Créatures Artificielles : Développement d'Organismes à partir d'une Cellule Unique

Sylvain Cussat-Blanc

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse Capitole - Toulouse 1
Discipline ou spécialité : Informatique

Présentée et soutenue publiquement par Sylvain CUSSAT-BLANC
Le 17 novembre 2009

Créatures Artificielles : Développement d'Organismes à partir d'une Cellule Unique

---

JURY

Rapporteurs :	Marc SCHOENAUER, Directeur de recherche, INRIA, Université Paris Sud Jacques TISSEAU, Professeur, Université Européenne de Bretagne, Brest
Examinateurs :	René DOURSAT, Directeur de recherche, Institut des Systèmes Complexes, Paris Stéphane DONCIEUX, Maître de conférences, Université Paris VI, Paris Yves DUTHEN, Professeur, IRIT, Toulouse Hervé LUGA, Maître de conférences, IRIT, Toulouse

---

École doctorale : Mathématiques-Informatique-Télécommunications de Toulouse (MITT)
Unité de recherche : IRIT - UMR CNRS 5505
Directeur de thèse : Pr. Yves DUTHEN
Co-directeur de thèse : M. Hervé LUGA

Remerciements

Je remercie en premier lieu Luis Fariñas Del Cerro pour m'avoir accueilli à l'Institut de Recherche en Informatique de Toulouse et Jean-Pierre Jessel pour son accueil au sein de l'équipe VORTEX.

Je tiens aussi à exprimer mes sincères remerciements aux membres de mon jury pour les remarques qu'ils ont pu apporter durant la relecture et la soutenance de ma thèse. Merci à René Doursat pour avoir examiné mon travail et présidé de mon jury, à Marc Schoenauer et Jacques Tisseau pour l'avoir rapporté et à Stéphane Doncieux pour avoir examiné mon travail.

Je voudrais tout particulièrement remercier Yves Duthen, mon directeur de thèse, et Hervé Luga, mon co-encadrant, pour l'aide qu'ils m'ont apportée tout au long de ces années. Merci de m'avoir fait confiance lorsque je vous ai présenté mon projet au début de ma thèse et de m'avoir ensuite supporté et guidé pour arriver à ce qu'il est aujourd'hui.

Je remercie également l'ensemble des personnes qui travaillent à l'IRIT-UT1 et qui a constitué un environnement de travail si sympathique : Chantal, Christophe, Clément, David, Eric, Franck, Hamdi, Jean-Marc, Jonathan, Julien (x2), Laurent, Marco, Nathalie, Nicolas (x2), Paul, Rémi, Ronan, Samuel, Souad, Stéphane, Sylvie, Thomas, Tristram, Vincent et tous ceux qui n'ont pas leur nom dans la liste précédente et qui pense devoir être remerciés dans ce paragraphe.

Je désire aussi remercier tous mes amis qui ont dû me supporter et qui devront le faire encore longtemps (je l'espère du moins). Un remerciement tout particulier à Gaëtan et Sylvain, les catalyseurs de l'humour. Je félicite Laure qui arriver à nous supporter quand nous sommes tous les trois réunis. Merci à Lionel, Morgane et Vincent pour ces longues randonnées roller (entre autres) que j'ai pu faire avec vous.

Tout ce travail n'aurait pas pu être réalisé sans l'appui de ma famille que je remercie dans tout son ensemble. Plus particulièrement, je voudrais remercier mon frère, Christophe, pour les longs débats rugbistiques et ma soeur, Aurélie, désignée correctrice orthographique officielle de ce manuscrit. Et bien évidemment, mes parents toujours présents lorsque j'ai besoin d'eux.

Enfin, merci à toi, Cornelia, sans qui la vie n'aurait pas la saveur exquise qu'elle a depuis que tu es à mes cotés.

A Céline,
Disparue bien trop tôt.

Résumé

Le développement de créatures artificielles est un domaine de recherche en plein essor. Depuis plus de vingt ans maintenant, de nombreuses techniques sont apparues afin de simuler à plusieurs niveaux des êtres artificiels : en commençant par la simulation de leur comportement au début des années 90, on a ensuite continué en modifiant leur morphologie pour qu'elle soit adaptée à leur environnement. Plus récemment, l'embryogenèse artificielle s'inspire des mécanismes de développement du vivant afin de générer de petites créatures de quelques dizaines à plusieurs centaines de cellules. Le but de ces systèmes est d'une part de mieux comprendre le vivant mais aussi de produire des modèles comportementaux pour les futurs robots modulaires.

Après avoir étudié ces différents niveaux de simulation, nous nous sommes aperçus qu'il n'existait pas de modèle transversal permettant une simulation à plusieurs échelles des créatures. Le but de ces travaux est de développer une créature complète en partant d'une cellule unique, possédant différents organes et des fonctionnalités haut niveau. Le but de cette thèse est de construire le modèle chimique de cet ensemble de simulateurs. Nous avons ainsi proposé un modèle basé sur une forte simplification du modèle de développement naturel. Les créatures devront de plus intégrer un métabolisme afin de pouvoir extraire de l'énergie des différents constituants de son environnement. Ce métabolisme est trop souvent oublié dans les modèles de développement de la littérature bien qu'il soit à la base de la vie de tous les êtres vivants.

A travers différentes expérimentations que nous avons effectuées, nous avons prouvé que ce modèle est capable de produire différents organes et de les assembler afin de créer un organisme plus complexe. Nous avons aussi montré la possibilité à produire une forme particulière. Enfin, nous avons observé d'importantes capacités d'auto-réparation inhérentes au modèle.

Ce modèle de développement est un premier simulateur qui sera inclu dans un ensemble de simulateurs agissants à différentes échelles de la créature. Comme nous le verrons dans les perspectives de ces travaux, nous avons commencé à imaginer un simulateur physique et un simulateur hydrodynamique permettant de plonger une créature en train de se développer dans un monde physique aux lois newtoniennes et un monde hydrodynamique répondant aux équations de Navier et Stokes.

Table des matières

Table des figures

Liste des tableaux

Introduction

Cadre de la thèse

Le développement de créatures artificielles est un thème de recherche âgé d'une vingtaine d'années. Plusieurs modèles permettant de simuler à des niveaux très différents ces "êtres" artificiels existent actuellement. Certains modèles se contentent de simuler seulement quelques cellules mais avec un réalisme poussé et permettent par exemple de vérifier les hypothèses formulées par les biologistes quant à l'étude des systèmes vivants. D'autres utilisent des blocs et des connecteurs entre ces blocs pour produire de grosses créatures capables de marcher, de nager ou de faire de la planche à roulettes dans leur environnement. Ici, les créatures n'ont pas vocation à simuler précisément le vivant mais plutôt à découvrir la morphologie, le contrôleur et les capacités de futurs robots modulaires.

Notre travail porte sur la simulation de développement cellulaire de créatures artificielles en partant d'une cellule unique possédant un certain nombre de caractéristiques telles que des capacités de division et de spécialisation. Il s'agit d'engendrer une variété de créatures dans un environnement permettant leur évaluation. Ce sous-domaine de l'informatique appelé embryogenèse artificielle a pour but de s'appuyer sur les mécanismes déployés durant la croissance du vivant pour produire diverses créatures adaptées à leur environnement. Les applications possibles de cette discipline sont nombreuses et semblent être prometteuses. Nous pouvons citer par exemple l'étude des propriétés biologiques du vivant, la création de robots s'inspirant du vivant ou la production de créatures virtuelles pour les jeux vidéos ou le cinéma.

Problématique

Différents modèles ont été produits pour générer des créatures artificielles. Ceux-ci utilisent différents niveaux d'abstraction pour produire des créatures de tailles et de formes variées. Alors que l'approche morphologique produit des créatures relativement grandes [Sims, 1994, Fukunaga et al., 1994, Lassabe et al., 2007], les modèles du domaine de l'em-

bryogenèse artificielle utilisent une cellule unique et un modèle de croissance pour produire un organisme complet réalisant parfois une ou plusieurs fonctions [Dellaert and Beer, 1994, Stewart et al., 2005, Chavoya and Duthen, 2008].

Cependant, aucun lien n'existe entre ces différentes échelles de simulation. Entre l'activité des cellules, des organes ou des créatures complètes, aucun lien n'existe actuellement. Or, il est possible de concevoir des simulateurs hiérarchiques qui prennent en compte plusieurs échelles et qui associent des lois physiques différentes aux différents niveaux hiérarchiques de la simulation. Nous pouvons citer par exemple les travaux de Jean Claude Torrel sur la prise en compte de plusieurs niveaux de complexité en cosmologie [Torrel, 2007]. De même, dans la simulation de foule en image de synthèse, on utilise des techniques similaires : la foule est représentée comme un liquide qui s'écoule dans son environnement quand on l'observe de loin, comme un essaim lorsqu'on se rapproche ou comme un ensemble d'individus lorsque l'observateur est à proximité de la scène.

En s'inspirant de cette approche multi-niveaux et des modèles de croissance déjà existants, le but de cette thèse est de produire un modèle permettant de développer un organisme composé de plusieurs centaines ou milliers de cellules liées au sein "d'organes". Cet organisme devra posséder un métabolisme, des capacités de régénération et respecter des contraintes morphologiques et comportementales. En faisant l'hypothèse que les blocs des créatures de Karl Sims peuvent être "remplis" par ces groupements de cellules formant des sortes d'organes, ils pourront alors se développer en parallèle afin de produire des créatures complètes, capables d'évoluer dans un environnement dans lequel la physique est simulée.

Contribution

Dans le but de créer des créatures entières composées de différents organes, nous avons développé un modèle, *Cell2Organ* [Cussat-Blanc et al., 2008a], capable de produire des organismes qui possèdent des fonctions spécifiques. Ces organismes répondent à la définition biologique d'un organe, c'est-à-dire un "ensemble normal ou pathologique de cellules d'un organisme qui ont la même fonction et présentent la même différenciation morphologique" (Source : CNRTL, dictionnaire du CNRS). Notre modèle contient un environnement avec une chimie artificielle très simplifiée [Rasmussen et al., 2003, Dittrich et al., 2001, Hutton, 2007, Ono and Ikegami, 1999] et des cellules qui accomplissent différentes actions. Ces cellules sont capables de se diviser et de se spécialiser pour optimiser certaines fonctions au détriment d'autres. De plus, nous montrons que *Cell2Organ* peut aussi produire diverses formes de créatures. Le but final de ce projet est de dévelop-

per une créature artificielle comportant plusieurs organes à partir d'une cellule unique [Cussat-Blanc et al., 2009b, Cussat-Blanc et al., 2009c]. Notre modèle peut permettre également des capacités d'auto-réparation de la structure des organes [Cussat-Blanc et al., 2009a]. Si certaines cellules de l'organe sont détruites au cours du développement ou même durant la phase de fonctionnement, le modèle est capable de régénérer la structure afin de rétablir la fonction de l'organisme.

Structure du mémoire

Le premier chapitre de cette thèse se propose de résumer succinctement une histoire de la vie sur Terre. Il décrit comment le vivant est apparu sur notre planète. Il est important de comprendre les principaux mécanismes qui ont permis à la vie d'apparaître afin de produire des modèles s'inspirant de ces mécanismes. Le chapitre conclut sur les bénéfices d'une approche bio-inspirée dans la recherche sur les nouvelles technologies.

La deuxième partie de la thèse présente un état de l'art du développement de créatures artificielles en informatique. En partant de l'observation que la complexification des créatures artificielles s'est faite parallèlement à l'augmentation de la puissance de calcul des ordinateurs, nous ferons l'historique des méthodes en commençant par la simulation comportementale, domaine pionnier de la création d'avatars, puis en présentant la morphogenèse artificielle qui a permis la modification de la morphologie des créatures artificielles et enfin, en étudiant l'embryogenèse artificielle, domaine sur lequel nous nous pencherons plus en détail dans la suite de cette thèse.

A partir de cet état de l'art, nous avons voulu développer un modèle faisant le lien entre la morphogenèse artificielle et l'embryogenèse artificielle. Pour cela, nous avons imaginé un modèle de développement artificiel qui sera présenté dans le troisième chapitre. Basé sur une forte simplification du modèle naturel de développement, il permet de produire des organismes possédant des fonctionnalités et une morphologie décrites par l'utilisateur.

Pour produire des organismes intéressants, ce modèle utilise un algorithme génétique pour trouver des génomes répondants aux spécifications de l'utilisateur. Ce type d'algorithme d'optimisation demande cependant beaucoup de puissance de calcul. Le chapitre 4 présente quelques méthodes que nous avons mises en oeuvre dans le but de réduire le temps de calcul de cet algorithme. Nous avons dans un premier temps parallélisé l'algorithme génétique en utilisant une grille de calcul et le middleware ProActive [Caromel et al., 2006]. La seconde étape a consisté à améliorer la convergence de l'algorithme. Nous avons testé des méthodes d'ajustement des paramètres afin d'adapter les paramètres de l'algorithme génétique au meilleur résultat en cours. Nous avons aussi mis en place une méthode d'aide

à la convergence permettant à l'algorithme d'être guidé dans sa recherche de solution.

A travers différentes expérimentations, le chapitre 5 présente les différents organismes que nous avons obtenus grâce à ce modèle de développement. Dans une première partie, nous présenterons trois organes différents ainsi que les capacités d'assemblage de ces organes. Après avoir montré les capacités fonctionnelles du modèle, nous montrerons ses capacités à produire des formes en développant des organismes possédant des morphologies décrites par l'utilisateur. Puis nous décrirons une propriété émergente de notre modèle : les capacités de régénération des créatures en cas de blessure. Enfin, ce chapitre montre les limites de notre modèle.

Dans un dernier temps, pour conclure cette thèse, nous donnerons les applications possibles des modèles de développement en biologie synthétique et pour les nano-systèmes. Nous donnerons aussi les perspectives du modèle en présentant une étude préliminaire que nous avons effectuée sur le couplage de trois simulateurs : le modèle présenté dans cette thèse, un modèle hydrodynamique permettant le déplacement de substrat dans l'environnent, et un moteur physique permettant une interaction haut niveau sur les cellules.

1

Bio-inspiration

—

Une histoire de la vie sur Terre

L'homme s'est toujours inspiré des stratégies développées par le vivant afin de résoudre des problèmes ou décrouvrir de nouveaux outils. De l'aviation à la recherche de nouveau matériaux, les exemples sont nombreux : matériaux superhydrophobes inspirés des feuilles de lotus, structures en nid d'abeilles, ailes d'avions inspirées des ailes des oiseaux, etc. Dans cette thèse, nous voulons générer des créatures en utilisant un système de développement. Essayer de comprendre le développement naturel pour en tirer une méthodologie est souvent très intéressant.

Alors que la première partie de ce chapitre s'attelle à définir la vie, la deuxième partie montre les origines de la vie sur Terre. Pour conclure ce chapitre, nous donnerons quelques exemples de technologies issues de l'observation du vivant.

1.1 Qu'est-ce que la vie ?

Aucune définition scientifique de la vie ne semble faire l'unanimité. Pourtant, une telle définition est primordiale dans différents domaines (biologie, biochimie, génétique, astrophysique, philosophie, vie artificielle...). Plusieurs définitions de la vie sont possibles, possédant chacune leurs qualités et leurs lacunes.

Une des premières définitions nous vient d'Aristote : "Par 'vie' nous entendons le fait de se nourrir, de grandir et de dépérir par soi-même" [Aristote, 330]. Il décrit alors la vie comme étant la période entre la naissance et la mort en donnant les prémices d'un

métabolisme et d'une croissance.

Une autre définition de la vie provient de la chimie darwinienne. Elle est reformulée par Joyce dans le livre de David W. Deamer en 1994 [Deamer and Fleischaker, 1994] qui la décrit comme étant "un système chimique indépendant capable de subir l'évolution darwinienne". Cette définition permet de dire que la vie possède la même origine que l'évolution darwinienne. Cependant, il est concevable qu'une période de l'évolution se soit faite sans duplication. Durant cette période, les créatures présentes dans l'environnement n'effectuaient que d'exactes duplications de leurs acides nucléiques [Dyson, 1985] et l'évolution darwinienne ne peut donc pas être prouvée. En se fondant sur cette définition, de telles créatures n'auraient pas été vivantes. Les formes de vie uniques n'entrent par exemple pas dans cette définition. Le dernier spécimen d'une espèce, incapable de se reproduire étant seul, ne peut pas être considéré comme vivant puisqu'incapable d'évoluer au sens de la définition donnée par Darwin.

Selon Claude Bernard, médecin et physiologiste du XIX^{eme} siècle, la vie est un état organisé et homéostatique de la matière. C'est-à-dire qu'un être vivant est un système capable de maintenir la stabilité de sa structure en cas de changement de son environnement. Cette définition implique des mécanismes pour que l'organisme soit capable de maintenir un état optimal dans un environnement dynamique. Ceux-ci vont engendrer des réactions chimiques qui donneront le métabolisme des organismes afin de transformer et d'utiliser l'énergie pour maintenir leur stabilité.

Ces dernières définitions possédant toutes des lacunes, on admet de nos jours que la vie peut ainsi être mise en évidence comme un ensemble de conditions. Un organisme vivant doit :

1. avoir un métabolisme,
2. être capable d'homéostasie,
3. être capable de se développer,
4. pouvoir répondre à des stimuli externes,
5. avoir le pouvoir de se reproduire,
6. grâce à l'évolution darwinienne, être capable de s'adapter à son environnement.

