Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider Biskra

Faculté des Sciences Exactes, des sciences de la Nature et de la Vie

Département d'Informatique

N° d'ordre :…..……
Série :……

Mémoire

Présenté en vue de l'obtention du diplôme de Magister en Informatique

Option**: Data Mining et Multimédia**

Titre :

Reconfiguration et évolution des morphologies de robots modulaires

Par :
Melle Chourouk GUETTAS

Soutenu le : 18 / 06 / 2014

Devant le jury :

Pr. NourEddine DJEDI	**Prof.**	**Université de Biskra**	**Président**
Dr. Foudil CHERIF	**MCA**	**Université de Biskra**	**Rapporteur**
Dr. Med. Chaouki BABAHENINI	**MCA**	**Université de Biskra**	**Examinateur**
Dr. Kamel Eddine MELKEMI	**MCA**	**Université de Biskra**	**Examinateur**

To my Parents, Family, Friends, Teachers …

A special gratitude to my loving Mother

Whose words of encouragement and push for tenacity ring in my ears,

For being there for me, listening and supporting whenever I need a loving heart and a shoulder lo lean on.

To my Father

Who tolerated, endured and lift up my spirit with an open heart along the way.

To my Brothers and Sisters

For believing in me.

To my Best Friends

Without names, you were always there backing me up and encouraging, you have been my best cheerleaders.

"To Those Who Were Around Me Whispering, "You Can Do It" When I Feel Like There Is No Way.

Remerciements

En premier lieu, je remercie « Dieu »
de m'avoir donné la patience, la santé et le courage
pour arriver jusqu'à là.

Je tiens à remercier le docteur
Foudil Cherif pour son encadrement, sa disponibilité, son suivi, ses conseils et ses critiques constructives.

Je remercie également l'ensemble des membres du jury pour avoir accepté de consacrer leur temps à examiner ce travail malgré leurs nombreuses responsabilités et occupations.

Je tiens à remercier, Mr Yves Duthen et Thomas Breton, pour faciliter notre recherche en nous donnant leur simulateur développés, la documentation nécessaire et leurs conseils.

En fin, Je remercie le Pr. Kasper Stoy et Glen Arnold Armadillio pour la documentation et la correction des articles.

Abstract

The general approach in modular self-reconfigurable robots is to hand design the morphology, then optimizes the controller of the structure for a given task and finally determines a sequence of motions to reconfigure the robot from an initial configuration to a goal configuration. Evolutionary robotics has proposed evolution as a bio-inspired approach to overcome the limitations of human intuition in designing robots and controllers; theoretically, the resulting structures will be better adapted. In this work, we propose an approach based on cooperative co-evolutionary genetic algorithms to design configurations and controllers for homogenous modular robots implicitly to support self-reconfiguration; The algorithm introduces some elements to make finding solutions easier and faster by co-evolving two populations; a population of motions sequence to search a sequence of movements that can rearrange a given modular configuration into a new one that suits a different task defined by its desired function and a population of homogenous fixed topology ANNs for the controllers to perform locomotion as a behavior evolved using genetic algorithm based on standard deviation norm. The modular robots are evaluated in a simulation environment implemented with NVidia physics engine; PhysX. The experiments carried out in this work indicate that co-evolving both the configuration and the controllers positively contributes to the robot's performance and optimizes its locomotion behavior.

Résumé

L'approche générale en robots modulaires est de concevoir la morphologie, puis optimiser les contrôleurs de la structure pour une tâche donnée et finalement rechercher la séquence de mouvements qui permet de passer d'une configuration initiale à une configuration but. La robotique évolutionnaire a proposé l'évolution comme une approche bio-inspirée pour surmonter les limites de l'intuition humaine dans la conception des configurations et de contrôleurs de robots, en théorie, les structures résultantes seront mieux adaptées. Dans ce travail, nous avons proposé une approche basée sur les algorithmes génétiques co-évolutionnaires pour concevoir des configurations et des contrôleurs de robots modulaires homogènes implicitement pour permettre l'auto-reconfiguration; nous avons utilisé deux populations, une première population pour évoluer une séquence de mouvements qui peuvent réorganiser une configuration modulaire donnée à une autre qui est plus adaptée à une tâche définie par sa fonction et une deuxième population de RNAs pour évoluer les contrôleurs. Les robots modulaires sont évalués dans un environnement de simulation implémenté avec le moteur physique de NVidia, PhysX. Les expériences menées dans ce travail indiquent que la co-évolution de la configuration et les contrôleurs à contribuer positivement à la performance de robots modulaires et optimiser son comportement de locomotion.

ملخص

يمكن تلخيص المقاربة العامة لتصميم الروبوتات الوحدوية في: تصميم الشكل، ومن ثم تحسين نظام التحكم لمهمة معينة. اقترح منهج الآلية التطورية التطور كنهج مستوحى من البيولوجيا للتغلب على نقائص الحدس الإنساني في تصميم الروبوتات و معالجة هذه المشكلة ؛ نظريا، سيتم تكييف هياكل الروبوت لتقديم أداء أفضل. في هذا العمل ، فإننا نقترح اتباع نهج يقوم على الخوارزميات التطورية الوراثية التعاونية لتصميم أنظمة تحكم و هياكل لروبوتات مكونة من وحدات متجانسة ، الخوارزمية المقترحة تقدم بعض العناصر لجعل إيجاد الحلول أسهل و أسرع من خلال التطور التعاوني المشترك بين صنفين مختلفين؛ صنف من سلسة من الحركات للبحث على التسلسل الحركي الذي يمكنه إعادة ترتيب التكوين الوحدوي للروبوت للوصول إلى شكل جديد يمكنه أداء المهمة والوظيفة المطلوبة بكفاءة أعلى و صنف آخر من أنظمة التحكم المتماثلة (ANNs) للسيطرة على أداء الروبوت الوحدوي في عملية التنقل، لتطويرهذا الصنف استخدامنا خوارزمية جينية تعتمد على قاعدة الانحراف المعياري. يتم تقييم الروبوتات الوحدوية في بيئة محاكاة باستعمال المحرك الفيزيائي PhysX. التجارب التي أجريت في هذا العمل تشير إلى أن التطور التعاوني المشترك بين أنظمة التحكم و تركيب الروبوت الوحدوي يسهم بشكل إيجابي في أداء الروبوتات وتحسين السلوك الحركي لها بدرجة كبيرة.

Table des matières

La liste des Figures

La liste des tables

INTRODUCTION

Introduction Générale

"Imagination is the beginning of creation. You imagine what you desire, you will what you imagine and at last you create what you will."

—George Bernard Shaw

Un système robotique auto-reconfigurable, qui vise à changer sa connectivité pour s'adapter à une tâche, est captivant. Un tel système se situe entre l'ambition de l'être humain et ses capacités technologique courantes ; imaginaire si en regardant les films de la science-fiction (par exemple : *morphing robots, liquid-metal robots, nano-robots, etc.*), réaliste en examinant les recherches sur les robots auto-reconfigurables qui se développent dans le monde entier, attirant des chercheurs actifs et gagnant l'intérêt public.

L'idée principale de la robotique modulaire est qu'à partir d'une collection de modules relativement simples aux fonctionnalités plutôt limitées il est possible de réaliser collectivement une tâche complexe. Les atouts d'une telle approche sont principalement l'adaptabilité à des situations nouvelles (changement de fonction, variations du terrain) et la robustesse du système (il suffit au robot de remplacer un module déficient pour s'auto-réparer).

Donc, les robots auto-reconfigurables ont un avantage sur les robots conventionnels de forme fixe dans les milieux non structurés, complexes et imprévus grâce à leurs capacités spéciales (en notant que les robots conçus pour une tâche spécifique performent toujours mieux que les robots modulaires). Les robots auto-reconfigurables peuvent facilement explorer un environnement inconnu et se déplacer sur sa surface, en dépit de son état. Des tâches et des applications diverses pour ces robots seront trouvées en orbite ; dans la mer profonde, dans les zones sinistrées par les radiations ou produits chimiques toxiques, le sauvetage, la manipulation de structures en environnement hostile (domaine spatial) ou encore la nanorobotique.

L'approche générale de la conception des robots modulaires est de concevoir la morphologie, puis d'optimiser les contrôleurs de la structure physique pour une tâche donnée et enfin de déterminer une séquence de mouvements pour reconfigurer le robot à partir d'une configuration initiale à une autre cible pour résoudre le problème d'auto-reconfiguration.

Mais, afin d'opérer dans des environnements non structurés, les robots modulaires devront être adaptatifs, ils doivent exhiber un comportement intelligent. La robotique évolutionnaire ; une méthode bio-inspirée dans laquelle les algorithmes évolutionnaires sont utilisés pour optimiser le système de contrôle d'un robot, a été prouvé à surmonter les limites de l'intuition humaine dans la conception de stratégies de contrôle pour les machines autonomes. Bien que

la robotique évolutionnaire propose tas de possibilités, la majorité des travaux a seulement optimisé les contrôleurs pour une morphologie conçue par l'être humain ou bio-inspirée. Cette méthodologie a de sérieux inconvénients: La fixation de la morphologie, en prenant en compte les contraintes émergeantes de la forme du module, pose des limites sur les types d'actions qu'un robot peut exercer, et même si elle trouve les contrôleurs optima, le robot résultant n'est certainement pas la meilleure solution possible pour la tâche.

Néanmoins, la nature manifeste la co-évolution, il n'y a pas de distinction entre un corps et un cerveau, chacun performe son rôle. L'évolution ne viendrait pas avec une morphologie et ensuite évoluer un contrôleur. La solidarité des solutions de la nature émerge du processus de co-évolution, coopératif ou compétitif, principalement pour permettre aux créatures de survivre dans leur environnement.

Les robots modulaires sont particulièrement adaptés à la co-évolution. La complexité de la conception de la configuration et la programmation des contrôleurs pour un robot croît d'une façon exponentielle avec le nombre de modules utilisés, et peut-être impossible pour un robot de milliers de modules. Les algorithmes co-évolutionnaires nous donnent la possibilité de faire évoluer la morphologie et les contrôleurs simultanément pour répondre à une tâche spécifique.

Ce travail vise à surmonter les limites de l'intuition humaine dans la conception des configurations et des contrôleurs des robots modulaires auto-reconfigurables. Plus précisément, il se concentre sur les méthodes qui évoluent à la fois le système de contrôle et la morphologie de robots simulés tout en gardant la séquence de mouvements nécessaire pour l'auto-reconfiguration. Nous proposons, ici, une approche basée sur les algorithmes génétiques co-évolutionnaires coopératifs pour évoluer deux populations; une population avec un objectif principal de générer une suite de mouvements, permettant de passer d'une configuration connue du système à une autre configuration à priori inconnue mais qui doit réaliser au mieux une fonction donnée et une deuxième population de réseaux de neurones (RNA) pour optimiser les contrôleurs et renforcer les caractéristiques particulières de robots modulaires. L'apprentissage dans ces deux populations s'effectuera par le biais des algorithmes génétiques.

Nous testons notre approche sur trois configurations différentes afin de vérifier la validité du modèle proposé pour l'évolution des configurations plus adaptées et la production des allures de locomotion. Pour chaque configuration, nous effectuons deux tests, le premier par l'implémentation des RNAs sur une configuration fixe et le deuxième par la mise en œuvre de l'approche co-évolutionnaire. Une comparaison sera effectuée entre les deux pour montrer la différence entre les résultats de ces deux méthodes.

Dans le but de mesurer le comportement d'un robot par une fonction de fitness on peut soit le construire soit le simuler. La première alternative n'étant bien entendu pas réalisable à notre échelle, alors on va utiliser un simulateur physique qui permettra une bonne évaluation des performances d'un robot modulaire.

Nous allons organiser notre mémoire de la manière suivante : dans le premier chapitre, nous allons aborder la vie artificielle et les techniques qu'elle a présentées au domaine de robots modulaires. Par la suite et dans un deuxième chapitre, on va présenter les robots modulaires en donnant quelques concepts de base, leurs différents types, les caractéristiques d'un système robotique modulaire et ses applications. Dans le troisième chapitre, nous détaillons les composants de base de la conception d'un système robotique, les méthodes utilisées pour les développer, et quelques travaux réalisés. Ensuite, comme un quatrième chapitre, on va suivre le chemin général, en proposant des configurations bio-inspirées puis implémenter notre modèle de contrôleurs basant sur les réseaux de neurones, ainsi ses spécifications, effectuer les tests de validation et présenter les résultats obtenues. Finalement, dans le chapitre cinq, nous allons aborder notre approche co-évolutionaire et les résultats des simulations et conduire une comparaison entre l'approche traditionnelle et celle la co-évolutionnaire au niveau de performance. Nous terminerons par une conclusion générale et quelques perspectives.

CHAPITRE 1

La Vie Artificielle

"It has yet to be proven that intelligence has any survival value."

—Arthur C. Clarke

1. Introduction

De tout le temps, l'homme à tenter de reproduire la vie artificiellement ou certaines de ses propriétés, pour la mieux comprendre et/ou pour élaborer des algorithmes capable d'adapter et évoluer, afin de concevoir des systèmes et des machines intelligents. La vie Artificielle nous propose une démarche de deux étapes : l'abstraction des principes fondamentaux du vivant et leur recréation sur d'autres supports, nous autorisant des expérimentations nouvelles.

Contrairement à l'approche classique qui décrit toutes les étapes afin d'obtenir la solution d'un problème, l'approche évolutionniste se suffise avec une description du type de solutions recherchées, puis sélectionne les solutions les plus adaptées au problème afin de les reproduire à l'aide d'opérateurs de croisement et de mutations.

Dans ce chapitre, on va présenter la vie artificielle et ses techniques qui sont utilisées comme des méthodes de base dans la robotique modulaire.

2. La vie artificielle

2.1. Définition de la vie artificielle

Définir la vie artificielle n'est pas chose triviale. Il s'agit d'un domaine qui se situe dans l'intersection de plusieurs sciences, entre autres l'informatique et la biologie.

Langton [LAN89] l'a défini comme étant "*l'étude de systèmes conçus par l'homme qui présentent des comportements caractéristiques des systèmes vivants naturels.*"

Selon Heudin [HEU94] la vie Artificielle peut être définie comme "*la science des propriétés émergentes. Un ensemble de structures, du fait de leurs interactions, engendre des propriétés qui ne résultent pas de la simple superposition de leurs contributions individuelles.*"

Ces définitions sont très vastes et pour une meilleure précision Farmer et Belin [FAR89] ont proposé une lise de propriétés caractérisant un système de vie artificiel, ces critères principaux sont :

1. L'homme a contribué à la création d'un système de vie artificielle.
2. Un système de vie artificielle est autonome.
3. Un système de vie artificielle est en interaction avec son environnement.
4. Il y a émergence de comportements dans un système de vie artificielle.

Ces quatre propriétés au-dessus sont indispensables dans un système de vie artificielle. Cependant Farmer et Belin [FAR89] donnent trois autres qui sont souhaitables :

5. Un système de vie artificielle peut se reproduire lui-même.
6. Un système de vie artificielle possède une capacité d'adaptation.
7. Un système de vie artificielle ne possède pas nécessairement d'unité spatiale (un robot peut par exemple être contrôlé à distance par un programme).

En ce qui concerne la génération de créatures artificielles virtuelles, toutes ces propriétés sont respectées sans être toutes simulées. Un exemple est la reproduction des créatures qui s'effectue le plus souvent sans le soucie des ressources qui sont généralement considérées illimitées. Le problème s'émerge lorsqu'on veut que des robots puissent se reproduire physiquement, car il y a toujours des contraintes physiques qui ne sont pas respectées dans les environnements virtuels. En effet, les créatures artificielles évoluent dans un environnement qui ne peut pas simuler tous détails du monde réel. La reproduction physique des robots est donc une contrainte difficile à surmonter. Malgré cela quelques travaux comme ceux de Zykov et al. [ZYK05], qui tentent à réaliser l'auto-reproduction. Effectivement leur robot - constitué de cubes - peut, si on lui fournit d'autres cubes, reconstituer son double.

2.2. Les créatures artificielles

Avant de passer aux techniques utilisées par la vie artificielle, il faut donner une définition de ses acteurs principaux.

Lassab [LAS08] a défini les créatures artificielles sous formes des propriétés basant sur les critères d'une vie artificielle :

1. L'être humain a contribué au processus d'apparition de toute créature artificielle.
2. Une créature artificielle est une unité.
3. Une créature artificielle possède une morphologie.
4. Une créature artificielle doit interagir avec son environnement.
5. Une créature artificielle est autonome.

Les propriétés suivantes sont souhaitables :

6. Une créature artificielle s'auto-produit.
7. Une créature artificielle peut se reproduire elle-même.
8. Une créature artificielle s'auto-répare.
9. Une créature artificielle possède des mécanismes d'adaptation.

Selon Lassab [LAS08], une créature artificielle évolutionniste comme étant une entité virtuelle ou réelle possédant une morphologie et un comportement issus de processus évolutionnistes.

A partir de cette définition et des propriétés que l'on a décrits au-dessus, on peut en déduire que les robots sont des créatures artificielles. En effet les robots sont une unité car ils peuvent

avoir une indépendance énergétique par le biais d'une batterie. Ils interagissent physiquement avec l'environnement, ils peuvent être autonomes.

2.3. Evolution et sélection naturelle

Charles Darwin dans son livre « l'origine des espèces » a proposé sa théorie « la sélection naturelle », qui considère que seulement les individus les mieux adaptés à leur environnement au sein d'une espèce se survivent et se reproduisent.

Au cours des années 1930, une approche corrective nommée « Néo-Darwinisme » qui rapproche la théorie de Darwin avec celle de l'hérédité de Gregor Mendel. *(Figure 1.1)*

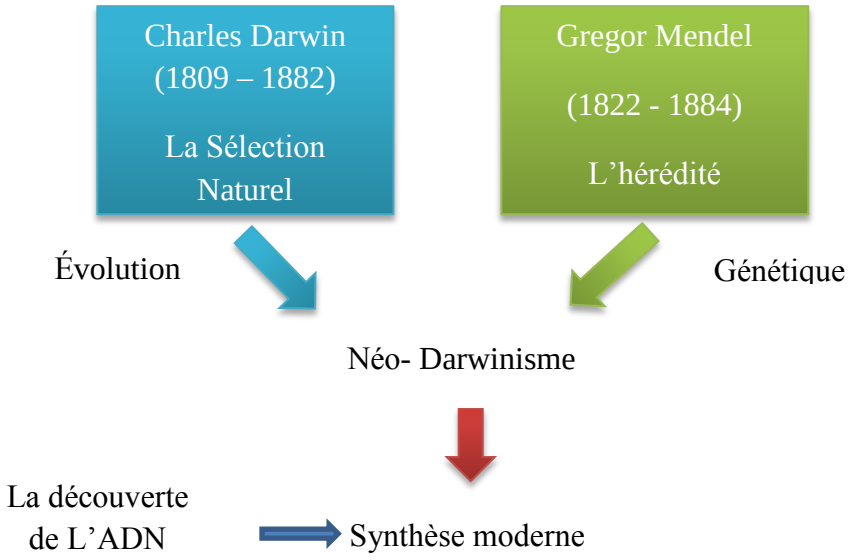


Figure 1.1 : La synthèse moderne [HEU94].

La découverte de L'ADN a conduit ensuite à ce qu'il s'appeler « La synthèse naturelle ». Celle-ci se repose sur trois idées principales :

1. L'évolution est le résultat d'un progrès continue des êtres vivants au cours des générations ;
2. La reproduction signifie l'existence d'une mémoire : les gènes subirent des modifications par mutation, recombinons, impliquant une grande diversité au niveau de l'hérédité ;
3. Le mécanisme principal est la sélection naturelle, les individus mieux adaptés survivent.

L'idée d'adaptation au milieu a fait intervenir la notion de « valeur d'adaptation » ou « fitness » qui s'augmente lorsqu'un individu améliore sa survie, dans les théories « d'adaptationnistes », cette valeur est liée à la « persistance » ; la probabilité d'éviter l'extinction de l'espèce.

L'évolution a deux processus principaux : la variation ; la génération d'une population diverse, et la sélection ; seuls les individus les plus adaptées survivent. L'évolution donc c'est un processus infini de variation-sélection.

L'une des projections de la théorie d'évolution sur le domaine de l'informatique, est les algorithmes génétiques, qu'on va les détaillés dans la section suivante.

2.4. Les Algorithmes génétiques

Les algorithmes génétiques sont basés sur la théorie de néo-darwinisme et qu'ont été développé initialement par Holland [HOL75], principalement pour tenter d'expliquer le processus d'évolution et d'adaptation, afin d'utiliser ce processus pour la conception de systèmes artificiels plus robustes, capables de s'adapter aux perturbations d'un environnement.

Les algorithmes génétiques sont des méthodes de recherche combinatoire, qui tentent à obtenir une solution optimale à un problème d'optimisation. Ils appartiennent à la famille des algorithmes évolutionnistes comme *les stratégies d'évolutions* inventées par Rechenberg dans les années 60 [REC65], *la programmation évolutionnaire* inventée par Fogel, Owens et Walsh en 1965 [FOG66] et *la programmation génétique* démocratisée par Koza en 1992 [KOZ92]. *Les algorithmes évolutionnistes* font eux partie de la famille des métaheuristiques comme le recuit simulé, les algorithmes de colonies de fourmis ou les algorithmes par essaims particulaires. Les algorithmes génétiques ont été popularisés à partir de 1989 notamment par des ouvrages tels que le livre de Goldberg [GOL89]. Ils sont l'une des méthodes utilisées, lorsque les algorithmes déterministes sont inefficaces ou inadaptés pour résoudre des problèmes dont l'espace de recherche est trop grand.

Les AG (*algorithmes génétiques)* utilisent la notion de gène. Une gène est un fragment d'information, plus généralement c'est une particularité d'un individu. On a par exemple chez l'être humain le gène de la couleur des yeux qui selon les informations qu'il contient va donner des yeux bleus, verts ou marrons. Le résultat "visible" de l'expression des gènes est ce qu'on appelle le *phénotype*.

Le *génotype* est un ensemble de gène, un individu a son génotype qui contient toutes les informations (gènes) permettant de le caractériser.

Le plus simple des algorithmes génétiques est composé de trois opérateurs : la reproduction, le croisement, la mutation. [GOL89]

- **La reproduction :** un processus qui copie une solution, pour produire une nouvelle solution identique à la première.
- **Le croisement :** une procédure qui, à partir de deux chromosomes de départ, produit deux nouveaux chromosomes, le croisement s'effectue en trois étapes : choisissant deux individus survivants ; les coupes dans le même endroit, celui-ci étant choisi aléatoirement ; et on recolle les deux chromosomes en les croisant, résultant ainsi des nouveaux individus.

- **La mutation :** qui modifie la valeur d'un allèle choisi en hasard, ce qui permet d'éviter une convergence prématurée en maintenant une diversité de solutions.

Dans les algorithmes génétiques, l'information à optimiser est représentée sous forme de chromosome (appelé également individu-solution). Par analogie avec la biologie ce terme désigne généralement une structure de données. Basant sur la théorie d'évolution, il associe la diversité et la survie de plus adapté. La première étape est de générer aléatoirement une population de chromosomes. Les individus sont ensuite évalués dans leur environnement par une fonction d'évaluation aussi appelée fonction de fitness. Cette fonction d'évaluation traduit le but que les individus doivent atteindre. Une fois les individus évaluent, ils sont sélectionnés en fonction de leur valeur d'adaptation en vue d'être croisés ou mutés. Les nouveaux individus générés ainsi que les anciens individus les plus adaptés forment la nouvelle génération. L'algorithme recommence avec la nouvelle population jusqu'à ce qu'il converge vers des individus qui satisfassent la fonction de fitness.

Les algorithmes génétiques ne garantissent pas de trouver les solutions optimales en un temps fini [GOL89][HOL78]. De plus il n'est pas toujours trivial de déterminer la bonne fonction d'évaluation qui donne les résultats désirés. Malgré cela le succès des algorithmes génétiques est grandissant et ils sont de plus en plus employés pour des problèmes difficiles.

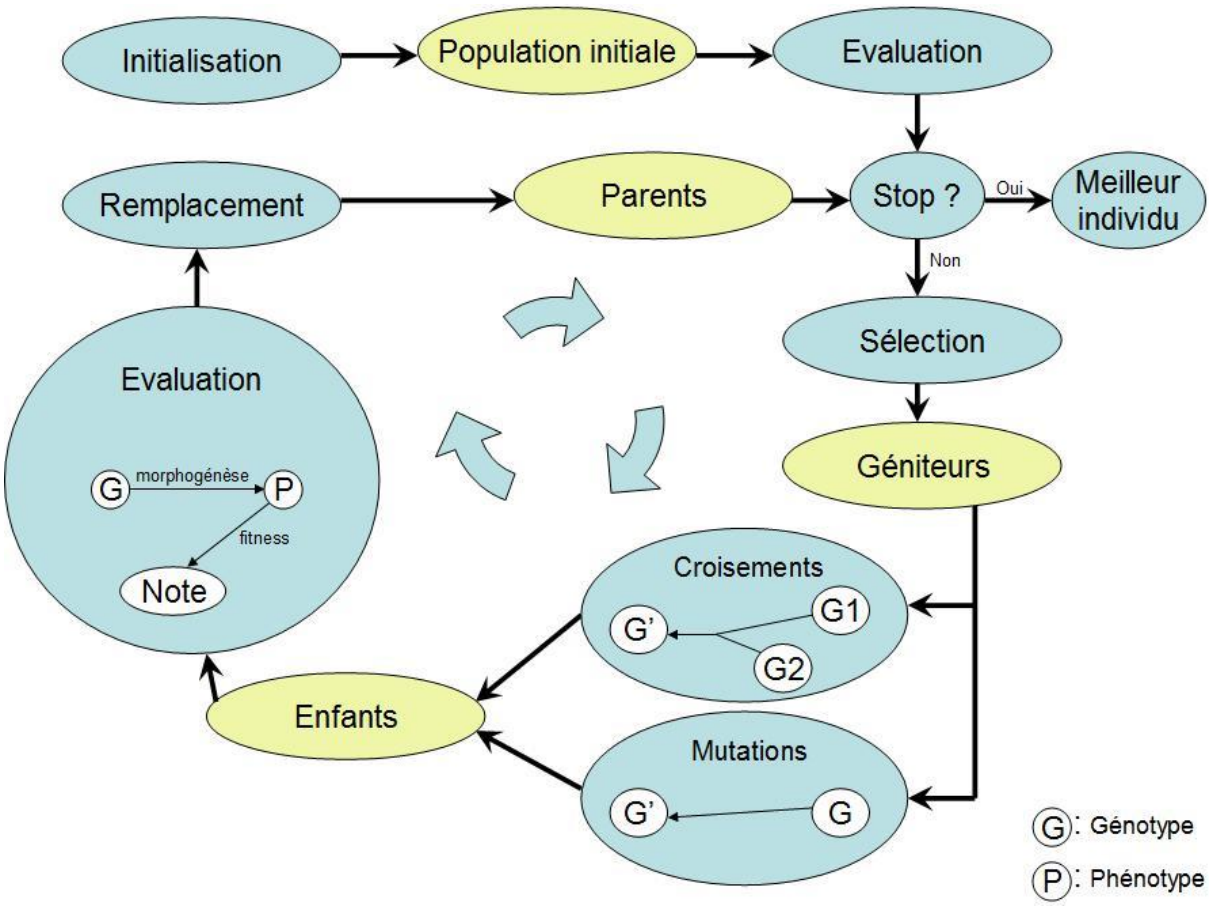


Figure 1.2. Les étapes d'un algorithme génétique [DUT10].

3. Modèles d'entités virtuelles

L'environnement d'une créature artificielle peut être simulé par un moteur physique ou un espace réel dans le cadre de robots. Les deux composantes principales sont la morphologie de la créature et son contrôleur qui lui permet d'interagir avec son environnement par le biais de capteurs et d'effecteurs.

Récemment beaucoup de travaux sur les créatures artificielles ont été réalisés, ces travaux se sont souvent focalisés sur la morphologie des créatures ou le comportement d'une ou deux créatures.

Il y a plusieurs méthodes pour représenter les morphologies et les contrôleurs des créatures artificielles. Jusqu'à maintenant les créatures de Karl Sims restent certainement les plus évoluées [SIM94a] [SIM94b]. Ce dernier a réalisé une évolution avec peu de contraintes et obtient ainsi une grande variété de créatures capables de se déplacer. Pour engendrer la morphologie de ces créatures, il utilise des *graftals*.

Par la suite beaucoup de travaux ont tenté de reproduire les résultats de Sims avec différentes approches de représentation de la morphologie et du contrôleur, tel que les *L-sytems* pour les créatures d'Hornby [HOR01]. Son projet, tout comme le projet Golem de Lipson [LIP00], a pour objectif de réaliser des robots sans la moindre intervention humaine. Pour cela, il a utilisé des mécanismes d'évolution afin de trouver la morphologie et le contrôleur de robots. Les éléments sont ensuite produits par une imprimante 3D qui les imprime couche par couche à l'aide de collagène. L'homme n'intervient sur le processus de fabrication qu'à la fin où il assemble les moteurs aux structures imprimées.

Shim en 2003 [SHI03] a simulé des créatures volantes en évoluant les ailes et le contrôleur de créatures *« coévolution »* afin qu'elles puissent suivre un chemin.

Miconi [MIC06], dans ses travaux, a reproduit les créatures de Sims sans obtenir de meilleurs résultats, mais il a par contre amélioré les algorithmes de coévolution.

3.1. Les premières utilisations de la vie artificielle

En 1991, Langton suggère d'utiliser la programmation génétique afin de générer les comportements de base des robots [MIC06][LAN92]. La même année, des résultats prometteurs avaient été obtenus par Koza [KOZ91] en appliquant la programmation génétique au langage Lisp.

Cependant, l'évolution de programmes pour des robots nécessite de nombreuses évaluations qui prennent beaucoup de temps. Pour corriger cela, il est possible de simuler les évaluations ce qui porte néanmoins d'autres problèmes. En effet, en premier lieu, en risque de perdre du temps générant des comportements simulés qui ne correspondent pas à la réalité. Le deuxième problème est qu'il est difficile et même impossible d'avoir une simulation qui reflète la réalité avec précision [BRO92]. Les environnements simulés sont plus simples et trop parfaits en comparaison avec la réalité dont les mesures sont toujours approximatives même dans des environnements stables [LAS08]. Pour le développement, Brooks [BRO92]

propose de réduire l'espace de recherche en facilitant l'apparition de structures naturelles telles que la symétrie et la modularité et de faire évoluer les programmes de manière incrémentale à mesure que des solutions viables apparaissent. Enfin Brooks propose au cours des générations de valider les résultats simulés sur un vrai robot cela afin d'orienter les recherches dans les bonnes directions et d'être en mesure au final d'obtenir les solutions souhaitées.

Rodney A. Brooks, en 1992, l'un des premiers à s'intéresser à l'utilisation de l'approche vie artificielle pour des robots réels [BRO92]. A cette époque les comportements des robots étaient programmés à la main. A la même période, Langton organise les premières conférences sur la vie artificielle où les programmes évolutionnistes générant des contrôleurs d'entités virtuelles étaient présentés [LAN89][HEU94]**.** Par la suite, les premiers travaux donnant des résultats sur les créatures artificielles à Sims [SIM94a][SIM94b], qui, sur une « *Connection Machine »*, est arrivé à générer par évolution des créatures adaptées à se déplacer dans un environnement virtuel.

4. La morphologie et le comportement des créatures artificielles

La simulation comportementale a pour but de donner les agents un comportement. Il est nécessaire que l'agent soit capable de percevoir son environnement où il évolue. Ces perceptions sont liées à sa morphologie et son intelligence doit lui permettre de s'adapter à l'environnement, et selon Jeff Bongard et Rolf Pfeifer dans leur livre *"How the Body Shapes the Way We Think: A New View of Intelligence"* [BON06], il est primordial qu'il évolue en même temps. Dans cet esprit, nous nous intéressons au cours de ce travail à l'évolution conjointe de la morphologie et du contrôleur afin que les robots modulaires puissent adapter à leur environnement. Dans les sections suivantes, on va présenter quelques concepts liés à la morphologie et les contrôleurs.

4.1. La morphologie

La modélisation et la réalisation de la morphologie des créatures artificielles sous forme de parties élémentaires, dont ses interactions font émerger une créature, posent la question comment assembler ces parties. En s'inspirant de la nature, l'approche évolutionniste apporte une solution et grâce à la sélection naturelle, elle permet de produire des créatures en utilisant des techniques de développement comme les L-systems, l'embryogenèse ou les graphes que nous verrons dans ce travail.

4.1.1. Représentation de la morphologie

Il est recommandé de ne pas utiliser une représentation directe de la morphologie des créatures. Comme les espèces que l'on retrouve dans la nature et qui comportent des milliards de cellules, Il est préférable que le phénotype ne soit pas une représentation directe du génotype. De ce fait la représentation génétique pourra être modifiée et évoluer, ce qui apporte des avantages tels que la réutilisation de certains composants ainsi qu'une compression de la représentation. [LAS08]

4.1.2. Les différents types de représentation de la morphologie

Il y a plusieurs solutions pour décrire la morphologie qui peut-être un graphe comme dans les travaux de Karl Sims [SIM94a] [SIM94b] (Figure 1.3.), ou l'utilisation d'une grammaire contexte free tel que les L-systems. Les L-systems context free peuvent avoir un haut degré de connectivité. Ce qui est moins souhaitable pour créer une créature se déplaçant. Cela est dû au fait que les L-systems peuvent imposer des contraintes de forte connexion entre les blocks. Si les contraintes sont trop fortes, la probabilité de générer une créature pouvant se déplacer devient donc faible [LAS08]. De ce fait, il faut utiliser une représentation qui évite de générer des contraintes morphologiques.

