Conclusion

We explored the hypothesis that multiple selection gradients and a modular genotype-phenotype map were two key-points to evolve complex artifacts. To that aim, we defined a bio-inspired modular encoding and employed it with a Pareto-based MOEA in which one objective rewarded the efficiency of a module to complete a sub-function. We further hypothesized that exaptations should occur in this evolutionary framework by co-opting modules evolved for simple sub-functions to solve more complex ones.

We tested these ideas on the evolution of neural networks to compute a modular Boolean function. Our results show that: (1) the proposed method is efficient to solve this task; (2) both modularity *and* multiple selections gradients were required to converge faster than the control experiments. This

[4]Actually, they are equals but proving this equality is out of the scope of this work.

prominent role of multiple selection pressures contradicts the basic assumption that underlies previously published modular methods for the evolution of neural networks.

In further work, we will try to link modules used in other modular encodings to selection gradients in order to understand the set of features required for modular encoding. Then, other methods to create multiple selection gradients should be investigated because Pareto-based MOEA are not well suited to optimize more than three antagonistic objectives. Last, the proposed method should be benchmarked on other tasks to assess its generality.

REFERENCES

[1] F. Gruau, "Automatic definition of modular neural networks," *Adaptive Behaviour*, vol. 3, no. 2, pp. 151–183, 1995.

[2] S. Nolfi and D. Floreano, *Evolutionary Robotics: Biology, Intelligence and Technology of Self-organizing Machines.* The MIT Press, 2001.

[3] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10(2), no. 99-127, 2002.

[4] J.-B. Mouret, S. Doncieux, and J.-A. Meyer, "Incremental evolution of target-following neuro-controllers for flapping-wing animats," in *From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*, 2006, pp. 606–618.

[5] J. J. Gauci and K. O. Stanley, "Generating large-scale neural networks through discovering geometric regularities," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

[6] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, 1859.

[7] S. Gould and E. Vrba, "Exaptation; a missing term in the science of form," *Paleobiology*, vol. 8, no. 1, pp. 4–15, 1982.

[8] M. Coates and J. Clack, "Polydactyly in the earliest known tetrapod limbs," *Nature*, vol. 347, no. 6288, pp. 66–69, 1990.

[9] T. Ray, "An approach to the synthesis of life," *Artificial Life II*, vol. 11, pp. 371–408, 1991.

[10] D. Ackley and M. Littman, "Interactions between learning and evolution," *Artificial Life II*, vol. 10, pp. 487–507, 1992.

[11] P. P. B. de Oliveira, "Simulation of Exaptive Behaviour," *Lecture notes in computer science*, pp. 354–354, 1994.

[12] S. Elfwing, E. Uchibe, K. Doya, and H. Christensen, "Biologically inspired embodied evolution of survival," *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, 2005.

[13] K. Deb, *Multi-objectives optimization using evolutionnary algorithms*. Wiley, 2001.

[14] W. Callebaut and D. Rasskin-Gutman, *Modularity: Understanding The Development And Evolution Of Natural Complex Systems*. MIT Press, 2005.

[15] L. Altenberg, "Modularity in Evolution: Some Low-Level Questions," in *Modularity: Understanding the Development and Evolution of Natural Complex Systems*, W. Callebaut and D. Rasskin-Gutman, Eds. MIT Press, 2005.

[16] R. Calabretta, S. Nolfi, D. Parisi, and G. Wagner, "A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science," in *Proceedings of Artificial Life VI*, C. Adami, R. Belew, H. Kitano, and C. Taylor, Eds. MIT Press, 1998.

[17] G. Hornby, "Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design," *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 1729–1736, 2005.

[18] S. Doncieux and J.-A. Meyer, "Evolution of neurocontrollers for complex systems: alternatives to the incremental approach," in *Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004)*, 2004.

[19] J. Reisinger, K. O. Stanley, and R. Miikkulainen, "Evolving reusable neural modules," *Proc. of the Genetic and Evolutionary Computation Conference*, 2004.

[20] N. Kashtan and U. Alon, "Spontaneous evolution of modularity and network motifs," *Proceedings of the National Academy of Sciences*, vol. 102, no. 39, pp. 13 773–13 778, 2005.

[21] D. Stork, B. Jackson, and S. Walker, "Non-optimality via preadaptation in simple neural systems," *Artificial Life II*, vol. 10, pp. 409–429, 1992.

[22] O. Miglino, S. Nolfi, and D. Parisi, "Discontinuity in evolution: How different levels of organization imply pre-adaptation," in *Adaptive Individuals in Evolving Populations: Models and Algorithms*, 1996, pp. 399–415.

[23] L. Graham and F. Oppacher, "A multiple-function toy model of exaptation in a genetic algorithm," *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 4591–4598, Sept. 2007.

[24] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: artificial evolution; real vision," in *From Animals to Animats 3, Proceedings of the third international conference on Simulation of Adaptive Behavior*, 1994.

[25] J. Kodjabachian and J.-A. Meyer, "Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects," *IEEE Transactions on Neural Networks*, vol. 9, pp. 796–812, 1997.

[26] J. Urzelai and D. Floreano, "Incremental Evolution with Minimal Resources," *Proceedings of IKW99*, 1999.

[27] J.-B. Mouret and S. Doncieux, "Incremental Evolution of Animats Behaviors as a Multi-objective Optimization," in *From Animals to Animats 10*, 2008.

[28] G. Hornby and J. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artificial Life*, vol. 8, no. 3, pp. 223–246, 2002.

[29] J.-B. Mouret and S. Doncieux, "Mennag: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars," *Evolutionary Intelligence*, pp. 187–207, 2008.

[30] S. Doncieux and J.-A. Meyer, "Evolving modular neural networks to solve challenging control problems," in *Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*, 2004.

[31] C. P. Bowers and J. A. Bullinaria, "Embryological modelling of the evolution of neural architecture," in *Modeling Language, Cognition and Action*, 2005, pp. 375–384.

[32] G. P. Wagner, J. Mezey, and R. Calabretta, "Natural selection and the origin of modules," in *Modularity: Understanding the Development and Evolution of Natural Complex Systems*, W. Callebaut and D. Rasskin-Gutman, Eds. MIT Press, 2005.

[33] H. Lipson, J. Pollack, and N. Suh, "On the origin of modular variation," *Evolution*, vol. 56, no. 8, pp. 1549–1556, 2002.

[34] E. A. Leicht and M. E. J. Newman, "Community Structure in Directed Networks," *Physical Review Letters*, vol. 100, no. 11, p. 118703, 2008.

[35] L. S. Shapley, "A value for n-person games," *Contributions to the theory of games*, vol. 2, pp. 307–317, 1953.

[36] A. Keinan, B. Sandbank, C. C. Hilgetag, I. Meilijson, and E. Ruppin, "Fair attribution of functional contribution in artificial and biological networks," *Neural Computuation*, vol. 16, no. 9, pp. 1887–1915, 2004.

[37] O. Teytaud, "On the hardness of offline multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 4, pp. 475–491, 2007.

[38] K. Praditwong and X. Yao, "How well do multi-objective evolutionary algorithms scale to large problems," *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3959–3966, 2007.

APPENDIX

- population size: 400
- weights/biases: $\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$
- weight/bias mut. rate: 0.2;
- min. neurons (random gen.): 10
- max. neurons (random gen.) : 20
- min. connections (random gen.) : 20
- max. connections (random gen.) : 35
- cross rate : 0.5
- parcellation rate: 0.25
- differentation rate: 0.02
- integration rate: 0.1
- rate of connection addition: 0.15
- rate of connection removal: 0.25
- rate of connection change: 0.1
- rate of neuron add: 0.025
- rate of neuron removal: 0.025
- activation function: $y_i = \tanh\left(-b + 5 \cdot \sum_j w_{ij} x_j\right)$

A.5 BEHAVIORAL DIVERSITY MEASURES FOR EVOLUTIONARY ROBOTICS

Doncieux, S. and Mouret, J.-B. (2010). *Behavioral diversity measures for Evo- lutionary Robotics*. **IEEE Congress on Evolutionary Computation, 2010 (CEC 2010).**

Travaux que j'ai réalisés suite aux résultats de thèse de Jean-Baptiste Mouret dont j'étais encadrant scientifique.

Behavioral diversity measures for Evolutionary Robotics

Stephane Doncieux Jean-Baptiste Mouret

Abstract— **In Evolutionary Robotics (ER), explicitly rewarding for behavioral diversity recently revealed to generate efficient results without recourse to complex fitness functions. The principle of such approaches is to explicitly encourage diversity in the robot *behavior* space instead of in the space of genotypes (the space explored by the evolutionary algorithm) or the space of phenotypes (the space of robot controllers and morphologies). To implement such approaches, a similarity between behaviors needs to be evaluated but, up to now, used similarity measures are problem-specific.**

The goal of this work is to explore *generic* behavioral similarity measures that only rely on sensori-motor values. With such a measure, we managed to evolve the topology and the parameters of neuro-controllers that make a simulated robot go towards a ball, take it, find a basket, put the ball into the basket, perform a half-turn, search and take another ball, put it into the basket, etc. In this experiment, two objectives were simultaneously optimized with NSGA-II: the number of collected balls and the generic behavioral diversity objective. Several generic behavioral measures are compared. To confirm the interpretation of behavioral diversity objective and in an attempt to characterize behavioral similarity measures, they are also compared to human-made behavioral similarity evaluations. They reveal to classify behaviors globally as humans did, but with no clear correlation between the closeness to human classification and the efficiency within an evolutionary run.

