République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département de l'Informatique

Thèse

Présentée pour l'obtention du diplôme de
Doctorat LMD en Informatique

Option : Techniques de l'Image et de l'Intelligence Artificielle

---

Étude de croissance de créatures artificielles résilientes

Présentée par : **Djouher AKROUR**

Devant le jury composé de :

Pr. Foudil CHERIF	Université de Biskra, Algérie	Président
Pr. NourEddine DJEDI	Université de Biskra, Algérie	Rapporteur
Pr. Hervé LUGA	Université de Toulouse 1, France	Co-Rapporteur
Pr. Ammar BALA	ESI, Alger, Algérie	Examinateur
Pr. Abdelouahab MOUSSAOUI	Université de Sétif, Algérie	Examinateur
Dr. Leila DJEROU	Université de Biskra, Algérie	Examinateur

Année Universitaire 2018/2019

Remerciements

Je tiens tout d'abord à adresser mes plus vifs remerciements à mon Directeur de recherche, Pr. DJEDI Noureddine, pour son encadrement, sa disponibilité, son aide et ses conseils avisés et précieux.

J'adresse également mes remerciements à Hervé LUGA, Professeur à l'université de Toulouse. En tant que co-Directeur français, il m'a guidé, encouragé et il m'a fait confiance durant toutes ces années de recherche. Je le remercie profondément, ainsi que Dr. Sylvain CUSSAT-BLANC et Dr. Stéphane SANCHEZ, pour leurs engagements, disponibilités, patiences et leurs judicieux conseils qui ont contribué à alimenter ma réflexion. Merci de m'avoir bien accueilli au sein de votre équipe de recherche REVA.

Je tiens à remercier les membres de jury pour avoir accepté de lire et d'évaluer mon travail de recherche : Pr. Foudil CHERIF, Dr. Leila DJEROU, Pr. Ammar BALA et Pr. Abdelouahab MOUSSAOUI.

Mes remerciements vont également au Professeur Yves DUTHEN pour sa compréhension, ses encouragements et ses commentaires judicieux qui m'ont permis d'aller de l'avant.

Une pensée particulière à Nesrine OUANNES, amie, disparue bien trop tôt. Elle, que j'ai tant voulu remercier à cette étape de ma thèse, restera gravée dans ma mémoire. Pleine d'énergie et d'ambition, elle n'a cessé de m'encourager et me soutenir.

Je voudrais remercier celles et ceux que la vie m'a permis de croiser, notamment mes ami · e · s de ma promotion pour leur aide et soutien. Je remercie également tous les professeurs qui m'ont enseigné et qui, par leurs compétences, m'ont encouragé et soutenu tout au long de mon parcours universitaire.

Finalement, j'aimerais remercier toute ma famille, à laquelle j'exprime toute ma gratitude. Je remercie particulièrement ma mère et mon père pour leur affection, amour inconditionnel, pour avoir toujours cru en moi et soutenu dans ce projet de thèse. Merci à mes chers frères Rabah et Yacine pour leurs présences et encouragements. Je remercie énormément mes grands-parents, mes oncles et mes tantes, particulièrement Ourida, Ali, Yahia, Kader, Mourad et Youcef. Un immense merci à Fatiha pour son appui moral et qui, durant toutes ces années, n'a cessé de m'aider et me soutenir.

À mes très chers parents.

Résumé

Pour être totalement autonomes, les robots doivent être résilients pour pouvoir se remettre des dommages et fonctionner pendant une longue période sans assistance humaine. La robotique évolutionnaire aspire à atteindre cet objectif en se reposant sur les algorithmes évolutionnaires afin d'évoluer conjointement les corps et les contrôleurs des robots. Dans cette thèse, nous décrivons deux contributions pour la conception automatique des créatures robotiques virtuelles. Dans un premier temps, nous nous intéressons à l'aspect réalisme des simulations. Pour cela, nous proposons une approche qui permet de générer automatiquement des robots modulaires réalistes en s'appuyant sur un simulateur crédible de robotique. Nous démontrons la capacité de notre modèle à générer une variété de robots capables de se comporter de manière réaliste. En fait, le comportement global d'un robot se fait par la synergie produite par la coopération des modules le composant. Ainsi, il peut réaliser des locomotions intéressantes tout en parcourant une longue distance. Dans une deuxième étape, nous avons proposé une nouvelle approche pour permettre aux robots d'acquérir des capacités de résilience. Nous montrons que lorsque les robots sont délibérément confrontés à des pannes durant le temps d'apprentissage, le processus d'évolution permet d'optimiser et de générer des robots adaptables dont le comportement est moins affecté par les dommages.

Mots clés : Robotique évolutionnaire, Robots modulaires, Résilience, Réalisme, Contrôle, Apprentissage.

Abstract

In order to be fully autonomous, robots have to be resilient so that they can recover from damages and operate for a long period of time with no human assistance. Evolutionary robotics aims to achieve this goal by using evolutionary algorithms in order to co-evolve bodies and controllers of robots. In this thesis, we describe two contributions for the automatic design of virtual robotic creatures. Firstly, we get interest in the realism aspect of simulations. To achieve this, we propose an approach that automatically generates realistic modular robots relying on a credible robotics simulator. We demonstrate the ability of our model to generate a variety of robots that can behave realistically. In fact, the global behavior of a robot is done by the synergy produced by the cooperation of the modules that make it up. Thus, the robot can achieve interesting locomotion while traveling a long distance. Secondly, we propose a new approach that allow robots to acquire resilience capabilities. We show that when robots are deliberately faced with failures during learning time, the evolution process can optimize and generate adaptive robots whose behavior is less affected by damage.

Key words : Evolutionary robotics, Modulars robots, Resilience, Realism, Control, Learning.

الملخص

لكي تكون الروبوتات ذاتية الاستقلالية, يجب أن تكون مرنة(résilient) بحيث يمكنها التعافي من الأضرار والعمل لفترة طويلة من الزمن دون مساعدة بشرية. يطمح علم الروبوتات التطوري إلى تحقيق هذا الهدف بالاعتماد على الخوارزميات التطورية من أجل التطور المشترك لهيئات و أنظمة تحكم الروبوتات. في هذه الأطروحة نصف مساهمتين للتصميم التلقائي للكائنات الروبوتية الافتراضية. أولا, نوجه اهتمامنا إلى جانب واقعية عمليات المحاكاة. تحقيقا لذلك, نقترح وضع نهجًا يولد تلقائيًا روبوتات وحدوية(robots modualires) واقعية اعتمادا على محاكي روبوتات ذات مصداقية. نظهر بعدها قدرة نموذجنا على إنشاء مجموعة متنوعة من الروبوتات التي يمكنها التصرف بشكل واقعي. في الحقيقة, السلوك العام للروبوت ينتج من خلال التآزر الناتج عن تعاون الوحدات التي تشكل هيكله. وبالتالي يمكنه تحقيق حركات مثيرة للاهتمام من أجل قطع مسافات كبيرة. ثانيا, اقترحنا نهجا جديدًا يسمح للروبوتات الحصول على قدرات المرونة. نظهر أنه عندما تواجه الروبوتات أضرارا عمدا أثناء وقت التعلم, عملية التطور تسمح بتوليد روبوتات قابلة للتكيف حيث نضام تحكمها يكون أقل تأثرا بالأضرار.

الكلمات المفتاحية: علم الروبوتات التطوري, الروبوتات الوحدوية, المرونة, المحاكاة الواقعية, أنظمة التحكم, التعلم.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Contexte et motivations

Durant ces dernières années, le développement des systèmes robotiques n'a cessé de progresser. Avec les nouvelles technologies de pointe, ils deviennent de plus en plus performants. En fait, aujourd'hui, les robots sont ancrés dans divers secteurs d'activités, tels que l'industrie, le secteur médical, spatial et militaire. Ils exécutent les tâches pour lesquelles ils sont programmés et rendent ainsi un grand service à l'humanité. À titre d'exemple, dans les usines, ils effectuent des manipulations répétitives, pénibles et parfois dangereuses, mais avec précision et rapidité.

Par ailleurs, les robots peuvent intervenir dans des milieux hostiles, pour venir en aide lors des catastrophes naturelles ou dans le cadre des explorations sous marines ou extraterrestres. Dans ce cas, les robots sont capables de résister à des conditions dangereuses, mais inévitablement ils sont sujets aux endommagements physiques. Il faut alors être résilient et pouvoir s'adapter rapidement. Cela signifie que les robots doivent se dépanner d'eux-mêmes lors des dommages et prévenir tout événement indésirable à l'effet d'exécuter leurs missions dans les délais et en toute sécurité. Néanmoins, concevoir de tels robots demeure une question extrêmement difficile au sein de la communauté des chercheurs, en raison de la dynamique non linéaire et l'interaction complexe entre les parties impliquées dans la réalisation de la mission. En outre, les conditions environnementales dans lesquelles les robots fonctionnent sont a priori non connues et éventuellement changeantes et non contrôlées.

Malgré leurs précisions, les robots conventionnels manquent de flexibilité et d'adaptabilité. Leurs conceptions physiques et contrôleurs sont destinés à exécuter une tâche unique et précise. Dès qu'un événement inattendu se produit, que se soit aux niveaux de la morphologie ou de l'environnement, le robot devient inutile. Par conséquent, les robots sont limités dans leurs capacités à réaliser des tâches par leurs

morphologies. Afin de surmonter cette problématique, les chercheurs s'orientent vers la robotique modulaire, là où le robot peut modifier de morphologie à la demande pour mieux convenir aux nouvelles conditions morphologiques ou environnementales.

Les robots modulaires sont donc plus adaptables. Ils ont l'avantage d'apporter versatilité, robustesse et un faible coût de fabrication. Les robots modulaires ont été spécialement conçus pour pouvoir agir dans des environnements inconnus, en prenant à chaque fois une nouvelle configuration pour traverser des trous étroits ou pour s'infiltrer dans des décombres et ainsi de suite. Les modules d'un robot modulaire sont homogènes ou hétérogènes avec des fonctionnalités variables. Ils ne vont pas fonctionner individuellement, mais la fonctionnalité de chacun va agir en conjonction avec les autres afin de coopérer et produire une synergie, qui va par la suite rendre le robot capable d'exécuter sa mission.

Afin de concevoir de tels robots, on s'oriente souvent vers la robotique évolutionnaire. Il s'agit d'un domaine de recherche visant la conception automatique de robots autonomes et adaptables en utilisant des mécanismes évolutionnaires. En fait, elle s'est beaucoup inspirée du domaine de la vie artificielle, particulièrement des travaux précurseurs de Karl Sims [Sims, 1994b]. Ces derniers visaient la conception des créatures artificielles via des évolutions conjointes des morphologies et des contrôleurs. Cela signifie que les paramètres définissant à la fois la structure morphologique et le système de contrôle sont optimisés ensemble durant le processus d'évolution.

En effet, les techniques évolutionnaires, inspirées de l'évolution naturelle proposée par Charles Darwin, font émerger des créatures aptes à exécuter leurs tâches, sans en avoir a priori des connaissances concernant leurs environnements, morphologies ou contrôleurs. Au fur et à mesure que les générations se succèdent, les créatures vont apprendre d'elles-mêmes quelles morphologies et quels comportements seraient les plus adéquats afin d'atteindre leurs objectifs. Par ailleurs, les techniques utilisées pour définir les morphologies et les systèmes de contrôle sont généralement bio-inspirés. Par conséquent, les créatures, en l'occurrence les robots virtuels, vont ressembler partiellement aux êtres vivants (dotés d'autonomie, de capacités de résilience, de capacités d'apprentissage, etc.) ou même en sortie du commun.

Impliquer la morphologie dans le système d'évolution, permet d'engendrer des robots plus créatifs et adaptables. Le fait de modifier continuellement les morphologies exige l'utilisation d'une plate-forme de simulation crédible. Cela va faciliter la conception du robot ainsi que son contrôleur. Après la terminaison du processus d'évolution ou d'optimisation, les meilleurs spécimens de robots peuvent être fabriqués et reproduits dans le monde réel.

1.2 Problématique et objectifs

Comment pourrons-nous concevoir des robots modulaires résilients par l'utilisation des évolutions conjointes des morphologies et des contrôleurs ? Dans le but de répondre à cette question de recherche, il est nécessaire d'atteindre un certain nombre d'objectifs :

- Concevoir un modèle qui permet de générer des morphologies complexes de robots modulaires. Dans ses travaux de référence, Karl Sims a réussi à développer un système de génération automatique de créatures artificielles extrêmement intéressantes. Nous allons alors nous inspirer pour en élaborer notre propre modèle. Les morphologies des créatures, en l'occurrence des robots simulés, vont être donc impliquées dans le processus d'évolution. Nous aspirons qu'elles soient développées par morphogenèse (croissance cellulaire) étant une méthode de codage proche des paradigmes biologiques.
- Concevoir un système de contrôle performant qui serait en mesure de contrôler la morphologie du robot afin de répondre à l'objectif. Le contrôleur doit être doté d'un système de communication des données compris de par les robots et les modules de robots.
- Mettre en œuvre des méthodes permettant aux robots d'acquérir des capacités de résilience. En effet, la résilience peut être vue comme un premier jalon nécessaire à l'acquisition d'un comportement autonome. Actuellement, la conception de robots résilients est un problème majeur au sein de la communauté robotique. Les approches proposées dans la littérature ont été élaborées et testées sur des robots conventionnels. Néanmoins, une morphologie figée de robots restreint l'espace des solutions possibles et affecte l'adaptabilité des contrôleurs. Il serait alors intéressant de coévoluer des robots modulaires qui seront plus adaptables et résilients.
- Étant l'objectif principal de cette thèse, la synthèse automatique de créatures artificielles résilientes, il est nécessaire de pouvoir choisir une plateforme de simulation crédible dans laquelle nous allons intégrer notre modèle ainsi que notre système d'évolution. La plateforme doit être crédible et fournit un ensemble de capteurs réalistes. En fait, nous envisageons de concevoir des créatures tridimensionnelles réalistes. Le réalisme est toujours important peu importe l'objectif des simulations.

1.3 Contributions

Afin de répondre à certains des défis décrits précédemment, cette thèse apporte deux contributions majeures ; (i) la mise en œuvre d'un modèle de génération de

robots modulaires réalistes et (ii) la proposition d'une nouvelle approche pour la conception de robots résilients.

La première contribution consiste en la proposition d'un modèle de génération de robots réalistes par la coévolution des morphologies et des contrôleurs. Cette contribution insiste sur l'aspect réalisme des simulations. Ainsi nous avons utilisé la plate-forme de simulation multi-robots Gazebo qui permet des simulations crédibles. Nous avons proposé donc un modèle qui permet de générer une variété de robots modulaires hétérogènes réalistes. Ceux-ci ont appris à trouver des stratégies créatives de locomotions, dont le but de parcourir les plus grandes distances.

La seconde contribution de cette thèse concerne la résilience. Pour cela, nous avons proposé une nouvelle approche qui permet aux robots de se rétablir des dommages et de poursuivre leurs déplacements tout en gardant la même direction. Nous avons fait évoluer les morphologies des robots avec leurs contrôleurs pour qu'ils soient en mesure de s'adapter aux pannes causées délibérément durant le temps d'évaluation. Ainsi, au cours des générations, les robots vont apprendre à développer des morphologies adaptables et deviennent en quelque sorte résilients.

1.4 Structure de la thèse

Dans cette section, nous proposons un résumé des différents chapitres composant cette thèse.

Le **second chapitre** présente un état de l'art concernant la robotique évolutionnaire. Il introduit d'abord les principes de base de l'évolution artificielle, ensuite il donne un aperçu des travaux de référence réalisés dans le domaine en question. Un accent particulier est mis sur la conception de robots virtuels par le biais de la coévolution des morphologies et des contrôleurs. Plus tard, le chapitre aborde les raisons pour lesquelles il est important d'avoir des structures corporelles qui évoluent, la capacité de résilience ainsi que les défis actuels de la robotique évolutionnaire.

Ensuite, le **troisième chapitre** sera consacré aux robots modulaires, leurs types et applications. Certains des robots conçus récemment seront décrits et comparés afin de retenir ce qui caractérise chacun par rapport aux autres et définir les caractéristiques à impliquer dans la conception de nos robots modulaires.

Dans le **quatrième chapitre**, nous décrirons les simulateurs robotiques et soulignons leurs importances dans le domaine de la robotique en général, et par rapport à notre étude en particulier. Nous discuterons ensuite les problèmes de *Reality Gap* ou la difficulté de pouvoir transférer les simulations en monde réel. Après une étude comparative de quelques plates-formes de simulations, nous effectuons un choix d'un simulateur qui semble bien répondre à nos besoins durant la suite de nos travaux.

Le **cinquième chapitre** traite le modèle de génération de robots modulaires proposé dans le cadre de nos études. Premièrement, les robots et leurs conceptions en simulation sont présentés. Par la suite, la manière dans laquelle notre modèle est intégré dans la plate-forme de simulation est décrite. En fin, les difficultés rencontrées à cet égard ainsi que les méthodes adoptées pour faire face, sont présentées.

Dans le **sixième chapitre**, nous exposons les expériences que nous avons réalisées ainsi que les résultats obtenus. Dans un premier temps, nous présentons une étude concernant la tâche de locomotion et examinons le rapport entre les tailles des robots (nombre de modules) et leurs capacités à parcourir de longues distances. Dans un deuxième temps, dans ce chapitre, nous présentons notre approche de génération de robots résilients.

Le **septième chapitre** évalue le modèle que nous avons proposé et le situe par rapport aux différents travaux de la littérature. Le chapitre présente également l'apport de cette thèse ainsi que les limites du modèle élaboré.

Enfin, le **huitième chapitre** conclut la thèse par un résumé des contributions et présente certaines perspectives potentielles.

Chapitre 2

La robotique évolutionnaire

2.1 Introduction

La robotique évolutionnaire vise à la conception automatique de robots autonomes adaptables, afin de pouvoir accomplir des tâches diverses et interagir avec des environnements changeants et peu connus. Ce domaine repose sur l'utilisation combinée du calcul évolutionnaire et des techniques bio-inspirées pour optimiser des politiques de contrôle, des structures corporelles de robots, et aussi des stratégies de comportements coordonnés pour des essaims de robots.

Dans ce chapitre, d'abord, nous introduirons les principes de base de l'évolution artificielle. Ensuite, nous donnerons un aperçu des travaux de référence réalisés dans le domaine en question (section 2.3.2). Un accent particulier est mis sur la conception de robots virtuels par le biais de la coévolution des morphologies et des contrôleurs. Nous aborderons plus tard les raisons pour lesquelles il est important d'avoir une structure corporelle qui évolue (section 2.3.3), la capacité de résilience chez les robots (section 2.4) et les défis actuels de la robotique évolutionnaire (section 2.5).

2.2 Rappel sur l'évolution artificielle

2.2.1 Les algorithmes évolutionnaires

Les Algorithmes Évolutionnaires (AE) sont des techniques de recherche stochastique. Ils ont été souvent utilisés pour résoudre des problèmes d'optimisation, [Mitchell, 1998, Yu and Gen, 2010] là où l'objectif est de trouver parmi un grand nombre de possibilités de solutions (espace de recherche) la solution optimale. Les AE imitent le processus de l'évolution biologique proposé par Charles Darwin en 1853 [Darwin, 1859]. De par son principe de sélection naturelle, le processus permet la

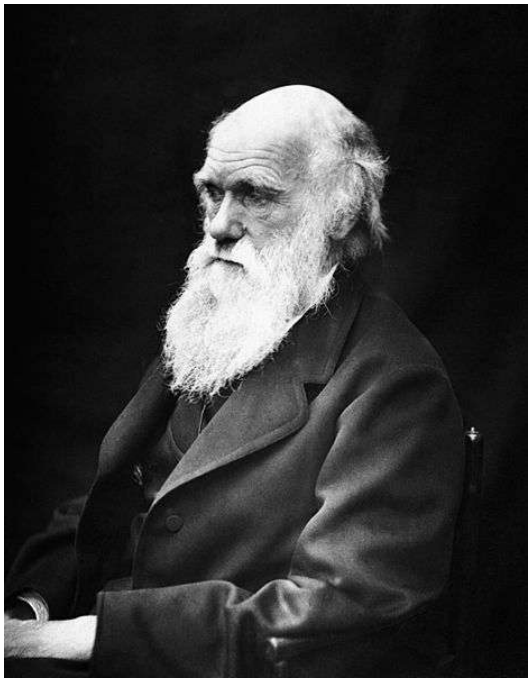


FIGURE 2.1 – Charles Darwin (1809 - 1882)

survie du plus apte « *Survival of the fittest* ». Ainsi les organismes microscopiques (p. ex. virus) et les insectes fragiles (p. ex. fourmis), ont pu vivre sur Terre pendant des millions d'années, non pas par leurs forces physiques, mais pour leurs extraordinaires capacités d'adaptations aux variations de l'environnement.

Le concept principal de cette catégorie d'algorithmes est de faire évoluer une population d'individus dont l'optique d'en trouver les meilleurs. En effet, les individus représentent des solutions à un problème donné ou des points dans un espace de recherche. Au sein d'une population, les individus sont en constante compétition afin de survivre et de se reproduire. Les meilleurs seront alors sélectionnés pour faire propager leurs informations génétiques aux générations futures. Cela est fait par l'utilisation des opérateurs génétiques de croisement et de mutation. L'application stochastique de la sélection et des mécanismes de reproduction au fil des générations va permettre une amélioration de la qualité des individus, et donc converger vers les solutions souhaitées. La figure 2.2 illustre le principe général de fonctionnement d'un algorithme évolutionnaire.

Nous trouvons dans la littérature plusieurs algorithmes qui font partie des AE. Par exemple les Stratégies d'Évolution (SE) [Michalewicz, 1996], les Algorithmes Génétiques (AG) [Holland, 1992], la Programmation Évolutionnaire (PE) [Fogel, 1995] et la Programmation Génétique (PG) [Koza, 1992]. Tous ils partagent les mêmes principes à l'exception de la manière de la représentation génétique des individus et des mécanismes de sélection et de reproduction. Cependant, ce sont les AG, présentés par Holland dans les années 1970s, qui ont été les plus utilisés. Les

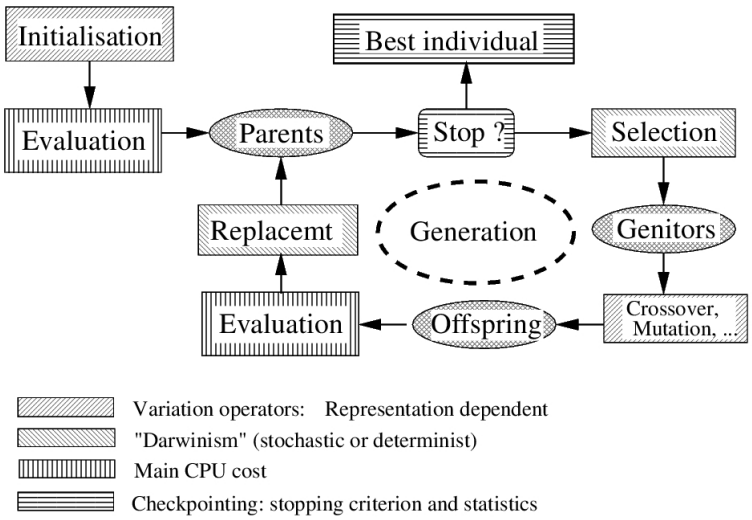


FIGURE 2.2 – Squelette d'un algorithme évolutionnaire par Marc Schoenauer. Image extraite de [Keijzer et al., 2001]

AE se caractérisent par des notions et des étapes essentielles que nous proposons de décrire ci-dessous :

La représentation des individus

Chaque individu représente une solution probable pour un problème donné. Il comporte à l'image des êtres vivants, un génotype et un phénotype. Le génotype comprend les caractéristiques génétiques de l'individu (ADN), tandis que le phénotype représente les caractères physiques observables des individus et il est déterminé par expression du génotype. Les AG proposent une représentation du génotype sous forme binaire, mais d'autres représentations restent possibles, par exemple, une structure de données complexe avec une taille qui peut varier. En fait, cela dépend étroitement de la nature du problème à résoudre.

Le codage, ou le mappage du génotype au phénotype représentent le processus de développement de l'individu à partir des informations disponibles dans le génotype. Selon le degré de réutilisation de ces informations, on a classifié deux types de codages ; directe et génératif. [Tarapore and Mouret, 2015] Le codage direct permet un mappage déterministe. Chaque gène du génotype est utilisé au maximum une fois pour déterminer un seul trait du phénotype. Dans le cadre de cette thèse, nous allons adopter ce type de codage, car il est facile à implémenter. Le codage génératif, quant à lui, est plus proche des paradigmes des systèmes biologiques complexes. Les gènes peuvent être utilisés plusieurs fois pour affecter le phénotype. Danesh Tarapore et Jean-Baptiste Mouret [Tarapore and Mouret, 2015] l'on définit comme étant « le codage qui mappe le génotype au phénotype grâce à un processus de croissance d'un

génotype simple dans un espace de recherche de faible dimension à un phénotype complexe dans un espace de grande dimension ».

L'initialisation des individus

La première étape du processus d'évolution consiste à choisir parmi un espace de recherche un ensemble fini d'individus (une population d'individus). Il est toutefois important de pouvoir générer une population représentant tout l'espace de recherche, afin qu'elle soit la plus diversifiée possible. Au fait, le choix des individus est fait de manière stochastique, cependant il est possible d'ajouter à la population initiale les meilleurs individus émergés d'une précédente expérience (p. ex. [Bongard et al., 2015]). Cela permet de se rapprocher rapidement des solutions optimales.

L'évaluation

L'évaluation est une étape très importante dans le processus d'évolution. Elle permet de déterminer les qualités des individus, et donc leurs probabilités de survie et de reproduction. Pour cela on utilise une fonction *objectif* (ou fonction *fitness*). Celle-ci est une fonction mathématique prédéterminée qui va décrire, dans une certaine mesure, la tâche à réaliser ou l'objectif de l'individu. Il est donc important de bien définir cette fonction afin d'obtenir les solutions qui répondent au mieux aux attentes des ingénieurs. Par exemple, en robotique, utiliser comme fonction *objectif* la distante euclidienne plutôt que la distance parcourue, est préférable si le but du robot est d'atteindre un point déterminé dans l'environnement. En fait, afin de développer des comportements robustes, on peut évaluer les robots de multiples fois en les exposant à différents environnements, à chaque fois, avec de petites variations et perturbations. (p. ex. [Cappelle et al., 2016, Bongard et al., 2015])

La sélection

La sélection est l'opérateur qui permet de choisir parmi tous les individus d'une population ceux qui vont survivre et reproduire. En fait, mieux l'individu s'adapte à l'environnement (lors de l'évaluation), plus il aura de chance d'être sélectionné. Ainsi, nous assurons, au cours des générations, la conservation des individus les plus performants et l'élimination progressive de ceux peu adaptés. Toute fois, rien ne nous assure la sélection de tels individus. En outre, même sélectionnés, leurs accouplements ne garantissent pas une progéniture de même qualité. De ce fait, toute solution émergée à un moment donné peut être perdue. Dans ce sens, Kenneth De Jong en 1975 [De Jong, 1975] a introduit le concept d'élitisme, qui sert à forcer de garder un nombre de meilleures solutions pour les prochaines générations. De nombreux

chercheurs ont trouvé que l'élitisme améliore significativement la performance des AG. [Mitchell, 1998]

On a proposé dans la littérature plusieurs méthodes de sélection. Nous citons à titre d'exemple la sélection par tournoi. Elle consiste à choisir de manière aléatoire un ensemble d'individus (participants) sans tenir compte de leurs *fitness*, ensuite prendre le meilleur pour être parent. Cette méthode donne une chance aux individus de piètre qualité d'être sélectionnés. En fait, les opérateurs de sélection introduisent une pression sélective[1], car ils comptent sur les qualités des individus pour déterminer la probabilité de leur sélection. Donc, plus la pression est forte, moins les individus peu adaptés seront sélectionnés.

Le croisement et la mutation

Après la sélection, les individus vont se reproduire via un ensemble d'opérateurs génétiques, à savoir, le croisement et la mutation. Ces opérateurs permettent de créer les nouveaux individus qui composeront la population de la prochaine génération. Le croisement permet de combiner les génotypes de deux parents et d'en produire un ou deux enfants. Intuitivement, cet opérateur a tendance à mener la population à proximité de ses meilleurs individus (exploitation). La mutation, quant à elle, permet d'introduire des variations au niveau du génotype d'un individu enfant nouvellement crée. Les petites variations apportées par cet opérateur engendrent de nouvelles caractéristiques, et donc assurer de maintenir une certaine diversité génétique. La mutation se concentre sur les individus faibles en explorant d'autres zones dans l'espace de recherche (exploration). [Fulcher, 2008] En fait, les méthodes d'application de croisement et de mutation dépendent essentiellement de la représentation génétique des individus.

Le critère d'arrêt

Les nouveaux individus générés, à côté des élites de la génération précédente, forment la population de la nouvelle génération. Les étapes de l'algorithme évolutionnaire sont alors répétées itérativement, à chaque fois avec une nouvelle population, jusqu'à la convergence vers les solutions souhaitées. Pour cela, on définit un critère d'arrêt qui va permettre de faire arrêter le processus d'évolution dès que les solutions presque optimales émergent. Cependant, ce n'est pas le cas des problèmes complexes où les solutions ne sont pas connues. On peut alors ne pas définir un critère d'arrêt et interrompre le processus d'évolution après un certain nombre de générations, ou lorsque les valeurs des *fitness* stagnent et ne s'améliorent plus.

1. La pression sélective entraîne la sélection naturelle.

2.2.2 Le multi-objectif

Lorsqu'il est nécessaire d'optimiser plus d'un critère à la fois, nous devons utiliser des Algorithmes Évolutionnaires Multi-Objectifs (AEMO). Ces derniers viennent pour optimiser, en maximisation ou en minimisation, des critères ou des objectifs contradictoires (par exemple, l'énergie, le coût et la vitesse d'un robot). Dans ce cas, on ne cherche pas à trouver une seule solution optimale, mais produire un compromis de solutions multiples à partir desquelles une solution peut être sélectionnée auprès du décideur. Plusieurs approches ont été alors proposées dans ce contexte, dont certaines reposent sur le principe de l'optimum de *Pareto* [Pareto, 1964]. Nous citons par exemple Non-dominated Sorting Genetic Algorithm (NSGA-II) [Deb et al., 2002], Strenght Pareto Evolutionary Algorithm (SPEA2) [Zitzler et al., 2001] et Indicator Based Evolutionary Algorithm (IBEA) [Zitzler and Künzli, 2004].

Dans le cadre de cette thèse, nous nous intéressons à l'algorithme NSGA-II. Ce dernier construit, de manière générale, une population d'individus concurrents, classe et trie chaque individu selon son niveau de non-domination, applique ensuite les opérations évolutives pour créer un nouveau groupe de descendants. Il combine par la suite les parents et la progéniture avant de partitionner cet ensemble combiné d'individus en fronts. [Coello et al., 2007] Afin de conserver la diversité dans la population, l'algorithme estime la densité de chaque individu en calculant la distance d'encombrement (*crowding distance* en anglais).

Principe de Pareto

Contrairement aux algorithmes d'optimisation mono-objectif, les AEMO produisent un ensemble de solutions non dominées, nommé le *front de Pareto*. Dans ce dernier, il existe un équilibre dont on ne peut pas améliorer un critère sans détériorer au moins un des autres.

Une solution avec de multiples objectifs est dite non dominée (ou dominante) par rapport à une autre, si seulement si elle n'est pas inférieure à la seconde sur tous les objectifs, et elle est strictement meilleure pour au moins un objectif. Avec ce principe de dominance, on peut donc ordonner les solutions, sélectionner ensuite celles qui vont se reproduire. La figure 2.3 illustre un exemple explicatif du principe du *front de Pareto*.

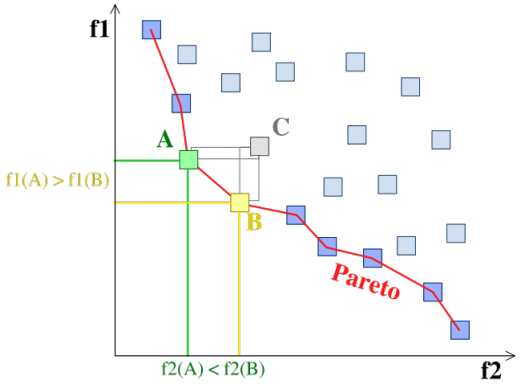


FIGURE 2.3 – Sur cette figure, on cherche à minimiser les deux objectifs $f1$ et $f2$. Les solutions non dominées sont les points situés sur le front de Pareto, indiqué en rouge. Le point C n'est pas sur le front de Pareto parce qu'il est dominé par les points A et B (graphique issu de Wikipédia).

2.3 L'évolution des créatures artificielles

2.3.1 Créatures artificielles

Synthétiser des créatures artificielles pouvant imiter des organismes naturels est l'un des objectifs principaux de la vie artificielle. Au fait, la vie artificielle est un domaine très important du point de vue des chercheurs s'intéressant à étudier et reproduire la vie, à tous niveaux de complexité et d'organisation. Christopher Langton, le fondateur de cette discipline en 1987, l'a définie comme suit : « La vie artificielle est l'étude des systèmes construits de mains d'hommes qui exhibent des comportements caractéristiques des systèmes naturels vivants. Elle vient en complément des sciences biologiques traditionnelles qui analysent les organismes vivants, en tentant de synthétiser des comportements semblables au vivant au sein d'ordinateurs et d'autres substrats artificiels. En étendant les fondements empiriques sur lesquels la biologie est basée au-delà de la vie à base de carbone qui a évolué sur Terre, la vie artificielle peut contribuer à la biologie théorique en positionnant la vie telle que nous la connaissons au sein d'un espace plus large : la vie telle qu'elle pourrait être ». [Langton, 1995] En d'autres termes, par utilisation des mécanismes principalement inspirés des systèmes biologiques, on tente de non seulement étudier et reproduire des formes de vie, mais également d'en créer de nouvelles.

Une créature artificielle est une entité virtuelle qui possède une morphologie [2] et un comportement simulé dans un environnement virtuel, en deux ou en trois

2. La morphologie peut être vue comme étant l'entité qui comprend la forme corporelle (type des membres et leurs liaisons, etc.), les propriétés physiques, les types des capteurs (les yeux, les oreilles, le nez, la peau, etc.) ainsi que leurs positionnements par rapport au corps de la créature. [Pfeifer and Bongard, 2006]

dimensions. Afin qu'elle soit réactive à l'environnement, la créature artificielle est dotée de capacités sensorimotrices. Selon son état interne et l'état externe perçus par le biais de capteurs, son cerveau (contrôleur) prend des décisions localement, suggère ensuite des actions et des mouvements appropriés en contrôlant les effecteurs. En fait, il est souhaitable pour une créature artificielle de posséder toutes les caractéristiques du vivant, entre autres l'apprentissage, l'évolution, l'adaptation, l'autonomie, l'autoproduction et l'autoréparation. [Lassabe, 2008] Toutefois, cela reste complexe et difficile, particulièrement si les créatures générées sont destinées pour être conçues physiquement en monde réel. Dans ce cas, l'objectif est de concevoir des robots virtuels et/ou physiques exhibant, au moins, certaines des propriétés des systèmes naturels vivants. Ici, nous nous trouvons dans le domaine de la robotique évolutionnaire.