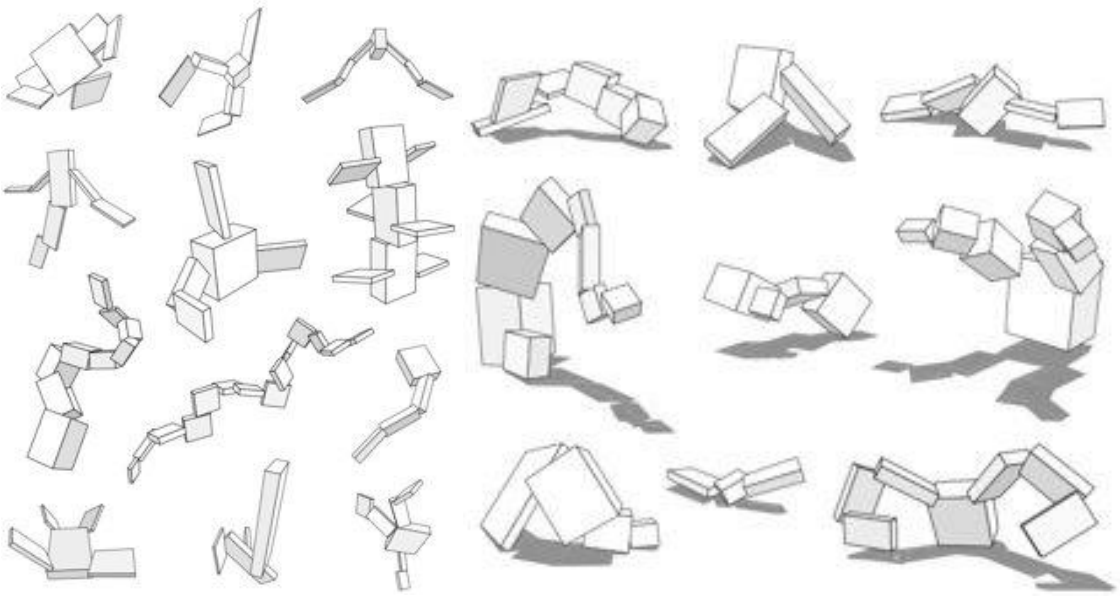


Figure 1.3. Créatures de Karl Sims [SIM94a] pouvant se déplacer, nager, ramper et sauter. Sims obtient une large variété de créatures ayant des stratégies de déplacement t'es originales. En comparaison avec la symétrie, la récursivité sur un même bloc est peu présente dans les créatures générées.

a. Représentation par arbre

Une représentation par arbre de Lipson [LIP04] permet une construction du phénotype de manière top-down ou bottom-up. L'approche top-down part de la racine et descend jusqu'aux feuilles en développant le génotype. Les nœuds parents sont donc construits avant leurs fils. Cette méthode part donc d'une structure embryogénique qui est enrichie par exemple par deux opérateurs D et T (Figure 1.4.(b, c)). L'opérateur crée un nouveau nœud et le connecte à un ensemble de points par la création de nouveaux tubes. L'opérateur T crée un nouveau point au milieu d'un tube existant et relie ce point à un autre point existant localement proche. L'application de ces opérateurs est montrée par (Figure 1.4.d). La structure se développe à partir d'une structure de base (Figure 1.5.a). La construction de la même structure est présentée (Figure 1.5.c) mais en utilisant l'approche bottom-up. C'est à dire que la structure est assemblée à partir d'éléments au lieu de se développer.

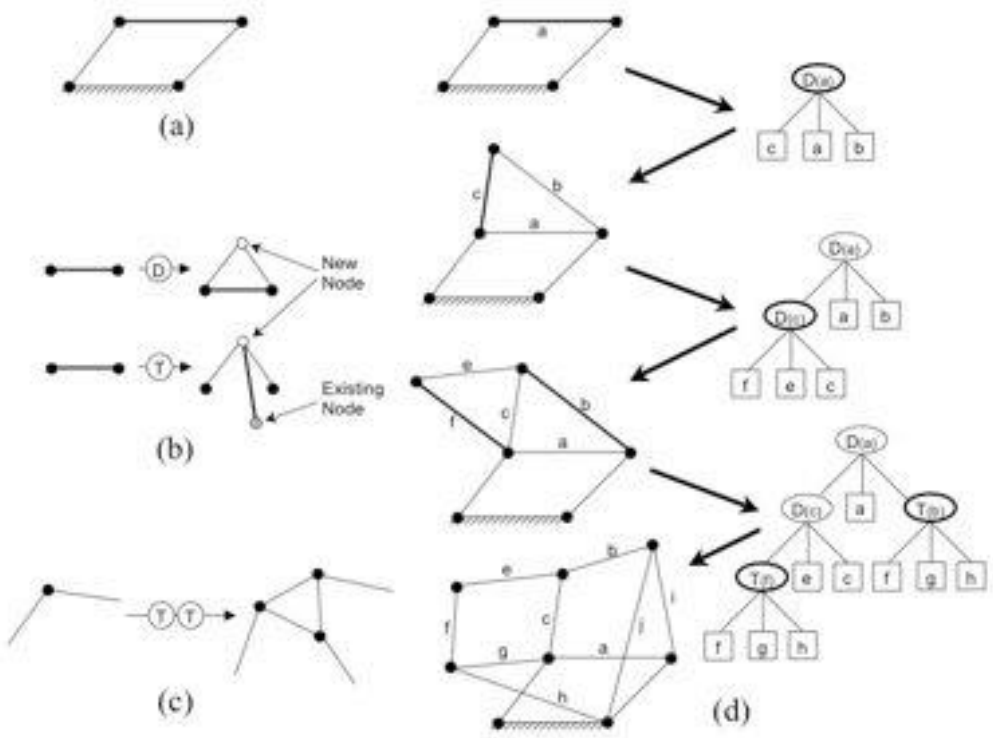


Figure 1.4. L'approche top-down permet à l'aide d'opérateur le développement d'une structure. En effet, la structure se développe en parcourant un arbre de sa racine (top) jusqu'à ses feuilles qui donnent plus de détails (down) (a) Structure de base (top). (b-c) Description des opérateurs. (d) Séquence montrant l'évolution de la construction d'une structure à partir de son embryon. Les détails de la structure augmentent plus on descend dans l'arbre, d'où le nom approche top-down. [LAS08]

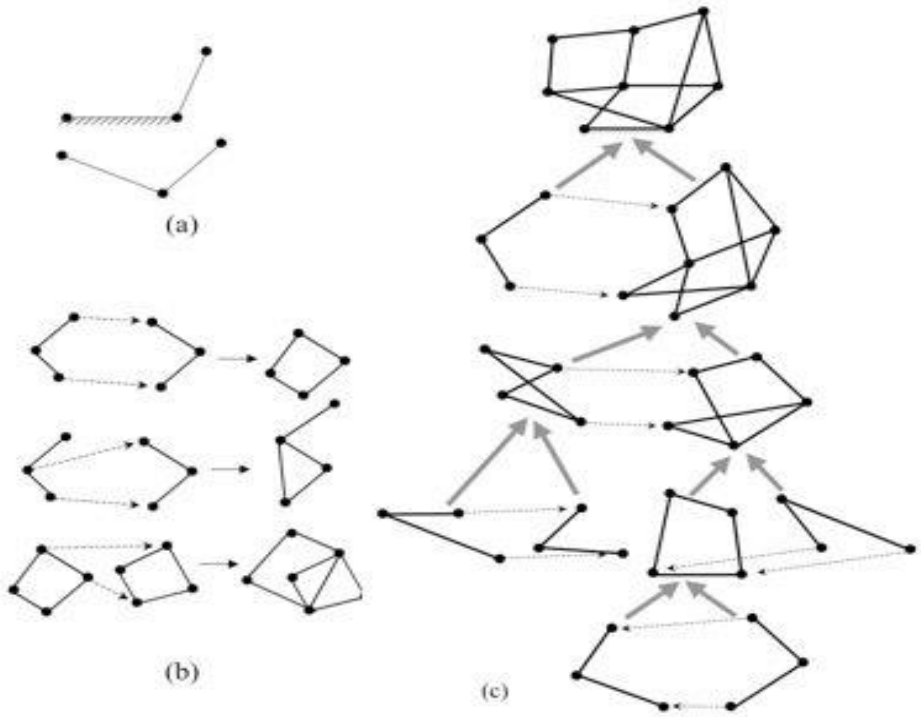


Figure 1.5. L'approche bottom-up est utilisée pour construire la même structure qu'à la figure précédente (Figure 1.4). En ici, au lieu de développer la structure en le détaillant, elle est assemblée à partir d'éléments. [LAS08]

b. Représentation par développement

On peut aussi développer la morphologie d'une créature en appliquant un ensemble de règles context-free à un axiome de départ. Ce type d'approche est similaire aux L-systems ou aux automates cellulaires. Elle peut être aussi bien appliquée à la morphologie qu'au développement d'un contrôleur. Deux règles très simples et un axiome de départ peuvent rapidement donner une structure très complexe.

4.2. Les contrôleurs

Le contrôleur d'une créature est lié à sa morphologie et à ses capteurs qui lui permettent de percevoir le monde. Pour émerger le contrôleur d'un assemblage d'éléments et non de la conception et l'expérience humaine, nous pouvons nous inspirer de la nature. Cette approche peut s'appliquer aux réseaux de neurones, en effet, l'assemblage de neurones permet de constituer une topologie dont un comportement pourra émerger. Les réseaux de neurones sont largement répandus dans l'ensemble des travaux traitant les créatures artificielles et les robots modulaires. Ils sont très performants lorsqu'il s'agit d'approcher une fonction mathématique et ils peuvent facilement composer des fonctions sinusoïdales pour produire un signal qui actionne les effecteurs des créatures. Leur utilisation est aussi due au fait que Karl Sims les a employés et que beaucoup de travaux ont pour objectif de refaire ses créatures ou de s'en inspirer.

5. L'évolution des créatures artificielles

Dans cet axe de recherche, les travaux s'intéressent principalement à l'évolution de la morphologie, ou des contrôleurs pour une morphologie fixe, ou la morphologie et les contrôleurs, afin d'évoluer soit des créatures artificielles, soit des robots modulaires. Dans les prochaines sections on va présenter ses trois domaines.

5.1. Evolution de la morphologie seule

L'évolution de la morphologie seulement n'est pas un axe de recherche assez riche, puisque il n'ajout presque aucun dimension à la simulation comportementale et en dehors, des travaux sur les plantes artificielles, il y a peu de recherches où l'on fait évoluer seulement une morphologie sans lui donner un comportement. Certains travaux en embryogenèse se consacrent uniquement au processus de développement des formes comme ceux de Chavoya, Doursatet Fleischer [CHA07]; [DOU07] ; [FLE96].

5.2. Evolution du contrôleur pour une morphologie fixe

En 1990, McKenna et Zeltzer sont parmi les premiers à réaliser un contrôleur pour une entité évoluant dans un environnement physique [MCK90]. Par la suite, de nombreux travaux sur l'évolution de contrôleurs pour une morphologie fixe ont apparu. Ces travaux ont ensuite inspiré Nolfi and Floreano qui travaillaient sur des contrôleurs pour des robots à roues [NOL94]. En parallèle de ces travaux, Bongard et Lipson [BON04] fabriquent des contrôleurs pour des créatures à pattes. Ici, il n'est pas question d'imiter la marche mais de trouver par évolution un contrôleur qui permette à une créature de se déplacer (Figure. 1.6.). La créature

évolue dans un environnement physique réaliste. Elle possède quatre pattes munies de capteurs de contact pour détecter le sol, de capteurs d'angle au niveau des articulations et de deux capteurs de phéromones. Le comportement de la créature est contrôlé par un réseau de neurones dont la topologie et le poids évoluent. Au bout de cinquante générations, le contrôleur produit arrive à contrôler la créature pour qu'elle se déplace en ligne droite ou si on le désire pour qu'elle suive la piste de phéromones. L'évolution a donc produit un contrôleur avec deux fonctions indépendantes.

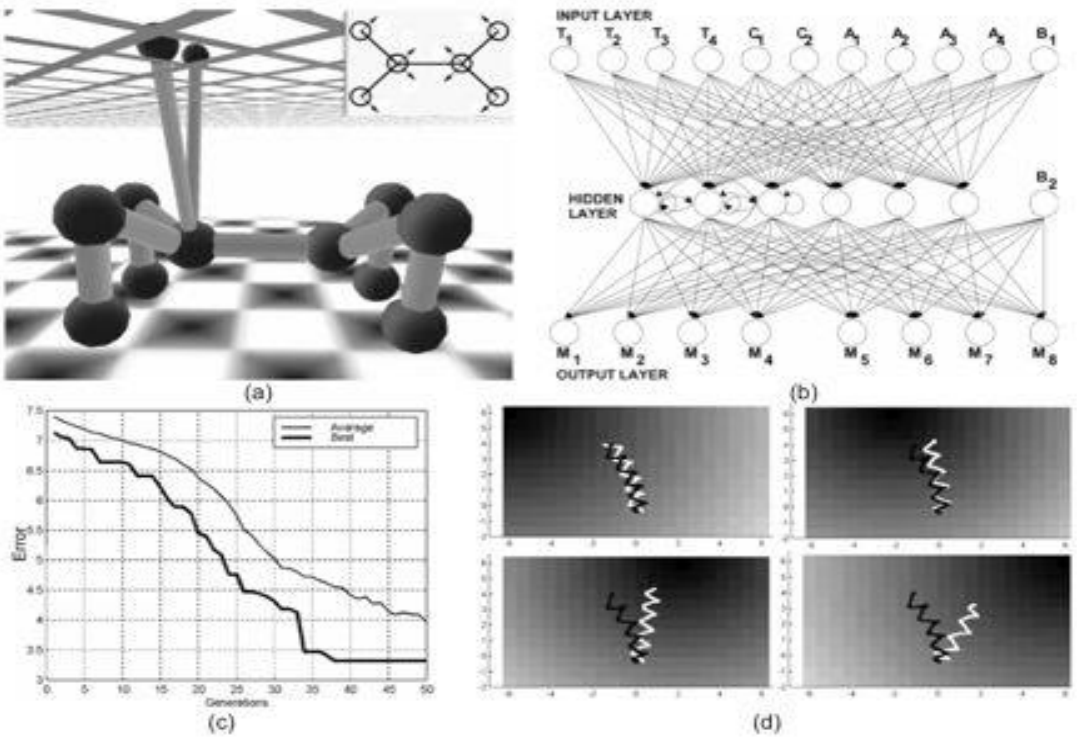


Figure 1.6. Travaux de Lipson et Bongard [BON04]. (a) Morphologie de la créature comportant quatre pattes, huit actionneurs et un capteur de contact par patte. (b) Réseau de neurones reliant les capteurs aux actionneurs. (c) La progression de l'erreur (distance qui sépare la créature de la plus forte concentration de phéromone) aux travers des générations. (d) Affichage de la concentration en phéromone (dégradés de gris). Le chemin en blanc est le trajet suivi par la créature quand les capteurs de phéromones sont actifs et le trajet est en noir quand les capteurs sont désactives.

5.3. Les inconvénients d'une morphologie figée

Afin de voir quel était l'impact de l'évolution de la morphologie sur l'évolution des contrôleurs, Paul C. and J. C. Bongard propose une comparaison entre une simulation où la morphologie est figée avec une où elle évolue [PAU01]. Dans cette simulation, un robot bipède muni de six actionneurs, deux capteurs de contact au niveau des pieds et des capteurs d'angle pour chaque articulation est modélisé dans un environnement physique simulé. Le but est de faire évoluer le contrôleur, un réseau de neurones, afin qu'il maximise la plus grande distance parcourue en un temps fini. Dans la simulation où la morphologie est figée seul le contrôleur évolue. Dans celle où la morphologie n'est pas figée, on fait évoluer la répartition des masses du robot. Lors de l'expérience, les simulations comprennent trois cents contrôleurs qu'on laisse évoluer sur une période de trois cents générations. À la fin des expérimentations,

la simulation où la morphologie évolue montre des résultats nettement supérieurs (Fig. 1.7). Les auteurs en concluent que l'évolution conjointe de la morphologie et du contrôleur donne de meilleurs résultats. En effet, figer la morphologie restreint l'espace des solutions ce qui contraint l'adaptation du contrôleur. De plus, rien ne garantit que la morphologie d'un robot soit adaptée pour la marche. Il y a donc un risque de faire évoluer une simulation pour un but que l'on ne peut pas atteindre. [LAS08]

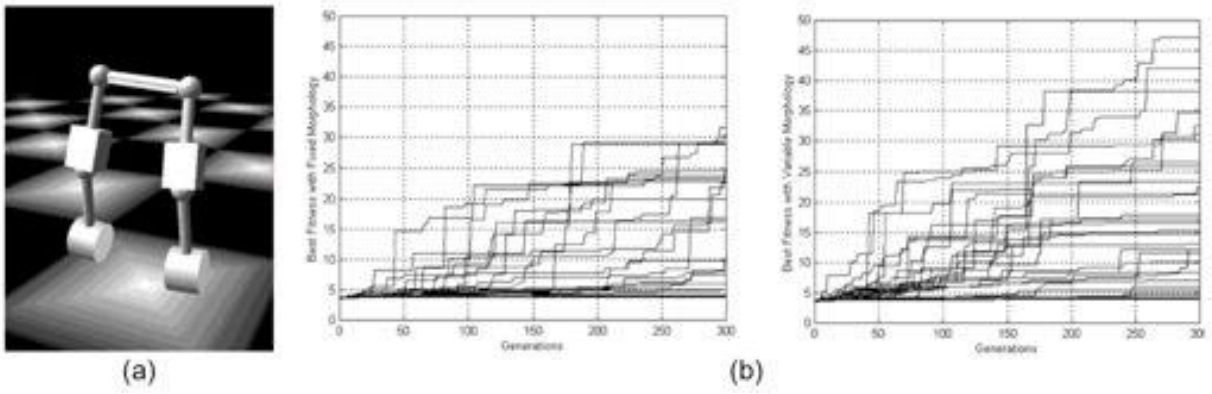


Figure 1.7. Evolution d'un contrôleur pour un robot bipède simulé [PAU01] (a) Représentation de la morphologie du robot dans le simulateur physique. (b) Comparaison des fonctions fitness au cours des générations. A gauche la morphologie est figée, à droite la répartition des masses évolue. On voit clairement que l'évolution de la morphologie permet une meilleure convergence de la fonction fitness.

5.4. Evolution de la morphologie et du contrôleur

Nous avons décrit les différents mécanismes d'évolution servant pour générer des créatures artificielles. Nous avons présenté les différentes représentations de ces créatures ainsi que les raisons pour lesquelles il est utile de faire évoluer conjointement la morphologie et le contrôleur, l'un des travaux de référence sur les créatures artificielles issues de cette approche est celui de Karl Sims.

Les créatures de Karl Sims [SIM94a][SIM94b] restent parmi les meilleurs travaux les plus achevés du domaine. En réalité virtuel, c'est difficile d'avoir une animation réaliste gardant le contrôle, si on définit tous les angles et mouvements d'un squelette. Au contraire, si on a une simulation réaliste où on définit les forces appliquées et où on calcule les lois de la physique, on aura un mouvement réaliste mais on en aura moins le contrôle. Dans ce cadre-là et en essayant de résoudre ce problème, Karl Sim sa proposé d'utiliser les algorithmes génétiques afin de générer la morphologie et le comportement de créatures artificielles. Ila utilisé une fonction de fitness afin d'automatiser l'évaluation des créatures. L'utilisateur garde tout de même un certain contrôle en définissant la fonction de fitness.

Pour la morphologie de ces créatures, il s'inspire des fractales, des L-systems et des graftals. Le génotype des créatures est représenté par un graphe orienté. Les nœuds contiennent les informations décrivant les différentes parties rigides du corps. Le graphe ne correspond pas directement au corps de la créature mais à son processus de développement (Figure. 1.8.). La construction de la morphologie se développe par récursivité à partir de la racine du graphe. Les parents peuvent avoir plusieurs connexions vers le même fils. Les

nœuds contiennent la description des blocs à savoir leur taille, le type de liaison qui les relie à leurs parents. Les types de liaisons peuvent être rigide, universal, revolute, bend-twist, twist-bend ou spherical. Il ne donne pas les définitions exactes de ce qu'il entend par bend-twist mais on peut l'interpréter comment étant une liaison résolut associée avec une liaison twist. Les blocs contiennent aussi les neurones associes qui contrôlent les articulations. Les connections entre les blocs indiquent le placement du bloc relativement à son bloc parent, la symétrie, l'échelle et l'orientation.

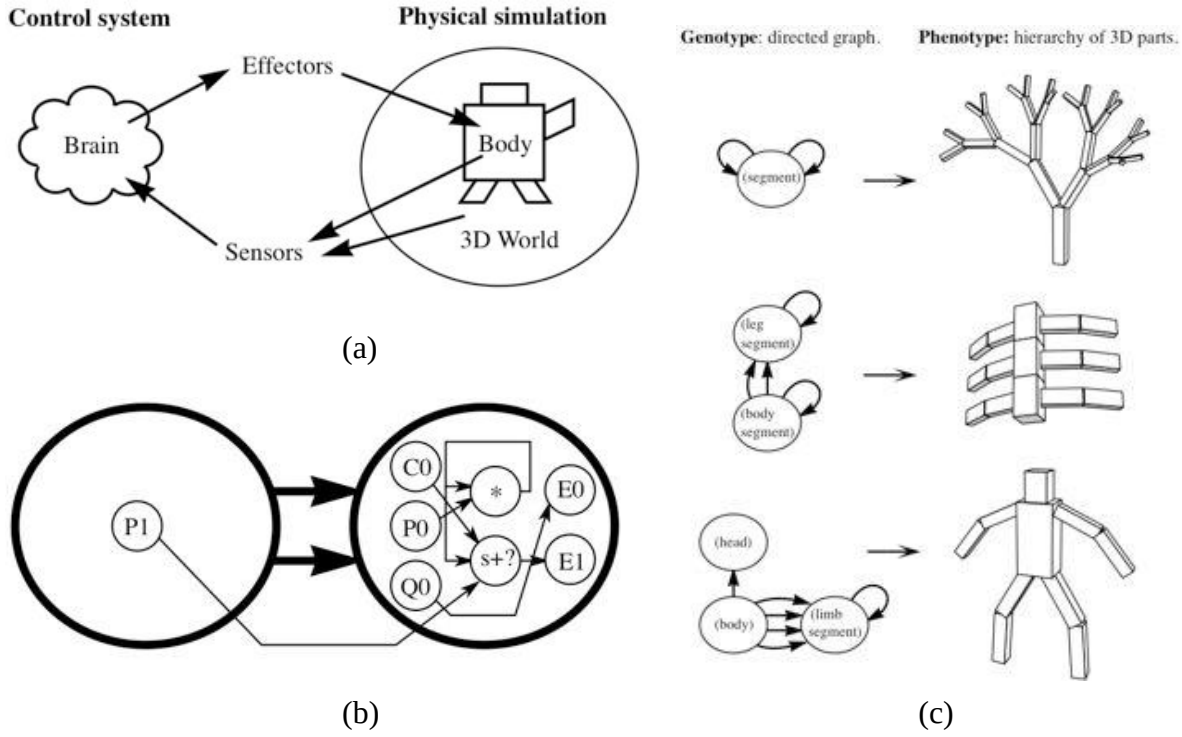


Figure 1.8. Créatures de Karl Sims [SIM94b]. (a) Simulation comportementale : Cycle entre le contrôleur, le corps de la créature et l'environnement. (b) Exemple de neurones à l'intérieur de nœuds d'un graftal. (c) Morphologie et processus de développement des créatures de Karl Sims.

Le contrôle s'effectue grâce à des capteurs qui mesurent différents aspects du monde tel que le contact entre deux surfaces, l'angle entre deux blocs. Un réseau de neurones interne composé de diverses fonctions permet de moduler le signal (Figure. 1.8). Chaque bloc possède un réseau de neurones, les réseaux de neurones sont reliés entre eux par un réseau de neurones qui supervise le comportement de la créature. Chaque neurone représente une fonction mathématique telle que des fonctions sinusoïdales qui modifient le signal.

L'environnement 3D, dont la gravitation et la détection de collisions sont simulées, est constitué d'un plan 2D sur lequel les créatures évoluent. Au début de la simulation, une population est initialisée aléatoirement. Après évaluation des phénotypes par la fonction d'évaluation, les créatures sont sélectionnées pour ensuite être reproduites par croisement ou subir des mutations. Il s'en suit une nouvelle génération et l'algorithme recommence jusqu'à que les créatures satisfassent la fonction d'évaluation ou atteignent un nombre limité d'itérations.

Durant la simulation les créatures sont évaluées pour une tache spécifique dans l'environnement 3D. Après plusieurs heures de calculs et plusieurs simulations sur une *Connection Machine*, les résultats obtenus sont les suivants : l'évolution a sélectionné différents individus pouvant se mouvoir en rampant, nageant et sautant. Parmi les créatures les plus élaborées, l'une d'entre elles peut suivre une source de lumière en nageant. Ainsi les résultats obtenus offrent une grande diversité de créatures ayant des stratégies très différentes. Ce qui n'influe pas sur les mouvements qui sont très réalistes et bien qu'il n'y ait pas de mécanisme de minimisation de l'énergie. En effet les créatures peuvent se déplacer en effectuant les mouvements qu'elles désirent sans limitation d'énergie.

En perspective de ces travaux, Karl Sims propose d'améliorer ses créatures afin qu'elles interagissent avec des environnements plus complexes. Il énonce aussi la possibilité de construire des robots à partir des créatures les plus évoluées. C'est ce que Lipson et Pollack réalisèrent avec le projet Golem [LIP00].

Les contrôleurs de Karl Sims génèrent moins de bruits car ils sont composés de sinusoïdes. De plus les morphologies et les stratégies comportementales des créatures de Karl Sims restent les plus élaborées. Ce qui aujourd'hui donne encore le plus d'importance aux travaux de Karl Sims est que malgré les tentatives récentes, personne n'a reproduit ces résultats.

6. Les Algorithmes génétiques Co-Evolutionnaire (ACE)

Ceux sont une classe des algorithmes évolutionnaires dans lesquels l'évaluation de fitness est basée sur les interactions entre plusieurs individus.

Les ACEs sont très similaires aux méthodes traditionnelles d'évaluation où :

- Les individus codent de solutions potentielles ;
- Ils sont modifiés pendant la recherche avec les opérateurs génétiques ;
- La recherche est dirigée par la sélection basée sur la fitness.

Mais ils se différent de manière fondamentale où:

- L'évaluation nécessite une interaction entre plusieurs individus ;
- Les individus interagissant appartiennent à la même population ou à des populations différentes ;
- Évoque les notions de coopération et de concurrence :

6.1. La co-évolution compétitive

Algorithmes compétitifs sont ceux dans lesquels les individus à réussir au détriment d'autres individus.

La co-évolution compétitive est une situation où deux espèces différentes co-évolve un contre l'autre, comme par exemple la situation de : Proie-Prédateur et Hôte-Parasite. La Fitness de chaque espèce dépend de la fitness des espèces adversaire ce qui peut augmenter

l'adaptabilité des individus et permettre l'émergence progressive de solutions plus complexes comme chaque population essaie de gagner sur l'adversaire. Comme, le changement continu de fitness peut aider à prévenir la stagnation aux locaux minima [HIL90].

6.2. La co-évolution coopérative

La co-évolution coopérative [POT97] est un paradigme récent dans le domaine des algorithmes évolutionnaires, qui divise un grand problème en sous-composants « sous-espèces » qui doivent être résolus de façon indépendante afin de résoudre le grand problème. Les sous-composants sont mis en œuvre en tant que sous-populations et interagissent d'une façon coopérative.

Les algorithmes co-évolutionnaires coopératifs, qui sont présentés par Potter et De Jong [POT00], modélisent un écosystème composé de deux ou plusieurs espèces. Comme dans la nature, les espèces sont génétiquement isolées, ce qui signifie que les individus ne s'accouplent qu'avec les autres membres de leur espèce. Chaque espèce est évoluée dans sa propre population et s'adapte à l'environnement par l'application répétée d'un algorithme évolutionnaire. Les espèces interagissent avec les autres dans un domaine partagé et avoir une relation de coopération ce qui implique que les individus sont notés sur leur coopération et réussissent ou échouent ensemble.

7. Conclusion

La vie artificielle est un champ de recherche interdisciplinaire alliant l'informatique et la biologie, mais avec des applications dans des domaines variés et un objectif de créer des systèmes artificiels s'inspirant des systèmes vivants, soit sous la forme de programmes, ou soit sous la forme de robots. Donc, la vie artificielle est le début de la robotique modulaire et le cadre général de toutes les méthodes utilisées pour l'évolution d'un système robotique modulaire.

Dans ce chapitre introductif, nous avons présenté les techniques utilisées dans le domaine de vie artificielle que nous les utiliserons dans notre système robotique. Que signifie un robot modulaire ? Quelles sont ses caractéristiques et ses domaines d'applications ? Les réponses à ces questions seront présentées dans le chapitre suivant.

CHAPITRE 2

La Robotique Modulaire

"Design can be art. Design can be aesthetics. Design is so simple, that's why it is so complicated."

— Paul Rand

1. Introduction

L'idée d'un système robotique auto-reconfigurables, qui vise à changer sa connectivité pour s'adapter à une tâche, est très intéressante. Cette vue se situe entre l'ambition de l'être humain et ses capacités technologique courantes ; imaginaire si en regardant les films de la science-fiction (par exemple : robots morphing, robots métal-liquide, nano-robots, etc.), réaliste en examinant les recherches sur les robots auto-reconfigurables qui se développent dans le monde entier, attirant des chercheurs actifs et gagnant l'intérêt public.

Dans ce chapitre-là, et dans les sections suivantes, on va présenter les robots modulaires en donnant quelques concepts de base, leurs différents types, les caractéristiques d'un système robotique modulaire, ses applications en concluant par une évaluation de ses avantages et ses inconvénients.

2. Les débuts de robots modulaires

2.1. De la révolution industrielle aux manipulateurs de robot

Les robots auto-reconfigurables et les robots en général sont probablement mieux compris dans le contexte de la révolution industrielle. La révolution industrielle a commencé vers 1733 dans l'industrie textile avec des inventions comme la machine à filer de Sir Richard Arkwright ; la première machine à filer automatique. Au cours du XVIIIe siècle, de nombreuses autres machines ont été inventées et la révolution s'est accélérée. Elle a en outre été renforcée par la machine à vapeur de James Watt, introduit en 1769.

Après l'invention de la roue hélicoïdale, introduit par Ramsdan en 1774, la précision des machines a été augmentée et étendue à d'autres industries. L'amélioration de la précision signifiait aussi que le concept de pièces de rechange est devenu possible parce que les machines avaient maintenant assez grande précision. Cela a ouvert la voie à la production de masse, qui a été affiné au cours du XIXe siècle dans l'usine d'automobile de Henry Ford en 1908, qui a lancé la première chaîne de montage de courroie transporteuse.

Sur la ligne d'assemblage, les tâches ont été coupés en petites tâches simples qui un ouvrier non qualifié pourrait faire rapidement et efficacement. La simplification de ces tâches a permis d'introduire la prochaine génération de machines : les robots manipulateurs. *General Motors* a présenté l'Unimation 1900 dans leur ligne d'assemblage de voitures en 1961. L'Unimation Inc., fondée par George Devol cinq ans plus tôt, a produit ce robot. Dans les années suivantes, et encore aujourd'hui, les robots manipulateurs sont optimisés pour faire une tâche rapide et précise. Le résultat de ceci est les robots manipulateurs incroyablement rapide et précis que nous avons aujourd'hui [STO10].

2.2. Les robots dans la fiction

En 1920, l'auteur Josef Capek publié sa pièce de théâtre *"robots universels de Rossum,"* dans laquelle il décrit les esclaves de la machine, c'est à dire, des robots, qui aideraient l'humanité. Josef Capek ne se prévoyait pas que son idée fasse la lumière qui guide et inspire de nombreux ingénieurs et scientifiques pour le reste du XXe siècle et du XXIe. La technologie était à l'époque pas assez mature pour même essayer de réaliser ce rêve.

Isaac Asimov était le prochain à élaborer les robots avec ses fameuses histoires courtes *"Robbie"* en 1940 et *"Runaround"* (Cycle fermé) en 1942 (deux histoires courtes peuvent être trouvées dans le livre 1950 intitulé : I, Robot). L'écriture d'Asimov sur les robots considéré principalement les aspects éthiques de robots et il a également formulé les trois lois connus de la robotique (à partir de "Runaround"):

- **Première Loi :** Un robot ne doit pas porter atteinte à un être humain ni, en restant passif, laisser cet être humain exposé au danger.
- **Deuxième Loi :** Un robot doit obéir aux ordres donnés par un être humain sauf si de tels ordres entrent en contradiction avec la *Première Loi*.
- **Troisième Loi :** Un robot doit chercher à protéger son existence dans la mesure où cette protection n'entre pas en contradiction avec la *Première Loi* ou la *Deuxième Loi*.

Après cela, les robots étaient un thème récurrent dans la science-fiction. Cependant, pour les robots auto-reconfigurables, les débuts étaient dans les films *Terminator 2: Judgment Day (1991), The Matrix Revolutions (2003),* et la série de transformateurs livres, dessins animés et de films, culminant dans le film *Transformers (2007).* Tous ces films présentent des robots qui pourraient modifier automatiquement leurs propres formes. Peut-être que certaines de ces œuvres de fiction ont été la source d'inspiration pour *Toshio Fukuda* (réalisateur du système CEBOT (**CE**llulairero**BOT**) [FUK88]), qui, entre autres, créé le fondement philosophique pour le domaine de robots auto-reconfigurables que nous connaissons aujourd'hui [STO10].

3. Robots Modulaires Auto-reconfigurables

Un robot modulaire est constitué de modules assemblés de manières diverses, offrant une grande diversité de robots à partir d'un nombre relativement restreint de modules de base. Typiquement, un robot se compose de dix à cent modules, mais des robots constitués de milliers de modules ont été déjà simulées. Chaque module est un simple robot contenant tous les composants embarqués nécessaires pour créer un robot: actionneurs, capteurs, batteries et un processeur. En outre, un module peut communiquer avec les autres modules.

En outre, si le robot est capable de se reconfigurer par lui-même on parle alors d'un système auto-reconfigurable. Un tel système a la capacité de contrôler l'état de ses connecteurs et capable de s'auto-manipuler, c'est à dire de se déplacer les modules qui le constituent afin de modifier sa morphologie. La figure 2.1.a et 2.1.b montre le système M-TRAN [MUR07] et le système ATRON dans différentes phases de reconfiguration, respectivement.

Même s'il est toujours difficile de capturer un concept comme celui de robots auto-reconfigurables, surtout en tenant en compte l'évolution rapide de ce domaine, nous donnons une définition inspirée par E. H. Ostergaard [OST04]. Les robots auto-reconfigurables sont des robots qui satisfont les critères suivants:

- **Modulaire:** Le robot est construit à partir de plusieurs unités physiquement indépendantes qui encapsulent une partie de la complexité de leur fonctionnalité.
- **Reconfigurable:** La reconfiguration correspond à un processus qui modifie la topologie d'un robot modulaire, par une intervention extérieure pour connecter les modules en plusieurs façons différentes en formant des robots différents en termes de taille, forme ou fonction.
- **Dynamiquement reconfigurable:** Les modules peuvent être déconnectés et connectés pendant que le robot est actif.
- **Auto-reconfigurable:** Le robot peut changer la façon dont les modules sont connectés par lui-même.