I. Introduction

Darwin's theory of evolution relies on natural selection but also on diversity of life forms. In Evolutionary Computation (EC), stochastic search operators, i.e. mutation and crossover operators, were introduced to create such a diversity. However, it has been observed in EC that these two operators are often insufficient to keep a population diverse enough to avoid a premature convergence. To tackle this problem, researchers have suggested to alter the search landscape in order to explicitely foster diversity. This idea led to fitness sharing methods [11], which decrease the fitness of similar individuals. Multi-objective formulations, in which fitness is not aggregated with diversity, have also proved to be efficient in improving evolutionary algorithms [7], [4], [18].

These methods rely on a *distance between genotypes*. Unfortunately, this makes them difficult to use in Evolutionary Robotics (ER) because complex genotypes are often used for which distances are either computationally infeasible or almost meaningless. Thus, many works in ER study neural networks whose topology is evolved: in the worst case, neural networks are encoded as directed graphs whereas typical graph distances is a NP-hard problem [5]; in the best case, a distance can be written for a specific encoding but it cannot be translated to other encodings. Besides these algorithmic considerations, the behavior of a robot results from the evaluation of a dynamic system made with the robot and its environment. Two individuals with close genotypes can exhibit a very different behavior but, conversely, two individuals with very different genotypes can have exactly the same behavior. For instance, two robots can be controlled by different neural networks but act in the same way if the parts that differ are not connected to the ouputs.

Several papers [12], [18], [19], [15] recently described an alternative to genotypic distance in ER: computing distance between behaviors. However, while the authors reported substantial improvements in their ER experiments by encouraging *behavioral diversity* using behavior similarity measures, they employed distances that relied on problem-specific descriptions of behaviors. For instance, in a maze navigation experiment, Lehman and Stanley [15] used the final position of the robot to compare behaviors. As another example, Mouret and Doncieux [18], evolved controllers for robots that had to switch some lights in a particular order; the vector of light states at the end of the experiment was used as a behavior descriptor. In each of these experiments, a vector of problem dependent features was used, allowing to compute a simple Euclidean distance to evaluate how similar behaviors were. All these measures rely on expert knowledge; but this knowledge may be unavailable and its use contradicts one of the main goals of ER: minimizing human intervention in the design process.

As a consequence, having recognized the importance of behavioral diversity in ER but also the lack of generic behavior distances, this paper focuses on defining and comparing problem-independent similarity measures. The questions addressed by this work can be summarized as follows:

- Does a problem-independent behavioral diversity enhance the search as with problem-specific measures?
- How critical is the choice of the behavioral similarity measure?
- If it is, what makes a behavioral similarity measure more efficient than another?

The first goal of this paper is to introduce and benchmark several problem-independant behavior similarity measures. The focus will be on the evolutionary design of behaviors for mobile robots. All of the considered measures will rely on easily available data in this context, i.e. sensori-motor values. In a second step, the considered similarity measures will be analyzed to understand why they perform differently. In the absence of any ground truth for behavior similarity, our analysis is based on the comparaison between these measures

Stephane Doncieux and Jean-Baptiste Mouret are with the Université Pierre et Marie Curie (UPMC) - Paris 6, Institut des Systèmes Intelligents et de Robotique (ISIR), CNRS UMR 7222, F-75005, Paris, France; e-mail: stephane.doncieux,jean-baptiste.mouret@isir.upmc.fr.

Doncieux, S., Mouret, J.-B. *"Behavioral diversity measures for Evolutionary Robotics"* IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)

and measures performed by *humans*.

II. Related works

A. Fitness sharing

Keeping a diverse population requires to balance the efficiency of a solution with its originality within the population. Initial attempts to take it into account relied on the idea of lowering the fitness of an individual by an amount equal to the number of similar individuals in the population [11].

Besides this method that directly modifies the fitness value, crowding methods aims at adapting the selection scheme to take into account the similarity between individuals when choosing which individual to replace [9]. Many over methods allow to build and maintain different *niches*, a niche being defined as a set of similar solutions, generally sharing a common resource, i.e. a fitness value, see [24] for a review and comparison of such methods.

The diversity can also be used as a separate selection pressure, just like any other problem-dependent fitness function. It was shown that this last choice, within a multi-objective approach, led to better results [7], [4], [1], [19], [18]. Likewise, different ways of evaluating a diversity objective have been compared and the distance to the whole population revealed to be more efficient [4].

Few work has focused on the use of *behavior* instead of genotype or phenotype to evaluate the similarity between individuals. This notion appears only when the fitness relies on the observation of a dynamical process, as for a robot in interaction with its environment. The behavior is then a description of this interaction that depends on the environment, and in particular on the initial conditions, on the robot features, i.e. on the phenotype, and on time.

[12] used a behavioral distance within a crowding selection scheme [9] to solve the Tartarus problem—a box pushing problem in a discrete world—while comparing different behavior similarity measures.

[19], [18] defined behavioral diversity as the use of a diversity objective within a multi-objective scheme. In [19], such an approach revealed to compensate the deceptiveness of a XOR-AND-XOR boolean function and in [18] this approach allowed to solve a sequential light-seeking task as efficiently as with a more directed incremental approach. The work presented here follows this approach.

B. Similarity measures

Whatever method is employed to foster behavioral diversity, a similarity measure is required. In the typical set-up, for each generation and for each individual, the distance to the rest of the population is evaluated. This implies to compute, for each generation, n^2 similarity measures, if n is the size of the population[1]. Short behavior observations might be used to accelerate computations, but these sequences must be long enough to actually be representative of robots behaviors. Computational time is then a critical issue. As an example, consider a population of size 100. For each generation, at least 4950 comparisons have then to be performed. If the similarity measure requires $0.01s$, the diversity objective needs $49.5s$ per generation, i.e. more than 27 hours for 2000 generations.

[1]Actually $\frac{n*(n-1)}{2}$ if the measure is symmetric.

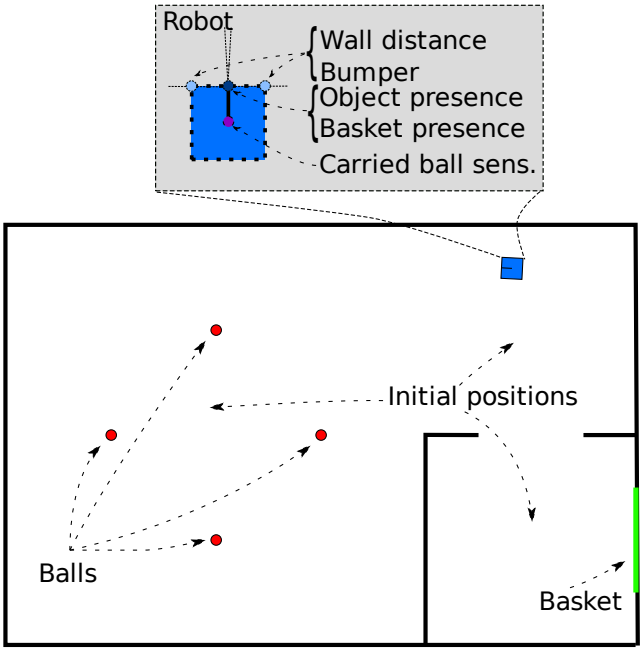


Fig. 1. Overview of the arena and of the robot. The robot has a square shape and has nine different sensors. Object and basket presence field of views are represented on the picture. Four balls are placed in the environment. The goal of the experiment is to put as many balls as possible into the basket. The initial positions of the three evaluation experiments used for the fitness computation are plotted on the figure.

Comparing sequences of values, may they be binary or real is a critical issue for numerous applications like genome study, data search for video or audio data or plagiarism detection. In all of those applications, algorithms have been designed to evaluate the similarity between sequences. Once a discretization has been performed, real value vectors can also be compared using similarity measures operating on discrete values. These methods will consequently be presented first before reviewing measures working on real values.

a) Similarity measures for sequences of discrete values: Evaluating the distance between sequences of discrete values may be done thanks to an *edit distance* [16] that measures the minimum number of operations that have to be done to transform one sequence into the other. In our context, such a distance can hardly be used as computing it is in $O(m^2)$ in time [13], if m is the length of the sequences. For the considered sequence length, it was far too slow to be of use.

Finding common subsequences may be the basis for a similarity search. Measures tolerant to noise and scaling have thus been developped [6], but they give only a Boolean answer relative to a given similarity threshold.

b) Similarity measures for sequences of real values: As for discrete sequences, a similarity measure has been developed to find out which sequences are similar to a

particular sequence out of a given set [3]. This method is designed to be robust to noise, scaling and translation; these features are interesting for our application, but it only provides a Boolean answer— similar or not — relying on a threshold provided beforehand. It can't then be used directly.

To speed up measures, smaller sequences supposed to be representative of the initial one can be used. The comparison may then rely on simple Euclidian distances, even if the initial sequence is long. [2] suggests to use the first coefficients of a Fourier transform as a descriptor of sequences, at least for a preliminary filtering. Likewise, [27] suggests to use the first components of a Principal Components Analysis in the case of multivariate sequences and directly compare them.

A more general information distance based on the notion of Kolmogorov complexity has also been proposed [17]: Normalized Compression Distance (NCD), in which an approximation of Kolmogorov complexity is evaluated with the help of real-world compressors. Such a distance was used in [12] but it revealed to be too slow for our application.