Cette définition est celle qui est la plus acceptée dans le monde de la biologie, mais la communauté n'est pas encore d'accord à son sujet. En effet, certaines entités sont proches du vivant mais ne respectent pas cette définition. C'est le cas par exemple des virus qui possèdent toutes les propriétés de la définition précédente sauf le métabolisme. Certains scientifiques s'accordent pourtant à dire que les virus sont des êtres vivants car ils sont

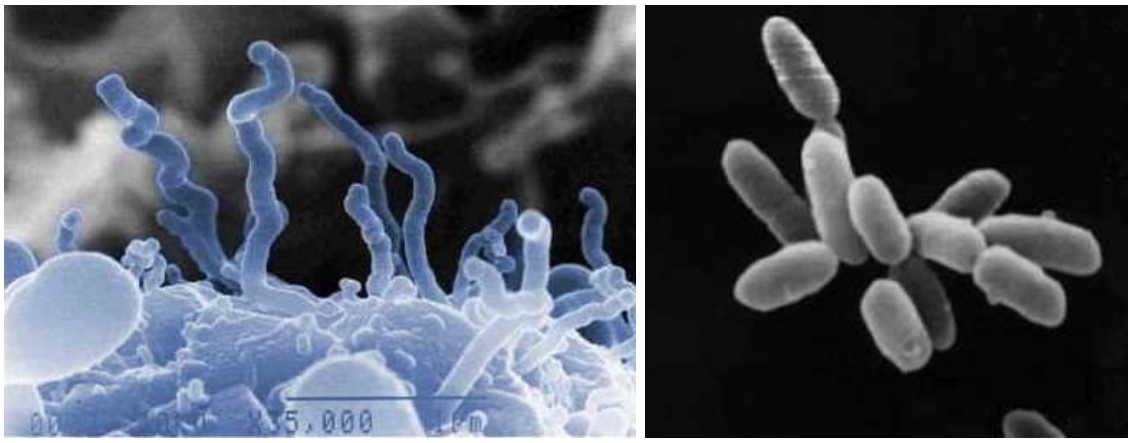


FIGURE 1.1 – *L'Eobacterium (à gauche) et la Cyanobactérie (à droite), parmi les premiers êtres vivants de la planète, sont toujours présents dans notre environnement actuel.* [1]

capables de se dupliquer. En s'appuyant sur ce type d'exemples, Cleland et Chyba s'accordent même à dire que c'est une erreur de définir la vie car une définition sera toujours trop restrictive par rapport aux possibilités des êtres vivants [Cleland and Chyba, 2002].

1.2 Evolution de la vie sur Terre

1.2.1 Apparition de la vie

La vie sur Terre est apparue rapidement après la formation de la planète, il y a 3,8 milliards d'années (soit 300 à 600 millions d'années après la formation de la Terre). On la trouve alors sous la forme d'inclusions carbonées contenues dans des cristaux d'apatite, phosphate de calcium présent à l'état naturel dans l'environnement. Cependant, les spécialistes ne sont pas encore certains que ce type d'organisme puisse déjà être qualifié de vivant car il est encore très proche du minéral.

1.2.2 Apparition du premier organisme unicellulaire

Les fossiles les plus anciens sont datés de -3,4 milliards d'années. Ils se présentent sous la forme de bactéries microsphériques telles que l'Eobacterium isolatum (illustré à gauche dans la figure 1.1). On retrouve une grande quantité de fossiles datés d'entre -3 et -1,6 milliards d'années. On parle alors de la période *monomère*. On commence alors à trouver des cyanobactéries (à droite sur la figure 1.1) et des stromatolithes. Ensemble,

1. Source : *http://dinoman.xooit.com/t1705-Dossier-sur-l-origine-de-la-vie.htm* et *http://en.wikipedia.org/wiki/Archeabacteria*

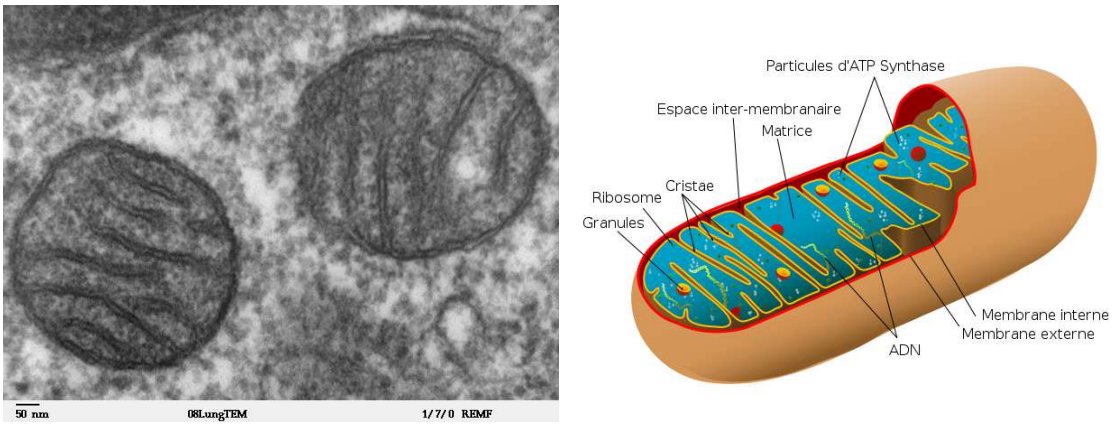


FIGURE 1.2 – *La mitochondrie, organisme unicellulaire vieux d'environ 2 milliards d'années, est capable de dégrader du matériel organique en énergie*[2].

elles commencent à modifier l'environnement en utilisant le dioxyde de carbone, présent en très grande quantité dans l'atmosphère, pour produire des composants organiques.

Durant 2,9 milliards d'années, les organismes unicellulaires évoluent vers des formes de vie plus complexes. On peut noter, par exemple, après une explosion de la quantité de bactéries présentes dans le milieu, l'apparition des *mitochondries* (illustrée par la figure 1.2), une structure intracellulaire spécialisée capable d'extraire de l'énergie de molécules organiques. Une autre avancée importante est le développement de la capacité des organismes à transformer du glucose en ATP (Adénosine TriPhosphate) qui est la molécule énergétique de la plupart des organismes actuellement présents sur la Terre.

En 1953, Stanley Miller, biologiste américain, réalise une expérience (connue sous le nom d'expérience de Miller-Urey, détaillée en Annexe A) qui tente de recréer les conditions hypothétiques de l'atmosphère primitive de la Terre, la "soupe primitive", dans laquelle la vie serait apparue. En utilisant quatre molécules présentes dans l'atmosphère (l'eau, le méthane, l'ammoniac et l'hydrogène), il arrive à synthétiser plus de la moitié des acides aminés présents dans la chimie organique et certains composants des acides nucléiques. Il n'arrive cependant pas à synthétiser d'acides nucléiques complets, c'est-à-dire qu'il n'est jamais arrivé à générer d'ARN ou d'ADN.

La première cellule eucaryote, c'est-à-dire possédant un noyau pour protéger son matériel génétique, apparaît il y a 1,5 milliard d'années. Ce sont les cellules les plus complexes de l'époque puisqu'elles doivent intégrer tous les systèmes nécessaires à leur survie. Elles vivent dans un milieu aqueux qui leur fournit une protection contre les parasites. A la même période, ces mêmes organismes développent aussi la reproduction sexuée qui a per-

2. Source : *http://fr.wikipedia.org/wiki/Fichier:Animal_mitochondrion_diagram_fr.svg* et *http://fr.wikipedia.org/wiki/Fichier:Mitochondria,_mammalian_lung_-_TEM.jpg*

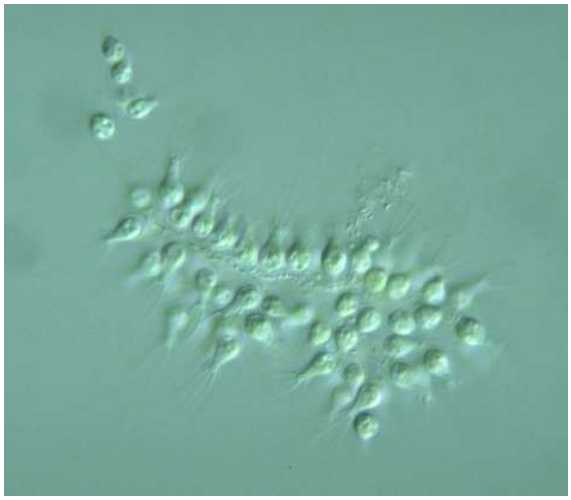


FIGURE 1.3 – *La Choanoflagellida (ici Salpingoeca), premier organisme multicellulaire*[3].

FIGURE 1.4 – *Le Porifera (ici Phylum Porifera), premier animal constitué d'une organisation en tissus*[4].

mis une explosion de la diversité des espèces.

1.2.3 Développement d'organismes multicellulaires

Il y a un milliard d'années, le premier être multicellulaire apparaît. Il a donc fallu environ 3 milliards d'années d'évolution pour passer d'un organisme unicellulaire à un organisme multicellulaire ! La forme initiale des organismes multicellulaires était en fait un rassemblement d'organismes unicellulaires. Un exemple de ces structures est la *Choanoflagellida*, présentée dans la figure 1.3 qui est considérée comme l'ancêtre direct des éponges.

Il y a 600 millions d'années, après une glaciation sévère de 400 millions d'années (seuls les océans proches des tropiques n'étaient pas gelés), les premiers animaux primitifs sont apparus. Les éponges (*Porifera*, figure 1.4) sont considérées comme les plus simples et les plus primitifs d'entre eux. Elles sont composées d'une colonie de cellules organisées en tissus (ensemble de cellules identiques qui effectuent une même action). Elles ne possédaient ni muscle, ni nerf, ni même d'organe. Les méduses (*Cnidaria* et *Ctenophora*, figure 1.5) sont ensuite apparues. Ce sont les premiers animaux possédant un système nerveux (non centralisé), des muscles leur permettant de se déplacer et un système digestif rudimentaire. Leur corps est organisé en organes (groupe de tissus qui effectuent une ou plusieurs

3. Source : *http://protist.i.hosei.ac.jp/PDB/Images/Mastigophora/Salpingoeca/sp_01/sp_4c.html* ; Copyright 1995-2009 Protist Information Server

4. Source : *http://www.middleschoolscience.com/sponges.htm*

FIGURE 1.5 – *La Cténophore (à gauche, une Bathocyroe fosteri) et la Cnidaria (à droite, une Chrysaora fuscescens), premiers animaux possédant un système nerveux*[5].

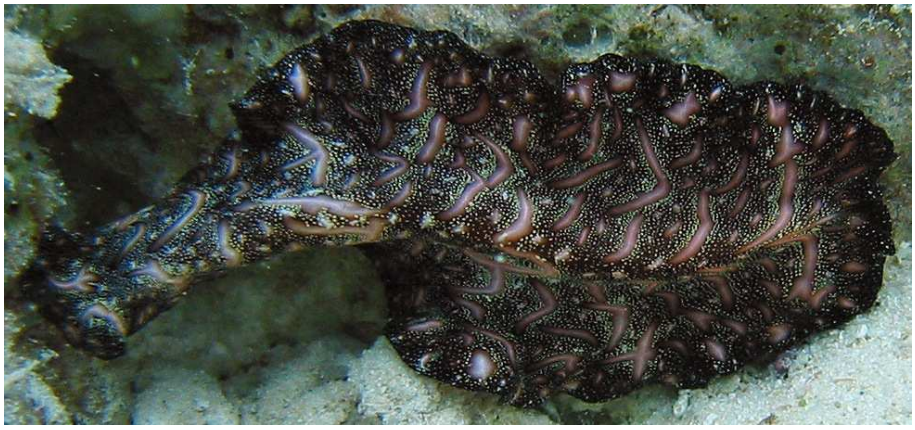


FIGURE 1.6 – *Le Platyhelminthe (ici, un Pseudobiceros bedfordi) est le premier animal possédant un système nerveux central*[6].

fonctions spécifiques). Enfin, les planaires (*Platyhelminthes* présentées dans la figure 1.6) sont les premiers animaux à posséder un système nerveux central, un système respiratoire et un système circulatoire.

On voit aussi apparaître les premières formes de plantes, très peu diversifiées. Elles sont les précurseurs possibles des flores actuelles. Ces sont les flores d'Ediacara, présentées dans la figure 1.7.

5. Sources : *http://en.wikipedia.org/wiki/File:Bathocyroe_fosteri.jpg* et *http://en.wikipedia.org/wiki/File:Sea_nettles.jpg*
6. Source : *http://fr.wikipedia.org/wiki/Fichier:Bedford%27s_Flatworm.jpg*

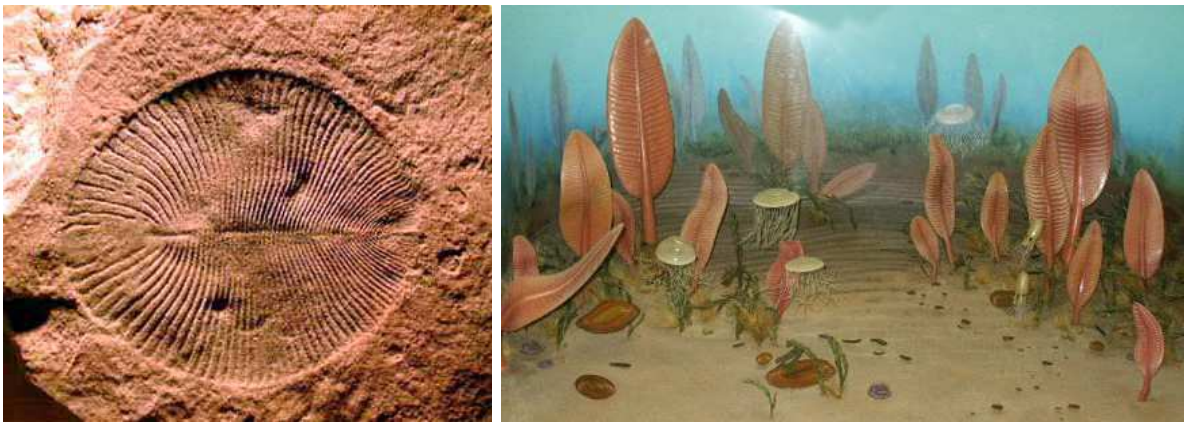


FIGURE 1.7 – *A gauche, l'Ediacara, ici Dickinsonia, première forme de végétal. A droite, panorama de la végétation primitive, toujours présente au nord de l'Australie*[7].

1.2.4 L'explosion Cambrienne

L'ère cambrienne commence il y a 542 millions d'années. Elle correspond à l'explosion de la diversité des espèces. La vie, jusque là exclusivement présente dans les océans, ne possède plus assez d'espace pour se développer. Les organismes doivent, pour survivre, conquérir de nouveaux espaces. Ils commencent dans un premier temps à se développer dans les fleuves, les rivières et les lacs. Pour cela, ils doivent s'adapter au nouveau milieu. L'eau dans ce type de milieu étant non saline, il leur a fallu développer des mécanismes respiratoires adaptés à ce nouveau milieu.

La vie terrestre apparaît également durant cette période. Les premiers organismes terrestres sont les végétaux. Ce sont en fait des plantes herbacées et vascularisées possédant de la sève. Viennent ensuite les premiers animaux terrestres. Les tétrapodes, déjà dotés de pattes lorsqu'ils vivaient dans l'eau pour fouiller les végétaux, semblent être les premiers à être sortis sur la terre. Leur sortie daterait de -365 millions d'années avec l'Ichtyostéga, illustré par la figure 1.8 dans son milieu marin (à gauche) et sur terre (à droite).

A partir de là, la vie évolue rapidement pour donner celle que nous connaissons aujourd'hui. Le plus difficile dans l'évolution des espèces a été de passer d'organismes ne possédant qu'une seule cellule aux organismes multicellulaires. Alors qu'il a fallu 3 milliards d'années pour qu'un animal tel que l'éponge, organisme très simple constitué de cellules très peu différenciées, apparaisse, la complexité de la vie a explosé en seulement quelques millions d'années pour arriver aux êtres actuels constitués de plusieurs dizaines de milliers de milliards de cellules pour le corps humain (10^{14}).

7. Source : *http://www.luciopesce.net/zoologia/arka.html* et *http://www.dinosaurcollector.150m.com/edicara.htm*

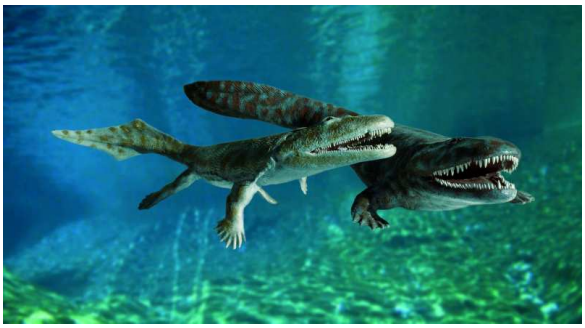



FIGURE 1.8 – *A gauche, l'Ichtyostéga dans son milieu aquatique d'origine. Il possède déjà des pattes pour fouiller les végétaux marins. A droite, c'est aussi le premier animal à être sorti de l'eau, il y a 365 millions d'années*[8].

1.3 La cellule

1.3.1 Définition

La cellule est la plus petite unité structurale des organismes vivants. Elle peut croître et se reproduire de façon autonome. Elle est la représentation de la dernière définition de la vie vue dans la partie 1.1 : elle représente un état hautement organisé de la matière et est capable de maintenir un état optimal, en résistant au mieux aux fluctuations de l'environnement (homéostasie). Pour cela, elle développe un métabolisme, aérobie ou anaérobie (en d'autres termes, avec ou sans oxygène). La cellule est aussi capable de se diviser afin de maintenir et/ou de multiplier l'organisme qu'elle constitue. Pour la grande majorité d'entre elles, les cellules possèdent un génome qui permet une adaptation au milieu dans lequel elles sont plongées.

1.3.2 Cycle cellulaire

Afin de maintenir son état, la cellule possède un cycle, quasiment commun aux différents types de cellules existants, qui contrôle son fonctionnement interne. La durée de ce cycle diffère cependant selon les espèces : de quelques minutes pour les oeufs (les treize premiers cycles de division d'un oeuf humain durent 9 minutes !) à une vingtaine d'heures pour une cellule humaine. La durée moyenne pour un organisme constitué d'un grand nombre de cellules (humain ou animal) varie en moyenne de quelques dizaines à quelques centaines d'heures. Certaines cellules dites postmitotiques telles que les neurones ne se divisent pas. Elles n'entrent en fait pas dans le cycle cellulaire et restent dans une phase métabolique. Le cycle cellulaire commence à la fin de la division qui lui donne naissance et

8. Source : *http://www.stuttgart.de/item/show/313949/1/2/3267772?* et *http://www.britannica.com/EBchecked/topic-art/281451/6035/Ichthyostega-model-by-JS-Collard*

se termine à la fin de la division de cette même cellule. Il représente l'ensemble des actions effectuées par la cellule pour se diviser. Il permet aussi le maintien qualitatif et quantitatif du patrimoine génétique qui donne la fonction de celle-ci [Le Peuch and Dorée, 2000].