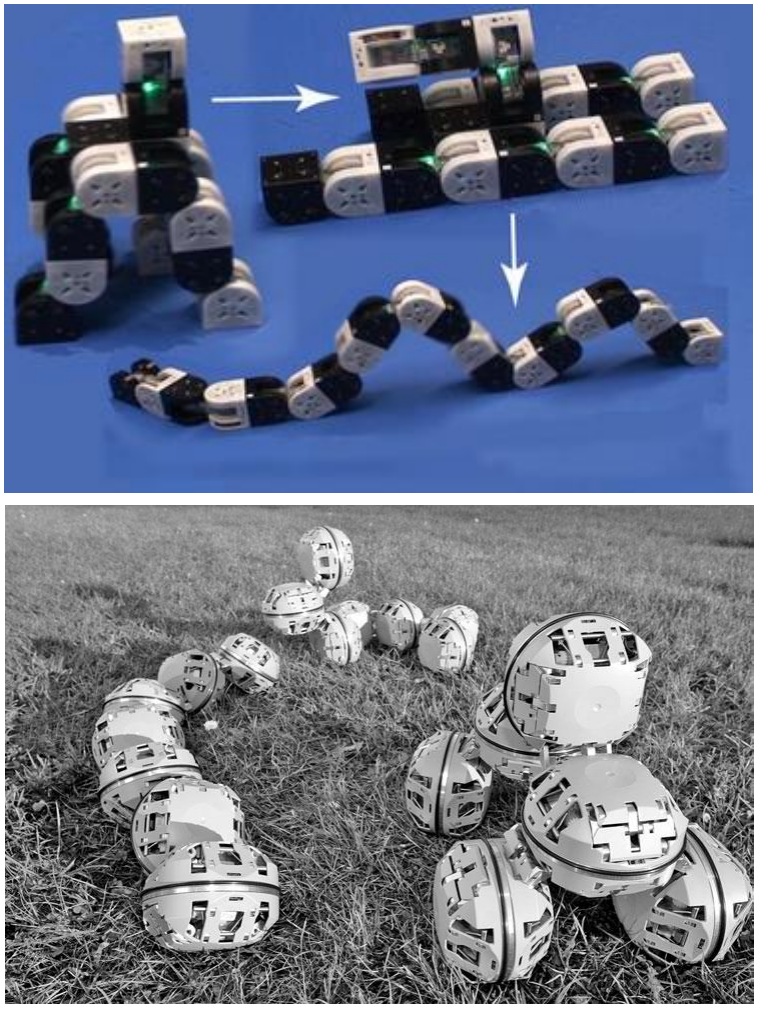


Figure. 2.1. (a) Étapes d'une auto-reconfiguration d'un robot modulaire M-TRAN [MUR07]. **(b).** Des modules ATRON connectés formant des robots comme un serpent et une voiture [STO10].

4. Concepts de Base

Classiquement, un système robotique est un dispositif mécatronique constitué d'actionneurs, de liaisons mécaniques, d'effecteurs, de capteurs, d'une architecture de commande, et d'une source d'énergie alimentant l'ensemble. L'architecture de commande coordonne les mouvements des actionneurs en fonction des signaux reçus par les capteurs externe et internes pour la réalisation d'une tâche donnée.

Dans les sections prochaines nous allons donner quelques concepts de base concernant le domaine de robotique modulaire.

4.1. Module

Un module constitue l'élément de base d'un système modulaire. Un module doit pouvoir être connecté. Il dispose donc d'un certain nombre de connecteurs (au minimum un). Si le système doit pouvoir être animé, notamment pour être auto-reconfigurable, il est nécessaire que certains modules soient pourvus de mécanismes internes permettant des déplacements relatifs entre les connecteurs. Par conséquent, en plus des connecteurs, un module possède généralement des degrés de mobilité. En outre, le module peut comporter un certain nombre de ressources, dont une liste non exhaustive est :

- des accumulateurs,
- des effecteurs,
- des systèmes de locomotion,
- des capteurs,
- des calculateurs, utilisés notamment pour la commande,
- des dispositifs de communications à distance.

4.2. Morphologie (Assemblage)

Une morphologie est un ensemble de modules interconnectés. Un module isolé est considéré comme un assemblage.

4.3. Configuration (Topologie)

La configuration d'un assemblage fait référence à la façon dont les modules d'un assemblage sont interconnectés plutôt qu'à l'assemblage lui-même.

4.4. Système, Système Modulaire ou Robot Modulaire

Ces dénominations sont équivalentes, mais peuvent avoir deux acceptions :

1. Il peut s'agir de la définition d'un ou plusieurs types de modules compatibles qui peuvent être connectés et former des assemblages.
2. Cela peut être également un ou plusieurs assemblages constitués de modules de types prédéfinis.

4.5. Connectivité

Dans un système, les différentes sortes de modules peuvent avoir des types de connecteurs différents et le nombre de connecteurs par module peut varier. La connectivité d'un module est définie par le nombre de connecteurs dont il dispose et les possibilités de connexion entre les différents types de connecteurs, en tenant compte également de leur symétrie.

De plus de ces concepts, il faut ne pas omettre la reconfiguration et l'auto-reconfiguration présenter dans la section précédente.

5. Les types de robots modulaires

Les robots modulaires peuvent être classés en trois catégories principales selon la nature des modules et leurs connections : les robots chaînés, les robots treillis (ou en réseau) et aussi les robots hybrides qui combinent les caractéristiques de robots chaînés et treillis.

5.1. Les robots chaînés « Chain type »

Les robots chaînés sont composés par des chaînes de modules connectées entres elles (avec possibilité de boucles internes) et se reconfigurent en détachant d'eux des chaînes de modules. Chaque chaîne reste pourtant connectée au robot en un ou plusieurs endroits et aucune d'entre elles ne peut se mouvoir seule comme par exemple les Molecubes [ZYK08] ou Roombots [SPR10] qui se concentrant sur l'auto-reconfiguration des robots modulaires.

Figure 2.2. (a). Module Molecubes. [ZYK08] **Figure 2.2. (b).** Assemblage de Molecubes [ZYK08].

5.2. Les robots treillis « Lattice type »

Les robots treillis (en réseau) sont des robots auto-reconfigurables dans lesquels les modules sont organisés en une structure en treillis similaire à la façon dont les atomes sont organisés dans un cristal ce qui implique que les positions que les modules peuvent prendre forment un espace discret. Autrement dit les modules se déplacent sur un treillis virtuel, comme des pièces sur un échiquier (mis à part qu'ici le treillis est en 3D). Ceci simplifie les problèmes de reconfiguration. Le plus connu des robots modulaires en treillis est ATRON développé par Jorgensen et al. [JOR04] (Jorgensen et al. 2004 [JOR04], Christensen and Stoy [CHR06], Ostergaard et al. [OST06]).

5.3. Les robots hybrides « Hybrid type »

Ceux sont des robots modulaires qui peuvent se comporter comme un robot chaîné ou comme un robot treillis en combinant les avantages des deux. On peut citer des systèmes hybrides comme M-TRAN [YOS03] et SuperBot [SHE10], et ATRON [STO10] qui peut aussi se comporter comme un robot chaîné.

Figure 2.3. Robots modulaire M-TRAN. [YOS03]

Les robots modulaires peuvent être aussi classés en trois catégories selon le nombre de modules qui composent chaque robot, malgré cette classification n'est tellement utilisée dans la littérature mais on va la mentionnée pour bien expliquer les différents caractéristiques de robots modulaires [STO10]:

- **Pack Robots (robots en pack) :**

  Pack robots sont constitués de dizaines de modules. Les modules sont généralement caractérisés par une force ; cela signifie que le module individuel est utile et peut certainement se lever, mais il est capable aussi de soulever une grande fraction des autres modules du robot. Chaque module du système joue un rôle déterminant, alors il est important qu'ils sont strictement coordonnés à travailler ensemble pour atteindre l'objectif du robot.

- **Herd Robots (robots en troupeau) :**

  Herd robots sont constitués de centaines de modules. La fonctionnalité d'un module est limitée et sa force est modérée. Cela signifie qu'un module est encore capable de se lever, mais ne peut pas être une unité fonctionnelle dans le robot lui-même. Les unités fonctionnelles sont toujours construites à partir des groupes de modules. Dans ces systèmes, il existe une redondance suffisante pour permettre une coordination moins stricte sans affecter de manière significative les performances du système, mais les modules peuvent être coordonnés globalement avec difficulté. Dans ces robots les modules travaillent généralement ensemble pour accomplir la tâche du robot, mais pas aussi bon que les robots en pack.

- **Swarm Robots (robots en essaim) :**

Ces robots sont constitués de myriades de modules. Les modules individuels sont faibles et ont une influence limitée sur le robot dans son ensemble et seulement un nombre considérable de modules est nécessaire pour créer une unité fonctionnelle dans le robot. Dans ces robots, il est impossible de contrôler les myriades de modules globalement et donc les modules doivent être autonomes à un grand degré. Ces robots agissent comme des essaims qui ne peuvent pas être contrôlés, mais vivre et se développer selon leurs propres règles, semblables à seuls d'abeilles ou de fourmis. On appelle donc ces robots : *les robots en essaim*.

6. Caractéristiques de robots modulaires

Les robots modulaires ont des caractéristiques uniques qui les rendent intéressants d'un point de vue technique, dans ce que suit on va citer un ensemble de propriétés qui peuvent être présente dans un robot modulaire.

6.1. Robuste

Grâce à leur nature modulaire, les robots auto-reconfigurables ont un degré élevé de redondance qu'ils peuvent exploiter pour devenir robuste. Une défaillance matérielle ou logiciel peut mettre un module à l'échec. Mais cela ne cause pas la défaillance du robot dans son ensemble. Par exemple, le robot peut compenser un actionneur brisé en utilisant ses actionneurs restants ou, Si un capteur défaille, il peut utiliser le reste des capteurs pour obtenir les informations nécessaires.

Nous pouvons mesurer la robustesse en mesurant la dégradation des performances en fonction des pièces défectueuses. Il est souhaitable que la performance se dégrade gracieusement au lieu de chuter rapidement avec le nombre de pannes.

6.2. Versatilité

Être versatile signifie être capable de s'adapter ou être adapté à de nombreuses fonctions ou activités différents. Un robot est versatile veut dire qu'il est capable d'effectuer de nombreuses tâches différentes dans de nombreux environnements différents. Mais que faut-il pour dire que les tâches ou les environnements sont différents, selon Stoy et al. [STO10], on considère que deux tâches ou deux environnements peuvent être différents si, pour les manipuler un robot a besoin de différents comportements ou des changements dans sa morphologie. En d'autres termes, les tâches ou les environnements sont différents si elles ne peuvent pas être manipulés en ajustant les paramètres du contrôleur du robot ou nécessite des modifications morphologiques qui sont contrôlés par ce contrôleur (par exemple, marcher et galoper sont des tâches différentes, alors que marcher et marcher rapidement ne sont pas. La même chose est vraie pour les environnements: marcher sur un sol de laboratoire n'est pas différent de marcher sur une route, mais il y a une différence entre marcher sur un sol en laboratoire et dans une dune de sable,).

La versatilité n'a pas la même importance dans toutes les applications, mais elle doit être une haute prioritaire dans les applications où la tâche peut changer ou être inconnue au moment de la conception. Cela pourrait être le cas pour une mission d'extra-planétaire ou une opération de recherche et sauvetage.

6.3. Adaptabilité

Il est également un objectif de conception de faire des robots qui sont adaptables, ce qui signifie qu'ils sont capables d'accomplir leur tâche, même si la tâche ou l'environnement change un peu. Par exemple, nous voulons faire des robots qui peuvent marcher à des vitesses différentes et sur de nombreuses surfaces différentes. Une façon d'augmenter la capacité d'adaptation d'un robot est d'exploiter sa versatilité. Toutefois, la versatilité ne garantit que le robot soit capable d'effectuer une large gamme de tâches ;elle n'indique pas la robustesse du robot aux changements de l'environnement ou de la tâche. C'est d'où vient l'idée derrière l'introduction de l'adaptabilité séparément, c'est en mettant l'accent sur l'adaptabilité nous élargissons l'utilité des comportements individuels, tandis que avec la versatilité nous assurons que le robot est équipé avec de nombreux comportements qui lui permettent d'effectuer de nombreuses tâches différentes dans des environnements différents.

L'adaptabilité dépend de la tâche et ne peut être mesurée qu'à base de la tâche. Par exemple pour une tâche de soulèvement, l'adaptabilité peut être mesurée comme la gamme de tailles et poids des objets qu'un robot peut soulever. En naviguant un terrain, elle peut être la gamme de vitesses de locomotion, et ainsi de suite.

6.4. Pas cher par rapport à leur complexité

Les modules sont assez complexes et coûteux à produire. Toutefois, un robot auto-reconfigurable se compose de nombreux modules identiques et donc le coût du module peut être réduit car ils peuvent être produits en masse.

Il est important de noter que ces caractéristiques ne sont que des caractéristiques potentielles. En théorie, il doit être possible de réaliser ces fonctionnalités. Cependant, dans la pratique, ces traits sont atteints qu'à un degré limité.

7. Les domaines d'application

Les robots modulaires sont flexibles et peuvent s'adapter à une large gamme de tâches et d'environnements, ce que les rend plus adaptés pour les tâches où les conditions de fonctionnement et les exigences de capacité ne sont pas connues ou pas bien spécifiées à priori. L'ensemble des exemples d'applications suivants illustrent certains domaines qui pourraient bénéficier du développement d'un système de MSR.

7.1. Locomotion et exploration

Lors d'une mission d'exploration, un système auto-reconfigurable peut optimiser sa morphologie en fonction du type de terrain. Par exemple, il peut adopter une morphologie de chenille pour rouler rapidement en terrain plat, ou une morphologie à pattes pour progresser

en terrain irrégulier, ou encore une morphologie serpentiforme pour évoluer dans un passage étroit comme illustré sur la figure 2.4.

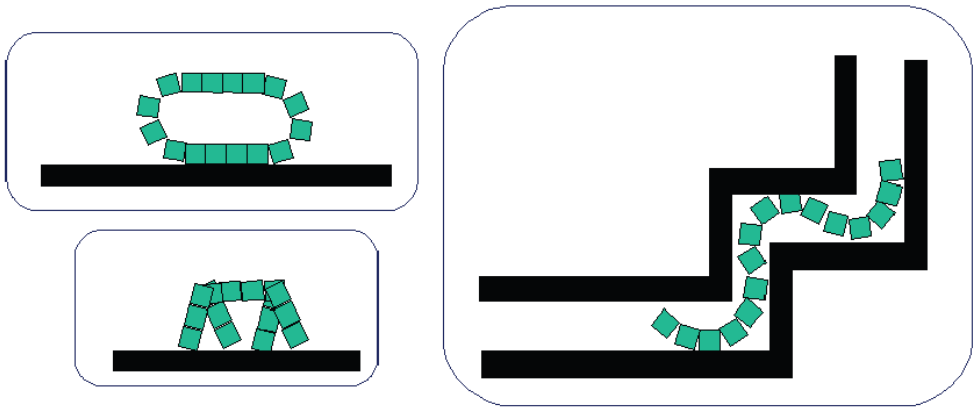


Figure 2.4. Locomotion [BRE09].

7.2. Robotique spatiale et exploration planétaire

L'exploration de l'espace et les missions spatiales présentent de nombreux défis, y compris un environnement imprévisible et la masse et le volume du matériel à envoyer, les systèmes robotiques auto-reconfigurables constituent un atout. Leur versatilité peut permettre d'envisager une multiplicité de tâches avec un minimum de matériel. Ce qui rend les robots MSR capable de résoudre les défis inattendus, tout en occupant peu d'espace et de masse.

La dégradation progressive due à l'échec est particulièrement important pour les robots opérant dans l'espace ; un dysfonctionnement d'un composant peut potentiellement conduire à l'échec de la mission. La nature redondante des systèmes MSR leur donne la possibilité d'éliminer les modules défaillants.

La figure 2.5 illustre ce principe sur un exemple complexe. Des modules cubiques du système Molecubes [YIM07] forment des manipulateurs dont les extrémités sont des modules effecteurs spécialisés, comportant des outils et des pinces. Ces pinces permettent, notamment, la saisie de modules statiques en forme de poutrelles et réalisent donc la connexion entre les Molecubes et les poutrelles.

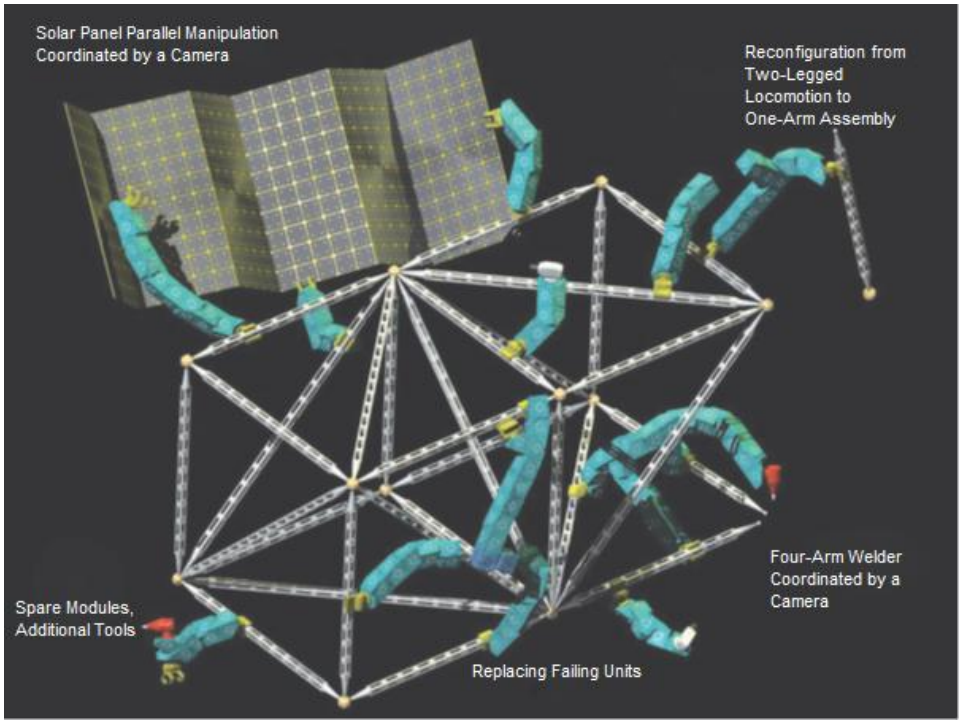


Figure 2.5. Construction spatiale [YIM07].

7.3. Construction, assemblage

L'auto-reconfiguration permet de reconfigurer un ensemble de modules en une structure déterminée. Cela peut avoir des applications intéressantes pour la construction de structures très érigées et difficilement accessibles, comme par exemple des tours ou des grues.

7.4. Jeux de construction

Les modules peuvent servir à construire des jouets animés et programmables, comme par exemple les LEGOs. De plus, ceux-ci peuvent servir de support à l'expérimentation d'un très grand nombre de morphologies et de modes de locomotion associés [BRE09].

7.5. Bucket of Stuff

Le terme *« Bucket of Stuff »* est une idée futuriste. Le système serait un produit de consommation composé d'un conteneur des modules reconfigurables qui reconfigure pour accomplir les tâches ménagères arbitraires. Cette application peut être considérée comme le but pratique général du MSR: un système qui peut s'adapter à n'importe quelle tâche en temps réel. Un seau des modules MSR pourrait être utilisé pour former la configuration souhaitée pour l'utilisateur final tel que nettoyage des gouttières au pliage de lessive [BRE09].

8. Les avantages et les inconvénients

Les robots modulaires sont comme tous les systèmes informatiques ; ont des avantages et des inconvénients, et leur conception consiste toujours d'un compromis entre ces deux-là. On va donner dans cette dernière section un bilan sur les robots modulaire.

8.1. Les avantages

Les avantages des systèmes robotiques modulaires sont nombreux, mentionnons:

- **Rapidité de déploiement et simplicité de mise en œuvre :**

  Un manipulateur humain peut assembler en quelques minutes un robot modulaire adapté à une tâche. Si on ne sait pas quel assemblage de modules convient le mieux pour une tâche donnée, un logiciel dédié doit permettre de générer une topologie adéquate en fonction de la tâche, soit en consultant une base de données soit à partir de critères d'optimisation. Les contrôleurs correspondants sont également fournis par le même logiciel et est téléchargée dans le robot.

- **Simplicité de maintenance et rapidité de réparation :**

  En raison de la généricité des modules, il est possible de pallier rapidement tout dysfonctionnement matériel en remplaçant le module défaillant dans le robot.

- **Versatilité, économie en matériel :**

  Un stock de modules de masse et de volume limités permet d'assembler un très grand nombre de robots différents. Sur un même site, la versatilité de la robotique modulaire évite de devoir stocker un modèle de robot distinct pour chaque type de tâche. En outre, il peut être impossible de prévoir toutes les tâches qui devront être exécutées sur le site. Dans ce cas un stock de modules robotiques peut permettre de dépasser cette difficulté.

- **Facilité de transport et de stockage :**

  Il est possible de conditionner et transporter les modules dans des caisses plus petites que les robots qu'ils permettent d'assembler.

- **Faible coût de production :**

  La généricité des modules peut permettre de les produire en grand nombre et ainsi d'abaisser leur coût de fabrication.

- **Granularité :**

  Comme les modules sont plus "petits" que les robots qu'ils permettent d'assembler, il est possible de gérer avec plus de finesse l'état d'un robot. L'ajout ou la suppression de modules, ou la mise à jour de certains composants peuvent se faire de façon plus progressive qu'avec des robots classiques. En outre les robots peuvent être graduellement démontés et remontés ailleurs.

En outre, les systèmes robotiques modulaires auto-reconfigurables présentent encore d'autres avantages :

- **Autonomie et robustesse :**

Le système peut se reconfigurer sans nécessiter d'intervention humaine, la reconfiguration pouvant toutefois être télécommandée. En cas de dysfonctionnement d'un module, le système peut s'auto-réparer en remplaçant le module défectueux.

- **Versatilité accrue :**

Le robot peut accomplir des missions nécessitant intrinsèquement des reconfigurations pendant leur déroulement. Par exemple, dans une tâche de locomotion où le robot optimise sa topologie en fonction des obstacles, ou progresse par reconfiguration. Ceci n'est pas possible avec les robots modulaires non auto-reconfigurables qui doivent être assemblés avant la tâche.

- **Auto-assemblage, auto-démontage, transformation, animation :**

Une structure peut être construite en s'assemblant d'elle-même à partir d'un stock de blocs modulaires. La structure peut également se démonter par elle-même, ou se modifier selon les besoins. Exemples : Un stock de modules s'auto-assemble sous forme d'une table basse puis se transforme en chaise. Une autre application est la reproduction de maquettes 3D, ou l'aspect ludique des transformations.

Par conséquent, les systèmes auto-reconfigurables sont utiles lorsque l'autonomie, la versatilité et la robustesse sont recherchées. Enfin, il existe des applications où la reconfiguration n'est pas un moyen mais un objectif.

8.2. Les inconvénients

L'inconvénient majeur des robots modulaires résulte de la généricité des modules. En effet, l'intégration d'un grand nombre de fonctions dans les modules induit une augmentation des masses et de l'encombrement et une réduction de la fiabilité par rapport à un système spécialisé.

Pour conclure, on peut dire que lorsque la spécialisation, la répétition et la performance sont recherchées, les systèmes modulaires ne sont pas forcément compétitifs. En revanche, si les tâches sont multiples et/ou imprévisibles la robotique modulaire est beaucoup plus adaptée.

9. Conclusion

La robotique modulaire est plus large qu'on peut la présenter dans un chapitre. Dans la dernière décennie les recherches dans ce domaine ont vu un élargissement, et les robots modulaires ont sorti de la simulation vers des applications réelles, tous en basant sur les techniques bio-inspirés.

Dans le chapitre suivant on va aborder les composants essentiels pour la conception d'un système robotique modulaire, et les méthodes utilisées pour le développer.

CHAPITRE 3

La Conception d'un Robot Modulaire

"By knowing where you are you, you will know where you are going."

— Anonymous

1. Introduction

L'espoir de créer des robots robustes, versatiles, adaptables, et pas cher a conduit les chercheurs à développer une suite des robots auto-reconfigurables améliorée, un robot modulaire avec toutes ses caractéristiques n'existe pas encore, mais afin de concevoir un tel système les chercheurs ont proposé l'ensemble de composants qui affecte le plus sa conception.

Dans ce chapitre, on va présenter les composants essentiels d'un système robotique, concentrant sur les contrôleurs et la morphologie qui sont les éléments de base de notre travail. Comme nous citons quelques travaux qui ont évolué les contrôleurs et d'autres qui ont coévolué les contrôleurs et la morphologie.

2. Les composants d'un système robotique modulaire

Un système robotique modulaire se compose de nombreux composants différents qui sont très dépendants les uns des autres, et celui-ci est l'aspect le plus important dans la conception d'un robot modulaire.

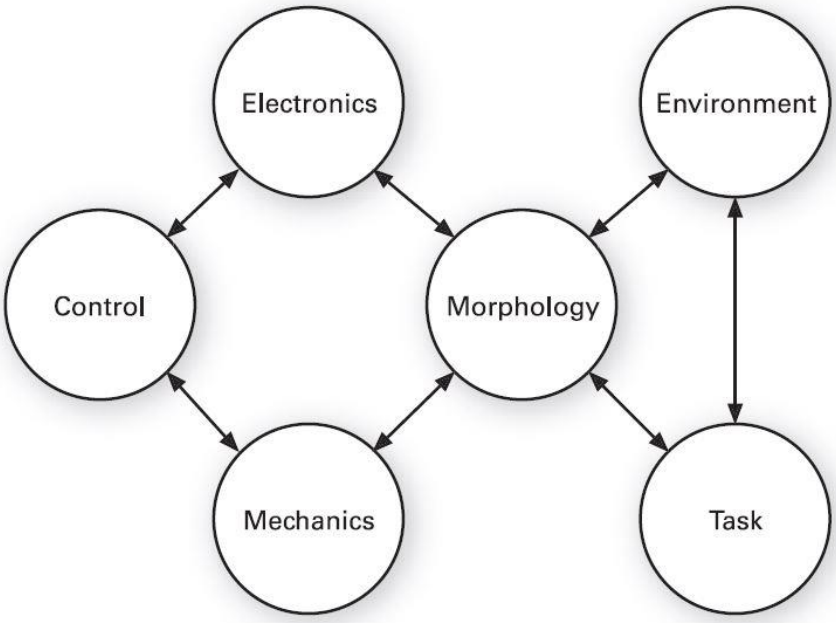


Figure 3.1. Les éléments impliqués dans la conception du robot et leurs interactions. [STO10]

Tout d'abord, un robot est conçu pour une tâche et un environnement précis. Un environnement sous-marin impose de contraintes complètement différentes avec celles d'un environnement terrestre ou d'un environnement spatial. La Même chose va pour les tâches: le robot a besoin d'être agile et rapide ou fort et lent? L'un des objectifs de robots auto-

reconfigurables est d'élargir l'éventail des tâches et des environnements qui peuvent être adressées avec un robot, mais pour l'instant il est difficile de penser de robots auto-reconfigurables universels.

L'interface d'interaction du robot modulaire avec l'environnement de travail est sa forme physique, les matériaux à partir desquels il est construit, et la position et le type de capteurs et d'actionneurs, dans l'ensemble, nous appelons ceux-ci la *morphologie* du robot. La morphologie est constituée de deux composantes: *mécanique* et *électronique*. Le composant mécanique définit la structure physique et souvent le type et l'emplacement des actionneurs du robot, tandis que le composant électronique fournit le robot avec son infrastructure de calcul, la puissance de communication, et les capacités de détection.

Enfin, le *contrôleur* est chargé de coordonner les actionneurs en réponse aux entrées des capteurs ou selon un plan généré.

En général, tous ces composants-là interagissent, de sorte que par exemple, un changement de tâche devrait résulter à des changements dans la morphologie, la mécanique, l'électronique et le contrôle du robot, mais pratiquement, le changement est souvent géré par les contrôleurs ; la partie la plus flexible du robot. Les robots auto-reconfigurables ont un avantage sur les robots conventionnels parce que leur morphologie est flexible, par conséquent, le contrôle et la morphologie sont des mines potentielles pour chercher des solutions si le matériel a déjà été construit.

Une conception d'un robot doit répondre aux exigences fonctionnelles des domaines d'application. Les conditions typiques incluent la force, la précision et la vitesse, par exemple dans une locomotion ou une tâche de manipulation. La fonctionnalité d'un système robotique auto-reconfigurable est décidée principalement par les fonctionnalités de modules qui le composent. Les caractéristiques qui sont souvent utilisés pour décrire la fonctionnalité de modules comprennent:

- *Capacité de s'auto-reconfigurer :* s'il peut s'auto-reconfigurer en deux ou trois dimensions.
- *Actuation :* le nombre, le type, la vitesse et la force d'actionnaire, degrés de liberté.
- *Capteurs :* le nombre et le type de capteurs.
- *Connecteurs :* le nombre, le type, la vitesse et la force de connecteurs.
- *L'infrastructure de communication et de calcul:* capacité à communiquer et effectuer des calculs
- *Puissance:* l'alimentation du robot et la capacité des modules de partager l'énergie.

On ne peut pas dire que nous avons respecté, tous ces caractéristiques présentées ci-dessus, puisque pour faire ça, on a besoin de connaissances électroniques et mécaniques. C'est pour ce raison là qu'on a utilisé un robot modulaire open-source comme un fondement de notre modèle.

Dans notre travail, on va concentrer sur trois composants ; les contrôleurs, La morphologie, et la tâche à accomplir y compris l'environnement.

3. Taxonomie de Robots Modulaire Reconfigurable

Afin que nous puissions comprendre et apprécier les progrès qui ont abouti à la génération moderne de robots auto-reconfigurables, nous passons en revue chronologique les robots développés [Figure. 3.2, Table 3.1], en présentant Molecubes [ZYK08] et Roombots qu'on a utilisés comme une inspiration pour développer notre modèle.

3.1. Molecubes (2007)

Ce système robotique [ZYK08] est homogène. Les modules sont constitués de deux demi-cubes articulés selon un axe de rotation autour d'une grande diagonale du cube et il comporte des connecteurs répartis sur les 6 faces du cube « Figure 3.2 ». C'est le seul système à utiliser des rotations à 120 degrés.

3.2. Roombots (2009)

Le Roombot présente les mêmes caractéristiques que Molecubes mais il possède en plus un axe de rotation sur chacun des connecteurs des six faces. Les Roombots sont composés de quatre semi-sphères alors leur forme générale diffère aussi dans le sens où le Roombot est plus sphérique que cubique « Figure 3.3 ».

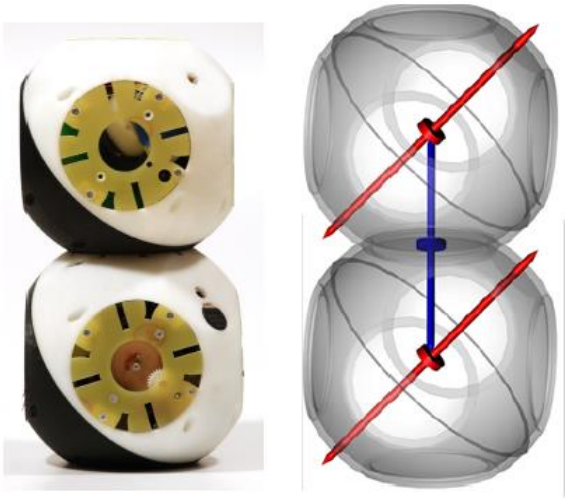


Figure 3.3. Un module de Roombot [SPR10].

3.3. Autres robots auto-reconfigurables

La table 3.1 représente le développement des robots modulaire à travers les générations [MAD12].

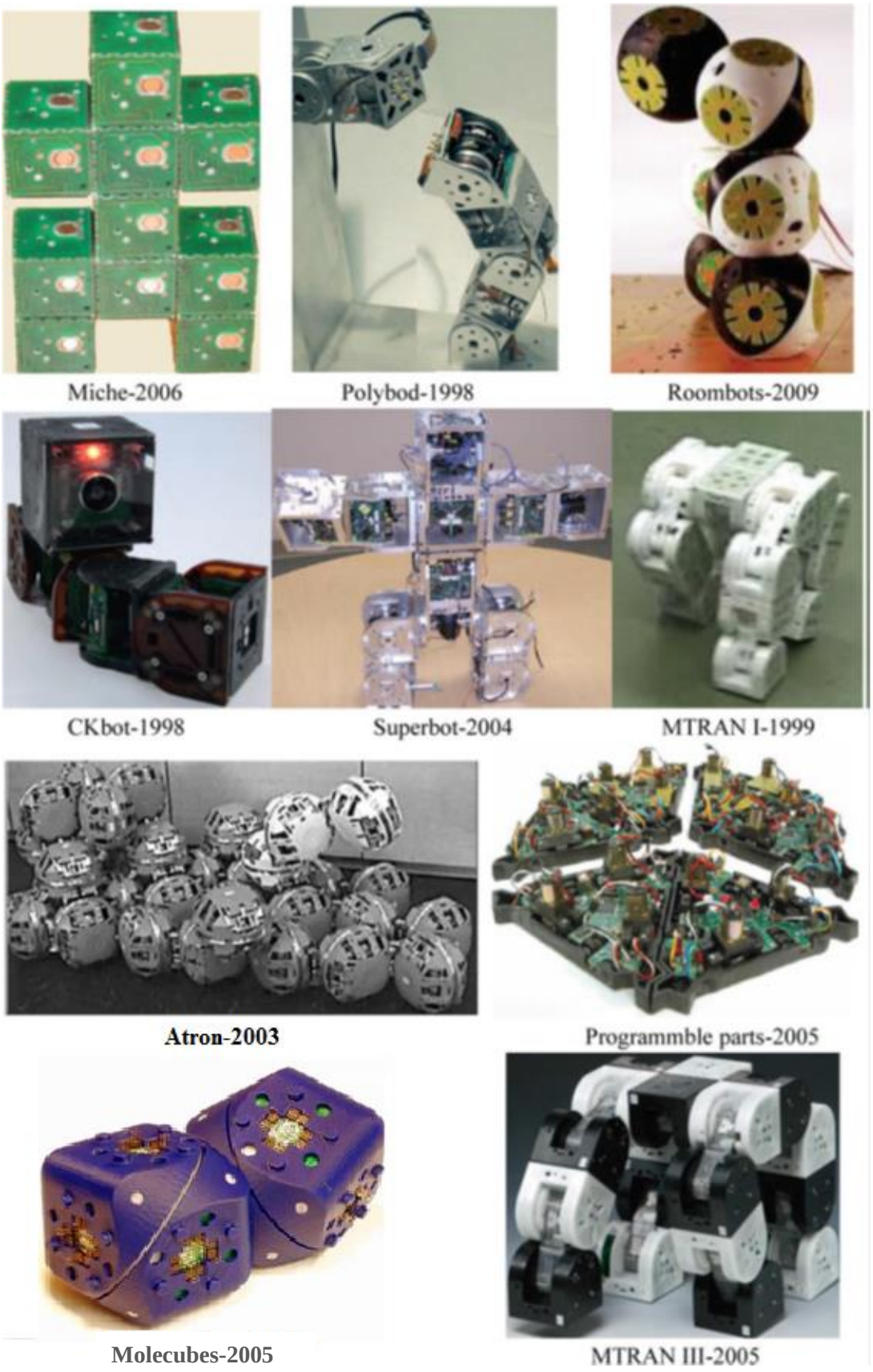


Figure 3.3. Quelques exemples de robots modulaires.