2.3.2 Travaux de références

Les travaux précurseurs réalisés par Karl Sims [Sims, 1994b, Sims, 1994a] au début des années 1990 ont ouvert la voie à un axe important de recherche en vie artificielle. Il s'agit de générer automatiquement des créatures artificielles par évolution. En effet, ceci représente un point essentiel pour la robotique évolutionnaire, car ça permet d'optimiser des structures de robots virtuels avec des contrôleurs associés avant qu'ils ne soient fabriqués en monde réel. En outre, le processus d'évolution nous permet de faire émerger des systèmes de robots créatifs auxquels on ne s'y attend pas.

Sims était le premier à faire évoluer, de manière jointe, les structures corporelles des créatures et leurs contrôleurs associés. Il a fait évaluer et tester ses créatures dans un environnement de simulation soumis aux lois de la physique newtonienne. À ce moment-là, les créatures avaient la particularité de se déplacer de manière réaliste et étaient visuellement impressionnantes. Les spécialistes du domaine considèrent que cela était faisable grâce, non seulement, à la puissance de calcul [3] et à la précision du moteur physique [Taylor and Massey, 2001], mais aussi au contrôleur utilisé [Lassabe, 2008]. Par la suite, en se basant sur les principes d'évolution utilisés par Sims, un grand nombre de chercheurs ont proposé différents algorithmes et stratégies pour évoluer des morphologies de créatures, des systèmes de contrôle, ou les deux à la fois. Dans certains travaux, on a réussi à fabriquer des robots physiques par une impression en trois dimensions (3D) des meilleurs spécimens émergés en simulation. [Lipson and Pollack, 2000, Hornby and Pollack, 2001] En fait, le but de ces recherches est de pouvoir synthétiser des créatures de plus en plus complexes, et aussi de faire progresser la robotique évolutionnaire avec la conception de robots qui possèdent

3. Sims a fait tourner son programme dans la machine parallèle *Connexion Machine CM-5*. A l'époque, elle était une machine puissante.

des propriétés de vivant. Dans ce qui suit, nous suggérons de présenter certains de ces travaux.

Évolution de la morphologie

Faire évoluer des morphologies seules sans comportements (ou avec un minimum de comportement) a été beaucoup considéré dans le domaine de la vie artificielle. Les chercheurs, dans ce cas, visent essentiellement l'étude de la modélisation des processus de développement naturel (p. ex. [Eggenberger, 1997]), et ce en utilisant des systèmes de codage génératif à différents niveaux d'abstraction. Les approches de ce type de codage peuvent être par exemple des approches grammaticales (de haut niveau) tel que les *L-systems* [Lindenmayer, 1968] ou des approches qui se basent sur la chimie cellulaire (de bas niveau) tels que les Réseaux de Régulation Génétique (GRN) [Banzhaf, 2003]. [Tarapore and Mouret, 2015]

Les *L-systems* ont été proposés en premier lieu pour modéliser le contrôle morphogénétique cellulaire [Lindenmayer, 1968], ensuite ils ont été approfondis et mis en œuvre graphiquement pour décrire formellement des structurations de plantes [Prusinkiewicz and Lindenmayer, 1990]. Ces dernières, par une application répétée des règles d'une grammaire donnée, avaient des formes et des structures récursives. Mais ce qui a pris de l'ampleur ces dernières années est l'utilisation des GRN pour la modélisation de développement biologique des organismes multicellulaires. Dans ce cas, on tente par morphogenèse et embryogenèse simuler la croissance des formes à partir d'une seule cellule souche. Le but est de développer des créatures qui possèdent à la fois une auto-organisation et une architecture déterminée implicitement. [Doursat et al., 2012] En robotique, Sylvain Cussat-Blanc et Jordan Pollack [Cussat-Blanc and Pollack, 2014] ont pu faire générer de véritables morphologies de robots modulaires [4] en se basant sur un GRN et sur le développement cellulaire.

Évolution du contrôleur

Durant ces deux dernières décennies, on a proposé de nombreux travaux concernant l'évolution des contrôleurs pour des morphologies fixes de robots. En effet, selon la mission à remplir, l'ingénieur peut intuitivement imaginer la structure corporelle adéquate. Il peut ainsi se focaliser essentiellement et uniquement sur la conception du contrôle. Toutefois, cela devient difficile avec les tâches les plus complexes. Les chercheurs se sont intéressés donc à réaliser différentes tâches entre autres, la locomotion de robots à pattes (p. ex. [Chernova and Veloso, 2004, Jakobi, 1998]), la navigation et l'évitement d'obstacles pour des robots sur roues (p. ex. [Ram et al., 1994, Meeden, 1996, Nelson et al., 2004]) et encore la poursuite et l'évasion des

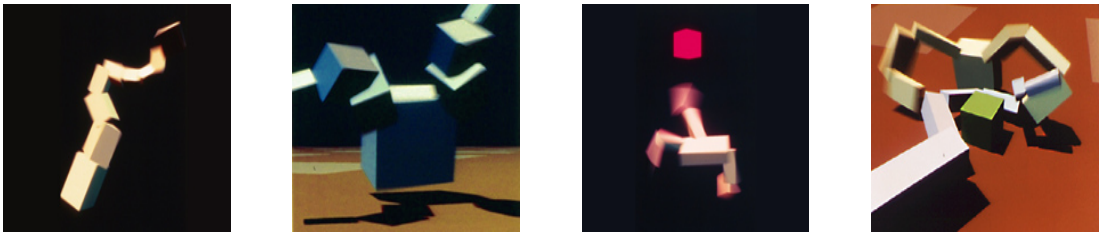
4. Les robots modulaires vont faire l'objet du chapitre suivant

agents virtuels de prédateurs et proies [Cliff and Miller, 1995]. Pour des revues plus détaillées, voir [Pinville, 2013]. Les robots à pattes peuvent être bipèdes [Wolff and Nordin, 2002, Ouannes et al., 2012], quadrupèdes [Yosinski et al., 2011] ou hexapodes [Cully et al., 2015]. Ici, le contrôleur cible les articulations. Le but est de pouvoir réaliser des mouvements de pattes réguliers et synchronisés, tels ceux des animaux, tout en assurant un certain équilibre et stabilisation. Les robots à multiples pattes sont plus adéquats pour les terrains irréguliers. Lors des missions de sauvetage ou d'exploration, ils peuvent facilement marcher sur les petits fossés et passer des zones sensibles en faisant des grands pas.

Les chercheurs, en grande partie, ont utilisé pour cela des contrôleurs à base des Réseaux de Neurones Artificiels (RNA) [Hagan et al., 1996], évolués par des AG ou de la PG. Cependant, d'autres types de contrôleurs ont été utilisés, notamment les GRN [Guo et al., 2009, Nicolau et al., 2010, Sanchez and Cussat-Blanc, 2014], et le NeuroEvolution of Augmenting Topologies (NEAT) [Stanley and Miikkulainen, 2002] (p. ex. [Cardamone et al., 2009, Kohl et al., 2006]). En outre, on a proposé d'utiliser la nouvelle approche Hypercube-based NEAT (HyperNEAT) [Stanley et al., 2009] basé sur les principes de NEAT. [Clune et al., 2009, Yosinski et al., 2011] C'est une méthode puissante de neuroévolution [5]. Elle utilise une forme de codage génératif pour optimiser à la fois les poids des RNA et leurs topologies. Les chercheurs affirment que HyperNEAT surpasse la neuroévolution de topologie fixe, car elle a permis la conception de contrôleurs complexes, mais aussi réguliers et coordonnées. Cependant, elle n'a pas réussi à faire évoluer des contrôleurs modulaires. [Bongard et al., 2015]

Récemment, les chercheurs ont suggéré d'optimiser des contrôleurs aussi diversifiés qu'efficaces en lançant une seule fois le processus d'évolution. Par conséquent, on obtient une collection de multiples contrôleurs pouvant réaliser différentes tâches selon l'espace de comportements possibles. On a proposé donc une nouvelle catégorie d'algorithmes évolutionnaire nommée Quality Diversity (QD) [Pugh et al., 2016], nous citons par exemple la *MAP-Elites* [Tarapore et al., 2016] et la *Novelty Search* combinée avec de la compétition locale [Lehman and Stanley, 2011b]. Ces derniers ont été utilisés pour concevoir des contrôleurs robustes et flexibles [Cully et al., 2015] (nous discuterons ce point dans la section numéro 2.4) et aussi pour simplifier des missions complexes en des séries de tâches simples de bas niveau, par exemple se déplacer, éviter des obstacles et ensuite saisir un objet, etc. [Mouret and Doncieux, 2012, Cully and Mouret, 2013]. Dans d'autres travaux, on a utilisé les AEMO pour optimiser plusieurs critères à la fois, notamment la maximisation des distances parcourues et la minimisation des consommations énergétiques. [Moore and McKinley, 2016]

5. Faire évoluer des RNA



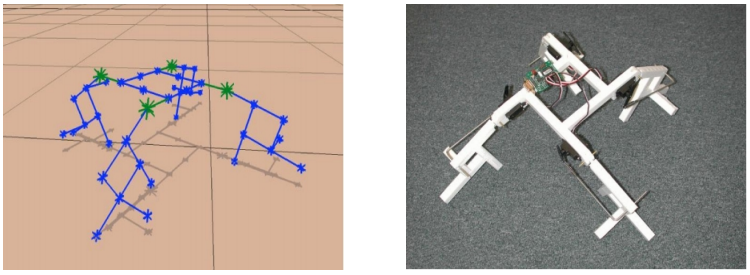
(a) nager (b) sauter (c) poursuite (d) compétition

FIGURE 2.4 – Exemples des créatures artificielles de Karl Sims

Évolution jointe des morphologies et des contrôleurs

Karl Sims [Sims, 1994b] a fait évoluer, de manière jointe, des morphologies et des contrôleurs de créatures artificielles simulées. Il a utilisé pour cela des AG. Les créatures se composaient de modules rigides de tailles différentes (voir figure 2.4). Elles pouvaient avoir trois types de capteurs, à savoir un capteur d'angles, de contact et un autre de lumière. Les génotypes des créatures étaient codés par des *Graphtal* [Sims, 1994b]. Ces derniers sont des graphes de développement orientés, qui se composent de nœuds et de liaisons. Les nœuds comprennent les caractéristiques physiques de chaque bloc de la créature, mais aussi un contrôleur local intégré basé sur un RNA. On a remarqué que le génotype à base de graphe fait produire naturellement des morphologies modulaires et symétriques. [Miconi and Channon, 2006] Selon la fonction *objectif* utilisée dans le processus d'évolution, les créatures pouvaient exécuter diverses tâches, telles que le développement de différentes manières de locomotion (nager, sauter, ramper) et la poursuite d'une source lumineuse. Dans un autre travail [Sims, 1994a], il a pu générer des créatures compétitives avec comme but commun ; s'emparer d'un cube. Les créatures développées avaient ainsi des membres articulés pour tenter de faire pousser le cube ou d'en garder la possession.

En 2001, Tim Taylor et Colm Massey [Taylor and Massey, 2001] ont repris exactement le travail de Karl Sims pour faire évoluer des créatures qui peuvent soit ramper ou nager. Cependant, les auteurs ont utilisé un autre moteur physique et ont fait tourner le processus d'évolution sur un ordinateur personnel. Ils ont démontré ainsi qu'il est possible, en une certaine mesure, de reproduire le travail de Sims sur des ordinateurs personnels. Thomas S. Ray [Ray, 2001], à l'opposé de Sims, s'est basé sur une évolution interactive dont laquelle il a permis aux utilisateurs de sélectionner manuellement les créatures qui vont se reproduire (sélection artificielle), et ce en se basant sur leurs émotions éprouvées envers les créatures. Il a appliqué un gradient de couleur sur les blocs des créatures. Comme résultat, il a obtenu des créatures intéressantes qui exposent des formes, des couleurs et des mouvements esthétiques. En outre, faire intervenir l'utilisateur lors de la sélection, lui permet d'une certaine manière comprendre les mécanismes de l'évolution.

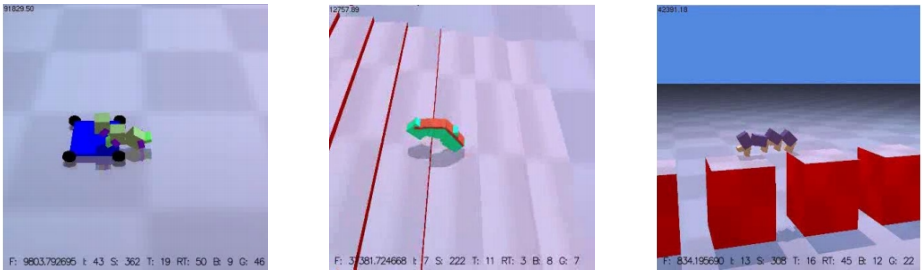


(a) Créature simulée (b) Le robot correspondant

FIGURE 2.5 – Les robots de Hornby et Pollack, générés par des *L-systems*

Hod Lipson et Jordan Pollack [Lipson and Pollack, 2000] ont reproduit les expérimentations de Sims, mais avec une étape additionnelle, qui consiste à fabriquer manuellement des copies physiques des meilleures créatures évoluées en simulation. Ils étaient les premiers à finaliser le processus d'évolution et construire de vrais robots. Les morphologies des robots se composent de blocs sous forme de bâtons liés par des joints, qui peuvent être soit fixes pour former des structures rigides, ou articulés pour permettre une liberté de mouvement. Les bâtons peuvent s'étendre et rétrécir à fin de permettre le déplacement. Les longueurs de ses derniers sont contrôlées par un signal sinusoïdal produit par un RNA. Par ailleurs, uniquement la mutation est introduite lors du processus d'évolution, ici un AG. En dépit de la tâche et de l'environnement relativement simples, les auteurs affirment avoir des solutions étonnamment différentes et élaborées. Quant aux chercheurs Jordan Pollack et Gregory Hornby [Hornby and Pollack, 2001], eux aussi ont pu reproduire des créatures évoluées en simulation dans le monde réel. Cependant, ils ont proposé un codage génératif à base des *L-systems* [Lindenmayer, 1968] pour la génération des morphologies et des contrôleurs. Les créatures, telles que présentées dans la figure 2.5, sont composées de plusieurs barres qui sont reliées par des joints fixes ou actionnés. En effet, Les *L-systems* permettent de produire des morphologies très modulaires. Le contrôleur utilise des RNA.

Yoon-Sik Shim et Chang-Hun Kim [Shim and Kim, 2006] se sont intéressés à concevoir des créatures artificielles volantes. Ils ont pour cela utilisé les AG pour évoluer les structures des ailes des créatures et leurs contrôleurs. Au lieu d'utiliser des blocs, ces auteurs ont utilisé des plans et des cylindres comme primitives pour former les ailes. Ils se sont inspirés des deltaplanes et des ailes des chauves-souris. Les contrôleurs sont des fonctions sinusoïdales. Les auteurs ont réussi à obtenir diverses créatures avec différentes formes d'ailes. Thomas Miconi et Alastair Channon [Miconi and Channon, 2006], quant à eux, ils ont fait évoluer des créatures capables de se mouvoir et de s'affronter pour prendre le contrôle d'une boîte cubique. À l'exception de Sims, ces chercheurs ont utilisé un contrôleur basé sur un RNA de type McCulloch-



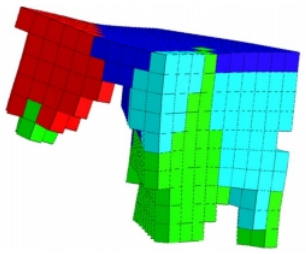
(a) (b) (c)

FIGURE 2.6 – Exemple des créatures de Nicolas Lassabe

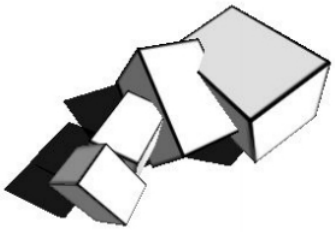
Pitts [Zhang and Zhang, 1999]. Bien que leurs créatures manquent de réalisme, Nicolas Lassabe [Lassabe, 2008] les a considérées comme celles qui se rapprochent le plus aux créatures de Sims, conçues il y a une dizaine d'années. Nicolas Chaumont et ses collègues [Chaumont et al., 2007] ont repris les créatures de Karl Sims, mais en plus de faire évoluer les créatures pour se mouvoir, ils proposent de les faire évoluer aussi pour être capable de catapulter un bloc. Pour s'adapter à cette tâche complexe, les créatures ont développé de stratégies intéressantes et étonnantes, par exemple, lancer ou pousser le bloc. Nicolas Lassabe et Yves Duthen [Lassabe et al., 2007] ont proposé de faire confronter les créatures à des environnements dynamiques et plus complexes (figure 2.6). Par exemple la présence d'escalier dont les créatures doivent monter ou interagir avec des objets mobiles (skateboard). Les auteurs ont aussi proposé un nouveau type de contrôleurs basé sur les systèmes de classifieurs. Ils ont conclu que réaliser des évolutions dans des conditions environnementales complexes peut en effet générer des créatures complexes et adaptées.

Plus récemment, Peter Krčah [Krčah, 2008] a également tenté de reproduire les créatures de Sims. Il s'est alors inspiré de ses travaux pour la représentation génétique de ses créatures (voire figure 2.7b), l'application d'opérateurs de reproduction ainsi que le contrôle. Cependant il suggère de faire évoluer les créatures par l'utilisation d'une extension de NEAT appelé Hierarchical NEAT (hNEAT). Comparé aux AG, il conclue que hNEAT donne des résultats meilleurs. Les valeurs des *fitness* de ses robots se sont significativement améliorées. Ceci est dû, au fait que hNEAT offrent une recherche plus robuste et échappent donc aux minima locaux. Nick Cheney et son équipe [Cheney et al., 2013] proposent de concevoir des robots mous. Le but de leur travail est de pouvoir complexifier les créatures. Ils agrandissent pour cela l'espace des solutions en changeant le matériau des robots, puisque l'exploitation de robots avec des segments rigides peut en effet limiter l'espace des solutions. L'idée principale consiste donc à décrire la morphologie du robot avec une grille tridimensionnelle de Voxels (voir figure 2.7a). Chaque Voxel peut avoir des propriétés physiques différentes. Il peut soit se contracter et s'étendre ou être passif avec un certain degré de rigidité.

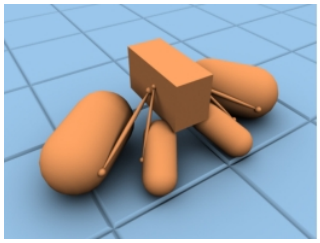
Pour la représentation génétique des robots, les auteurs se sont orientés vers un codage génératif basé sur le Compositional Pattern-Producing Network (CPPN) évolué par la méthode NEAT. Cependant, les auteurs affirment que ce qui a contribué à avoir de robots complexes c'est l'utilisation d'un codage génératif et non la présence des matériaux mous. Lessin et al. [Lessin et al., 2014] ont aussi pu développer des comportements complexes de créatures virtuels par l'utilisation des principes de l'*encapsulation*, du *syllabus*[6] et du *pandémonium*[7]. Les créatures avaient alors la possibilité d'apprendre de nouveaux comportements tout en préservant ceux appris précédemment. Les auteurs ont utilisé les mécanismes proposés par Sims pour générer les morphologies et les comportements des créatures. Cependant, contrairement aux créatures conventionnelles, ces dernières (voir la figure 2.7c) possèdent des muscles simulés au niveaux des jointures (implémenté comme des ressorts).



(a) Le robot mou présenté par Cheney et son équipe.



(b) Le robot de Peter Krčah.



(c) Le robot de Lessin et son équipe.

FIGURE 2.7 – Exemple de robots évolués pour leurs capacités de déplacement.

2.3.3 Importance de la morphologie

Le monde du vivant comprend une diversité énorme d'espèces de plantes, d'animaux, de champignons et de micro-organismes. Ces espèces possèdent des morphologies et des formes structurelles exceptionnellement différentes. En effet, ces dernières exercent un impact majeur sur la survie des espèces, car elles permettent des comportements et des actions, que ceux-ci à leurs tours, jouent le rôle crucial déterminant leurs sorts. Bien évidemment, ceci tout en prenant compte des conditions imposées par l'environnement. De ce point de vue, il est donc important d'avoir des morphologies de robots qui évoluent, puisque ça va permettre progressivement générer des robots différenciés et diversifiés de plus en plus adaptables à leurs environnements. Dans leur livre *How the Body Shapes the Way We Think* [Pfeifer and Bongard, 2006], Rolf Pfeifer et Josh Bongard ont déclaré que les cerveaux des robots dépendent

6. Combiner systématiquement des comportements de bas niveaux pour en concevoir un comportement de haut niveau

7. Un mécanisme pour résoudre les conflits entre les comportements encapsulés d'une créature

essentiellement des corps et doivent être évolués ensemble. Eiben [Eiben et al., 2013] affirma ces propos et ajouta que ce n'est que par cette évolution conjointe que nous pouvons atteindre la véritable intelligence artificielle. Il est donc important, lors de la conception du robot, de considérer à la fois l'aspect de la morphologie, du système de contrôle et des capteurs intégrés, car c'est l'interaction entre tous ces aspects qui va déterminer le comportement du robot. Cela représente un défi pour la robotique évolutionnaire. [Eiben et al., 2013, Doncieux et al., 2015]

En 2001, Chandana Paul et Josh C. Bongard [Paul and Bongard, 2001] ont proposé de comparer la locomotion d'un robot bipède de morphologie fixe avec une autre où la morphologie évolue. On a fait évoluer dans ce cas les répartitions des masses en changeant les tailles des blocs composant le squelette du robot. Les robots ont été évolués pour réaliser des locomotions avec des contrôleurs neuronaux dont on maximisait la distance parcourue en un temps fini. Après plusieurs expériences, les auteurs ont conclu qu'évoluer des paramètres de la morphologie à côté du contrôleur donne des résultats nettement meilleurs que lorsque la morphologie ne change pas.

Dans une série d'expérimentations récentes [Bongard et al., 2015], Josh C. Bongard et .al ont démontré qu'évoluer conjointement la morphologie avec le contrôleur permet de générer des robots évolvables. En d'autres termes, des robots qui s'adaptent rapidement aux environnements nouveaux. Ils ont étudié pour cela la conception de contrôleurs modulaires [8], puisqu'il a été démontré expérimentalement [Clune et al., 2013] que la modularité peut augmenter l'évolvabilité en faisant subir les individus une combinaison de pression de sélection [9]. Les auteurs présument alors et démontrent qu'un contrôleur modulaire permet à chacune des parties du corps d'un robot de réagir indépendamment aux stimuli. En effet, c'est ce qui rend le robot évolvable, c'est à dire, lorsque le robot est face à un nouvel environnement (situation vue comme étant une nouvelle combinaison de percepts familiers), il va s'adapter rapidement en trouvant un nouveau contrôleur sans refaire l'étape d'apprentissage (évolution). Ils affirment que la modularité au niveau des contrôleurs est atteinte si on fait d'une part évoluer ensemble le contrôleur et la morphologie et d'une autre part, sélectionner les individus pour leurs performances (*fitness*), leurs degrés de conservatisme [10] et la robustesse de leurs contrôleurs. Cependant, sélectionner directement les robots ayant un contrôleur de structure modulaire donne l'effet inverse (diminution de la performance). Bongard et son équipe démontrèrent qu'effectivement il y a une relation entre morphologie, modularité et évolvabilité, c'est à dire que la coévolution aide à trouver des morphologies qui peuvent avoir des contrôleurs modulaires, d'où des robots évolvables.

8. Dense connectivité à l'intérieur des modules et peu de connexions entre les modules
9. Une combinaison de sélection ; directionnelle sur une partie du phénotype et stabilisante sur l'autre partie.
10. Concept utilisé pour la pression de sélection

2.4 La résilience

De nos jours, il est important de construire des robots autonomes qui, sans intervention humaine, sont capables d'effectuer des tâches complexes dans des environnements dynamiques et peu connus. Cela représente l'ambition de beaucoup de chercheurs roboticiens depuis des décennies. [Thrun et al., 2005] Leur souhait est de mettre au point des robots qui sont aptes, par exemple, à conquérir l'espace et construire des colonies extraterrestres [Yim et al., 2007], réaliser des opérations de sauvetage et d'exploration en se mettant dans des situations aussi difficiles que dangereuses où les humains ne peuvent pas intervenir. [Mohiuddin et al., 2010] A cet effet, le robot doit être armé de plusieurs capacités comme l'intelligence, la résilience et l'adaptation. Cependant, ces notions sont toutes liées et se ressemblent. Dans nos travaux de thèse, nous allons nous intéresser à la résilience qui contribuera fortement à l'autonomie des robots.

Le terme «*résilience*» vient du verbe latin «*resilio*» qui signifie sauter en arrière ou rebondir. Il est employé dans plusieurs domaines tels que l'industrie, l'informatique, la psychologie et l'écologie, portant ainsi chacun sa propre définition. Au sens large, il est défini en termes de capacité de poursuivre une opération ou retrouver un état stable après un événement majeur suite à une perturbation ou un choc. [Hollnagel et al., 2007] En robotique, on fait référence à la capacité des robots à faire face aux situations imprévues et se rétablir des dommages subis afin de poursuivre leurs missions pour lesquels ils sont programmés. Toutefois, la performance d'exécution des tâches diminue, ce qui est normal par rapport à ce qui se passe dans les milieux évolutifs comme la nature et le comportement des animaux.

Généralement, un robot est programmé pour effectuer une tâche précise. Durant l'exécution de celle-ci, le robot inévitablement va rencontrer des obstacles dans l'environnement et au risque de subir des dommages dans sa structure physique. Il perdra donc toutes ses capacités apprises et son système de contrôle devient vain et inopérant. Dans ce cas, il est indispensable d'adopter rapidement la meilleure action pour remplir sa tâche. Cependant, l'espace de recherche contenant les actions potentielles est tellement grand qu'il est difficile de deviner efficacement et rapidement la future action, surtout si le robot est complexe et comprend plusieurs capteurs et actionneurs. Pour cela, à l'effet de rendre la résilience possible, un nombre d'approches est proposé dans les recherches effectuées dans ce domaine. Dans ce qui suit, nous présenterons les principaux travaux effectués actuellement dans ce sens.

À notre connaissance, le premier qui a introduit le terme «résilience» dans le domaine informatique est Hod Lipson et ses collègues [Bongard et al., 2006] en 2006, et ce, après avoir mené un travail qui a abouti au développement d'un robot-araignée résilient. Toutefois, la notion de résilience était étudiée bien avant les recherches

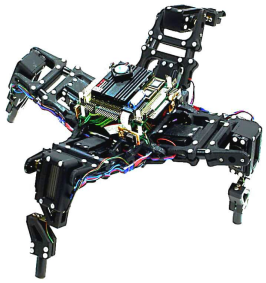
de Hod Lipson dans divers travaux et elle était définie différemment, entre autres, la tolérance aux pannes (*fault tolerance* en anglais) [Visinsky et al., 1994, Blanke et al., 2006] et la résistance aux dommages (*damage recovery* en anglais) [Mahdavi and Bentley, 2003]. D'après les travaux que nous trouvons dans la littérature, les pannes et les perturbations peuvent affecter soit le software ou le hardware. Nous nous focaliserons sur les perturbations qui affectent le hardware.

Les recherches classiques souvent s'orientent vers la conception de contrôleurs robustes [Shin and Lee, 1999, Lin and Chen, 2007] ou vers la prévention [Visinsky et al., 1994, Erden and Leblebicioğlu, 2008, Zilberstein et al., 2002]. Cependant, prédire toutes les défaillances possibles et en concevoir préalablement pour chacune une solution ou un plan de secours est impossible. En outre, on a tenté d'utiliser des robots qui sont cinématiquement redondants. C'est-à-dire que le robot a plus de degrés de liberté ou autant de mouvements que nécessaire, ce qui permet de créer des algorithmes de tolérances aux pannes qui utilisent les configurations alternatives pour réaliser la tâche avec des joints défectueux. [Visinsky, 1992] Par exemple, les robots modulaires auto-reconfigurables, avec leurs avantages que nous allons discuter dans la section 3.2, ont la capacité de réarranger leurs modules de manière autonome et donc modifier la configuration d'un système défaillant pour en avoir une nouvelle.

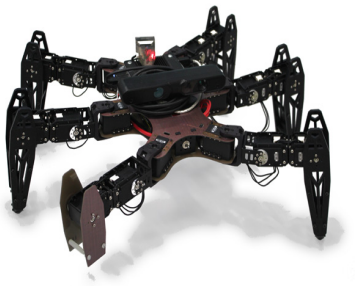
Récemment on tentait de laisser le robot utiliser les AE, pour apprendre un nouveau comportement compensateur, et ce après la détection de dommage [Bongard et al., 2006, Mahdavi and Bentley, 2003]. En d'autres termes, on a mis en action un autre contrôleur qui relève à chaque dégradation de performance. Le processus d'apprentissage est fait donc en ligne, soit directement sur le robot physique[11] [Mahdavi and Bentley, 2003, Mahdavi and Bentley, 2006, Berenson et al., 2005] ou soit en simulation [Koos et al., 2013, Bongard et al., 2006, Cully et al., 2015]. Le premier cas d'apprentissage demande de réaliser des tests essais et erreurs sur la machine physique. Afin d'obtenir de bons résultats, il est nécessaire de réaliser des centaines de milliers de tests, ce qui est risqué et coûteux en termes de temps et d'argent. Les auteurs de [Mahdavi and Bentley, 2006] ont adopté cette approche et l'ont testé sur un robot serpent qui, à la huitième génération a subi de véritables cassures.

Transférer le processus d'apprentissage en simulation s'avère alors important pour diminuer le nombre d'évaluations sur le robot réel. Du coup, le temps d'exécution du processus d'apprentissage sera considérablement réduit. Sur cette voie, Hod Lipson et son équipe [Bongard et al., 2006] ont construit un robot résilient à quatre pattes (voire la figure 2.8a). Ils ont proposé un modèle qui, premièrement, permet d'attribuer le robot une capacité de perception de soi-même en continue "*self-model*". Cette capacité rend le robot capable de détecter l'endroit de l'endommagement en faisant

11. On parle de *Embodied Learning*



(a) Le robot araignée résilient proposé par [Bongard et al., 2006]



(b) Le robot hexapode utilisé pour tester les algorithmes T-Resilience et IT&E

FIGURE 2.8 – Exemple de robots résilients

quelques actions de base sur la machine physique, et ainsi créer dans sa conscience un modèle définissant sa nouvelle morphologie. Néanmoins, la morphologie à prédire est l'une d'un nombre de candidats anticipés. Deuxièmement, une fois le *self-model* est deviné, un AE est appliqué sur un simulateur à fin de trouver le contrôleur le plus adapté à la nouvelle morphologie. En effet, le robot-araignée, malgré le dommage causé à l'une de ses pattes, il a réussi à changer de comportement et continuer à se déplacer. Cependant, cela est fait en plusieurs heures. Cela est dû au fait que la recherche d'un autre contrôleur commence sans aucune connaissance de l'espace de solutions possibles *"from scratch"*.

Antoine Cully et son équipe ont poursuivi le même principe du self-model et ont proposé l'algorithme T-Resilience [Koos et al., 2013] . Ce dernier repose sur le fait de trouver implicitement un contrôleur également efficace pour le robot intact (*self-model*) et pour le robot endommagé. On cherche alors un contrôleur qui ne se sert pas des parties endommagées pour se mouvoir. En outre, les auteurs considèrent qu'au moment de la défaillance, le self-model est une erreur de Reality Gap [12], puisque, comme ils le soulignent, de toute façon, la différence entre les deux morphologies n'est pas grande. L'algorithme, de manière périodique, transfère des contrôleurs au robot physique pour calculer la différence de comportement entre la simulation et la réalité (valeur de transférabilité). Avec un algorithme de régression, on prédit les valeurs de transférabilité pour les contrôleurs non testés en réalité. T-Resilince utilise ensuite un AEMO pour optimiser à la fois l'efficacité du comportement du robot en simulation et la transférabilité estimée. La particularité de cette approche est qu'elle était utilisée initialement pour traverser le Reality Gap. Les auteurs ont testé leurs approches sur un robot hexapode (figure 2.8b) qui a réussi à trouver un comportement compensateur en une durée de 20 minutes.

12. Une grande différence de comportement entre la simulation et la réalité. Pour plus de détail, voir la section 4.5

Plus récemment, les mêmes auteurs du travail cité ci-dessus ont proposé l'algorithme Intelligent Trial and Error (IT&E) [Cully et al., 2015], qui permet de fournir au robot, avant d'être déployé en mission, une connaissance représentée sous forme de base de données. Cette connaissance, qu'on a appelée la *«Behavior-performance map»*, est un ensemble de milliers de contrôleurs qui permettent au robot de faire une tâche donnée de différentes manières. Ensuite, lors d'exécution de celle-ci, et au moment de dommage, le robot va itérativement sélectionner depuis la base de données le contrôleur le plus adéquat qui est performant à la fois en simulation et en réalité. Ils utilisent pour cela une recherche en ligne avec un algorithme d'optimisation bayésien. Le robot ne réalise pas donc de diagnostic. Toutes fois, il y a nécessité de faire un nombre d'essais sur le robot réel. Les auteurs ont démontré que IT&E permet à un robot à six pattes de se remédier à une variété de dommages en moins de 2 minutes.

Les essais effectués sur le robot réel lors des moments de défaillances, que ce soit pour faire un diagnostic, une optimisation ou une sélection d'un nouveau contrôleur, font perdre du temps et risquent d'endommager le robot davantage en testant des comportements trop extrêmes pour la conception mécanique du robot. [Mouret and Chatzilygeroudis, 2017] Par conséquent, cela va empêcher le robot de continuer d'exécuter sa mission qui peut être critique dans certains cas.

L'un des objectifs de cette thèse est la conception de robots résilients. Les approches présentées ci-dessus ont été utilisées sur des robots de morphologies fixes et prédéfinies. Dans notre travail, nous allons proposer de soumettre les morphologies à évolution. Vu les avantages apportés par l'évolution conjointe des contrôleurs et des morphologies, nous aspirons à générer des configurations de robots qui permettent une résilience sans changer de contrôleur à chaque fois un dommage survient. En d'autres termes, nous souhaitons optimiser et faire émerger des morphologies, dont les mouvements autorisés ne pouvant être affectés par les dommages.