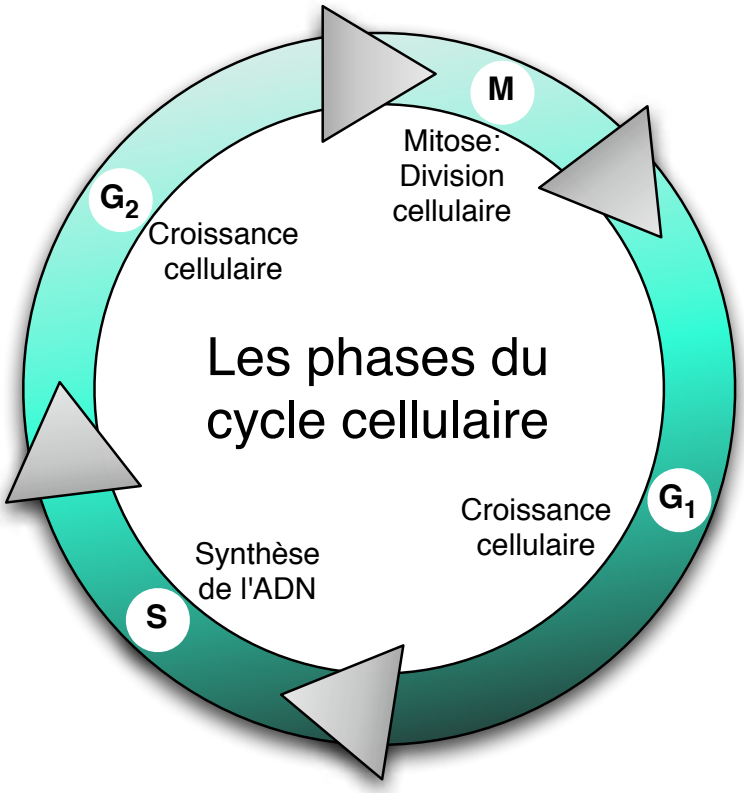


FIGURE 1.9 – *Le cycle cellulaire d'une cellule est composé de quatre phases : deux phases de croissance et de vérifications (G_1 et G_2), une phase de division (la mitose M) et une phase de synthèse d'ADN S.*

Le cycle cellulaire est constitué de 4 phases appelées G_1, S, G_2 et M. Le schéma 1.9 est une illustration de ce cycle. La phase S correspond à une phase de synthèse de l'ADN. Elle dure environ la moitié du temps du cycle. La phase M, pour Mitose, est beaucoup plus courte (environ une heure pour un être humain). Elle correspond à la division cellulaire proprement dite. Durant la mitose, une suite d'opérations a lieu dans la cellule (résumée par le schéma 1.10) :

1. Prophase : le matériel génétique, l'ADN présent en temps normal dans le noyau sous forme de chromatine, se condense sous la forme de chromosomes. La membrane nucléaire disparaît pendant ce temps. Les centrosomes, qui créeront le plan de division par la suite, se déplacent vers les bords de la cellule.
2. Métaphase : le plan de division se crée grâce aux centrosomes précédemment placés

dans la cellule. Il est en fait constitué de fibres symétriques capables de capturer les chromosomes se trouvant au centre de la cellule. A cet instant de la mitose, les chromosomes sont parfaitement alignés au centre de la cellule.

3. Anaphase : les centres des chromosomes s'ouvrent et les chromosomes, toujours attachés aux fibres du plan de division, se séparent en deux.
4. Télophase : les chromosomes se décondensent en se désolidarisant du réseau de fibres. Celui-ci se dissout et le sillon de division se forme.

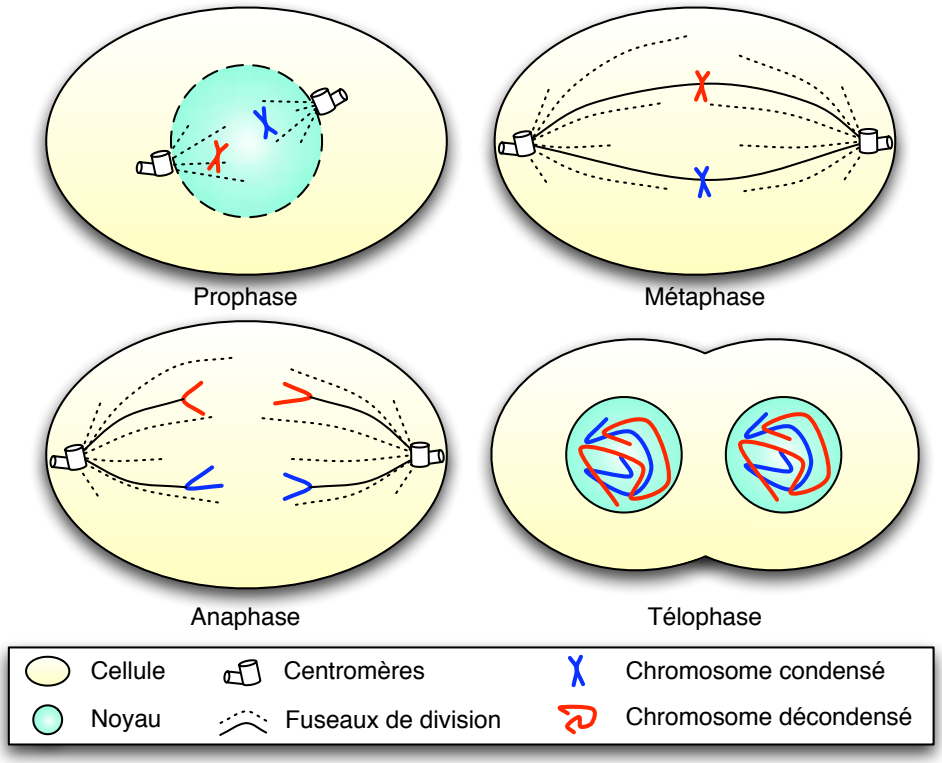


FIGURE 1.10 – *La phase de mitose dans une cellule, phase du cycle cellulaire, est décomposée en quatre principales phases : la prophase (condensation de l'ADN et dissipation du noyau), la métaphase (alignement des chromosomes sur le plan de division), l'anaphase (séparation des chromosomes) et la télophase (déconcentration des chromosomes et apparition du sillon de division).*

A la fin de ces quatre opérations appelés mitose, une phase de cytokinesis a lieu. Elle achève la division en séparant complètement la cellule fille de sa cellule mère. Un nouveau noyau est créé autour des chromosomes et la membrane avec son cytoplasme (liquide intracellulaire) sépare la nouvelle cellule.

Les phases G_1 et G_2 correspondent à des phases de croissance et de vérification. La phase G_1 peut être très longue : c'est elle qui régule la division de la cellule.

1.4 La bio-inspiration

La biologie et les êtres vivants ont toujours été une source d'inspiration pour les humains. De nombreux exemples en sont la preuve.

Dans le domaine de l'aviation par exemple, on a toujours imaginé un objet volant avec des ailes, comme en possèdent les oiseaux. Léonard de Vinci fut le premier à dessiner les plans d'une machine volante au XVI^{eme} siècle dont la voilure s'inspirait des ailes des oiseaux. La figure 1.11 montre l'étude qu'il en avait faite. Alors que les avions d'aujourd'hui ont plus suivi la voie de l'aérodynamique, les premiers planeurs, plus légers que les avions, ont été construits en se basant sur les ailes fabriquées avec du bois et du coton. Le pionnier du domaine, Otto Lilienthal, a ainsi réussi à effectuer près de deux mille vols planés entre 1891 et 1896 à bord de seize engins différents. La figure 1.12 présente l'un d'entre eux.

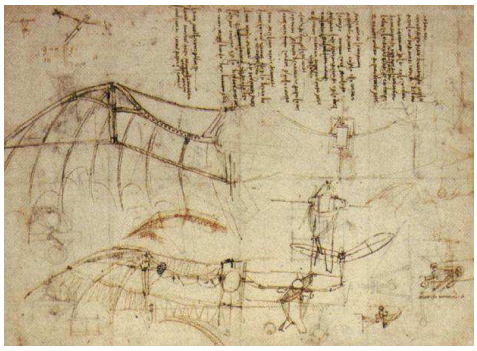


FIGURE 1.11 – *Plan de l'étude de l'aile de Léonard de Vinci.*

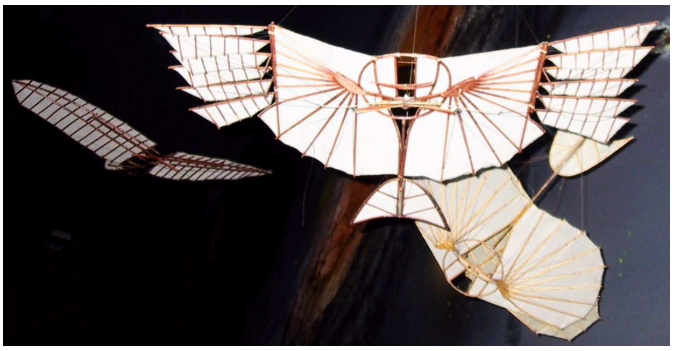


FIGURE 1.12 – *Modèle de l'aile de planeur de Lilienthal.*

Dans le domaine des matériaux, les propriétés hydrophobes des feuilles de certaines plantes et plus particulièrement des fleurs de Lotus sont très recherchées. Il a fallu plusieurs dizaines d'années pour comprendre le fonctionnement des feuilles de cette plante. En plus d'être capable d'évacuer l'eau de leur surface très rapidement, on a observé qu'elles pouvaient être nettoyées de n'importe quelle saleté en utilisant uniquement de l'eau : on décrit ainsi la surface de la feuille de la fleur de lotus comme étant une surface superhydrophobe. Être capable de produire un matériau possédant de telles caractéristiques peut engendrer de nombreuses applications : textile, conteneur, construction de bateaux, etc. L'avènement du microscope électronique à balayage dans les années 60 a permis de mieux comprendre le phénomène de la fleur de Lotus, couramment appelé "l'effet Lotus" : la feuille de Lotus est constituée de rugosités nanométriques qui ne permettent pas aux gouttes d'eau d'adhérer à sa surface (figure 1.13). Les nanotechnologies permettent maintenant de fabriquer des matériaux ayant ces propriétés. La figure 1.14 illustre deux applications des surfaces superhydrophobes : la capacité d'auto-nettoyage (à gauche) permet un nettoyage immédiat

FIGURE 1.13 – *Les rugosités nanométriques de la feuille de Lotus ne permettent pas aux gouttes d'eau d'adhérer à sa surface*[9].

des surfaces en utilisant seulement de l'eau et permet de déverser la totalité d'un produit contenu dans un récipient traité avec ces systèmes (à droite).

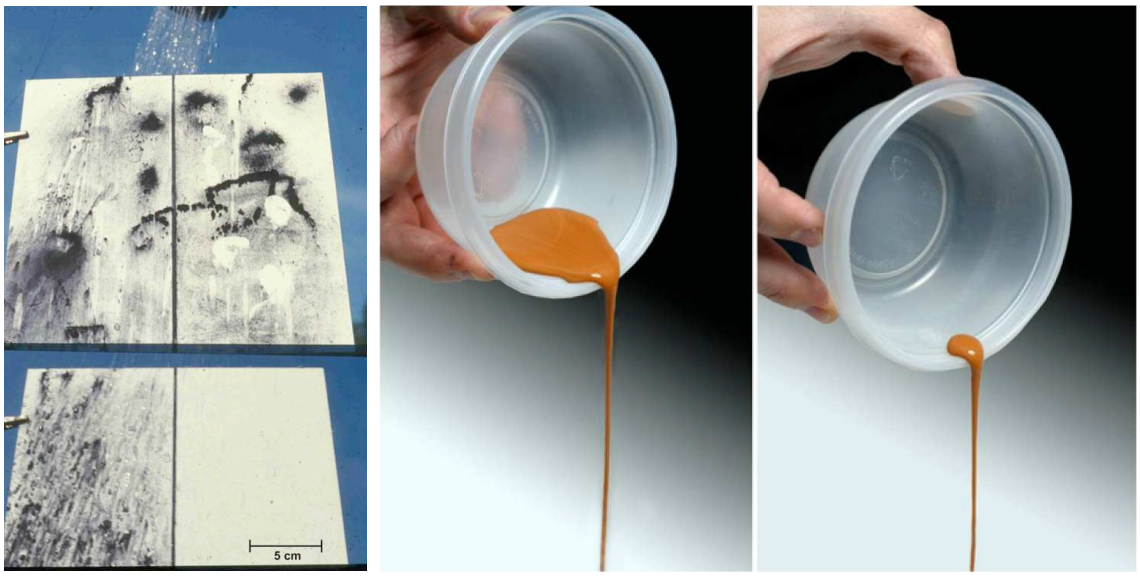


FIGURE 1.14 – *Deux exemples d'application des surfaces superhydrophobes : à gauche, les capacités d'auto-nettoyage de celles-ci et, à droite, le traitement de récipient permet de récupérer la totalité du contenu*[10].

Plus proche de nos préoccupations, l'intelligence artificielle s'est elle aussi fortement inspirée de la nature. La définition donnée par le Centre National de Ressources Textuelles et Lexicales (CNRTL, encyclopédie du CNRS) décrit l'intelligence artificielle comme étant

9. Source : *http://fr.wikipedia.org/wiki/Fichier:LotusEffekt1.jpg* et *http://fr.wikipedia.org/wiki/Fichier:Lotus3.jpg*
10. Sources des images : [Solga et al., 2007]

la "recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains"[11]. Cette définition montre l'importance de la comparaison à un produit développé par la Nature dans la conception d'un système produit par l'Homme. En plus de cette base conceptuelle, les mécanismes de l'intelligence artificielle s'inspirent fortement des mécanismes du vivant pour créer des systèmes intelligents. On peut citer par exemple les réseaux de neurones [McCulloch and Pitts, 1943], basés sur une simplification de notre système nerveux.

S'inspirer de la biologie est donc capital pour développer les technologies du futur. Ainsi, utiliser des techniques du vivant pour concevoir les futurs systèmes robotiques nous permettra d'obtenir des résultats bien meilleurs et plus rapidement. C'est pourquoi étudier le vivant, ses origines et son fonctionnement est une des étapes primordiales à la conception des futurs bio et nano systèmes.

11. Voir la page sur l'intelligence du site du CNRTL : *http://www.cnrtl.fr/lexicographie/intelligence*

2

Etat de l'art

Une histoire des créatures artificielles

Le développement des créatures artificielles en synthèse d'images s'est fait en plusieurs étapes. Dans les années 80-90, beaucoup de chercheurs ont travaillé sur la simulation du comportement de créatures artificielles. Partant d'une forme prédéfinie, l'évolution porte uniquement sur leur comportement. Après quelques années, on a commencé à faire évoluer la forme et le comportement des créatures pour qu'elles s'adaptent à leur environnement. Depuis une quinzaine d'années, un certain nombre de travaux tendent à produire des créatures basées sur une unité de base bien plus petite : la cellule artificielle. C'est en fait l'échelle de simulation des créatures qui a évolué au cours du temps : partant de la simulation du comportement d'une entité fixe, on en simule aujourd'hui le processus de *développement cellulaire*.

Il est intéressant de noter que l'évolution de l'échelle de simulation s'est faite conjointement à l'évolution de la puissance de calcul des machines : il faut aujourd'hui quelques secondes pour calculer le comportement macroscopique d'une créature, quelques heures pour évoluer la morphologie de celle-ci et quelques jours pour simuler son développement cellulaire. Si on revient une dizaine d'années en arrière, ces temps de calcul étaient décalés d'un niveau : il fallait quelques heures pour simuler le comportement d'une créature et quelques jours pour évoluer sa morphologie. Quant à simuler son développement cellulaire, cela semblait inaccessible !

(a) Les Sims 3 (Electronic Arts) est un jeu très populaire dans lequel les personnages non joueurs évoluent dans un monde virtuel très complexe.

(b) Dans film "le seigneur des anneaux : le retour du roi" (Peter Jackson), la scène de bataille du Mûmakil est entièrement contruite à l'aide de figurants virtuels intelligents.

FIGURE 2.1 – *Les applications dans le monde du cinéma et des jeux vidéos sont multiples. Les personnages virtuels y sont utilisés pour peupler les mondes virtuels dans lesquels les utilisateurs évoluent.*

Dans ce chapitre, nous allons nous intéresser à l'histoire des créatures artificielles en approfondissant ces trois phases de leur étude. Nous étudierons différents modèles en portant une attention particulière au développement cellulaire. Nous conclurons ce chapitre en positionnant nos travaux par rapport à ce vaste domaine de recherche.

2.1 Simuler le comportement

La simulation comportementale a pour but de simuler le comportement d'agents dans un environnement virtuel. Elle est issue de l'informatique graphique. Les domaines d'application de la simulation comportementale sont principalement axés sur les jeux vidéos (pour les personnages non joueurs, présentés dans la figure 2.1(a)) et le cinéma (pour l'animation de foule ou les films d'animation par exemple, présentés dans la figure 2.1(b)).

2.1.1 L'agent virtuel

Définition

L'agent virtuel dans la simulation comportementale désigne une créature artificielle autonome qui possède une représentation dans un environnement virtuel [Heguy et al., 2001]. L'agent doit prendre ses décisions localement en fonction de son environnement perceptible. Cela permet de supprimer la vision globale du système qui, en plus de son manque de réalisme, est impossible dans un environnement virtuel possédant un grand nombre

d'acteurs virtuels, comme dans une foule par exemple.

La propriété d'autonomie est primordiale pour un agent virtuel. Elle implique un mécanisme de sélection de l'action à effectuer par l'agent qui doit prendre en compte ses capacités (les actions que celui-ci est capable de réaliser). Ce mécanisme de sélection d'action est souvent représenté dans une boucle Perception-Décision-Action que nous verrons par la suite. Les agents poursuivent un but qui est soit donné par l'utilisateur, soit produit par l'agent lui-même en fonction de son état interne et de son environnement.

Un agent virtuel est souvent composé de différents capteurs (vision, collision, positionnement spatial, etc.) permettant l'acquisition d'informations dans l'environnement, et d'effecteurs (roues, bras, etc.) permettant d'interagir avec l'environnement.

Les propriétés d'un agent

Un grand nombre de propriétés permettent de définir un agent. David Panzoli donne dans sa thèse [Panzoli et al., 2008] la liste des principales propriétés suivantes :

- L'autonomie permet à l'agent de sélectionner les actions qu'il va accomplir sans intervention extérieure (et en particulier humaine). Pour cela, il doit posséder une représentation du problème, de son environnement ainsi que de son état interne.
- La réactivité donne le temps de réponse de l'agent lorsqu'il est confronté à un problème.
- L'intentionnalité permet à l'agent de produire des plans à partir de buts explicites. Elle peut s'opposer à la propriété de réactivité si le temps de production de ces plans est trop long.
- La sociabilité est la capacité de l'agent à communiquer avec les autres agents de l'environnement.
- La situation dénote l'importance de l'environnement pour la définition du comportement de l'agent.
- L'adaptabilité est l'aptitude de l'agent à s'adapter aux modifications de son environnement.