Sytème	Classe	DOF	Forme	Auteur	Année
Self replicator	Mobile	--	--	John Von Neumann	1950
Mechanical SR	Mobile	--	--	Lionel Penrose	1959
CEBOT	Mobile	Various	--	Fukuda et al. (Tsukuba)	1988
Polypod	Chain	2	3D	Yim (Stanford)	1993
Metamorphic	Lattice	6	2D	Chirikjian (Caltech)	1993
Fracta	Lattice	3	2D	Murata (MEL)	1994
Fractal Robots	Lattice	3	3D	Michael(UK)	1995
Tetrobot	Chain	1	3D	Hamline et al. (RPI)	1996
ANAT Robot	Chain	--	3D	Charles Khairallah (CA)	1997
3D Fracta	Lattice	6	3D	Murata et al. (MEL)	1998
Molecule	Lattice	4	3D	Kotay and Rus (Dartmouth)	1998
CONRO	Chain	2	3D	Castano et al. (USC/ISI) [CAS02]	1998
PolyBot	Chain	1	3D	Yim et al. (PARC) [YIM02]	1998
TeleCube	Lattice	6	3D	Suh et al. (PARC)	1998
Vertical	Lattice	--	2D	Hosakawa et al. (Riken)	1998
Crystalline	Lattice	4	2D	Vona and Rus (Dartmouth) [FIT00]	1999
I-Cube	Lattice	--	3D	Unsal (CMU) [UNS01]	1999
M-TRAN I	Hybrid	2	3D	Murata et al.(AIST)	1999
Pneumatic	Lattice	--	2D	Inou et al. (TiTech)	2002
Uni Rover	Mobile	2	2D	Hirose et al. (TiTech)	2002
M-TRAN II	Hybrid	2	3D	Murata et al. (AIST) [MUR02]	2002
Atron	Lattice	1	3D	Stoy et al. (U.S Denmark) [STO02]	2003
S-bot	Mobile	3	2D	Mondada et al. (EPFL)	2003
Stochastic	Lattice	0	3D	White, Kopanski, Lipson (Cornell)	2004
Superbot	Hybrid	3	3D	Shen et al. (USC/ISI)	2004
Y1 Modules	Chain	1	3D	Gonzalez-Gomez et al. (UAM)	2004
M-TRAN III	Hybrid	2	3D	Kurokawaa(AIST)[KUR06]	2005
AMOEBA-I	Mobile	7	3D	Liu JG et al. (SIA)	2005
Catom	Lattice	0	2D	Goldstein et al. (CMU)	2005
Stochastic-3D	Lattice	0	3D	Zykov, Lipson (Cornell)	2005
Molecubes	Chain	1	3D	Zykov, Mytilinaios, Lipson (Cornell)	2005
Prog. Parts	Lattice	0	2D	Klavins (U. Washington)	2005
Miche	Lattice	0	3D	Yim et al. (MIT) [YIM07]	2006
GZ-I Modules	Chain	1	3D	Zhang and Gonzalez-Gomez (U.,Hamburg, UAM)	2006
Odin	Hybrid	3	3D	Lyder et al., Modular Robotics Research (USD)	2008
Evolve	Chain	2	3D	Chang Fanxi, Francis (NUS)	2008
Roombots	Hybird	3	3D	Sproewitz, Moeckel, Ijspeert, Biorobotics Labaratory (EPFL)	2009
SMORES	Hybrid	4	3D	Daveyet al. (Modlab) [DAV12]	2012
M-Blocks	Hybird	--	3D	Romanishin et al. MIT CSAIL [ROM13]	2013

Table 3.1. Développement de robots modulaires.

4. Les contrôleurs

L'auto-reconfiguration est un aspect important dans les robots auto-reconfigurables. Cependant, Les fonctionnalités de ces robots ne se limitent pas à leur capacité de changer ses formes par l'auto-reconfiguration. En fait, les robots auto-reconfigurables passent la plupart de leur temps dans des configurations fixes et auto-reconfigure uniquement pour adapter aux modifications dans la tâche ou l'environnement [STO10]. Pour Un système de contrôle, soit

centralisé ou distribué « Figure 3.4 », Il faut savoir comment contrôler les robots auto-reconfigurables afin de réaliser la locomotion.

Plusieurs approches peuvent être envisagées pour contrôler un robot. Il peut en premier lieu être totalement dirigé par un opérateur humain mais le système ne possède alors aucune autonomie.

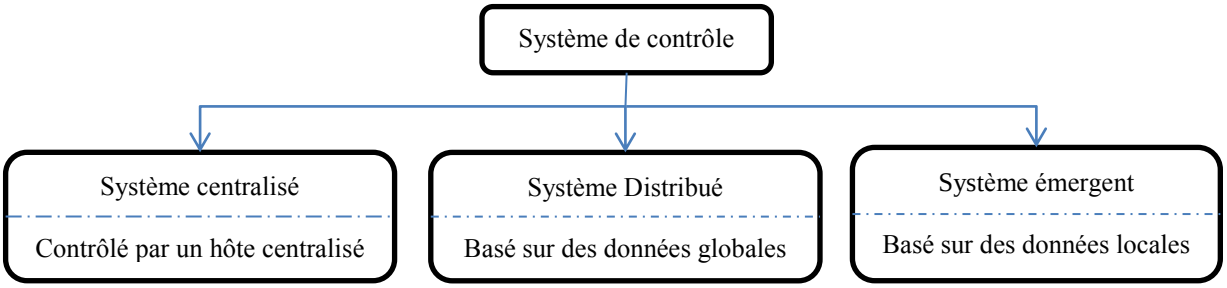


Figure 3.4. Système de contrôle pour les robots auto-reconfigurables. [MAD12]

Certaines solutions envisagent de laisser le robot apprendre par lui-même les actions à mettre en œuvre pour réaliser une tâche donnée. Là encore il existe plusieurs approches, ce sont des méthodes d'apprentissage artificiel, qu'il soit supervisé, non-supervisé ou encore par renforcement (exemple des réseaux de neurones).

On va présenter les approches les plus utilisées pour le contrôle des robots modulaires dans les sections suivantes.

4.1. Locomotion

La locomotion est le pouvoir de déplacer les modules de robot à partir d'un endroit à un autre. Les modes de locomotion peuvent être classés en deux catégories principales :

- Pour les configurations dynamiques, la locomotion est réalisée par l'auto-reconfiguration. Ce type de locomotion est souvent désigné comme *cluster-flow* ou *water-flow*. L'idée est que les modules de l'arrière de robot se déplacent vers l'avant du robot. Ce processus est répété et de telle façon la locomotion est générée. Ce mode de locomotion est bien adapté pour les robots treillis et hybrides. La locomotion *cluster-flow* est lente parce que le robot ne peut avancer qu'un module à la fois.
- Afin de pallier les limites de *cluster-flow*, un autre type de locomotion, qui imite les allures des animaux et qui se caractérise par la possibilité de générer des mouvements et qui n'implique pas l'auto-reconfiguration, était présenté. Pour n'importe configuration fixe, un mouvement est obtenu en contrôlant les articulations des modules.

Dans la section suivante on va présenter les différentes méthodes utilisées pour générer les mouvements dans les configurations fixes et leur classification.

4.2. Classification des contrôleurs de locomotions

Au cours des dix dernières années, les chercheurs ont transféré avec succès une large gamme d'allures de robots auto-reconfigurables, y compris mais non limité ; *caterpillaring, sidewinding*, *la marche en utilisant à la fois quatre et six pattes*, *rouler*, *grimper*, et même *le funambulisme*! La caractéristique qui se distingue la locomotion à part des autres tâches est que la séquence de mouvement nécessaire pour produire un tel mouvement est cyclique: les mouvements sont reproduits étape après étape, avec un peu de variation. Cette caractéristique a été exploitée pour créer plusieurs méthodes de contrôle simple et puissant pour la locomotion.

Les contrôleurs de locomotion pour les robots modulaires peuvent être classés en deux catégories (Figure 3.5.); les contrôleurs classiques et les contrôleurs bio-inspirés. La première classe provient du domaine de la robotique industrielle et elle est basée sur la cinématique inverse et la génération de trajectoire. Ce type de contrôleurs est difficile à mettre en échelle avec l'augmentation des degrés de liberté, et il nécessite une puissance de calcul élevée. Au contraire, la classe ultérieure des contrôleurs sont inspirés de processus biologiques. Ces contrôleurs ont été implémentés avec succès sur différentes plates-formes robotiques modulaires. Basant sur la méthode utilisée, ces contrôleurs peuvent en outre être sous-classés en contrôleurs basés sur les automates cellulaires, méthode d'hormone numérique, et des méthodes basées sur l'oscillation.

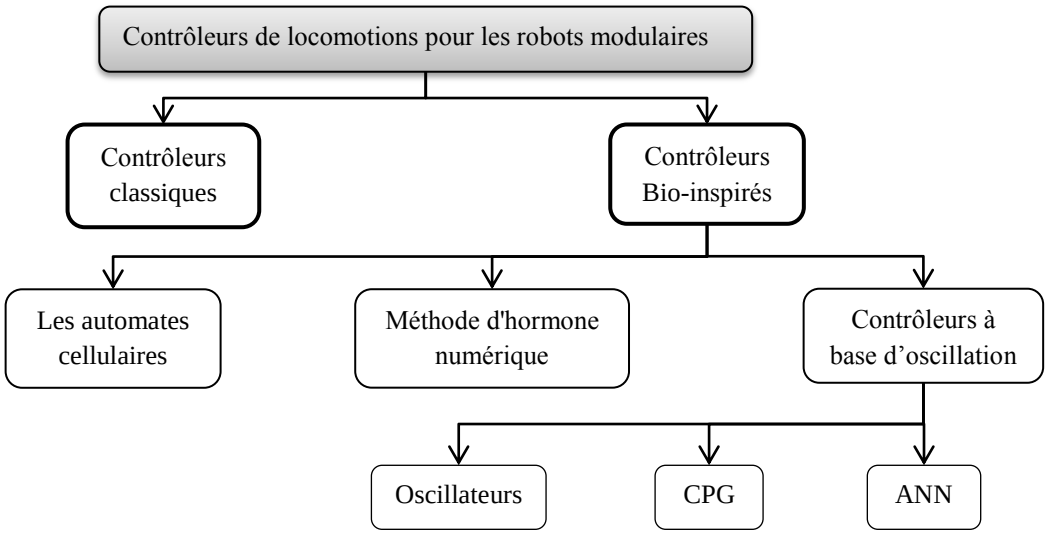


Figure 3.5. Classification des contrôleurs.[RAN11]

4.2.1. Tables de contrôle de démarche (Gait Control Table)

Les tables de contrôle de démarche *(GCT)* étaient l'une des premières méthodes de contrôle de la locomotion des robots auto-reconfigurable [YIM93], [YIM94], [YIM01]. Bien qu'elles sont relativement limitées, elles sont toujours utiles en raison de leur simplicité.

La GCT est un tableau représentant un cycle complet d'une démarche, où la colonne représente les étapes qu'un module spécifique doit exécuter dans chaque cycle de la marche, et la ligne du tableau contient les mouvements que le robot doit initier à chaque étape.

Les GCTs sont implémentées plus facilement à l'aide d'un contrôleur centralisé. Le contrôleur centralisé va attendre les déclencheurs et une fois ils déclenchés, il donne les instructions à tous les modules pour passer à l'étape suivante.

4.2.2. Contrôleur à base d'hormones

Contrôleur à base d'hormones aspire à être un framework complet de contrôle de robots auto-reconfigurables [SAL04][SHE06]. Cependant, au début, sa contribution principale était de fournir une solution distribuée au problème de la synchronisation et la dépendance de GCTs.

Un module est choisi comme un initiateur ; au début de chaque étape, il envoie un message, une hormone, au module suivant dans la chaîne. Cette hormone contient des informations à propos de l'étape que le module d'initiateur exécute actuellement. Sur la base de cette information, le module suivant choisit la prochaine étape et passe une hormone dans le module ci-dessous pour l'informer de son choix et ainsi de suite. Ce processus se poursuit jusqu'à ce qu'une hormone atteigne le dernier module de la chaîne.

Cette méthode peut être facilement étendue pour gérer les démarches de configurations plus complexes, comme la marche quadrupède, comme cela a été démontré en [SHE02], [SHE00], [SAL01], [HOU06], [HAM10], [HAM11]. Cependant, elle introduit un point faible dans le système car si une hormone est perdue, l'ensemble du robot cesse de bouger depuis tous les modules sont en attente de l'hormone. Il peut donc être nécessaire de mettre en œuvre une stratégie pour relancer la locomotion si une hormone est perdue. Un autre problème est le sélectionnement du module initiateur. En d'autres termes, comment le module de tête sait qu'il est l'initiateur? La réponse la plus simple à cette question, c'est au programmeur ou l'utilisateur à choisir, ou d'implémenter des méthodes pour sélectionner le module initiateur.

4.2.3. Générateurs de motifs centraux (CPGs)

Le contrôleur à base d'hormones est une méthode de contrôle qui est originalement orienté vers le codage manuelle de démarche. Cependant, le développement automatique des allures a été étudié en utilisant les générateurs de motifs centraux « Central Pattern Generator (CPG) » dans plusieurs travaux [POU10], [IJS08], [SPR08].

Le CPG est une approche bio-inspiré basé sur les réseaux de neurones. La principale fonction de la CPG est de produire une sortie rythmique sans entrées. Il est possible d'utiliser les algorithmes génétiques pour évoluer un réseau de CPG pour contrôler la démarche d'un robot auto-reconfigurable [KAM05]. En outre, Kamimura et al. [KAM03] ont également montré comment les CPGs peuvent détecter un changement dans le terrain à travers le changement de la démarche et utiliser cette information pour optimiser automatiquement la démarche.

4.2.4. Les réseaux de neurones Artificielles (RNA)

Pour développer des allures automatiquement, plusieurs travaux ont implémenté différent types de réseaux de neurones (le plus simple avec un seul neurone jusqu'au les réseaux NEAT) [GON05], [ZHA09], [LAL07].

Avinash et al. dans [RAN11] ont proposé un modèle neuronal de locomotion en robotique modulaire, ils ont validé leur modèle dans des configurations linéaires pour produire des oscillations locomotrices. Un contrôleur neuronal a été évolué, convergé et s'installé en un motif oscillant stable, ce qui se traduit par une démarche de locomotion stable sur la base de la sortie oscillatoire des RNAs simples.

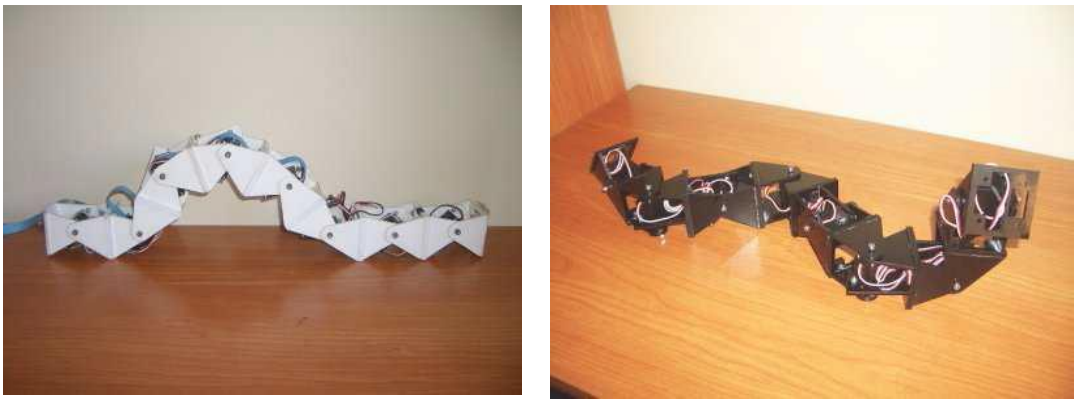


Figure 3.6. Configuration linéaire de huit modules. [GOM08]

Toutes ces méthodes ont des avantages différents. Les tables de contrôle de la démarche sont extrêmement simples à mettre en œuvre, mais manquent de certains mécanismes de synchronisation comme les méthodes à base d'hormones. Enfin, les générateurs de motifs centraux comme les RNAs sont un moyen classique pour contrôler la locomotion des robots et ont été utilisé pour la génération automatique des allures, mais avec les CPGs, si on modifie la configuration des modules du robot, le couplage de CPGs doit être modifié pour s'adapter à la nouvelle configuration, ce qui pourrait être un inconvénient si on prévoit que le contrôleur soit implémenté dans des robots modulaires auto-reconfigurables robots [RAN11].

Dans notre travail, on a utilisé les RNAs pour produire des allures automatiques pour différents types de configurations, le modèle utilisé est présenté dans le chapitre suivant.

5. La morphologie

Puisque l'objectif de ce travail est de créer des robots modulaires plus adaptés en sens de morphologie, contrôleurs et de tâche, on va présenter une vue générale sur les travaux qui ont essayé d'évoluer les morphologies de robots tout en détaillant le travail qu'on a utilisé comme une base de le nôtre. La reconfiguration morphologique d'un robot modulaire.

Il y a eu plusieurs publications décrivant des travaux dans lesquels les algorithmes évolutionnaires ont été employés dans la production de morphologies en trois dimensions.

Souvent, l'évolution de structure physique seulement est une première étape dans une trajectoire de recherche finalement orientée vers l'évolution conjointe de la morphologie et les contrôleurs de robot [BUA05]. Il y a des travaux qu'ont étudié l'évolution de morphologies eux-mêmes comme un objectif final soit pour des raisons d'ingénierie ou artistique. Bien que cette seconde classe peut faire usage de nombreuses techniques utilisées pour l'évolution des morphologies de robots, ses objectifs sont différents.

5.1. De l'échafaudage vers la robotique modulaire

Des travaux récents ont été réalisés sur la reconfiguration morphologique de structures en échafaudage selon une approche théorique d'une part [YUN08][LOB09] et selon une approche pratique d'autre part [HJE09].

L'échafaudage est représenté par une structure arborescente où les transformations élémentaires de la structure sont définies comme étant des échanges ou rotations de branches. A partir d'une morphologie initiale, un algorithme génétique fait évoluer une suite de transformations élémentaires en associant la fitness en fonction du but morphologique recherché (hauteur, overhead, etc.) (Figure 3.7, Figure 3.8).

Cette approche a été adaptée par Yves Duthen et al. **[DUT10]** aux robots modulaires en chaîne et plus particulièrement un modèle inspiré de Molecubes et Roombots.

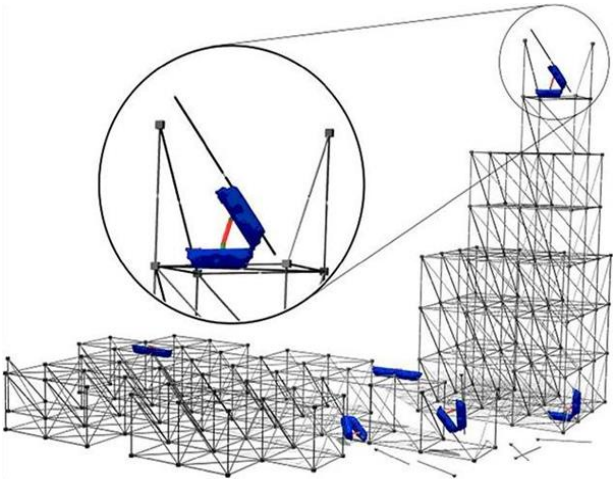


Figure 3.7. Vue d'artiste de robots manipulant une structure en échafaudage [HJE09].

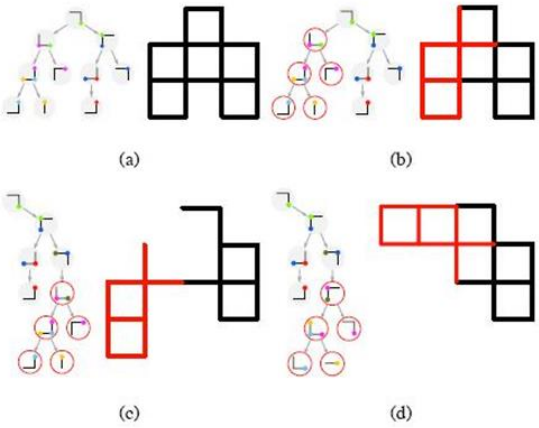


Figure 3.8. Représentation du déplacement puis de la rotation d'une branche au sein de d'une structure arborescente [HJE09].

A cause de la possibilité de boucles internes la structure arborescente n'est pas adaptée, ils ont utilisé un graphe. De plus la liste des transformations élémentaires est réduite. En effet dans les travaux sur les échafaudages un opérateur extérieur (un robot) est responsable d'opérer la transformation, ce qui autorise des mouvements de déplacement d'une branche d'un endroit de l'arbre à l'autre comme ce serait le cas pour un robot mobile. Ce n'était pas le cas ici puisque les auteurs ont fait le choix d'un robot chaîné qui par définition doit rester connexe et doit être le seul intervenant dans les mouvements de ses modules.

Les individus évolués sont des suites de mouvements de base à effectuer les uns après les autres par le robot modulaire. Afin d'évaluer la population de ces individus dans un environnement le plus réaliste possible ils ont implémenté un simulateur en utilisant le moteur physique de NVidia : PhysX.

5.1.1. Le modèle robotique

Ils ont utilisé un modèle inspiré de Roombots [SPR10]. On rappelle que le module est de forme cubique et présente un connecteur permettant une rotation sur chacune de ses six faces. Il possède de plus un axe de rotation interne défini par deux coins opposés du cube. Il est donc modélisé par deux polyèdres (représentés ci-après) qui lorsqu'ils sont reliés par le centre de leur face hexagonale et orientés de manière correcte forment un cube (Figure 3.9.a.).

Ce polyèdre a été la première version de module. Celle-ci présentait l'inconvénient de ne pas permettre certains mouvements de rotations entre modules que permet la forme pseudo-sphérique du Roombot. Il a donc été décidé de rogner les angles afin de s'approcher de cette forme tout en conservant une représentation relativement simple (et donc moins couteuse en terme de calculs). Le résultat est représenté en « Figure 3.9.b. » et un rendu de module comme il est en simulateur en « Figure 3.9.c. ».

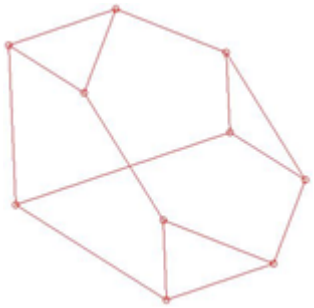


Figure 3.9. (a). Une moitié de module (1ère version)

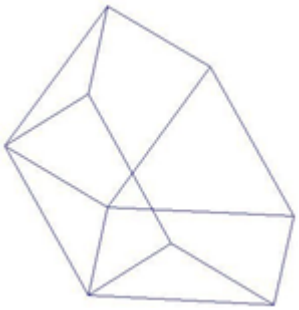


Figure 3.9. (b). Une moitié de module (2ème version)

Figure 3.9. (c). Un module: l'assemblage de deux moitiés.

5.1.2. Le robot

Le robot est composé de l'ensemble des modules présents dans la simulation. Les relations entre les modules qui définissent la morphologie du robot étant déjà implémentée au sein de la représentation de ces modules. Le robot possède aussi un attribut flottant codant pour la vitesse de rotation des moteurs, vitesse commune à tous les modules (Figure. 3.10.).

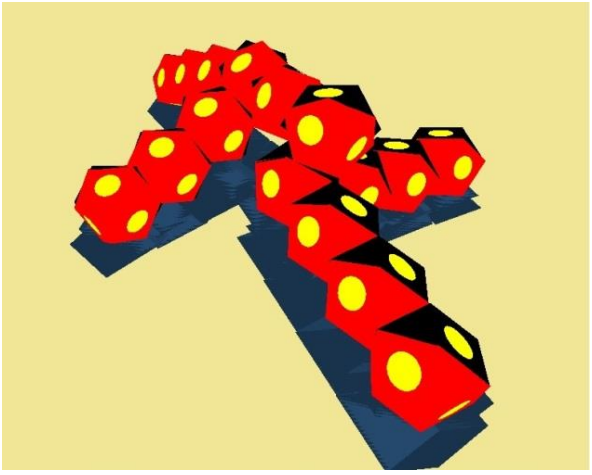


Figure 3.10. Une configuration du robot modulaire (4 pattes).

5.1.3. Fonction de fitness

Pour l'évaluation de performance du robot, ils ont choisi la hauteur maximale atteinte par un module en particulier comme une fonction de fitness afin d'arriver à traduire le but recherché par un critère facilement calculable.

Après avoir évalué la morphologie du système modulaire par le biais de la fonction de fitness et après avoir positionné l'attribut mark du gène correspondant, le robot est détruit. Ceci revient à vider la scène de tous ses acteurs à l'exception du sol et un nouveau robot en configuration initiale est créé et ajouté à la scène afin d'être évaluer. L'algorithme qu'ils ont utilisé pour l'évoluer les populations est schématisé dans la Figure 3.11.

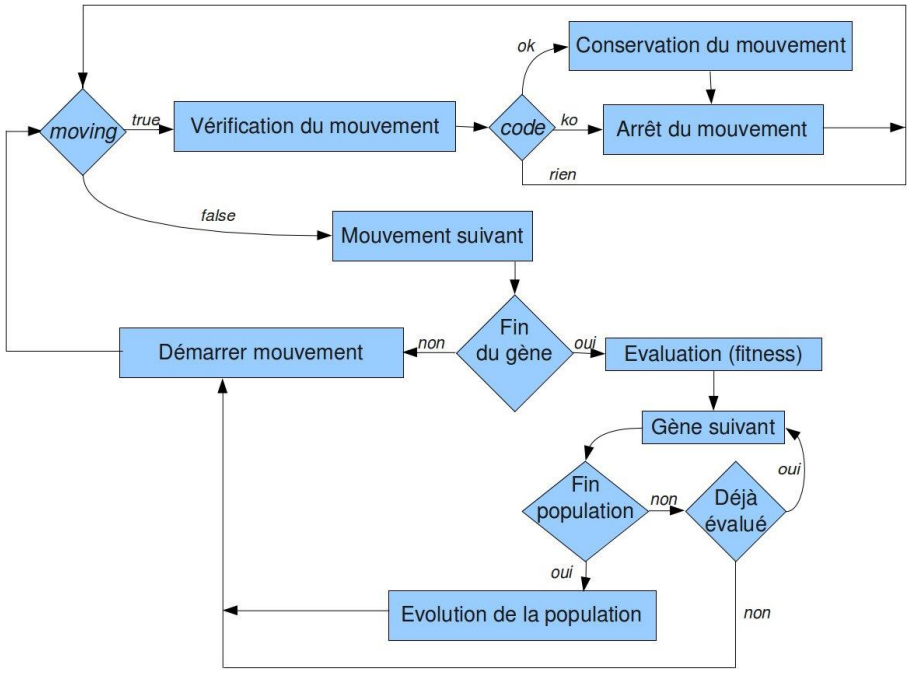


Figure 3.11. Diagramme de l'algorithme [DUT10].

5.1.4. Les résultats obtenus

Cette approche a été testée pour trois configurations : serpent de trois modules, serpent de 5 modules et une forme d'araigne avec 21 modules. Les populations ont été évoluées en utilisant les paramètres dans la table 3.2.

Paramètres	Taux
individus géniteurs sélectionnés à l'itération précédente	20%
individus obtenus par mutation d'individus sélectionnés	20%
individus obtenus par croisements d'individus sélectionnés	10%
individus obtenus par mutations d'individus tirés au hasard parmi les 30% meilleurs	30%
nouveaux individus	20 %
un taux de mutation aléatoirement tiré entre	10% et 20%

Table 3.2. Les paramètres utilisés pour l'évolution de la configuration.

5.1.5. Evaluation

La grande difficulté de ce travail réside à trouver les bons paramètres qui accélèreront les convergences malgré que les configurations atteint des bons résultats, un robot serait sans grand intérêt s'il était dépourvu de contrôleur. Le contrôleur d'un module peut être vu comme étant son cerveau, ce qui commande son comportement et le modèle présenté dans ce travail est dépourvu de contrôleurs. Ce qui est un grand manque dans leur travail. Cependant sur certaines simulations et au niveau d'auto configuration, les résultats obtenus sont encourageants.

Nous avons l'utilisé comme base pour notre travail, comme premier pas, on a proposé un contrôleur neuronal pour générer des mouvements rythmique et dans une deuxième pas, on a proposé une approche co-évolutionaire coopérative pour développer des configurations et contrôleurs plus adaptés au l'environnement et à la tâche. On va les détailler dans les chapitres 4 et 5.

6. La co-évolution de morphologie et du contrôleur

La robotique évolutionnaire [HAR97][NOL00] est une technique bio-inspirée dans laquelle les algorithmes évolutionnaires sont utilisés pour optimiser le système de contrôle d'un robot. Cette technique a fourni une plateforme pour surmonter les limites de l'intuition humaine dans la conception de stratégies de contrôle robuste et non linéaire pour les machines autonomes [BON11]. Cependant, la majorité des travaux en robotique évolutive n'a optimisé que les stratégies de contrôle pour des morphologies conçues par l'être humain ou du robot bio-inspiré. Cette méthodologie a ses limites: fixer la morphologie du robot pose des limites et des biais sur les types d'actions qu'un robot peut effectuer, et donc aussi sur les tâches les plus complexes que ces actions pourraient éventuellement réaliser.

Cependant, il y a des façons de pallier ces limitations. Les algorithmes évolutionnaires peuvent être utilisés pour optimiser la morphologie d'un robot et en plus son système de contrôle. Karl Sims [SIM94a] a introduit un cadre évolutif dans lequel il a optimisé la morphologie et les contrôleurs des créatures virtuelles dans des environnements simulés pour

produire un comportement adaptatif. Ce travail a été suivi par d'autres études qui ont essayé d'évoluer la morphologie et les contrôleurs de robots dans des environnements virtuels. Cette approche a l'avantage de découvrir et développer des configurations, des capteurs et des contrôleurs plus adaptés à l'environnement et à la tâche que seuls conçus par l'être humain.

Ceci nous amène à une différence fondamentale [AUE13] entre les algorithmes évolutionnaires et les méthodes d'optimisation plus formelles telles que l'apprentissage [SUT99] [ABB04][TAS08]: ces derniers sont adaptées pour l'optimisation paramétrique qui exige le garanti de la convergence, tandis que le premier permettre une amélioration topologique où il n'y a pas de telles garanties. La majorité des techniques d'optimisation actuellement se basent sur l'optimisation d'un nombre fixe de paramètres: ces paramètres peuvent spécifier le système de contrôle d'un robot, ou dans certains cas aussi des aspects de sa morphologie [DOL07]. Cependant, plusieurs algorithmes évolutionnaires ne présument pas l'optimisation de paramètres d'une structure fixe mais ils améliorent les structures et leurs paramètres en même temps [KOZ92][BON07].

Dans ce sens plusieurs travaux ont essayé d'évoluer conjointement la morphologie et les contrôleurs pour explorer un espace de solutions plus large, en paramétrant certaines propriétés d'un robot, et en utilisant un algorithme évolutionnaire pour évoluer ces paramètres.

Balakrishnan et Honavar (1996)

Au début, les chercheurs ont employé les algorithmes génétiques pour modifier les aspects du système sensoriel du robot. Un premier travail dans lequel cela a été accompli par Balakrishnan et Honavar [BAL96]. Dans ce travail, les auteurs ont évolué les positions des capteurs pour un robot « Khepera » simulé. Plusieurs travaux après ont évolué les paramètres sensoriels de robots comme Mark et al. (1998) [MAR98], Ce travail a exploré plusieurs techniques, y compris: l'évolution des capteurs, concurrence entre les robots, et la spéciation (les robots ont été placés dans des familles et la reproduction sexuelle n'a été autorisée qu'entre les membres d'une même famille).

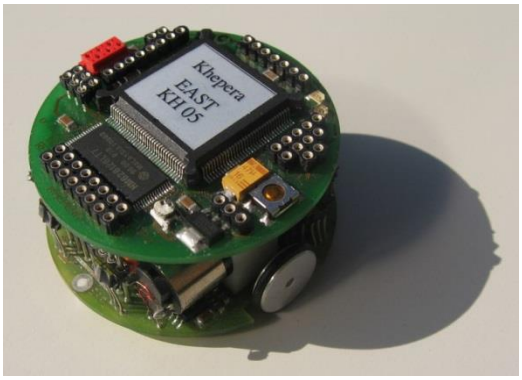


Figure 3.12. La première version de Khepera.

Outre les capteurs d'un robot, il est également possible de paramétrer d'autres aspects de la morphologie d'un robot comme la taille des segments ou des modules, les intervalles

d'articulation, les rayons de roue, l'évitement d'obstacles et ainsi de suite. Plusieurs publications dans lesquelles les chercheurs ont adopté cette approche comme [BON11].