C. Other related works

Evaluating the novelty of a behavior is also an issue for developmental robotics [26]. Focusing learning on new behaviors, or at least behaviors for which the learning rate is the fastest [23], discovering how to build internal representations of its own body [14], using embodiment to structure input spaces [25]: all these tasks require to find patterns or similarities from high dimensional streams of robot perceptions and actions.

Nonetheless, developmental robotics is mainly centered on experiments with a single robot and a long life span—at least longer than for the robot considered in the present paper. Moreover, DR compares and finds similarity within a unique stream of sensor-effector values whereas behavioral diversity need to compare many different streams of sensor-effector values. On the long term, such issues may converge, but as for now, drawing a direct link between diversity in ER and DR is not straightforward.

III. Method

As suggested in [19], [18], [4], [7], we add a diversity objective to the fitness function in a Pareto-based multi-objective optimization. Following the conclusions of [4], the behavioral diversity objective to maximize $o_{bd}(x)$ is the average distance to the rest of the population. Hence the maximization of the fitness function $F(x)$ is transformed to the multi-objective maximization:

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

where P denotes the current population, x, y two individuals, $\sigma(x,y)$ the measure of the similarity of x and y. $\sigma(x,y) = 0$ if $x = y$ and the greater $\sigma(x,y)$, the more different they are.

A. Considered similarity measures

The set of sensor and effector data ϑ is defined as follows:

$$\vartheta = [\{\mathbf{s}(t), \mathbf{e}(t)\}, t \in [0,T]]$$

where $\mathbf{s}(t)$ is the vector of size n_s of the perceptions at time t, i.e. the values coming from the n_s sensors; $\mathbf{e}(t)$ is the vector of size n_e of the effector values at time t; T is the observation length. For simplicity, in the following $\mathbf{s}(t)$ is in $[0,1]^{n_s}$ and $\mathbf{e}(t)$ is in $[0,1]^{n_e}$.

a) Hamming distance: The Hamming distance counts the number of bits that differ between two binary sequences. It can be used to evaluate behavior similarity with, as inputs, ϑ_{bin}, the binarized version of ϑ, computed as follows:

$$\vartheta_{\mathbf{bin}} = [\vartheta_{\mathbf{bin}}(t), t \in [0,T]] = [\{\mathbf{s_{bin}}(t), \mathbf{e_{bin}}(t)\}, t \in [0,T]]$$

with $\mathbf{s_{bin}}(t) = \{s_{bin,0}(t), \ldots, s_{bin,n_s}(t)\}$, $\mathbf{e_{bin}}(t) = \{e_{bin,0}(t), \ldots, e_{bin,n_s}(t)\}$ and

$$s_{bin,i}(t) = \begin{cases} 1 & \text{if } s_i(t) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$e_{bin,i}(t) = \begin{cases} 1 & \text{if } e_i(t) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

with this definition, the hamming distance is computed as follows:

$$\sigma_{ham}(\vartheta_1, \vartheta_2) = \sum_{t=0}^{T} h(\vartheta_{\mathbf{1},\mathbf{bin}}(\mathbf{t}), \vartheta_{\mathbf{2},\mathbf{bin}}(\mathbf{t}))$$

$$h(\vartheta_1, \vartheta_2) = \sum_{i=0}^{len(\vartheta_1)} \delta(\vartheta_1[i], \vartheta_2[j])$$

where $len(\vartheta_1) = len(\vartheta_2)$ denotes the length of the binary sequences ϑ_1 and ϑ_2 and where $\delta(i,j)$ is the Kronecker delta:

$$\delta(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

b) Measure based on Fourier coefficients: Parseval's theorem guarantees that the distance between two sequences is the same in the frequency or in the temporal domain [22]. As suggested in [2], the very first coefficients of the Discrete Fourier transform may carry information descriptive of the sequence they are associated to. Using it instead of the complete sequence has the advantage to reduce the dimensionality: the size of vectors to consider goes from $(n_s + n_e) * T$ to $(n_s + n_e) * n_F$, if n_F is the number of retained coefficients. The measure is then not dependent on the observation length anymore and simple Euclidean distances between the resulting vectors can be used.

c) State count: This measure relies on discretizing available data in order to define perception-action states; the number of times the robot was in a particular state is then evaluated. This results in a vector of n integers, n being the number of such states. States are user-defined. The similarity measure is defined as the mean distance between the vectors:

$$\sigma_{sc}(x,y) = \frac{1}{n}\sum_{i=1}^{n} d_i(x,y)$$

$d_i(x,y)$ is a squared Euclidean distance normalized by the maximum number of times state st_i has been reached by x and y. This avoids individuals reaching a huge number of times a particular state to be over-rewarded:

$$d_i(x,y) = \begin{cases} \frac{(n_i(x)-n_i(y))^2}{max(n_i(x),n_i(y))^2} & \text{if } max(n_i(x),n_i(y)) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$n_i(x)$ is the number of times x has spent in a state st_i. States can be defined either from a coarse discretization, i.e. using $\vartheta_{\mathbf{bin}}$, for instance, or they can include some expert knowledge. The two different cases will be considered here.

As this objective has to be maximized by evolution, individuals that are the only ones to reach a particular state will be rewarded, as do individuals that reach a state a different number of times relative to other individuals of the current population. The normalization increases the reward granted to individuals that are the first to reach a particular state, even if it is reached only one time.

d) Trajectory based similarity measure: This measure consists in discretizing the environment and counting the number of times spent in each square. It is then similar to the state count measure, but with position data instead of perception/action data. This similarity measure should be considered as a control measure that directly exploits the knowledge of the trajectory of the robot; it doesn't exploit the data coming from sensors and effectors.

Other measures could be defined on the basis of the trajectory, but the focus here was on those exploiting information easily available: trajectory is easy to get in simulation, but is more tricky to get on a real robot[2].

B. Robotics experimental setup

The choice of the setup stems from several motivations:

- it should be a difficult task from an ER point of view:
  - the fitness function must not be too directive; hence, we chose a sequential task that rewards only the achievement of the whole sequence;
  - the simulation should make a random search as inefficient as possible;
- the problem should not be on the controller abilities but rather on the fitness design; we selected then sensors that are easy to interpret for an evolved neural network;
- it should be easy and fast to simulate to facilitate re-implementations and comparisons.

The chosen task consists in picking up some balls in an arena and putting them into a basket (fig. 1). The robot is a two-wheeled robot with the following sensors:

- two wall distance sensors (linear between 0 and 8 times the robot size and then constant, the output is normalized between 0 and 1, see figure 1);
- two bumpers (1 if it touches a wall, 0 otherwise);
- two ball detection sensors (1 if a ball is in the view field of the sensor, 0 otherwise);
- two basket detection sensors (1 if the basket is in the view field of the sensor, 0 otherwise);
- one carry ball sensor (1 if a ball is carried, 0 otherwise).

The effectors are left and right wheel motors and a "catch ball" motor (if greater than 0.5, pick up a ball if possible, or keep the carried ball, else throw the carried ball if any).

Figure 1 shows the arena with the details of the sensor configuration of the robot.

This setup is difficult from an evolutionary perspective:

- the robot may catch a ball when it moves on it if the catch effector is above 0.5; as soon as the catch effector goes below 0.5, the ball is thrown and disappears from the arena; if the robot touches the basket at this time, it will be counted as a collected ball and otherwise it just disappears; it is then difficult for the robot to release a ball by chance into the basket as it previously has to learn to keep the ball, reach the basket and then, and only then, release the ball;
- the robot shape is a square and the collision detection system avoids the robot to slide along the walls: once the robot collides with a wall, it has to go backwards or to turn, if possible, to get out of it; once a ball has been put into the basket, the robot is then blocked against the wall and need to learn how to escape from this situation;
- the basket is surrounded by walls to make it more difficult to find and to make new ball catching longer and trickier, as the robot first has to go out of the basket room before being able to see any new ball.

[21] used a similar garbage collecting experiment, but in a different setup and for a different purpose. The fitness counted the objects put outside of the arena no matter where and a small reward was also granted to individuals able to pick up objects. The pick-up process was more realistic than our and closely resembling the one of a Khepera robot.

IV. Behavioral diversity within an evolutionary process

These first experiments aim at evaluating how the different similarity measures behave within an evolutionary context. To test the efficiency of the proposed behavior similarity measures, several different setups were designed.

A. Setups

1) Control experiments: The first control experiment use the ball count objective only. It aims at evaluating the difficulty of the task with a straightforward approach.

[2]a simple integration of the odometry has too much drift, a precise trajectory evaluation must then rely on a SLAM algorithm or on an external motion tracking device.

The second control experiment use the ball count objective together with a random objective. This experiment aims at evaluating the role of the second objective and check if a random objective also enhances the search.

The third control experiment use the state count measure including some expert knowledge in the choice of states. It aims at comparing the proposed generic measures with a problem-specific one. The states are chosen to be representative of situations were we know (or at least suppose) what the robot has to do. Following [10], the states are (the action supposed to be done in this circumstance is given in paren, but not used at all in the fitness):

1) no ball carried, no basket around and no ball around (look for a ball);
2) no ball carried, no basket around, ball nearby (go towards a ball);
3) ball carried, no basket around (look for the basket);
4) ball carried, basket ahead (go towards the basket);
5) ball carried, basket ahead, bumpers on (release the ball);
6) no ball carried, basket ahead, bumpers on (go back to escape from the wall);
7) no ball carried, basket ahead, bumpers off (go away from the basket room).

By construction, the state count objective will encourage individuals to reach at least once each of these states, thus putting individuals in a situation were they can collect balls.