La boucle Perception-Décision-Action

Dans le but de sélectionner la meilleure action à effectuer à chaque instant de la simulation, l'agent virtuel possède un mécanisme basé sur un schéma défini par Tu et Terzopoulos en 1994 [Tu and Terzopoulos, 1994]. Ce schéma, présenté dans la figure 2.2, est une boucle composée de trois phases :

1. une phase de *perception*, durant laquelle l'agent détermine l'état de son environnement à l'aide de ses différents capteurs,
2. une phase de *décision*, qui a pour but de prendre en compte la perception précédemment effectuée pour choisir la meilleure action à effectuer en fonction du but de l'agent, de son environnement perceptible et de son état interne,
3. une phase d'*action*, qui accomplit au niveau des effecteurs de l'agent l'action précédemment choisie.

La grande majorité des agents virtuels utilisés actuellement en simulation comportementale possède ce principe de fonctionnement.

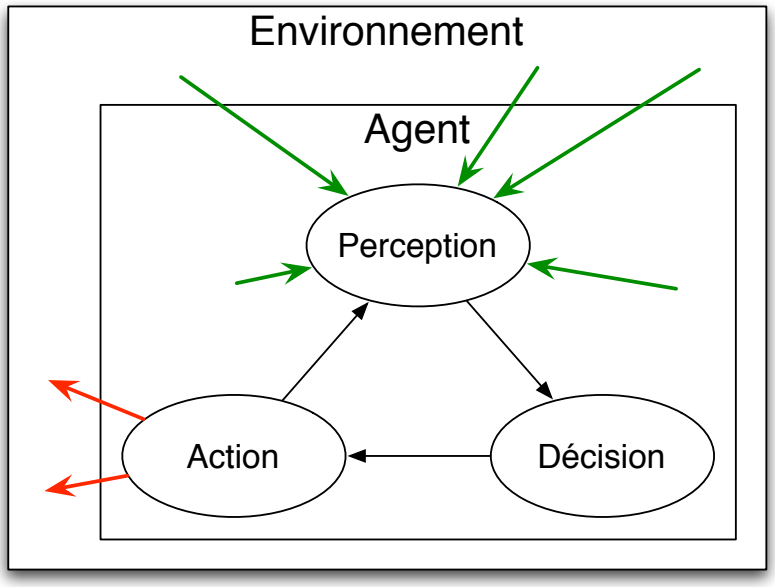


FIGURE 2.2 – *La boucle Perception-Décision-Action d'un agent virtuel lui permet de sélectionner la meilleure action en fonction de son environnement local et de son but propre.*

En fonction des environnements dans lesquels ils sont plongés (dynamique ou markoviens), différentes approches existent pour produire des comportements cohérents et proches de ceux attendus. La partie suivante introduit ces différentes approches.

2.1.2 Les différentes approches

Les agents délibératifs

L'approche délibérative est issue de l'intelligence artificielle. Elle se base sur des algorithmes de planification afin de trouver la suite d'actions à effectuer en fonction des buts courants de l'agent. Le schéma 2.3 montre la partie décisionnelle d'un agent délibératif.

La couche modélisation permet à l'agent de créer un modèle cognitif de son environnement. Leur principe a été modélisé par John D. Funge en 1999 [Funge et al., 1999]. La planification est la phase de délibération de l'agent. De nombreuses techniques existent telles que le raisonnement à partir de cas [Ontanon et al., 2007], la logique floue, les systèmes experts, etc. Ces deux couches sont souvent très coûteuses en temps de calcul et il est difficile de garantir leur temps d'exécution. Il est de plus obligatoire de les relancer intégralement après l'exécution d'une ou plusieurs d'actions.

Comme nous l'avons vu précédemment, la sélection de l'action est le point faible de ce type de modèles. L'incapacité de choisir une bonne action dans un court temps de calcul ne permet pas leur intégration facile dans des systèmes temps réels tels que les robots car ils doivent parfois prendre des décisions rapides afin de protéger leur intégrité.

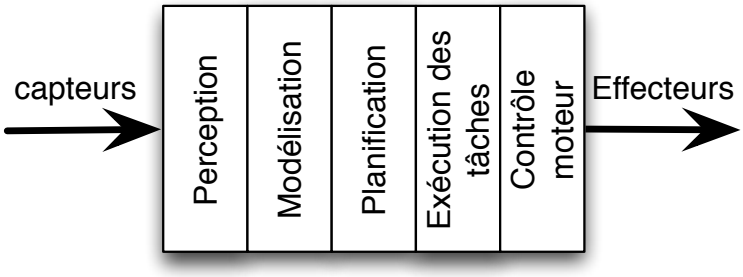
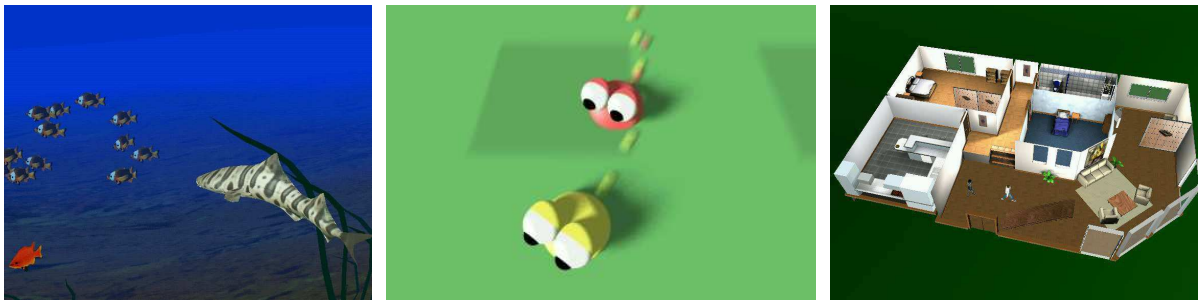


FIGURE 2.3 – *L'architecture en couche de l'agent cognitif lui permet d'avoir une représentation du monde avant de faire une planification dessus qui lui permettra de choisir la ou les actions à déclencher.*

Les agents réactifs

Même si l'approche délibérative est encore très utilisée et particulièrement dans les jeux vidéos, les limites de l'intelligence artificielle sont souvent atteintes et induisent un manque de réactivité [Brooks, 1986]. Brooks propose alors une architecture dite de subsomption qui est basée sur la décomposition paraléllisable de la sélection de l'action. Différentes couches spécialisées dans les différents domaines d'action (évitement d'obstacles, exploration, cartographie, planification) sont directement reliées aux capteurs de l'agent. Un système d'arbitrage permet de décider quelle couche est prioritaire dans une situation particulière de l'environnement. L'arbitre envoie ensuite les actions à effectuer aux effecteurs. Cette approche permet de s'abstraire complètement de toute représentation interne de l'environnement ce qui facilite le travail de l'agent.

D'autres techniques issues de la vie artificielle ont vu le jour dans les années 80 et 90. Reynolds [Reynolds, 1987] développe un modèle basé sur des agents au comportement



(a) Terzopoulos (b) Cliff & Miller (c) Sanchez

FIGURE 2.4 – *Les agents réactifs et hybrides sont les plus utilisés dans la simulation comportementale. Du fait de leur rapidité de réaction, ils s'intègrent beaucoup plus facilement à un monde virtuel dynamique.*

très simple, appelé les boids. Il permet de simuler de façon très réaliste une nuée ou un troupeau d'individus se déplaçant dans l'environnement. Pour cela, Reynolds définit ses boids à l'aide de trois règles simples :

- l'évitement des collisions entre les boids,
- l'alignement de la vitesse avec la vitesse globale du groupe,
- le centrage de la position du boid par rapport à son groupe.

En 1989, Goldberg utilise des méthodes évolutionnaires afin de faire émerger des comportements sans décomposer précisément le problème [Goldberg, 1989]. Il utilise pour cela le principe de l'évolution par sélection naturelle de Darwin [Darwin, 1859]. Dans ce cas là, seule une modélisation générique du contrôleur ainsi qu'une fonction d'évaluation sont nécessaires. La fonction d'évaluation permet de noter les agents par rapport au résultat final qu'on attend d'eux. La pression de l'évolution génétique permettra l'émergence d'un contrôleur parfaitement adapté au problème posé.

Terzopoulos est un des premiers à utiliser cette technique dans l'animation comportementale [Tu and Terzopoulos, 1994]. Il développe des poissons dont le squelette est dessiné par la main de l'homme et dont le comportement défini par des réseaux de neurones évolue à l'aide d'un algorithme génétique. Il obtient grâce à cela une nage des poissons très réaliste dans un simulateur physique (figure 2.4(a)).

L'avènement des réseaux de neurones [McCulloch and Pitts, 1943, Rosenblatt, 1958, Hopfield, 1982] et des systèmes de classeurs [Holland, 1975, Goldberg, 1989, Wilson, 1994] couplés aux algorithmes évolutionnistes a permis une complexification des comportements des agents virtuels. Des comportements de fuite et de poursuite [Cliff and Miller, 1996] (figure 2.4(b)), des stratégies d'équipe [Sanza, 2001] ou d'anticipation [Panzoli et al., 2008] ont émergé en n'exprimant que le but global du système.

Bien que plus rapide pour sélectionner l'action que les agents délibératifs, les agents ré-

actifs ne sont pas capables d'effectuer des tâches complexes. L'approche hybride présentée dans la partie suivante permet de concilier les avantages des deux approches précédentes.

Les agents hybrides

Les agents hybrides sont des agents possédant deux couches : une couche réactive permettant de réagir rapidement lorsque la situation l'exige et une couche délibérative permettant de planifier des actions sur le long terme.

Innes Fergusson a présenté en 1992 une architecture [Ferguson et al., 1992] qui montre parfaitement le fonctionnement de tels agents. Entre les deux couches précédemment présentées, il ajoute un séquenceur permettant de sélectionner quelques types de comportements déclenchés dans une situation précise.

Cette architecture est celle la plus souvent utilisée dans l'animation d'humanoïdes virtuels. On les trouve à la base des comportements des acteurs de la visite du musée virtuel de la Cité des Sciences et de l'Industrie de Paris [Donikian and Rutten, 95] ou pour l'apprentissage d'actions de bas niveau dans ViBes [Sanchez et al., 2004] (figure 2.4(c)).

2.2 Evolution conjointe de la morphologie et du comportement

Alors que la simulation comportementale travaille uniquement sur des morphologies figées, la morphogenèse permet de faire *évoluer* la morphologie et le comportement des créatures en assemblant différents blocs fonctionnels. Ceci permet de générer des créatures mieux adaptées à leur environnement en se focalisant sur des comportements de plus bas niveau. Ainsi, alors que la simulation comportementale va plutôt travailler sur le comportement global d'humanoïdes ou d'avatars, l'évolution conjointe de la morphologie et du comportement permettra de retrouver par évolution des comportements basiques tels que la marche ou la nage.

Le comportement de ces créatures est souvent basé sur des réseaux de neurones [Rosenblatt, 1958] ou des systèmes de classeurs [Holland and Reitman, 1978]. L'évolution de la morphologie implique une représentation logique de celle-ci permettant son codage dans un génome. Ainsi, un algorithme évolutionnaire pourra faire évoluer cette morphologie et son comportement dans l'environnement. Il existe deux principaux types de codage de la morphologie : les grammaires et les graphes de développement. Ils seront présentés dans la partie suivante avec les créatures qu'ils ont permis de produire. Nous terminerons cette section par un modèle qui fait intervenir l'utilisateur pour aider à la sélection des

créatures, et deux autres intégrant un écosystème complet en donnant aux créatures la possibilité de se nourrir et de se reproduire.

2.2.1 Représentation de morphologies à base de grammaires

Les systèmes de morphogenèse basés sur des grammaires ont été introduits par Aristid Lindenmayer en 1968 dans [Lindenmayer, 1968]. Couramment appelés L-Systèmes, ils sont définis à l'aide d'un formalisme mathématique composé d'un triplet (Σ, ω, P) où :

- Σ est un alphabet contenant l'ensemble des symboles utilisables dans la grammaire (on note souvent en minuscule les noeuds terminaux et en majuscules les noeuds intermédiaires),
- ω est le mot initial de la grammaire (composition de symboles de l'alphabet A),
- P est l'ensemble des règles de la forme $symbole \rightarrow mot$ qui donne les règles de développement de la grammaire.

Dans un L-Système, les règles de production de mots peuvent être exécutées en parallèle. Une fois le système développé, on interprète les symboles de l'alphabet Σ à l'aide de représentations graphiques. Par exemple, à partir du triplet

$$\begin{aligned} \Sigma &= \{A, B\} \\ \omega &= \{A\} \\ P &= \{A \rightarrow AB \ ; \ B \rightarrow A\} \end{aligned}$$

le mot suivant est produit, génération après génération,

Génération 0 : A
Génération 1 : AB
Génération 2 : ABA
Génération 3 : $ABAAB$
Génération 4 : $ABAABABA$
Génération 5 : $ABAABABAABAAB$
Génération 6 : $ABAABABAABAABABAABABA$
Génération 7 : $ABAABABAABAABABAABABAABAABABAABAAB$

qui peut être interprété graphiquement par la figure 2.5, si A représente une tige finissant par un noeud capable de produire deux tiges et B représente une tige simple.

Ce système permet de générer des formes géométriques diverses. Il est très utilisé pour la génération de plantes [Prusinkiewicz and Lindenmayer, 1990]. Dans sa version paramétrique, il est possible de remplacer certains symboles par des fonctions mathématiques

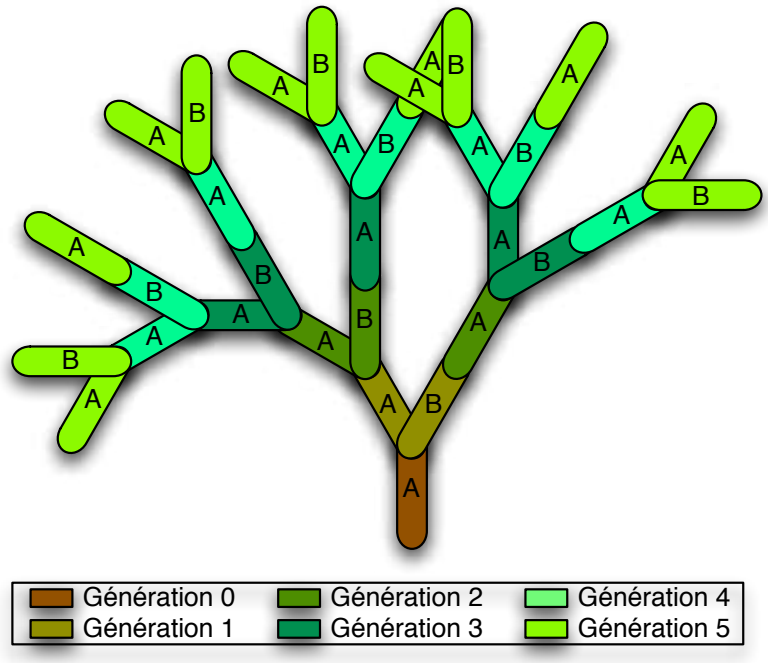


FIGURE 2.5 – *Interprétation d'une grammaire simple développée à l'aide d'un système de Lindenmayer.*

dont les paramètres sont donnés par le L-Système. On parvient alors à produire le design des bâtiments en générant une architecture souvent novatrice et intéressante comme illustré dans la figure 2.6.

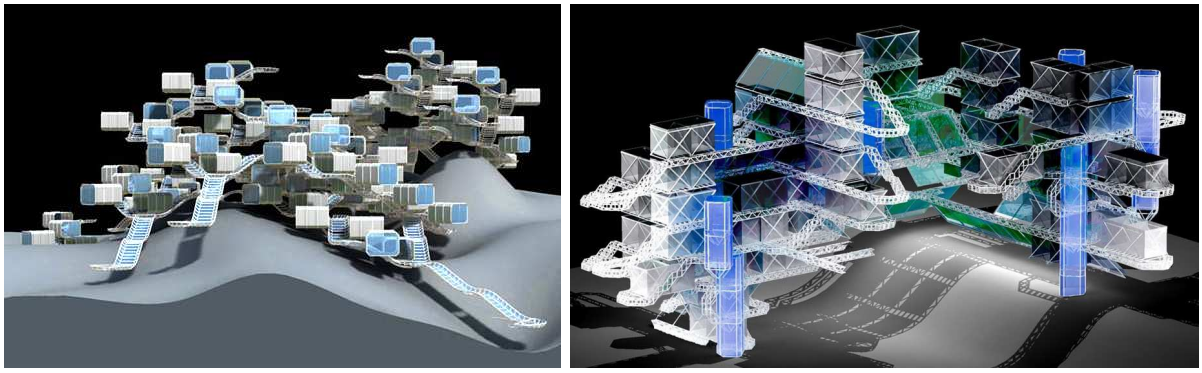
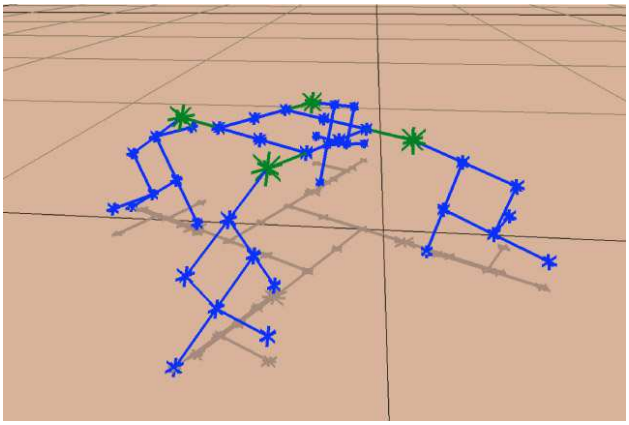


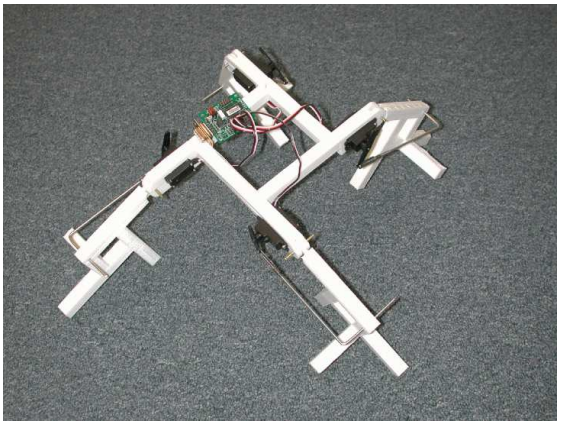
FIGURE 2.6 – *Design évolutif de bâtiment à l'aide de L-Systèmes*[12]

En utilisant des L-Systèmes connectés à l'aide de réseaux de neurones, Hornby et Pollack [Hornby and Pollack, 2001] ont fait évoluer des créatures artificielles dans un simulateur physique, afin qu'elles apprennent à se déplacer dans leur environnement virtuel

12. Source : Avec l'aimable autorisation de Michael Hansmeyer, http://www.michael-hansmeyer.com/



(a) La créature dans un monde simulé



(b) le robot issu de la créature artificielle

FIGURE 2.7 – *(a) A l'aide d'une représentation morphologique en L-Système et d'un réseau de neurones, la créature apprend à se déplacer dans un monde simulé. (b) Une fois le robot construit dans le monde réel, il est capable d'évoluer dans son environnement.*

(figure 2.7(a)). Une fois l'apprentissage terminé, ils ont construit les robots correspondant aux meilleures créatures (figure 2.7(b)) qui se sont avérés être capable de se déplacer dans le monde réel [Pollack et al., 2003].