2.5 Défis de la robotique évolutionnaire

En robotique évolutionnaire, on trouve souvent des difficultés notables à optimiser des robots autonomes et adaptés, notamment lors de la coévolution de la morphologie et du contrôleur. [Cheney et al., 2016, Taylor, 2017] En effet, durant ces deux dernières décennies, on a toujours des capacités limitées à dépasser les travaux initiaux de Karl Sims, que ce soit en termes de complexité ou de diversité des créatures évoluées, et ce malgré une augmentation significative de la puissance de calcul. [Cheney et al., 2016] Il faut néanmoins noter qu'on a pu optimiser avec succès des contrôleurs efficaces pour des robots à morphologies fixes. (p. ex. [Cully et al., 2015]) Dans ce cas, les robots pouvaient remplir leurs tâches efficacement, mais comme déjà discuté, ils sont

peu adaptables. Il faut savoir cependant qu'il y a d'autres défis connus actuellement en robotique évolutionnaire, que nous n'allons pas évoquer dans cette partie (pour plus détail voir [Doncieux et al., 2015]).

De nombreux chercheurs ont proposé alors des hypothèses et des pistes de solutions pour tenter de résoudre cette problématique de scalabilité (évolutivité) de coévolution des robots virtuels. On a suggéré par exemple d'agrandir les tailles des populations ainsi que les durées des évaluations en simulation, car tout simplement cela n'a pas été assez grand à ce jour. [Taylor, 2017] En outre, on a tenté de complexifier l'environnement, les tâches chargées aux robots ainsi que les interactions entre ces derniers [Auerbach and Bongard, 2014, Lassabe et al., 2007, Miconi and Channon, 2006]. Les créatures émergées étaient effectivement plus complexes et variées. Ceux qui introduisaient des interactions entre les individus ont fait émerger des morphologies et des comportements adaptatifs créatifs. Par exemple l'interaction entre deux adversaires a produit certains des résultats les plus impressionnants. [Miconi and Channon, 2006, Sims, 1994a] Nous soulignions toutefois que ces résultats ne dépassent pas considérablement ceux de Sims.

D'autres hypothèses supposent que l'espace des solutions possibles est limité ou alors les systèmes de codage sont réguliers, ne permettant pas des variations complexes. [Cheney et al., 2013]. Bien qu'il n'y ait pas de notion concrète de ce qui cause cette problématique, nombreux sont qui ont théorisé qu'elle pourrait être due aux lacunes présentes dans les AE, notamment la convergence prématurée.

2.5.1 Convergence prématurée

Il est reporté dans la littérature [Cheney et al., 2017, Lehman and Stanley, 2011b, Mouret and Clune, 2015] que la convergence prématurée est le facteur majeur derrière la non scalabilité de la coévolution des morphologies et des contrôleurs des systèmes robotiques. En fait, elle représente un problème classique auquel sont confrontés les algorithmes évolutionnaires. Il est connu que le processus d'évolution converge souvent rapidement sur une seule famille d'individus, provoquant ainsi l'empêchement d'explorer de nouvelles régions dans l'espace de recherche. En réalité, la convergence prématurée est étroitement liée à la perte de la diversité génétique dans la population. Par ailleurs, David Goldberg, dans son livre *The Design of Innovation* [Goldberg, 2013], présume que cela est causé par le fait que le temps d'acquisition (référence au temps nécessaire pour que le meilleur individu domine la population) est supérieur au temps d'innovation (référence au temps nécessaire pour créer un meilleur individu). Ainsi, la sélection favorisera la convergence de la population vers les minima locaux avant l'apparition d'un nouveau meilleur individu. De ce fait, les qualités des individus générés sont sans doute limitées et non souhaitées.

Afin de résoudre ce problème, ou du moins le diminuer, les chercheurs tendent à utiliser, essentiellement, des algorithmes qui permettent de maintenir un certain degré de diversité génétique, notamment la *Novelty Search* [Lehman and Stanley, 2011a]. Cette méthode a été développée par les deux chercheurs Joel Lehman et Kenneth O. Stanley ayant comme but la recherche de la nouveauté comportementale sans pour autant utiliser explicitement une fonction objectif. Au fait, la *Novelty Search* pourrait encourager la diversité des morphologies des créatures d'un côté, et de l'autre récompenser par compétition locale celles qui sont performantes par rapport aux créatures ayant la même morphologie (ou presque). Des créatures intéressantes et diverses vont être découvertes avec des comportements originaux, potentiellement efficaces. [Lehman and Stanley, 2011b] Par ailleurs, cette méthode évitera toute évaluation explicite de la qualité des individus. En effet, l'utilisation d'une fonction *objectif* peut être trompeuse [Doncieux et al., 2015], car elle définit de manière approximative la tâche, fournissant très peu d'informations sur les performances réelles d'une solution donnée. Chercher à améliorer les valeurs des *fitness* conduit vraisemblablement la population aux minima locaux, alors que la *Novelty Search* conduit à la découverte des solutions de plus en plus complexes, explorant de nouvelles régions dans l'espace des solutions.

Nick Cheney et son équipe [Cheney et al., 2016] affirment que la notion de la convergence prématurée affecte en premier lieu la morphologie, autrement dit, cette dernière converge avant le contrôleur. Les auteurs déclarent que ceci est causé lorsqu'on introduit des mutations au niveau des morphologies, dont celles-ci, semblent affecter fortement le contrôleur et la façon dans laquelle il interprète les effecteurs. Cela signifie que le contrôleur n'est plus adapté pour la nouvelle morphologie. Du coup, si la morphologie et le contrôleur d'un robot s'accordent parfaitement, la mutation et la recombinaison vont potentiellement faire perdre cette cohérence. En effet, ces propos sont cohérents avec ceux que nous avons observés lors de nos expériences. Une légère variation, aussi petite soit elle, peut faire changer complètement le comportement du robot. En d'autres termes, une petite variation du génotype engendre un grand changement du phénotype. Par ailleurs, la recombinaison perd ainsi tout son intérêt, puisqu'elle est censée se rapprocher des solutions optimales dans l'espace local.

Afin de permettre aux contrôleurs de s'ajuster aux nouvelles morphologies, nous pouvons penser à introduire des phases d'apprentissage périodiques uniquement pour le contrôleur. Cela peut être fait lors des générations avancées, là où les valeurs de *fitness* ne progressent plus. Dans ce cas, les morphologies des robots sont plus ou moins adaptées à l'environnement et à la tâche donnée, et donc, perfectionner le contrôleur pour donner de meilleurs résultats.

Nick Cheney et son équipe reviennent alors à la théorie de la cognition [13] incarnée [14] pour proposer une théorie qui pourrait expliquer un tel obstacle. Pour cela, ils prennent en considération la relation d'interdépendance entre le cerveau et le corps du robot durant le processus d'optimisation. En effet, selon cette théorie, les interactions dynamiques entre le cerveau, le corps du robot et l'environnement, sont un important moteur pour produire la cognition ou le comportement. Dans un autre travail [Cheney et al., 2017], les mêmes auteurs font réduire temporairement la pression de sélection sur les morphologies de robots nouvellement mutés, leur permettant ainsi de réadapter leurs contrôleurs et échapper aux optima locaux. Les auteurs affirment que cette technique a permis de faire tarder la convergence prématurée, et générer des robots plus efficaces.

2.6 Conclusion

Ce chapitre a donné une vue d'ensemble de la robotique évolutionnaire. Nous avons pu voir que c'est un domaine fortement lié à la vie artificielle, puisqu'il s'inspire des techniques qu'elle propose pour la conception de robots ayant des propriétés du vivant. Depuis l'état de l'art, nous avons vu que les créatures conçues par coévolution des morphologies et des contrôleurs peuvent être plus variées et mieux adaptables aux nouvelles conditions imposées par l'environnement. Nous remarquons que les stratégies engendrées vont dépendre essentiellement des morphologies, et ensuite les exploiter pour pouvoir exécuter les tâches. Cependant, cela représente un défi.

Nous allons donc suivre cette voie et faire coévoluer des robots virtuels, comme ceux présentés dans la littérature, modulaires composés de segments ou blocs rigides connectés par des articulations. Par railleurs, nous allons prendre en considération la coopération et l'interaction entre les modules. Bien que les créatures proposées par les chercheurs meuvent de manière créative, nous constatons qu'elles manquent de réalisme. Elles ne peuvent pas être directement fabriquées physiquement. Nous remarquons facilement que la majorité des robots virtuels présentent des interpénétrations entre les modules. Nous allons proposer donc de rendre plus réalistes les simulations. En outre, nous aspirons que nos robots soient résilients. Ainsi, ils peuvent reprendre leurs entreprises suite aux perturbations qui surviennent à n'importe quel moment.

13. La cognition au sens large représente le processus naturel ou artificiel qui permet l'acquisition de connaissance (apprentissage, mémoire, etc.).

14. La cognition incarnée suggère qu'une partie fondamentale du processus de contrôle cognitif d'un individu est située [Wilson, 2002]. Cela signifie que le corps d'un agent peut influencer la cognition.

Chapitre 3

Robotique modulaire

3.1 Introduction

Le domaine de la robotique modulaire s'adresse à la conception, la fabrication et la planification de contrôle de robots autonomes à morphologies variables. [Yim et al., 2007] Il englobe à la fois, robotique, intelligence artificielle et vie artificielle. [Pfeifer and Bongard, 2006] En fait, les premiers travaux ont été entamés par le chercheur roboticien Toshio Fukuda à l'université Nagoya au Japon vers la fin des années 1980. Il a construit le robot modulaire CEBOT (Cellular Robot) [Fukuda and Nakagawa, 1988]. Ce robot avait une conception inspirée de la biologie cellulaire. Chacun de ses modules représente une cellule qui possède une certaine intelligence avec des capacités mécaniques. Par la suite, les chercheurs ont commencé à s'intéresser en premier lieu à la conception mécanique des modules, puis au développement des politiques de contrôle, qui peuvent être locales, maître/esclave ou évoluées par des AE, etc.

Dans ce chapitre, nous allons présenter, dans un premier temps, les robots modulaires ainsi que les avantages qu'ils apportent par rapport aux robots conventionnels. Nous allons effectuer par la suite une étude dans laquelle nous tentons de décrire quelques robots modulaires réalisés récemment. En fin nous comparerons ces robots afin de retenir ce qui caractérise chacun par rapport aux autres, et définir ce que nous allons inclure dans la conception de nos robots modulaires.

3.2 Les robots modulaires

Les robots modulaires [Yim et al., 2007] sont des robots qui se composent d'un ensemble de modules homogènes ou hétérogènes. Chaque module est généralement équipé par ses propres capacités indépendantes de calcul, de détection et de com-

munication. En outre, il peut être doté par des actionneurs et des connecteurs lui permettant une certaine liberté de mouvement. Dans ce cas, les modules sont considérés comme étant des unités ou des agents indépendants. En fait, ces derniers peuvent être assemblés en différentes configurations, pour construire à chaque fois un nouveau robot avec de nouvelles fonctionnalités. Par exemple, une configuration semblable à un serpent peut être plus appropriée pour traverser des trous étroits et s'infiltrer dans des décombres, alors qu'une configuration semblable à une araignée est meilleure pour marcher sur des terrains accidentés. Par railleurs, l'interaction entre les modules permet d'engendrer des comportements coordonnés intéressants. Cela rend le robot polyvalent et versatile.

Dans les environnements incertains et changeants, les robots modulaires sont plus avantageux par rapport aux robots conventionnels de morphologies fixes. Ces derniers sont généralement conçus pour un usage prédéterminé. Ils exécutent alors leurs tâches spécifiques avec précision, mais dès qu'il y a un changement de mission, leurs performances deviennent vaines et leur adaptation est difficile. Les robots modulaires quant à eux sont reconfigurables et polyvalents. Leurs morphologies peuvent être modifiées intentionnellement afin de mieux s'adapter aux nouvelles conditions exigées par l'environnement. Ils peuvent apporter trois avantages ; [Yim et al., 2007] la polyvalence, la robustesse et le low-cost :

- **Polyvalence :** c'est la capacité de modifier la morphologie du robot en fonction de la tâche à accomplir et de son environnement.
- **Robustesse :** en raison de la redondance et de la modularité, le robot est robuste aux pannes mécaniques. Si les modules sont identiques, réparer le robot devient facile, il suffit de remplacer le module endommagé par un autre.[Haji Agha Memar et al., 2008]
- **Low-cost :** un grand nombre de modules est construit en série avec un faible coût. Certains fabricants utilisent des techniques de prototypage rapide tels que l'impression 3D. [Auerbach et al., 2014, Krupke et al., 2015]

Cependant, le grand nombre de modules qu'un robot peut en avoir peut entraîner un compromis de performance et de complexité accrue que ce soit au niveau de la réalisation du robot ou de la conception du contrôleur. [Yim et al., 2007]

3.3 Les types des robots modulaires

Les chercheurs ont étudié et proposé différents types de systèmes robotiques modulaires, en vue de réaliser plusieurs tâches variées. Pour exemple, les opérations de sauvetage [Yim et al., 2000] et les applications spatiales [Hancher and Hornby, 2006, Salemi et al., 2006], bien entendu dans des environnements incertains et non

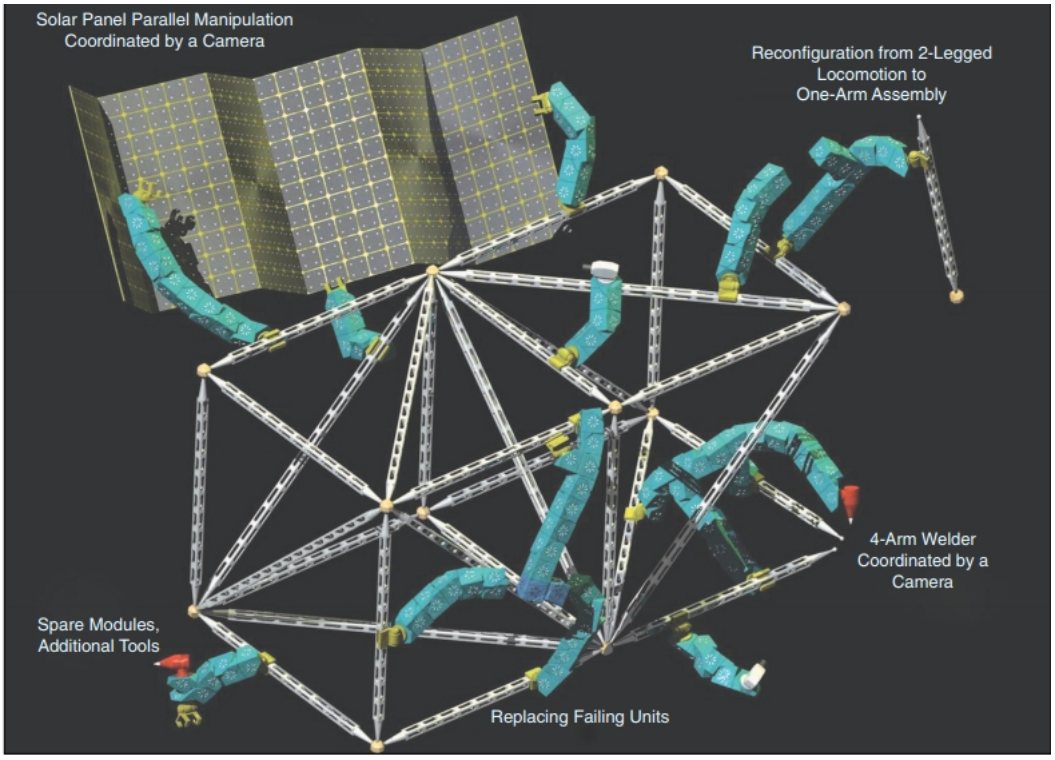


FIGURE 3.1 – Représentation artistique d'une application spatiale de la robotique modulaire, montrant une colonie de robots en chaînes composés, configurés en diverses morphologies pour une variété de tâches (figure prise de [Yim et al., 2007])

connus. Un exemple d'une application spatiale est illustré dans la figure 3.1. Les robots modulaires ont été utilisés également pour le divertissement [1] et l'éducation [Auerbach et al., 2014].

Selon l'objectif pour lequel le robot est conçu, il peut exhiber des propriétés différentes, par rapport aux types des modules utilisés, leurs arrangements ou la façon dans laquelle ils se reconfigurent. Au fait, la reconfiguration des modules se fait soit de manière manuelle ou autonome, sans intervention humaine. Il est évident que la caractéristique d'auto-reconfiguration est l'une des motivations principales de ce domaine. Elle permet une totale autonomie aux robots pour s'adapter à de nouveaux environnements, effectuer de nouvelles tâches ou se remettre des dommages. Malgré les avantages potentiels promis par l'auto-reconfiguration, ils restent encore à être concrétisés et visualisés sur le terrain.

Par ailleurs, Yim et al. [Yim et al., 2007] classifient les robots modulaires auto-reconfigurables en trois catégories en se basant sur l'arrangement de leurs topologies. Il y a les robots modulaires treillis, les robots modulaires chaînés et les robots modulaires mobiles. En outre, plusieurs systèmes robotiques présentent des propriétés hybrides.

1. https://www.barobo.com/

En revanche, ceux qui sont manuellement reconfigurables font partie des robots chaînés.

- **Les robots chaînes :** dans cette catégorie de robots, les modules sont arrangés en chaînes uniques ou multi-branches. Avec leurs articulations, les robots, qui peuvent être par exemple des quadrupèdes, des serpents ou des bras manipulateurs, sont capables d'atteindre potentiellement n'importe quel point ou orientation dans l'espace. Ils sont donc plus polyvalents, mais difficile à représenter et à analyser, et plus encore difficile à contrôler.[Yim et al., 2007]
- **Les robots treillis :** ici les modules sont capables de se déplacer les uns par rapport aux autres sur une grille virtuelle (treillis). En effet, le seul moyen de locomotion des robots treillis est l'auto-reconfiguration, devenant ainsi la principale problématique.
- **Les robots mobiles :** les modules, dans ce cas, sont capables d'opérer individuellement sur le terrain, généralement par des roues. Ils peuvent aussi se connecter pour former soit des chaînes en un réseau virtuel, où ils sont considérés comme des robots plus petits qui exécutent des mouvements coordonnés.

En fait, quel que soit le type du robot, on trouve généralement deux axes de recherches ; l'auto-reconfiguration et la locomotion. Durant cette thèse nous contentons d'étudier la locomotion des robots reconfigurables manuellement avec des modules hétérogènes. Les modules peuvent être reconfigurés pour former des topologies arbitraires en chaînes.

3.4 Les modèles proposés

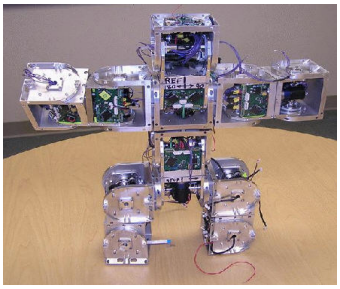
Les chercheurs ont proposé différents modèles de robots modulaires. Ils présentent tous des caractéristiques uniques. Nous proposons dans cette section de retenir les modèles les plus récents et ceux que nous trouvons intéressants. Nous allons les présenter séparément selon leurs manières de reconfiguration.

3.4.1 Les robots modulaires auto-reconfigurables

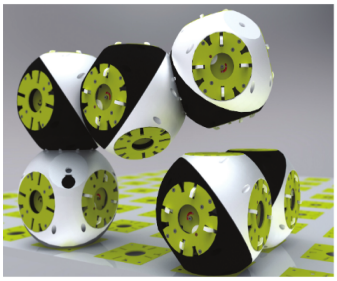
M-tran III

Le robot M-tran III [Kurokawa et al., 2008] est la version modifiée du M-tran (Modular Transformers) réalisé en 1998 à l'institut de recherche AIST au Japan. Le système M-tran est homogène et a une architecture hybride (treillis et chaîne). Ses modules ont deux degrés de liberté (DOF) et sont constitués de deux blocs dont chacun possède trois connecteurs. Dans les versions I et II de ce système, la connexion

(a) Smores



(b) Superbot



(c) RoomBot

FIGURE 3.2 – Exemple de robots modulaires auto-reconfigurables

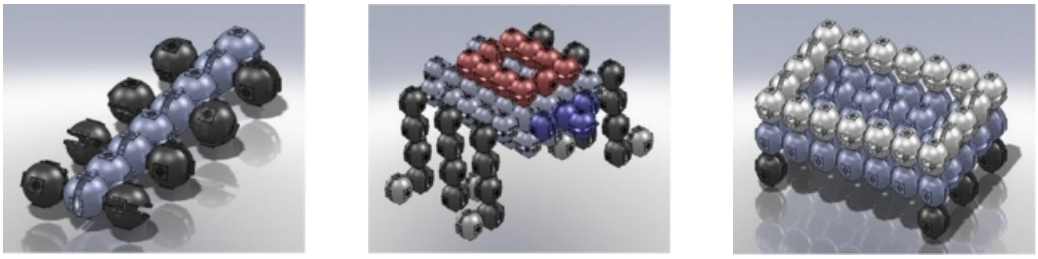
se faisait par des aimants. Dans la version III, la connexion est purement mécanique, et donc, la vitesse et la fiabilité de la connexion sont largement améliorées. Avec un système de contrôle embarqué et distribué, M-tran III peut réaliser différents modes de locomotions et reconfigurations. Les expériences d'auto-reconfigurations ont été effectuées sur des robots ayant jusqu'à 24 modules.

SuperBot

Le robot modulaire homogène SuperBot (Super Robot) [Salemi et al., 2006] est conçu par le Dr Wei-Min Shen et son équipe à l'université de Californie du sud, dont le but de réaliser des explorations extraterrestres. Tout comme M-tran, les modules de SuperBot se composent de deux blocs liés par une articulation, mais avec un degré de liberté supplémentaire. Un module possède donc 3 DOF avec la possibilité de rotation autour du même axe. Au début, les prototypes ne pouvaient pas s'auto-reconfigurer car les connecteurs n'étaient pas opérationnels, mais après 2008 ils ont subi une amélioration leur permettant l'auto-reconfiguration. Les chercheurs ont utilisé un système de contrôle distribué maître/esclave inspiré du fonctionnement des hormones. On utilise pour cela une communication entre les différents modules à fin de choisir la meilleure action à entreprendre. Plusieurs locomotions ont été alors développées pour des configurations prédéterminées différentes. Par exemple le robot peut ramper, marcher et rouler. Le nombre maximal de modules utilisés dans les expériences en simulation est 20 modules.

RoomBot

RoomBot [Sproewitz et al., 2009] a été conçu à BIOROB Laboratory en Suisse pour une utilisation spéciale de robot modulaire. Il s'agit d'assister ou former des meubles afin d'améliorer la qualité de la vie quotidienne. Comme il est montré dans la figure 3.2c, le robot est un système hybride et homogène qui peut s'auto-reconfigurer. Il peut se mouvoir comme un quadrupède ou une chenille. Chacun de ses modules



(a) (b) (c)

FIGURE 3.3 – Quelques exemples de configuration du robot Cross-Ball. (a) un robot ressemblant à une chenille (b) un robot semblable à un hexapode. (c) un robot semblable à un véhicule

est composé de quatre demi sphères avec 10 faces de connexion. Les modules ont 3 DOF de rotation. Le software de RoomBot permet la coévolution de la forme et du comportement, pouvant ainsi générer différentes morphologies de robots. La morphologie est déterminée par les L-System, tandis que le contrôleur est un Central Pattern Generator (CPG) [Ijspeert, 2008] implémenté comme un réseau d'oscillateurs couplés.

Cross-Ball

Yan Meng et son équipe [Meng et al., 2011] proposent Cross-Ball, un nouveau robot modulaire auto-reconfigurable de type treillis. Les chercheurs s'adressent au problème de contraintes mécaniques qui se trouvent au niveau des connecteurs. Ils proposent alors un mécanisme amélioré d'attachement qui permet aux modules des mouvements parallèles, rotationnels et diagonaux. Cela fournit de la flexibilité et de la robustesse pour permettre l-'auto-reconfiguration selon différentes topologies complexes, tel que présenté dans la figure 3.3. Les modules de Cross-Ball sont sphériques et homogènes avec 6 points d'attachements. Pour réarranger les modules, on a proposé un contrôleur distribué à 3 niveaux de contrôle. Il est basé sur l'approche morphogénétique et sur le développement embryonnaire des organismes multicellulaires.

Transmote

Le robot modulaire homogène Transmote [Qiao et al., 2012] est hybride et auto-reconfigurable. Il a été conçu pour être adapté aux opérations de recherche et de sauvetage dans les environnements complexes. Pour cela, on a apporté des nouvelles caractéristiques avancées au niveau des mécanismes de connexion entre modules. Il s'agit d'un système d'accrochage mâle/femelle qui dépend de deux paires de capteurs infrarouges, pour mesurer la distance entre les interfaces de connexions. Chaque module possède une jointure à 3 DOF et 4 faces de connexion. La face avant

du module est équipée d'une structure conique utilisée pour l'attachement avec la douille femelle présente à l'arrière du module. Les auteurs de ce robot proposent un algorithme basé sur la planification de mouvements pour contrôler le robot.

Smores

Smores [Davey et al., 2012] (Self-assembling Modular Robot for Extreme Shape shifting) a été développé par des chercheurs du laboratoire Modlab de l'université de Pennsylvanie. Ce robot modulaire homogène est capable de réorganiser ses modules selon les 3 classes de reconfiguration (treillis, chaîne et mobile). Les modules sont indépendamment mobiles et capables de s'auto-assembler et s'auto-reconfigurer. Chaque module possède 4 DOF et 4 connecteurs. En fait, le mécanisme de connexion est facilité par l'utilisation des aimants montés sur les roues. Les chercheurs ont pu effectivement construire un système capable de reproduire les capacités de mouvement de nombreux systèmes existants.

M-Blocks

Ce robot [Romanishin et al., 2013] modulaire a été créé par des chercheurs du MIT. Ses modules se sont de petits cubes qui peuvent faire des sauts et des rotations autour de leurs centres de gravité sans actionneurs externes. Ils utilisent pour cela des masses internes qu'on fait tourner à des vitesses élevées. Ensuite, l'arrêt soudain de ces masses fait transférer l'inertie au module, provoquant ainsi son déplacement. Pour connecter les modules entre eux et permettre une auto-reconfiguration, on a mis des aimants au niveau de chaque bordure du module. Le pivotement sur les bordures permet des reconfigurations sur un treillis 3D.

CoSMO

CoSMO (Collective Self-reconfigurable Modular Organism) [Liedke et al., 2013] est un robot modulaire hétérogène hybride, développé au sein des deux projets européens REPLICATOR et SYMBRION. Il peut être composé de 3 types de modules, dont chacun est conçu pour fonctionner en conjonction avec les autres modules. Toutefois, ils ont des caractéristiques individuelles différentes en fonction de la tâche pour laquelle ils sont conçus. Les modules possèdent 4 faces d'accrochage avec des jointures de 1 DOF. Ces dernières sont exceptionnellement puissantes et capables de soulever jusqu'à 4 modules d'un seul coup. Par ailleurs, les modules peuvent partager de l'énergie et se déplacer indépendamment dans toutes les directions principales. Par conséquent, une grande variété de topologies et de comportements seront possibles

3.4.2 Les robots modulaires manuellement reconfigurables

ModRed

ModRED [Dasgupta et al., 2013] (Modular Robot for Exploration and Discovery) est un robot modulaire qui a une architecture chaîne. Le robot physique n'est pas doté de capacité d'auto-reconfiguration, cependant en simulation quelques reconfigurations autonomes ont été expérimentées. Les auteurs visent en réalisant ce robot à améliorer les techniques d'exploration automatiques. Le robot doit alors surpasser les obstacles et se mouvoir sur des terrains non structurés, tels que la surface de la lune ou un cratère volcanique. Les modules de ModRED sont homogènes et chacun possède 4 DOF, 3 sont rotatifs et un est prismatique. Cela permet à chaque module de pivoter le long de son axe longitudinal de $\pm 90°$ ainsi de s'étendre le long de ce même axe. Le mécanisme de connexion se fait par des disques rotatifs.

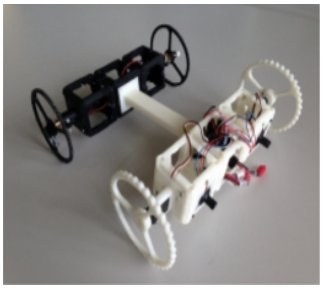
Mobot

Mobot [Gucwa and Cheng, 2014] est un robot modulaire de type chaîne développé par Barobo [2]. Il représente une version éducative du robot iMobot. L'objectif principal est de simplifier l'utilisation de ce dernier pour l'enseignement. Les modules de Mobot sont homogènes et peuvent comprendre plusieurs types d'accessoires. Par exemple, des disques rotatifs (roues) ou des pinces pour attraper des objets. Nous pouvons alors considérer le robot comme hétérogène. Chaque module dispose de 4 DOF (2 au niveau des extrémités et 2 au niveau des jointures) ainsi que 6 surfaces de connexion. La connexion entre les modules et les accessoires est manuelle. Pour contrôler l'ensemble des modules, on utilise un programme C++ sur ordinateur (avec interface graphique) qui envoie ses directives grâce à une liaison Bluetooth.

RoboGen

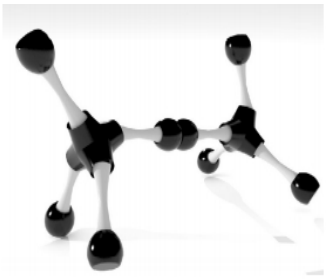
RoboGen [Auerbach et al., 2014] (Robot Generation Through Artificial Evolution) est une plate-forme open source destinée à réaliser des coévolutions artificielles des robots modulaires. La morphologie du robot est représentée par un arbre et elle est contrôlée par un réseau de neurones récurent entièrement connecté. L'avantage de cette plate-forme est que les robots évolués peuvent être facilement imprimés en 3D et le contrôleur peut fonctionner sur une carte Arduino. La plate-forme permet d'ajouter des variations à l'environnement, tel que l'ajout aléatoire d'obstacles, ce qui permet l'émergence de robots robustes. Le robot peut réaliser deux objectifs ; couvrir la plus longue distance en un temps donné (course) et de se rapprocher le

2. https://www.barobo.com/



(a) RoboGen

(b) Fable II



(c) PetRo

FIGURE 3.4 – Exemple de robots modulaires manuellement reconfigurables

plus possible d'une source lumineuse (poursuite). Les modules sont cubiques et ne peuvent pas s'auto-reconfigurer. Des modules de type jointure sont proposés pour former des connexions entre les modules. Au moment d'écriture de ce document, ils présentent 6 types de connexions. De ce fait, nous considérons le robot comme non homogène.

PetRo

PetRo (abréviation pour Pet Robot) [Salem, 2014] est un robot modulaire homogène créé pour être utilisé comme animal de compagnie expressif, et également pour des opérations de recherche et de sauvetage. Les modules ont des formes tétrapodes inespérées des chausse-trapes ; un corps central avec quatre bâtons qui sortent de lui formant des pattes. Ayant cette forme, le module peut être connecté à 4 autres. Les jointures liant les bâtons au corps central possèdent 2 DOF, tandis que celles des extrémités possèdent qu'une DOF. Par railleurs, un module peut se déplacer indépendamment en faisant des rotations et des translations. Un ensemble de modules peut effectuer des locomotions semblables à celles des animaux, notamment le chien.

Fable II

Fable II [Pacheco et al., 2015] est un système robotique modulaire puissant, mais aussi facile d'utilisation. En effet, les modules s'assemblent facilement, et seulement en quelques minutes. Ils sont aussi programmables via une simple Interface de Programmation Applicative (API), basée sur le langage de programmation éducatif LOGO. L'objectif des auteurs est de permettre aux utilisateurs non experts d'apprendre par interactions afin de concevoir et programmer leurs propres robots. Les modules Fable II sont hétérogènes et peuvent être actifs ou passifs. Le module actif contient un micro-contrôleur, une alimentation embraquée et un dispositif radio pour une communication sans fil. Il peut avoir une jointure de 1, 2 ou 3 DOF. Tandis que le module passif est un moule (de forme I, Y ou X) en plastique utilisé pour connecter

différents modules et donner des configurations de robots de type chaîne. Selon sa forme, il peut avoir 2, 3 ou 4 faces de connexions.

ChainFORM

Ken Nakagaki et son équipe ont proposé récemment le robot ChainFORM [Nakagaki et al., 2016]. Ce dernier est un robot modulaire chaîné composé de petits modules identiques (voir figure 3.5). ChainForm n'est pas capable de s'auto-reconfigurer. Les modules de ce robot ont la particularité d'offrir des capacités de détection tactile sur différentes surfaces. Ces senseurs permettent d'actionner les moteurs ainsi que les lumières du robot. Les modules forment souvent des serpents multifonctions dotés d'une communication basée sur une architecture linéaire. Comme le robot est de type chaîne, il possède 2 DOF et 2 connecteurs. Bien que l'accent des systèmes robotiques modulaires chaînes soit principalement mis sur la locomotion, les auteurs se sont concentrés sur la manière dans laquelle le matériel peut interagir avec les utilisateurs pour représenter l'information et détecter les entrants humains.

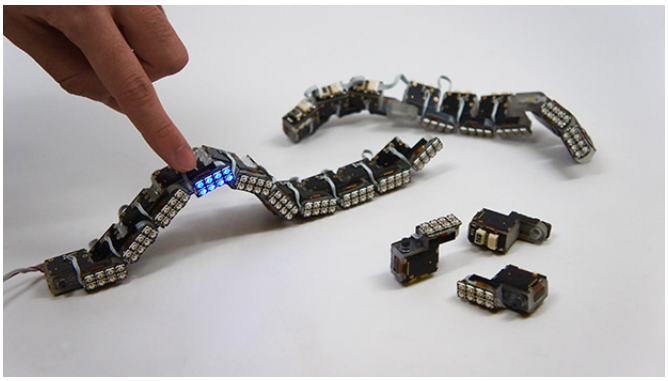


FIGURE 3.5 – Le robot modulaire ChainForm

3.5 Étude comparative

Nous proposons au début de cette partie, le tableau comparatif 3.1 qui récapitule les caractéristiques des robots retenus pour notre étude. Nous prenons en considération les critères suivants : architecture, homogénéité, capacité d'auto-reconfiguration et les propriétés physiques des modules. Dans le tableau, les robots surlignés en gris sont auto-reconfigurables.