The fourth and last control experiment also uses, as a behavioral diversity objective, a state count, but that relies on the trajectory. The arena has been discretized in 24 different squares measuring $5 \times 5m^2$ (arena size is $20 \times 30m^2$, robot length is $0.5m$). It aims at comparing sensory-motor values with another kind of information, i.e. trajectory.

2) Experiments on tested behavior similarities: The Hamming experiment use the Hamming distance on $\vartheta_{\mathbf{bin}}$ to evaluate behavior similarity. Each sensor or effector data is then transformed into a single bit binary value.

The Discrete Fourier Transform (DFT) experiment relies on the Euclidian distance between the two first components of a DFT, for each separate data stream.

The systematic state count experiment uses the state count similarity measure with a straightforward definition of states based on $\vartheta_{\mathbf{bin}}$. As there are 9 sensors and 3 effectors, this leads to 4096 different states.

Experiments are thus the following:

- *ballcount*: 1 objective, ballcount;
- *random*: 2 obj., ballcount and random;
- *state count (inf.)*: 2 obj., ballcount and behavioral diversity with state count distance on "informed" states;
- *trajectory*: 2 obj., ballcount and behavioral diversity with state count distance based on the trajectory.
- *hamming*: 2 obj., ballcount and behavioral diversity with Hamming distance;
- *DFT (2 comp.)*: 2 obj., ballcount and behavioral diversity with Euclidean distance between the 2 first coefficients of the Discrete Fourier Transform;
- *state count (sys.)*: 2 obj., ballcount and behavioral diversity with state count distance on "systematic" states;

An individual is evaluated in three different contexts defined by the initial position of the robot (see figure 1). The position of the balls and the three initial positions of the robot are constant to guarantee that the difficulty of the problem is the same for all and to make behavior comparisons easier. The evaluation in a particular context lasts 2000 time steps. The theoretical maximum fitness value is 12 (4×3), but within 2000 time steps, only 3 balls can be collected; the maximum value is then 9 in practice. The first 2000 time steps out of the total 6000 are kept for behavior similarity measures. This is empirically chosen as a compromise between the ability to discriminate behaviors and similarity measure computation time. NSGA-II [8] is used with a population size of 600 and for 2000 generations. Each experiment is repeated five times. The topology and the synaptic weights of the neural network controlling the robot are evolved with a simple direct encoding. The random generation process creates neural networks with 10 to 20 hidden neurons, 20 to 100 connections, 9 inputs (directly linked to sensors) and 3 outputs (directly linked to actuators). Network size is unbounded after the initial generation.

Several mutation operators are used (probability in paren):

- adding a neuron (0.025)
- deleting a neuron (0.025)
- adding a connection (0.15)
- deleting a connection (0.25)
- modifying a connection weight, possible values are: $\{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$ (0.2)

No crossover is used. Further details on the neural encoding can be found in [19].

The source code of the experiments is available for download (`http://www.isir.fr/evorob_db`). Experiments were implemented in the Sferes$_{v2}$ framework [20].

B. Results

The mean number of maximum collected balls per run together with the variability is shown on figure 2. The p-values of a Mann-Withney test comparing experiment results is shown on figure 3.

The *ballcount* experiment does not generate any controller able to put more than one ball into the basket per context, as do experiments with a second random objective. No matter what similarity measure is used, using it always allowed to generate controllers, at least once out of the 5 runs, that collect more than one ball per evaluation. The trajectory experiment had a large variability and is thus not statistically different ($p < 0.05$) from every other experiment (except the *random* one).

The *Hamming* experiment generated the most efficient controllers (statistically different from every other setup except trajectory, $p = 0.167$ and state count informed, $p = 0.329$). The performance of this measure was similar in [12] where it almost equalled NCD. It should anyway be emphasized that most of the employed sensors return binary

values. Only one sensor and the three effectors return real values. Experiments in other contexts should be performed to confirm these results. Furthermore, sensor and effector values are comparable as the initial conditions are always the same; with changing initial conditions, such a distance might not be as efficient.

The *state count (inf.)* is the next most successful experiment, confirming results of [10]. The *trajectory*, *DFT (2 comp.)* and *state count (sys.)* give statistically similar results.

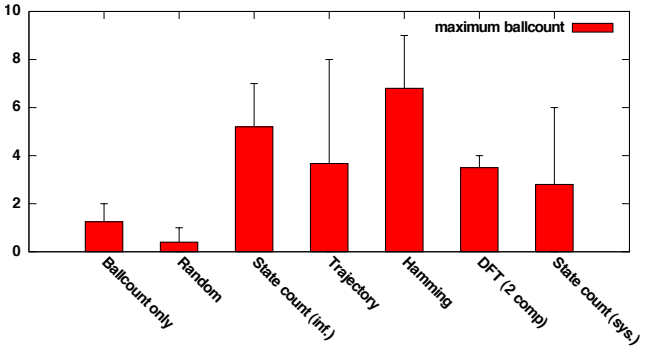


Fig. 2. Mean of the maximum number of collected balls in the three contexts used for evaluation and for each setup after 2000 generations (mean and standard deviation over five different runs).

The maximum value of 9 has been reached, meaning that individuals have been generated that are able to pick up a ball, go to the basket room, release the ball in the basket, go backwards and perform a half-turn, search other balls, collect one, ... Three balls have been collected at best per evaluation (out of four), but actually, some of the evolved individuals revealed to be able to collect the four with some more time.

Coming back to the initial motivations of each control experiment, the following conclusions can be drawn:

- the problem can't be solved directly;
- the behavioral diversity measure doesn't act as a random objective (and vice versa);
- generic behavioral diversity measures can be as efficient as an expert-designed behavioral diversity measure;
- sensori-motor values are competitive with trajectory-based values.

Behavioral diversity objectives can then be both generic and efficient. Furthermore, all behavioral diversity measures do not perform the same. There is a huge difference between the measures, so finding out what makes a similarity measure more efficient than another is an issue to investigate.

V. Behavior similarity measures study

Once it is proved that generic measures can be used, in front of the huge number of different similarity measures that can be defined, it is interesting to try to characterize those measures. The question studied here is then the following: *can we find a way to forecast the efficiency of a behavioral similarity measure within a behavioral diversity objective?*

As the goal of a behavioral similarity measure is to compare behaviors and in the absence of any ground truth, we now look at how it compares to human-made comparisons. Our goal is to answer three questions:

- is it easy to compare behaviors for a human?
- how far or how close are behavioral similarity measures to human-made measures?
- is it correlated to the efficiency within evolution?

If the answer to this last question is positive, looking for the most efficient behavior similarity measure for a behavioral diversity objective would be greatly simplified as it would be possible to make predictions on the efficiency without launching an evolutionary experiment.

Two different setups are considered. In each setup, a set of controllers is selected and, sequentially, for each behavior, we present two other behaviors to a human that has to say which one from the two is the closest to the first behavior. Such a single comparison defines a test case. Answers of different human subjects are compared and we next consider only the cases were all humans agree and see what the behavioral similarity measures answer in these cases. The two setups differ by the choice of controllers. In the first setup, we selected 7 controllers that show an increasing efficiency. In the second, we chose 10 controllers that show a more random behavior relative to the task.

A. Setup1

The chosen behaviors of the first setup can be described as follows:

- no balls collected and the robot ends against the wall;
- one ball is collected, the robot ends against the wall;
- one ball is collected and the robot goes towards the basket but collides with walls before reaching it;
- one ball is collected and put into the basket with no further movements;
- two balls are collected and put into the basket;
- three balls are collected and put into the basket;
- three balls are collected and put into the basket and the robot goes back to pickup the last ball.

This description is not known from the human subjects that can only look at the behavior after a short introduction to the problem, but it is relatively easy to deduce from observation, even for a non-expert. We have considered 35 test cases involving these behaviors.

B. Results for setup1

25 test cases out of 35 had identical results for all the 6 human volunteers. Even in this simple context were behaviors are easy to differentiate and compare, there is then almost 30% of variation. No advices were given on which feature to use to perform the comparison. We don't have any ground truth for the comparison, but at this stage, we can conclude that there are several different ways to perform the comparisons and that they agree on 70% of the data.

We now consider these 70% of data as a ground truth and have looked at how behavioral similarity measures perform on these cases. In order to explore the potential relations between behavior classification efficiency and efficiency of

	random	ballcount	trajectory	state count (inf.)	state count (sys.)	DFT(2 comp.)	hamming
random	1.000	0.041	0.018	0.011	0.018	0.010	0.010
ballcount	0.041	1.000	0.101	0.011	0.147	0.028	0.010
trajectory	0.018	0.101	1.000	0.596	0.523	0.595	0.167
state count (inf.)	0.011	0.011	0.596	1.000	0.070	0.043	0.329
state count (sys.)	0.018	0.147	0.523	0.070	1.000	0.662	0.033
DFT (2 comp.)	0.010	0.028	0.595	0.043	0.662	1.000	0.015
hamming	0.010	0.010	0.167	0.329	0.033	0.015	1.000

Fig. 3. p-values for Mann-Whitney statistical tests performed on each pair of experiments. Comparisons relied on the vectors of the max number of balls put into the basket for each run.

the behavioral diversity objective, figure 4 shows the percentage of closeness between human-made measures and a particular similarity measure relative to the performance in the evolutionary experiment.