En les plongeant dans un environnement complexe et en dotant les noeuds terminaux de fonction, les L-Systèmes peuvent produire des plantes avec métabolisme. Ainsi, dans sa thèse [Bornhofen, 2008], Bornhofen utilise un alphabet particulier, donné par la table 2.1, pour décrire les capacités de chaque segment produit par le L-Système. L'environnement pourra ainsi influer sur le développement de la plante en fonction de la luminosité locale ou de la quantité de minéraux proche d'une racine. De plus, des noeuds terminaux sont aussi alloués afin de permettre à la plante de se reproduire en développant des graines

Symbole	Localisation	Terminal	Fonction
l	Aérienne	oui	Capte la lumière
f	Aérienne	oui	Produit une fleur
s	Aérienne	oui	Produit une graine
b	Aérienne	oui	Produit une branche de structure
r	Souterraine	oui	Assimile les minéraux du sol
c	Souterraine	oui	Produit une racine de structure
A..Z	-	non	Représente un apex
[ ]	-	oui	Indique une ramification
+ - \ / & ^	-	oui	Indique une rotation 3D

TABLE 2.1 – *Alphabet du L-Système utilisé pour la croissance de plantes avec métabolisme de Bornhofen*

et/ou des fleurs qui permettront son développement dans l'environnement au gré du vent. Cette approche est particulièrement intéressante car elle prend en compte le métabolisme global de la plante qui influera sur son développement en fonction de l'environnement dans lequel elle sera plongée. Ainsi, comme illustré sur la figure 2.8, les feuilles suivront plutôt un gradient de luminosité, les racines suivront les filons de nutriments dans le sol et la plante développera plus ou moins de fleurs en fonction de sa stratégie de reproduction.

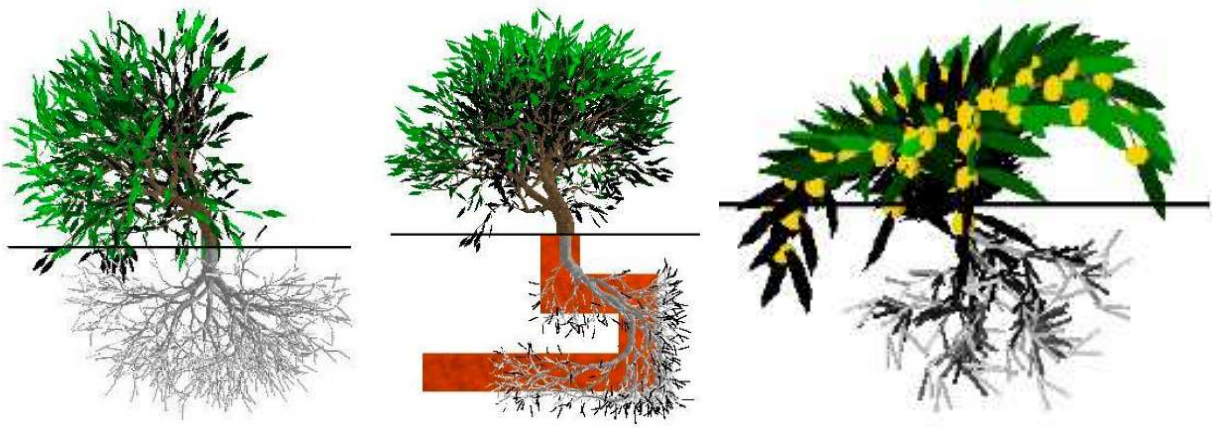


FIGURE 2.8 – *Les plantes avec métabolisme de Stéphane Bornhofen utilisent des gradients de luminosité (à gauche) ou de nutriments (au centre) pour se développer et développent des fleurs pour se reproduire (à droite).*

Appliqué à des créatures, le modèle Framstick [Komosinski and Ulatowski, 1999] de Komosinski permet de produire des écosystèmes complets dans lesquels "vivent" des créatures artificielles. Les créatures sont construites à l'aide de bâtonnets ("sticks") qui sont reliés par des muscles leur permettant de se déplacer. Dotées de capteurs olfactifs, elles recherchent de la nourriture dans leur environnement. Celle-ci augmente le niveau d'énergie de la créature qui peut ensuite continuer à en rechercher. Cependant, même si ce modèle possède un module permettant une évolution génétique sur les créatures, la plupart d'entre elles sont produites par la main de l'homme, l'émergence de bonnes solutions demandant un effort combinatoire important.

2.2.2 Les systèmes à base de graphe de développement

Les systèmes à base de graphe de développement, aussi appelés Graphtal, permettent le développement de créatures possédant d'intéressantes propriétés de cycles ou de symétries. Le précurseur dans ce domaine est le fameux Karl Sims [Sims, 1994]. Il définit ces réseaux de la manière suivante :

- les noeuds du réseau représentent des blocs élémentaires de la créature avec leurs propriétés (forme, masse, capteur, couleur, etc.),
- les arcs entre les noeuds représentent les liaisons entre les blocs et le type de liaison qui les unit (fixe, rotation, élongation, etc.).

La figure 2.9 montre un exemple de développement d'une morphologie à l'aide d'un Graphtal.

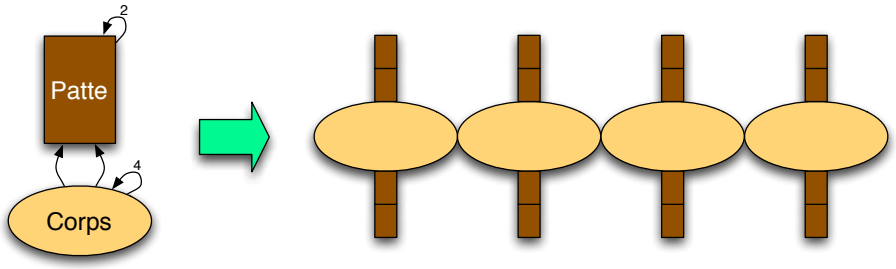
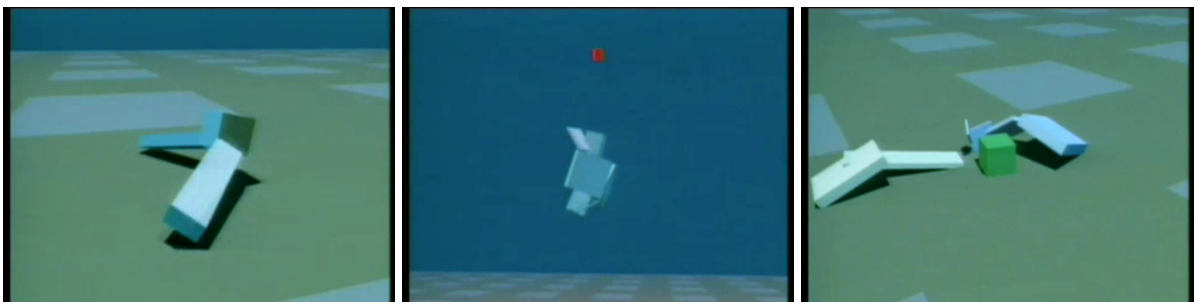


FIGURE 2.9 – *Exemple de développement de la morphologie d'une créature à partir d'un Graphtal.*

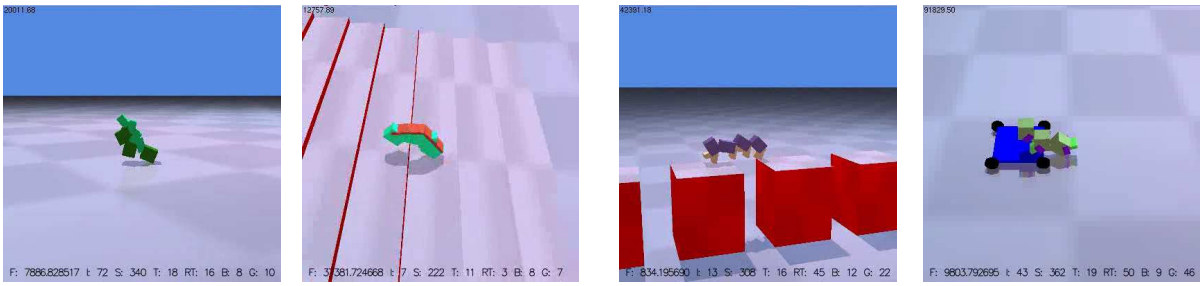
Ainsi, avec des réseaux relativement simples et faciles à travailler, il obtient des morphologies de créatures très variées. En utilisant un algorithme génétique pour faire évoluer en même temps le réseau de neurones et la morphologie et en donnant un but bien précis aux créatures (se déplacer vers un point, attraper un cube...), il obtient des créatures adaptées à leur environnement et au problème qui leur est posé. Elles peuvent posséder des pattes, des bras, avoir la forme de vers, ou encore développer des formes d'hélices pour se déplacer dans un liquide. La figure 2.10 montre quelques exemples de ces créatures.

Il a fallu une dizaine d'années pour arriver à reproduire les travaux de Karl Sims. Différents modèles sont apparus, s'appuyant fortement sur ce modèle de Graphtal animé



(a) Marcher (b) Nager (c) Coévolution

FIGURE 2.10 – *Quelques exemples de créatures développées par Karl Sims à l'aide de Graphtal et se déplaçant en utilisant un système de neurones. En leur demandant de résoudre différents problèmes (marcher, nager, attraper un cube, etc), elles développent différentes morphologies et différents comportements pour arriver à leur fin.*



(a) Marcher (b) Monter un escalier (c) Traverser des fossés (d) Faire du skate

FIGURE 2.11 – *Quelques exemples de créatures développées par Nicolas Lassabe. En se basant sur les travaux de Karl Sims et en complexifiant l'environnement, il obtient différentes morphologies adaptées à leur problème : (de gauche à droite) elles apprennent à se déplacer dans l'environnement, à escalader un escalier, à marcher sur des cubes séparés par le vide ou même à faire de la planche à roulettes.*

grâce à un réseau de neurones [Taylor, 2000, Ray, 2001, Krčah, 2007]. Nicolas Lassabe a produit un modèle basé sur un système de classeurs afin de développer des créatures ressemblant à celles de Karl Sims [Lassabe et al., 2007]. En les confrontant à des problèmes plus complexes, il a obtenu des créatures répondant correctement aux problèmes qui leur étaient posés, aussi bien au niveau morphologique qu'au niveau comportemental. La figure 2.11 illustre quelques unes de ces créatures.

Cette technique d'assemblage de cubes a été appliquée aux robots modulaires. Hod Lipson a produit des modules robotiques cubiques permettant des mouvements de rotation [Yim et al., 2007]. A l'aide de modèles identiques à celui de Nicolas Lassabe, il a produit des robots reconfigurables capables de se déplacer dans leur environnement. Ces robots sont aussi capables de s'adapter en cas de défaillance d'un de leurs modules

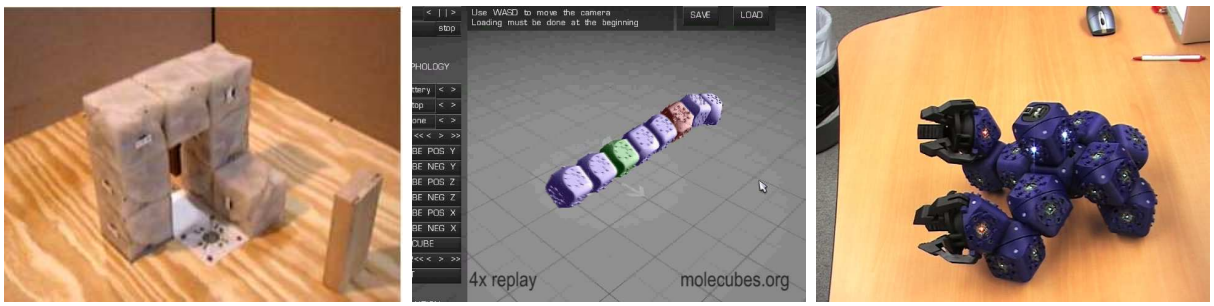
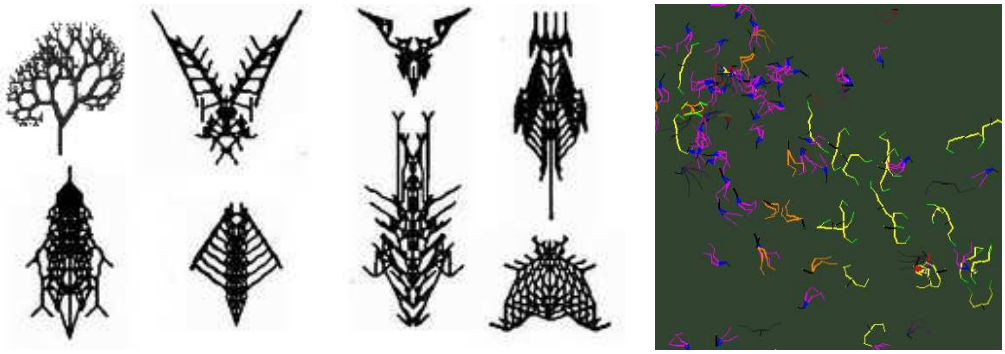


FIGURE 2.12 – *Les robots modulaires utilisent des modèles de morphogenèse artificielle afin de produire divers comportements en modifiant leur morphologie. De gauche à droite : la capacité de duplication du robot en récupérant des blocs dans l'environnement ; le simulateur permettant au robot de développer un comportement adapté au problème posé en fonction de sa morphologie ; le robot modulaire en fonctionnement après l'évolution du comportement et de la morphologie.*

en utilisant une représentation interne de leur morphologie et en apprenant à l'utiliser pour se déplacer, d'abord dans un environnement simulé, puis dans le monde réel [Lipson, 2006, Lipson, 2007]. On observe même des capacités de duplication du robot : en récupérant des modules dispersés dans l'environnement, le robot assemble un double de lui-même qui sera à son tour capable de se reproduire. La figure 2.12 montre quelques exemples de capacités de tels robots modulaires. Un autre exemple est le robot ATRON [Christensen, 2007] qui est capable de modifier sa morphologie également basée sur des modules afin de se déplacer dans l'environnement.

2.2.3 Evolution interactive et Ecosystèmes

Les créatures présentées précédemment utilisent des composants de haut niveau pour créer leur morphologie et leur comportement. Une approche bio-inspirée a été introduite par Dawkins [Dawkins, 1986] avec ses Biomorphs (figure 2.13(a)). En utilisant des règles simples pour produire des segments continus, il a développé un modèle capable de produire de petites créatures graphiques. La particularité de son modèle est la possibilité d'intervenir dans le processus de sélection des créatures développées en sélectionnant la meilleure créature lors de chaque génération de l'algorithme génétique.



(a) biomorphs (b) Darwin Pond

FIGURE 2.13 – *(a) Exemples de créatures dont l'évolution est guidée par l'utilisateur. (b) Créatures produites par coévolution dans Darwin Pond de Ventrella.*

L'ajout d'un système comportemental à ces formes de vies simples a permis l'obtention d'un monde virtuel 2D complexe [Ventrella, 1998b, Ventrella, 1998a] dans lequel coévoluent différentes petites créatures filiformes plongées dans un environnement composé de différentes sources d'énergies (figure 2.13(b)). Chaque créature possède un niveau d'énergie vitale et doit survivre dans l'environnement en cherchant de la nourriture. Ce modèle produit un écosystème complet avec une chaîne alimentaire complète. Les créatures sont aussi capables de se reproduire entre elles pour créer de nouvelles formes de vie. EvolGL

[Garcia Carbajal et al., 2004] est un autre projet d'écosystème dans lequel les créatures appartiennent à différentes classes telles que les herbivores, les carnivores ou les omnivores. Ceci permet l'émergence de stratégies de survie complexe.

2.3 Simuler leur développement

Après avoir fait évoluer la morphologie et le comportement en parallèle et obtenu des résultats très encourageants, l'étape suivante consiste à simuler le développement complet des créatures en s'appuyant sur les connaissances biologiques de plus en plus nombreuses dans le domaine de l'embryogenèse. L'idée est de développer des créatures artificielles en partant d'une cellule unique, comme pour les êtres vivants naturels. Ce domaine, couramment appelé embryogenèse artificielle, s'appuie sur les automates cellulaires développés par John Von Neumann [Von Neumann and Burks, 1966] tout en s'inspirant de données biologiques pour produire des modèles de développement capables de développer des formes et/ou de produire des fonctions élémentaires.

2.3.1 Les automates cellulaires

Les premiers modèles de développement cellulaire sont les automates cellulaires. Ils utilisent des règles de voisinage pour faire évoluer une matrice de cellules. Les règles donnent l'état à l'instant *t+1* de chaque cellule (en vie, morte, etc.) en fonction de l'état à l'instant t des cellules voisines. Dans ces règles, il est possible de prendre en compte le voisinage de Von Neumann (l'état de la cellule elle-même ainsi que les états des quatre voisines situées sur les quatre points cardinaux), le voisinage de Moore (l'état de la cellule actuelle ainsi que celui des huit cellules l'entourant), le voisinage radial (l'état de la cellule et des cellules dans un cercle de rayon 2) ou encore le voisinage de Margolus (l'état de la cellule ainsi que celui des deux blocs diagonaux).