Les travaux récents dans ce domaine ont permis le développement de robots modulaires variés et complexes. Certains sont hybrides et combinent les avantages du type treillis et chaîne de robots. L'un des robots hybrides le plus connu à avoir une conception mécanique robuste est M-tran. Plus récemment on a proposé d'ajouter des roues aux modules pour accroître les capacités de locomotion dans les terrains

Tableau 3.1 – Tableau comparatif de quelques robots modulaires

Robot	Architecture	Homogénéité	Forme	Nombre de DOF	Nombre de connexions	vitesse de DOF (rpm)	Moment de DOF (nm)	Poids du module (kg)	Dimension du module (mm)
M-tran III (2005)	Hybride (T,Ch)	Oui	Demi cylindre(x2)	2	6	14	1.9	0.42	130*65*65
SuperBot (2006)	Hybride (T,Ch)	Oui	Demi cylindre(x2)	3	6	20	0.5 - 2.1	1.2	168*84*84
RoomBot (2008)	Hybride (T,Ch)	Oui	Demi sphérique(x4)	3	10	19.4-26.6	3.6 - 4.9	1.4	220*110*110
Cross-Ball (2011)	Treillis	Oui	Sphère	2	6	-	-	-	Diamètre : 25.4
Transmote (2012)	Hybride (T,Ch)	Oui	Cuboïde(x3)	3	4	-	-	-	65*72*263
ModRed (2012)	Chaîne	Oui	Cuboïde(x3)	4	2	-	-	2.7	368*114*119
Mobot (2014)	Chaîne	Non	Cuboïde	4	6	-	-	-	-
Smores (2012)	Hybride (T,Ch,M)	Oui	Cuboïde	4	4	23	1.2 - 2.3	0.52	100*100*90
CoSMO (2013)	Hybride (T,Ch,M)	Non	Cuboïde	1	4	20	17.5	1.25	105*105*105
RoboGen (2014)	Chaîne	Non	Cuboïde	4	2, 4	-	-	-	46.5*46.5*46.5
M-blocks (2014)	Treillis	Oui	Cuboïde	1	6	-	-	0.143	50*50*50
PetRo (2014)	Chaîne	Oui	Tétrapode	1, 2	4	-	-	-	-
Fable II (2015)	Chaîne	Non	Cylindrique	1, 2, 3	2, 3, 4	-	-	0.3 - 0.55	Hauteur : 142-275
ChainFORM (2016)	Chaîne	Oui	Cuboïde	2	2	0.10 sec/60°	0.078	-	-

plats, par exemple le robot Smores et RoboGen. Il faut noter que certains des robots sont significativement avancé au niveau de la conception mécanique.

Généralement les modules sont relativement petits (de 5 à 10 *cm*) à l'exception de quelques-uns comme ceux de RoomBot et SuperBot, qui dépassent les 20 *cm* de hauteur. En fait, nous pensons qu'il est préférable de construire de petits modules qui ne pèsent pas lourd, afin de mieux contrôler le robot. Il faut cependant avoir l'espace nécessaire pour la partie électronique, tels que les moteurs, les batteries et les micro-contrôleurs.

À travers ce tableau, nous remarquons que les robots qui se reconfigurent en totale autonomie sont généralement homogènes avec une architecture treillis. Ils disposent des mécanismes de connexion avancés et sophistiqués (magnétique et

électromécanique), qui semblent être forts et flexibles. En fait, cela est indispensable pour assurer une connexion et une déconnexion autonome de modules. Cependant, les chercheurs affirment avoir des difficultés concernant les contraintes mécaniques pour la conception matérielle des robots. Par ailleurs, l'auto-reconfiguration consomme beaucoup d'énergie et exige une force considérable.

Bien que l'auto-reconfigurabilité soit l'objectif principal de la robotique modulaire, et ce depuis ses débuts, beaucoup de chercheurs s'orientent vers la conception de robots manuellement reconfigurables, tels que RoboGen, ChainForm, MOBOT. Nous remarquons que ce type de robots est exclusivement de type chaîne, parfois mobile. En outre, les modules peuvent être hétérogènes avec des fonctionnalités différentes grâce aux accessoires ajoutés. Les robots manuellement reconfigurables présentent l'avantage d'avoir des mécanismes d'attachement beaucoup plus simple que ceux auto-reconfigurables. Ils ont tendance à réaliser les tâches les plus utiles grâce à leurs capacités de former des membres articulés. Ces robots sont donc utilisés dans des situations où l'auto-reconfiguration n'est pas nécessaire.

Un autre élément à retenir de cette comparaison concerne les degrés de liberté. Ils permettent aux jointures de réaliser des rotations, et aussi fournir aux modules une flexibilité pour atteindre potentiellement tous points et orientations dans l'espace. Nous remarquons que le DOF varie selon les robots de 1 à 4, sans que cela ait une influence directe sur le niveau d'autonomie de reconfiguration. Cependant le type chaîne de robots nécessite le plus de DOF afin de bouger arbitrairement, tandis que le type treillis, avec uniquement 1 DOF, le module peut pivoter et se déplacer vers des positions voisines au sein de la grille.

Les robots retenus dans le cadre de cette étude utilisent des systèmes de contrôle différents. Nous proposons de les résumer dans le tableau suivant. La plupart des robots auto-reconfigurables utilisent des contrôleurs distribués. En fait, cela répond en partie au critère de robustesse, vu l'absence d'un contrôle global. Nous pouvons distinguer deux approches de contrôle ; celles sans apprentissage et celles avec apprentissage. Dans le premier type d'approche, on utilise des algorithmes de planification et des tables de mouvements, écrites sous forme de règles. Il est généralement adopté lorsque l'objectif des concepteurs est le développement cinématique des robots où la performance n'est pas une priorité. De plus, il est adapté lorsque les robots conçus sont destinés aux utilisateurs non professionnels. Concernant le deuxième type d'approche, il est utilisé pour réaliser des tâches de locomotions complexes. Il comporte des algorithmes évolutionnaires et bio-inspirés tels que vus dans le chapitre précédant (section 2.3.2).

Tableau 3.2 – Les contrôleurs utilisés

Contrôleur	M-tran III	SuperBot	RoomBot	Cross-Ball	Transmote	ModRed	Mobot	Smores	CoSMO	RoboGen	M-blocks	PetRo	Fable II	ChainForm
Maître/Esclave	x	x												
Planification					x	x	x	x	x		x	x	x	x
Inspiré d'hormones		x												
Évolué par les AE			x							x				
GRN				x										
RNA										x				
CPG			x											
Distribué	x	x	x	x										
Avec communication	x	x	x	x	x	x		x	x		x			

3.6 Conclusion

Dans ce chapitre nous avons donné un aperçu général sur les robots modulaires reconfigurables. Ce type de robots représente une révolution dans le domaine de la robotique, car ils apportent polyvalence, robustesse et un moindre coût de fabrication. Cependant, si le robot est complexe, sa conception devient difficile. La recherche dans ce domaine est actuellement très active. Les chercheurs proposent des systèmes robotiques différents en tentant toujours de faire des améliorations que se soit au niveau de la cinématique, des mécanismes de connexions ou des politiques de contrôle. À travers une étude comparative sur quelques robots modulaires réalisés récemment, nous avons pu retenir certaines caractéristiques à intégrer dans nos robots modulaires. En fait, afin de répondre à notre objectif et concevoir des robots résilients capables de réaliser des locomotions réalistes, il est important que les modules possèdent des DOF variés afin que le robot puisse réaliser différents types de locomotion. En outre, la communication entre les modules joue un rôle important pour échanger les informations et synchroniser les mouvements.

Chapitre 4

Simulateurs robotiques

4.1 Introduction

Les simulateurs jouent un rôle important dans le domaine de la robotique évolutionnaire. En effet, ce sont des environnements avancés de simulation, qui permettent de simuler le comportement de plusieurs types de robots et leurs interactions avec l'environnement. Ainsi, les chercheurs peuvent, avec cet outil, tester et évaluer de nouvelles stratégies et algorithmes de contrôle avant de les appliquer sur de vraies plates-formes robotiques. Par conséquent, on pourra multiplier les expérimentations tout en évitant les coûts et la durée d'une réalisation réelle en laboratoire.

Nous proposons donc, dans ce chapitre de présenter brièvement certains des simulateurs les plus connus dans le domaine ainsi que leurs principales caractéristiques. Ensuite, nous nous focaliserons sur ceux qui ont attiré le plus notre attention ; MORSE et Gazebo que nous décrirons en vue de réaliser un choix pour la suite de nos travaux.

4.2 Description des simulateurs

Plusieurs simulateurs sont donc proposés aux utilisateurs, chacun a ses avantages et ses inconvénients. Ces simulateurs peuvent être libres ou propriétaires (commerciaux). Selon les caractéristiques du simulateur et selon son domaine d'application, l'utilisateur peut choisir celui qui répondra le plus à ses besoins.

Dans le domaine de la robotique, parmi les simulateurs les plus connus, nous pouvons citer, Gazebo [Koenig and Howard, 2004] et MORSE [Echeverria et al., 2011] qui sont deux simulateurs libres et multi-robots, ReMod3D [Collins et al., 2013] qui est assez récent et spécialement dédié aux robots autonomes reconfigurables.

Comme on peut trouver d'autres simulateurs qui sont propriétaires, par exemple le simulateur de robot mobile Webots [1] et V-Rep PRO [2].

4.2.1 Le moteur physique

Le moteur physique est un composant important, car il influence grandement la performance et la scalabilité du simulateur, ainsi que sa capacité à supporter des structures complexes. [Collins et al., 2013] C'est une bibliothèque indépendante qu'on peut intégrer dans des environnements de simulation et qui sert à résoudre les équations de la physique et les problèmes de la mécanique classique, tel que les collisions, les forces et la cinétique. Parmi les moteurs les plus connus, nous examinerons : Open Dynamics Engine (ODE), Bullet et PhysX. [Roennau et al., 2013]

ODE

ODE [3] est libre et sert à simuler l'interaction physique de corps rigides d'une manière précise. Il possède une riche documentation et est utilisé comme le moteur physique 3D primaire dans de nombreux simulateurs robotiques comme Gazebo, Webots et V-Rep. En utilisant ODE, les simulateurs montrent une haute fidélité dans la simulation tridimensionnelle de robots complexes. [Collins et al., 2013] La limite de ce moteur telle que reportée dans la littérature [Collins et al., 2013, Roennau et al., 2013] consiste, à ce que la performance et la fidélité de la simulation diminuent avec l'augmentation du nombre de modules et en particulier de petits blocs. Cela a pour cause des échecs de détection de collisions et provoque un ralentissement de la vitesse de simulation. Ceci rend ODE inefficace pour la simulation des robots modulaires (avec beaucoup de modules). Il est aussi reporté que ODE ne permet pas l'intégration de calculs sur GPU, ni le support du multithreading pour le calcul physique.

Bullet

Bullet [4] est un moteur physique open source qui est généralement utilisé dans les jeux vidéo et pour réaliser des effets visuels dans les films. Il est utilisé comme base de calcul physique par plusieurs simulateurs dont V-rep et MORSE. Contrairement à ODE, Bullet gère bien les collisions [Roennau et al., 2013] et offre une performance scalable même avec des structures complexes de robots. [Collins et al., 2013] Il permet

1. http://www.cyberbotics.com/
2. http://www.coppeliarobotics.com/
3. http://www.ode.org/
4. http://bulletphysics.org/wordpress/

non seulement de simuler la dynamique des corps rigides, mais aussi celle des corps mous, ainsi que les systèmes de particules et les fluides. Ce moteur supporte le calcul physique multithread. Cependant, au moment d'écriture de ce chapitre, la documentation reste assez limitée et l'API est en constant changement.

PhysX

PhysX [5] est un moteur physique propriétaire (le PhysX SDK est libre). Il a été créé par la société AEGIA et fut racheté par la suite par NVIDIA. Il est utilisé principalement par plusieurs développeurs de jeux vidéos, car il permet de gérer en temps réel les collisions et les processus physiques les plus complexes. Il supporte la dynamique des corps rigides et mous, la dynamique des véhicules et la simulation de systèmes de particules, des fluides et des tissus. Il permet le calcul multithread et est nativement supporté par les cartes graphiques.

4.2.2 Gazebo

Gazebo [Koenig and Howard, 2004] est un simulateur multi-robots open source qui a été créé par Andrew Howard et son étudiant Nate Koenig au laboratoire USC Robotics Research Laboratory en 2004. Il est capable de simuler dans des environnements tridimensionnels réalistes une population de robots, capteurs et objets. Il est l'un des simulateurs 3D les plus populaires [Ivaldi et al., 2014, Echeverria et al., 2011] et a une communauté d'utilisateurs très active.

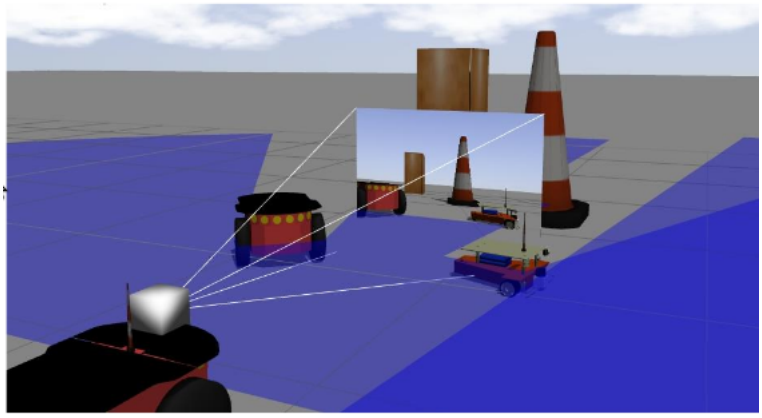


FIGURE 4.1 – Un robot mobile Pioneer équipé d'un télémètre laser et d'une caméra

Ce simulateur peut intégrer plusieurs moteurs physiques ; à savoir ODE (par défaut), Bullet, Simbody et DART. On peut alors utiliser le moteur physique que nous souhaitons en changeant uniquement un seul paramètre. Le système de visualisation

5. http://www.geforce.com/hardware/technology/physx

par défaut est OpenGL[6], mais on peut utiliser le moteur de rendu graphique 3D OGRE[7]. Ce dernier fournit un rendu 3D réaliste, un éclairage de haute qualité, ainsi que les ombres et les textures. La figure 4.1 représente une scène simulée avec Gazebo qui permet de voir un robot mobile Pioneer de la société ActivMedia doté d'un télémètre laser et d'une caméra.

Gazebo est séparé en deux parties, un serveur et un client. Le serveur gère le moteur physique et la génération des données des capteurs, tandis que le client est responsable de la visualisation et de l'interface graphique. L'architecture de Gazebo est illustrée dans la figure 4.2. En lançant Gazebo sans l'interface graphique (*mode Headless*), on gagne en temps de calcul et en performance. [Koenig and Howard, 2004]

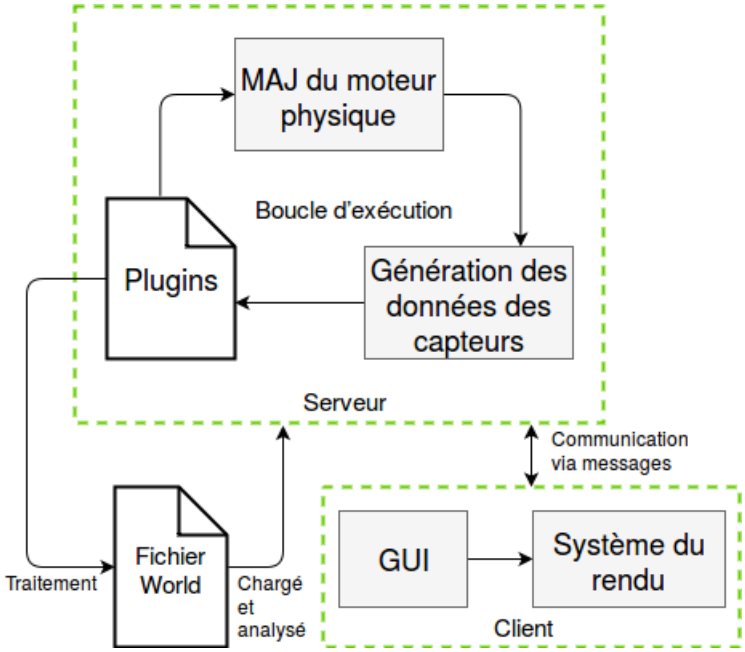


FIGURE 4.2 – Architecture du simulateur Gazebo

Les éléments impliqués dans l'exécution d'une simulation Gazebo sont :

- **Le fichier « world » :** Il s'agit d'un fichier Extensible Markup Language (XML) de format Simulation Description Format (SDF) qui est analysé par le serveur. Il permet de spécifier les propriétés globales de la scène simulée, tels que le moteur physique utilisé, la force de gravité et l'éclairage. Dans ce fichier on peut aussi spécifier les modèles et les plugins qui vont être chargés dans l'environnement.

6. http://www.opengl.org/
7. http://www.ogre3d.org/

- **Les fichiers « model » :** Ces fichiers de format SDF décrivent en XML les modèles (robots) qui doivent être simulés dans l'environnement. Un modèle est une collection de liens, de jointures, de capteurs et de plugins. Les corps sont connectés les uns aux autres par des jointures. Les jointures sont contrôlées par des plugins. Il est possible d'importer différents formats de modèles pour construire des environnements encore plus réalistes
- **Les plugins :** Un plugin est un morceau de code (C++) qui est compilé en tant que bibliothèque partagée pour modifier la simulation en temps réel. Ce code peut être attaché à des modèles, des capteurs, à l'environnement ou au simulateur lui-même. Par exemple, un plugin modèle pourra permettre de contrôler les jointures d'un robot. Les plugins fournissent aux utilisateurs un accès direct aux propriétés physiques des « models » et aux bibliothèques de Gazebo via des classes standard C++.

4.2.3 MORSE

MORSE "Modular Open Robots Simulator Engine" [Echeverria et al., 2011] est un simulateur open source orienté robotique initialement développé au sein du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS). Il est basé sur l'environnement de modélisation Blender [8] et sur le moteur physique Bullet. L'architecture modulaire et composable de MORSE lui permet de simuler plusieurs robots mobiles et hétérogènes dans tout type d'environnement (aérien, terrestre, maritime). MORSE est adaptable à différents niveaux de réalisme et utilise la philosophie "Software-in-the-Loop" qui permet d'évaluer en simulation exactement les mêmes logiciels que ceux qui sont embarqués sur le robot réel. [Degroote, 2012] La figure 4.3 ci-dessous représente une scène simulée avec MORSE.

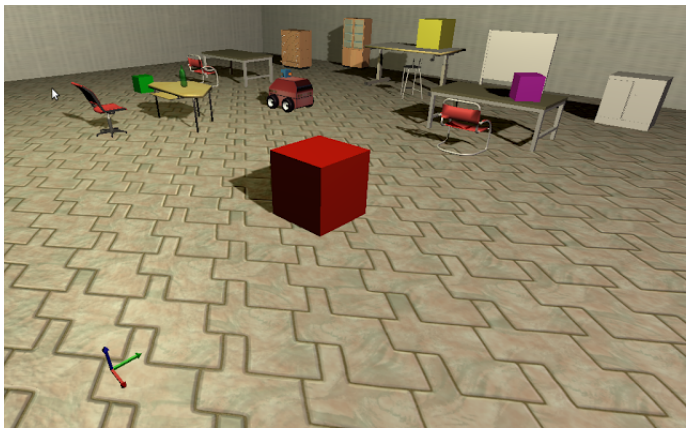


FIGURE 4.3 – Une scène simulée avec MORSE

8. http://www.blender.org/

Les composants principaux d'une simulation MORSE sont les robots, les actionneurs et les capteurs. Ces composants vivent dans un environnement et peuvent être reliés à des interfaces et des modificateurs. Ces derniers permettent d'ajouter des bruits, des filtres, et de manière générale de transformer les données calculées par le capteur, afin de les rendre plus réalistes (moins parfaites), ou bien les adapter au mieux au type de sortie attendue. [Degroote, 2012] Morse utilise aussi des middlewares. Ce sont des logiciels qui créent un réseau d'échange d'informations entre les composants logiciels. En effet l'utilisation d'un middleware dans des systèmes robotiques est impérative, puisque ces derniers sont généralement des systèmes complexes qui nécessitent des interactions et des communications entre des éléments matériels et logiciels hétérogènes. Parmi les plus utilisés, nous citons OROCOS, Orca, ROS et BRICS.

L'utilisateur doit écrire différents scripts sous python pour pouvoir construire la scène de simulation :

- **Script Builder :** Ce script repose sur l'API de Builder pour définir les composants, l'environnement et les middlewares qui vont être utilisés dans la simulation. Chacun des composants peut utiliser un middleware différent, ce qui permet l'utilisation de Morse dans un environnement hétérogène. Les propriétés visuelles et dynamiques des composants sont spécifiées dans des fichiers Blender.
- **Script client :** Ce script repose sur l'API pymorse. Il permet de se connecter avec MORSE via Python pour contrôler un robot.

La figure 4.4 représente l'architecture de MORSE. Pour créer une scène de simulation, MORSE interprète le script «Builder» et charge l'environnement ainsi que tous les composants dans le moteur 3D Blender. Ensuite, les composants sont liés aux middlewares spécifiés dans le script, pour être en mesure de transmettre et de recevoir les données avec des programmes externes.

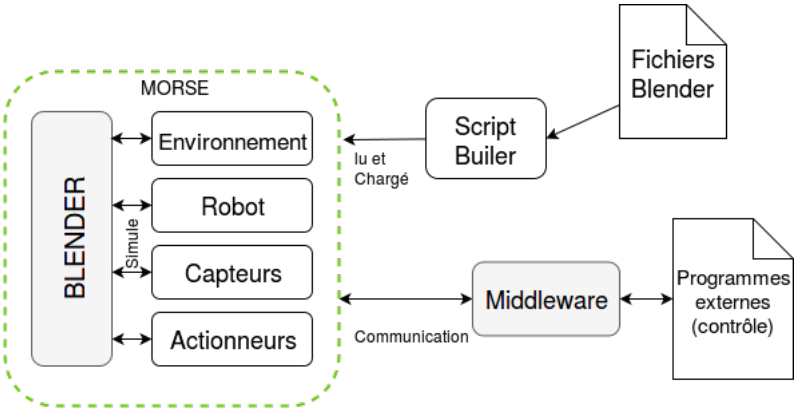


FIGURE 4.4 – Architecture du simulateur MORSE

4.3 Comparaison entre MORSE et Gazebo

Nous entamons notre comparaison par le tableau 4.1 suivant dans lequel nous présentons brièvement les caractéristiques de plusieurs simulateurs multi-robots. Nous avons aussi inclus ReMod3D qui ne fait partie de ces précédents, mais nous avons choisi de l'inclure dans notre comparatif, car il a été spécialement conçu pour les robots modulaires.

Tableau 4.1 – Comparaison de simulateurs

	MORSE	**Gazebo**	**V-Rep**	**Webots**	**ReMod3D**
Multi-robot	Oui	Oui	Oui	Oui	Non
Environnement réaliste	Oui	Oui	Oui	Oui	Non
Licence	Open source (BSD)	Open source (apache 2.0)	double licence ; comerial pour V-Rep pro et libre (GNU GPL)	commercial	Open source (GNU GPLv2)
Système de rendu	Blender	OpenGL, OGRE	Interne	OGRE	OpenGL
Moteur physique	Bullet	ODE, Bullet, Simbody, DART	ODE, Bullet, Vortex	ODE	PhysX
Accès directe au moteur physique	Non	Oui	Oui	Oui	-
Plate-forme	Linux, Mac OSX	Linux, Mac OSX, Windows	Linux, Mac OSX, Windows	Linux, Mac OSX, Windows	Linux, Mac OSX, Windows
Simulation *Headless* [9]	Oui	Oui	Oui	Oui	Non
Portabilité [10]	Oui	Oui	Uniquement avec ROS	Oui	-
Date de première publication	2010	2004	2010	1998	2013
Langage de programmation	Python	C++	Lua, C/C++, Python, Java, Matlab, Urbi	C/C++, Java, Python, Matlab, Urbi	C++
Format de description du robot	Python builder API	SDF, URDF	URDF	VRML'97	-
Middleware	YARP, ROS, Pocolibs, MOOS, Sockets	ROS, Player, Sockets	ROS, Sockets	ROS	-
Manuel d'utilisation	Oui	Oui	Oui	Oui	Non
Forum publique	Non	Oui	Oui	Oui	Non

Comme le montre ce tableau ci-dessus, Gazebo et MORSE sont deux simulateurs multi-robots, qui peuvent simuler des morphologies complexes de robots dans des environnements réalistes. Ils sont libres, leur code source est disponible et ils sont supportés pour les systèmes Linux et Mac OS X. Gazebo utilise C++ comme langage de programmation, tandis que MORSE utilise Python, mais offre aussi la possibilité de programmer en C/C++ en utilisant Simplified Wrapper and Interface Generator (Swig) pour l'interfacer avec Python.

Gazebo utilise par défaut le moteur physique ODE, mais peut aussi utiliser, tout comme Morse, Bullet. Comparé à ODE, Bullet est plus performant, cependant ODE dispose d'une riche documentation et sa capacité concernant la stabilité des objets est bonne. Toutefois, même si Morse utilise Bullet, on lit dans le site officiel de Morse [11] que nous ne pouvons pas encore considérer Morse comme un simulateur physique précis.

La description d'une scène de simulation dans Gazebo nécessite d'écrire un script en XML, ce qui peut être lent et difficile. De la même manière, MORSE permet de tout modéliser via une API Python. Il offre cependant une interface native permettant la modélisation via l'interface graphique de Blender. Cette dernière fournit une vue d'ensemble et un accès immédiat sur les propriétés que offertes pas MORSE. Ces deux simulateurs présentent l'avantage de pouvoir importer des modèles CAD de différents formats.

4.4 Choix du simulateur

Avant d'entamer notre travail, il est important de déterminer dès le départ le simulateur qui pourrait convenir le plus à nos besoins. Nous rappelons qu'il est important pour nous de réaliser une simulation physique réaliste et performante dans le but de pouvoir tester d'une manière efficace nos contrôleurs. De plus, pour pouvoir contrôler des structures complexes de robots d'une manière appropriée, il est nécessaire d'utiliser les principes d'évolutions qui exigent de mener de nombreuses expériences.

Ceci nous mène à considérer les caractéristiques suivantes :

- Rapidité de la simulation,
- Performance et qualité de la simulation physique,
- Extensibilité à de nouveaux types de capteurs et de contrôleurs,
- Simplicité d'implantation de nouveaux modules,
- Capacité d'intégration d'un processus d'évolution artificiel.

11. www.openrobots.org

Nous allons utiliser le simulateur, principalement, pour évaluer nos robots des centaines de milliers de fois. Il est donc nécessaire qu'il puisse fournir des évaluations le plus rapidement possible, et ce pour gagner du temps en apprentissage. Ce qui est intéressant alors est la simulation *Headless* et l'accès direct au moteur physique, qui permet d'éviter toute une couche de communication. En outre, il est indispensable de lancer plusieurs évaluations de robots à la fois, afin de pouvoir paralléliser le calcul et les évaluations des robots. La précision de simulation est importante, puisque plus c'est précis plus on peut éviter le problème de la *Reality Gap* (voir la section 4.5). Cependant, il est reporté dans la littérature, que peu importe la précision du simulateur, ce problème est inévitable.

Avec Gazebo, l'accès direct aux propriétés du moteur physique, et la simulation sans la visualisation sont possibles .La performance de la simulation dépend du moteur physique utilisé, puisqu'on a la possibilité d'utiliser soit Bullet ou ODE. L'extensibilité de nouveaux types de capteurs et de contrôleurs est relativement simple, ceci grâce à son API. La structure modulaire de son architecture lui permet facilement l'implantation de nouveaux modules.

MORSE exige de passer par un moyen de communication pour contrôler un robot. La parallélisation et relativement difficile. Donc cela va ralentir le processus d'évolution artificielle. MORSE utilise le moteur Bullet. Ce dernier permet une simulation physique réaliste. Tout comme Gazebo, dans MORSE, on peut ajouter des capteurs et des contrôleurs personnalisés. L'implantation de nouveaux modules est simple grâce à son architecture modulaire.

Après avoir installé et essayé les deux simulateurs, et sur la base des caractéristiques que nous avons énumérées ci-dessus, nous avons choisi le simulateur Gazebo. Bien que Morse présente des caractéristiques très intéressantes, nous trouvons que Gazebo est plus facile à utiliser. De plus, il est extensible et relativement performant par rapport au processus d'évolution artificiel. Donc, Gazebo est le simulateur le plus approprié dans le cas de l'étude que nous allons entreprendre.

4.5 Reality Gap

Les simulateurs robotiques offrent la possibilité aux chercheurs de mettre en œuvre et d'explorer leurs idées sans faire des tests sur des robots physiques. Cependant, peu importe la précision du simulateur, cela ne peut pas remplacer le monde réel pour pouvoir évaluer les robots de manière précise. En effet, les myriades de contraintes que nous en trouvons dedans ne peuvent pas être toutes modélisées et simulées. Le chercheur Rodney Brooks avait déclaré en 1990 que « *The world is its own best model*» et qu'aucun système ne peut modéliser parfaitement le monde réel. [Brooks, 1990] En outre, durant l'étape d'apprentissage, le processus d'évolution peut éventuellement

générer des solutions qui sont réalisables uniquement dans le contexte de la simulation, et ce en exploitant les erreurs de modélisations ou les instabilités présentes dans le simulateur. [Hupkes et al., 2018] A cet effet, ce que nous obtenons en simulation et qui répond à nos attentes peut échouer en monde réel. Cela cause une problématique pour les chercheurs qui souhaitent essayer leurs contrôleurs sur des robots physiques. Ce phénomène est connu au sein de la communauté robotique sous le nom de «*Reality Gap*».[Jakobi et al., 1995]

Afin de combler l'écart entre la perception et la réalité, différentes approches et méthodes ont été proposées dans la littérature. Par exemple, développer des contrôleurs robustes a été beaucoup considéré. L'écart dans ce cas est vu comme étant des perturbations que le contrôleur évolué doit pouvoir les rejeter. Les auteurs dans cette référence [Scheper and de Croon, 2017] ont pu optimiser des contrôleurs robustes avec un niveau d'abstraction élevé. Ajouter des bruits [Jakobi et al., 1995] aux simulations pour permettre une meilleure perception contribue à augmenter la robustesse des contrôleurs.[Doncieux et al., 2015] Cela permet de ne pas s'appuyer sur les parties sensibles du conception des simulateurs. Dans des travaux les plus récents [Cully et al., 2015, Koos et al., 2013], les auteurs ont essayé, sans chercher à réaliser des simulations réalistes, d'évoluer des contrôleurs efficaces à la fois sur le plan simulation et réalité. Cependant, la plupart des évaluations sont faites en simulation. Ils ont conçu les algorithmes *T-resilience* et IT&E que nous avons évoqués précédemment (section 2.4) et ils les ont testés sur un robot hexapode. Bien que ces approches fonctionnent plus ou moins bien, mais elles restent limitées et peu fiables. Comme alternative à ce type d'approche, on peut soit créer une nouvelle génération de simulateurs [Bongard et al., 2006, Pretorius et al., 2013, Mouret and Chatzilygeroudis, 2017, Klaus et al., 2012], soit faire des expériences et des évolutions directement sur de vrais robots physiques [Eiben and Smith, 2015, Eiben et al., 2013, Rieffel et al., 2017]. Certains chercheurs présument qu'il ne peut pas y avoir de simulateurs assez précis pour faire des évolutions de robots tout en évitant le problème de *Reality Gap*. [Rieffel et al., 2017, Mouret and Chatzilygeroudis, 2017]

Récemment, en 2017, John Rieffel avec un groupe de chercheurs a proposé de regrouper sous le terme Évolution des Systèmes Physiques (EPS), toutes les approches évolutives qui se produisent dans le monde réel sur des plates-formes physiques plutôt que dans des environnements de simulation. [Rieffel et al., 2017] Parmi ces approches de l'EPS, nous trouvons les techniques classiques dont on fait évoluer les contrôleurs, les uns après les autres, sur un seul robot. [Floreano and Mondada, 1994] Cependant, le nombre des tests à effectuer sur la machine est limité et l'évolution peut s'interrompre si le robot se casse. Nous trouvons aussi, les techniques de l'évolution incarnée parallèle "*Parallel Embodied Evolution*", [Watson et al., 2002, Haasdijk et al., 2014] dans lesquelles l'évolution est répartie sur une

population de robots. Théoriquement, cela accélère en conséquence le processus d'évolution par un facteur égal au nombre de robots. [Doncieux et al., 2015]

Les chercheurs A. E. Eiben et J. Smith [Eiben et al., 2013] se sont intéressés dernièrement à l'étude de l'évolution en ligne des robots. Ils proposent pour cela un nouveau cadre conceptuel (modèle théorique) qu'on a appelé le triangle de vie. Il comprend trois étapes ; la morphogénèse, l'enfance et la vie mature. Il s'agit d'une approche dans laquelle tout le processus de co-évolution des corps des robots et de leurs contrôleurs se déroule dans l'espace réel et en temps réel. En d'autres termes, au lieu de réaliser l'évolution avant le déploiement des robots dans les terrains, ils évoluent en temps réel et doivent vivre et exécuter leurs tâches de manière compétitive dans le même environnement physique.

Toutefois, afin de rendre tout cela possible en pratique, des technologies mécaniques avancées doivent être mises en œuvres et déployées dans les terrains. Il est nécessaire de construire des robots autoréplicateurs qui vont permettre l'autogénération, c'est à dire, fabriquer de manière autonome des copies d'eux-mêmes en utilisant des matières premières prises de leurs environnements. [Ellery, 2017] Ces robots doivent être équipés par des capteurs et des actionneurs robustes. En outre, il faut utiliser des machines qui produisent de manière autonome des robots adaptés à chaque génération du processus d'évolution, telles que les techniques de prototypage rapide d'impression 3D. [Hupkes et al., 2018, Kuehn and Rieffel, 2012, Ellery, 2017]

Néanmoins, jusqu'à présent, la plupart des expériences sont faites au niveau des simulations. Faire évoluer des robots physiques reste un large domaine abstrait, qui exige beaucoup d'efforts pour avancer dans la recherche. Il faut atteindre un niveau appréciable de technologie mécanique qui va ainsi permettre probablement la fabrication autonome de robots. Mais c'est une voie qu'il faut entreprendre pour réaliser de futurs robots révolutionnaires. Dans le cadre de cette thèse, notre concentration va vers l'étude et la conception des systèmes de contrôle, des morphologies robotiques et la génération des robots résilients. Construire les robots et leur intégrer nos contrôleurs performants et adaptés est une deuxième étape envisageable à l'avenir. Pour cela nous avons choisi le simulateur «crédible» Gazebo qui va d'une certaine manière nous permettre de réaliser des simulations qui se rapprochent le plus possible de la réalité et nous espérons que ceci nous aidera à mieux combler l'écart entre la perception et la réalité.

4.6 Conclusion

Ce chapitre nous a permis de souligner l'importance des simulateurs dans le domaine de la robotique en général et par rapport à notre étude en particulier. En effet, comme nous l'avons déjà dit, ces outils permettent de simuler le comportement

et l'interaction des robots avec des environnements réalistes en particulier pour tester de nouvelles stratégies de contrôles. Nous avons établi un bref comparatif entre quelques simulateurs utilisés dans le domaine en question. Aussi nous avons réalisé une comparaison plus détaillée entre Morse et Gazebo et qui s'est soldée par le choix de Gazebo. Ce dernier nous permettra de simuler et faire évoluer différentes morphologies de robots, ainsi que pour tester et valider nos contrôleurs. Au final de ce chapitre, nous avons discuté le problème de *Reality Gap*, qui empêche les chercheurs à transférer leurs contrôleurs évolués en simulation à de vrais robots physiques.

Dans le prochain chapitre, nous allons décrire le modèle que nous avons proposé pour la génération de robots modulaires résilients. Le processus d'évolution qui permettra la co-évolution des morphologies et des contrôleurs sera intégré dans le simulateur Gazebo afin qu'il puisse générer des structures robotiques réelles avec des comportements crédibles.