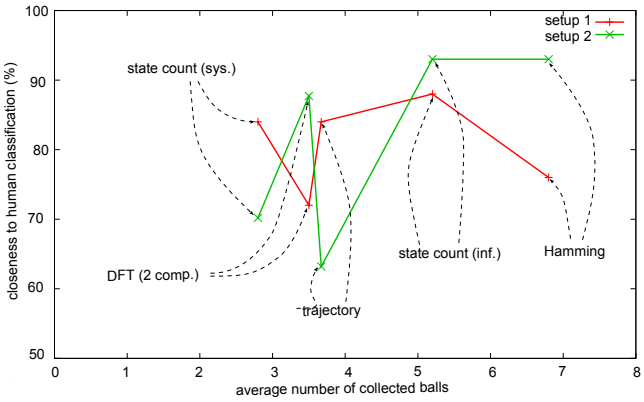


Fig. 4. Closeness of proposed similarity measures related to human-made classification in terms of the performance in a behavioral diversity context. Each dot represents a particular experimental setup. X-axis: average number of collected balls (results from section IV-B), Y-axis: percentage of similarity with human-made measures.

The similarity measures behave as human did as they give the same answer in between 70% and 90% of the cases. This is comforting for the interpretation of the behavioral diversity. Anyway, there is no clear dependency between the performance of the behavioral diversity objective within evolution and the closeness to human measures.

C. Setup2

This second setup aims at evaluating if the previous results are confirmed if we try to compare individuals, i.e. controllers, that appear only during the very first generations. The behavioral similarity measures efficiency is critical during the bootstrap of the process: its efficiency to differentiate individuals that have different ball count values might not be that important as the selection will then be possible according to the ball count objective.

Among the ten considered behaviors, only one has a ball count of one, all the others do not put any ball into the basket. This time, we have considered 120 different test cases. To facilitate and accelerate the comparisons, rather than the video of the behavior, we have shown only the trajectory with the "carrying-ball" information included. As before, in each case, the volunteers have to say, for a particular behavior, which one out of two other behaviors is the closest. In this setup, it revealed much more difficult to perform such comparisons. We have thus made a third choice possible: "impossible to decide with enough confidence".

D. Results for setup2

The first observation is that people confidence in their ability to compare behaviors is very variable. It goes from 76% to 96% with a mean of 88%. As before, we have considered the set of comparisons on which all humans agree. Out of 120 different test cases, only 57, i.e. 47.5%, gathered the same answers. This confirms that the difficulty is not the same as before, as agreements were obtained on 70% of the comparisons in setup1. It also confirms that similarity may differ a lot depending on the priority given to observed features of the behaviors. The closeness of behavioral similarity measures relative to this 47.5% of data, now considered as ground truth, is plotted on figure 4.

The trajectory based measure has a poor result (63.2%), but this is not surprising as the trajectory discretization is very coarse and most trajectories remain in the same area. If we except this value, the similarity measures give on average answers that are similar to the ground truth set in 86% cases (between 70.2 and 93%). This further confirms that such behavior similarity measures globally behave as expected. If we except the trajectory outlier, we can observe a positive correlation between the similarity to human measures and the performance of the evolutionary run.

E. Conclusion

In both setups, the behavior similarity measures give answers that are reasonably close to those of humans. The hypothesis that such measures really behave as a behavioral similarity measure is thus confirmed. These results do not allow to conclude concerning the links between the closeness to a human classification and the performance in a behavioral diversity approach. Characterizing efficient similarity measures, beyond a minimal behavior classification ability, thus remains an open question.

VI. Discussion

The chosen task can be solved by a succession of reactive behaviors. Contrary to incremental approaches, the fitness rewards the whole sequence of behaviors, when it is successful only, and not the intermediate behaviors. Applying it to problems requiring abilities that go beyond simple reactive controllers—memory or learning, for instance—should be

investigated, as a useful evaluation of behavioral similarity might be more difficult to build in these cases.

Testing on other problems is also critical for another issue: do the measures perform globally the same on every problem? If not, the genericity of behavioral diversity approaches will be questioned and the expert knowledge on the problem at hand would just be replaced by another kind of expertise. It is interesting to notice that Hamming distance also gave very good results in Gomez's work [12] in a different setup.

Experiments were repeated a low number of times: five times only. It stems from the evolutionary run length. A single run lasted up to four days (the average is around two days) on a 2.4GHz PC. The total evaluation time of the presented experiments is then near 70 days of computation (7 experiments repeated 5 times, each lasting 2 days on average). More runs have to be launched to further confirms the tendencies observed here.

VII. Conclusion

Results show that behavioral diversity approaches can be defined on the basis of simple behavior similarity measures relying on sensor-effector values. Using a behavioral diversity objective with whatever tested similarity measure gave globally better results than not using such an objective. Among the tested measures, the Hamming distance on a roughly discretized vector of sensor and effector values gave better results, but more runs have to be performed to confirm this. Anyway the behavioral diversity using Hamming distance generated near optimal individuals, able to repeat the following sequence: pick up a ball, find the basket, release the ball in the basket, go back picking another ball, until almost all balls are put into the basket, whereas without the diversity objective, individual to collect one ball only at best have been observed. Sequential behaviors were thus generated on the basis of the desired effect only—besides the behavioral diversity objective, the only other objective was a simple count of the balls released in the basket—and without recourse to any fitness shaping or incremental approach.

Behavioral similarity measures revealed to compare favorably to measures made by humans, thus confirming the interpretation of behavioral diversity objectives. Although behavioral similarity measures gave very different results when used to evaluate the behavioral diversity objective, there didn't seem to be a clear correlation between the closeness to human classification and the performance of behavioral diversity experiment, at least not in every case.

VIII. Acknowledgement

The authors thank Benoît Girard, Tony Pinville, Sylvain Koos and Paul Tonelli for their help doing the comparisons.

References

[1] H.A. Abbass and K. Deb. Searching under multi-evolutionary pressures. In *Evolutionary Multi-criterion Optimization*, volume 2632 of *LNCS*. Springer, 2003.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, volume 730 of *LNCS*, pages 69–84. 1993.

[3] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB '95: Proc. of the 21th Int. Conf. on Very Large Data Bases*, pages 490–501, 1995.

[4] L.T. Bui, H.A. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. *In proc. of IEEE-CEC'05*, pages 2349–2356, Sept. 2005.

[5] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998.

[6] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, volume 1263 of *LNCS*, pages 88–100. 1997.

[7] E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. *In proc. of GECCO'01*, pages 11–18, 2001.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[9] K.A. DeJong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, Uni. of Michigan, Ann Arbor, 1975.

[10] S. Doncieux and J.-B. Mouret. Single step evolution of robot controllers for sequential tasks. In *Proc. of GECCO'09*, pages 1171–1172. ACM, 2009.

[11] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *In Proc. of the Second Int. Conf. on Genetic Algorithms and their application*, pages 41–49, 1987.

[12] F.J. Gomez. Sustaining diversity using behavioral information distance. In *Proc. of GECCO'09*, pages 113–120. ACM, 2009.

[13] D.S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[14] F. Kaplan and V. Hafner. Information-theoretic framework for unsupervised activity classification. *Advanced Robotics*, 20(10):1087–1103, 2006.

[15] J. Lehman and K.O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proc. of ALIFE XI. MIT Press*, volume 54, 2008.

[16] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966.

[17] Ming Li, Xin Chen, Xin Li, Bin Ma, and P. M. B. Vitanyi. The similarity metric. *Information Theory, IEEE Transactions on*, 50(12):3250–3264, November 2004.

[18] J.-B. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *proc. of IEEE-CEC'09*, pages 1161–1168, 2009.

[19] J.-B. Mouret and S. Doncieux. Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In *proc. of GECCO'09*, pages 627–634. ACM, 2009.

[20] J.B. Mouret and S. Doncieux. Sferes$_{v2}$: Evolvin' in the multi-core world. In *Proc. of IEEE-CEC'10*, 2010. to appear.

[21] S. Nolfi. Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. *Evolutionary Robotics*, pages 21–48, 1997.

[22] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1975.

[23] P.-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Trans. on Evolutionary Computation*, 11(2):265, 2007.

[24] B. Sareni and Krähenbühl. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3):97–106, 1998.

[25] O. Sporns and T.K. Pegors. Information-theoretical aspects of embodied artificial intelligence. In *Embodied Artificial Intelligence*, volume 3139 of *LNCS*, page 629. 2004.

[26] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Artificial intelligence: Autonomous mental development by robots and animals. *Science*, 291(5504):599, 2001.

[27] K. Yang and C. Shahabi. A PCA-based similarity measure for multivariate time series. In *MMDB '04: Proc. of the 2nd ACM international workshop on Multimedia databases*, pages 65–74. ACM, 2004.