John H. Conway [Gardner, 1970] a défini le célèbre "jeu de la vie" qui se base sur

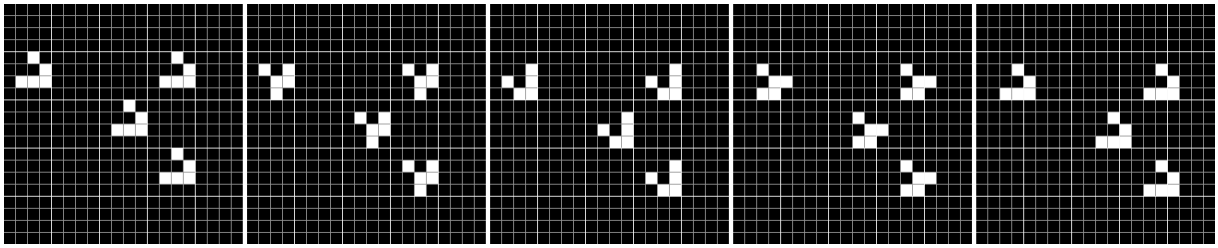


FIGURE 2.14 – *Exemple d'une forme produite par le jeu de la vie de Conway. Ici, un planeur, capable de se déplacer dans son environnement génération après génération.*

des cellules à deux états (vivante ou morte), utilisant un voisinage de Moore avec deux règles simples [13]. Grâce à cette méthode, différentes formes émergent et sont capables de se déplacer, de se reproduire, de fusionner, etc. (illustré par la figure 2.14).

En ajoutant un algorithme génétique pour trouver les règles de transition d'un état à l'autre, Hugo de Garis développa un certain de nombre de formes en 2D [de Garis, 1999]. Pour cela, les règles employées diffèrent quelque peu de celles du jeu de la vie. On part d'une cellule unique dans l'environnement qui a la possibilité de se reproduire. Une cellule ne peut se reproduire que si elle possède un voisin libre. En partant de ceci, on obtient 14 possibilités d'état de reproduction des cellules. Le génome des règles de reproduction est alors le suivant :

- Gène 0 : Nombre d'itération de développement.
- Gène 2^i : Quel état de reproduction est possible pour l'itération i ?
- Gène $2^i + 1$: Direction de reproduction pour l'itération i.

La fonction d'évaluation guidant la convergence de l'algorithme génétique est donnée par la formule suivante :

$$fitness = \frac{Nb_{in} - \frac{1}{2}Nb_{out}}{des}$$

avec

- Nb_{in} le nombre de cellules vivantes dans la forme désirée,
- Nb_{out} le nombre de cellules vivantes en dehors de la forme désirée,
- des le nombre total de cellules de la forme désirée.

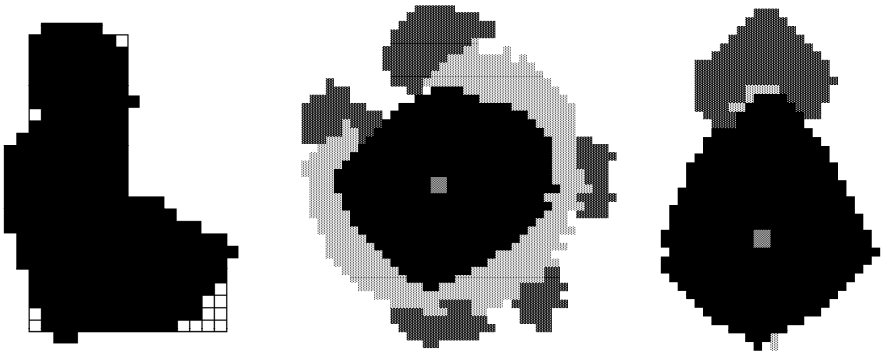


FIGURE 2.15 – *Exemples de formes produites avec des automates cellulaires.*

Il arrive alors à produire différentes configurations telles que des formes simples (tri-

13. Les deux règles sont les suivantes :
- Si la cellule est vivante et est entourée par deux ou trois cellules vivantes, elle reste en vie à la génération suivante, sinon elle meurt ;
- Si la cellule est morte et est entourée par exactement trois cellules vivantes, elle devient vivante.

angles, carrés...) ou plus complexes (des lettres, des tortues, des bonshommes de neige...) comme illustrés en figure 2.15.

2.3.2 Les réseaux booléens aléatoires

Pour simuler la spécialisation cellulaire, Kauffman [Kauffman, 1969] a introduit les réseaux booléens aléatoires (en anglais RBN pour Random Boolean Network) qui ont été réutilisés par Dellaert [Dellaert and Beer, 1994]. Un RBN est un réseau dans lequel chaque noeud possède un état booléen : actif ou inactif. Les noeuds sont interconnectés à l'aide de fonctions booléennes, représentées par des arcs dans le réseau. L'état $t+1$ dépend de ces fonctions booléennes appliquées aux valeurs des noeuds au temps t. L'interprétation de ce réseau de régulation est simple : à chaque noeud du réseau correspond un gène et l'état de ce noeud représente son activation ou son inhibition. La fonction finale de la cellule sera déterminée lors de la transcription du génome.

Dans son modèle de développement, Dellaert utilise un réseau booléen comme réseau de régulation génétique afin de contrôler la spécialisation cellulaire. Son modèle est basé sur une matrice 2D permettant une division simple des cellules : l'organisme initial est constitué d'une seule cellule recouvrant toute la grille ; lors de la première division, la grille est séparée en deux horizontalement ou verticalement (le sens de division est contrôlé par le réseau de régulation) et la nouvelle cellule a la possibilité de se spécialiser. La spécialisation est observée par un changement de couleur de la cellule. Un algorithme génétique permet l'évolution du réseau de régulation afin d'obtenir la forme désirée. Il arrive ainsi a obtenir les images illustrées dans la figure 2.16 avec un comportement de division et de spécialisation en fonction du pattern à réaliser.

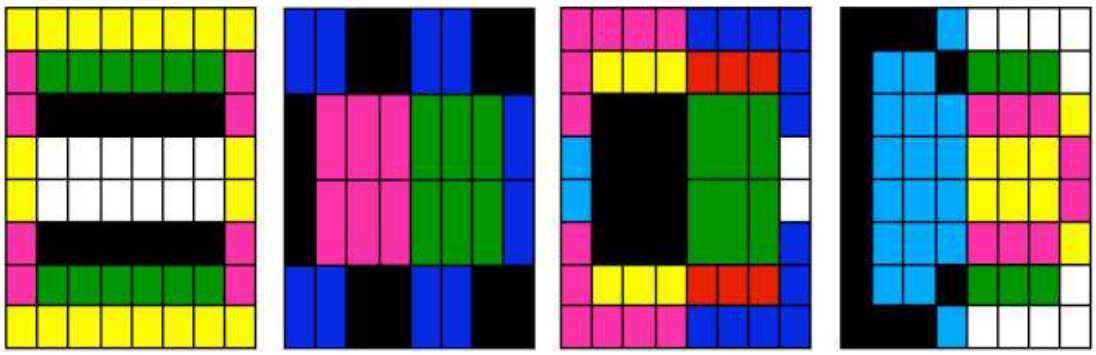


FIGURE 2.16 – *Illustration du modèle de développement de Dellaert qui utilise un réseau booléen pour la régulation génétique.*

2.3.3 Les réseaux artificiels de régulation de gènes

Un autre but important de l'embryogenèse artificielle est la simulation des mécanismes de spécialisation cellulaire. Différents travaux existent sur ce sujet. Comme au sein des cellules réelles, ils utilisent dans la plupart des cas un réseau de régulation de gènes (GRN pour Gene Regulation Network).

Chez les êtres vivants, les cellules d'un même organisme ont différentes fonctions, celles-ci étant décrites dans le génome de l'organisme et contrôlées par un réseau de régulation (GRN) [Davidson, 2006]. Les cellules utilisent des signaux de l'environnement extérieur grâce à des recepteurs protéiques. Le GRN, lui aussi décrit dans le génome, utilise ces signaux pour activer ou inhiber la transcription de certains gènes en ARN messager, le futur modèle de l'ADN de la cellule fille. L'expression de ces gènes dans la cellule fille donnera ses différentes capacités. La figure 2.17 schématise le fonctionnement du GRN.

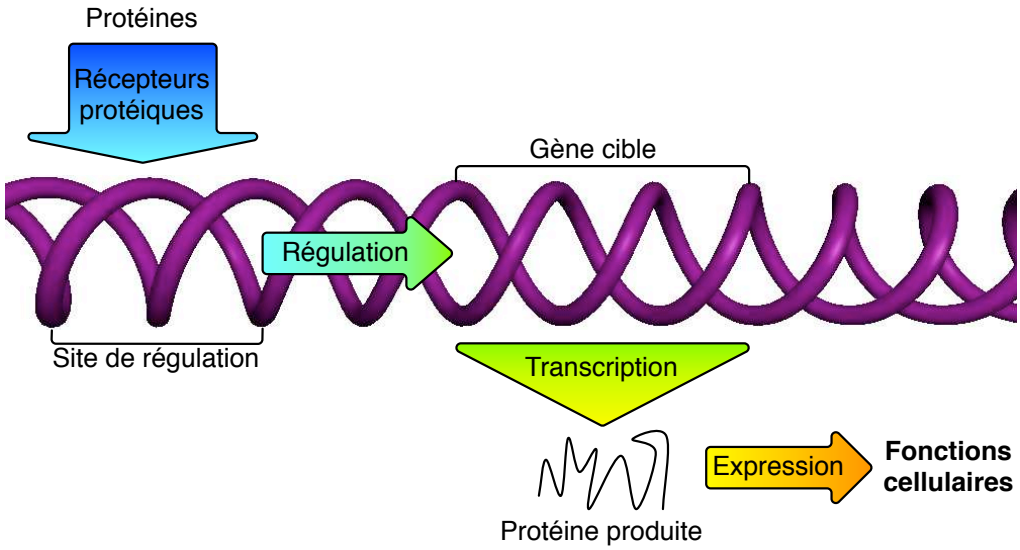
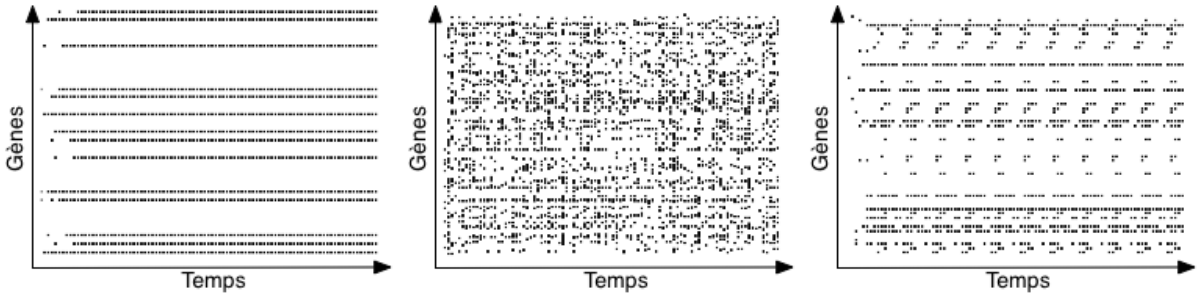


FIGURE 2.17 – *Schéma de l'action du réseau de régulation de gène durant la division cellulaire.*

Les modèles de réseaux de régulation de gènes inspirés de la biologie sont apparus il y a environ 10 ans. Torsten Reil fût le premier à présenter un modèle biologiquement plausible [Reil, 1999]. Il définit le génome de ses créatures comme une suite de nombres. Le nombre de gènes du génome n'est pas défini à l'avance. Chaque gène débute par la séquence particulière 0101, appelée "promoteur". Ce type de structure se retrouve dans la nature avec le bloc $TATA$[14] tenant cette fonction de promoteur dans les génomes. Les bits suivants un promoteur définissent le gène lui-même. Comme présenté dans la

14. T=Thymine et A=Adénine

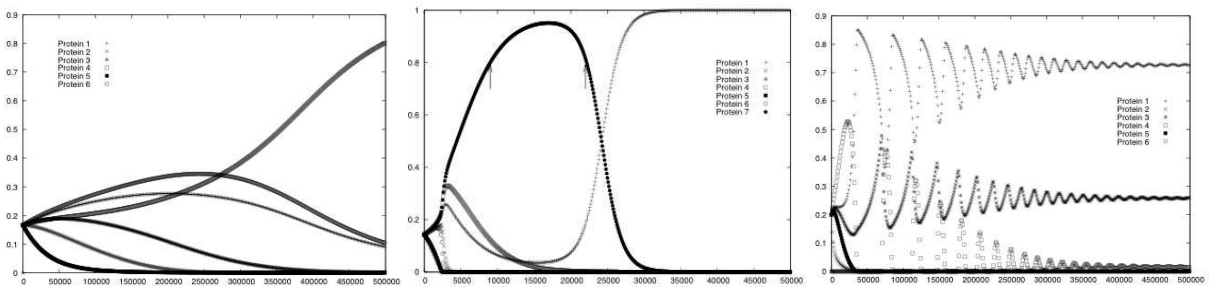


(a) Ordonné (b) Chaotique (c) Cyclique

FIGURE 2.18 – *Représentation graphique de l'expression des gènes de Torsten Reil. Il observe différents patterns d'expression des gènes générés aléatoirement.*

figure 2.18, il utilise une visualisation sous forme de graphe pour observer l'activation et la désactivation des gènes au cours du temps à partir de réseaux générés aléatoirement. On observe alors différents patterns tels qu'un certain ordonnancement de l'activation des gènes, des expressions chaotiques ou la formation de cycles. Reil remarque de plus que même en cas d'altérations aléatoires dans le génome, celui-ci retrouve le même pattern d'expression après un certain temps d'oscillation.

En 2003, Wolfgang Banzhaf imagine un modèle de réseau de régulation artificiel [Banzhaf, 2003]. Dans ses travaux fortement inspirés des réseaux de régulation de gènes issus de la biologie, le génome est codé comme une chaîne d'entiers de 32-bits (et donc comme une chaîne de bits). Chaque gène de ce génome commence par un "promoteur" constitué par la séquence "XYZ01010101" avec XYZ une séquence quelconque de bits complétant l'entier. La combinaison "01010101" a une probabilité d'apparition dans une chaîne de bits de 2^{-8} soit environ 0.39%. Le gène codé après ce promoteur a une taille fixe de cinq entiers (160 bits, chaque entier étant composé de 32 bits). Les deux premiers entiers de cette série de cinq permettent de coder la production de protéines (une activatrice



(a) régulières (b) transitoires (c) oscillantes

FIGURE 2.19 – *Exemples de courbes de concentration de protéines en fonction du temps obtenues grâce au réseau de régulation de gènes de Banzhaf (Source [Banzhaf, 2003]).*

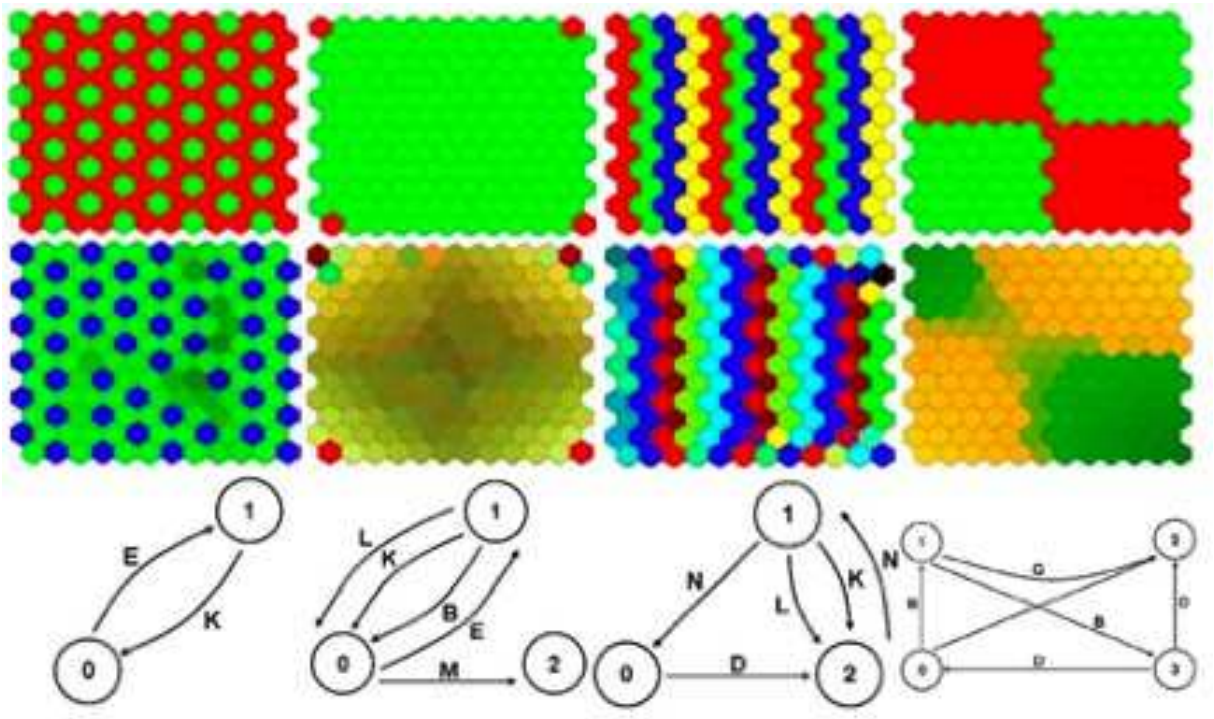


FIGURE 2.20 – *Exemples de formes produites par le réseau de régulation de gènes de Flann. La première ligne montre la forme attendue, la ligne centrale montre le pattern obtenu après évolution du réseau et la dernière montre le réseau lui-même.*

et une inhibitrice) permettant la régulation génétique du gène. Dans ce modèle, toute la partie de transcription de l'ADN, via l'ARN par exemple, est exclue pour se focaliser sur la régulation des gènes. Ce type de génome permet d'obtenir une évolution différente de la production des protéines au cours du temps, comme montré sur les courbes de la figure 2.19. Ces courbes ont été obtenues en générant aléatoirement un génome et en exprimant ensuite les gènes en fonction de leur activation et inhibition.

Un certain nombre de chercheurs se sont basés sur ces deux derniers modèles pour développer leur réseau de régulation de gènes ou pour l'appliquer sur des problèmes particuliers. C'est le cas de Bongard et Pfeifer qui ont utilisé un modèle proche de celui de Reil pour développer un robot modulaire [Bongard and Pfeifer, 2003]. Ce robot possède un réseau de neurones permettant le contrôle du déplacement de chaque module. L'expression de l'activation et de l'inhibition génétique permet d'activer ou d'inhiber 23 transformations phénotypiques prédéfinies telles que augmenter la taille d'un module, diviser un module en deux, modifier les paramètres ou même la topologie du réseau de neurones...