Chapitre 5

Le modèle proposé

5.1 Introduction

Ce chapitre traite le modèle de génération de robots modulaires proposé dans le cadre de nos études. Dans un premier temps, nous présenterons nos robots modulaires et expliquerons comment les morphologies et les contrôleurs ont été générés ensemble. Par la suite, nous décrirons la manière dans laquelle nous avons intégré notre système d'évolution dans le simulateur multi-robot Gazebo. Nous présenterons également les difficultés que nous avons rencontrées à cet égard et les méthodes que nous avons adoptées pour y faire face. Le modèle a la particularité de produire des robots très réalistes, et ce grâce au simulateur Gazebo et à l'ajustement correct de ses attributs physiques.

5.2 Le robot modulaire

Dans le but de trouver automatiquement des solutions inédites de robots, où l'ingénieur ne possède pas la capacité d'assumer quelle morphologie serait la plus optimale pour une situation donnée, nous nous sommes orientés vers des morphologies de robots variables. Cependant, la morphologie ne change que via évolution et reste figée durant la durée de simulation. Le corps du robot est donc généré automatiquement avec un contrôleur embarqué.

La méthode de génération de nos robots est inspirée par les travaux de Karl Sims [Sims, 1994b, Sims, 1994a], qui consistent à faire évoluer des créatures artificielles simulées. Nous avons utilisé les *Graphtals* qu'il a employés pour la représentation génétique des robots (morphologie et contrôleur). Le génotype est ensuite interprété pour réaliser le phénotype correspondant. En fait, les travaux de Sims sont toujours considérés comme des travaux de référence et aucun autre travail n'a pu les dépasser

considérablement, que ce soit en termes de complexité ou de diversité des créatures obtenues. En outre, la représentation génétique à base de *Graphtals* a permis de générer naturellement la modularité et la symétrie au niveau de morphologies.

5.2.1 La morphologie

Nos robots sont modulaires. Ils se composent d'un certain nombre de modules parallélépipèdes hétérogènes liés par des jointures de tailles égales. Ajouter à cela d'autres géométries de modules (cylindrique, roue, etc.) et faire soumettre les tailles des jointures à évolution est envisageable dans des travaux futurs. Cela contribuera à avoir des morphologies plus variées [1]. Un module cuboïde peut avoir au plus six jointures, dont la probabilité de se retrouver toutes sur la même face est possible. Un module peut contenir ou ne pas contenir de capteurs. Un exemple d'un robot est illustré dans la figure ci-dessous. En fait, la structure modulaire des créatures convient au processus d'évolution, car elle lui permet la modification des structures uniquement par le réarrangement des segments et des blocs.

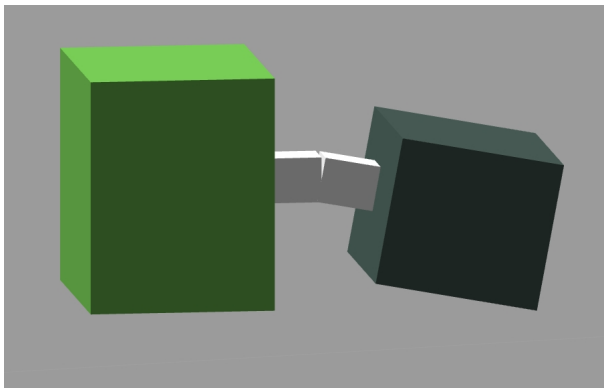


FIGURE 5.1 – Un exemple de robot modulaire. Il se compose de deux modules de tailles différentes liés par une jointure, présentée elle aussi par un ensemble de deux modules.

Représentation génétique

Pour générer les robots (morphologie dotée d'un contrôleur) nous avons utilisé les *Graphtals*. Ces derniers sont devenus un choix commun pour les chercheurs, vu leurs flexibilités et capacités de produire des formes modulaires et symétriques. Les *Graphtals* sont des graphes orientés composés d'un ensemble de nœuds et de connexions. Chaque nœud du graphe contient les paramètres phénotypiques qui vont décrire le module correspondant tel que, les dimensions, les capteurs intégrés, les axes

1. Il faudra dans ce cas avoir des algorithmes de recherches efficaces qui permettent d'explorer et de fouiller tout l'espace de recherche

de rotation des jointures et leurs positions par rapport à la face de connexion. Le nœud contient aussi un ensemble de neurones composant son contrôleur local. Nous expliquons en détail le système de contrôle dans la section 5.2.2. En ce qui concerne les liaisons, elles vont déterminer le plan d'assemblage des modules. Autrement dit, de quelle manière chaque module va être positionné par rapport aux autres modules voisins. La récursivité [2] et la réflexivité dans le graphe permettent la réutilisation de nœuds. Donc, il est possible d'avoir des modules dupliqués montrant des formes symétriques. Cela par conséquent est intéressant afin d'obtenir des mouvements et des comportements coordonnés et efficaces, comme ceux des animaux.

L'ensemble des paramètres et variations définis au niveau du génotype pour générer le robot, que se soit au niveau de son apparence physique ou à son contrôle, détermine l'espace de recherche. Nous parlons par exemple des méthodes d'application de la récursivité et de la réflexivité, des géométries des modules, des jointures qui peuvent être statiques avec des angles prédéterminés ou alors dynamiques avec des axes de rotation combinés. Plus cet espace de recherche est large, plus on peut avoir des phénotypes plus variés, et donc augmenter les chances de trouver des robots très intéressants pour différentes situations. Cependant, cela reste un défi dans le domaine de la robotique évolutionnaire tel que présenté dans la section 2.5.

Le graphe est représenté dans notre travail par une matrice d'adjacence. Il est créé aléatoirement avec les paramètres suivants :

- Un graphe peut contenir de 1 à 4 nœuds et peut avoir au maximum 3 nœuds réflexifs et 3 nœuds récursifs,
- La récursivité au niveau d'un nœud peut être appliquée jusqu'à 10 fois tandis que la réflexion peut être appliquée jusqu'à 4 fois,
- Le facteur d'échelle peut être dans cet intervalle [0.7 1],
- Préciser quel nœud sera le nœud racine,
- Préciser la méthode de récursivité et de réflexion pour chaque nœud,
- Préciser la méthode de parcours du graphe afin de générer la structure finale du robot.

Du génotype au phénotype

Tout d'abord, la génération de la morphologie commence à partir du nœud racine du graphe. Une fois créés, les nouveaux modules sont affectés par un unique identifiant et un parent, le module racine n'ayant pas de parents. La récursivité et la réflexion sont ensuite appliquées au fur et à mesure que nous parcourons le graphe. Afin de réduire la complexité, les cycles ne sont pas permis. Finalement, le but est

2. Via les self-loop

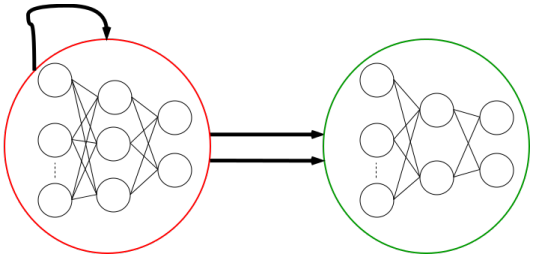


FIGURE 5.2 – Un exemple typique d'un génotype. Le graphe externe décrit la morphologie (assemblage et caractéristiques des modules) tandis que les graphes internes présents sur chaque nœud représentent les contrôleurs neuronaux

de générer un arbre qui définira le phénotype. Un exemple d'un génotype est illustré dans la figure 5.2 dont le phénotype correspondant est illustré dans la figure 5.3. Il faut noter que nous appliquons deux méthodes de parcours de graphe ; le parcours par profondeur et le parcours par largeur. Avec le même graphe, chaque méthode donne une structure différente avec un nombre de modules variés, pouvant atteindre la centaine. Au fait, dans notre cas, le nombre de modules joue un rôle important sur l'efficacité comportementale du robot. Si ce dernier possède un nombre important de modules (plus de 20), son contrôle devient complexe. Le simulateur va mal gérer la physique et va produire des simulations lentes. De ce fait, nous éliminons un nombre de modules pour ne pas dépasser un seuil prédéfini.

Dans le but d'élargir l'espace de recherche et avoir une diversité de morphologies, nous proposons deux méthodes d'application de récursivité : 1) répéter le module et l'ensemble de ses enfants pour toutes les réplications 2) répéter le module lui seul n fois et à l'extrémité introduire les modules enfants. Quant aux réflexions, nous répétons les modules connectés à la liaison réflective. Les nouveaux modules répliqués seront ajoutés dans une face prédéterminée. Cette dernière peut être elle-même la face d'où sort la connexion réflective. Dans ce cas, avec un seul nœud racine, des quadrupèdes et des morphologies de type main sont possibles.

Puisque nous avons introduit un facteur d'échelle décroissant, la taille des modules créés à partir du même nœud peut changer. Ensuite, après avoir créé tous les modules et avoir déterminé les liens entre eux, il nous ne reste plus qu'à calculer la position et les caractéristiques physiques de chacun d'entre eux (la masse, le moment d'inertie et la force maximale permise (le couple) de chaque jointure).

5.2.2 Le système de contrôle

Un système de contrôle est un outil qui permet de contrôler les jointures d'un robot et de les mettre en rotation tout au long de la simulation. En effet, il permet de générer un signal continu qui varie au cours du temps tout en tenant compte des valeurs issues des capteurs. Donc à chaque pas de temps il produit des paramètres

qui vont déterminer les forces qui vont par la suite être appliquées aux moteurs. Pour contrôler des robots autonomes et résilients, il est nécessaire de concevoir des systèmes de contrôle efficients pour pouvoir répondre rapidement aux conditions externes et internes changeantes.

Au cours de l'état de l'art, nous avons pu discerner plusieurs types de contrôleurs, notamment les systèmes de classifieurs, les RNA et les GRN. Cependant, on a principalement utilisé des RNA pour la conception de contrôle des créatures artificielles évoluées. Dans le cadre de nos travaux, nous avons choisi d'utiliser des RNA de type Perceptron Multicouche (MLP) [Hagan et al., 1996]. En plus du fait qu'ils donnent de bons résultats, ils permettent la modularité [Bongard et al., 2015]. Cela va apporter un avantage pour nos robots de structures modulaires.

Nous proposons donc de concevoir un système de contrôle distribué. Chaque robot possède alors deux niveaux de contrôleurs : un contrôleur local de bas niveau et un contrôleur global de haut niveau. Le contrôleur local est inclus dans chacun des modules du robot. Il va activer la jointure qui relie le module à son module parent. Par conséquent, le module racine n'a pas de contrôleur local. Les modules dupliqués qui ont été construits à partir du même nœud auront le même contrôleur. En ce qui concerne le contrôleur global, il représente le cerveau du robot et il est positionné au niveau du module racine. Le cerveau a la possibilité de contrôler toutes les jointures pour modifier globalement le comportement du robot lorsque cela est nécessaire. C'est-à-dire, intervenir à tout moment et non pas à chaque pas du temps, par exemple lors des changements inattendus dans l'environnement (apparition soudaine d'un obstacle, etc.) ou par rapport à la morphologie du robot. On peut considérer cela comme un réflexe. En fait, actuellement, cette fonctionnalité peut ne pas avoir d'effet sur le comportement du robot puisque l'environnement jusqu'à présent reste simple et non complexe. Mais on peut étudier cela plus profondément au futur. Nous pouvons imaginer qu'avec les contrôleurs locaux, le robot va apprendre de manière off-ligne comment exécuter ça tache. Tandis qu'avec le cerveau, le robot va apprendre en ligne comment se comporter devant des situations particulières. C'est à dire que l'apprentissage sera fait durant les simulations et non pas durant l'évolution. Nous pouvons aussi doter le cerveau d'un ou plusieurs contrôleurs. Chaque contrôleur permet de faire une tâche selon une situation donnée. Comme on a des robots de morphologies différentes et variées, pourquoi pas ne pas regrouper les morphologies qui se ressemblent en groupes, et chaque groupe de robots va avoir le même cerveau (ou conscience collective).

Pour permettre la coopération entre les modules, des neurones de communication sont nécessaires. Ils vont connecter directement la sortie d'un contrôleur à une entrée d'un autre pour rendre possible l'émergence du protocole de communication au sein du robot. Ils permettent d'échanger des messages entre modules voisins, par

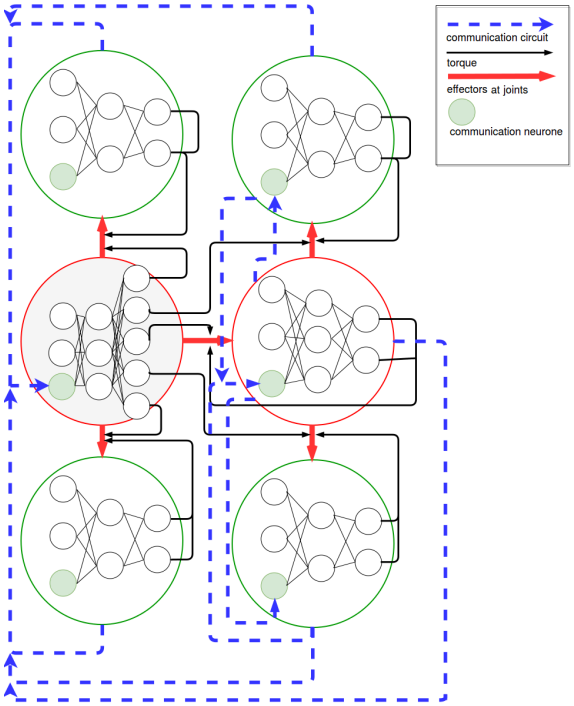


FIGURE 5.3 – Exemple d'un système de contrôle distribué. Il représente le phénotype généré à partir du génotype montré dans la figure 5.2. Le nœud phénotypique gris (module) représente la racine. Il contient le contrôleur global.

exemple, une information traitée obtenue à partir d'un capteur éventuel. Ils sont également utilisés pour envoyer des messages au cerveau du robot. Ce dernier reçoit des informations plutôt générales sur le robot tel que sa masse et le nombre de ses modules. À cet effet, avec notre contrôleur, le comportement global d'un robot se fait par la coopération de tous les mouvements locaux de ses modules. Un exemple d'un système de contrôle distribué est montré dans la figure numéro 5.3.

Les contrôleurs neuronaux MLP générés utilisent la fonction d'activation *Tangente Hyperbolique*. Ils possèdent au plus 3 couches cachées, dont le nombre de neurones dans chacune varie entre 3 et 10. La couche d'entrée contient 15 neurones. Ceux-ci ont en entrée les données issues des capteurs et ceux via communication (voir la figure 5.4). Cependant à la concrétisation du phénotype, le nombre de neurones sera réduit et va varier en fonction du nombre des jointures du module et des capteurs intégrés. La couche de sortie contient 3 neurones, un pour envoyer des messages et les deux autres sont utilisés pour calculer le couple qui sera appliqué à l'articulation. La valeur de résultat est ensuite normalisée pour s'adapter à la plage de couple autorisée.

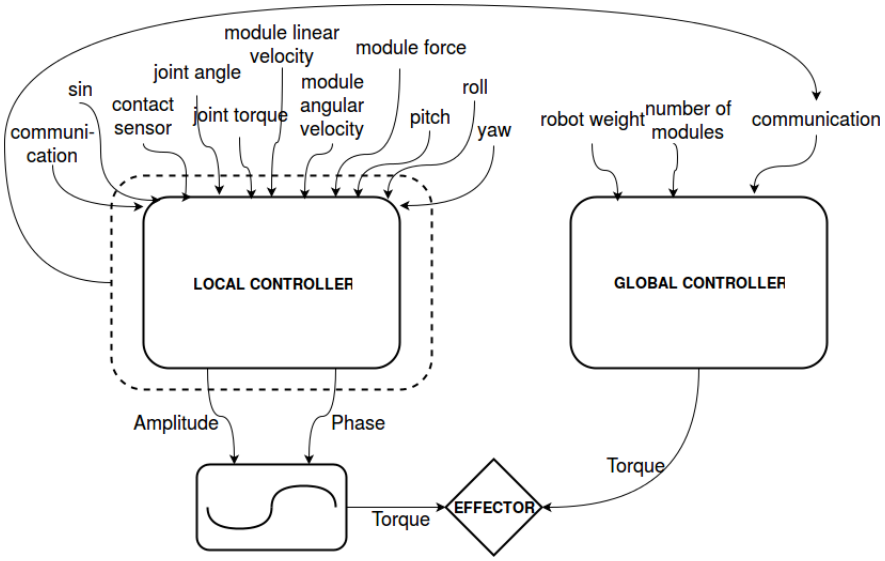


FIGURE 5.4 – Contrôleur local et contrôleur global avec les entrées et les sorties définies

Chaque module utilise un ensemble de capteurs qui l'informent du monde extérieur ainsi que de son état interne (l'état interne du module et de la jointure). Les entrées d'un réseau de neurones sont présentées dans la figure 5.4. Les effecteurs des robots appliquent des couples au niveau des jointures. Le couple appliqué à chaque jointure est une fonction sinusoïdale. Le premier neurone de la couche de sortie représente l'amplitude et le second représente la phase.

5.3 L'évolution jointe de robots

L'évolution des robots virtuels permet d'optimiser et de faire émerger ceux qui sont les plus aptes à exécuter leurs tâches. Dans ce contexte, nous avons fait évoluer les morphologies de nos robots aux côtés de leurs contrôleurs dans le simulateur Gazebo. L'objectif du système d'évolution est donc de faire optimiser la morphologie ainsi que les paramètres des RNA afin d'obtenir une combinaison appropriée. Cela veut dire qu'on va avoir une morphologie de robot, à la fois adéquate à l'environnement et à la tâche donnée, et aussi un contrôleur efficace capable de la mettre en action. Toute fois, définir la morphologie et le contrôleur ensemble, en une seule solution, implique l'exploration et l'exploitation d'un espace de recherche assez complexe. Le système d'évolution va optimiser non seulement des paramètres, mais aussi des structures de données.

Pour le bon fonctionnement d'un AE, il est important de fouiller un espace de solutions valides. Nous avons remarqué lors des évolutions effectuées dans nos expériences, que certains robots se comportaient de manière irréaliste et malgré cela, ils émergeaient et devenaient des élites. Ces robots invalides apparaissent lorsque

l'environnement de simulation n'est pas réaliste, ne répondant pas parfaitement aux lois de la physique newtonienne. Nous discutons profondément ce point dans la section 5.4.3. En fait, la présence de ces cas spéciaux de robots incite l'AE à les favoriser au détriment des autres qui sont plus appropriés. Cela est dû, en grande partie, à la fonction *objectif* qui évalue les robots sans avoir une sémantique de l'objectif réel du robot précisé par l'ingénieur.

Parmi les algorithmes d'évolutions proposés dans la littérature, nous avons choisi d'utiliser les AG et NSGA-II pour évoluer nos robots modulaires. Le principe de base de leurs fonctionnements est présenté dans la section 2.2.1. De manière générale, il s'agit d'un processus qui se répète. Au début, les robots sont créés aléatoirement. Ensuite, à chaque cycle, à partir de la génération actuelle, de nouveaux individus sont créés en utilisant des opérations génétiques. Ces derniers composeront la population de la prochaine génération, et ainsi de suite jusqu'à la satisfaction du critère d'arrêt ou l'émergence des individus souhaités. Les paramètres utilisés pour ces algorithmes sont cités dans le prochain chapitre.

5.3.1 Les opérateurs génétiques

Les opérations génétiques sont effectuées au niveau du génotype du robot virtuel. Tous les paramètres de génotype peuvent être affectés par ces opérations. Concernant la mutation, elle peut affecter la morphologie du robot, son contrôleur, ou les deux à la fois.

Mutation

Selon une probabilité donnée, nous pouvons :

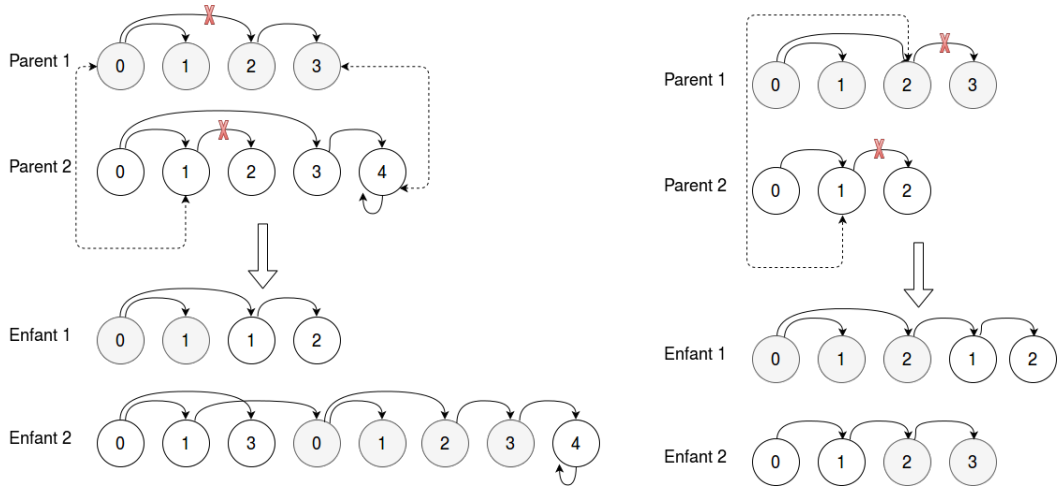
- Ajouter ou enlever un nœud du graphe.
- Ajouter ou enlever une liaison entre deux nœuds (qui peut être récursive ou réflective).
- Changer le nombre d'applications d'une connexion récursive et réflective.
- Modifier le nœud à partir duquel la génération du phénotype (nœud racine) commence.
- Modifier le facteur d'échelle.
- Modifier la méthode de réflexion et/ou de récursivité.
- Au niveau d'un nœud (possibilité de choisir l'occurrence à muter)
  - Changer la dimension du module(largeur, longueur, hauteur).
  - Changer l'axe de rotation d'une seule jointure et/ou sa position par rapport au module.

– Supprimer ou ajouter un capteur (on peut préciser quel module s'il y a de réplications)
– Muter le contrôleur du nœud (une variation au niveau d'un poids du RNA)

Accouplement

Deux nouveaux robots sont créés à partir de l'accouplement de deux autres sélectionnés depuis la population. Nous avons utilisé deux méthodes pour l'accouplement (figure 5.5) ; le croisement et le greffage (*grafting* en anglais) [Sims, 1994b]. Nous choisissons une aléatoirement.

Au niveau de chaque graphe, on élimine une connexion de manière stochastique et on lie les deux graphes par une ou deux connexions selon la méthode suivie. Après la création des enfants au niveau génotype, il va falloir développer les graphes pour générer les phénotypes correspondants.



(a) Croisement (b) Greffage

FIGURE 5.5 – Deux méthodes d'accouplement de deux génotype (*Graphtals*)

5.4 L'environnement de simulation

5.4.1 Gazebo

L'étude comparative que nous avons effectuée préalablement s'est soldée par le choix du simulateur multi-robot Gazebo. Après avoir intégré notre système évolutif dans ce dernier, nous avons trouvé qu'il répondait bien à nos attentes. Il nous a permis de fournir des simulations réalistes de haute qualité, que se soit par rapport à

la visualisation ou par rapport au comportement. De par son architecture modulaire, nous avons pu interagir directement avec le moteur physique, ce qui nous a fait gagner beaucoup de temps et de précision. Sa communauté active sur le web nous a été sans doute très utile. L'un des composants clés de la plate-forme est l'utilisation des capteurs simulés qui produisent un flux de données qui correspond le plus aux données provenant des capteurs physiques réels. Par railleurs, modifier les phénotypes des robots, les traduire et les écrire dans un fichier XML de format SDF était facile. Aussi, modifier la simulation pendant l'exécution était possible grâce aux Plugins performants écrits en C++.

5.4.2 Les capteurs

Gazebo fournit des modèles de capteurs les plus utilisés. Pour exemple, le Global Positioning System (GPS), l'Inertial Measurement Unit (IMU), les lasers et les caméras. Ces capteurs sont parfaits et observent le monde extérieur sans exposer de bruits. Cependant, pour avoir des simulations proches du réel, il est possible d'ajouter explicitement des bruits aux données [3]. Avoir de tels capteurs intégrés dans le simulateur, nous permet de les utiliser sans pour autant en développer. Chaque module pourra ainsi être spécialisé et comporter un capteur donné différent. Par exemple un module doté d'une caméra jouera le rôle d'un œil et le module doté d'un laser jouera le rôle d'une oreille.

Les capteurs se sont des outils qui permettent aux robots d'interagir avec le monde extérieur. En fonction des données perçues, ils vont prendre des décisions et agir selon leurs environnements et leurs états internes. Théoriquement, plus le robot possède de capteurs, plus il est conscient de ce qui se passe autour de lui. Il prendra donc les bonnes décisions. Cependant, sur Gazebo, il est recommandé de ne pas utiliser beaucoup de capteurs sur un seul robot simulé, car cela rend les simulations lentes pouvant être interrompues avec des calculs énormes [4]. De ce fait, nous avons choisi d'utiliser, en premier lieu, seulement des capteurs de contact, puisque l'environnement et les missions ne sont pas complexes. Au fait, Le capteur de contact va pouvoir générer des signaux lorsque le module entre en contact avec une autre surface, notamment le sol et un autre module.

En ce qui concerne les capteurs internes. Nous avons utilisé les capteurs de détection de position, de vitesse et d'accélération des jointures et des modules. Nous avons utilisé en addition à cela, un capteur de détection des forces externes subies. En effet, ces capteurs sont importants et ils sont requis par presque tout comportement

3. Dans nos travaux nous choisissons de ne pas ajouter de bruits pour les raisons citées dans la section 5.4.3

4. Éventuellement, ce problème sera réglé avec les nouvelles versions de Gazebo. Dans nos travaux nous avons utilisé la version 5.1.

dynamique. Dans un environnement simulé, c'est facile de les mettre en œuvre. Au lieu d'utiliser de vrais capteurs simulés tels qu'un accéléromètre, un gyroscope et un GPS, il suffit d'accéder à l'état actuel du robot pour en récupérer les paramètres.

5.4.3 Problèmes rencontrés par rapport à Gazebo

Gazebo présente des avantages intéressants pour la robotique évolutionnaire. Cependant, comme tout autre simulateur, il présente aussi des limitations. Nous avons donc rencontré quelques difficultés pour y intégrer notre système d'évolution. Nous allons les citer dans ce qui suit et aussi présenter les méthodes que nous avons utilisées afin de les surmonter. Cela va être utile pour tous les utilisateurs de Gazebo qui ressentent le besoin de réaliser des évolutions de robots simulés.

Paramètres physiques

Gazebo fournit pour tous objets simulés de nombreux attributs physiques. Cependant, il est nécessaire d'initialiser ces attributs par des valeurs se rapprochant le plus de la réalité. Les attributs que nous trouvons les plus importants à ajuster correctement sont les suivants :

- **Les dimensions du module :** elles vont de 5 *cm* à 10 *cm*. Depuis la littérature, les dimensions des modules des robots modulaires concordent avec ces valeurs.
- **Les dimensions de la jointure :** la longueur est de 2.5 *cm*, la largeur et la hauteur sont chacune à 1 *cm*. Ces dimensions sont choisies en fonction des dimensions des modules.
- **La masse du module :** la masse est calculée à partir du volume du module et de sa masse volumique. Nous avons choisi la masse volumique d'eau. En fait, la masse des modules doit être aussi légère que possible. Ainsi, le module pourra supporter le poids des autres modules connectés à lui.
- **Les frottements :** si elle est appliquée sur une surface d'un module, cette force opposée peut éviter le comportement de glissement du robot et, si appliquée au niveau des jointures, elle évite les mouvements de vibrations irréalistes. Nous avons fixé le frottement de la surface du module à 0.5 et nous avons fixé les frottements au niveau des jointures à 0.2.
- **Le moment d'inertie du module :** le moment d'inertie détermine la difficulté de mettre un objet en rotation. En fait, c'est le paramètre le plus influent sur le réalisme des mouvements. S'il est mal défini, le robot devient instable et réalise des mouvements ne réagissant pas aux lois de la physique, notamment la

gravité. Le moment d'inertie dépend de la masse, de la taille et de la forme du corps, et il est exprimé en une matrice symétrique comme montré ci-dessous :

$$Matrice\ du\ moment\ d'inertie \begin{cases} I_h = \frac{1}{3}m(w^2 + d^2) \\ I_w = \frac{1}{3}m(d^2 + h^2) \\ I_d = \frac{1}{3}m(w^2 + h^2) \end{cases} \tag{5.1}$$

où m représente la masse du module, h, w et d représentent respectivement la hauteur, la largeur et la profondeur du module.

- **L'amortissement des jointures :** en fonction de la vitesse de la jointure, l'amortissement permet de dissiper l'énergie. Cela peut éviter les mouvements de rebondissement. Avec les expérimentations, nous l'avons mis à 0.02.
- **La vitesse des jointures :** la vitesse maximale autorisée pour une articulation est de 5 rad/s. Cette valeur est déterminée par expérimentations.
- **Le couple des jointures :** le couple ou le moment d'une force (*torque* en anglais) représente la force requise permettant à une jointure d'effectuer une rotation et soulever l'ensemble des modules qui lui sont liés sans pour autant atteindre la rupture de la jointure. Après plusieurs tests, nous avons fixé une limite de 1.75 Nm. Nous en parlerons plus en détail dans la section 5.4.3.

Bien déterminés, ces attributs permettent d'éviter les comportements irréalistes de robot. Nous présentons trois cas de simulation irréaliste que l'on peut rencontrer. Cas A : il n'y a pas assez de force appliquée aux jointures, le robot reste immobile. Cas B : la force est trop grande, le robot rebondit aléatoirement en parcourant une distance qui peut être démesurée. Cas C : la force est suffisante, mais l'un des attributs physiques manque de précision, le robot se déplace, mais en glissant sans que les jointures fassent de rotations. Avec une fonction *fitness* basée sur la distance parcourue, l'évolution favorise le comportement aberrant des individus du cas B et C au détriment de ceux des cas plus réalistes.

Donc, plus il y a de restriction et de limitation au niveau du réglage des paramètres physiques, plus nous diminuons l'émergence de cas particuliers qui peuvent fausser l'évolution. Car les valeurs limites rendent le contrôle des jointures impossible devenant difficile pour les moteurs physiques de calculer avec précision les forces et les positions à appliquer. Cependant, nous avons remarqué que même après avoir bien ajusté ces paramètres (frottement et force appliquée), le comportement de glissement de cas C, que nous avons considéré irréaliste, apparaît sur certaines évolutions. Cela est dû au fait qu'un module puisse pousser un autre module pendant une longue

période du temps. En d'autres termes, il y a une application continue de force toujours dans le même sens au niveau d'une jointure. Donc, afin d'éviter ce genre de comportement, nous arrêtons d'appliquer la force dans le cas où les forces réactives des points de contact sont excessives.

Couple

Les moteurs contrôlant les jointures doivent être suffisamment puissants pour pouvoir supporter et manipuler un ou deux autres modules sous l'effet de la gravité. Toutes fois, il ne faut pas dépasser une certaine capacité qui va induire le rampement des jointures. Dans un environnement de simulation, les robots rebondissent et deviennent totalement incontrôlables. Dans notre cas il est difficile de trouver pour chaque jointure la valeur du couple maximum à appliquer, car les morphologies sont variables en taille et en masse.

Cette valeur de couple se traduit par Newton Mètre. Il s'agit de la force appliquée à la jointure multipliée par la longueur séparant le point de rotation au centre de masse de l'objet à soulever (appelé *levier*). Un exemple est illustré dans la figure numéro 5.6. Il faut cependant prendre le pire des cas, là où cette longueur est la plus grande. Dans ce cas, c'est lorsque les modules sont tous tenus horizontaux. Cependant, si le robot possède une structure cinématique complexe, il devient difficile d'estimer cette force. Les chercheurs mécaniciens reviennent souvent aux équations de Lagrange, qui pour chaque actionneur ou jointure, on calcule la force maximale requise pour fournir un mouvement simultané.

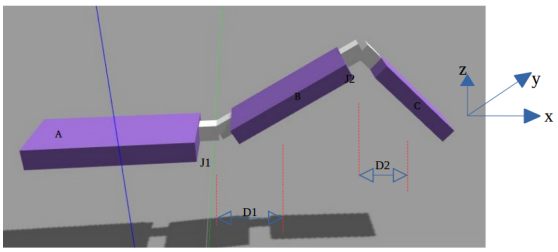


FIGURE 5.6 – Le calcul d'un couple d'une jointure dépend de la taille et de la masse des modules à soulever. La valeur du couple de la jointure J1 est présentée dans la fonction 5.2 ci-dessous

$$\begin{aligned} Moment_{J1} = {} & Distance_{J1}/2 + Distance_B + Distance_{J2} + Distance_C/2 * Poids_C \\ & + Distance_{J1}/2 + Distance_B + Distance_{J2}/2 * Poids_{J2} \\ & + Distance_{J1}/2 + Distance_B/2 * Poids_B \\ & + Distance_{J1}/4 * Poids_{J1} \end{aligned} \tag{5.2}$$

Interpénétration

À la création du phénotype du robot, et avant qu'il soit plongé dans l'environnement pour évaluation, il se peut qu'il y ait des modules qui vont s'interpénétrer. C'est un phénotype inapproprié. Dans ce cas, on peut tout simplement retirer la créature de la population en lui donnant une valeur de *fitness* nulle, et d'en refaire une nouvelle jusqu'à en avoir une morphologie appropriée [Sims, 1994b, Krcah, 2007]. Dans notre travail, nous avons choisi de faire un prétraitement pour vérifier la validité des phénotypes et de les ajuster. Ça nous évitera de perdre des créatures potentiellement performantes et efficaces malgré l'interpénétration.

Pour ce faire, on tente de changer la position du module. On le décale suffisamment pour le faire sortir de l'autre module avec lequel il est en interpénétration. Si ça ne marche pas, on lui change d'orientation et si ça ne marche toujours pas, on lui change carrément de face de connexion. On refait cela trois fois, si l'interpénétration persiste on élimine le robot et on génère un autre nouveau. Nous présentons le pseudo-code 1 résumant les étapes de prétraitement.

Algorithm 1 vérifier interpénétration (Module m)

```
m.visited = true
for enfants de module m do
  for 3 tentations do
    if interpénétration entre m et tous les modules visités then
      avec un certain ordre essayer l'une des actions suivantes :
      Pousser les modules dans l'une des directions
      changer la face de connexion
      changer la rotation en faisant une rotation dans un axe donné
    else
      break
    end if
  end for
  vérifier interpénétration (enfant de m)
end for
```

Puisque cette procédure est une séquence d'actions où il n'y a pas d'aléatoire, le même génotype produira toujours le même phénotype. L'algorithme, comme il peut être appliqué à de nouveaux phénotypes de robots générés aléatoirement, il pourrait également être appliqué à des robots issus d'opérations génétiques. Lorsque le robot ne peut pas être récupéré, un nouveau génotype est créé. En faisant les expérimentations, 3.93% des morphologies inadaptées ont été générées. Le taux de récupération était de 71.56%.