B CURRICULUM VITÆ

Stéphane DONCIEUX
Maître de Conférence UPMC
ISIR/UPMC, Pyramide Tour 55, BC 173
4, place Jussieu, 75252 PARIS CEDEX 05
Tél. : 01 44 27 87 45
E-mail : Stephane.Doncieux@isir.upmc.fr

Né le 12/08/1976
Nationalité Française

FORMATION

- 1994–1996 : classes préparatoires aux grandes écoles, lycée Champollion
- 1996–1999 : ENSEA (École Nationale Supérieure de l'Électronique et de ses Applications)
- 1998–1999 : DEA IARFA, UPMC
- 1999–2003 : Thèse de doctorat (allocataire-moniteur MESR), UPMC : « *Évolution de contrôleurs neuronaux pour animats volants : méthodologie et applications* ». Soutenue le 23 juin 2003 devant le jury suivant :
  - Jean-Marc Alliot, rapporteur
  - Yves Brière, examinateur
  - Patrick Galllinari, examinateur
  - Patrick Hénaff, examinateur
  - Auke Jan Ijspeert, rapporteur
  - Jean-Yves Jaffray, examinateur
  - Jean-Arcady Meyer, directeur de thèse

EXPÉRIENCE

- 2002 – 2004 : Attaché Temporaire d'Enseignement et de Recherche (ATER), Université Pierre et Maris Curie
- depuis 2004 : Maître de Conférence (titularisé le 01/09/2005), Université Pierre et Marie Curie

C RAYONNEMENT ET RESPONSABILITÉS SCIENTIFIQUES

C.1 RESPONSABILITÉS ADMINISTRATIVES ET SCIENTIFIQUES

Depuis début 2007, je suis, avec Bruno Gas, **responsable de l'équipe SIMA de l'ISIR** (Institut des Systèmes Intelligents et Robotique, créé début 2007). L'équipe regroupe 13 permanents, 18 doctorants et 3 post-doctorants, elle représente un tiers des effectifs du laboratoire environ. Avec Bruno Gas, nous avons défini le positionnement de l'équipe, issue du regroupement de chercheurs et d'enseignants chercheurs de disciplines différentes (mécanique/robotique, traitement du signal et informatique) et provenant de plusieurs laboratoires d'origine. Je fais partie du bureau exécutif de l'ISIR, qui regroupe les responsables d'équipe et le directeur. Je suis également **membre du conseil de laboratoire** et **membre du conseil scientifique**.

Je suis responsable du déploiement du site web de l'ISIR (http://www.isir.fr) dont j'ai défini le cahier des charges, qui a été implémenté par Ludovic Billard. Je suis responsable de l'achat du cluster de l'ISIR début 2010 : 10 noeuds bi-quadcore DELL (recensement des besoins, interaction avec les fournisseurs). Je suis également le contact pour les recrutements en 27eme section pour les besoins du laboratoire (1 poste par an depuis 2007).

C.2 RAYONNEMENT SCIENTIFIQUE

Je suis ou ai été organisateur des événements scientifiques suivants :

- "Exploring New Horizons in Evolutionary Design of Robots" : workshop organisé dans le cadre de la conférence de robotique IROS le 11/10/2009 à Saint Louis (co-organisateurs : J.B. Mouret, ISIR/UPMC et N. Bredeche, LRI/Paris Sud). Site web : http://www.isir.fr/evoderob ;
- RTP Bionique : organisation de la journée thématique "Moving in fluids for animals and robots : physics, (bio)mechanics, control and perception", à l'ESPCI et à l'UPMC les 12 et 13 novembre 2009 ;
- JET : organisateur local des Journées Evolutionnaires Trimestrielles le 26 mars 2010 à l'UPMC ;
- SAB2010 : organisateur local de la conférence internationale SAB2010 (Simulation of Adaptive Behavior) à Paris du 24 au 28 août 2010.

Je suis membre du comité éditorial de la revue française RIA (Revue d'Intelligence Artificielle).

Je suis relecteur d'articles pour les journaux ou conférences suivants :
- IEEE-Transaction on Mental Development
- IEEE-Transaction on Robotics
- Journal of Robotics
- Bioinspiration and Biomimetics (journal)
- Revue d'Intelligence Artificielle
- International Conference on the Simulation of Adaptive Behavior (SAB)
- International Conference on Robotics and Automation
- IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
- IEEE International Conference on Robotics and Automation (ICRA)

J'ai donné des séminaires dans un cadre scientifique (liste non exhaustive) :
- 24/09/2005 : conférence Jacques Monod à Roscoff
- 09/10/2006 : "groupe modèle" au Collège de France
- 02/11/2007 : workshop IROS drones
- 22/09/2008 : workshop IROS Multi-objectif
- 13/11/2009 : Journée RTP Bionique
- 28/01/2010 : Institut Jean le Rond d'Alembert
- 11/02/2010 : Séminaire dans le cadre du projet EvoNeuro
- 01/04/2010 : GT2 Drone

et également dans un cadre grand public :
- Séminaire dans le cadre des 100 ans de l'inventeur de l'hélicoptère à Montbéliard en juillet 2007
- Séminaire dans le cadre de la Fête de la Science en octobre 2007
- Séminaire organisé par le Muséum d'Histoire Naturelle de Toulouse, dans le cadre du bi-centenaire de la naissance de Charles Darwin en juin 2009
- Séminaire dans le cadre de la Fête de la Science en novembre 2009

J'étais membre des jurys de thèse suivants :
- Cédric Hartland, 16/11/2009, LRI. Titre : A contribution to robust adaptive robotic control acquisition
- Sylvain Cussat-Blanc, 17/11/2009, IRIT. Titre : Créatures Artificielles : Développement d'Organismes à partir d'une Cellule Unique

C.3 PARTICIPATION À DES PROJETS

Je suis actuellement responsable d'un work package dans le cadre du **projet ANR EvoNeuro** qui a commencé fin 2009 pour une durée de 3 ans.

J'étais responsable d'un projet interne nommé ROBUR, visant à réaliser un drone autonome à ailes battantes. Cette application sert de démonstrateur dans le cadre des travaux menés sur les algorithmes évolutionnistes ou la localisation et la cartographie robotique. J'ai obtenu un **BQR UPMC** d'un montant de 15k€ en 2007 pour financer ce projet. Ce projet interne a donné lieu au **contrat REI/DGA 'ROBUR'** (2007-2009), dont j'étais le responsable scientifique, contrat de 48k€.

J'ai participé à des montages de projets tant sur un plan national (projet ANR), qu'international (projets européen de type STREP et également IP).

J'ai participé à des évaluations de projets ou de demandes de financement auprès de l'ANR ou de la région Aquitaine.

C.4 RELATIONS AVEC LE MONDE INDUSTRIEL OU SOCIO-ÉCONOMIQUE

Le projet ROBUR a fait l'objet d'une démarche de valorisation qui nous a permis de remporter le **concours OSEO Emergence** ainsi qu'un financement par le **comité PARINOV** (40k€). À cette occasion, j'ai pu rencontrer des acteurs industriels du domaines des drones, notamment BERTIN et Infotron à un niveau national ou l'israëlien IAI à un niveau international.

Toujours dans le cadre du projet ROBUR, nous avons déposé en septembre 2007 un **brevet** portant sur le mécanisme d'actionnement des ailes. Je suis co-inventeur de ce brevet, ainsi que les personnes ayant contribué à son développement, à savoir :

- Thomas Ravasi : étudiant (Paris XI - IUT Cachan)
- Pascal Martinelli : enseignant (Paris XI - IUT Cachan)
- Christophe Grand : enseignant (Paris XI - IUT Cachan)
- Emmanuel de Margerie : post-doc (UPMC - ISIR)
- Jean-Baptiste Mouret : doctorant (UPMC - ISIR)

En 2010, je suis le contact scientifique pour une **chaire d'excellence UPMC** financée par un industriel sur le thème des drones avec l'entreprise RTE (Réseaux de Transport d'Electricité).

D ENSEIGNEMENT

J'ai enseigné à tous les niveaux de la formation universitaire : de la première année de licence à la dernière année de master et dans des matières très variées de l'informatique.

Récapitulatif des matières et niveaux enseignés :

Intitulé	Niveau	Type
Pascal et Excel	DEUG SCM	TD/TME
Scripts shell et internet	DEUG MIAS	TD/TME
Machine et système (assembleur INTEL)	DEUG MIAS	TD/TME
Programmation récursive	L1	TD/TME
Introduction au langage C	L1	TME
C Avancé	L2	Cours/TD/TME
Réseaux	Licence (éq. L3)	TD/TME
Génie Logiciel	M1	TD/TME
Animat	M2	Cours
Introduction aux agents adaptatifs	M2	Cours

De par le passé, j'ai été enseignant dans les UE de Pascal, de programmation récursive, de scripts shell et internet ainsi que de réseaux. A ce titre, j'ai participé à l'équipe pédagogique de ces différentes UE. J'ai également été chargé de cours dans l'UE de M2 "Introduction aux agents adaptatifs" où je présentais les algorithmes évolutionnistes, en introduction à l'UE Animat se déroulant à la période suivante.

Je suis chargé de cours de l'UE C avancé et ai participé à son montage. J'ai conçu les cours, en collaboration avec Jean-Lou Desbarbieux, responsable de l'UE et chargé de l'autre cours, et ai également participé à la rédaction des TD/TME. Je rédige un examen par an. En TD, je fais de petites interrogations pour vérifier régulièrement le niveau des étudiants. J'ai conçu un système de correction automatique qui a été déployé dans l'UE pour permettre aux étudiants d'avoir un retour immédiat pendant les séances de TME. Ce système permet également aux enseignants d'avoir directement les notes de leur groupe, séance par séance (notes automatiques qui peuvent être complétées par la lecture de quelques compte rendus et la prise en compte de l'assiduité).

Je suis chargé de TD dans l'UE d'Ingénierie du Logiciel (M1), UE ouverte aux étudiants de plusieurs parcours du master d'informatique : STL et IAD notamment. L'équipe pédagogique est d'ailleurs mixte entre enseignants dépendant de ces deux parcours. J'ai participé à la rédaction initiale de sujets de TD/TME et, au même titre que les autres enseignants de l'UE, participe à la rédaction des partiels et examens, ainsi qu'à la correction des copies.