Lund (2003)

Lund [LUN03] a évolué les poids du contrôleur de réseau de neurones de topologie fixe et les paramètres morphologiques des robots LEGO en simulation et ensuite les construite avec des kits LEGO MINDSTORM. Plus précisément, l'algorithme évolutionnaire pourrait choisir parmi trois différents types de roues LEGO (ayant un diamètre différent, la largeur,…), 25 empattements différents sur un axe LEGO au centre du robot, et 11 positions différentes de deux capteurs de lumière LEGO. Une variabilité des morphologies évoluées a été observée.

Shen et al. (2006)

En 1999, l'Institut national de science industrielle avancée et de la technologie du Japon a développé un robot modulaire auto-reconfigurable (M-TRAN). Il était novatrice en fusionnant deux de type de robots auto-reconfigurables ; treillis et chaîne [MUR00]. M-TRAN a réalisé avec succès la locomotion robotique multi-mode. Il se compose de deux parties semi-cylindriques qui sont reliés par une charnière. Chaque partie semi-cylindrique peut tourner sur 180 degrés et a trois connecteurs, qui peuvent se relier à d'autres modules par la force magnétique. En Shen et al. [SHE06] SuperBot a apparu, il comprenait un degré de liberté supplémentaire par rapport à la M-TRAN robot, en SuperBot un degré de liberté a été ajouté pour rendre l'orientation entre ces deux actionneurs contrôlables.

Hornby et Pollack (2000)

Hornby et Pollack [HOR00] ont utilisé une approche co-évolutionnaire pour faire évoluer des robots modulaires. Des blocs élémentaires comprennent des bars et des actionneurs pour la morphologie encodés en utilisant L-systèmes et les réseaux de neurones pour le contrôleur Les meilleures créatures ont été utilisées après pour la création des robots réels.

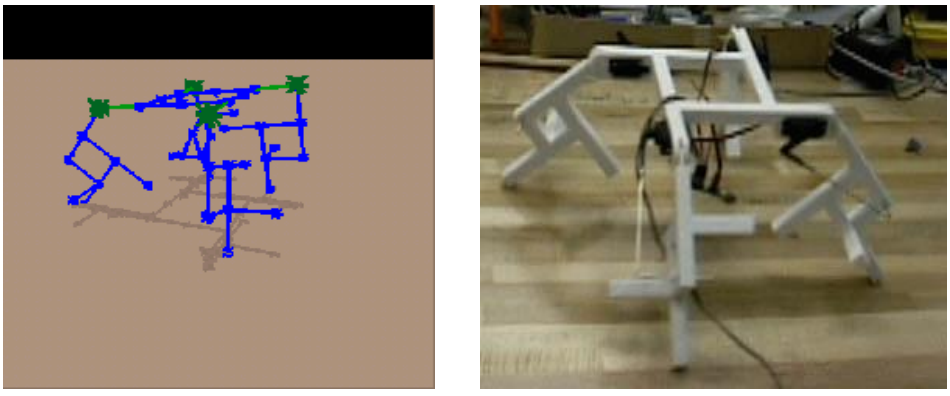


Figure 3.13. Un des robots modulaires évolué dans [HOR00].

Marbach et Ijspeert [MAR04], ont également travaillé sur la co-évolution de la configuration et des contrôleurs de robots modulaires. Ils ont utilisé une représentation basée sur les arbres pour encoder les configurations et les contrôleurs pour Adam [MAR04] ; un

outil de simulation de robots modulaires basé sur une approche co-évolutionnaire. En utilisant des modules qui contiennent un articulaire et des connecteurs plates ; les créatures évolués développent une large gamme de stratégies de locomotion, souvent semblables à ceux des organismes vivants dans la nature. En outre, les individus ont une tendance à être symétrique, même si cela n'a pas été directement codé.

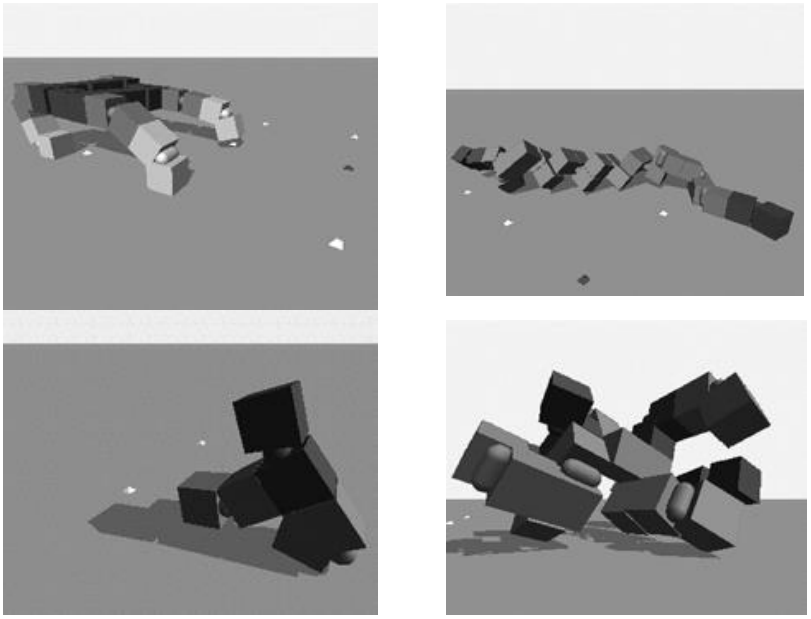


Figure 3.14. Exemples de robots évolués dans Adam [MAR04].

Auerbach et Bongard (2012)

Auerbach et Bongard [AUE12] ont évolué la morphologie et les contrôleurs des robots simulés dans des environnements virtuels, les morphologies sont composées d'un nombre de mailles triangulaires qui peuvent modéliser des formes arbitraires et ainsi permettre la création de morphologies plus complexes que ceux qui sont créées avec des rectangles ou des sphères. Ils ont utilisé les CPPNs « *Compositional Pattern Producing Networks* » comme encodage génératif afin d'évoluer simultanément la morphologie et les contrôleurs du robot. Une méthode a été présentée pour traduire les CPPNs en robots complets, y compris leurs topologies physiques, le positionnement des capteurs, et les contrôleurs (réseaux de neurones). Il est montré que cette méthode peut évoluer des robots adaptés à une tâche donnée.

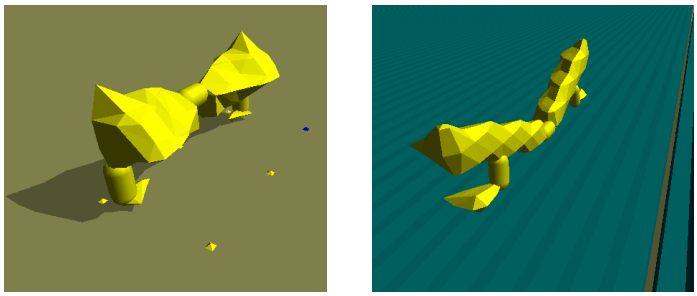


Figure 3.15. Deux robots évolués dans des environnements expérimentaux [AUE12].

Bongard (2009)

Bongard [BON09] met en évidence la raison pour laquelle la morphologie et les contrôleurs doivent être évolués afin de réaliser des comportements de plus en plus complexes pour les robots autonomes et d'augmenter le taux d'adaptation. Cela a été démontré pour un robot à pattes qui doit locomote puis manipuler un objet. Ceci est considéré comme une tâche complexe où le même contrôleur doit coordonner le mouvement des pattes durant la locomotion et le mouvement de la pince pendant la manipulation.

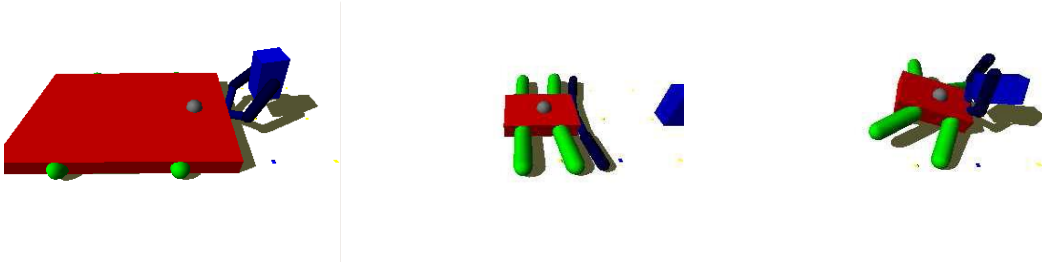


Figure 3.16. L'objet est placé hors de sa portée, le robot évolue pour locomote à l'objet, puis le soulevez [BON09].

Faíña et al. (2013)

Récemment, Faíña et al. [FAI13] ont proposé une approche co-évolutionnaire pour l'encodage, l'évaluation ou le transfert à la réalité, en utilisant des structures modulaires hétérogènes avec un contrôle distribué. Ils ont utilisé un algorithme évolutionnaire basé sur des représentations arborescentes pour la morphologie. L'évaluation des individus est effectuée dans les simulations, puis transféré à des robots réels assemblés à partir de modules évolués. Tous ces problèmes sont analysés au moyen d'un système de conception évolutionniste appelé EDHMoR *(Evolutionary Designer of HeterogeneousModular Robots)* qui contient tous les éléments impliqués dans ce processus. Pour montrer des preuves concrètes des conclusions qui ont été extraites de ce travail, deux problèmes de référence en robotique modulaire sont considérés et EDHMoR est testé sur eux. Le premier est l'avancement aussi loin que possible dans un terrain raboteux transportant une charge de poids indéterminée et minimisant la consommation d'énergie et le second est une tâche statique du robot qui ne nécessite pas des déplacements ;faire peintre un espace bien défini.

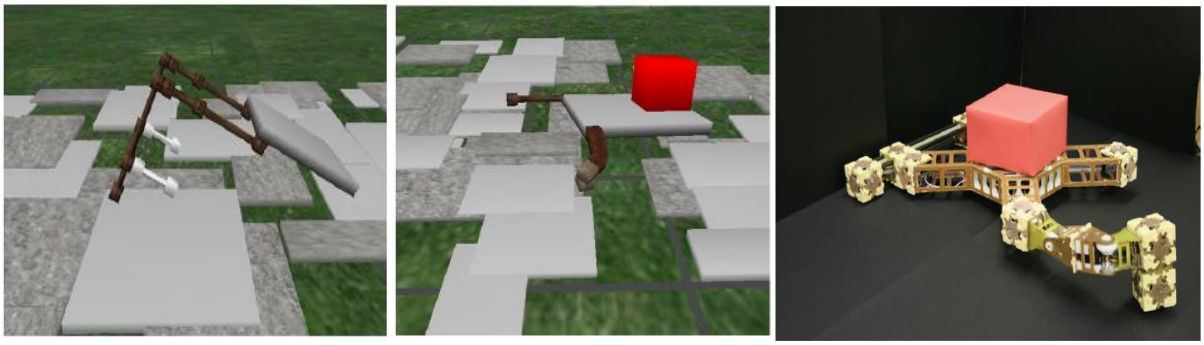


Figure 3.17. Exemples de robots évolués et un prototype du meilleur robot évolué. [FAI13]

7. Conclusion

Les robots modulaires avec ses caractéristiques, les tâches à réaliser et les environnements prévus pour fonctionner dans, posent plusieurs contraintes sur leur conception.

Afin d'opérer dans des environnements non structurés les robots modulaires devront être adaptatifs, ils doivent présenter un comportement intelligent. En réponse à cela, la robotique évolutionnaire a été présenté et plusieurs projets ont évolué conjointement la morphologie et les contrôleurs pour réaliser les comportements désirés et surmonter les limites de l'intuition humaine dans la conception.

Dans le chapitre suivant, nous allons présenter notre premier partie de travail ; des contrôleurs homogènes distribués pour la locomotion dans les robots modulaires, avec les résultats des simulations de différentes configurations.

CHAPITRE 4

Contrôleurs Distribués Homogènes

"Don't be afraid to give your best to what seemingly are small jobs. Every time you conquer one it makes you that much stronger. If you do the little jobs well, the big ones will tend to take care of themselves."

— Dale Carnegie

1. Introduction

L'approche générale dans les robots modulaires est de concevoir la morphologie, puis optimiser les contrôleurs de la structure pour une tâche donnée et enfin de déterminer une séquence de mouvements pour reconfigurer le robot à partir d'une configuration initiale à une autre finale pour résoudre le problème d'auto-reconfiguration.

Alors, les robots auto-reconfigurables passent la plupart de leur temps dans des configurations fixes et s'auto-reconfigure uniquement en réponse à des modifications en tâche ou en environnement. Avec ses capacités d'auto-reconfiguration, les robots modulaires peuvent être utilisés dans des environnements imprévisibles. Cette caractéristique est, inutile de le dire, un aspect important de robots auto-reconfigurables. Cependant, ces robots ont besoin des contrôleurs pour accomplir leur locomotion comme l'une des fonctionnalités de base d'un tel système.

Dans ce chapitre, on va prendre le chemin général, qui consiste à proposer des configurations bio-inspirées puis implémenter notre modèle de contrôleurs basant sur les réseaux de neurones, et finalement tester sa performance dans la tâche de locomotion.

2. L'approche envisagée

Notre objectif est de réaliser une simulation réaliste des robots modulaires, même si nous ne sommes pas en train de travailler sur un prototype matériel, nous voulons que nos résultats soient théoriquement transférables à la réalité.

Nous visons à une simulation réaliste utilisant des modules homogènes avec six connecteurs et sept degrés de liberté et on évolue les contrôleurs du robot à l'aide de ce type de module prédéfini. Le modèle du RNA proposé a pour but de renforcer les caractéristiques distinctives de robots modulaires ; la distributivité, l'homogénéité, la simplicité.

Les algorithmes génétiques sont utilisés pour optimiser les contrôleurs pour les adapter aux différentes configurations de robots.

2.1. Le robot modulaire proposé

Le module a une forme cubique avec six surfaces de connexion. La forme et la description sont inspirées de Molecubes [ZYK08]. Chaque module est composé de deux moitiés avec un axe de rotation interne définit par les deux angles diagonalement opposés et six axes de rotation supplémentaires sur chacune des surfaces de connexion. Chaque degré de liberté à un intervalle de rotation de 360 degrés. Les dimensions de ces modules sont 20x20x20 mm.

2.2. Les contrôleurs

Dans notre travail, nous dotons ces modules cubiques avec des contrôleurs de réseaux de neurones distribués et homogènes. Nous allons étudier leur efficacité pour la locomotion dans des configurations robotiques bio-inspirés fixes, les mouvements se basent sur les sorties oscillatoire des RNAs simples. Nous avons utilisé un algorithme génétique pour optimiser les poids synaptiques du RNA pour produire des allures rythmiques qui maximisent la distance parcourue par le robot pour un nombre prédéfini d'étapes de temps.

Le modèle de RNA est entièrement connecté (Figure. 4.1) avec trois nœuds d'entrée, deux nœuds de sortie, et une seule couche cachée avec cinq nœuds. Nous avons essayé de garder l'architecture du RNA le plus simple possible, afin d'investiguer l'architecture minimale qui peut générer des allures de locomotion rythmiques en utilisant différentes configurations et renforcer les caractéristiques distinguées de robots modulaires :

- *Nœuds d'entrée 1*: Ce neurone représente le nombre de voisins qui sont connectés à ce module. Ce neurone est alimenté avec une entrée compris entre 1 et 6.
- *Nœuds d'entrée 2*: est alimenté avec la valeur de l'angle précédent de l'actionneur, entre -120° et 120°. Ces valeurs sont choisies après plusieurs expériences.
- *Nœuds d'entrée 3*: Ce neurone est alimenté avec l'identification du module, il n'y a pas d'intervalle spécifique à cette valeur, car elle varie en fonction de nombres des modules qui composent le robot.
- *Nœuds de sortie 1*: représente l'angle d'actionnement et cette valeur est mise à l'échelle pour être comprise entre -120° et 120°.
- *Nœuds de sortie 2*: génère l'identification du connecteur qui va exécuter l'actionnement, et cette dernière ne peut être réalisé que si le module a un voisin sur ce connecteur.

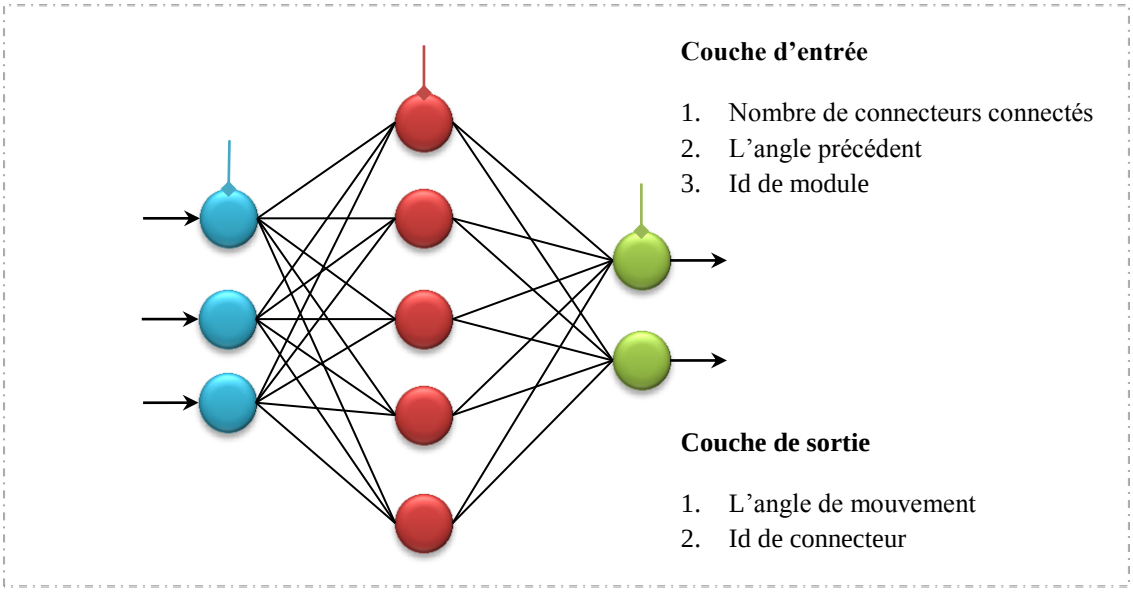


Figure 4.1. Schéma de l'architecture de réseaux de neurones proposé.

Chaque module, dans une configuration donnée, a son propre contrôleur RNA dont les nœuds de sorties sont connectés aux actionneurs du module, ce qui en fait un contrôleur distribué. En outre, tous les modules ont la même architecture de RNA, avec exactement le même vecteur de poids, résultant un contrôleur distribué homogène. Bien que tous les modules aient des RNAs identiques, la différence dans leur comportement émerge de différentes entrées appliquées à leurs RNAs à chaque cycle.

Avec les GPCs *(Central Pattern Generator),* si la configuration du robot modulaire se change, le couplage des GPCs doivent être modifiés pour s'adapter à la nouvelle configuration, ce qui pourrait être un inconvénient dans les robots modulaires auto-reconfigurables [RAN11]. Le modèle de RNA proposé a la possibilité d'être complexifier en changeant le nombre des couches cachées ou des nœuds de l'architecture du RNA.

2.3. Le processus d'évolution des contrôleurs

L'évolution des réseaux de neurones en utilisant les algorithmes génétiques dépend généralement de la façon de codification du RNA et les étapes suivirent lors de la simulation.

Pour notre approche nous allons utiliser une version ajustée d'algorithmes génétiques afin d'obtenir des contrôleurs plus adaptés, le plus rapide que possible.

La Figure. 4.2 représente le processus suivi pour évoluer les contrôleurs, on va expliquer les étapes de cette processus une par une dans cette sous-section.

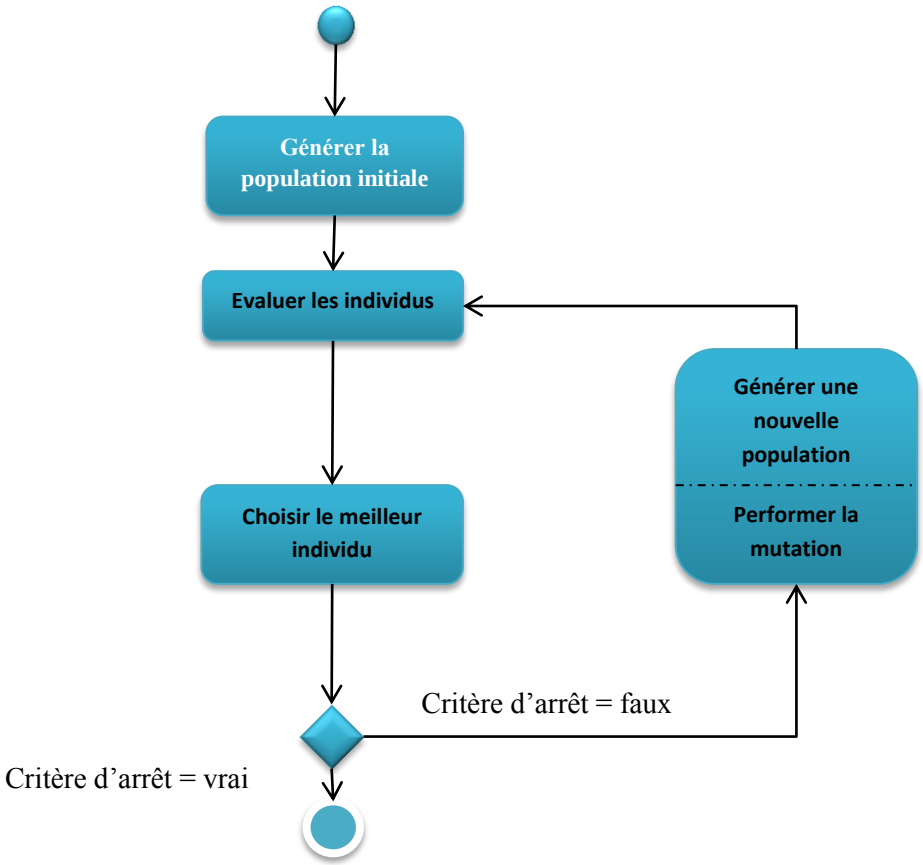


Figure 4.2. Processus d'évolution de contrôleurs.

2.3.1. La population initiale

L'algorithme commence avec une population initiale d'individus (un vecteur de vecteurs) où chaque individu est codé sous la forme d'un vecteur de paramètres (l'ensemble de poids et de seuils du RNA), les valeurs de ces vecteurs sont distribuées en utilisant la méthode d'écart-type.

On va utiliser la méthode de Box-Muller (George Edward Pelham Box et Mervin Edgar Muller, 1958) pour générer chaque paramètre ***p***. Cette méthode consiste à générer des paires de nombres aléatoires à distribution normale centrée réduite, à partir d'une source de nombres aléatoires de loi uniforme (Figure. 4.3).

$$p1 = mean + \sqrt{-2.0 \times log(u_1)} \times \sin(2\pi u_2) \times standard\_deviation$$

$$p2 = mean + \sqrt{-2.0 \times log(u_1)} \times \cos(2\pi u_2) \times standard\_deviation$$

Tel que : u_1 et u_2 sont deux variables aléatoires indépendantes uniformément distribuées dans l'intervalle]0,1].

À ce qui concerne notre approche, on va générer un seul nombre à la fois en utilisant l'équation de ***p1.***

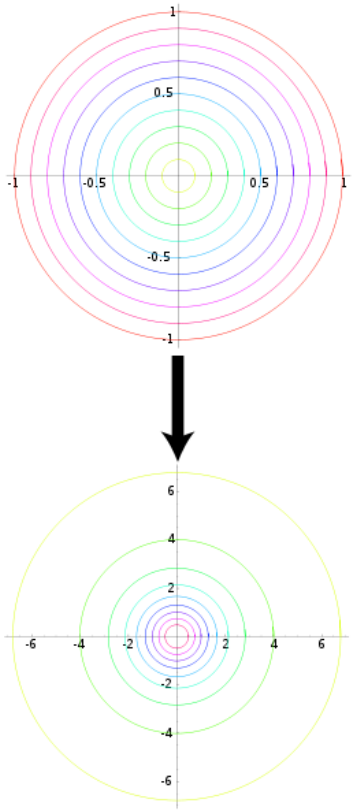


Figure 4.3. Représentation graphique de la transformation : les cercles de départ, répartis uniformément autour de l'origine, deviennent un nouvel ensemble de cercles centrés, dont la répartition est proche de l'origine puis s'étiole rapidement. Les plus grands cercles de départ correspondent aux plus petits cercles d'arrivée, et vice-versa. [BOX13].

2.3.2. L'évaluation

L'algorithme génétique est implémenté pour optimiser les contrôleurs pour les adapter aux différentes configurations des robots, afin d'évaluer les individus de chaque génération, nous avons utilisé la locomotion comme la tâche à accomplir ; c.-à-d. la fitness d'un individu donné est la distance qui a parcouru.

2.3.3. La sélection

À la fin de l'évaluation de chaque population on choisit le meilleur individu, si le critère d'arrêt n'est pas achevé, on calcule le moyen de son vecteur de paramètres et l'écart-type et génère une nouvelle population de vecteurs en utilisant l'écart-type calculé.

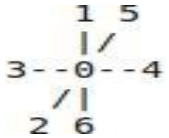
2.3.4. La mutation

Enfin, certains offsprings subissent une mutation afin d'obtenir la nouvelle génération. La mutation est obtenue en ajoutant ou en soustrayant un nombre aléatoire, ***i***, au chaque paramètre, tel que ***i*** est une valeur obtenue à partir d'une distribution normale de moyenne 0 et un écart-type.

2.4. Les mouvements

Les individus que l'on fera évoluer seront donc des vecteurs de paramètres codant des réseaux de neurones. Les sorties calculés des contrôleurs sont utilisé pour générer une instance de la classe ***Movement***, qui code les mouvements à effectuer par un module donné du robot sous forme d'un vecteur de ***Move***. Les attributs codés au sein de la classe ***Move***:

- Un attribut code pour déterminer s'il s'agit d'une rotation ou bien d'une connexion/déconnexion (on va la détailler dans le chapitre suivant). Il est de type booléen.
- Un booléen code, dans le cas d'une rotation, si celle-ci s'effectue dans le sens direct ou indirect. Dans le cas contraire cet attribut permet de faire la différence entre connexion et déconnexion.
- L'angle de rotation, dans le cas d'une rotation, reliée au premier nœud de sortie de réseaux de neurones.
- Un entier détermine le connecteur mis en jeu lors de la rotation (deuxième nœud de sortie de RNA). Les connecteurs sont ainsi codés :



- Enfin un entier code pour le module impliqué dans le mouvement. Il correspond à l'indice du module dans le robot.

Les principales méthodes sont ***start(), check() et end().*** Nous verrons plus loin comment elles s'utilisent.

Afin de tester la validité du modèle proposé pour la production d'oscillations de locomotion, on affecte chaque individu aux modules de robot (le même contrôleur pour tous les modules) et pour n pas de temps, on calcule les sorties de chaque module, et après la mise à l'échelle de la valeur de sortie dans la plage de l'actionneur, chaque module va effectuer son mouvement.

3. Simulateur

Afin d'étudier les propriétés d'un système robotique modulaire, c'est nécessaire de le simuler. Plus précisément, un simulateur permet d'évaluer les performances des individus que l'on manipule. C'est au sein de celui-ci que s'effectue l'opération de transformation de l'individu en mouvements. A l'issue de cette opération on obtient la locomotion que l'on désire évaluer.

Bien que des travaux intéressants ont été réalisés dans le domaine de la simulation de robots modulaire (comme par exemple : SYMBION and REPLICATOR [WIN09]), ces simulateurs sont tous implémentés pour modéliser et évoluer leurs systèmes modulaires et même s'il n'est pas impossible, il est très difficile de les utiliser pour implémenter notre approche, et puisque notre travail se base sur [DUT10], alors il a été jugé plus intéressant d'utiliser le simulateur développé en [DUT10] pour implémenter notre approche (contrôleur ou co-évolution) afin de mieux répondre au besoin.

Afin de plonger le robot modulaire dans un environnement le plus réaliste, le simulateur se base sur un moteur physique ; NVIDIA : PhysX. Ce puissant moteur créé à l'origine par la société AEGIA fut racheté par la suite par NVIDIA et est utilisé à l'heure actuelle dans la réalisation de plus de 150 jeux par an. Libre d'utilisation, il a l'avantage d'offrir une interface de programmation en C++ relativement simple d'utilisation fournissant des outils tels que des jointures motorisées entre les objets ce qui convient parfaitement au cadre de notre étude.

La représentation interne d'un module au sein du moteur physique se fait grâce au système d'acteurs. Les acteurs sont les objets principaux de la simulation physique réalisée par PhysX. Ils possèdent leurs caractéristiques propres telles que leur densité, la forme de leur enveloppe (leur corps), la position globale dans l'espace, l'orientation ou encore la vitesse. Les acteurs sont gérés entre autres par la classe ***PhysXNxActor***.

Puisque plusieurs classes et méthodes sont utilisées dans ce simulateur pour implémenter l'approche co-évolutionaire, on va retarder son explication pour le chapitre suivant afin de mieux comprendre les relations entre les classes et le diagramme général.

3.1. Le codage de réseaux de neurones

Il y a plusieurs bibliothèques qui implémentent les réseaux de neurones multicouches comme : *Flood* [LOP10], *OpenNN* [OPE] et *FANN* [FAN]. Toutes ces bibliothèques présentent différentes méthodes pour le codage des RNAs, leurs types, ou pour les évoluer comme une solution potentielle dans un algorithme génétique.

Pour encoder les contrôleurs de notre robot modulaire (individus de la population) nous avons utilisé la bibliothèque de réseaux neurones *« Flood [LOP10] ». Flood* a nous donné la possibilité de coder les réseaux de neurones d'une façon simple et convenable à notre approche et son intégration dans le simulateur était facile. Dans les sous-sections suivantes on va présenter les éléments inclus pour le codage d'un réseau de neurone avec *Flood.*

3.1.1. Flood

Flood est une bibliothèque open source qui implémente les réseaux de neurones utilisant le langage de programmation C++. Elle a été élaborée à la base des théories de calcul des variations. Cette bibliothèque peut être utilisée pour un ensemble varié d'applications comme par exemple : fonction de régression, la reconnaissance des formes, prédiction temporelles, le contrôle optimal, ou la conception optimale de forme. Tous ces problèmes peuvent être formulés comme des problèmes variationnels qu'un réseau de neurones peut apprendre soit à partir d'une base de données ou à partir d'un modèle mathématique. Flood offre également une solution pour les problèmes d'optimisation et d'évolution des réseaux de neurones, utilisant les algorithmes génétiques.

3.1.2. Le problème d'apprentissage

Flood implémente le *«Perceptron Multicouche»* comme un type de réseaux de neurones. Le perceptron multicouche se caractérise par un modèle neuronal, une architecture de réseau, une fonction objective associée et les algorithmes d'entrainement. Le problème d'apprentissage est ensuite formulé comme la trouvaille d'un perceptron multicouche qui permet d'optimiser une fonction objectif à l'aide d'un algorithme d'apprentissage.

Le Perceptron multicouche est un classifieur linéaire de type réseau neuronal formel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau de type *feed-forward*. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la couche de sortie correspondant toujours aux sorties du système.

Le Perceptron a été inventé en 1957 par Frank Rosenblatt au *Cornell Aeronautical Laboratory*, inspiré par la théorie cognitive de Friedrich Hayek et celle de Donald Hebb. Dans cette première version le perceptron était alors monocouche et n'avait qu'une seule sortie à laquelle toutes les entrées sont connectées.

Dans le Perceptron multicouche, les neurones d'une couche sont reliés à la totalité des neurones des couches adjacentes. La mise en place d'un Perceptron multicouche pour résoudre un problème passe donc par la détermination des meilleurs poids applicables à chacune des connexions inter-neuronales.

Le problème d'apprentissage dans le perceptron multicouche est formulé en termes de trouver une fonction qui résolvent un problème variationnel.

Figure 4.4 représente un diagramme d'activité pour le problème d'apprentissage. L'approche de résolution ici se compose de trois étapes. La première étape est de choisir un

perceptron multicouche approprié qui approxime la solution au problème. Dans la deuxième étape, le problème variationnel est formulé en sélectionnant une fonction objective appropriée. La troisième étape consiste à résoudre le problème en fonction d'un algorithme d'apprentissage capable de trouver un ensemble optimal de paramètres.

Flood implémente plusieurs méthodes pour entrainer les perceptrons multicouches comme : *la méthode de Newton, Descente de gradient, Quasi-Newton, et les algorithmes évolutionnaires,* ce dernier et un algorithme génétique standard.

Il y a deux types de procédures de mutation en *Flood* : mutation uniforme et mutation normale. Les deux opérateurs de mutation uniforme et normale sont calculés par un seul paramètre appelé l'intervalle de mutation, ***r***.

Dans la mutation uniforme, i est un nombre choisi aléatoirement de l'intervalle [0, r]. Alors que dans la mutation normale, i est une valeur obtenue à partir d'une distribution normale de moyenne 0 et un écart-type.

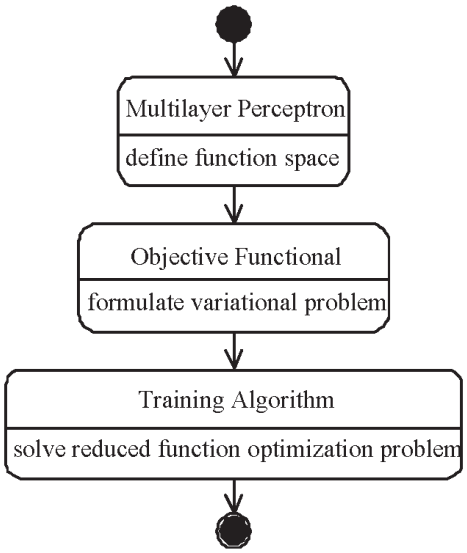


Figure 4.4. Problème d'apprentissage du perceptron multicouche. [LOP10]

3.1.3. La classe « MultilayerPerceptron »

Flood implémente le perceptron multicouche dans la classe MultilayerPerceptron qui a différents attributs, constructeurs et méthodes. Les attributs qu'on a utilisés dans notre modèle sont :

- Le nombre d'entrées.
- Les tailles des couches cachées.
- Le nombre de sorties.
- Les fonctions d'activation des couches cachées.
- La fonction d'activation de la couche de sortie.
- L'écart-type et le moyen des variables d'entrée et de sortie.
- Les valeurs minimales et maximales des variables d'entrée et de sortie.