Jointures

La représentation des jointures dans le simulateur Gazebo est invisible, elles ne sont que des axes de rotation qui lient deux modules. Pour permettre une certaine liberté de mouvement, Gazebo permet les interpénétrations entre les modules voisins. Cela signifie qu'il n y aura pas de gestion de collisions entre les modules. Cela par conséquent fait perdre en partie le réalisme de la simulation, surtout si nous envisageons de reproduire les robots en monde réel. C'est ce qui nous a amenés à ajouter des jointures physiques sous forme de modules aux robots. L'interpénétration se fait uniquement entre les deux modules composant la jointure.

Parallélisme

L'étape d'évaluation des robots est cruciale, car elle permet de noter la performance de chacun d'entre eux. Elle est l'étape qui prend le plus de temps d'exécution du processus d'évolution. Intuitivement, nous pouvons, lors de l'évaluation des robots d'une génération, attribuer à chacun une position de départ différente dans l'environnement de simulation. Néanmoins, Gazebo ne permet pas ça, parce que plus le nombre de robots simulés est grand plus le temps de calcul augmente et la durée d'une simulation devient très lente[5]. Du coup, nous avons proposé d'évaluer chaque robot seul dans son environnement et lancer sa simulation sur un serveur séparé avec un port différent. Nous avons pu évaluer en conséquence un ensemble de robots simultanément de manière parallèle avec l'API OpenMP. Le nombre de robots simulés en même temps est déterminé en fonction du nombre de cœurs du processeur. Afin de comparer les temps d'exécutions des simulations, nous avons effectué des tests dont nous exposons les résultats dans le tableau 5.1. Nous avons utilisé pour cet effet la version 4.1 de Gazebo et une machine qui possède les caractéristiques suivantes :

- Processeur : Intel® Core™ i7 CPU 930 @ 2.80GHz × 8
- Carte graphique : GeForce GTX 780/PCIe/SSE2
- RAM : 12 GO

Le test, s'agit de comparer les durées des simulations d'une génération de 100 robots. Nous considérons deux cas : 1) simuler en un seul *run* tous les robots simultanément dans le même environnement partagé, 2) simuler les robots séparément en parallèle, chacun dans son environnement. Nous notons que l'évaluation de chaque robot dure 7 s de temps de simulation et la fréquence des capteurs est de 30 Hz. Nous examinerons également le rapport entre la complexité (nombre de jointures et

5. L'une des raisons du choix du Gazebo est sa plate-forme multi-robot, mais ici on parle d'une centaine de robots

Tableau 5.1 – Temps de simulation de 100 robots

Simulation		Capteurs		Robot à 4 DOF		Robot à 12 DOF	
Simultané*	Parallèle**	Contact	Caméra	Temps réel	Temps CPU	Temps réel	Temps CPU
x		4	0	3m17.791s	3m18.732s	4m46.432s	4m57.962s
x		4	2	5m20.620s	10m59.148s	6m58.702s	14m20.913s
	x	4	0	3m16.015s	7m11.820s	3m16.388s	8m25.370s
	x	4	2	4m29.234s	20m29.095s	4m42.012s	22m34.314s

* Les robots sont simulés tous à la fois dans un seul environnent.
** Les robots sont simulés en parallèle (8 à la fois), chacun dans son environnement.

de capteurs intégrés) des robots est le temps de calcul nécessaire pour réaliser leurs simulations.

Nous remarquons que si la morphologie d'un robot est simple, c'est-à-dire avec un nombre réduit de capteurs et de jointures, le parallélisme n'apporte pas un bénéfice en temps de simulation. Tandis que, plus la morphologie est complexe, plus le parallélisme a un impact sur l'accélération du temps de calcul. Comme nous pouvons le voir dans le tableau, la simulation de 100 robots simultanément prend $6m58.702s$, alors que la simulation de ces mêmes robots séparément en parallèle prend $4m42.012s$. Théoriquement, si on double le nombre de cœurs, on devrait réduire plus le temps d'exécution.

Dans le cadre de nos travaux, réduire le temps d'apprentissage n'est pas une tâche dans laquelle une grande importance est accordée. En effet, nous faisons des évolutions or ligne, qui se font avant de déployer les robots dans les terrains pour effectuer réellement leurs missions. Toutefois, avoir un temps de calcul réduit reste signifiant et important, car cela nous évite d'attendre plusieurs heures, voir plusieurs jours de calcul pour visualiser les résultats, qui ne sont pas satisfaisants dans certains cas. Ainsi, nous pouvons multiplier le nombre d'expériences et gagner du temps.

Instabilités et perturbations

Nous avons remarqué que Gazebo est sensible aux perturbations et aux variations. Ces dernières, introduites volontairement dans le but d'imiter le réel et de concevoir des contrôleurs robustes, empêchent le processus d'évolution de fonctionner normalement. Par ailleurs, les perturbations peuvent aussi provenir du simulateur lui-même, notamment, le non-déterminisme de l'interface de communication.

La communication via des messages simplifie l'interaction avec le serveur. Cependant, elle implique beaucoup de problèmes. En plus du fait de présenter des contraintes et de ralentir la simulation, elle produit des flux de données différents à chaque simulation. Bien évidemment, nous parlons de la simulation du même robot avec les mêmes conditions environnementales et le même contrôleur. Ceci est du au fait qu'il y a un petit délai variable entre l'envoi du message et la réaction de Gazebo. En outre, même sans utiliser de messages, changer la position initiale du robot ou le temps marquant le début de simulation semble affecter les capteurs, et donc produire des données différentes.

Par conséquent, le même robot va se comporter différemment à chaque simulation, ainsi le robot élite d'une génération pourrait dégrader en efficacité lors de la génération suivante [6]. Il est donc impératif pour nous de garder le même comportement du robot. Pour rendre cela possible, nous avons utilisé l'API de Gazebo pour accéder directement au moteur physique. Nous avons aussi dû arrêter et reprendre le simulateur Gazebo à chaque génération. Avec Gazebo, démarrer de nouvelles simulations ne prend presque pas de temps. Cela est important vu le nombre exorbitant d'évaluations des fonctions *fitness*

5.5 Conclusion

Nous avons présenté dans ce chapitre le modèle de génération de nos robots modulaires. Nous avons discuté par la suite l'évolution de ces robots avec le simulateur multi-robot Gazebo. On constate que l'élaboration d'un algorithme évolutionnaire efficace est une tâche difficile. En effet, il s'agit d'un processus stochastique sensible aux paramètres de l'algorithme et également aux paramètres de simulation. La présence de perturbations au niveau de la simulation engendre l'émergence de solutions invalides, qui vont empêcher l'évolution de fonctionner normalement. Nous avons donc ajusté ces paramètres afin qu'ils soient le plus appropriés possible et proches au monde réel. Dans le chapitre suivant, nous allons réaliser des expériences avec nos robots et les faire évoluer pour réaliser des tâches de locomotion.

6. Cela cause aussi un problème pour les évolutions en ligne des robots.

Chapitre 6

Expérimentations et résultats

6.1 Introduction

Dans ce chapitre, nous allons exposer les expériences que nous avons réalisées ainsi que les résultats obtenus. Les expériences consistent à faire évoluer les robots modulaires présentés dans le chapitre précédent à l'effet de leur permettre d'apprendre à exécuter différentes tâches. Dans un premier temps, nous avons voulu étudier la tâche de locomotion, que nous considérons comme étant une tâche basique, mais très importante. Nous avons examiné le rapport entre les tailles des robots (nombre de modules) et leurs capacités à parcourir de longues distances. En outre, nous avons étudié la résilience. Nous accordons une importance particulière au sens de déplacement des robots après endommagement. Le robot défectueux doit non seulement poursuivre son déplacement, mais aussi garder la même direction afin d'atteindre son objectif. Le fait de réaliser ces expériences sur une plate-forme de simulation crédible avec une modélisation de robots assez réalistes, nous avons obtenu des comportements proches du réel.

6.2 Évolution jointe des contrôleurs et des morphologies pour des robots modulaires réalistes

6.2.1 Description

Les robots sont conçus pour pouvoir accomplir différentes tâches, plus ou moins complexes, tels que la manipulation d'objets, l'évitement d'obstacles et la poursuite. Cependant, dans ce travail, nous nous sommes intéressés à la locomotion. Cette dernière est l'une des tâches fondamentales de la robotique modulaire. Elle permet à un robot de se déplacer d'un point à un autre, peu importe sa morphologie ou

son environnement. En théorie, la locomotion est la base de toutes missions se déroulant dans des environnements isolés loin des ingénieurs spécialistes. Par ailleurs, les travaux de référence présentés dans la partie état de l'art, peu importe leurs objectifs, avaient tous généré des créatures et des robots qui pouvaient se déplacer. Nous pouvons citer à titre d'exemple, les travaux de Karl Sims [Sims, 1994b] et ceux de Nicolas Lassabe [Lassabe et al., 2007], qui ont récompensé les créatures qui parcourrait les plus grandes distances. Par conséquent, elles développaient des moyens créatifs de locomotion, entre autres, la nage, le saut et le rampement.

Les robots doivent se déplacer de manière réaliste dans les environnements de simulation. Nous distinguons deux aspects de réalisme ; l'aspect du comportement et l'aspect de la simulation. Premièrement, au niveau comportemental, les robots doivent effectuer des mouvements et des locomotions efficaces tels ceux des animaux. Par exemple, pouvoir soulever un membre articulé afin de réaliser un grand pas, ainsi faire de longues distances en un minimum de temps. Dans ce cas, la modularité est très importante, car elle permet une coopération et une synergie entre les différents modules. En outre, la représentation génétique des robots joue également un rôle crucial pour permettre de tels comportements. Le deuxième aspect concerne le réalisme des simulations et du rendu visuel. Les robots doivent répondre parfaitement aux lois de la physique newtonienne. En fait, cela a un lien direct avec la plate-forme de simulation utilisée. De plus, la conception de ces robots doit être proche de celle des robots physiques.

Cependant, la plupart des créatures proposées dans l'état de l'art ne sont pas autant réalistes que ça. Nous pouvons facilement distinguer des interpénétrations entre les modules lors des simulations. En outre, les simulateurs utilisés, par rapport à Gazebo, manquent de crédibilité. Cela engendre des différences comportementales entre la simulation et la réalité. Dans ce cas, les créatures ou les robots virtuels ne sont pas constructibles. Néanmoins, il convient de mentionner que l'objectif principal de ces travaux est l'étude de génération automatique de robots complexes et variés. Ils obtenaient alors des comportements intéressants au détriment du réalisme des simulations et de la conception mécanique.

Nous aspirons dans ce travail à améliorer les deux aspects du réalisme des robots. Pour cela, nous privilégions de suivre les travaux faits dans ce domaine et adopter les évolutions conjointes des morphologies et des comportements. Nous rappelons que la morphologie est le facteur majeur qui décide la qualité du robot. Elle ouvre la voie à des comportements efficaces dans les situations les plus complexes. Ainsi, nous pourrons générer des morphologies réalistes, qui vont s'associer avec des contrôleurs permettant de longues distances, en réalisant des suites de mouvements efficaces.

6.2.2 Objectifs

Dans le cadre de cette étude, le but est de simuler des robots modulaires réalistes capables de parcourir de longues distances. Dans ce sens, il est nécessaire d'impliquer la morphologie, car c'est elle qui déterminera les comportements permis et donc l'entreprise de locomotions efficaces. L'évolution va favoriser et faire émerger les robots qui agissent bien et qui répondent à la fonction *objectif*. Nous allons étudier l'influence de la taille du robot par rapport à sa capacité de se mouvoir. Par taille, nous désignons le nombre de modules et d'articulations d'un robot et non sa grandeur.

6.2.3 La fonction d'évaluation

Le système d'évolution va récompenser les créatures qui semblent répondre le mieux à la fonction *objectif*. Si la sélection des robots est basée sur cette dernière, il est nécessaire de bien la définir pour obtenir les résultats que nous souhaitons. Nous choisissons donc de calculer la distance euclidienne parcourue par le robot afin de déterminer sa capacité à effectuer un déplacement. Les deux positions initiale et finale sont les positions isobarycentres du robot. Les robots sont évalués dans l'environnement pendant 30 s de temps de simulation. Le point initial est pris après la stabilisation du robot, qui est de 3 s.

Définir un temps de stabilisation est important. S'il est mal défini (trop petit ou nul), la fonction *objectif* risque de prendre un robot comme efficace, dont le déplacement effectué est assez important, alors qu'il n'a fait aucun pas ou avancement réel. En d'autres termes, au début de la simulation on sauvegarde la position initiale. Juste après, le robot tombe dans un endroit loin de sa position initiale. Cela peut être causé par exemple par sa morphologie longue tenue verticalement. Ensuite, il reste immobile sur place. On voit ici qu'il y a clairement une grande différence entre le point initial et final. Cependant, le robot est inefficace. En concurrence avec les autres robots, qui eux font des vrais déplacements, le robot inefficace peut éventuellement être classé meilleur.

6.2.4 Paramètres de l'étude

Afin de tester et évaluer l'efficacité de notre système, nous avons mené une série d'expériences. Nous présentons tous les paramètres utilisés pour l'évolution dans le tableau suivant :

Tableau 6.1 – Les paramètres définis pour le système d'évolution

Paramètre	**Valeur**
Taille de la population	50
Nombre de génération	100
Taux de croisement	35%
Taux de mutation	75%
Élitisme	10
Méthode de sélection	Sélection par tournoi avec 7 concurrents

6.2.5 Résultats expérimentaux

Expérience 1

Dans la première série d'expériences, nous avons choisi de fixer un nombre maximal de modules à ne pas dépasser. Nous avons pris 3 cas. Des robots avec un nombre maximal de 5, 10 et 15 modules. Nous comparerons ainsi les distances parcourues pour chacun des cas. L'expérience est répétée 20 fois pour chaque taille. Donc nous aurons 60 expériences en tout.

Le graphe de la figure 6.1 montre qu'avec notre système, les robots ayant un plus grand nombre de modules parcourent plus de distance. Les petits robots à 5 modules au maximum parcourent en moyenne 0.55 m, tandis que ceux avec 10 et 15 modules au maximum peuvent parcourir respectivement 1.7 m et 2.18 m en 30 s.

Toutefois, nous avons remarqué que le nombre de modules des robots émergés pour chaque expérience est égal au nombre de modules maximum autorisé. C'est-à-dire qu'on a obtenu uniquement des robots à 5, 10 et 15 modules, durant ces 60 expériences. Cela est dû au fait que les grands robots parcourent toujours les grandes distances, d'où leurs émergences. En effet, se sont les paramètres utilisés pour déterminer la morphologie et ses capacités à réaliser des mouvements qui ont permis cela. Nous parlons de la fréquence d'application de la récursivité lors de la génération du génotype, et de la valeur du couple permise pour être appliquée.

Une récursivité qui peut aller jusqu'à 10 applications, sans doute, engendre des robots de forme serpent. Cette forme allant en longueur, pouvant atteindre plusieurs dizaines de centimètres, permet de réaliser des rampements et des sauts, et donc parcourir plus de distance par rapport aux autres morphologies. En conséquence, les petits robots ne vont pas survivre malgré le fait qu'ils fonctionnent bien et parcourent des distances considérables.

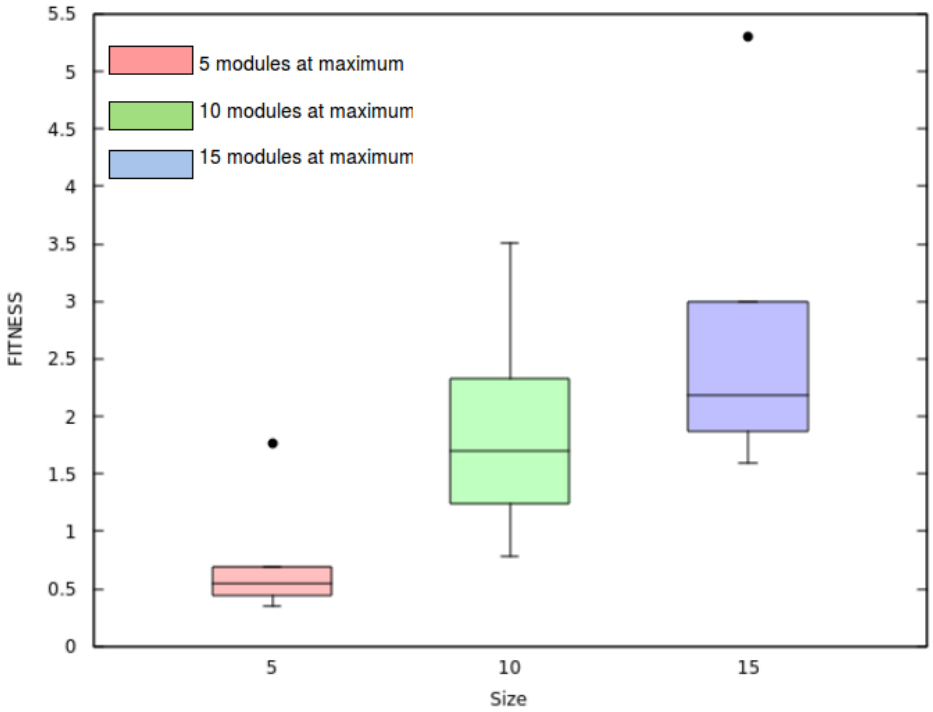


FIGURE 6.1 – L'influence de la taille du robot sur la distance parcourue. Avec ces paramètres d'évolution définis (un couple maximal autorisé de 1.75 Nm et une récursivité appliquée jusqu'à 10 fois), les grands robots fonctionnent mieux. Les boîtes à moustaches montrent la médiane, l'écart interquartile et les valeurs min et max.

Il faut noter que le couple que nous avons défini à 1.75 Nm était suffisamment élevé pour faire bouger ces grands robots. Toutes fois, théoriquement, les grands robots ne sont pas nécessairement les plus adaptés aux déplacements longues distances. Les petits robots légers peuvent aller vite et ainsi parcourir plus de distance. Mais on remarque que ce n'est pas le cas avec cette expérience : les plus petits des robots ont parcouru moins de distance en les comparant aux plus grands, même si le couple requis était suffisant. Ils étaient soient lourds ou de simples morphologies, et d'une certaine manière, ça ne permettait pas la réalisation de mouvements efficaces.

Expérience 2

Dans cette expérience, nous avons voulu donner l'occasion aux petits robots d'évoluer de manière à ce que les grands le puissent aussi. Nous avons empêché alors l'apparition des formes semblables aux robots serpents. Premièrement, nous avons réduit la fréquence d'applications de la récursivité à 2 fois. De cette façon, les morphologies des grands robots seront plus variées (pas nécessairement des serpents). Deuxièmement, nous avons limité l'effort appliqué au niveau des articulations. Le couple maximum était alors de 0.17 Nm. Le nombre maximum de modules autorisés

dans un robot était de 15. Nous avons répété l'expérience 80 fois. Avec chaque expérience, le nombre de modules du robot émergé est entre 1 et 15. Dans le tableau ci-dessous, nous montrons le nombre de robots évolués pour chaque taille de robot.

Tableau 6.2 – La taille des robots évolués

Taille du robot	4	5	6	7	8	9	10	11	12	13	14	15
Nombre de robots évolués	1	4	7	5	13	8	2	8	15	2	5	10

Nous remarquons à travers le graphe de la figure 6.2, que les robots qui ont un nombre de modules allant de 6 à 10 sont généralement meilleurs pour les déplacements de longs trajets. Les robots à 10 modules et plus étaient en moyenne moins efficaces. Cela peut être causé par la limitation du couple (0.17 Nm). Les moteurs n'étaient pas assez puissants pour faire bouger les articulations. Cependant, malgré la limitation du couple, certaines morphologies permettaient des mouvements efficaces. Les robots parcourraient alors plus de distance. Par exemple, un robot à 12 modules avait parcouru plus de 3 m. En ce qui concerne les petits robots à 4 et 5 modules, ils montraient des performances acceptables.

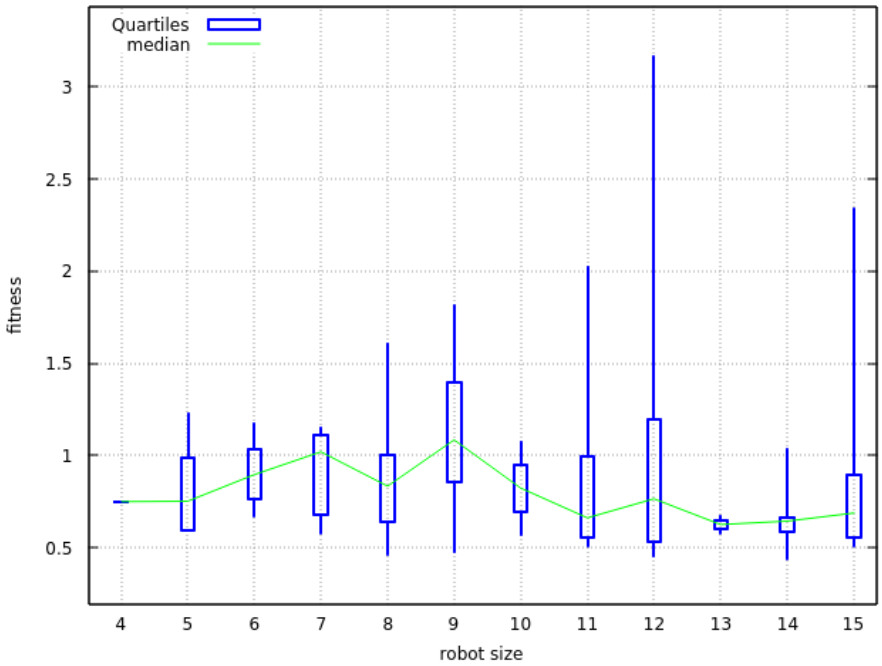


FIGURE 6.2 – Influence de la taille du robot sur la distance parcourue. Les paramètres d'évolution sont (le couple maximum autorisé est de 0.17 Nm, jusqu'à 2 applications de récursivité). Les boîtes à moustaches montrent la médiane, l'écart interquartile et les valeurs min et max.

6.2.6 Discussion

Les résultats expérimentaux montrent qu'en effet les tailles des robots impactent leurs capacités à parcourir de longues distances. Plus il y a de modules et d'articulations, plus des morphologies variées et intéressantes peuvent être générées. Avec ces dernières, on a de meilleures possibilités d'obtenir des comportements efficaces permettant des locomotions sur grandes distances. Les mouvements vont avoir des amplitudes plus élevées, ce qui leur permet de faire de grands pas vers l'avant en un seul mouvement. En fait, nous croyons que les grands robots, ayant plus de DOF, sont plus appropriés pour cet effet. Il suffit de fournir le couple nécessaire pour chaque jointure.

Les images de la figure ci-dessous illustrent quelques robots modulaires générés par notre système. Ils ont la particularité de se mouvoir de façon réaliste [1]. En effet, les jointures sont contrôlées de manière à exploiter la structure morphologique afin d'en tirer avantage dans la réalisation de la tâche. Nous avons donc obtenu des structures variées avec des comportements créatifs. Les robots peuvent par exemple serpenter, ramper, sauter ou même rouler autour d'eux même. Les figures 6.3, 6.4 et 6.6 montrent des captures d'écrans prises en plusieurs laps de temps de quelques mouvements réalisés par nos robots. En fait, la locomotion qui se répète dans la majorité des expériences est le rampement. Les deux membres d'un robot vont aller pousser tout le corps vers l'avant en s'appuyant au sol. Par ailleurs, nous avons observé l'émergence de quelques robots possédant des structures symétriques. Cependant, elles n'étaient pas favorables pour les déplacements de longues distances.

1. Vidéo : https://www.youtube.com/watch?v=FUFv75Nf0gg

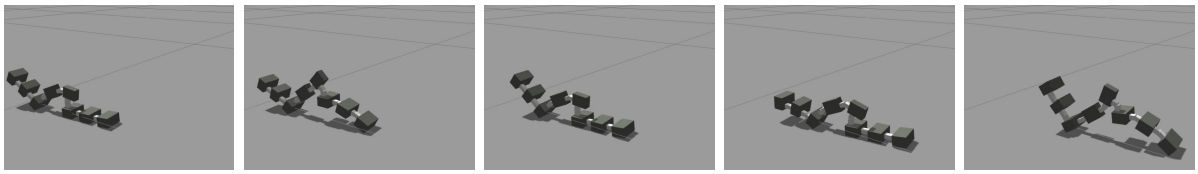


FIGURE 6.3 – Un robot qui exécute un mouvement de chenille

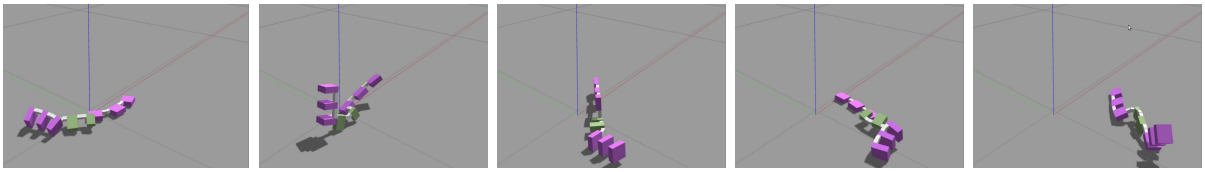


FIGURE 6.4 – Un robot qui se déplace par roulement

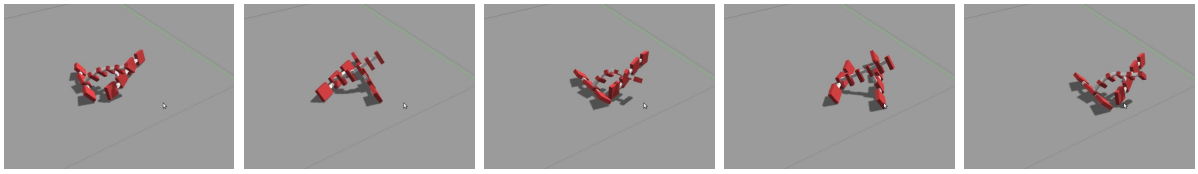


FIGURE 6.5 – Un robot à quatre pattes en marche (de droite à gauche)

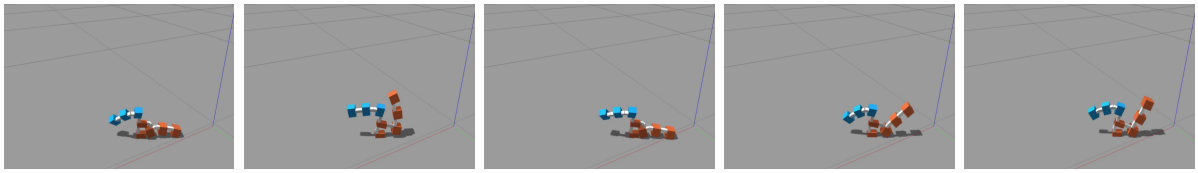
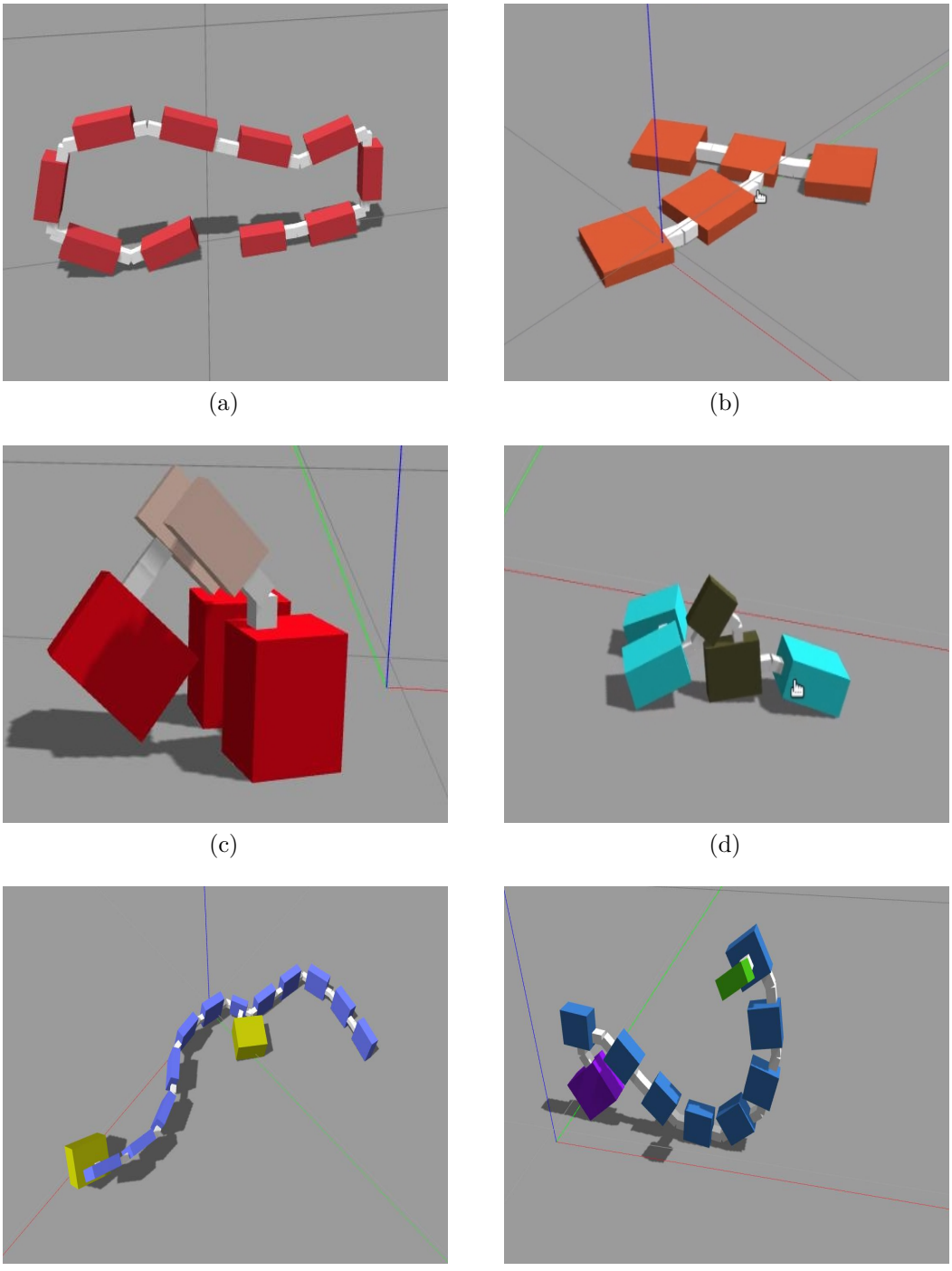


FIGURE 6.6 – Un robot réalisant un saut (de droite à gauche)



(a) (b) (c) (d) (e) (f)

FIGURE 6.7 – Quelques robots modulaires évolués

6.3 Une approche évolutionnaire pour la génération de robots modulaires résilients

6.3.1 Description

Construire des robots résilients capables de se remettre des dommages est une question centrale et difficile [Cully et al., 2015, Bongard et al., 2006]. Faire de tels robots les rendrait, sans assistance humaine, capables de maintenir leurs capacités de poursuivre leurs missions quand une dégradation matérielle se produit. Les chercheurs ont proposé donc différentes approches et techniques (regarder la section numéro 2.4). Celles-ci étaient testées sur des machines physiques avec des morphologies fixes et prédéfinies, notamment des robots serpents [Mahdavi and Bentley, 2006] et des robots multipattes [Bongard et al., 2006]. En fait, ne pas avoir des morphologies qui évoluent, minimise les chances de production de robots adaptables et créatifs dans leurs comportements. Par ailleurs, réaliser des tests sur le robot physique, dont le but de trouver un nouveau contrôleur, risque de l'endommager davantage.

Nous remarquons que ces robots ont des articulations critiques et certains de leurs membres jouent un rôle crucial déterminant l'efficacité des mouvements effectués. Se sont les parties les plus actives, par exemple l'articulation genou d'une patte quelconque. Lorsqu'une défaillance affecte l'une de ces parties, la capacité du robot à se déplacer est fortement compromise. Afin de dépasser cette limite, nous visons à faire évoluer des morphologies de robots sans ces parties, et plutôt, celles qui vont collaborer pour compenser les échecs possibles. Cela veut dire que nous allons obtenir des morphologies dont le comportement est moins affecté lors des dommages. En réduisant l'impacte de ces derniers, les morphologies évoluées vont permettre en quelque sorte la résilience. Pour pouvoir générer de tels robots, il est d'abord important de faire évoluer conjointement les morphologies et les contrôleurs. En suite, empêcher l'émergence des parties (jointures et modules) critiques. Pour cela, nous proposons de faire subir les robots des dommages au moment de l'évaluation. Ceux qui possèdent des parties critiques vont échouer et disparaître au cours des générations. Par conséquent, les robots vont apprendre à faire face et à s'adapter.

Dans le cadre de cette étude, nous allons accorder une importance particulière au sens de déplacement des robots. Après une défaillance technique, le robot doit continuer à avancer toujours dans la même direction, et ce pour arriver à destination. Si à un moment donné, le robot change de direction après un certain dommage, ça va produire des résultats indésirables dans certaines situations et missions. Prenons par exemple le cas des robots de surveillance, qui doivent maintenir des trajectoires spécifiques afin d'assurer une protection optimale. Donc, à notre avis, l'efficacité et

la vitesse de la locomotion sont d'une importance moindre que le maintien d'une direction donnée.

Les robots peuvent subir plusieurs types de dommages tels que des membres (p. ex. pattes) manquants ou cassés, ou des moteurs non fonctionnels. Afin de réduire la complexité du processus de récupération dans cette étude, nous nous focalisons sur les pannes moteur. Ces derniers ne seront plus alimentés. Les jointures qu'ils contrôlent sont désormais inactives. En outre, la communication entre les deux modules voisins sera interrompue. Nous nous limitons également lors des évaluations à une seule panne possible.

6.3.2 Objectifs

Dans le cadre de cette étude, nous aspirons à concevoir des robots modulaires qui sont dotés de capacités de résilience. Par le biais de l'évolution conjointe des morphologies et des contrôleurs, nous cherchons à obtenir des robots qui puissent se remettre des dégâts sans qu'il soit nécessaire de refaire une étape d'apprentissage supplémentaire (en ligne ou or ligne). En d'autres termes, nous cherchons à faire évoluer des robots dont l'efficacité locomotrice n'est pas déterminée par des articulations et des membres spécifiques. Par conséquent, la défaillance d'un moteur qui rend un membre incontrôlé et inactif n'affectera pas substantiellement l'efficacité de la locomotion. Afin de rendre cela possible, il faut générer des morphologies et des contrôleurs qui ensemble concordent parfaitement et en harmonie pour répondre à la fonction *objectif*.

6.3.3 La fonction d'évaluation

Les robots sont évalués avec et sans joints défectueux. Chaque évaluation prend 10 s de temps de simulation. Le point initial est pris après la stabilisation du robot (après 2s). Ainsi, le temps global d'évaluation d'un robot dépend du nombre de ses articulations.