Je suis responsable de l'UE Animat depuis plusieurs années. Cette UE présente différents aspects de l'approche Animat, chacun présenté par un spécialiste du sujet : neurosciences computationnelles, apprentissage du contrôle moteur, apprentissage par renforcement, navigation et cartographie, robotique évolutionniste. Les étudiants sont évalués sur la base d'un projet consistant à reproduire les résultats d'un article scientifique. Au cours d'une soutenance orale devant un jury, ils doivent présenter et positionner le travail dans son domaine, présenter les résultats qu'ils ont obtenus et les comparer à ceux des auteurs. Ils sont encouragés à proposer des extensions. Leur présentation doit s'achever sur une discussion critique (au sens constructif du terme), ce qui les pousse à prendre du recul sur le sujet qu'ils ont traité. L'objectif de ce mode de notation est, outre l'évaluation de la compréhension du cours, de parfaire leur formation au travail d'un scientifique, en complément aux UE d'initiation comme les TER ou l'UE d'initiation à la recherche (IREC). Ils ont ainsi à restituer un travail dans un contexte aussi proche que possible de celui de conférences ou de séminaires.

Je propose régulièrement des sujets de TER en M1 (Travaux Encadrés de Recherche), voilà quelques exemples de sujets proposés :

- réseaux de neurones sur GPU
- simulation d'un robot quadrupède de type "Robudog"
- génération de la structure des ailes d'un oiseau artificiel par algorithmes évolutionnistes
- évaluation de la robustesse d'un comportement de soaring à l'aide d'un mélange de HMM
- modélisation d'un drone à ailes battantes et optimisation de sa commande

J'ai également proposé des sujets d'Initiation à la Recherche, comme par exemple, "neurosciences computationnelles et algorithmes évolutionnistes".

Je suis responsable d'une UE de projet en langage C à l'EPU/UPMC. Les étudiants ont le choix entre quinze projets différents comme la programmation d'un Pacman, d'un Tetris ou encore d'un simulateur de robot. La plupart des projets comprennent la programmation d'une interface graphique et sont conçus pour être motivants pour les étudiants. Pour cette raison, ils peuvent choisir leur sujet et dans l'ensemble travaillent sur des sujets différents. Le niveau des étudiants est très hétérogène, certains n'ont qu'une très faible expérience en programmation. Certains sujets sont donc conçus pour être abordables même avec un faible niveau, d'autres au contraire représentent des défis à même de motiver les meilleurs étudiants. De plus, la rédaction des sujets est suffisamment ouverte pour que des étudiants de niveau varié puissent atteindre l'objectif qu'ils se sont fixés. Dans ce but, les étudiants doivent rendre un rapport intermédiaire présentant le cahier des charges qu'ils se fixent ainsi qu'un programme de développement. Un suivi très régulier permet de s'assurer de leur avancement et du réajustement de leurs objectifs si jamais ils les atteignent plus vite que prévu. Une soutenance finale permet de vérifier l'état d'avancement de leur programme comparativement au plan de développement et la maîtrise de leur code. Un des objectifs principaux de ce module est de développer leur autonomie. Je leur montre donc comment utiliser des outils de développement, de debogage et je les encourage à utiliser des bibliothèques existantes. Mon rôle est de les guider dans leur démarche de recherche d'information plutôt que de leur donner des solutions clé en main, même si, bien sûr, je les conseille lorsqu'ils ont trop de difficulté à résoudre

un problème technique particulier. Jusqu'à présent, tous les étudiants sont parvenus, à la fin du projet, à avoir un programme fonctionnel, avec une interface graphique utilisable, même ceux dont le niveau initial était très faible.

J'enseigne également à l'ENSTA, une école d'ingénieur où je participe au cours de robotique mobile (C10-2) depuis plusieurs années pour présenter la robotique évolutionniste.

Je suis responsable d'une UE d'optimisation, proposée dans le cadre du master SAR pour la première fois en novembre 2009. L'UE n'a pas encore ouvert, faute d'un effectif étudiant suffisant.

E ENCADREMENT

J'ai été (ou suis encore) **encadrant de thèse** pour les étudiants suivants :

- Jean-Baptiste Mouret (thèse commencée en septembre 2005 et soutenue le 5 décembre 2008, actuellement Maître de Conférences à l'ISIR/UPMC)
- Adrien Angeli (thèse commencée en septembre 2005 et soutenue le 11 décembre 2008, actuellement Post-doc à l'Imperial College à Londres)
- Mohamed Hamdaoui (thèse commencée en octobre 2006 et soutenue le 16 décembre 2009, actuellement en post-doc à l'école Centrale de Paris)
- Sylvain Koos (thèse commencée en septembre 2008)
- Paul Tonelli (thèse commencée en septembre 2008)
- Tony Pinville (thèse commencée en décembre 2009)
- Charles Ollion (thèse commencée en octobre 2010)

J'ai été l'encadrant de **stage de master** pour les étudiants suivants :

- Charles Ollion (stage de master réalisé en 2010, actuellement en thèse à l'ISIR), sujet : étude des mesures d'exploration dans l'espace des comportements
- Jérémie Decock (stage de master 1^{ere} année réalisé en 2009, actuellement en master IAD 2eme année), sujet : locomotion pour un robot hybride roues-pattes
- Sylvain Koos (stage de master IAD réalisé en 2008, actuellement en thèse à l'ISIR), sujet : identification automatique et passage de la simulation à la réalité sur un quadrirotor. Ces travaux ont donné lieu à la publication suivante : **"Automatic system identification based on coevolution of models and tests. IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)."**
- Alexandre Ortiz (stage du master IAD réalisé en 2008, actuellement développeur C++), sujet : navigation d'un drone en mileu intérieur
- Peter Dardel (stage de master 1^{ere} année réalisé en 2008, décédé accidentellement fin 2008), sujet : structure d'aile et tenségrité
- Jérémie Rubinsztajn (stage de master 1^{ere} année réalisé en 2008, situation actuelle inconnue), sujet : évolution de contrôleurs pour le vol battu
- Guillaume Tatur (stage du master SDI réalisé en 2007, situation actuelle inconnue), sujet : génération d'un comportement de soaring pour un planeur par algorithmes évolutionnistes.
- Mathieu Schmitt (stage du master IAD réalisé en 2006, actuellement salarié chez Thales)
- Adrien Angeli (stage du master IAD réalisé en 2005, actuellement en post-doc à l'Imperial College à Londres) . Ces travaux de stage ont donné lieu à la publi-

cation suivante : **"2D simultaneous localization and mapping for micro aerial vehicles. In Proceedings of the European Micro Aerial Vehicles (EMAV 2006)"**

- Renaud Barate (stage du master IAD réalisé en 2005, actuellement en poste chez EDF après une thèse à l'ENSTA). Ces travaux ont directement donné lieu à une publication en revue internationale à comité de lecture : **"Design of a bio-inspired controller for dynamic soaring in a simulated UAV. Bioinspir. Biomim., 1 :76-88"**.
- Jean-Baptiste Mouret (stage du master IAD réalisé en 2005, actuellement maître de conférences à l'ISIR). Les travaux réalisé pendant son stage d'ingénieur en 2004 également sus ma direction ont donné lieu à publication : **"Evolution of neuro-controllers for flapping-wing animats. In Proceedings of the Journées MicroDrones, Toulouse"**.
- Laurent Muratet (stage réalisé du master IAD réalisé en 2003, actuellement chef d'entreprise) : A donné lieu à la publication suivante : **"A biomimetic reactive navigation system using the optical flow for a rotary-wing UAV in urban environment. In Proceedings of ISR2004, CD ROM ISR, Paris"**.

J'ai également encadré des **projets de fin d'étude d'élèves ingénieurs** :

- Steve N'Guyen (école : ECE, stage réalisé en 2005, actuellement en post-doc au collège de France après une thèse à l'SIR)
- Patrick Adigbli (école : centrale Paris, stage réalisé en 2007, a suivi ensuite un master au collège des ingénieurs, situation actuelle inconnue). A donné lieu à la publication suivante : **"Nonlinear Attitude and Position Control of a Micro Quadrotor using Sliding Mode and Backstepping Techniques. 7th European Micro Air Vehicle Conference (MAV07). Toulouse."**
- Julien de Farcy (école : ESIA, stage réalisé en 2006, actuellement ingénieur en informatique)

Je participe régulièrement à la formation des masters de première année en proposant des sujets en rapport avec mes activités de recherche dans le cadre d'UE de type TER (Travaux Encadrés de Recherche).

F PUBLICATIONS

F.1 JOURNAUX INTERNATIONAUX À COMITÉ DE LECTURE

- Hamdaoui, M., Chaskalovic, J., **Doncieux, S.**, and Sagaut, P. (2010). Using Multiobjective Evolutionary Algorithms and Data Mining Methods to Optimize Bird-like UAVs' Kinematics. Journal of Aircraft (à paraître).
- Angeli, A. and Filliat, D. and **Doncieux, S.** and Meyer, J.-A. (2008). A Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. Accepted for publication in IEEE Transactions On Robotics, Special Issue on Visual SLAM.
- Mouret, J.B. and **Doncieux, S.** (2008). MENNAG : a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. Evolutionary Intelligence. Vol 1 No 3 Pages 187–207.
- de Margerie, E., Mouret, J.-B., **Doncieux, S.**, and Meyer, J.-A. (2007). Artificial evolution of the morphology and kinematics in a flapping-wing mini UAV. Bioinspir. Biomim., 2 :65-82.
- Barate, R., **Doncieux, S.**, and Meyer, J.-A. (2006). Design of a bio-inspired controller for dynamic soaring in a simulated UAV. Bioinspir. Biomim., 1 :76-88.
- **Doncieux, S.** and Meyer, J.-A. (2005). Evolving PID-like neurocontrollers for nonlinear control problems. International Journal of Control and Intelligent Systems (IJCIS). Special Issue on nonlinear adaptive PID control, 33(1) :55-62.
- Muratet, L., **Doncieux, S.**, Brière, Y., and Meyer, J.-A. (2005). A contribution to vision-based autonomous helicopter flight in urban environments. Robotics and Autonomous Systems, 50(4) :195-209.