Flann et ses collègues ont utilisé une implantation en graphe de réseau de régulation de gènes pour développer des images composées de cellules spécialisées (illustré par la figure 2.20 issue de [Flann et al., 2005]). Les noeuds du réseau correspondent aux différents

niveaux d'expression des protéines, et les arcs les reliant représentent les interactions entre les protéines. Alors que les formes simples sont facilement obtenues avec ce type de réseau, il a fallu en utiliser plusieurs en parallèle en combinant les niveaux de concentration de protéines pour produire des formes plus complexes. Dans ces travaux, la spécialisation cellulaire est mise en exergue grâce à la couleur de celle-ci.

Une application courante de ces réseaux de régulation de gènes est le problème du drapeau français [Wolpert, 1968]. Introduit par Wolpert à la fin des années 1960, il consiste à développer un drapeau français, avec ses trois couleurs (bleu, blanc et rouge) en partant d'une seule cellule au centre. Il permet de mettre en valeur les capacités de différenciation des modèles de développement. Lindenmayer l'a utilisé pour montrer les capacités des L-Systèmes à générer une forme prédéfinie [Lindenmayer, 1971]. Miller l'a aussi résolu en utilisant son modèle de programmation génétique cartésienne, en y ajoutant la propriété d'auto-réparation du drapeau [Miller, 2003]. Bowers l'utilise pour sa part pour tester son modèle de développement embryogénique [Bowers, 2005].

Dans sa thèse [Devert, 2009], Devert tente de résoudre ce problème en utilisant différentes méthodes :

- les réseaux de neurones NEAT de Stanley [Stanley, 2004],
- les "Echo State Networks" de Jaeger [Jaeger, 2001],
- un modèle de réaction-diffusion proche de celui de Miller
- et l'étude de l'influence du critère d'arrêt.

La dernière partie de son étude est particulièrement intéressante puisqu'elle met en valeur l'importance du critère d'arrêt du processus de développement si on veut obtenir un organisme stable. Devert en conclut que plus le processus de développement est court et plus l'organisme obtenu sera stable et peu influencé par les perturbations de son milieu durant son développement. Ainsi, en ajoutant un critère de temps de développement à la fonction d'évaluation des créatures, il est possible d'obtenir de meilleurs individus.

Les réseaux de régulation génétique sont aussi parfaitement adaptés à ce genre de problème. Alors que certaines formes de Flann ressemblent déjà à un drapeau, Chavoya et Duthen ont développé un modèle de réseau de régulation basé sur le modèle de Banzhaf pour résoudre ce problème [Chavoya and Duthen, 2008] (illustré par la figure 2.21). Ils utilisent un automate cellulaire comme présenté précédemment dont les règles sont activées et désactivées à l'aide du réseau. Des gradients de morphogènes, pré-positionnés dans l'environnement, permettent aux cellules de se localiser dans l'environnement et de générer des informations supplémentaires pour le réseau de régulation de gènes. Il obtient ainsi un drapeau parfait et de taille variable mais aussi un certain nombre de formes telles que des carrés multicolores, des triangles, des pyramides (3D), etc.

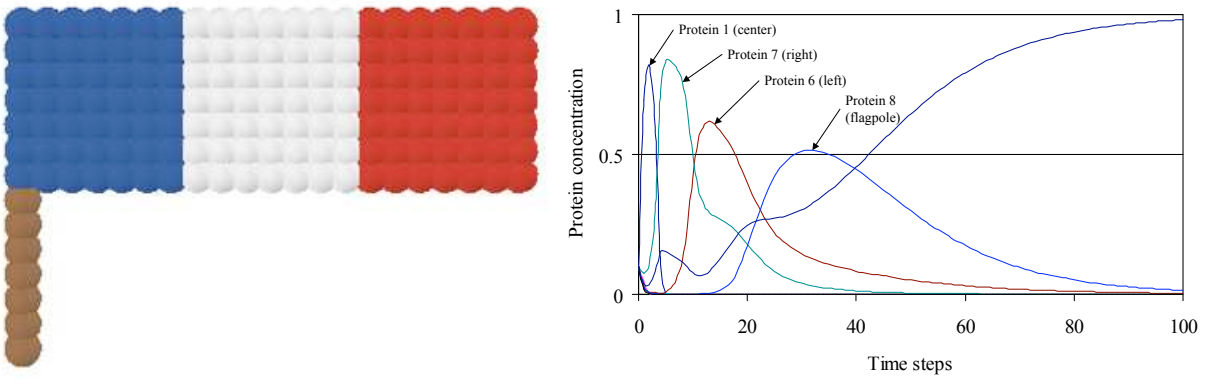


FIGURE 2.21 – *Le drapeau français développé par Arturo Chavoya (à gauche) avec la courbe des concentrations en protéines associées (à droite). Ce problème permet de tester les capacités de spécialisation cellulaire des modèles de développement. La spécialisation est dans ce cas illustrée par le changement de couleur des cellules. (Source : [Chavoya and Duthen, 2008])*

Dans son étude des systèmes complexes, Doursat utilise un modèle basé sur des niveaux d'expression de gènes pour simuler le processus de développement des formes complexes [Doursat, 2009, Doursat, 2008]. Son modèle intègre un réseau de régulation de gène composé de trois couches :

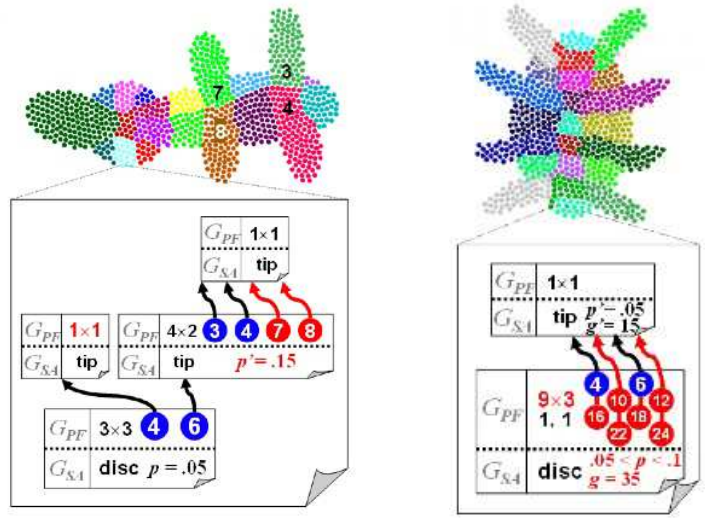
- la couche basse utilise les données de positionnement fournies par les morphogènes,
- la couche centrale contient les gènes de "démarcation" qui permettent une segmentation horizontale et verticale de l'embryon ainsi que la description de la régulation génétique,
- la couche supérieure permet quant à elle de déterminer la production de la protéine régulatrice en fonction de l'expression des gènes activateurs et inhibiteurs de la couche centrale.

Ce réseau de régulation s'appuie sur un modèle de développement fonctionnant grâce à deux règles simples :

1. la *division* cellulaire qui permet à chaque cellule de l'organisme de se diviser avec une certaine probabilité,
2. l'*adhésion* intercellulaire permettant la cohérence de l'organisme et qui se base sur des forces élastiques proches d'un modèle masse/ressort.

En partant d'un amas de cellules plongé dans un environnement 2D contenant deux types de morphogènes permettant le positionnement des cellules, l'auteur arrive à produire un organisme possédant différentes régions de cellules diversifiées et dont la forme ressemble à une salamandre (figure 2.22(a)). En modifiant les paramètres de l'expérimentation (en particulier les différents poids ou les fonctions intervenants dans le réseau de

régulation), il parvient à modifier radicalement la morphologie de l'organisme obtenu en partant du même amas cellulaire et en produisant un organisme possédant huit pattes (figure 2.22(b)).



(a) La salamandre (b) l'organisme à 8 pattes

FIGURE 2.22 – *Formes développées grâce au modèle de développement de Doursat [Doursat, 2009]*

2.3.4 La chimie artificielle

Définition

La notion de chimie artificielle a été formellement introduite par Dittrich et ses collègues en 2001 [Dittrich et al., 2001]. Ils définissent le concept de chimie artificielle comme étant "un système créé par l'homme similaire à un système chimique réel". Ils la décrivent plus formellement comme étant un triplet (S, R, A) dans lequel :

- S est l'ensemble des molécules possibles,
- R est l'ensemble des règles d'interaction entre les molécules d'un sous-ensemble de S,
- A est un algorithme de réaction qui définit la méthode d'application des règles de l'ensemble S à une collection P de molécules qui contient toutes les molécules de l'environnement (avec de possibles répétitions si une ou plusieurs molécules sont présentes) aussi appelée *soupe* ou *population*.

De façon plus précise, une règle $r \in R$ peut être écrite comme une équation chimique conventionnelle, sous la forme :

$$s_1 + s_2 + ... + s_n \rightarrow s'_1 + s'_2 + ...s'_m$$

Une règle de réaction r détermine quelles molécules $s_i \in S$ peuvent interagir (les réactifs, dans la partie gauche de l'équation), pour être remplacées par un certain nombre de composants $s'_k \in S$ (les produits, dans la partie droite de l'équation). n est appelé l'ordre de la réaction chimique. Une règle ne peut être appliquée que si un certain nombre de conditions sont respectées dont la principale est que tous les réactifs doivent être disponibles ; les autres étant décrites par l'algorithme de réaction A.

Il existe différents algorithmes de réaction. Dans l'approche par *collisions moléculaires stochastiques*, chaque molécule est explicitement simulée. Cet algorithme sélectionne aléatoirement un sous-ensemble des molécules présentes dans la population P. Si une règle d'interaction de R peut s'appliquer à cet ensemble, les réactifs de cette règle sont supprimés de P à l'instant t et les produits sont ajoutés à P à l'instant $t + 1$.

Une deuxième méthode, à l'aide d'*équations différentielles continues*, consiste à décrire les variations de concentration de chaque molécule du système. Ici, une réaction r peut être écrite sous la forme

$$a_1 s_1 + a_2 s_2 + ... + a_N s_N \rightarrow b_1 s_1 + b_2 s_2 + ... + b_N s_N$$

ou N est le cardinal de S, $s_i \in S, i \in 0..N$ est une molécule et $a_i, b_i \in \mathbb{R}, i \in 0..N$ sont les coefficients stoechiométriques. Si ce facteur est égal à zéro, cela signifie que la molécule associée n'intervient pas dans la réaction. Ainsi, on peut écrire la modification globale de concentration d'une molécule s_i à l'aide de l'équation :

$$\frac{ds_i}{dt} = \sum_{r \in R} \Big(\prod_{j=1}^{N} b_j^r s_j^r - \prod_{k=1}^{N} a_k^r s_k^r \Big)$$

avec dans le cas de cette équation, s_i qui représente la concentration de la molécule et non la molécule elle-même et a^r et b^r qui correspondent aux coefficients stoechiométriques à la règle d'interaction r.

L'approche *métadynamique* du problème suppose que le nombre de molécules et de règles peut changer durant la simulation. Un ensemble de molécules est ici activé ou désactivé si leur concentration dans l'environnement est supérieure ou inférieure à un certain seuil. Etant donné que les concentrations varient au cours du temps, les molécules

activées et désactivées varient elles aussi.

Les modèles appliqués à la vie artificielle

Ces approches ont principalement été utilisées pour simuler le développement et la maintenance de la membrane d'une ou d'un petit groupe de cellules. Ainsi, Ono et Ikegami ont utilisé cette approche pour construire un modèle simple de proto-cellule[15] qui simule les dynamiques chimiques dans un matrice 2D [Ono and Ikegami, 1999]. Ce modèle a aussi permis de simuler la division cellulaire. En modifiant les forces répulsives entre les molécules, ils ont obtenu différentes formes de division ou, pour certaines valeurs, une instabilité du système provoquant la mort de la cellule par destruction de la membrane.

Rasmussen et al. proposent dans [Rasmussen et al., 2003] une approche plus simple permettant la simulation d'une proto-cellule capable de se servir de l'environnement pour se nourrir. Ils ont simulé la partie métabolique de la cellule en utilisant des données thermodynamiques issues d'expériences *in vivo*.

Dans [Mavelli and Ruiz-Mirazo, 2007], Mavelli et Ruiz-Mirazo ont utilisé un modèle plus réaliste pour simuler la formation d'une membrane cellulaire et d'en étudier la stabilité au cours du temps. Ils ont ainsi réussi à retrouver certaines caractéristiques biologiques issues d'expériences *in vitro* telles que l'homéostasie, c'est-à-dire la capacité du modèle à trouver un équilibre chimique même en cas de forte perturbation de l'environnement.

2.3.5 Les modèles de développement

Le modèle de Fleischer et Barr

Fleischer et Barr ont développé un modèle de développement cellulaire donc le but initial était l'application à la génération de réseaux de neurones [Fleischer and Barr, 1992]. Cependant, leur intérêt a rapidement été d'étudier les différents mécanismes impliqués dans la morphogenèse. Au moment de leur étude, les travaux précédents tels que [Turing, 1952, Odell et al., 1981, Lindenmayer, 1968, Fraser and Perkel, 1990] simulaient individuellement les différents mécanismes intervenant durant le développement d'un être vivant, tels que les facteurs chimiques, mécaniques, électriques, génétiques ou la spécialisation cellulaire. Ils décidèrent alors de combiner tous ces mécanismes pour produire un modèle complet permettant de comprendre l'interaction entre ces différents facteurs.

En plus des mécanismes précédemment évoqués, le modèle laisse la possibilité aux cellules de se déplacer librement dans l'environnement. La migration cellulaire est un

15. une cellule simple sans noyau

aspect essentiel de la simulation de développement de réseau de neurones. Les cellules ont aussi la capacité de communiquer de manière directe, de cellule à cellule (contact physique entre les cellules par exemple), ou indirecte, par le biais de l'environnement (modification d'une concentration chimique).

Ce modèle leur a ainsi permis de simuler des comportements cellulaires simples tels que le suivi de morphogènes, le regroupement de cellules, la formation de patterns ou la génération de réseaux.

Le modèle d'Eggenberger Hotz

Eggenberger Hotz [Eggenberger Hotz, 2004] a imaginé un modèle capable de produire une créature simple dont la forme est définie par l'utilisateur et qui est capable de se déplacer dans un environnement en utilisant seulement les réseaux de régulation de gènes. Pour cela, il a imaginé un modèle permettant de simuler le mécanisme naturel de division asymétrique des cellules. Cette division asymétrique permet la spécialisation cellulaire en produisant des cellules filles avec différentes protéines. Ceci permet un placement précis de cellules spécialisées dans l'organisme qui se développe [Horvitz and Herskowitz, 1992]. La production de protéines particulières par le réseau de régulation de gènes permet d'orienter le plan de division de la cellule. Ce plan donnera par la suite la direction de division. Le réseau de régulation contrôle en même temps la régulation de ses propres gènes qui correspondent à des coefficients d'adhésion cellulaire et de dynamique physique des cellules. Les cellules émettent périodiquement des molécules qui modifient les propriétés d'adhésion entre cellules et entre les cellules et l'environnement. Eggenberger Hotz a ainsi développé un simulateur et produit une créature en forme de T qui croît et se déplace dans l'environnement en modifiant sa morphologie [Eggenberger Hotz, 2004]. Il permet aussi de simuler les mécanismes d'invagination de la phase de gastrulation des vertébrés [Eggenberger Hotz, 2003]. Cette phase aboutit au placement des morphogènes et des cellules déjà spécialisées dans l'embryon afin de faire apparaître par la suite les quatre membres et la tête.

EDS (Evolutionary Development System)

Kumar et Bentley ont développé un modèle de développement évolutionnaire (EDS pour Evolutionary Development System) qui a pour but d'examiner les processus de développement cellulaire et leurs possibles applications en informatique. Le modèle implémente pour cela les différents concepts-clés du développement multicellulaire tels que les embryons, les cellules, le cytoplasme cellulaire, les protéines, les récepteurs protéiques,

la transcription génétique et les sites de régulation génétique.

La différence majeure avec les autres modèles est la décomposition du génome en deux chromosomes. Le premier code les valeurs des paramètres liés aux protéines régulatrices (taux de synthèse, vitesse de diffusion et d'évaporation, force des interactions, types de protéines). Le second chromosome décrit le rôle de chaque protéine dans le processus de régulation des différents gènes. Alors que le premier génome n'est utilisé que pour l'initialisation des paramètres des protéines, le second est contenu dans chaque cellule de l'organisme.

Les cellules du modèle sont représentées comme des agents autonomes contenant différents capteurs leur donnant les informations sur les concentrations des différents constituants de leur environnement.

Un algorithme génétique permet l'évolution du génome des créatures. Du fait de la complexité du modèle, peu d'expérimentations ont pu être réalisées. Elles ont principalement consisté à montrer les possibilités des gènes et les protéines à interagir et former un réseau de régulation génétique dans une cellule unique et à faire évoluer un embryon multicellulaire 3D afin d'obtenir une forme prédéfinie simple telle qu'une sphère.

METAMorph

METAMorph (Model for Experimentation and Teaching in Artificial Morphogenesis) est un modèle open source pour la simulation de développement cellulaire avec un haut niveau d'abstraction [Stewart et al., 2005]. Les organismes multicellulaires générés grâce à ce modèle se développent à partir d'une cellule unique et utilisent un réseau de régulation de gènes pour contrôler leur développement. Les protéines de ce réseau de régulation possèdent différentes propriétés :

- un *nom* qui permet d'identifier de façon unique chaque protéine,
- un *type*, interne ou externe, qui indique si la protéine est à l'intérieur ou à l'extérieur de la cellule,
- une constante de *diffusion* qui donne la vitesse de diffusion de la protéine dans l'environnement,
- une constante d'*évaporation* qui donne la vitesse d'évaporation de la protéine.