Nous avons vu que les réseaux de neurones en *Flood* sont caractérisés par un modèle neuronal, une architecture de réseau, une fonction objective et un algorithme d'apprentissage. La caractérisation des classes de ces quatre concepts pour le perceptron multicouche est comme suit:

- *Modèle Neurone :* La classe qui représente ce concept est appelé ***Perceptron***.
- *Architecture du réseau* : La classe qui représente le concept de l'architecture de réseau dans le perceptron multicouche est appelé ***MultilayerPerceptron***.
- *Fonction Objective* : La classe qui représente ce concept dans un perceptron multicouche est appelé ***ObjectiveFunctional.***
- *Algorithme d'apprentissage* : La classe représente le concept d'algorithme d'apprentissage dans un perceptron multicouche est appelé ***TrainingAlgorithm***.

La Figure 4.5 montre le diagramme UML des classes ci-dessus avec leurs associations.

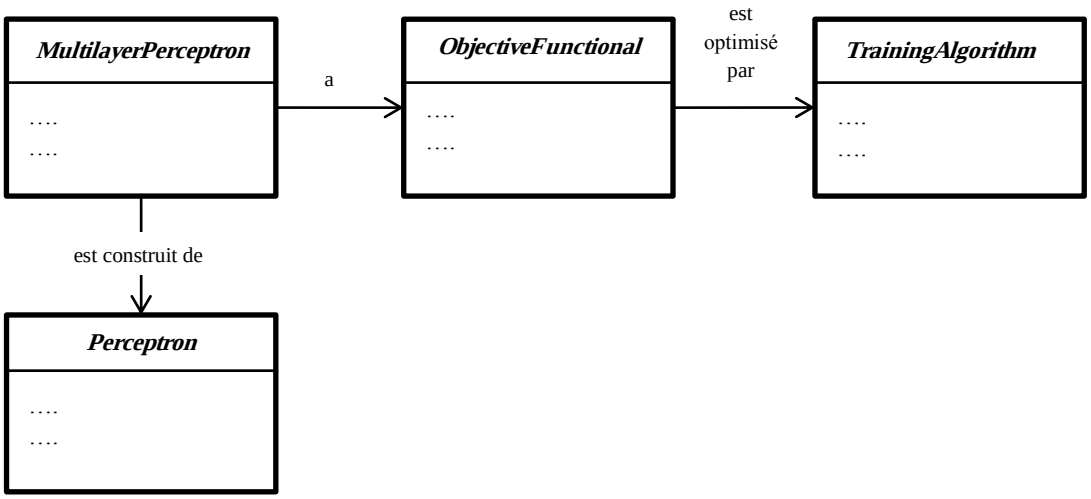


Figure 4.5. Le diagramme conceptuel du perceptron multicouche. [LOP10]

4. L'application générale

Notre but ici est d'optimiser les contrôleurs du robot pour performer la locomotion, les réseaux de neurones sont évoluer au moyen de l'algorithme génétique présenter dans ce chapitre-là.

Nous présenterons ici la structure algorithmique de l'application principale ainsi que certains détails des étapes clés. Mais pour cela il nous faudra en premier lieu mentionner quelques points sur l'architecture et le mode de fonctionnement de PhysX. Les objets principaux d'une simulation PhysX sont les acteurs. Nous aurons recours à d'autres objets.

A l'initialisation nous aurons besoin d'instancier le SDK (classe NxPhysicsSDK). Le SDK est l'objet qui regroupe tous les processus de calculs physiques. Ce sera un attribut statique de l'application. Il en sera de même pour l'attribut gScene. La scène est l'objet qui gère, entre autres, les acteurs. C'est lors de l'initialisation de la scène que l'on y ajoute un acteur un peu

particulier : le sol (Figure. 4.6.). C'est aussi à ce moment que l'on règle la gravité ambiante de la scène.

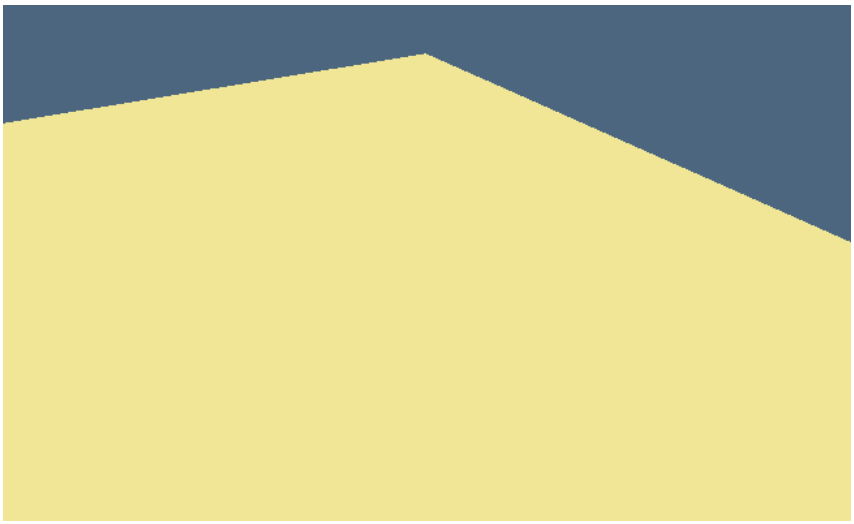


Figure. 4.6. L'environnement utilisé pour la simulation.

Après initialisation de ces différents objets, la création du robot et son ajout à la scène, on lance la boucle de simulation. A chaque itération on demande au SDK de calculer ce qui a évolué dans la scène. Nous allons nous servir de cette boucle pour créer les mouvements et faire évoluer notre simulation.

La structure globale est le parcours imbriqué d'un vecteur de vecteurs de paramètres qui représentent les réseaux de neurones (la population). On parcourt la population, vecteur par vecteur, et pour évaluer chaque individu on affecte ses paramètres aux modules du robot afin de générer les mouvements. Pour n pas du temps, chaque module calcule les sorties de son contrôleur, génère un mouvement et l'exécuter.

4.1. Génération d'un mouvement

Chaque module et après le calcul des sorties de réseaux de neurones, va générer un mouvement, un instance du classe ***Move()***, qui a comme attributs, l'angle de rotation, le connecteur qui va réaliser le mouvement.

On vérifie que le module possède bel et bien un voisin sur ce connecteur puis que l'angle appartient à l'intervalle que l'on autorise le joint à parcourir. Les valeurs des limites hautes et basses sont respectivement de plus et moins cent-vingt degrés pour les rotations des connecteurs sur les faces et le connecteur central.

On suit cette méthodes pour tous les modules, on générant ses mouvement (s'il y'en a), puis on crée un instance du classe ***Mouvement() ;*** un vecteur de ***Move(),*** et on les exécute simultanément.

Une fois les mouvements démarrés on demande au SDK de rafraichir ses données et on appelle les méthodes d'affichage du robot afin d'avoir une simulation visuelle.

Générer un mouvement

Pour (chaque module) **faire**

Calculer les sorties du réseau de neurones

générer un instance de ***Move()***

Si (L'angle $\epsilon[-120°, 120°]$ est le connecteur à un voisin) **alors**

ajouter ***Move()*** à ***Mouvement()***

Fin Si

Exécuter les mouvements simultanément

Fin Pour

Fin

4.2. Fin de mouvement

Si les mouvements générés ne produisent aucune collision entre les modules de robot, la simulation va être continuée jusqu'à la fin, si non, on considère que les contrôleurs ont échoué, on note l'individu avec ***zéro***, et on passe vers le contrôleur suivant.

Si tout s'est bien passé, on passe au cycle suivant et calcule une autre fois les sorties de réseaux de neurones et refaire le processus d'exécution.

4.3. Fonction de fitness

C'est au moment où l'on arrive à la dernier pas du temps, on évalue la performance du robot, qu'on le note. C'est donc ici que l'on choisit la fonction de fitness à utiliser. Pour un robot modulaire une de ses fonctionnalités principales est la locomotion. Afin d'arriver à traduire le but recherché par un ou plusieurs critères facilement calculables, on a choisi la distance parcourue par le centre de gravité du robot comme la fonction de fitness.

Après avoir évalué le contrôleur du robot modulaire par le biais de la fonction de fitness, on détruit le robot. Ceci revient à vider la scène de tous ses acteurs à l'exception du sol. On crée alors un nouveau robot en configuration initiale que l'on ajoute à la scène, et passer au réseau de neurones suivant.

4.4. Évolution de la population

Lorsqu'on arrive à la fin de la population, il est temps de faire évoluer la population. La première des choses à faire est de choisir l'individu (le contrôleur) qui a eu la meilleure note, en autre terme, qui a parcourue la distance maximale.

Il faut ensuite calculer l'écart-type et la moyenne de ses paramètres pour les utiliser à l'engendrement de la génération suivante comme on a expliqué dans la section précédente.

Une mutation (taux de mutation prédéfini) consiste à remplacer quelques paramètres d'individus.

Pour (chaque individu) **faire**

    Pour (tous les paramètres) **faire**

        Générer un nombre uniforme aléatoire ***p*** entre 0 et 1 ;

        Si (***p*** < Taux de mutation)

            Paramètre + ←un nombre normal aléatoire entre 0 et l'intervalle de mutation

        Fin Si

    Fin Pour

Fin Pour

Une fois la nouvelle population est générée à partir de l'ancienne, on recommence les évaluations.

5. Expérimentations et résultats

Le simulateur a été implémenté utilisant ***Microsoft Visiual Studio 2010*** sur la plate-forme de ***Windows 7 64-bit SP1*** avec un ordinateur de ***4 Go*** de RAM et un ***Intel Core i3*** possédant une horloge de ***3.07 Ghz*** et une carte graphique ***NVIDIA GeForce 210***.

Nous avons testé la validité du modèle proposé pour la production d'oscillations de locomotion. Pour n étapes de temps, les entrées ont été introduits dans le RNA de chaque module, les sorties calculées, après la mise à l'échelle des valeurs de sorties de façon appropriée à l'intervalle de l'actionneur, chaque module effectue son actionnement.

Nous avons implémenté nos contrôleurs RNA sur trois configurations différentes, les paramètres utilisés pour la simulation sont illustrés dans la « Table. 4.1 ». Nous allons présenter les configurations avec les résultats dans cette section.

Paramètres	Valeurs
Population	60
Générations	100
Taux de Mutation	1/Taille du génome
Intervalle de mutation	0.1
Nombre de pas du temps	100
La gravité	9.8

Table 4.1. Les paramètres de simulation.

5.1. Expérience 1

Pour le premier test, nous avons choisi la forme la plus simple et intuitive qui est la configuration de serpent, L'étude ici réalisée porte sur un robot composé de cinq modules assemblés à la chaîne "Figure 4.7.a ".

La simulation se déroule selon les paramètres du tableau ci-dessus, les résultats sont présentés dans le "Figure 4.7.b ".

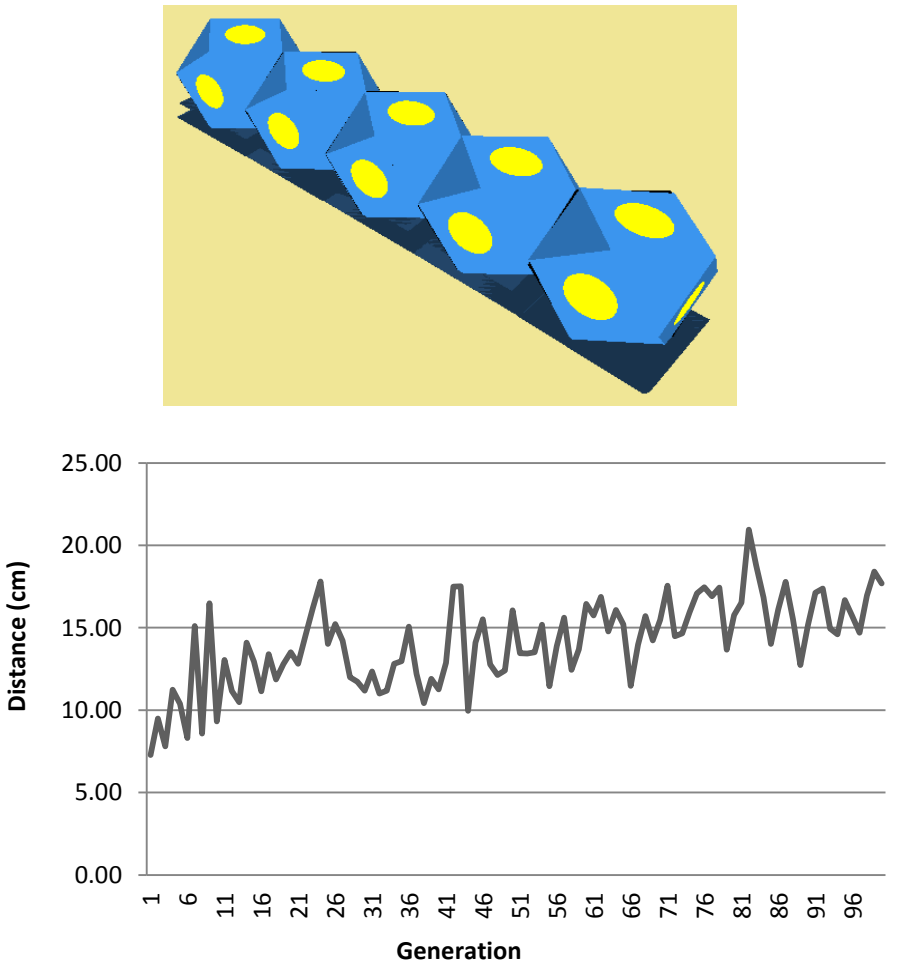


Figure. 4.7. (a). Une configuration serpent (5 modules). **(b).** Distance parcourue par l'individu le plus performant de chaque génération.

La simulation de cette configuration a duré trois heures au maximum. "La Figure 4.7.b," trace la valeur de la distance parcourue par le meilleur individu contre les générations.

L'individu le plus performant de la génération quatre-vingt-deux était capable de couvrir une distance de 20,96 cm. Et comme nous voyons dans cette courbe, le contrôleur RNA est capable de converger rapidement. La démarche de locomotion résultante est stable et cyclique dans un très courte période de temps.

La Figure. 4.9 présente des captures deux comportements évolués de deux individus pris aléatoirement et ne sont pas les meilleurs.

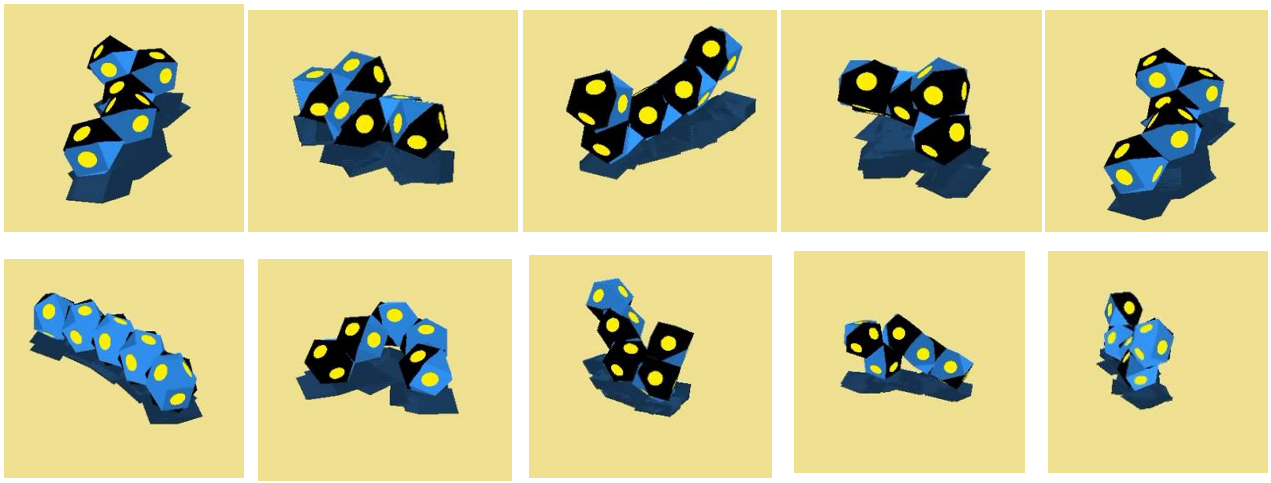
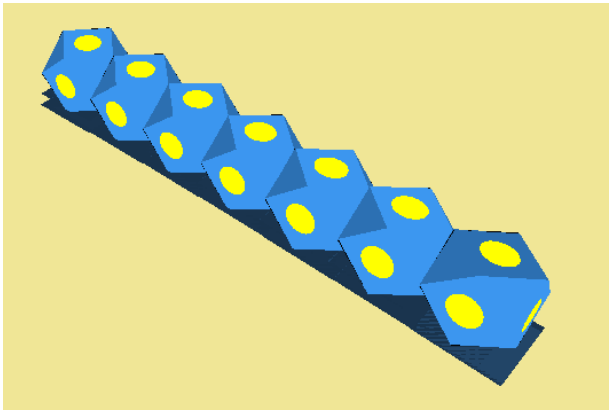


Figure. 4.8. Deux suites de captures de locomotions évoluées.

Nous avons aussi validé le même modèle par l'évolution d'une configuration serpent linéaires de sept modules "Figure 4.9.a et Figure. 4.9.b", qui a produit des locomotions avec des démarches semblables. La distance parcourue par le meilleur individu de la génération soixante-quinze est 37,12 cm.



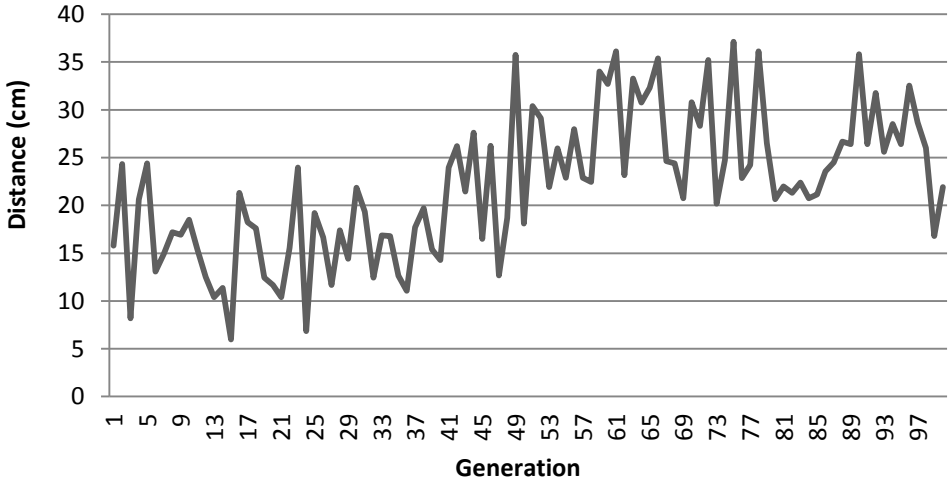
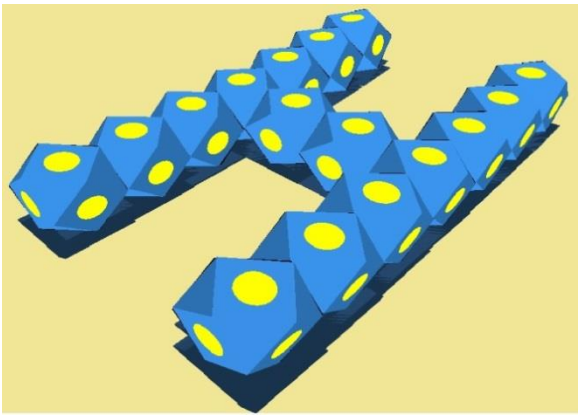


Figure. 4.9. (a). Une configuration de serpent (7 modules). **(b).** Distance parcourue par l'individu le plus performant de chaque génération.

5.2. Expérience 2

Pour le deuxième test, nous avons implémenté le contrôleur neuronal sur une configuration à quatre pattes avec seize modules, "Figure. 4.10.a", sous les mêmes conditions des deux premiers tests.

Les résultats comme indiqué dans la "Figure. 4.10.b" qui trace la distance couverte par le meilleur individu contre les générations. Le meilleur individu a pu locomote pour une distance de 42,56 cm au moment où il atteint les dernières générations. Comme il a été observé dans la première expérience, le contrôleur convergent rapidement et s'installer dans un modèle d'oscillation stable.



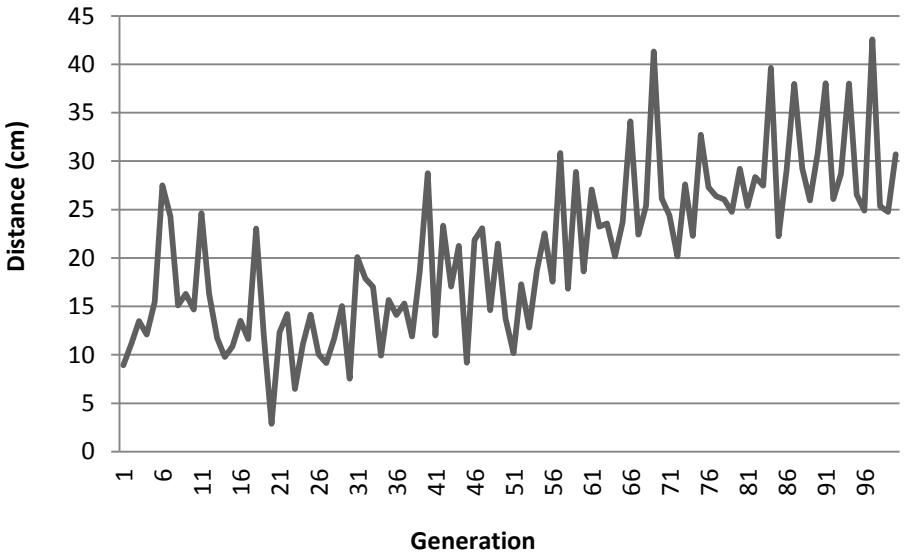
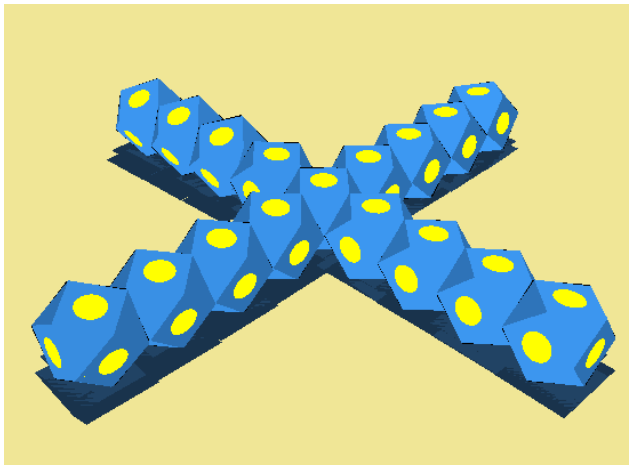


Figure. 4.10.a. Une configuration à quatre-pattes (16 modules). **b.** Distance parcourue par l'individu le plus performant de chaque génération.

5.3. Expérience 3

Comme une dernière expérience, nous avons testé le contrôleur de neurones sur une configuration d'araignée avec 17 modules, « Figure. 4.11.a », et nous déroulons le test selon les paramètres indiqués dans le tableau 1.

Les résultats sont présentés dans la "Figure 4.11.b", qui trace la distance couverte contre les générations. La meilleure distance était 31,80 cm parcourue par l'individu de la dernière génération. Le contrôleur n'a pas convergé rapidement comme pour les autres configurations, mais la démarche évolué est stable et cyclique.



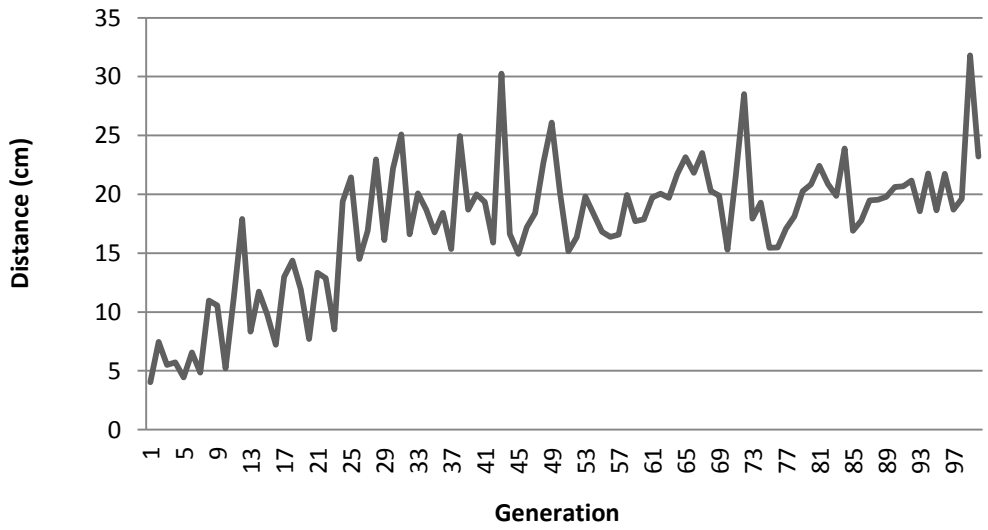


Figure. 4.11.a. Une configuration d'araignée (17 modules). **b.** Distance parcourue par l'individu le plus performant de chaque génération.

6. Discussion

Le modèle de contrôleur est capable de converger très rapidement et à s'établir dans un motif d'oscillation stable et avec une allure de locomotion stable pour les deux configurations linéaires.

Comme il a été observé dans la première expérience que les contrôleurs des deux configurations à quatre-pattes et araignée convergent et s'installent dans un modèle d'oscillation stable, mais pas assez rapide que les configurations linéaires dû à leur complexité et la difficulté de synchroniser les mouvements de quatre pattes.

L'instabilité des courbes d'évolution est due au processus utilisé pour générer de nouvelles populations. Cette instabilité n'empêche pas la production d'allures plus efficaces, puisque la courbe d'évolution augmente progressivement.

Une variété de stratégies de locomotion est évoluée, alors que, quelques allures de robots évolués semblent comme si elles étaient symétriques.

Le processus de l'évolution des contrôleurs donne les différentes configurations la possibilité de converger rapidement, en utilisant un modèle RNA simple et étudiant l'architecture neuronale minimale requise pour la locomotion.

7. Conclusion

Dans cette enquête primaire du RNA proposée pour la locomotion en robotique modulaire, nous avons validé le modèle pour produire des oscillations de locomotion dans trois configurations différentes, ce qui est le but principal de notre proposition. Un contrôleur neuronal évolué est capable de faire converger et s'installé dans un modèle oscillatoire stable, générant une démarche de locomotion stable. Nous avons testé le modèle que sur un environnement de simulation simple.

Mais les modules de robots modulaires ont leurs caractéristiques matériaux spécifiques qui posent des contraintes sur la conception de la meilleure configuration possible. Cela nous ramène à la deuxième partie de notre travail ; L'implémentation d'une approche co-évolutionnaire qui évolue la configuration et les contrôleurs d'un robot modulaire conjointement afin de surmonter l'intuition humain de la conception des morphologies, tout en gardant la séquence de mouvements pour performer l'auto-reconfiguration.

CHAPITRE 5

La co-évolution coopérative de configurations et contrôleurs

"Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid."

— Albert Einstein

1. Introduction

La robotique évolutionnaire comme une méthode bio-inspirée dans laquelle les algorithmes évolutionnaires sont utilisés pour optimiser le système de contrôle d'un robot modulaire, a été prouvé à surmonter les limites de l'intuition humaine dans la conception de stratégies de contrôle pour les machines autonomes. Cependant, la majorité des travaux a seulement optimisé les contrôleurs pour des morphologies conçues par l'humain ou bio-inspirées. Mais la fixation de la morphologie pose des limites sur les actions que le robot peut effectuer et le robot résultant n'est certainement pas la meilleure solution possible pour sa mission.

Dans ce chapitre, nous proposons une approche basée sur les algorithmes co-évolutionnaires coopératifs pour concevoir des configurations et des contrôleurs pour des robots modulaires homogènes, en adressant implicitement le problème d'auro-reconfiguration, L'algorithme co-évolue deux populations; une population de séquence de motions pour rechercher une séquence de mouvements qui peut réorganiser une configuration modulaire proposée à une autre plus adaptée à une tâche définie par sa fonction de fitness et une population de RNAs homogènes pour effectuer la locomotion.

2. La co-évolution coopérative

La nature manifeste la coévolution, il n'y a pas de distinction entre un corps et un cerveau, chacun faisant son travail. L'évolution ne viendrait pas avec la morphologie et puis évolué son contrôleur. La solidarité des solutions de nature émerge de processus de co-évolution, principalement pour que les créatures survivent dans leur environnement. La co-évolution, comme nous avons présenté dans le chapitre 1, peut être soit coopérative entre ou compétitive entre plusieurs espèces ou une espèce et son environnement.

Les robots modulaires sont particulièrement appropriés à la co-évolution. La complexité de la conception de la configuration et de programmer des contrôleurs pour un robot croît exponentiellement avec le nombre de modules utilisés, et elle est impossible pour un robot de milliers de modules.

Dans cette section nous allons présenter notre approche co-évolutionnaire coopérative pour faire évoluer la morphologie et les contrôleurs simultanément pour s'adapter à la tâche de locomotion.

2.1. Le modèle proposé

La co-évolution peut optimiser le contrôleur tout en développant la structure, de cette façon, les contrôleurs s'adaptent à différentes configurations des robots, développant une large variété de configurations et de contrôleurs.

Pour évoluer la configuration et les contrôleurs, nous implémentons une version adaptée de l'algorithme co-évolutionnaire coopérative présentée par Potter et De Jong [POT00] pour obtenir des configurations et des contrôleurs bien adaptés pour une tâche et un environnement donnés.

L'algorithme commence par l'initialisation de deux populations (espèces) ; une population pour la reconfiguration sous forme de séquence de mouvements et une autre pour les contrôleurs sous forme de réseaux de neurones (on va les détaillées dans ce qui suit). Pour co-évoluer ces deux populations, nous avons suivi le processus ci-dessus (Figure 5.1.) pour les deux populations (contrôleurs / séquence de mouvement):

Pour (chaque individu) **faire**

Créer un robot;

Choisissez un individu aléatoirement de l'autre population;

Exécutez la séquence de motions pour reconfigurer le robot;

Générer les mouvements pour n étapes de temps en utilisant les contrôleurs RNAs;

Évaluer les deux individus pour leur coopération et affecter leur fitness (la distance parcourue par le robot en cm), seulement si sa valeur est plus grande que son fitness actuel (si cet individu a déjà été évalué).

Détruire le robot ;

Fin Pour

Après l'évaluation de ces deux populations, nous générons deux nouvelles populations utilisant l'algorithme génétique de chacune et refaire le même processus, jusqu'à ce que nous atteignions le nombre maximal de génération.

Dans les deux sections suivantes nous allons détaillés les deux populations utilisés et les algorithmes génétiques implémentés pour leurs évolutions.

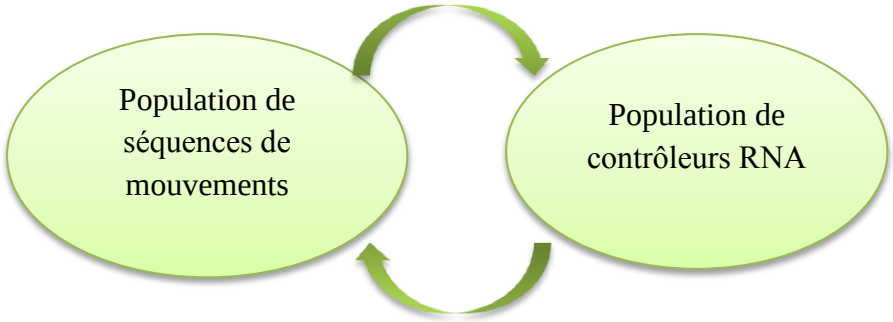


Figure. 5.1. Processus de co-évolution des deux populations.

2.2. La configuration et la reconfiguration

Comme nous avons déjà expliqué, ce travail est basé sur une recherche effectuée par Yves Duthen et al. [DUT10], qui vise à rechercher une séquence de mouvements qui peuvent réarranger une configuration modulaire proposée à une autre qui s'adapte à une tâche différente. La configuration cible n'est pas explicitement spécifié, elle est définie que par sa fonction objective (hauteur maximale). Ils ont utilisé un algorithme génétique pour évoluer la séquence optimale des mouvements qui reconfigure le robot modulaire.

Nous allons pratiquement utiliser la même approche pour l'évolution de configuration, pour reconfigurer un robot modulaire d'une configuration bio-inspiré proposé à un autre qui est plus adapté à la tâche choisie qui est la locomotion. En utilisant cette approche nous espérons à résoudre d'une façon implicite le problème de trouver la séquence optimale de mouvement pour la reconfiguration.

Les robots modulaires sont chaînés et donc un mouvement n'est pas autorisé que si elle maintient la connectivité des modules. La connectivité du robot est représentée sous forme d'un graphe.

Un module donné a une forme cubique avec six surfaces de connexion. Il est composé de deux moitiés avec un axe de rotation interne définie par les deux angles diagonalement opposés et six axes de rotation supplémentaires sur chacune des surfaces de connexion. Chaque degré de liberté à un intervalle de rotation de 120 degrés.

2.2.1. Modélisation des individus

Les individus que l'on fera évoluer seront donc des suites de mouvements de base à effectuer les uns après les autres. On peut d'ores et déjà définir quels seront ces mouvements que chaque module aura à sa disposition.

1. Pour les connecteurs de chacune des six faces on autorise, s'il est effectivement connecté et si la morphologie courante du robot le permet, une rotation de plus ou moins quatre-vingt-dix degrés.
2. Pour chaque axe de rotation interne il est possible, selon les circonstances, d'effectuer une rotation de plus ou moins cent-vingt degrés.
3. Si la morphologie du système le permet chaque module a la possibilité de se déconnecter d'un module voisin.
4. Si la configuration spatiale le permet, chaque module peut se connecter à un module voisin.

Ces opérations sont codées au sein de la classe ***Move()*** de la façon suivante :

- Un attribut code pour déterminer s'il s'agit d'une rotation ou bien d'une connexion/déconnexion. Il est de type booléen.
- Un booléen code, dans le cas d'une rotation, si celle-ci s'effectue dans le sens direct ou indirect. Dans le cas contraire cet attribut permet de faire la différence entre connexion et déconnexion.

- Un entier détermine le connecteur mis en jeu lors de la rotation.
- Enfin un entier code pour le module impliqué dans le mouvement. Il correspond à l'indice du module dans le robot.
- Un flottant qui code l'angle dans le cas de rotation.

Le constructeur de cette classe positionne de façon aléatoire les attributs du mouvement. Il suffit de lui passer en argument le nombre de modules du robot.