Pour faire évoluer des robots sur lesquels les dommages n'affectent pas la locomotion, notamment la direction, nous optimisons deux objectifs :

(1) maximiser la distance parcourue du robot intact sur n'importe quelle direction (équation 6.1).

(2) maximiser la distance parcourue du même robot, mais avec un joint défectueux, dans le sens donné par l'objectif (1). Le deuxième objectif peut être réécrit comme la minimisation de la distance entre le point atteint par le robot endommagé et un certain point cible mis dans la direction donnée par objectif (1). Parmi toutes les

évaluations des joints défectueux, nous choisissons de prendre la plus grande distance restante, telle que présentée dans l'équation 6.2.

$$dis = \sqrt{(f_x - i_x)^2 + (f_y - i_y)^2} \tag{6.1}$$

$$rem = \max_{\text{j : indice du joint défectueux}} (\sqrt{(o_x - f_x^j)^2 + (o_y - f_y^j)^2}) \tag{6.2}$$

où i et f sont respectivement la position initiale et la position finale du robot à la fois intact et endommagé. o est la position de la cible qui doit être atteinte par le robot avec un joint défectueux. Elle se trouve à un mètre depuis la position initiale du robot intact. Elle est donc différente pour chaque robot. La position de la cible est déterminée en fonction de la direction de locomotion du robot non endommagé, tel que présenté dans l'équation 6.3. Ainsi, nous assurons que la locomotion du robot est toujours vers l'avant même s'il y a des échecs.

Nous rappelons que le but du robot est de maintenir la même direction de locomotion, et non atteindre la cible. La distance restante est une mesure qui nous permet de sélectionner les robots qui respectent la direction de locomotion et continuent d'aller de l'avant.

$$o = \begin{cases} o_x = i_x + \frac{1}{dis}(f_x - i_x) \\ \\ o_y = i_y + \frac{1}{dis}(f_y - i_y) \end{cases} \tag{6.3}$$

6.3.4 Paramètres de l'étude

Comme mentionné dans la section précédente, le comportement souhaité des robots nécessite de satisfaire deux objectifs à la fois. Cela exige l'utilisation des AEMO. Ici, on ne cherche pas à optimiser une seule solution, mais plutôt, on produit un compromis de solutions à partir desquelles un décideur peut en sélectionner une. Dans ce contexte, nous utilisons NSGA-II, qui a été appliqué avec succès pour résoudre plusieurs problèmes multiobjectifs.

Tableau 6.3 – Les paramètres définis pour le système d'évolution

Paramètre	**Valeur**
Taille de la population	80
Nombre de génération	400
Taux de croisement	35%
Taux de mutation	75%
Élitisme	10
Méthode de sélection	Sélection par tournoi avec 7 concurrents

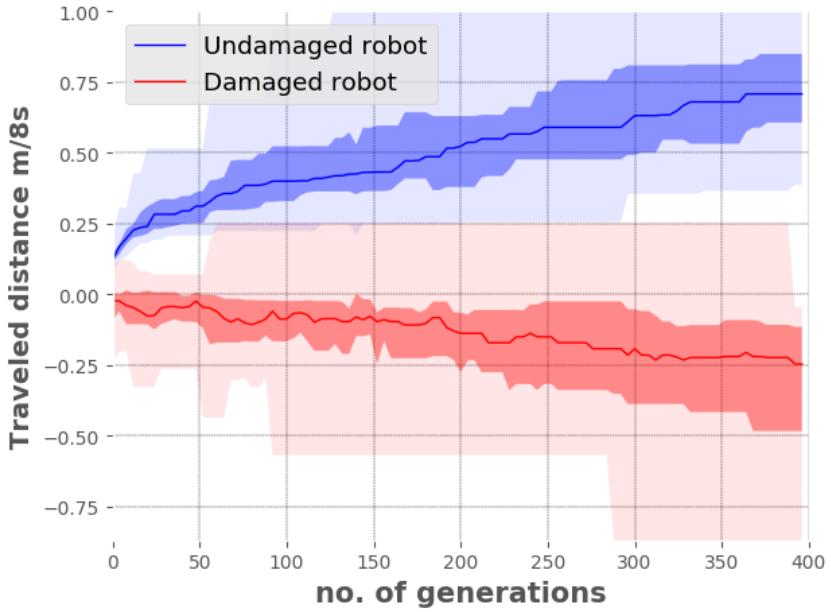
6.3.5 Expérimentation et résultats

L'expérience s'agit de faire évoluer les robots pour être capable de poursuivre le déplacement dans la même lignée après une panne d'un moteur donné. Nous optimisons alors la distance parcourue par un robot intact et la distance parcourue par le même robot, mais avec une jointure endommagée. Au cours de l'évaluation, les pannes sont introduites successivement. Toutes les jointures sont donc désactivées et évaluées séparément. Ceci est nécessaire, car si nous en sélectionnons seulement quelques-unes pour être endommagées, le processus d'évolution favorisera les robots dont ces dernières ne sont pas d'une grande importance et n'apportent pas de l'efficacité aux comportements. Par exemple des articulations telles que le cou et les doigts. Ceci est dû au fait que le seul but de l'AE est de satisfaire la fonction objectif, même si elle ne répond pas aux attentes et au but final des ingénieurs dans le domaine. Par conséquent, si un moteur contrôlant une jointure critique et importante échoue pendant l'exécution de la mission, le robot se trouve fortement affecté et reste sur place.

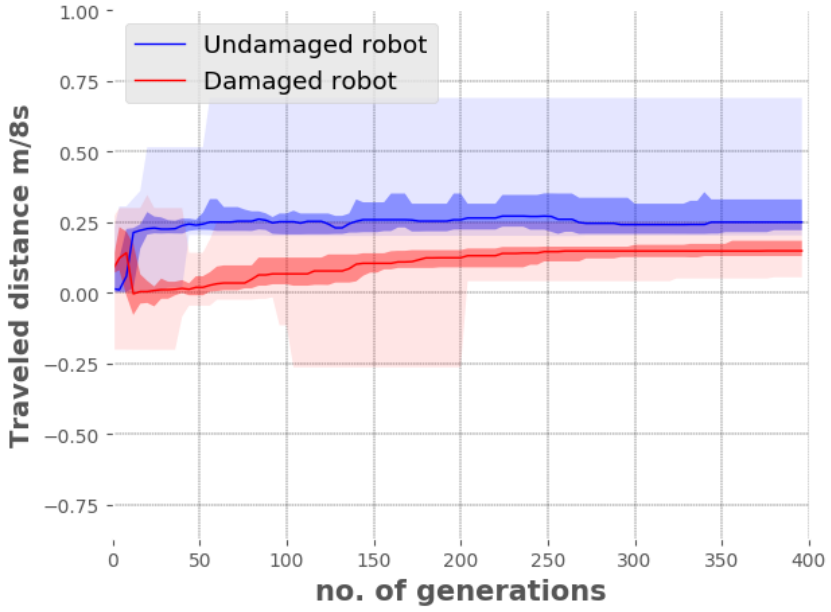
Nous remarquons qu'il est difficile de trouver des robots qui parcourent des distances considérables avant, et en même temps après l'endommagement. Afin de surmonter ce problème, nous avons introduit une pénalité. Celle-ci va faire une pression pour favoriser l'émergence des morphologies et des contrôleurs appropriés. Il s'agit de réduire la valeur *fitenss* qui détermine la résilience (deuxième objectif) du robot ne dépassant pas les 20 *cm* de déplacement, en l'absence de panne moteur. Nous multiplions cette valeur par 1,5.

résultats

Les graphes de la figure 6.8 illustrent les courbes de convergence du processus évolutif. Elles montrent la médiane, l'écart interquartile (ruban foncé) et les valeurs minimum et maximum (ruban clair) de 25 expériences évolutives indépendantes. Le graphe 6.8a représente les meilleures solutions selon le premier objectif (robots



(a)



(b)

FIGURE 6.8 – Solutions prises des extrémités du front de Pareto ; (a) les robots les moins résilients, pouvant parcourir de longues distances quand ils sont intacts (b) les robots les plus résilients. La courbe bleue représente la distance parcourue sans joints défectueux. La courbe rouge représente la distance parcourue avec un joint défectueux.

capables de parcourir de longues distances en l'absence de défaillances), alors que la 6.8b représente les meilleures solutions selon le deuxième objectif (robots pouvant parcourir une longue distance vers la cible quand il y a une défaillance). Les graphes de la figure montrent la distance parcourue par les robots avec et sans défaillance à chaque génération.

La figure 6.8a montre que les robots peuvent parcourir jusqu'à 70 *cm* en moyenne, mais en contrepartie, lorsqu'un dommage survient, ils restent sur place ou continuent à se déplacer vers une autre direction (distance négative) environ 25 *cm*. En ce qui concerne les autres solutions prises de l'autre extrémité du front de Pareto (6.8b), les robots intacts se déplacent en moyenne de 25 *cm* et lorsque des dommages surviennent, ils se déplacent en moyenne de 15 *cm*. Cependant, le déplacement est dirigé vers la cible, comme le montre la 6.8b.

La figure 6.9 illustre les fronts de Pareto des dernières générations des 25 expériences effectuées. Les robots qui sont dans le côté gauche sont les plus résilients. Lorsqu'une défaillance survient, ils conservent toujours la même direction de locomotion, alors que les autres ne le font pas. Nous pouvons remarquer que les robots qui parcourent entre 20 *cm* et 40 cm, lorsqu'ils sont endommagés, ils ont la capacité de maintenir la même direction de locomotion. De plus, ils peuvent faire presque la moitié de la distance dans le même laps du temps.

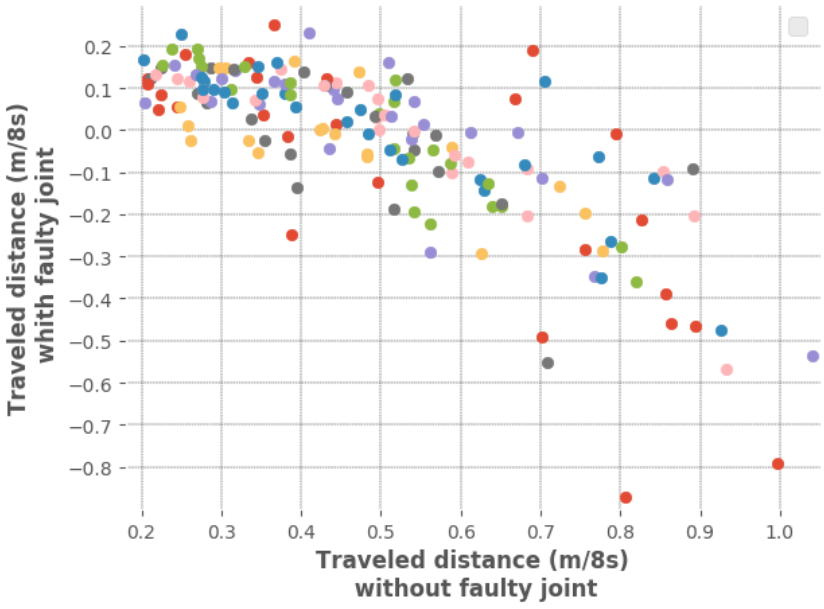


FIGURE 6.9 – Les individus pris des fronts de Pareto des dernières générations des 25 expériences évolutives.

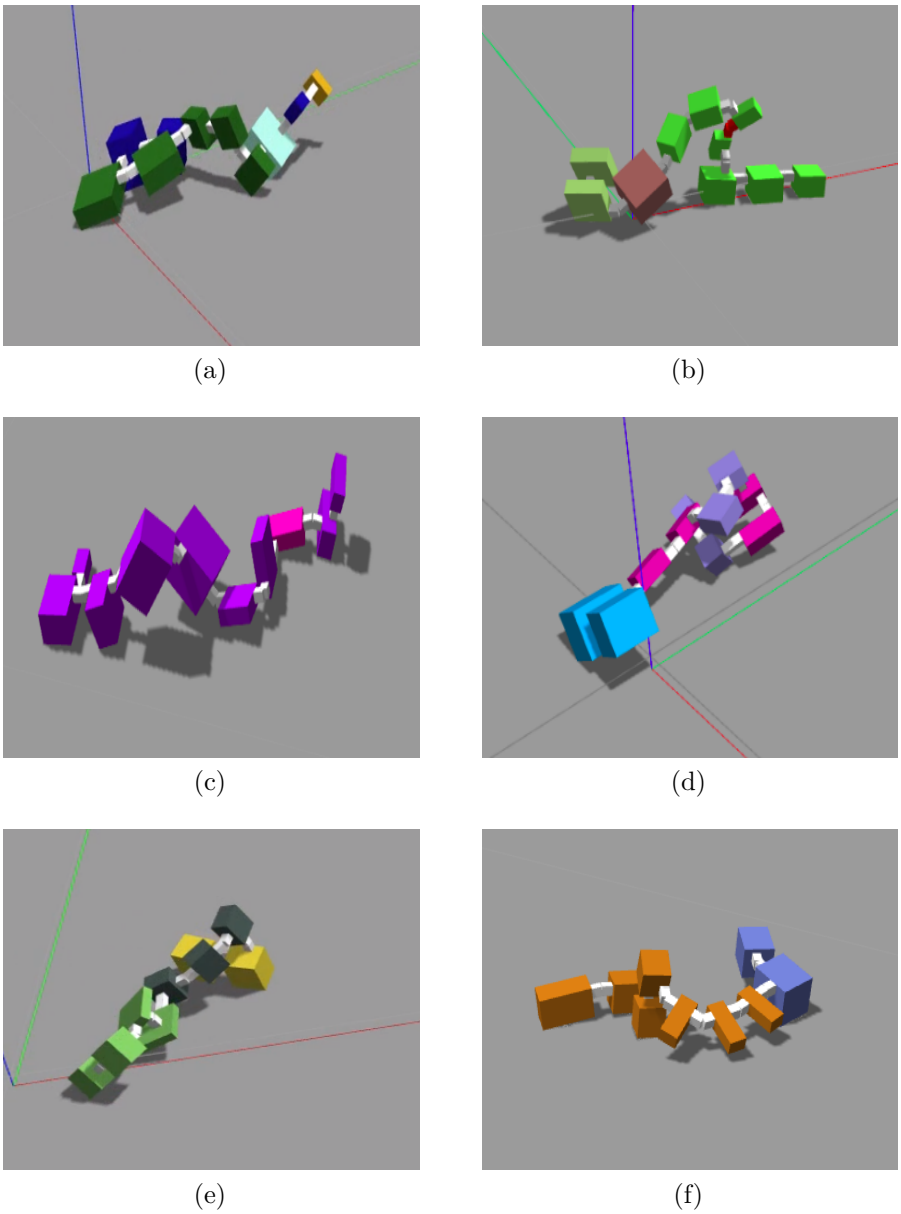
6.3.6 Discussion

Nous pouvons constater que les robots qui meuvent lentement peuvent être facilement résilients, alors que ceux qui meuvent plus vite ne le sont pas. Ces derniers ont des difficultés à maintenir la direction de déplacement en cas de dommage. Les morphologies générées dans ce cas sont intéressantes et montrent de bonnes stratégies de locomotion. Néanmoins, il y a toujours une ou deux articulations qui jouent le rôle crucial pour faire les mouvements efficaces. Par conséquent, le robot peut soit rester en place, soit changer complètement la direction de locomotion.

En ce qui concerne les robots qui se déplacent lentement et parcourent moins de distance, ils sont capables de garder la même direction de déplacement après les dommages. Nous avons remarqué que la défaillance d'un moteur n'affecte pas radicalement la manière de locomotion. La performance, dans le pire des cas, diminue à 40%. Le système de contrôle a réussi à contrôler toutes les articulations d'un robot afin de coopérer lorsqu'il y a un joint défectueux. C'est parce qu'aucune des articulations ne joue un rôle important pour la réalisation de mouvements effectifs. Cependant, en échange, les robots meuvent lentement.

Une grande proportion des morphologies des robots résilients obtenus (regarder la figure 6.10) sont semblables à des serpents, mais avec des axes de rotation différents au niveau des articulations. Avoir ce genre d'articulations permet aux robots de développer des comportements de rampement. Chaque fois une articulation est désactivée, d'autres articulations peuvent compenser et ainsi continuer le déplacement. Les robots qui en résultent sont capables de se déplacer sur la même ligne de déplacement avec et sans dommage, même si leur performance (vitesse) peut être légèrement réduite. Cependant, si ces robots avaient plus de temps d'évaluation, ils pourraient parcourir les mêmes distances et atteindre les positions atteintes par les robots intacts.

Cette approche de co-évolution, à l'opposé des autres approches, elle a cherché à déterminer si la résilience peut être obtenue par l'évolution de la morphologie. Nous avons intentionnellement causé des dommages aux moteurs du robot durant le temps d'évaluation. Effectivement, l'évolution a généré des morphologies qui ne peuvent pas être considérablement affectées par les dommages. Notre approche est capable de générer des locomotions intéressantes de robots ainsi qu'une capacité à se remettre des dégâts et à continuer d'avancer. Le même contrôleur était alors en mesure de contrôler le robot intact et aussi le robot avec les dommages.



(a) (b) (c) (d) (e) (f)

FIGURE 6.10 – Certains des robots modulaires évolués qui ont des capacités de résilience.

6.4 Conclusion

Nous avons démontré que notre système peut générer des robots modulaires hétérogènes réalistes et crédibles. L'intégration de notre système dans Gazebo et l'accès à de nombreux paramètres physiques permettent d'obtenir des simulations réalistes. Notre système a généré des robots intéressants qui ont appris à trouver des stratégies créatives de locomotion pour parcourir les plus longues distances. Ces robots sont en outre résilients. Soumettre les morphologies à évolution a permis l'émergence de celles qui tolèrent les pannes moteur. Nous avons intentionnellement causé des dommages aux moteurs du robot durant le temps d'évaluation. Effectivement, l'évolution a généré des morphologies qui ne peuvent pas être considérablement affectées par les dommages. Notre approche est capable de générer des locomotions intéressantes de robots ainsi qu'une capacité à se remettre des dégâts et à continuer d'avancer. Le même contrôleur était alors en mesure de contrôler le robot intact et aussi le même robot avec des dommages.

Chapitre 7

Évaluation du modèle

7.1 Introduction

Ce chapitre se consacre à l'évaluation et la validation de notre modèle de génération de robots modulaires. Pour cela, une étude comparative sera faite avec les différents modèles présentés dans la littérature. La comparaison sera structurée autour des critères suivants : la résilience, le réalisme des simulations et des comportements développés.

7.2 Récapitulatif des résultats obtenus

Comme la plupart des modèles de la littérature [Sims, 1994b, Krcah, 2007, Chaumont et al., 2007, Lassabe et al., 2007], celui que nous présentons ici considère l'évolution conjointe des morphologies et des contrôleurs. Cette manière d'évolution permet l'émergence de robots adaptables et créatifs [Bongard et al., 2015, Pfeifer and Bongard, 2006]. Cependant, contrairement à ces modèles, nous avons mis l'accent sur l'aspect réalisme des simulations. Nous avons obtenu ainsi des robots réalistes capables de se mouvoir en toute crédibilité dans l'environnement de simulation. Les mouvements étaient proches de ceux des vrais robots. Les articulations pouvaient réaliser des rotations tout en soulevant avec elles un certain nombre de modules.

Comme nous l'avons pu le voir, l'interpénétration entre les modules fait en grande partie perdre le réalisme. Nous avons réglé cette limitation par la conception de jointures en tant que modules. L'utilisation du simulateur multi-robot Gazebo est sans doute l'une des raisons pour lesquelles nous avons obtenu de tels résultats. La plate-forme a la particularité d'utiliser des moteurs physiques les plus précis, et de fournir des capteurs simulés qui produisent un flux de données qui correspond étroitement aux données provenant des capteurs physiques réels. Afin d'empêcher l'émergence

des cas spéciaux irréalistes de robots, pouvant fausser le fonctionnement normal du système d'évolution, nous avons ajusté par des valeurs réalistes les attributs fournis pour l'environnement et les objets simulés. Par ailleurs, avant d'évaluer les robots en simulation, nous avons conçu un prétraitement de vérification d'interpénétration de modules pour corriger les créatures inappropriées.

Nous avons dans un premier temps, généré des robots capables de réaliser des déplacements de longues distances. En effet, c'est une tâche complexe, car leurs optimisations doivent respecter à la fois des contraintes morphologiques et aussi fonctionnelles. Les résultats expérimentaux montrent qu'en effet les tailles des robots impactent leurs capacités à parcourir de longues distances. Plus il y a de modules et d'articulations, plus des morphologies variées et intéressantes peuvent être générées. Avec ces dernières, on a de meilleures possibilités d'obtenir des comportements efficaces permettant des locomotions distantes. Les comportements générés sont émergés via la coopération entre les modules composant le robot. Nous rappelons que chacun des modules est contrôlé séparément, mais avec une possibilité de communication. Nous avons conclu que les grands robots sont plus appropriés pour cet effet. Il suffit seulement de fournir le couple nécessaire pour chaque jointure.

Dans une deuxième série d'expérimentations, nous avons pu faire évoluer des robots résilients. Ceux-ci pouvaient continuer à avancer après une éventuelle panne moteur, mais aussi garder la même direction de déplacement. Cependant, la performance (vitesse) diminue. Cela est normal par rapport à ce qui se passe dans la nature et avec les êtres vivants. En fait, les approches présentées dans la littérature déployaient des robots conventionnels aux morphologies prédéterminées. Elles présentaient des résultats satisfaisants. Néanmoins, ce type de robots est limité dans le sens où il peut être conçu que pour convenir à des missions particulières et des modes de locomotions précis. Par ailleurs, la présence des parties critiques et très actives, dans lesquelles l'efficacité de locomotion y repose, représente un inconvénient. Nous avons proposé donc de concevoir des robots sans ces parties. Pour cela, nous avons impliqué la morphologie dans le processus d'évolution. Ce dernier engendre alors des robots qui peuvent tolérer les dommages. Au cours des générations, les morphologies vont s'adapter aux types de dommages causés, et les contrôleurs vont apprendre à être en mesure de manipuler les jointures de façon à satisfaire la fonction *objectif*. En fait, c'est la synergie qui se produit entre ces éléments (morphologie et contrôleur) qui finalement va définir le comportement global souhaité du robot.

7.3 Validation

Afin de valider notre système de génération de robots modulaires, il est nécessaire de pouvoir effectuer une étude comparative entre nos résultats et ceux présentés dans

la littérature. Cela va permettre d'une part mettre en avant les points forts de notre système, et d'une autre part montrer ses points faibles. Toutefois, il demeure difficile dans notre cas de réaliser une telle comparaison, car comme nous avons pu le constater, la robotique évolutionnaire est un domaine immense et les chercheurs possèdent divers objectifs et motivations. Par ailleurs, même si nous possédons des objectifs communs, souvent on utilise des types de robots différents, des environnements de simulation dissemblables et des paramètres expérimentaux totalement hétérogènes dont les valeurs dépendent de ces derniers. Par conséquent, on ne peut pas avoir un *Benchmark* d'évaluation. Nous proposons donc de mener une comparaison factuelle, en se basant sur les trois critères présentés ci-dessous.

7.3.1 Réalisme des créatures

Nous croyons que le réalisme des simulations, que se soit au niveau de la visualisation ou au niveau comportemental, est très important et doit être considéré en simulation. Certes le réalisme ne va pas résoudre radicalement le problème de *Reality Gap*, mais nous espérons qu'il contribuera à le dépasser. En plus du fait que c'est agréable à regarder, une simulation réaliste nous permet de mieux comprendre le fonctionnement des systèmes robotiques évoluant physiquement. Afin de mener cette comparaison, nous allons récapituler les caractéristiques concernant le réalisme des créatures vues précédemment au chapitre de l'état de l'art. Nous allons prendre celles dont les morphologies et les contrôleurs ont été évolué ensemble.

Tableau 7.1 – Comparaison des créatures du point du vu réalisme

Créature	**Non présence d'interpénétration**	**Animation ou vitalité perçue**	**Possibilité de transfère en monde réel**
[Sims, 1994b]	-	x	-
[Taylor and Massey, 2001]	-	x	-
[Ray, 2001]	-	-	-
[Lipson and Pollack, 2000]	-	-	x
[Hornby and Pollack, 2001]	x	-	x
[Shim and Kim, 2006]	x	-	-
[Miconi and Channon, 2006]	-	-	-
[Lassabe et al., 2007]	-	x	-
[Chaumont et al., 2007]	-	-	-
[Krcah, 2007]	-	x	-
[Cheney et al., 2013]	-	-	-
[Lessin et al., 2014]	x	x	-
[Auerbach et al., 2014]	x	x	x
[Akrour et al., 2017]	x	x	-

Nous pouvons voir à travers le tableau 7.1 que les créatures montrant une certaine animation et vitalité perceptuelle ne sont pas forcément réalistes. Elles présentent des interpénétrations qui faussent ainsi toutes crédibilités, que se soit au niveau du réalisme des simulations ou des comportements effectués. Il est par conséquent difficile de reproduire les créatures en monde réel. Toutefois certains travaux [Lipson and Pollack, 2000, Hornby and Pollack, 2001] arrivent à construire des robots physiques à partir des créatures simulées, malgré les interpénétrations. Dans ce cas, il est nécessaire d'apporter des changements au niveau de la conception du robot réel, notamment par rapport aux jointures et l'emplacement des parties électroniques. Cela n'est pas pratique et demande de fournir des efforts supplémentaires inutiles. Par contre Auerbach et son équipe, dans leur récent travail [Auerbach et al., 2014], ont conçu les robots simulés sans en apporter de changements et modifications. Cependant, leur objectif principal est la conception physique de robots modulaires. En fait, dans le cadre de nos travaux, cela ne fait pas partie de nos objectifs actuels, mais nous pouvons en conclure que c'est une tâche faisable malgré les difficultés rencontrées en simulations.

Il est difficile de montrer en images le réalisme des robots virtuels. En effet, cela demande de visualiser les simulations afin de voir comment ils se comportent réellement. Nos robots sont réalistes et se déplacent de façon crédible lorsqu'ils sont soumis aux différentes forces imposées par les contrôleurs et l'environnement. En fait, un environnement de simulation précis et un système de génération de robots variés sont d'autant plus importants que le contrôleur pour l'obtention de tels résultats.

7.3.2 Comportements développés

Les robots cités dans l'état de l'art montrent différents modes de locomotion. En fait, cela dépend fortement de leurs environnements et structures morphologiques. Certains peuvent nager ou même voler, mais dans cette étude notre attention sera portée particulièrement sur les locomotions terrestres. Par ailleurs, pour réaliser une comparaison juste, nous allons prendre les robots dans lesquels les comportements sont évolués conjointement avec leurs morphologies. Car, les robots modulaires physiques vus dans la section 3.4, font évoluer des contrôleurs pour un nombre de topologies connues [1] prédéterminées. Dans ce cas, les locomotions sont prévisibles et attendues.

Comme toutes les approches citées dans le tableau 7.2, la nôtre permet de générer des locomotions différentes et plus variées. Nous avons observé majoritairement des rampements, des roulements et des mouvements créatifs de serpents et de chenilles.

1. Généralement des quadrupèdes, des serpents, en forme de H ou des cercles roues qui font des mouvements de roulement circulaire

Tableau 7.2 – Les modes de locomotions permis par la co-évolution des quelques créatures artificielles

Approche	Ramper	Réaliser des sauts	Rouler sur sois même	Poussoir	Serpenter	Tel une chenille	Tel un quadrupède	Avec plusieurs pattes	Multiple *	Unique **
	Locomotion								Fitness	
[Sims, 1994b]	x	x	-	-	-	-	-	-	x	-
[Taylor and Massey, 2001]	x	-	-	-	-	-	-	-	x	-
[Ray, 2001]	-	-	-	-	-	-	-	-	-	x
[Lipson and Pollack, 2000]	-	-	-	x	-	-	-	-	-	x
[Hornby and Pollack, 2001]	-	-	x	-	-	x	-	-	x	-
[Shim and Kim, 2006]	-	-	-	-	-	-	-	-	-	x
[Miconi and Channon, 2006]	x	-	-	-	-	-	-	-	-	x
[Lassabe et al., 2007]	x	x	x	-	-	x	-	x	x	-
[Chaumont et al., 2007]	x	-	x	-	-	-	-	-	-	x
[Krcah, 2007]	x	x	-	-	-	-	-	-	x	-
[Lessin et al., 2014]	-	x	-	-	x	-	x	-	x	-
[Akrour et al., 2017]	x	x	x	-	x	x	x	-	-	x

* Pour l'émergence d'un certain mode de locomotion, on utilise une *fitness* différente ou des pénalités.

** On utilise une seule *fitness*, à partir de laquelle les différents modes de locomotions sont émergés.

Sur certaines expériences, les créatures ont développé des techniques de saut et de glissement afin d'avancer. Les images de ces robots ont été montrées dans la partie résultat (voir section 6.2.5).

Nous remarquons que la plupart des approches utilisent des fonctions *fitness* différentes, ou alors exercent des pénalités et des pressions pour encourager l'émergence des modes précis de locomotions. Par exemple mesurer la hauteur maximale au-dessus du sol de la partie la plus basse de la créature afin qu'elle puisse sauter. [Sims, 1994b] Contrairement à cela, notre approche a pu générer les différentes locomotions en utilisant une seule fonction *objectif* ; maximiser les distances parcourues. En fait, cela prouve la complétude de notre approche. Le système d'évolution est capable de fouiller de manière plus efficace (pas parfaitement) l'espace de recherche qui fournit les solutions variées proposées par notre modèle de génération de robots modulaires.

La majorité des travaux ne fournissent pas d'informations concernant les distances parcourues ou les vitesses atteintes, à l'exception de quelques-uns que nous rassemblons dans le tableau 7.3. Les robots présentés dans ce dernier ont des topo-

Tableau 7.3 – Les performances des créatures et des robots réalisés physiquement cités dans l'état de l'art

	Configuration	DOF	Taille* (cm)	Mode de locomotion	Vitesse (cm/s)
SuperBot [Shen et al., 2006]	Quadrupède	8	25*25*25	Marcher	36
	Roue	6	25*8*25	Rouler	100
M-tran II [Kamimura et al., 2005]	Quadrupède	8	18*18*18	Marcher	23
	Roue	6	18*6*18	Rouler	49.6
	Serpent	6	72*6*6	Tel une chenille	7.9
T-resilience*** [Koos et al., 2013]	Hexapode	18	62 (longueur)	Marcher	26
IT&E*** [Cully et al., 2015]	Hexapode	18	62 (longueur)	Marcher	30
Nos robots** [Akrour et al., 2017]	Variés	9	Pas plus de $\simeq$ 25*25*25	Variés	$\simeq 7.5$

* Taille approximative déduite à partir du nombre de modules du robot et leurs tailles. (longueur*largeur*hauteur)
** Nous avons pris les robots à 10 modules qui peuvent avoir majoritairement des formes semblables aux serpents.
*** Performance sur robot physique.

logies prédéterminées. Les ingénieurs réarrangent les modules afin de construire à chaque fois une forme particulière. C'est plus facile alors d'optimiser des contrôleurs permettant des comportements efficaces. Par exemple, synchroniser les mouvements des pattes d'un quadrupède pour réaliser des grands pas, ou alors faire rouler un robot roue qui facilement pourra atteindre un $1m/s$ de vitesse en descente [Shen et al., 2006].

En revanche dans notre cas, les topologies et les comportements des robots sont émergés conjointement par évolution. Comme discuté dans la section 2.5, faire co-évoluer des robots modulaires est une tâche difficile et jusqu'à présent reste une question et une problématique à résoudre. Par ailleurs, les systèmes d'évolution ne permettent pas l'émergence de robots bien conçus tels que nous pourrons le construire manuellement, par exemple des quadrupèdes et des humanoïdes [2].

Cela est premièrement dû à l'espace de recherche qui est toujours faible en termes de solutions possibles. Dans ce cas, il faut définir et préciser tous les aspects de construction d'un robot. Par exemple optimiser les forces nécessaires permettant de tenir debout un humanoïde ou un torse avec des pattes. Cela signifie qu'il faut prendre en compte un nombre exorbitant de paramètres pour être optimisés, qui reste toutefois difficile, voire impossible. Deuxièmement, c'est dû au processus d'évolution lui-même, qui ne peut pas explorer et exploiter un tel espace de recherche. Par conséquent, il

2. On a jamais pu faire concevoir de tels robots par co-évolution. Néanmoins, certains travaux comme celui de Nicolas Lassabe [Lassabe et al., 2007] génèrent des formes lointainement semblables aux vrais robots quadrupèdes et humanoïdes.

va tenter de satisfaire la fonction *objectif* avec les moyens les plus faciles. Ainsi, il va ignorer les solutions complexes [3], qui potentiellement sont meilleures. Celles-ci demandent un apprentissage supplémentaire et par la suite vont finir par disparaître, laissant la place à celles dont il leur est facile et moins complexe de répondre à l'objectif. C'est pour cela que nos robots sont généralement des serpents qui rampent, roulent ou serpentent.

Nous remarquons que malgré le fait que nos robots sont générés par évolution et par rapport aux autres ils sont relativement petits, ils arrivent à parcourir quand même de longues distances. Pour des robots à 9 DOF, en moyenne, ils parcourent $170cm/27s$ et donc atteignent des vitesses approximatives de 7.5 cm/s.

7.3.3 Capacité de résilience

Nous allons prendre sous cet angle de l'étude comparative les travaux récents concernant la conception des robots résilients aux pannes physiques, particulièrement les pannes moteur affectant une seule articulation. Nous montrons à travers le tableau ci-dessous que nos robots peuvent, contrairement aux autres, s'adapter aux changements physiques en seulement quelques secondes, sans la mise en œuvre d'un plan de secours. Ils sont donc considérablement plus rapides.

Les approches évoquées dans ce tableau utilisent des contrôleurs différents pour compenser du dommage. Cela leur permet de contrôler le robot avec sa nouvelle morphologie sans les parties endommagées. Avec notre approche, nous utilisons un seul contrôleur robuste émergé conjointement avec une morphologie adéquate. Les parties endommagées ne vont pas être contrôlées, et dans ce cas le contrôleur a déjà appris comment agir.

Les vitesses et les distances parcourues par les robots après la détection des dommages baissent, causant ainsi une dégradation de performance. Nous remarque une dégradation de 66.7% et 31.25% respectivement pour les deux premières approches, et 40% pour notre approche. Il faut dire que cela est naturel et prévisible. Le robot endommagé ne peut pas faire mieux que le robot intact.

3. Les robots conventionnels avec des structures symétriques (humanoïdes, quadrupèdes, etc.) ou même ceux que nous ne pouvons pas imaginer

Tableau 7.4 – Les distances parcourues par les robots résilients avant et parés l'endommagement.