F.2 JOURNAUX NATIONAUX À COMITÉ DE LECTURE

- Landau, S. and **Doncieux, S.** and Drogoul, A. and Meyer, J.-A. (2002). SFERES : un framework pour la conception de systèmes multi-agents adaptatifs. Technique et Science Informatiques Hermès, publisher. Vol 21 No 4 Pages 427–446. Paris.

F.3 CONFÉRENCES INTERNATIONALES À COMITÉ DE LECTURE

- Mouret, J.-B. and **Doncieux, S.** (2010). Sferesv2 : Evolvin' in the Multi-Core World. IEEE Congress on Evolutionary Computation, 2010 (CEC 2010).
- Mouret, J.-B. and **Doncieux, S.** and Girard, B. (2010). Importing the Computational Neuroscience Toolbox into Neuro-Evolution—Application to Basal Ganglia. GECCO'10 : Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher .
- Koos, S. and Mouret, J.-B. and **Doncieux, S.** (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers. GECCO'10 : Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher .
- **Doncieux, S.** and Mouret, J.-B. (2010). Behavioral diversity measures for Evolutionary Robotics. IEEE Congress on Evolutionary Computation, 2010 (CEC 2010).
- Angeli, A. and Filliat, D. and **Doncieux, S.** and Meyer, J.-A. (2009). Visual topological SLAM and global localization. Proceedings of the International Conference on Robotics and Automation (ICRA).
- Koos, S. and Mouret, J.-B. and **Doncieux, S.** (2009). Automatic system identification based on coevolution of models and tests. IEEE Congress on Evolutionary Computation, 2009 (CEC 2009).
- Mouret, J.-B. and **Doncieux, S.** (2009). Evolving modular neural-networks through exaptation. IEEE Congress on Evolutionary Computation, 2009 (CEC 2009). (Best student paper award).
- Mouret, J.-B. and **Doncieux, S.** (2009). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. IEEE Congress on Evolutionary Computation, 2009 (CEC 2009).
- Mouret, J.-B. and **Doncieux, S.** (2009). Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. GECCO'09 : Proceedings of the 11th annual conference on Genetic and evolutionary computation ACM, publisher
- **Doncieux, S.** (2009). Evolutionary Algorithms as Exploration and Analysis Helper tools, Application to a Flapping Wings aircraft. IROS Workshop "Exploring New Horizons in Evolutionary Design of Robots" . Pages 19–25. Saint Louis, USA.
- **Doncieux, S.** and Mouret, J.-B. and Bredeche, N. (2009). Exploring New Horizons in Evolutionary Design of Robots. IROS Workshop "Exploring New Horizons in Evolutionary Design of Robots" . Pages 5–12. Saint Louis, USA.
- Angeli, A. and Filliat, D. and **Doncieux, S.** and Meyer, J.A. (2008). Real-Time Visual Loop-Closure Detection. in the proceedings of the IEEE International Conference on Robotics and Automation (ICRA).
- Angeli, A. and Filliat, D. and **Doncieux, S.** and Meyer, J.-A. (2008). Incremental vision-based topological SLAM. to appear in the proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS2008).

- Mouret, J.-B., **Doncieux, S.** (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. A paraître dans : Proceedings of the 10th International Conference on the Simulation of Adaptive Behavior (SAB).
- Angeli, A., Filliat, D., **Doncieux, S.**, and Meyer, J. (2008). Real-time visual loop-closure detection. A paraître dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).
- Adigbli, P., Grand, C., Mouret, J.-B., and **Doncieux, S.** (2007). Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques. In 7th European Micro Air Vehicle Conference (MAV07), Toulouse.
- de Margerie, E., Mouret, J.-B., **Doncieux, S.**, Meyer, J.-A., Ravasi, T., Martinelli, P., and Grand, C. (2007). Flapping-wing flight in bird-sized UAVs for the ROBUR project : from an evolutionary optimization to a real flapping-wing mechanism. In 7th European Micro Air Vehicle Conference (MAV07), Toulouse.
- **Doncieux, S.**, Mouret, J.-B., and Meyer, J.-A. (2007). Soaring behaviors in UAVs : 'animat' design methodology and current results. In 7th European Micro Air Vehicle Conference (MAV07), Toulouse.
- Mouret, J.-B., **Doncieux, S.**, and Meyer, J.-A. (2006). Incremental evolution of target-following neuro-controllers for flapping-wing animats. In Nolfi, S., Baldassare, G., Calabretta, R., Hallam, J., Marocco, D., Meyer, J.-A., Miglino, O., and Parisi, D., editors, From Animals to Animats : Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB), pages 606-618, Rome, Italy.
- **Doncieux, S.**, Landau, S., and Guelfi, N. (2004). EcoSFERES : A tool for the design of self-organized agent-based applications. In GECCO 2004 Late-Breaking Paper.
- **Doncieux, S.** and Meyer, J.-A. (2004a). Evolution of neurocontrollers for complex systems : alternatives to the incremental approach. In Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004).
- **Doncieux, S.** and Meyer, J.-A. (2004b). Evolving modular neural networks to solve challenging control problems. In Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004).
- Muratet, L., **Doncieux, S.**, and Meyer, J.-A. (2004). A biomimetic reactive navigation system using the optical flow for a rotary-wing UAV in urban environment. In Proceedings of ISR2004, CD ROM ISR, Paris.
- **Doncieux, S.** and Meyer, J.-A. (2003). Evolving Neural Networks for the Control of a Lenticular Blimp. Applications of Evolutionary Computing, EvoWorkshops2003 : EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM. Pages 626–637.

F.4 CONFÉRENCES NATIONALES À COMITÉ DE LECTURE

- Pinville, T., **Doncieux, S.** (2010). Automatic Synthesis of Working Memory Neural Networks with Neuroevolution Methods. In Cinquième conférence française de Neurosciences Computationnelles (Neurocomp'10).

– Angeli, A., Filliat, D., **Doncieux, S.**, and Meyer, J.-A. (2006). 2D simultaneous localization and mapping for micro aerial vehicles. In Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference.
– **Doncieux, S.**, Mouret, J.-B., Angeli, A., Barate, R., Meyer, J.-A., and de Margerie, E. (2006). Building an artificial bird : Goals and accomplishments of the ROBUR project. In Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference.

F.5 CONFÉRENCES NATIONALES AVEC SÉLECTION SUR RÉSUMÉ

– **Doncieux, S.**, Mouret, J., Muratet, L., and Meyer, J.-A. (2004). The ROBUR project : towards an autonomous flapping-wing animat. In Proceedings of the Journées MicroDrones, Toulouse.
– Mouret, J.-B., **Doncieux, S.**, Muratet, L., Druot, T., and Meyer, J.-A. (2004). Evolution of neuro-controllers for flapping-wing animats. In Proceedings of the Journées MicroDrones, Toulouse.

F.6 ÉDITION D'OUVRAGES SCIENTIFIQUES

– **Doncieux, S.**, Girard, B., Guillot, A., Hallam, J., Meyer, J.-A., Mouret, J.-B. (Eds.) From Animals to Animats 11 : 11th International Conference on Simulation of Adaptive Behavior, SAB 2010, Paris - Clos Lucé, France, August 25-28, 2010. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), Vol. 6226
– **Doncieux, S.**, Bredeche, N., Mouret J.-B. (Eds.) New Horizons in Evolutionary Robotics : post-proceedings of the 2009 EvoDeRob workshop, Studies in Computational Intelligence. Springer-Verlag (à paraître).

F.7 CHAPITRES D'OUVRAGES SCIENTIFIQUES

– **Doncieux, S.**, Mouret, J.-B., Bredeche, N., and Padois, V. (2010). Evolutionary Robo- tics : Exploring New Horizons. In Doncieux, S., Bredeche, N., and Mouret, J.-B., editors, New Horizons in Evolutionary Robotics : post-proceedings of the 2009 EvoDeRob workshop. Studies in Computational Intelligence. Springer-Verlag (à paraître).
– **Doncieux, S.** and Hamdaoui, M. (2010). Evolutionary Algorithms to Analyse and Design a Controller for a Flapping Wings Aircraft. In Doncieux, S., Bredeche, N., and Mouret, J.-B., editors, New Horizons in Evolutionary Robotics : post-proceedings of the 2009 EvoDeRob workshop. Studies in Computational Intelligence, Springer (à paraître).

– **Doncieux, S.** and Angeli, A. (2007). Objets volants miniatures : modélisation et commande embarquée. Hermes-Lavoisier, publisher. *Ch Navigation des drones par flux optique* Pages 305–328.
– Meyer, J.-A. and **Doncieux, S.** and Filliat, D. and Guillot, A. (2002). Evolutionary Approaches to Neural Control of Rolling, Walking, Swimming and Flying Animats or Robots. Biologically Inspired Robot Behavior Engineering Springer-Verlag, publisher. Pages 1–43.