Les principales méthodes sont ***start(), check() et end().*** Nous verrons plus loin comment elles s'utilisent.

2.2.2. Les gènes

Le gène est l'objet principal de la modélisation d'un individu. C'est lui qui est passé au simulateur pour l'évaluation de sa fitness. Il est représenté par la classe ***Gene()***. Celle-ci possède deux attributs :

- Un tableau dynamique de pointeurs sur des mouvements (le gène à proprement parler).
- Un flottant pour la note d'évaluation de chaque individu.

Les méthodes les plus importantes de cette classe sont ***mutate() et cross().*** Comme leurs noms l'indiquent elles permettent de retourner respectivement une version mutante d'un gène et une paire de gènes offsprings issus d'un croisement entre deux gènes parents. Nous en verrons les détails plus loin.

2.2.3. La population

Les algorithmes génétiques manipulent un ensemble d'individus. Cet ensemble fait l'objet de la classe ***Population()*** (Figure. 5.2.), représentée sous la forme d'un tableau dynamique de pointeurs sur des gènes. L'intérêt de cette classe réside donc dans ses méthodes et en particulier dans l'un de ses constructeurs qui permet d'instancier une population à partir d'une autre. C'est ainsi que l'on fera évoluer la population (voir plus loin).

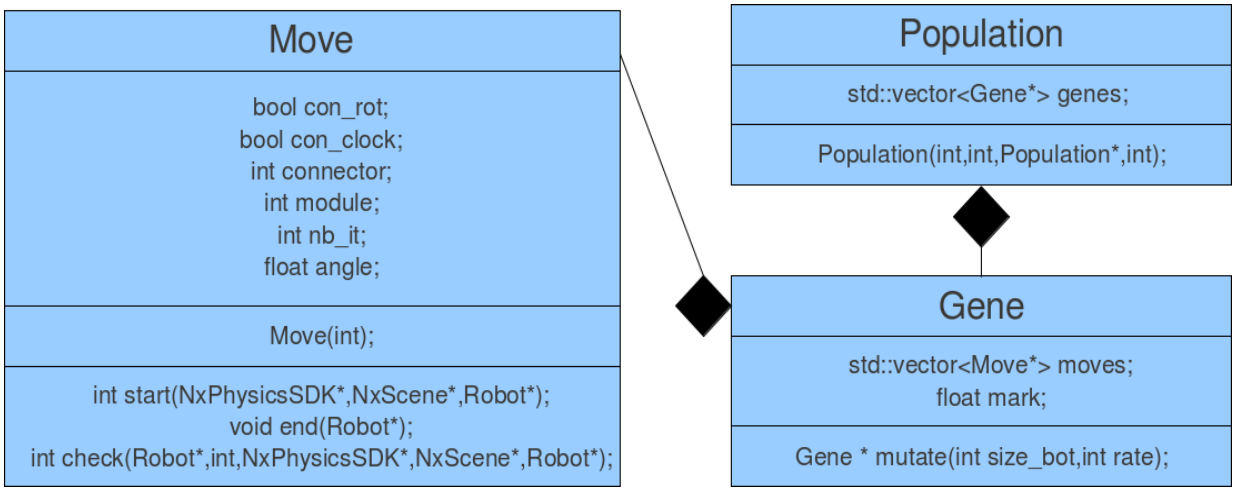


Figure. 5.2. Le diagramme UML des classes [DUT10].

2.3. Les contrôleurs

Nous allons doter ces modules cubiques avec des contrôleurs de réseaux de neurones distribués et homogènes. Nous allons étudier leur efficacité pour la locomotion. Premièrement, on reconfigure le robot modulaire puis on génère les mouvements basent sur les sorties oscillatoire de RNA simples. Nous avons utilisé un algorithme génétique pour optimiser les poids synaptiques du RNA pour produire des allures rythmiques qui maximisent la distance parcourue par le robot pour un nombre prédéfini d'étapes de temps.

Le modèle de RNA proposé est déjà présenté dans le chapitre 4 ainsi son processus d'évolution.

3. Simulateur

Afin d'étudier le mieux possible les propriétés de notre système robotique modulaire et l'approche co-évolutionnaire, c'est nécessaire de les simuler. Plus précisément, un simulateur permet d'évaluer les performances des individus que l'on manipule. Nous avons abordé le simulateur utilisé généralement dans le chapitre précèdent, mais dans ce chapitre-là nous allons le détailler pour une meilleure compréhension des méthodes utilisés.

Le simulateur se base sur un moteur physique ; NVIDIA : PhysX. La représentation interne d'un module au sein du moteur physique se fait grâce au système d'acteurs. Les acteurs sont les objets principaux de la simulation physique réalisée par PhysX et ils sont gérés entre autres par la classe ***PhysXNxActor***.

3.1. Représentation des modules

Rappellent que le module est de forme cubique et présente un connecteur permettant une rotation sur chacune de ses six faces et un axe de rotation interne défini par deux coins opposés du cube. Il sera donc modélisé par deux polyèdres avec des angles rognés qui lorsqu'ils sont reliés par le centre de leur face hexagonale et orientés de manière correcte forment un cube.

Ces polyèdres sont gérés par la classe ***HalfModule()*** dont chaque moitié de module est une instance. L'attribut principal de celle-ci est un pointeur sur un acteur. L'instanciation de l'acteur est réalisée dans le constructeur. Pour générer sa forme on utilise un maillage convexe du polyèdre (classe ***PhysX NxConvexMesh***). Une moitié de module possède de plus une couleur et une méthode permettant de le dessiner à l'écran.

En assemblant deux moitiés de module on obtient un module entier (Figure. 5.3). Cet assemblage est réalisé au sein de la classe ***Module()***. L'un des attributs de celle-ci est donc un tableau de deux pointeurs sur des moitiés de module.

Pour représenter la morphologie du robot nous avons déjà évoqué le fait que l'on utilisait un graphe. Au point de vue de l'implémentation ceci se traduit par l'utilisation d'un tableau de six pointeurs sur des modules. L'indice du module dans le tableau correspond au code du connecteur correspondant dans la modélisation des mouvements. Si un module n'a pas de

voisin sur un certain connecteur, la valeur correspondante du pointeur dans le tableau est laissée à NULL. Ce tableau est le deuxième attribut de la classe Module.

PhysX utilise des jointures motorisées entre les acteurs. C'est ainsi que nous modéliserons les connecteurs. La classe PhysX qui gère ces moteurs est ***NxRevoluteJoint()***. Le troisième attribut de la classe Module sera donc un tableau de sept pointeurs sur des joints motorisés. L'indexation correspond cette fois parfaitement à celui de la modélisation des mouvements, si un module n'a pas de voisin sur un connecteur la valeur du pointeur est laissée à NULL.

Le dernier attribut de la classe module est une instance de la classe ***MultilayerPerceprtron()*** qui définit le contrôleur pour chaque module. Lors de la simulation, on affecte un individu de la population des contrôleurs à cet attribut.

Module() a comme méthodes principales : ***CanConnect()*** qui détecte si deux modules peuvent se connecter, ***connect()*** qui réalise cette connexion, ***CanDisconnect()***, et ***disconnect()*** qui réalisent les fonctions pour la déconnexion.

Figure 5.3. Un module simulé où les cercles jaunes représentent les connecteurs.

3.2. Le robot

Le robot est composé de l'ensemble des modules. Les relations entre les modules qui définissent la morphologie du robot sont implémentées au sein de la représentation des modules.

L'attribut principal de la classe est un tableau dynamique de pointeurs sur des modules. Le robot possède aussi un attribut flottant codant pour la vitesse de rotation des moteurs, vitesse commune à tous les modules.

3.3. L'affichage

Toutes les fonctions graphiques du simulateur ont été regroupées au sein d'une même classe ***Dessin()***. Cette classe ne possède aucun attribut. Elle se résume à des méthodes d'affichage. On peut citer les deux méthodes clés: ***DessineSquellette()*** qui se contente de dessiner les arrêtes des moitiés modules et ***DessineHalfModule()*** qui synthétise l'appel à différentes méthodes qui permettent d'afficher une moitié de module plus aboutie. La couche graphique qu'ajoute cette dernière méthode n'a pour but qu'un rendu plus esthétique du robot.

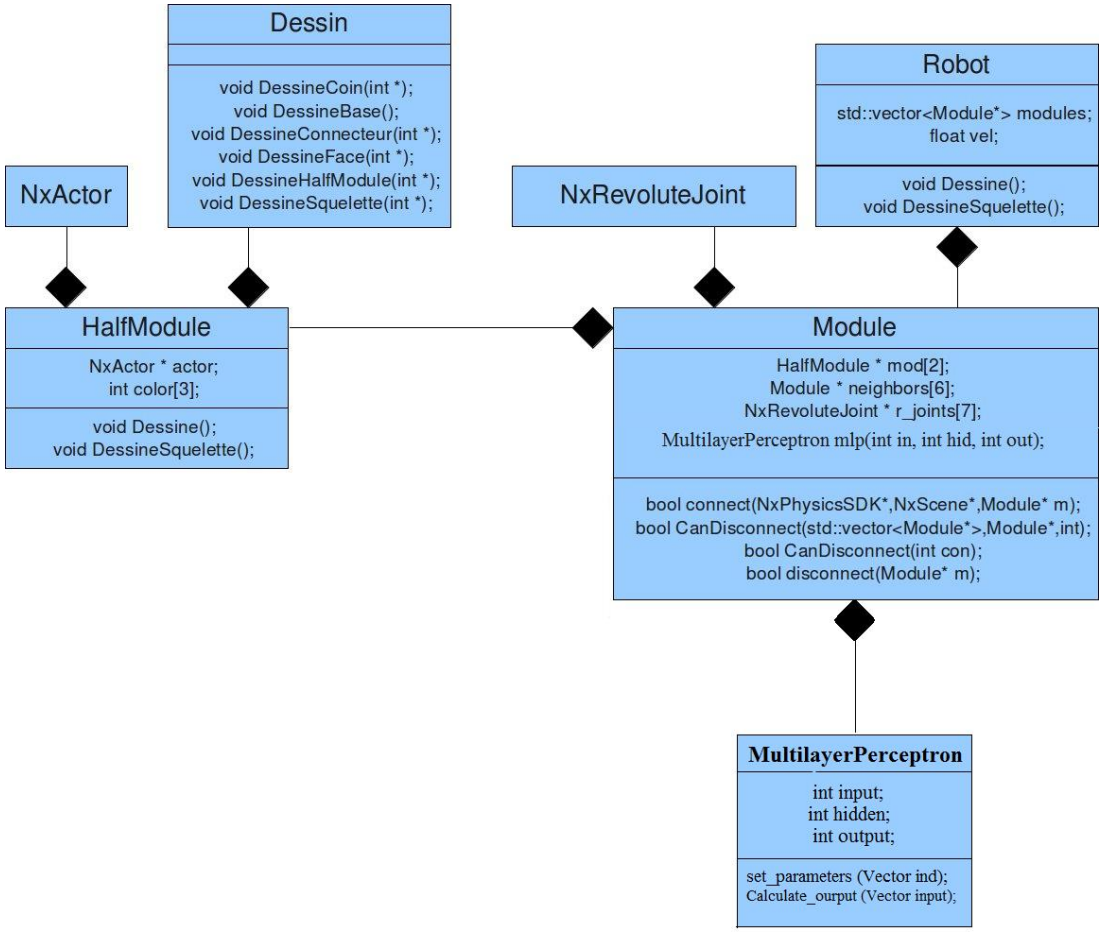


Figure 5.4. Le diagramme UML du simulateur.

4. L'application générale

Notre but ici est d'optimiser la configuration et les contrôleurs du robot pour performer la locomotion. La configuration est évoluée utilisant un algorithme génétique standard et les réseaux de neurones sont évolués au moyen de l'algorithme génétique présenté dans le chapitre 4. Nous présenterons ici quelques étapes de l'application principale.

A l'initialisation nous instancions le SDK (classe ***NxPhysicsSDK***). Le SDK sera un attribut statique de l'application. Il en sera de même pour l'attribut ***gScene.*** La scène est l'objet qui gère tous les acteurs. Lors de l'initialisation de la scène que l'on y ajoute le ***sol***. C'est aussi avec la gravité ambiante de la scène.

Après initialisation de ces différents objets, la création du robot et son ajout à la scène, on lance la boucle de simulation. A chaque itération on demande au SDK de calculer ce qui a évolué dans la scène. Nous allons nous servir de cette boucle pour créer les mouvements et faire évoluer notre simulation.

La structure globale est le parcours imbriqué de deux populations, gène par gène pour la population de configuration et vecteur par vecteur pour la population des réseaux de neurones. On parcourt la population de configuration, pour évaluer chaque individu on choisit

aléatoirement un individu de la population de contrôleur, exécute la séquence de mouvement pour la reconfiguration et après on affecte les paramètres de réseaux de neurones aux contrôleurs de modules du robot afin de générer les mouvements. Pour n pas du temps, chaque module calcule les sorties de son contrôleur, génère un mouvement et l'exécuter.

On suit le même processus pour la population des contrôleurs, choisissant un individu aléatoirement de la population des configurations.

4.1. Fonction de fitness

À la fin de l'exécution de la séquence de motions et la génération de mouvements via les contrôleurs, on passe à l'évaluation. Les deux individus vont être notés sur leur coopération pour la réalisation la tâche de la locomotion.

Après avoir évalué le système modulaire par le biais de la fonction de fitness on affecte à chaque individu son fitness, si et seulement si sa note et plus grand que son fitness actuel (si l'individu a été évalué antérieurement), et on détruit le robot. Ceci revient à vider la scène de tous ses acteurs à l'exception du sol. On crée alors un nouveau robot en configuration initiale que l'on ajoute à la scène et on passe à l'individu suivant.

4.2. Évolution des deux populations

Pour la population de configuration, la première des choses à faire est de trier les gènes au sein de la population en fonction de leur note utilisant la fonction ***sort().*** Les gènes sont repositionnés de la plus grande fitness à la plus petite.

Il faut ensuite éliminer les éventuelles redondances au sein de la population. Il se peut en effet que deux mutations sur deux gènes différents produisent le même gène. On utilise pour cela la méthode ***eliminateRedundancies()*** pour favoriser la diversité .

On doit ensuite faire évoluer la population. La première phase est toujours la même dans toute stratégie d'évolution : on sélectionne les individus qui nous servirons à engendrer de nouveaux gènes. On choisit ici de sélectionner un certain pourcentage de la population parmi les meilleurs individus. A partir de ce groupe de géniteurs on engendre de nouveaux individus, par croisement et mutation dans des proportions à paramétrer.

Une mutation consiste à remplacer un certain pourcentage (taux de mutation tiré aléatoirement) des mouvements d'un gène par un nouveau mouvement.

Un croisement entre deux gènes est ici une simple opération d'échanges deux à deux de parties du gène.

Choisir aux mieux que faire, avec quels paramètres et dans quelles proportions est souvent le fruit de l'expérience. Pour compléter notre population afin de conserver sa taille, on ajoute une petite proportion de gènes totalement nouveaux afin d'introduire de la diversité, ce qui permet d'éviter de se retrouver coincé dans une niche évolutive, et on a une nouvelle population générée à partir de l'ancienne.

Pour (chaque population) **faire**

Trier la population ;

Éliminer les redondances ;

Sélectionner des individus géniteurs;

Effectuer le croisement ;

Effectuer la mutation ;

Ajouter de nouveaux individus ;

Fin Pour

Pour la population des contrôleurs, on suivit le processus démontré dans le chapitre précédent. Après la génération des deux populations, on refaire le même processus pour évaluer la nouvelle génération.

5. Expérimentations et résultats

Nous expérimentons notre approche sur trois configurations différentes, nous avons voulu tester la validité du modèle proposé pour l'évolution des configurations plus adaptées les contrôleurs à produire des oscillations de locomotion. On a évolué les deux populations suivant les paramètres des tables 5.1. et 5.2.

Chaque paire d'individus est évalué par la distance parcourue ainsi. Une comparaison effectuée entre les résultats du chapitre 4 et ceux évolué en utilisant la co-évolution afin de montrer les différences entre les deux méthodes.

Paramètres	Taux
individus géniteurs sélectionnés à l'itération précédente	10%
individus obtenus par mutation d'un autre ensemble d'individus sélectionnés	10%
individus obtenus par croisements d'individus sélectionnés	30%
individus obtenus par mutations d'individus tirés au hasard parmi les 40% meilleurs	20%
nouveaux individus	30 %
un taux de mutation aléatoirement tiré entre	10% et 20%

Table 5.1. Les paramètres utilisés pour l'évolution de la configuration.

Paramètres	**Valeurs**
Population	60
générations	100
Taux de Mutation	1/Taille du génome
Intervalle de mutation	0.1
Nombre de pas du temps	100
La gravité	9.8

Table 5.2. Les paramètres de l'évolution des contrôleurs.

5.1. Environnement simple

Pour un premier test, nous avons implémenté notre approche dans un environnement simple et plaine avec une gravité standard. Les résultats présentés ci-dessous sont ceux de meilleures expériences, tout en présentant le minimum et la moyenne de chaque génération. Les expériences sont comme suit :

5.1.1. Expérience 1

Nous avons choisi la forme du serpent, les robots modulaires se composent de cinq modules et nous déroulons la simulation à deux reprises. La courbe ci-dessus (Figure 5.5) représente la fitness du meilleur individu de la population des configurations et celui de la population des contrôleurs, les deux fitness ; minimum et moyenne.

Le Meilleur individu a été capable de parcourir une distance de 54.47 cm à la génération quatre-vingt-quatre.

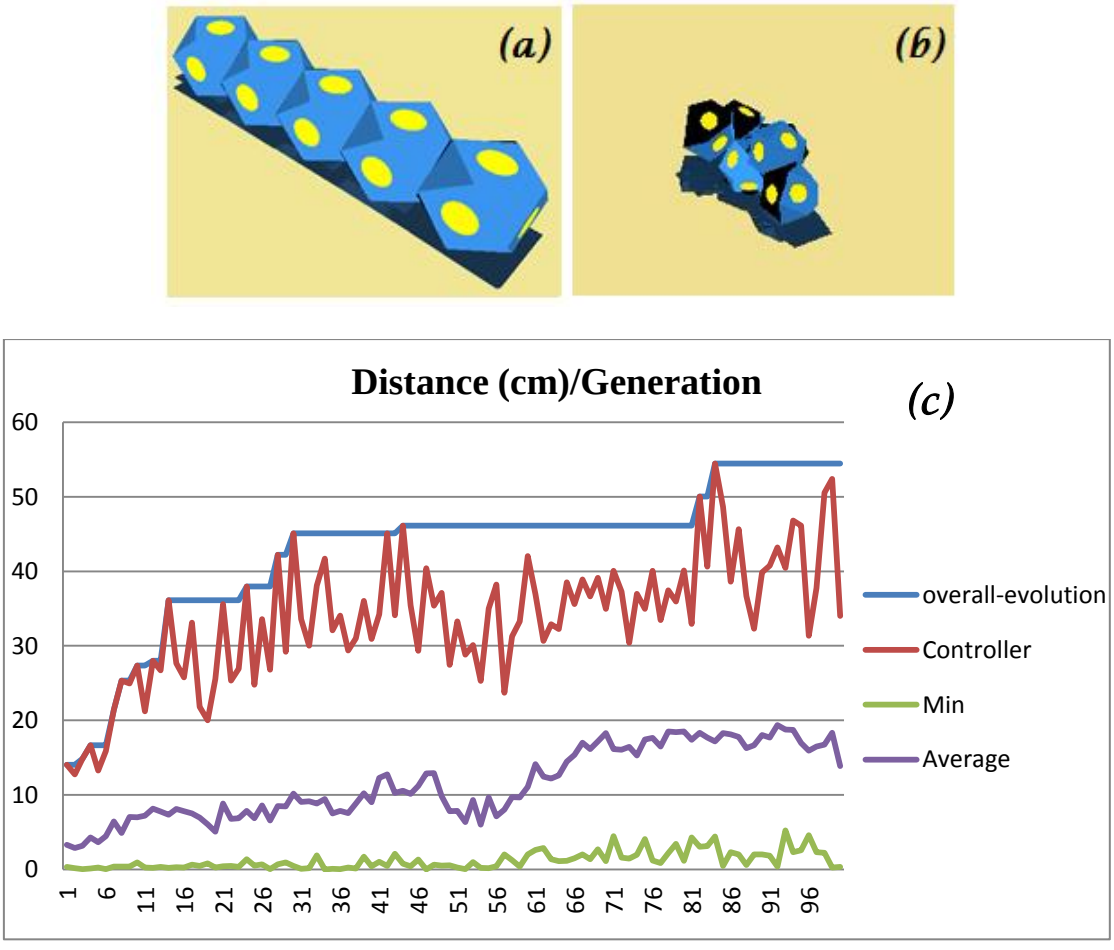


Figure 5.5. (a). Le robot modulaire (5 modules) **(b).** Une des configurations co-évoluées. **(c).** Distance parcourue par les meilleurs individus des deux populations.

Nous avons également validé le même modèle par l'évolution d'une configuration linéaire de sept modules (La figure 5.6), qui a produit une démarche de locomotion similaire à celle de la précédente avec une distance parcourue de 65.47.

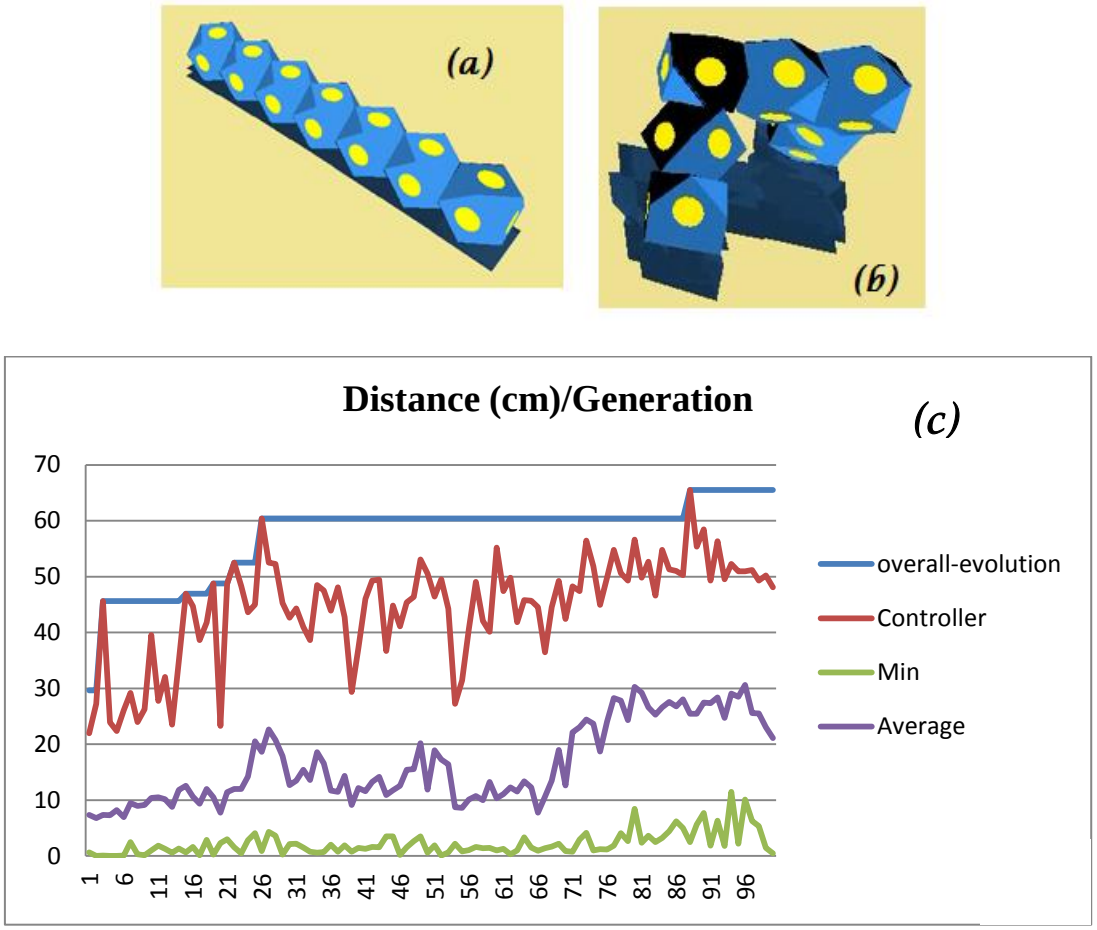


Figure 5.6. (a). Le robot modulaire (7 modules) **(b).** Une des configurations co-évoluées. **(c).** Distance parcourue par les meilleurs individus des deux populations.

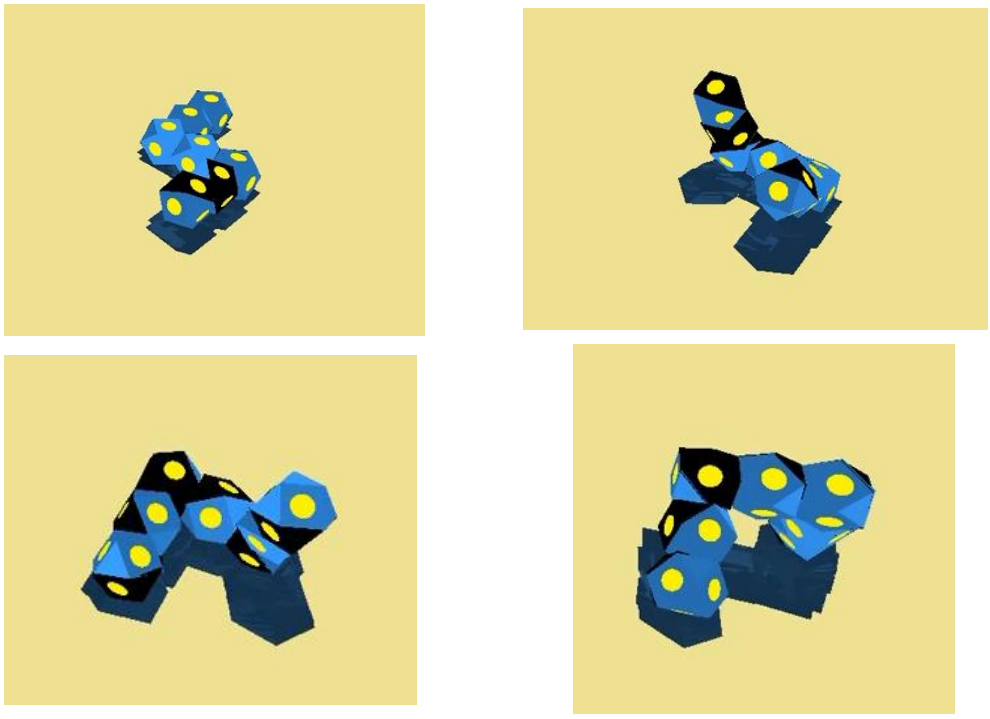
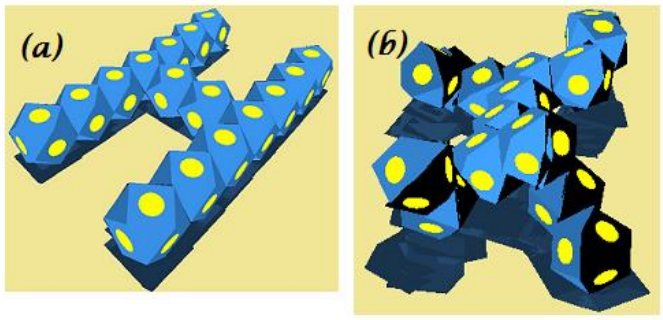


Figure 5.7. Des captures de scènes de la simulation de la configuration de 7 modules.

5.1.2. Expérience 2

Pour essayer le modèle pour la production des démarches pour des configurations complexes, nous avons implémenté notre approche dans une configuration à quatre pattes, Figure 5.8.a, Figure 5.8.b, et nous simulons le robot dans les mêmes conditions de la première épreuve et avec les mêmes paramètres.

Les résultats comme indiqué dans la figure 5.8.c qui trace la valeur de la distance parcourue contre les générations des robots modulaires, le meilleur individu a pu couvrir une distance de 69.63 cm au moment où il atteint la génération quatre-vingt-une.



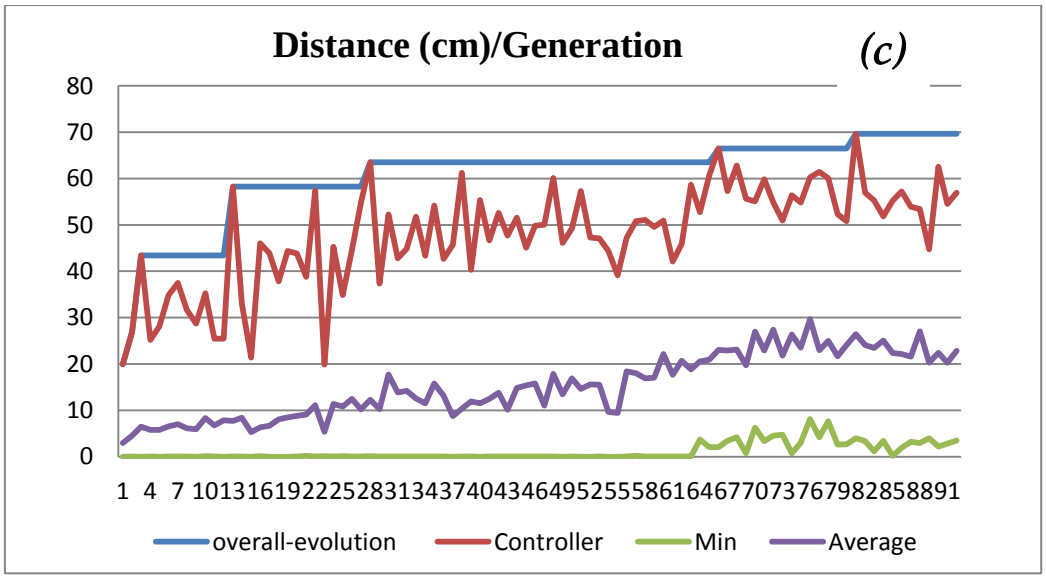
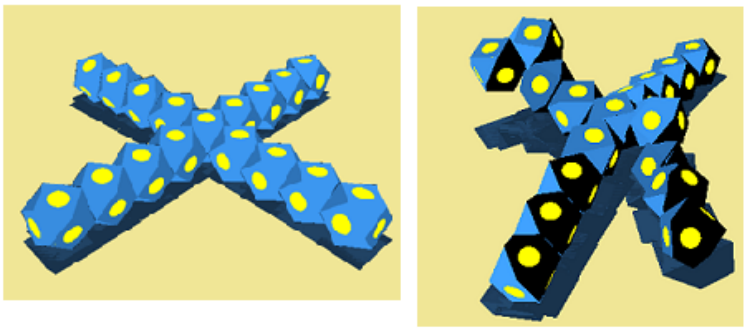


Figure 5.8. (a). Le robot modulaire à quatre pattes (16 modules) **(b).** Une des configurations co-évoluées. **(c).** Distance parcourue par les meilleurs individus des deux populations.

5.1.3. Expérience 3

Comme une dernière expérience, nous avons testé le modèle sur une configuration d'araignée, Figure 5.9.a, Figure 5.9.b, et nous suivons les paramètres du processus de l'évolution.

"Figure 5.9.c" trace la distance parcourue le meilleur individu de chaque génération. Ce robot modulaire a été capable de couvrir une distance de 94,58 cm à la génération quatre-vingt-quatre.



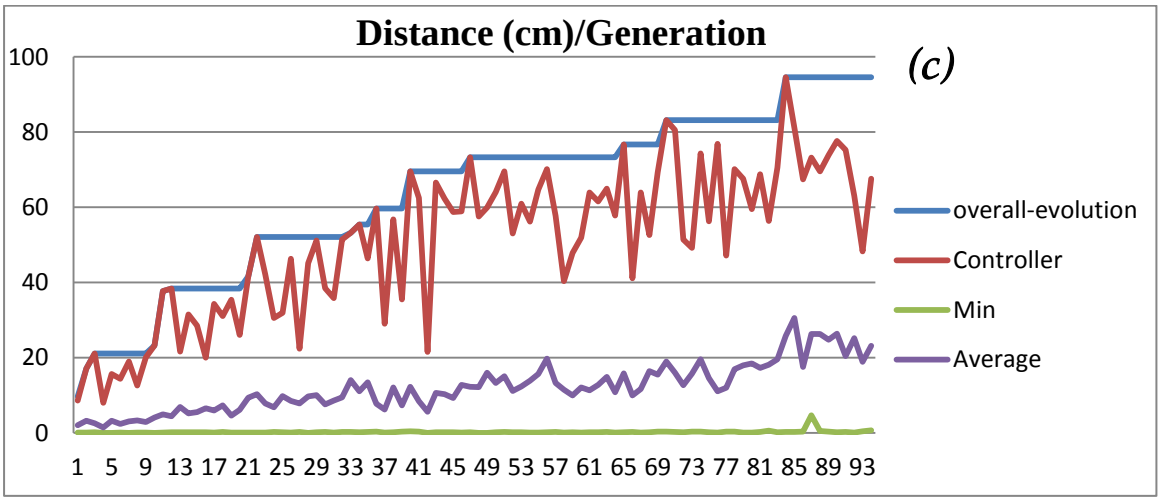


Figure 5.9. (a). Le robot modulaire araignée (17 modules) **(b).** Une des configurations co-évoluées. **(c).** Distance parcourue par les meilleurs individus des deux populations.

5.2. Environnement complexe

Le deuxième teste va être tenir dans un environnement complexe qui est implémenté au sein du PhysX (Figure 5.10). La configuration est sous forme d'un bloc (Figure 5.11.a, Figure 5.11.b), qui doit se reconfigure et évoluer une locomotion pour se mouvoir dans cet environnement. La figure 5.11.c montre la courbe d'évolution de la distance parcourue.