Robot	Structure	Taille en longueur	DOF	Performance avant le dommage	Performance après le dommage	Baisse de performance	Maintien de la direction de déplacent	Temps de récupération
[Koos et al., 2013]	Hexapode	62cm	18	0.78m/3s	0.26m/3s	66.7%	non	20m
[Cully et al., 2015]	Hexapode	62cm	18	0.32m/s	0.22m/s	31.25%	oui	2m
[Akrour and Djedi, 2019]	Variée	≃30cm	9	0.25m/8s	0.15m/8s	40%	oui	0m

7.4 Limitation

Tout d'abord, notre modèle de génération de robots est testé sur des terrains simples sans présence d'obstacles. De plus, les tâches réalisées par les robots sont uniquement des taches de locomotions. En ce qui concerne la résilience, notre modèle n'a pas pris en compte d'autres types de dommages. Les robots sont résilients seulement aux pannes moteur.

Par railleurs, nous avons remarqué que les morphologies de nos robots résilients émergés ne sont pas assez variées. Cela en effet reste un défi dans ce domaine de la robotique évolutionnaire. Il n'y a pas assez de diversité dans les résultats des travaux que nous trouvons dans la littérature, alors que l'objectif principal de ce type d'étude et d'expérimentation est justement de découvrir de nouvelles variations de morphologie et comportement. La préoccupation principale est donc de pouvoir produire continuellement de la nouveauté et aussi des robots qui sont de plus en plus complexes.

7.5 Bilan

Tout d'abord, notre modèle a montré une nouvelle approche de génération de robots modulaires résilients et réalistes. En nous basant sur l'évolution conjointe des morphologies et des contrôleurs, nous avons pu obtenir des robots variés avec des comportements créatifs de locomotion. Les différents points sur lesquels nous avons mené notre étude comparative montrent la complétude du modèle vis-à-vis des différents modèles de la littérature. En effet, nous avons contribué à résoudre d'une part la problématique de résilience des robots et d'un autre part, améliorer l'apparence visuelle, la crédibilité et le réalisme de ces derniers, et ce par l'utilisation du simulateur multi-robot Gazebo et le développent d'un système de génération de robots variés par l'élargissement de l'espace de recherche ou de solutions possibles. Cependant, il convient de noter que notre modèle ne répond pas à tous les objectifs

fixés au tout début de la thèse (décrits dans la section 1.2). Nous rappelons que l'objectif principal était d'étudier la croissance des créatures artificielles résilientes. Le point non abordé est la génération des créatures artificielles via morphogénèse. Cela signifie qu'elles doivent subir une phase de développement et de croissance à partir d'une seule cellule à l'image des organismes multicellulaires. Au cours des recherches effectuées durant l'étape de l'état de l'art, nous avons pris d'autres orientations. Nous nous sommes intéressés à la robotique et particulièrement aux robots modulaires. Nous avons donc utilisé un système de codage direct qui a permet de traduire le génotype des créatures en phénotypes.

7.6 Conclusion

Dans ce chapitre nous avons pu évaluer notre approche et la situer par rapport aux travaux de la littérature. En l'absence d'un *Benchmark* d'évaluation, il était nécessaire de procéder à une comparaison factuelle, dans laquelle s'est structurée autour de trois critères ; la résilience, le réalisme des simulations et des comportements développés. En fait, l'étude comparative que nous avons effectuée a prouvé la complétude de notre système de génération de robots modulaire réalistes et résilients.

Chapitre 8

Conclusion générale

8.1 Discussion

Nous avons traité dans cette thèse la problématique de génération automatique de robots modulaires résilients, à laquelle deux contributions ont été proposées dans le domaine de la robotique évolutionnaire.

Premièrement, nous avons proposé un modèle de génération de robots modulaires réalistes par l'évolution conjointe des morphologies et des contrôleurs. Nous avons intégré le système d'évolution dans la plateforme de simulation multi-robot Gazebo. Cela nous a permis d'obtenir des comportements de robots réalistes. Notre approche est motivée par l'aide qu'elle apporte pour combler l'écart entre la perception et la réalité "*Reality Gap*". Par ailleurs, à travers une série d'expérimentations, nous avons démontré que notre modèle est capable de produire une diversité de comportements et de morphologies. Les robots générés peuvent parcourir de longues distances en un minimum de temps en effectuant des locomotions créatives. Comparés à d'autres robots ou créatures artificielles, les robots générés dans le cadre de nos expérimentations ont la particularité de se comporter de manière réaliste avec une vitalité perçue au niveau des simulations.

Deuxièmement, nous avons proposé une nouvelle approche pour générer des robots modulaires résilients. Cette caractéristique de résilience est la capacité qui permet aux robots de se remettre des dommages subis de façon autonome, c'est-à-dire sans intervention externe. Nos robots pouvaient alors, après une éventuelle panne de moteur, poursuivre leurs déplacements tout en gardant la même direction. En fait, nous avons fait évoluer les robots pour qu'ils soient en mesure de s'adapter aux pannes causées délibérément durant leurs temps de vie, ou au moment de l'évaluation. Ainsi, au cours des générations, les robots vont apprendre à développer des morphologies adaptables et deviennent, en quelque sorte, résilients. Notre approche actuelle est

différente de celles présentées dans la littérature, puisqu'elle permet aux robots de s'adapter aux changements en un temps très réduit sans la mise en œuvre d'un plan de secours.

8.2 Perspectives

En tant que travaux futurs, nous suggérons diverses perspectives liées aux systèmes robotiques modulaires. Premièrement, nous proposons d'améliorer notre modèle en complexifiant davantage la conception morphologique des robots. Par exemple ajouter des modules ayant des formes de roue ou de cylindre. En outre, nous proposons de complexifier les tâches à exécuter ainsi que l'environnement de simulation correspondant. Cela va en effet agrandir l'espace de recherche et ouvrir la voie à une émergence de robots encore plus créatifs que variés, capables par exemple de se déplacer dans des terrains accidentés, inondés ou glacés. Cependant, des capteurs supplémentaires doivent être mis en œuvre pour permettre de nouvelles capacités de détections. Nous allons donc exploiter les capteurs proposés par Gazebo et inclure des capteurs lasers. Bien que cela ait été étudié dans la littérature, la complexification reste une tâche sans limites.

Dans cette thèse, nous n'avons pas abordé la partie morphogénèse des robots virtuels. Elle fera donc l'une des perspectives de ce mémoire. En effet, le choix de la méthode de représentation génétique des créatures impacte le type des solutions générées. Dans le cadre de nos travaux, nous avons utilisé les *Graphtal*. Il s'agit d'une méthode de codage directe, qui traduit directement le génotype en son phénotype correspondant. La morphogénèse (codage génératif), quant à elle, elle est la plus proche des organismes multicellulaires complexes, et potentiellement va permettre la conception de robots plus créatifs.

En ce qui concerne la résilience, nous proposons premièrement d'induire des dommages plus extrêmes, tels que des membres manquants ou cassés, et aussi introduire plus qu'un dommage comme c'est ce qui arrive inévitablement dans le monde réel. Deuxièmement, nous proposons de garder les mouvements réalisés suites aux dommages subis pour s'en servir plus tard. En fait, après la détection des dommages, généralement les vitesses et les distances parcourues baissent, causant ainsi une dégradation de performance. Cependant dans certains cas, les robots endommagés peuvent réaliser des comportements différents que de rester sur place immobiles. Comme nous l'avons vu dans nos expériences, les robots peuvent tourner à gauche ou à droite et continuer le déplacement tout en gardant la même performance. Cela va dépendre bien entendu de la jointure rompue. Il serait intéressant de pouvoir collecter ces données, et de désactiver une jointure particulière afin de réaliser une

rotation ou aller vers une autre cible. Cela va nous éviter de changer à chaque fois de contrôleur.

Nous envisageons de réaliser des applications en monde réel. Nous allons reproduire physiquement les meilleurs robots émergés en évolution. Il demeure cependant un défi à cause de la *Reality gap* et la non-fluidité du transfert des simulations à la réalité. Par ailleurs, cette perspective conduirait à la considération des consommations énergétiques et les emplacements des parties électroniques (câbles, moteurs, batteries, etc.) au sein du robot. Nous allons intégrer ces éléments dans les génotypes et donner la main aux systèmes évolutionnaires de trouver les meilleures solutions.

Bibliographie

[Akrour et al., 2017] Akrour, D., Cussat-Blanc, S., Sanchez, S., Djedi, N., and Luga, H. (2017). Joint evolution of morphologies and controllers for realistic modular robots. In *The Twenty-Second International Symposium on Artificial Life and Robotics*, pages 57–62. Artificial Life and Robotics. 7.1, 7.2, 7.3

[Akrour and Djedi, 2019] Akrour, D. and Djedi, N. (2019). Damage recovery for simulated modular robots throught joint evolution of morphologies and controllers. *Journal of Automation, Mobile Robotics & Intelligent Systems*. 7.4

[Auerbach et al., 2014] Auerbach, J., Aydin, D., Maesani, A., Kornatowski, P., Cieslewski, T., Heitz, G., Fernando, P., Loshchilov, I., Daler, L., and Floreano, D. (2014). Robogen : Robot generation through artificial evolution. In *Artificial Life 14 : Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, number EPFL-CONF-200995, pages 136–137. The MIT Press. 3.2, 3.3, 3.4.2, 7.1, 7.3.1

[Auerbach and Bongard, 2014] Auerbach, J. E. and Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS computational biology*, 10(1) :e1003399. 2.5

[Banzhaf, 2003] Banzhaf, W. (2003). Artificial regulatory networks and genetic programming. In *Genetic programming theory and practice*, pages 43–61. Springer. 2.3.2

[Berenson et al., 2005] Berenson, D., Estevez, N., and Lipson, H. (2005). Hardware evolution of analog circuits for in-situ robotic fault-recovery. In *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, pages 12–19. IEEE. 2.4

[Blanke et al., 2006] Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., and Schröder, J. (2006). *Diagnosis and fault-tolerant control*, volume 2. Springer. 2.4

[Bongard et al., 2006] Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802) :1118–1121. 2.4, 2.8a, 4.5, 6.3.1

[Bongard et al., 2015] Bongard, J. C., Bernatskiy, A., Livingston, K., Livingston, N., Long, J., and Smith, M. (2015). Evolving robot morphology facilitates the evolution of neural modularity and evolvability. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 129–136. ACM. 2.2.1, 2.3.2, 2.3.3, 5.2.2, 7.2

[Brooks, 1990] Brooks, R. A. (1990). Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2) :3–15. 4.5

[Cappelle et al., 2016] Cappelle, C. K., Bernatskiy, A., Livingston, K., Livingston, N., and Bongard, J. (2016). Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features. *Frontiers in Robotics and AI*, 3 :59. 2.2.1

[Cardamone et al., 2009] Cardamone, L., Loiacono, D., and Lanzi, P. L. (2009). Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM. 2.3.2

[Chaumont et al., 2007] Chaumont, N., Egli, R., and Adami, C. (2007). Evolving virtual creatures and catapults. *Artificial life*, 13(2) :139–157. 2.3.2, 7.2, 7.1, 7.2

[Cheney et al., 2016] Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. (2016). On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Proceedings of the Artificial Life Conference*, volume 2016, pages 226–234. 2.5, 2.5.1

[Cheney et al., 2017] Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. (2017). Scalable co-optimization of morphology and control in embodied machines. *arXiv preprint arXiv :1706.06133*. 2.5.1

[Cheney et al., 2013] Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution : evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 167–174. ACM. 2.3.2, 2.5, 7.1

[Chernova and Veloso, 2004] Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2562–2567. IEEE. 2.3.2

[Cliff and Miller, 1995] Cliff, D. and Miller, G. F. (1995). Co-evolution of pursuit and evasion ii : Simulation methods and results. 2.3.2

[Clune et al., 2009] Clune, J., Beckmann, B. E., Ofria, C., and Pennock, R. T. (2009). Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2764–2771. IEEE. 2.3.2

[Clune et al., 2013] Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proc. R. Soc. B*, 280(1755) :20122863. 2.3.3

[Coello et al., 2007] Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer. 2.2.2

[Collins et al., 2013] Collins, T., Ranasinghe, N. O., and Shen, W.-M. (2013). Remod3d : A high-performance simulator for autonomous, self-reconfigurable robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4281–4287. IEEE. 4.2, 4.2.1

[Cully et al., 2015] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553) :503. 2.3.2, 2.4, 2.4, 2.5, 4.5, 6.3.1, 7.3, 7.4

[Cully and Mouret, 2013] Cully, A. and Mouret, J.-B. (2013). Behavioral repertoire learning in robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 175–182. ACM. 2.3.2

[Cussat-Blanc and Pollack, 2014] Cussat-Blanc, S. and Pollack, J. (2014). Cracking the egg : Virtual embryogenesis of real robots. *Artificial life*, 20(3) :361–383. 2.3.2

[Darwin, 1859] Darwin, C. (1859). *The origin of species by means of natural selection*. Modern Lib. 2.2.1

[Dasgupta et al., 2013] Dasgupta, P., Baca, J., Hossain, S., Dutta, A., and Nelson, C. (2013). Mechanical design and computational aspects for locomotion and reconfiguration of the modred modular robot. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1359–1360. International Foundation for Autonomous Agents and Multiagent Systems. 3.4.2

[Davey et al., 2012] Davey, J., Kwok, N., and Yim, M. (2012). Emulating self-reconfigurable robots-design of the smores system. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4464–4469. IEEE. 3.4.1

[De Jong, 1975] De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. 2.2.1

[Deb et al., 2002] 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. 2.2.2

[Degroote, 2012] Degroote, A. (2012). *Une architecture de contrôle distribuée pour l'autonomie des robots*. PhD thesis. 4.2.3, 4.2.3

[Doncieux et al., 2015] Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). Evolutionary robotics : what, why, and where to. *Frontiers in Robotics and AI*, 2 :4. 2.3.3, 2.5, 2.5.1, 4.5

[Doursat et al., 2012] Doursat, R., Sayama, H., and Michel, O. (2012). Morphogenetic engineering : Reconciling self-organization and architecture. In *Morphogenetic Engineering*, pages 1–24. Springer. 2.3.2

[Echeverria et al., 2011] Echeverria, G., Lassabe, N., Degroote, A., and Lemaignan, S. (2011). Modular open robots simulation engine : Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51. IEEE. 4.2, 4.2.2, 4.2.3

[Eggenberger, 1997] Eggenberger, P. (1997). Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the fourth european conference on Artificial Life*, pages 205–213. 2.3.2

[Eiben et al., 2013] Eiben, A., Bredeche, N., Hoogendoorn, M., Stradner, J., Timmis, J., Tyrrell, A., Winfield, A., et al. (2013). The triangle of life : Evolving robots in real-time and real-space. *Advances in artificial life, ECAL*, 2013 :1056. 2.3.3, 4.5

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, 521(7553) :476. 4.5

[Ellery, 2017] Ellery, A. (2017). Building physical self-replicating machines. In *Proceedings of the 14th European Conference on Artificial Life ECAL*, volume 14, pages 146–153. 4.5

[Erden and Leblebicioğlu, 2008] Erden, M. S. and Leblebicioğlu, K. (2008). Free gait generation with reinforcement learning for a six-legged robot. *Robotics and Autonomous Systems*, 56(3) :199–212. 2.4

[Floreano and Mondada, 1994] Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent. In *From Animals to Animats 3 : Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 421–430. The MIT Press. 4.5

[Fogel, 1995] Fogel, D. B. (1995). Evolutionary computation : toward a new philosophy of machine intelligence. 2.2.1

[Fukuda and Nakagawa, 1988] Fukuda, T. and Nakagawa, S. (1988). Approach to the dynamically reconfigurable robotic system. *Journal of Intelligent and Robotic Systems*, 1(1) :55–72. 3.1

[Fulcher, 2008] Fulcher, J. (2008). Computational intelligence : an introduction. In *Computational intelligence : a compendium*, pages 3–78. Springer. 2.2.1

[Goldberg, 2013] Goldberg, D. E. (2013). *The design of innovation : Lessons from and for competent genetic algorithms*, volume 7. Springer Science & Business Media. 2.5.1

[Gucwa and Cheng, 2014] Gucwa, K. J. and Cheng, H. H. (2014). Robosim : A simulated environment for programming modular robots. In *Mechatronic and Em-*

bedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on, pages 1–7. IEEE. 3.4.2

[Guo et al., 2009] Guo, H., Meng, Y., and Jin, Y. (2009). A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. *BioSystems*, 98(3) :193–203. 2.3.2

[Haasdijk et al., 2014] Haasdijk, E., Bredeche, N., and Eiben, A. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PloS one*, 9(6) :e98466. 4.5

[Hagan et al., 1996] Hagan, M. T., Demuth, H. B., Beale, M. H., et al. (1996). *Neural network design*, volume 20. Pws Pub. Boston. 2.3.2, 5.2.2

[Haji Agha Memar et al., 2008] Haji Agha Memar, A. H., Haji Bagher, P. Z., and Keshmiri, M. (2008). Mechanical design and motion planning of a modular reconfigurable robot. In *Advances In Mobile Robotics*, pages 1090–1097. World Scientific. 3.2

[Hancher and Hornby, 2006] Hancher, M. D. and Hornby, G. S. (2006). A modular robotic system with applications to space exploration. In *Space Mission Challenges for Information Technology, 2006. SMC-IT 2006. Second IEEE International Conference on*, pages 8–pp. IEEE. 3.3

[Holland, 1992] Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1) :66–73. 2.2.1

[Hollnagel et al., 2007] Hollnagel, E., Woods, D. D., and Leveson, N. (2007). *Resilience engineering : Concepts and precepts*. Ashgate Publishing, Ltd. 2.4

[Hornby and Pollack, 2001] Hornby, G. S. and Pollack, J. B. (2001). Evolving l-systems to generate virtual creatures. *Computers & Graphics*, 25(6) :1041–1048. 2.3.2, 2.3.2, 7.1, 7.3.1, 7.2

[Hupkes et al., 2018] Hupkes, E., Jelisavcic, M., and Eiben, A. (2018). Revolve : A versatile simulator for online robot evolution. In *International Conference on the Applications of Evolutionary Computation*, pages 687–702. Springer. 4.5

[Ijspeert, 2008] Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots : a review. *Neural networks*, 21(4) :642–653. 3.4.1

[Ivaldi et al., 2014] Ivaldi, S., Padois, V., and Nori, F. (2014). Tools for dynamics simulation of robots : a survey based on user feedback. *arXiv preprint arXiv :1402.7050*. 4.2.2

[Jakobi, 1998] Jakobi, N. (1998). Running across the reality gap : Octopod locomotion evolved in a minimal simulation. In *European Workshop on Evolutionary Robotics*, pages 39–58. Springer. 2.3.2

[Jakobi et al., 1995] Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap : The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer. 4.5

[Kamimura et al., 2005] Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., and Kokaji, S. (2005). Automatic locomotion design and experiments for a modular robotic system. *IEEE/ASME Transactions on mechatronics*, 10(3) :314–325. 7.3

[Keijzer et al., 2001] Keijzer, M., Merelo, J. J., Romero, G., and Schoenauer, M. (2001). Evolving objects : A general purpose evolutionary computation library. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 231–242. Springer. (document), 2.2

[Klaus et al., 2012] Klaus, G., Glette, K., and Tørresen, J. (2012). A comparison of sampling strategies for parameter estimation of a robot simulator. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 173–184. Springer. 4.5

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE. 4.2, 4.2.2, 4.2.2

[Kohl et al., 2006] Kohl, N., Stanley, K., Miikkulainen, R., Samples, M., and Sherony, R. (2006). Evolving a real-world vehicle warning system. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1681–1688. ACM. 2.3.2

[Koos et al., 2013] Koos, S., Cully, A., and Mouret, J.-B. (2013). Fast damage recovery in robotics with the t-resilience algorithm. *The International Journal of Robotics Research*, 32(14) :1700–1723. 2.4, 4.5, 7.3, 7.4

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA. 2.2.1

[Krcah, 2007] Krcah, P. (2007). Evolving virtual creatures revisited. In *GECCO*, volume 7, pages 341–341. 5.4.3, 7.2, 7.1, 7.2

[Krupke et al., 2015] Krupke, D., Wasserfall, F., Hendrich, N., and Zhang, J. (2015). Printable modular robot : an application of rapid prototyping for flexible robot design. *Industrial Robot : An International Journal*, 42(2) :149–155. 3.2

[Krčah, 2008] Krčah, P. (2008). Towards efficient evolution of morphology and control. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 287–288. ACM. 2.3.2

[Kuehn and Rieffel, 2012] Kuehn, T. and Rieffel, J. (2012). Automatically designing and printing 3-d objects with evofab 0.2. *Artificial life*, 13 :372–378. 4.5

[Kurokawa et al., 2008] Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., and Murata, S. (2008). Distributed self-reconfiguration of m-tran iii modular robotic system. *The International Journal of Robotics Research*, 27(3-4) :373–386. 3.4.1

[Langton, 1995] Langton, C. G. (1995). Artificial life : An overview. 2.3.1

[Lassabe, 2008] Lassabe, N. (2008). *Morphogenese et Evolution de Créatures artificielles*. PhD thesis, Université des Sciences Sociales, Toulouse, France. 2.3.1, 2.3.2, 2.3.2

[Lassabe et al., 2007] Lassabe, N., Luga, H., and Duthen, Y. (2007). A new step for artificial creatures. In *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*, pages 243–250. IEEE. 2.3.2, 2.5, 6.2.1, 7.2, 7.1, 7.2, 2

[Lehman and Stanley, 2011a] Lehman, J. and Stanley, K. O. (2011a). Abandoning objectives : Evolution through the search for novelty alone. *Evolutionary computation*, 19(2) :189–223. 2.5.1

[Lehman and Stanley, 2011b] Lehman, J. and Stanley, K. O. (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218. ACM. 2.3.2, 2.5.1

[Lessin et al., 2014] Lessin, D., Fussell, D., and Miikkulainen, R. (2014). Adapting morphology to multiple tasks in evolved virtual creatures. In *Proceedings of The Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, volume 2014. 2.3.2, 7.1, 7.2

[Liedke et al., 2013] Liedke, J., Matthias, R., Winkler, L., and Wörn, H. (2013). The collective self-reconfigurable modular organism (cosmo). In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 1–6. IEEE. 3.4.1

[Lin and Chen, 2007] Lin, C.-M. and Chen, C.-H. (2007). Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1) :110–123. 2.4

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3) :280–299. 2.3.2, 2.3.2

[Lipson and Pollack, 2000] Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799) :974. 2.3.2, 2.3.2, 7.1, 7.3.1, 7.2

[Mahdavi and Bentley, 2003] Mahdavi, S. H. and Bentley, P. J. (2003). An evolutionary approach to damage recovery of robot motion with muscles. In *European Conference on Artificial Life*, pages 248–255. Springer. 2.4

[Mahdavi and Bentley, 2006] Mahdavi, S. H. and Bentley, P. J. (2006). Innately adaptive robotics through embodied evolution. *Autonomous Robots*, 20(2) :149–163. 2.4, 6.3.1

[Meeden, 1996] Meeden, L. A. (1996). An incremental approach to developing intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3) :474–485. 2.3.2

[Meng et al., 2011] Meng, Y., Zhang, Y., Sampath, A., Jin, Y., and Sendhoff, B. (2011). Cross-ball : a new morphogenetic self-reconfigurable modular robot. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 267–272. IEEE. 3.4.1

[Michalewicz, 1996] Michalewicz, Z. (1996). Evolution strategies and other methods. In *Genetic algorithms+ data structures= evolution programs*, pages 159–177. Springer. 2.2.1

[Miconi and Channon, 2006] Miconi, T. and Channon, A. (2006). An improved system for artificial creatures evolution. *Proceedings of Artificial Life X*, pages 255–261. 2.3.2, 2.3.2, 2.5, 7.1, 7.2

[Mitchell, 1998] Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press. 2.2.1, 2.2.1

[Mohiuddin et al., 2010] Mohiuddin, A., Khan, M., Billah, M., and Farhana, S. (2010). Walking hexapod robot in disaster recovery : developing algorithm for terrain negotiation and navigation. In *New Advanced Technologies*. InTech. 2.4

[Moore and McKinley, 2016] Moore, J. M. and McKinley, P. K. (2016). A comparison of multiobjective algorithms in evolving quadrupedal gaits. In *International Conference on Simulation of Adaptive Behavior*, pages 157–169. Springer. 2.3.2

[Mouret and Chatzilygeroudis, 2017] Mouret, J.-B. and Chatzilygeroudis, K. (2017). 20 years of reality gap : a few thoughts about simulators in evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1121–1124. ACM. 2.4, 4.5

[Mouret and Clune, 2015] Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv :1504.04909*. 2.5.1

[Mouret and Doncieux, 2012] Mouret, J.-B. and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics : An empirical study. *Evolutionary computation*, 20(1) :91–133. 2.3.2

[Nakagaki et al., 2016] Nakagaki, K., Dementyev, A., Follmer, S., Paradiso, J. A., and Ishii, H. (2016). Chainform : A linear integrated modular hardware system for shape changing interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 87–96. ACM. 3.4.2

[Nelson et al., 2004] Nelson, A. L., Grant, E., Galeotti, J. M., and Rhody, S. (2004). Maze exploration behaviors using an integrated evolutionary robotics environment. *Robotics and Autonomous Systems*, 46(3) :159–173. 2.3.2

[Nicolau et al., 2010] Nicolau, M., Schoenauer, M., and Banzhaf, W. (2010). Evolving genes to balance a pole. In *European Conference on Genetic Programming*, pages 196–207. Springer. 2.3.2

[Ouannes et al., 2012] Ouannes, N., Djedi, N., Duthen, Y., and Luga, H. (2012). Gait evolution for humanoid robot in a physically simulated environment. In *Intelligent computer graphics 2011*, pages 157–173. Springer. 2.3.2

[Pacheco et al., 2015] Pacheco, M., Fogh, R., Lund, H. H., and Christensen, D. J. (2015). Fable ii : Design of a modular robot for creative learning. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 6134–6139. IEEE. 3.4.2

[Pareto, 1964] Pareto, V. (1964). *Cours d'économie politique*, volume 1. Librairie Droz. 2.2.2

[Paul and Bongard, 2001] Paul, C. and Bongard, J. C. (2001). The road less travelled : Morphology in the optimization of biped robot locomotion. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 226–232. IEEE. 2.3.3

[Pfeifer and Bongard, 2006] Pfeifer, R. and Bongard, J. (2006). *How the body shapes the way we think : a new view of intelligence*. MIT press. 2, 2.3.3, 3.1, 7.2

[Pinville, 2013] Pinville, M. T. (2013). *Robotique évolutionniste : influence des pressions de sélection sur l'émergence d'une forme de mémoire interne*. PhD thesis, Université Pierre et Marie Curie, Paris, France. 2.3.2

[Pretorius et al., 2013] Pretorius, C. J., du Plessis, M. C., and Cilliers, C. (2013). Simulating robots without conventional physics : A neural network approach. *Journal of Intelligent & Robotic Systems*, 71(3-4) :319–348. 4.5

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer Science & Business Media. 2.3.2

[Pugh et al., 2016] Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity : A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3 :40. 2.3.2

[Qiao et al., 2012] Qiao, G., Song, G., Zhang, J., Sun, H., Wang, W., and Song, A. (2012). Design of transmote : a modular self-reconfigurable robot with versatile transformation capabilities. In *Robotics and biomimetics (ROBIO), 2012 IEEE international conference on*, pages 1331–1336. IEEE. 3.4.1

[Ram et al., 1994] Ram, A., Boone, G., Arkin, R., and Pearce, M. (1994). Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation. *Adaptive behavior*, 2(3) :277–305. 2.3.2

[Ray, 2001] Ray, T. S. (2001). Aesthetically evolved virtual pets. *Leonardo*, 34(4) :313–316. 2.3.2, 7.1, 7.2

[Rieffel et al., 2017] Rieffel, J., Mouret, J.-B., Bredeche, N., and Haasdijk, E. (2017). Introduction to the evolution of physical systems special issue. 4.5

[Roennau et al., 2013] Roennau, A., Sutter, F., Heppner, G., Oberlaender, J., and Dillmann, R. (2013). Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–7. IEEE. 4.2.1

[Romanishin et al., 2013] Romanishin, J. W., Gilpin, K., and Rus, D. (2013). M-blocks : Momentum-driven, magnetic modular robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4288–4295. IEEE. 3.4.1

[Salem, 2014] Salem, B. (2014). Petro : development of a modular pet robot. In *Robot and Human Interactive Communication, 2014 RO-MAN : The 23rd IEEE International Symposium on*, pages 483–488. IEEE. 3.4.2

[Salemi et al., 2006] Salemi, B., Moll, M., and Shen, W.-M. (2006). Superbot : A deployable, multi-functional, and modular self-reconfigurable robotic system. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3636–3641. IEEE. 3.3, 3.4.1

[Sanchez and Cussat-Blanc, 2014] Sanchez, S. and Cussat-Blanc, S. (2014). Gene regulated car driving : using a gene regulatory network to drive a virtual car. *Genetic Programming and Evolvable Machines*, 15(4) :477–511. 2.3.2

[Scheper and de Croon, 2017] Scheper, K. Y. and de Croon, G. C. (2017). Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics. *Artificial life*, 23(2) :124–141. 4.5

[Shen et al., 2006] Shen, W.-M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., and Venkatesh, J. (2006). Multimode locomotion via superbot reconfigurable robots. *Autonomous Robots*, 20(2) :165–177. 7.3, 7.3.2

[Shim and Kim, 2006] Shim, Y.-S. and Kim, C.-H. (2006). Evolving physically simulated flying creatures for efficient cruising. *Artificial Life*, 12(4) :561–591. 2.3.2, 7.1, 7.2

[Shin and Lee, 1999] Shin, J.-H. and Lee, J.-J. (1999). Fault detection and robust fault recovery control for robot manipulators with actuator failures. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 861–866. IEEE. 2.4

[Sims, 1994a] Sims, K. (1994a). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4) :353–372. 2.3.2, 2.3.2, 2.5, 5.2

[Sims, 1994b] Sims, K. (1994b). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM. 1.1, 2.3.2, 2.3.2, 5.2, 5.3.1, 5.4.3, 6.2.1, 7.2, 7.1, 7.2, 7.3.2

[Sproewitz et al., 2009] Sproewitz, A., Billard, A., Dillenbourg, P., and Ijspeert, A. J. (2009). Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4259–4264. IEEE. 3.4.1

[Stanley et al., 2009] Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2) :185–212. 2.3.2

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2) :99–127. 2.3.2

[Tarapore et al., 2016] Tarapore, D., Clune, J., Cully, A., and Mouret, J.-B. (2016). How do different encodings influence the performance of the map-elites algorithm ? In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 173–180. ACM. 2.3.2

[Tarapore and Mouret, 2015] Tarapore, D. and Mouret, J.-B. (2015). Evolvability signatures of generative encodings : beyond standard performance benchmarks. *Information Sciences*, 313 :43–61. 2.2.1, 2.3.2

[Taylor, 2017] Taylor, T. (2017). Evolution in virtual worlds. *arXiv preprint arXiv :1710.06055*. 2.5

[Taylor and Massey, 2001] Taylor, T. and Massey, C. (2001). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial Life*, 7(1) :77–87. 2.3.2, 2.3.2, 7.1, 7.2

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press. 2.4

[Visinsky, 1992] Visinsky, M. L. (1992). *Fault detection and fault tolerance methods for robotics*. PhD thesis, Rice University. 2.4

[Visinsky et al., 1994] Visinsky, M. L., Cavallaro, J. R., and Walker, I. D. (1994). Robotic fault detection and fault tolerance : A survey. *Reliability Engineering & System Safety*, 46(2) :139–158. 2.4

[Watson et al., 2002] 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. 4.5

[Wilson, 2002] Wilson, M. (2002). Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4) :625–636. 14

[Wolff and Nordin, 2002] Wolff, K. and Nordin, P. (2002). Evolution of efficient gait with an autonomous biped robot using visual feedback. In *Proceedings of the Mechatronics Conference. University of Twente, Enschede, the Netherlands*. Citeseer. 2.3.2

[Yim et al., 2000] Yim, M., Duff, D. G., and Roufas, K. (2000). Modular reconfigurable robots, an approach to urban search and rescue. In *Proceedings of the 1st International Workshop on Human-friendly Welfare Robotics Systems, Taejon, Korea*. 3.3

[Yim et al., 2007] Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., and Chirikjian, G. S. (2007). Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1) :43–52. (document), 2.4, 3.1, 3.2, 3.2, 3.1, 3.3

[Yosinski et al., 2011] Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J. C., and Lipson, H. (2011). Evolving robot gaits in hardware : the hyperneat generative encoding vs. parameter optimization. In *ECAL*, pages 890–897. 2.3.2

[Yu and Gen, 2010] Yu, X. and Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media. 2.2.1

[Zhang and Zhang, 1999] Zhang, L. and Zhang, B. (1999). A geometrical representation of mcculloch-pitts neural model and its applications. *IEEE Transactions on Neural Networks*, 10(4) :925–929. 2.3.2

[Zilberstein et al., 2002] Zilberstein, S., Washington, R., Bernstein, D. S., and Mouaddib, A.-I. (2002). Decision-theoretic control of planetary rovers. In *Advances in Plan-Based Control of Robotic Agents*, pages 270–289. Springer. 2.4

[Zitzler and Künzli, 2004] Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842. Springer. 2.2.2

[Zitzler et al., 2001] Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2 : Improving the strength pareto evolutionary algorithm. *TIK-report*, 103. 2.2.2

Acronyms

3D trois dimensions. 13, 29, 34, 35, 41–44, 46, 51

AE Algorithmes Évolutionnaires. 6–8, 22, 23, 25, 28, 40, 59, 60, 82

AEMO Algorithmes Évolutionnaires Multi-Objectifs. 11, 15, 23, 81

AG Algorithmes Génétiques. 7, 8, 10, 15–18, 60

API Interface de Programmation Applicative. 36, 43, 46, 48, 49, 67, 69

CPG Central Pattern Generator. 33, 40

CPPN Compositional Pattern-Producing Network. 19

DOF degrés de liberté. 31–37, 39, 40, 68, 76, 94

EPS Évolution des Systèmes Physiques. 50

GPS Global Positioning System. 62, 63

GRN Réseaux de Régulation Génétique. 14, 15, 40, 57

hNEAT Hierarchical NEAT. 18

HyperNEAT Hypercube-based NEAT. 15

IBEA Indicator Based Evolutionary Algorithm. 11

IMU Inertial Measurement Unit. 62

IT&E Intelligent Trial and Error. 24, 50, 93

MLP Perceptron Multicouche. 57, 58

NEAT NeuroEvolution of Augmenting Topologies. 15, 18, 19

NSGA-II Non-dominated Sorting Genetic Algorithm. 11, 60, 81

ODE Open Dynamics Engine. 42, 43, 48, 49

PE Programmation Évolutionnaire. 7

PG Programmation Génétique. 7, 15

QD Quality Diversity. 15

RNA Réseaux de Neurones Artificiels. 15–17, 40, 57, 59, 61

SDF Simulation Description Format. 44, 45, 62

SE Stratégies d'Évolution. 7

SPEA2 Strenght Pareto Evolutionary Algorithm. 11

Swig Simplified Wrapper and Interface Generator. 48

XML Extensible Markup Language. 44, 45, 48, 62