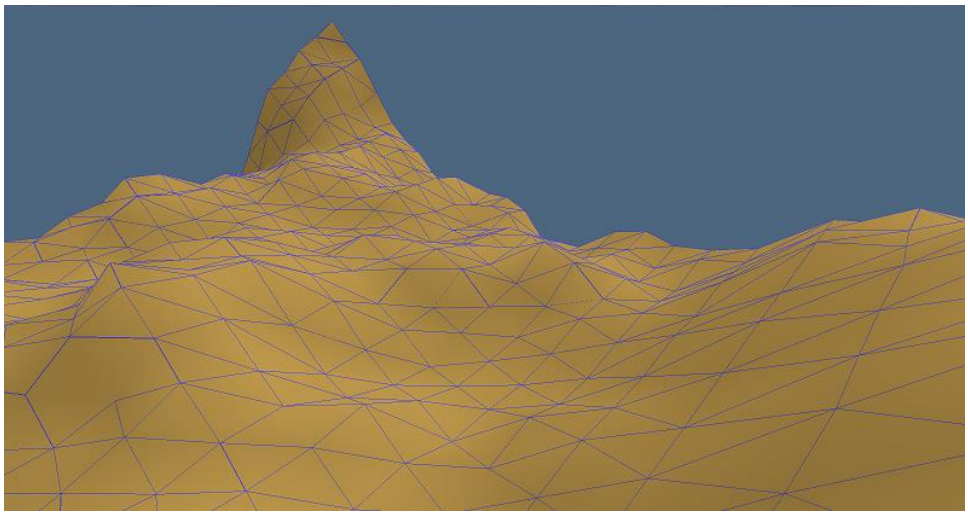


Figure 5.10. L'environnement complexe utilisé pour la simulation.

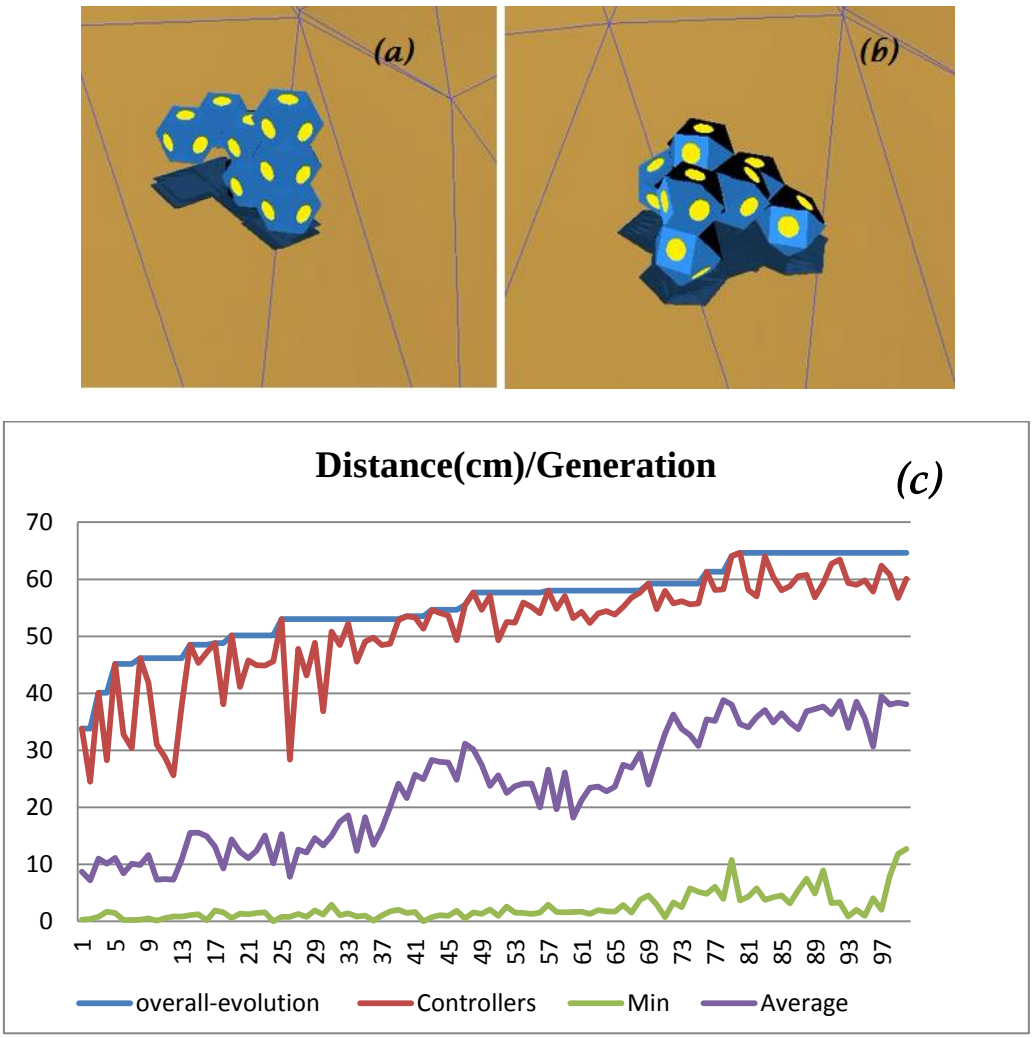


Figure 5.11. (a). Le robot modulaire (7 modules) **(b).** Une des configurations co-évoluées. **(c).** Distance parcourue par les meilleurs individus des deux populations.

5.3. Discussion

Tous les individus évolués se sont reconfigurés au moins en changeant un angle entre deux modules, mais ils n'ont pas tous été capable de se mouvoir ou se déplacer de manière significative loin de la position de départ.

La durée de la simulation varie entre les différentes configurations, et augmente graduellement avec la complexité de la configuration.

Il existe plusieurs observations importantes basant sur les résultats obtenus :

- Le contrôleur neuronal est capable de converger rapidement et s'installer dans un modèle d'oscillation stable générant une démarche de locomotion pour toutes les configurations, mais on remarque que quelques démarches de locomotion produites pour les configurations complexes, ont pu couvrir des larges distances, mais en examinant ces créatures, ils ont montré qu'ils ont reçu de forts impacts sur le sol. Ces

impacts élevés ne sont pas tolérables par le matériel réel de robot modulaire, puisque les modules pourraient se défaillir.

- Une variété de stratégies de locomotion est évoluée, alors que, quelques allures de robots évolués semblent comme si elles étaient symétriques.
- Il n'y a pas toujours des individus avec un fitness ***zéro*** à chaque génération. Mais s'il y en a, ce sont des robots avec des contrôleurs qui ne produisent pas des mouvements ou les mouvements de configuration/contrôleurs créent des collisions entre les modules de robots.
- L'instabilité des courbes d'évolution est due à l'approche utilisée pour la génération des nouvelles populations. cette instabilité n'empêche pas de produire des démarches de locomotions plus performantes, comme l'orientation générale de la courbe augmente progressivement.
- En mettant en œuvre une approche de co-évolution coopérative, nous montrons que l'évolution à la fois de la configuration et des contrôleurs sa achever de meilleurs comportements pour les robots modulaires et augmenter le taux d'adaptation en couvrant la plus grande distance. Ceci a été démontré pour les différentes configurations (Figure 5.12).
- Parfois le changement d'un angle entre deux connecteurs génère un meilleur comportement, ce qui prouve que les robots modulaires ont leurs propres caractéristiques qui nous devons prendre en considération lors de conception.
- En évolution une configuration bloque dans un environnement complexe qui s'évolue à une configuration capable de se mouvoir, on a illustré l'effectivité de l'approche proposée. En plus, les contrôleurs ont montré un comportement adaptatif contre les obstacles, en changeant la démarche de locomotion pour les surpasser.

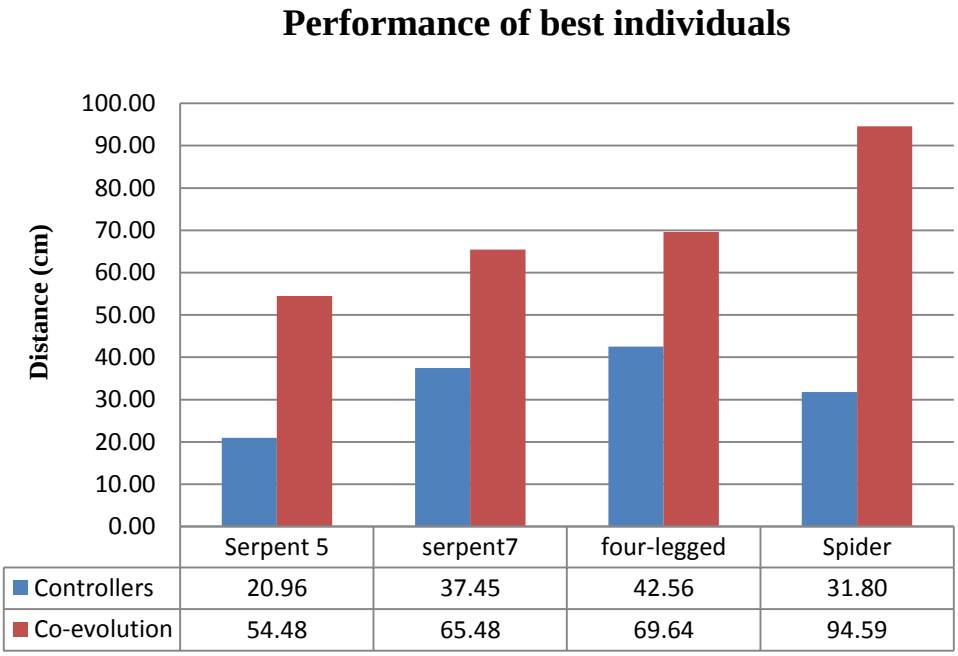


Figure 5.12. Comparaison entre les meilleurs fitness des individus évolués dans les deux approches.

- Dans un environnement complexe et en utilisant une configuration sous la forme d'un bloc, on a pu tester l'approche proposée pour la reconfiguration est la génération de la locomotion.
- Les caractéristiques du robot modulaire proposé restreignent l'espace de recherche des configurations, ce qui empêche l'évolution des configurations plus performantes. L'utilisation d'un robot modulaire hybride va peut-être résoudre ce problème et permettre L'auto-reconfiguration.

6. Conclusion

Avant de procéder à la simulation et aux tests de performance, nous avons tout d'abord présenté notre approche co-évolutionaire coopérative pour l'évolution de configuration et des contrôleurs du robot modulaire et le simulateur utilisé, puis nous avons effectué une série de tests. Ces tests nous ont permis d'évaluer les performances de notre approche par rapport à l'approche générale qui consiste à optimiser les contrôleurs pour des configurations données. À cet effet, nous avons choisi la locomotion comme la tâche à performer comme elle est l'une des fonctionnalités essentielles pour un robot modulaire.

Les résultats observés pendant ces tests de simulation, nous ont permis de déduire que la co-évolution de la configuration et les contrôleurs produit des individus plus performant et plus adaptés à une tâche donnée que ceux proposés par un être humain. Ceci a été prouvé par l'évolution de configurations différentes dans des environnements simples et complexe.

CONCLUSION

Conclusion Générale

"When I read a book I seem to read it with my eyes only, but now and then I come across a passage, perhaps only a phrase, which has a meaning for me, and it becomes part of me."
— W. Somerset Maugham

Un robot modulaire auto-reconfigurable peut être considéré comme un robot universel, car il a la capacité de simuler n'importe quel autre robot physique simplement en connectant ses modules de différentes façons pour adapter sa forme aux différentes tâches. Un tel robot peut sembler comme le produit d'une imagination fertile, mais ils sont en fait des idées envisagées sérieusement aujourd'hui à la communauté de robots auto-reconfigurables. Naturellement, nous sommes seulement au début du début de la révolution robotique, comme Isaac Asimov a dit.

La robotique évolutionnaire -l'utilisation des algorithmes évolutionnaires pour automatiser la production de robots autonomes- est un domaine de recherche actif depuis deux décennies. Mais, La pluparts des travaux dans ce domaine, se limitent à l'optimisation des contrôleurs pour des structures (morphologies) de robots modulaires conçus par l'humain ou bio-inspiré afin de réaliser une tâche prédéfinie. Cependant, ces robots doivent fonctionner dans des environnements non structurés et imprévus, et présentent un comportement intelligent et adaptatif. De quoi résulte un tel comportement intelligent? Les principes de l'intelligence artificielle incarnée stipulent qu'un tel comportement intelligent émerge de l'interaction entre le corps, le cerveau et l'environnement d'un agent [AUE13]. Un corollaire de ce concept est que la complexité des contrôleurs et de la morphologie d'un robot modulaire doit correspondre à la complexité de la tâche ou les tâches qu'il doit effectuer.

Dans ce travail, une méthode inspirée de processus de développement biologique est proposée pour faire évoluer la configuration et les contrôleurs de robots modulaires pour accomplir une tâche donnée tout en gardant la séquence de mouvements pour l'auto-reconfiguration. En implémentant une approche co-évolutionaire coopérative, nous avons essayé d'optimiser les configurations et les contrôleurs des robots modulaires.

Pour le processus d'évolution de configuration, nous codons les individus sous formes des séquences de motions, que leurs phénotypes résultent un changement dans la configuration proposée initialement. Afin d'évoluer les contrôleurs, on a utilisé une population de réseaux de neurones avec des topologies fixes, et l'évolue en utilisant des

algorithmes génétiques qui se basent sur la théorie de l'écart-type. Les robots ont été évalués dans un simulateur qui se base sur un moteur physique pour réaliser la tâche de locomotion.

Nous avons testé les deux approches ; l'approche générale qui consiste à l'optimisation des contrôleurs, et l'approche co-évolutionaire. Les expériences réalisées montrent que la co-évolution augmente la performance des configurations et évolue des robots qui sont plus adaptés à une tâche et/ou un environnement donnée, des fois par un simple changement tel que l'angle entre deux connecteurs.

Les contrôleurs neuronaux ont été capables de converger et de s'installer à des oscillations stables qui génèrent des démarches de locomotion pour toutes les configurations dans les deux environnements simples et complexes, en utilisant un modèle RNA simple, homogène et distribué qui renforce le concept de reconfiguration et d'autres caractéristiques spécifiques de robots modulaires, afin d'investiguer l'architecture neuronale minimale requise pour la locomotion.

Perspectives

L'une des choses à prendre en considération, postérieurement, est l'utilisation d'un système robotique modulaire qui offre plus de liberté soit pour la locomotion ou la reconfiguration comme par exemple un système robotique hybride pour avoir de meilleurs résultats au niveau de configurations évoluées et la réalisation de l'auto-reconfiguration.

Dans ce travail, la fonction de fitness testée c'était la locomotion, mais on n'a pas pris en compte le critère d'énergie ou l'impact de collision avec le sol, ceux qui peuvent être ajoutés pour l'obtention des résultats plus réalistes. Cependant, le système n'a pas encore mis au point une limitation sur les critères de fonction d'adaptation. Par conséquent, les robots peuvent être co-évolués pour des tâches différentes, il suffit de modifier le contrôleur et/ou ajouter d'autres types de robots modulaires.

Il y a aussi la possibilité d'évoluer à la fois la topologie et le poids des réseaux de neurones en utilisant la méthodologie NEAT (*NeuroEvolution of Augmenting Topologies*); afin de complexifier l'architecture de réseaux de neurones pour une locomotion adaptative dans des configurations différentes de robots et d'évoluer les robots modulaires dans des environnements plus complexes, afin d'obtenir des résultats plus précis sur plusieurs niveaux et aborder des scénarios réels.

BIBLIOGRAPHIES

Bibliographies

[ABB04] P. Abbeel, and A. Ng, *Apprenticeship learning via inverse reinforcement learning*. In Proc. of the twenty-first international conference on Machine learning, pp. 1–8. ACM New York, NY, USA, 2004.

[AUE12] J. E. Auerbach and J. C. Bongard, "On the relationship between environmental and morphological complexity in evolved robots", In Proc. of the Genetic & Evolutionary Computation Conf., ACM, pp. 521-528, 2012.

[AUE13] J. E. Auerbach, *The evolution of complexity in autonomous robots*, PhD thesis, university of Vermount, May 2013.

[BAL96] K. Balakrishnan and V. Honavar, *On sensor evolution in robotics*. In GECCO '96: Proc. of the First Annual Conference on Genetic Programming, Cambridge, MA, USA, pp. 455–460. MIT Press, 1996.

[BON04] J. C. Bongard and H. Lipson, *Once more unto the breach, automated tuning of robot simulation using an inverse evolutionary algorithm*, Procgs. of the Ninth Int. Conference on Artificial Life (ALIFE IX), 2004.

[BON06] J. Bongard and R. Pfeifer, *How the Body Shapes the Way We Think: A New View of Intelligence*, Bradford Books, 2006.

[BON07] J. Bongard, and H. Lipson , *Automated reverse engineering of nonlinear dynamical systems*. Proc. of the National Academy of Science 104(24), 9943–9948, 2007.

[BON09] J.C. Bongard, *The impact of jointly evolving robot morphology and control on adaptation rate*, In: Proc. of the 2009 Genetic and Evolutionary Computation Conference, pp. 1769–1770, 2009.

[BON11] J.C. Bongard, *How evolution shapes the way roboticists think*, Proc. Computer Science, The European Future Technologies Conference and Exhibition, pp. 8–10, 2011.

[BOX13] http://fr.wikipedia.org/wiki/M%C3%A9thode_de_Box-Muller, 2013.

[BRE09] N. Brener, *Analyse et Conception de Systèmes Robotiques Modulaires et Réticulaires*, Thèse de Doctorat, Université Pierre et Marie Curie, 2009.

[BRO92] R. Brooks, *Artificial life and real robots*, In European Conference on Artificial Life, pp. 3–10, 1992.

[BUA05] G. Buason, N. Bergfeldt and T. Ziemke, *Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments*, Genetic Programming and Evolvable Machines 6(1), pp. 25–51, 2005.

[CAS02] A. Castano, A. Behar and P. M. Will, *The conro modules for reconfigurable robots*, IEEE/ASME Trans Mechatron, 7(4):403–409, 2002.

[CHR06] D. J. Christensen and K. Stoy, Selecting a Meta-Module to Shape-Change the ATRON Self-Reconfigurable Robot, Proc. of the 2006 IEE International Conference on Robotics and Automation, Orlando, Florida, May, 2006.

[CHA07] A. Chavoya and Y. Duthen, *An artificial development model for cell pattern generation*. In Progress in Artificial Life, Third Australian Conference, 2007.

[DAV12] J. Davey, N. Kwok, and M. Yim, *Emulating Self-reconfigurable Robots – Design of the SMORES System*, in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4464-4469, Vilamoura, Algarve, Portugal, 2012.

[DOL07] A. Dollar and R. Howe, *Simple, robust autonomous grasping in unstructured environments*, In Proc. of the IEEE International Conference on Robotics and Automation, pp. 4693–4700, 2007.

[DOU07] R. Doursat, *Organic Computing, Chapter Organically Grown Architectures: Creating Decentralized*, Autonomous Systems by Embryomorphic Engineering. Springer-Verlag, 2007.

[DUT10] Y. Duthen, H. Luga, N. Lassabe, S. Cussat-Blanc, T. Breton and J. Pascalie, *An introduction to the Bio-Logic of Artificial Creatures*,Studies in Computational Intelligence, Vol. 321, Intelligent Computer Graphics, pp. 1-23, 2010.

[FAI13] A. Faíña, F. Bellas, F. López-Peña, R, J. Duro, *EDHMoR: Evolutionary Designer of Heterogeneous Modular Robots*, Engineering Applications of Artificial Intelligence, Elsevier Ltd, October, 2013.

[FAN] FANN, *Fast Artificial Neural Network Library*, http://leenissen.dk/fann/wp/.

[FAR89] D. F. Farmer and A. Belin, *Artificial life: the coming evolution*, In Artificial Life, pp. 815–840, 1989.

[FIT00] R. Fitch, D. Rus and M.A. Vona, *Basis for self-repair using crystalline modules*, In Proc. of intelligent autonomous systems conference, pp. 903–909, 2000.

[FLE96] K. Fleischer, I*nvestigations with a multicellular developmental model*, In Artificial Life V, pp. 229–236, 1996.

[FOG66] L. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, wiley, 1966.

[FUK88] T. Fukuda and S. Nakagawa, *Approach to the dynamically reconfigurable robotic system*, Intelligent and Robotic Systems, 1(1):55–72, 1988.

[GOL89] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[GOM08] J. G. GÓMEZ, *Modular robotics and locomotion: application to limbless robots*, Ph.D. thesis, Universidad Autónoma De Madrid, May 2008.

[GON05] J. Gonzalez-Gomez and E. Boemo, *Motion of Minimal Configurations of a Modular Robot: Sinusoidal, Lateral Rolling and Lateral Shift*, Proc. of the 8th International Conference on Climbing and Walking Robots, CLAWAR, London. pp. 667-674, September 2005.

[HAM10] H. Hamann, J. Stradner, T. Schmickl, and K. Crailsheim, *A Hormone-Based Controller for Evolutionary Multi-Modular Robotics: From Single Modules to Gait Learning*, Proc. of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, pp. 244 – 251, 2010.

[HAM11] H. Hamann, T. Schmickl and K. Crailsheim, *A hormone-based controller for evaluation-minimal evolution in decentrally controlled systems*, Artificial Life, 2011.

[HAR97] I. Harvey, P. Husbands, D. Cliff, A. Thompson and N. Jakobi, *Evolutionary robotics: the Sussex approach*, Robotics and Autonomous Systems, pp. 205–224, 1997.

[HIL90] W. D. Hillis, *Co-evolving parasites improve simulated evolution as an optimization procedure*, Physica D, 42, 228–234, 1990.

[HEU94] J. C Heudin, *La Vie Artificielle*, Eds. HERMES, 1994.

[HJE09] D. Hjelle and A. Lipson, *A Robotically Reconfigurable Truss*, In Proc. of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots (ReMAR), 2009.

[HOL75] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[HOL78] J. H. Holland and J. S. Reitman, *Cognitive systems based on adaptive algorithms*, SIGART Bull, 1978.

[HOR00] G. S. Hornby and J. B. Pollack, *Body-Brain Co-evolution Using L-systems as a Generative Encoding*, Nature, 406:974-978, 2000.

[HOR01] G. Hornby and J. B. Pollack, *Evolving L-systems to generate virtual creatures*, Computers & Graphics 25, 6, 1041–1048. 2001.

[HOU06] F. Hou, W. M Shen, *Hormone-inspired Adaptive Distributed Synchronization of Reconfigurable Robots,* In The 9th Intl. Conf. Intelligent and Autonomous Systems (IAS-9), Tokyo, Japan, March 2006.

[IJS08] A. J. Ijspeert, *Central pattern generators for locomotion control in animals and robots: A review*, Neural Networks, vol. 21, no. 4, pp. 642 – 653, 2008.

[JOR04] M. W. Jorgensen, E. H. Ostergaard and H. H. Lund, *Modular ATRON: Modules for a self-reconfigurable robot*, Proc. of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, October, 2004.

[KAM03] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Murata, and S. Kokaji, *Automatic locomotion pattern generation for modular robots,* In Proc., IEEE Int. Conf. on Robotics and Automation, pp. 714–720, Taipei, Taiwan, 2003.

[KAM05] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, *Automatic locomotion design and experiments for a modular robotic system*, IEEE/ASME Transactions on Mechatronics, 10(3):314–325, 2005.

[KOZ91] J. R. KOZA, *Evolution and co-evolution of computer programs to control independent-acting agents*, Eds. J.A. Meyer and S. W. Wilson, From Animals to Animats: Proc. of the First International Conference on Simulation of Adaptive Behavior, Paris, France, MIT Press, pp. 366–375. 1991.

[KOZ92] J. R. KOZA, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[KUR06] H. Kurokawaa ,*Self-reconfigurable M-TRAN structures and walker generation*, Robot Auton System 54:142–149, 2006.

[LAL07] S. P. Lal, K. Yamada, and S. Endo, *Evolving Motion Control for a Modular Robot*, eds. R. Ellis, T. Allen, M. Petridis, Applications and Innovations in Intelligent Systems XV, pp. 245–258. Springer, London, 2007.

[LAN89] C. Langton, *Artificial life,* In Artificial Life, pp. 1–47. 1989.

[LAN92] C. G. Langton, C. TAYLOR and S. RASMUSSEN, *Artificial Life II*, Proc. of the Workshop on Artificial Life, Santa Fe, New Mexico, February, 1992.

[LAS08] N. Lassabe, *Morphogenèse et Evolution de Créatures Artificielles*, Thèse de Doctorat, l'Université de Toulouse, 2008.

[LIP00] H. Lipson and J. B. Pollack, *Automatic design and manufacture of artificial life forms*, Nature 406. 2000.

[LIP04] H. Lipson, *How to draw a straight line using a GP: Benchmarking evolutionary design against 19th century kinematic synthesis*, Ed. M. Keijze, Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference, 2004.

[LOB09] D. Lobo, D. Hjelle and H. Lipson, *Reconfigurable Algorithms for Robotically Manipulatable Structures*, In Proc. of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots (ReMAR), 2009.

[LOP10] Lopez, R.: Flood: An Open Source Neural Networks C++ Library, www.cimne.com/flood, 2010.

[LUN03] H. H. Lund, *Co-evolving control and morphology with Lego robots*, Eds. F. Hara and R. Pfeifer, Morpho-functional machines: the new species; designing embodied intelligence, Springer, pp. 59–79, Tokyo, Japan, 2003.

[MAD12] B. Madhevan and M. Sreekumar, *Structures and Characteristics in Reconfigurable Modular Robots,* Advances in Reconfigurable Mechanisms and Robots I, ed. J. S. Dai et al., Springer-Verlag London, 2012.

[MAR98] A. Mark, D. Polani and T. Uthmann, *A framework for sensor evolution in a population of braitenberg vehicle-like agents,* In ALIFE: Proc. of the sixth international conference on Artificial life, Cambridge, MA, USA, pp. 428–432. MIT Press, 1998.

[MAR04] D. Marbach and A. J. Ijspeert, *Co-evolution of configuration and control for homogenous modular robots*, In Proc. of the eighth conference on intelligent autonomous systems, pp. 712-719, 2004.

[MCK90] M. Mckenna and D. Zeltzer, *Dynamic simulation of autonomous legged locomotion*, In SIGGRAPH '90: Proc. of the 17th annual conference on Computer graphics and interactive techniques, 1990.

[MIC06] T. Miconi and A. Channon, *The n-strikes-out algorithm: A steady-state algorithm for coevolution*, Evolutionary Computation, Vancouver, BC, pp. 1639 – 1646, 2006.

[MUR00] S. Murata, K. Tomita, E. Yoshida, H. Kurokawa and S. Kokaji, *Self-reconfigurable robot-module design and simulation*, In Proc. 6th Int. Conf. on Intelligent Autonomous Systems, Venice, Italy, pp. 911–917, 2000.

[MUR02] S. Murata, E. Yoshida, A. Kamimura et al., M-TRAN: *self-reconfigurable modula rrobotic system*, IEEE/ASME TransMechatron 7(4):431–441, 2002.

[MUR07] S. Murata and H. Kurokawa, *Shape-Changing Cellular Robots Can Exceed Conventional Robot Flexibility*, IEEE Robotics & Automation Magazine, pp. 71–78, March 2007.

[NOL94] S. Nolfi, D. floreano, O. Miglino and F. Mondada, *How to evolve autonomous robots: different approaches in evolutionary robotics*, Proc. of the International Conference Artificial Life IV, 1994.

[NOL00] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology Intelligence and Technology*, Cambridge, MA, USA: MIT Press, 2000.

[OPE] OpenNN, *open neural networks library*, http://flood.sourceforge.net/.

[OST04] E. H. Ostergaard, *Distributed control of the ATRON self-reconfigurable robot*, PhD thesis, Maersk McKinney Moller Institute for Production Technology, University of Southern Denmark, 2004.

[OST06] E.H. Ostergaard, K. Kassow, R. Beck and H.H. Lund, *Design of the ATRON lattice-based self-reconfigurable robot*, Autonomous Robots, vol. 21, 2:165-183, 2006.

[PAU01] C. Paul and J.Bongard, *The road less travelled: Morphology in the optimization of biped robot locomotion*. In International Conference on Intelligent Robots and Systems, Vol. 1, 2001.

[POT97] M. A. Potter, *The design and analysis of a computational model of cooperative coevolution*, Ph.D. dissertation, Goerge Mason Univ., Fairfax, VA, 1997.

[POT00] M. A. Potter and K. A. De Jong, *Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents*, Evolutionary Computation, Vol.8, n.1, pp.1-29, 2000.

[POU10] S. Pouya, J. van den Kieboom, A. Spröwitz,nd A. J. Ijspeert, *Automatic Gait Generation in Modular Robots: to Oscillate or to Rotate? that is the question*, Proc. of IEEE/RSJ IROS, Taipei, Taiwan. 18-22, 2010.

[RAN11] A. Ranganath, J. Gonzalez-Gomez and L. M. Lorente, *A Distributed Neural Controller for Locomotion in Linear Modular Robotic Configurations*, Book chapter (VII), ISBN: 978-84-7484-238-8. Centro de automática y Robótica CSIC-UPM, pp. 129-144, 2011.

[REC65] I. Rechenberg, *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment Library Translation, 1965.

[ROM13] J. Romanishin, K. Gilpin and D. Rus, *M-Blocks: Momentum-Driven, Magnetic Modular Robots*, In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November, 2013.

[SAL01] B. Salemi, W. M. Shen, P. Will, *Hormone-Controlled Metamorphic Robots*, International Conference on Robotics and Automation. Seoul, Korea, 2001.

[SAL04] B. Salemi, P. Will, and W.M. Shen, *Autonomous discovery and functional response to topology change in self-reconfigurable robots*, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2667–2672, Sendai, Japan, 2004.

[SIM94a] K. Sims, *Evolving 3d morphology and behavior by competition*, In Artificial Life I, pp. 28–39. Artificial Life: MIT Press, 1994.

[SIM94b] K. Sims, *Evolving virtual creatures*. In Siggraph'94, pp. 15–22. 1994.

[SHE00] W. M. Shen, B. Salemi, P. Will, *Hormones for Self-Reconfigurable Robots*, Intelligent Autonomous Systems, Venice, Italy. pp. 918–925, 2000.

[SHE02] W.M. Shen, B. Salemi, and P. Will, *Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots*, IEEE Transactions on Robotics and Automation, 18(5):700–712, 2002.

[SHE06] W.M. Shen, M. Krivokon, M. Rubenstein, C.H. Chiu, J. Everst, and J. B. Venkatesh, *Multimode locomotion via self-reconfigurable robots*, Autonomous Robots, 20(2) : 165–177, 2006.

[SHE10] W. M. Shen, F. Hou, M. Rubenstein, H. Chiu and A. Kamimura, *Recent Progress of SuperBot*, Proc. of the IEEE 2010 International Conference on Robotics and Automation workshop, Alaska, AK, USA, 2010.

[SHI03] Y.S. Shim and C. H. KIM, *Generating flying creatures using body-brain co-evolution*, Proc. of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2003.

[SPR08] A. Sproewitz, R. Moeckel, J. Maye and A. J. Ijspeert, *Learning to move in modular robots using central pattern generators and online optimization*, Int. J. Rob. Res., vol. 27, num. 3-4, pp. 423—443, 2008.

[SPR10] A. Sproewitz, P. Laprade , S. Bonardi, M. Mayer, R. Moeckel, P. A. Murdry, and A.J. Ijspeert, *Roombots – Towards Decentralized Reconfiguration with Self-Reconfiguring Modular Robotic Metamodules*, Proc. of the 2010 IEEE/RSJ International Conference on Intelligent Robot System, pp. 1126-1132, 2010.

[STO02] K. Støy, W. M. Shen and P. M. Will, *Using role-based control to produce locomotion in chain-type self-reconfigurable robots*, IEEE/ASME Trans Mechatron 7(4):407–410, 2002

[STO10] K. Stoy, D. Brandt and D. J. Christensen, *Self-Reconfigurable Robots: An Introduction*, Intelligent Robotics and Autonomous Agents series, ed. by R. C. Arkin, MIT Press, 2010.

[SUT99] R. Sutton and A. Barto, *Reinforcement learning*, Journal of Cognitive Neuroscience 11(1), 126–134, 1999.

[TAS08] Y. Tassa, T. Erez, and W. Smart, *Receding horizon differential dynamic programming*, Advances in Neural Information Processing Systems 20, 1465–1472, 2008.

[UNS01] C. Unsal, H. Kiliccote and P. Kohsla, *A modular self-reconfigurable bipartite robotic system: implementation and motion planning*, Auton Robot 10(1):23–40, 2001.

[WIN09] L. Winkler, H. Wörm, *Symbricator3D – A Distributed Simulation Environment for Modular Robots*, In Proc. of the 2nd Interntionnl Conference on Intelligence Robotics and Applicatins (ICIRA '09), pages 1266-1277. Springer-Verlag, Berlin, Heidelberg, 2009.

[YIM93] M. Yim, *A reconfigurable modular robot with many modes of locomotion*, Proc. JSME Int. Conf. on Advanced Mechatronics, pp. 283–288, Tokyo, Japan, 1993.

[YIM94] M. Yim, *Locomotion with a unit-modular reconfigurable robot*, PhD thesis, Department of Mechanical Engineering, Stanford University, Stanford, CA, 1994.

[YIM01] M. Yim, S. Homans and K. Roufas, *Climbing with snake-like robots*, Proc. IFAC Workshop on Mobile Robot Technology, Jejudo, Korea, 2001.

[YIM02] M. Yim, Y. Zhang, K. Roufas et al., *Connecting and disconnecting for chain self-reconfiguration with polybot*, IEEE/ASME Trans Mechatron 7:442–451, 2002.

[YIM07] M. Yim, W.M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins and G. S. Chirikjian, *Modular Self-Reconfigurable Robot Systems – Challenges and Opportunities for the Future*, IEEE Robotics and Autonomation Magazine, pp. 43–53, March, 2007.

[YOS03] E. Yoshida, S. Murata A. Kamimura, K. Tomita, H. Kurokama and S. Kokaji, *Self-Reconfigurable Modular Robots*, Hardware and Software Development in AIST, Proc. of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing (RISSP 2003), 2003.

[YUN08] S. Yun and D. Rus, *Optimal Distributed Planning for Self Assembly of Modular Manipulatots*, In Proceedings of International Conference on Intelligence Robots and Systems (IROS '08), pp. 1346-1352, 2008.

[ZHA09] H. Zhang, J. González-Gómez and J. Zhang, *A New Application of Modular Robots on Analysis of Caterpillar-like Locomotion*, Proc. of the IEEE International Conference on Mechatronics, Malaga, Spain, April 2009.

[ZYK05] V. Zykov, E. Mytilinaios, B. Adams and H. Lipson, *Self-reproducing machines*, Nature 435, pp. 163–164, 2005.

[ZYK08] V. Zykov, W. Phelps, N. Lassabe and H. Lipson, *Molecubes Extended: Diversifying Capabilities of Open-Source Modular Robotics* In Self-Reconfigurable Robots Workshop (IROS '08), 2008.