Le génome d'un organisme est composé de différents gènes, identiques pour toutes les cellules, produisant une seule et unique protéine. Cependant, une protéine peut être produite par différents gènes. Chaque gène possède zéro ou plusieurs promoteurs permettant la régulation de la production de la protéine en fonction des autres protéines contenues dans l'environnement de la cellule. Un promoteur est ici composé du nom de la protéine régulatrice et d'un poids positif pour activer ou négatif pour inhiber la production de la

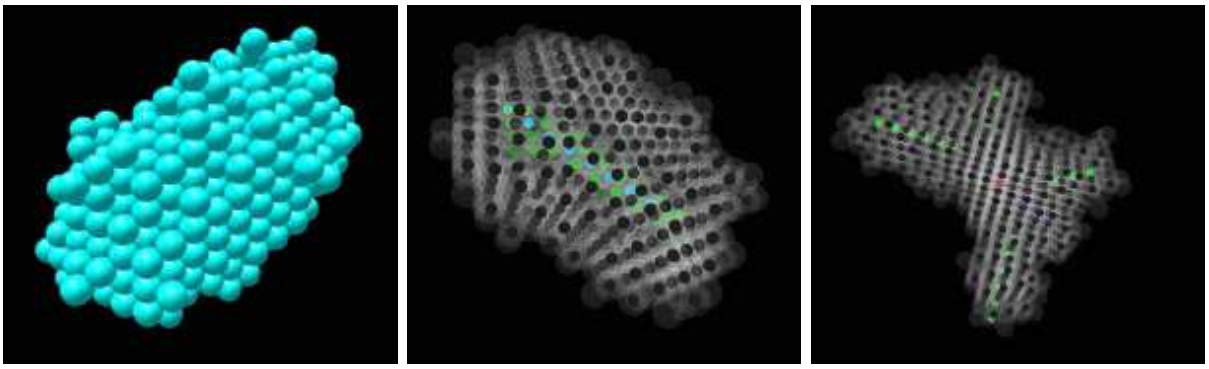


FIGURE 2.23 – *Exemple de créatures générées avec le modèle METAMorph : à gauche, un cigare, au centre, un tube (le centre est creux) et à droite, un quadrupède.*

protéine du gène. On peut ainsi calculer la concentration de la protéine produite par un gène x à l'instant $t+1$ par la formule :

$$conc_{prot}(x, t+1) = conc_{prot}(x, t) + \frac{1}{1 + e^{-pente(a_{prot} - biais)}}$$

$$a_{prot} = \sum_{prot}^{promoteur} poids_{prot} conc_{prot}(x, t)$$

La *pente* et le *biais* sont des paramètres donnés, eux aussi issus du génome de l'organisme. a_{prot} correspond au coefficient d'activation ou d'inhibition donné par la région régulatrice du gène.

Les cellules sont représentées comme des sphères dans un environnement 3D constitué d'une grille isospatiale. Ainsi, chaque cellule possède douze voisins équidistants. Elle a la possibilité d'effectuer quatre actions :

- *division* : lorsqu'elle a un voisin disponible, la cellule peut se diviser selon un plan de division,
- *modification du plan de division* : permet d'orienter la cellule en vue d'une division vers un des douze voisins de la cellule,
- *apoptose* : permet à la cellule de libérer sa place dans l'environnement en déclenchant une mort programmée,
- *différenciation* : permet à la cellule de se spécialiser en modifiant sa couleur mais cette différenciation ne modifie pas son comportement.

Ce modèle ne possède cependant aucun système d'évolution du génome, ce qui réduit grandement ses capacités de développement d'organisme de grande taille. En effet, le génome de l'organisme doit être trouvé par l'utilisateur à l'aide d'une suite d'essais/erreurs souvent fastidieuse. L'auteur a cependant réussi à développer un organisme de la forme

d'un cigare et un quadrupède (figure 2.23).

Le modèle de Knabe

Knabe a produit un modèle contenant un réseau de régulation de gènes dans lequel l'automate cellulaire de Chavoya est remplacé par un système de croissance cellulaire [Knabe et al., 2008]. Les cellules évoluent dans une matrice de pixels. Elle se développent en absorbant des pixels libres en suivant un axe de développement contrôlé par le réseau de régulation de gènes. Celui-ci indique à la cellule son ratio $\frac{hauteur}{longueur}$ et lui indique donc sa direction de croissance en fonction de sa forme actuelle. Une fois que la cellule est composée de 24 pixels, elle entre en phase de mitose. Elle se divise alors selon l'axe orthogonal à sa dernière direction de développement. De plus, dans ce modèle, les morphogènes ne sont pas pré-positionnés par l'utilisateur mais directement produits par les cellules. Ces dernières doivent ainsi réguler leur production et la varier afin d'obtenir des concentrations permettant de réaliser la forme désirée. Les auteurs ont testé ce modèle en utilisant le problème du drapeau français précédemment présenté. Pour cela, ils ont utilisé un algorithme génétique avec une fonction d'évaluation qui évalue la différence d entre l'individu testé T et le pattern attendu R^i de taille w x h :

$$d(R^i, T) = \frac{1}{wh} \sum_{x=0}^{h-1} \sum_{y=0}^{h-1} |sgn(R^i_{xy} - T_{xy})|$$

Les résultats obtenus (illustrés par la figure 2.24) sont proches du pattern désiré dans 75% des cas.

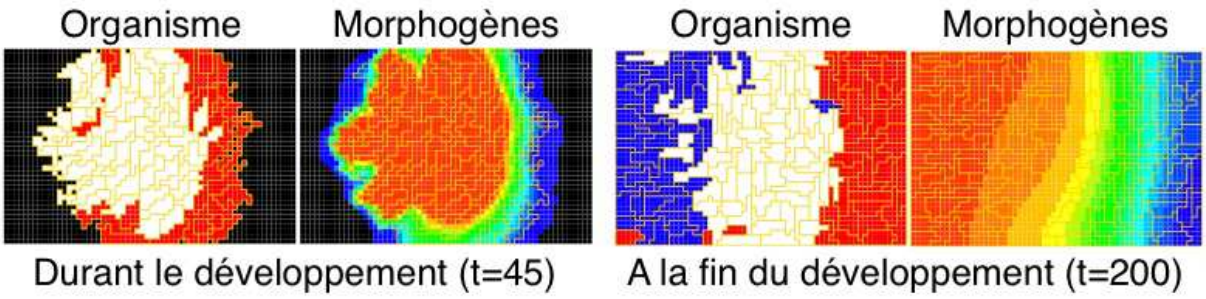


FIGURE 2.24 – *Exemple de développement du drapeau français avec le modèle de Knabe. A gauche, on peut observer l'organisme en cours de développement (pas de simulation 45) avec l'organisme lui-même et la diffusion des morphogènes produits par l'organisme. A droite, l'organisme final, en fin de développement (pas de simulation 200). On observe des cellules de différentes formes et correctement spécialisées pour obtenir un pattern proche de celui attendu (Source : [Knabe et al., 2008]).*

Les modèles récents

Nous pouvons citer un certain nombre de nouveaux travaux très intéressant mais sur lesquels il n'y a actuellement que peu de recul. Ils sont apparus dans les dernières conférences ALife XI, CEC 2009 et ECAL 2009. Ces travaux sont dans la continuation des travaux précédemment présentés.

Nous pouvons citer le modèle de Joachimczak [Joachimczak and Wróbel, 2008] qui a développé un modèle de croissance basé sur un environnement physique 3D. Il permet le développement de structures 3D en partant d'une seule cellule possédant un réseau de régulation de gènes et régulant elle-même la quantité de morphogènes à produire dans l'environnement afin de guider le développement de ses cellules filles. Les cellules possèdent aussi différentes tailles en fonction de leur stade d'évolution. Il arrive ainsi à obtenir des formes relativement simples telles qu'un écrou ou une haltère. Dans ses tout derniers travaux [Joachimczak and Wróbel, 2009], il montre les capacités de différenciation de son modèle en étendant le problème du drapeau français à la troisième dimension. Il introduit aussi les possibilités de réparation en détruisant un certain nombre de cellules au début du développement de son organisme.

Tufte a quant à lui proposé un modèle de développement pour étudier le phénomène de plasticité phénotypique, c'est-à-dire la capacité d'adaptation du génome à un environnement variable [Tufte, 2008]. Nous reviendrons sur ce concept intéressant de plasticité phénotypique par la suite, dans la section 5.2.3. Ces organismes basés sur un automate cellulaire sont capable de se développer afin d'obtenir des fonctions mathématiques simples tels que des compteurs. Il étudie pour cela l'effet du changement de la morphologie de ces organismes pour engendrer la fonction qui leur est demandée [Tufte, 2009]. Pour cela, il utilise une fonction d'évaluation qui est calculée à chaque pas de simulation et non seulement en fin de simulation comme dans la plupart des modèles précédemment présentés.

Haddow a proposé un modèle permettant d'étudier l'implication du réseau de régulation de gènes dans la morphologie finale de l'organisme, principalement en cas de modification de la concentration initiale des protéines dans l'environnement [Haddow and Hoye, 2009].

2.3.6 Propriétés de ces systèmes de développement

La partie précédente regroupe des modèles de développement. Les tableaux 2.2, 2.3 et 2.4 présentés dans cette partie résument les différentes propriétés de ces modèles et nous allons en déduire les caractéristiques que nous cherchons à créer dans notre modèle.

Le tableau 2.2 montre les caractéristiques de l'environnement des principaux modèles précédemment présentés. On peut remarquer qu'une grande majorité d'entre eux utilisent

Modèles	Environnement		
	Représentation graphique	Chimie	Actions mécaniques
Conway	Matrice 2D	Absente	Non
Ono	Environnement 2D	Réaliste	Non
Rasmussen	Environnement 2D	Réaliste	Non
Reil	Courbes	Diffusion	Non
Banzhaf	Courbes	Diffusion	Non
Flann	Grille 2D	Diffusion	Non
Fleisher & Barr	Matrice 2D	Diffusion	Oui
Chavoya & Duthen	Matrice 2D & 3D	Diffusion	Non
Doursat	Environnement 2D	Absente	Oui
Dellaert	Grille 2D	Diffusion	Non
Knabe	Matrice 2D	Diffusion	Non
Eggenberger Hotz	Environnement 3D	Diffusion	Oui
Kumar	Environnement 3D	Diffusion	Oui
Stewart	Environnement 3D	Diffusion	Oui
Joachimczak	Environnement 3D	Diffusion	Oui
Haddow	Matrice 3D	Diffusion	Non
Tufte	Matrice 2D	Absente	Non

TABLE 2.2 – *Table des propriétés des environnements des systèmes de développement.*

une matrice ou un environnement continu à deux dimensions comme base pour l'environnement. Cette technique permet une grande simplification de la simulation en gardant un maximum de degrés de liberté. La chimie est souvent oubliée dans les modèles de développement, sauf dans les modèles où elle est explicitement la cible de l'expérimentation. Comme palliatif, beaucoup de modèles intègrent souvent des techniques de diffusion, souvent capitales pour le positionnement de morphogènes. Enfin, les interactions mécaniques (les forces d'adhésion entre les cellules en particulier) ne sont prises en compte que dans les modèles trois dimensions récents.

Le tableau 2.3 montre les propriétés des cellules contenues dans l'environnement des différents modèles. Tout d'abord, la majorité des modèles utilisent soit une sphère soit un carré pour représenter graphiquement une cellule. Certains modèles utilisent des polygones qui permettent la déformation des cellules lors de la croissance. Ensuite, quand elle est présente, la spécialisation cellulaire, principalement induite par un réseau de régulation de gènes, est représentée par un changement de couleur de la cellule mais pas encore comme une vraie spécialisation cellulaire à proprement parler, c'est-à-dire par une modification du comportement et du type de cellule. La régulation génétique est présente dans une grande majorité des modèles. Elle permet à la fois la spécialisation cellulaire mais aussi la

Modèles	Cellule				
	Forme	Spécialisation	Régulation génétique	Communication	Evolution génétique
Conway	Carré	Non	Non	Aucune	Oui
Ono	Polygone	Non	Non	Indirecte	Non
Rasmussen	Polygone	Non	Non	Aucune	Non
Reil	-	Non	Oui	Aucune	Oui
Banzhaf	-	Non	Non	Aucune	Oui
Flann	Hexagonale	Couleur	Oui	Aucune	Oui
Fleisher	Cercle	Non	Oui	Indirecte	Oui
Chavoya	Cercle/Sphère	Couleur	Oui	Indirecte	Oui
Doursat	Hexagonale	Couleur	Oui	Indirecte	Oui
Dellaert	Quadrilatère	Couleur	Oui	Aucune	Oui
Eggenberger Hotz	Sphère	Non	Oui	Indirecte	Oui
Kumar	Sphère	Non	Oui	Indirecte	Oui
Stewart	Sphère	Couleur	Oui	Indirecte	Non
Knabe	Polygone	Couleur	Oui	Aucune	Oui
Joachimczak	Sphère	Couleur	Oui	Indirecte	Oui
Haddow	Cube	Couleur	Oui	Aucune	Oui
Tufte	Carré	Fonction	Non	Aucune	Oui

TABLE 2.3 – *Table des propriétés des cellules des systèmes de développement.*

régulation de la croissance des organismes dans le but de produire des formes particulières. La communication entre cellules est, quand elle existe, indirecte. En d'autres termes, les cellules utilisent l'environnement via diverses molécules pour communiquer. Enfin, la plupart des modèles utilisent un algorithme génétique pour évoluer un codage génétique des cellules afin d'obtenir le résultat recherché.

Le tableau 2.4 montre les capacités des modèles. La taille des organismes est très variable. Elle peut être de quelques cellules pour les modèles tachant de simuler en détail les activités cellulaires en se rapprochant des mécanismes biologiques. C'est par exemple le cas de la chimie artificielle qui détaille le fonctionnement de tous les échanges intra et extra membranaires. De même, les modèles de Reil et Banzhaf se focalisent sur une seule cellule afin de comprendre l'utilité d'un réseau de régulation de gènes vis à vis de la production de protéines régulatrices. Le but de beaucoup de modèles est de développer une forme prédéfinie par l'utilisateur plutôt que de donner une fonction d'interaction particulière avec l'environnement. Seuls quelques modèles intègrent cette fonction. C'est le cas des modèles basés sur des automates cellulaires qui produisent différents déplacements ou fonctions mathématiques. Un cas plus particulier est le modèle développé par Eggenberger Hotz qui permet à la créature qu'il a fait croître de se déplacer dans son environnement. Il est

Modèles	Capacités			
	Taille de l'organisme	Forme	Fonction	Métabolisme
Conway	∼75 cellules	Oui	Oui	Non
Ono	∼ 6 cellules	Non	Non	Oui
Rasmussen	∼ 6 cellules	Non	Non	Oui
Reil	1 cellule	Non	Non	Non
Banzhaf	1 cellule	Non	Non	Non
Flann	∼ 150 cellules	Oui	Non	Non
Fleisher & Barr	∼ 30 cellules	Non	Oui	Non
Chavoya & Duthen	∼ 250 cellules	Oui	Non	Non
Doursat	∼ 250 cellules	Oui	Non	Non
Dellaert	∼ 30 cellules	Oui	Non	Non
Eggenberger Hotz	∼ 75 cellules	Oui	Oui	Non
Kumar	∼ 100 cellules	Oui	Non	Non
Stewart	∼ 250 cellules	Oui	Non	Non
Knabe	∼ 100 cellules	Oui	Non	Non
Joachimczak	<200 cellules	Oui	Non	Non
Haddow	∼ 20 cellules	Oui	Non	Non
Tufte	∼ 100 cellules	Oui	Oui	Non

TABLE 2.4 – *Table des capacités des systèmes de développement.*

aussi à noter que, sauf dans le cas particulier de la chimie artificielle, le métabolisme des organismes n'est jamais pris en compte, ce qui constitue pourtant une fonction primordiale de toute cellule biologique.

2.4 Positionnement

Le but de nos travaux est de créer un pont entre la morphogenèse artificielle et l'embryogenèse artificielle afin de produire des créatures virtuelles. Pour ce faire, nous avons fait l'hypothèse que les blocs ou les bâtonnets de l'approche morphogénétique pouvaient être considérés comme des organes, c'est-à-dire des parties de la créature possédant une ou plusieurs fonctions. En utilisant des techniques de développement cellulaire pour faire croître la créature, nous voulons créer ses organes en partant d'une cellule unique. Dans ce but, la cellule devra être capable de se spécialiser pour s'adapter à son environnement. L'organisation des cellules en tissus (c'est-à-dire en groupes de cellules qui ont la même fonction) puis l'organisation des tissus permettra la création d'organes. Après avoir créé une bibliothèque d'organes, un mécanisme évolutionniste tel que celui utilisé par Nicolas Lassabe dans sa thèse [Lassabe, 2008] assemblera des organes afin de faire émerger une

créature adaptée à son environnement. Le chapitre suivant détaille notre modèle de développement, en commençant par l'environnement puis en montrant le fonctionnement des cellules qui agiront à l'intérieur.

3

Le modèle *Cell2Organ*

Dans le but de générer des organes qui, une fois assemblés, pourront animer des créatures entières, nous avons imaginé un modèle de développement basé sur une forte simplification du modèle naturel [Cussat-Blanc et al., 2008b]. Ce chapitre décrit les différents éléments de ce modèle, le fonctionnement de l'environnement, de la chimie artificielle et les interactions dont sont capables les cellules. Nous détaillerons aussi le patrimoine génétique d'une cellule et son cycle de vie. Enfin, nous verrons notre implantation du modèle, basée sur une architecture hautement parallèle.

3.1 Description du modèle

3.1.1 Objectif du modèle

L'objectif du modèle est d'être capable de générer des créatures possédant une morphologie et des fonctions décrites par l'utilisateur. La morphologie sera souvent guidée par la fonction et laissée libre à l'évolution génétique. De plus, comme nous l'avons vu dans l'état de l'art, il n'existe que très peu de modèles intégrant un métabolisme. Celui-ci est pourtant primordial dans le maintien de l'intégrité de tout l'organisme en permettant à chaque cellule de survivre dans son environnement. Nous voulons intégrer ce métabolisme aux créatures à cette fin. Cependant, cela va entraîner une augmentation de la complexité du modèle. En effet, les cellules devront choisir à chaque instant de la simulation entre produire de l'énergie via leur métabolisme et atteindre la fonction qui leur est demandée.

3.1.2 L'environnement et sa chimie artificielle

La topologie de notre environnement est une grille torique 2D. Ce choix permet une importante réduction de la complexité de la simulation et donc du temps de calcul tout en gardant suffisamment de liberté dans le modèle.

Notre modèle intègre une chimie artificielle très simplifiée. Comme nous l'avons vu dans le chapitre 2.3.4, Dittrich décrit formellement une chimie artificielle comme étant un triplet (S, R, A) où S est l'ensemble des molécules possibles, R est l'ensemble de règles de collisions représentant les interactions entre les molécules et A est un algorithme qui décrit la méthode d'application des règles dans l'environnement.

A notre environnement est associé un ensemble de molécules, que nous appellerons aussi *substrats*. Ils possèdent une vitesse de diffusion (dont nous verrons le fonctionnement par la suite). Cet ensemble de substrats est l'ensemble S de la chimie artificielle décrite par Dittrich. Ils peuvent interagir entre eux et cette interaction correspond à la définition formelle des réactions de l'ensemble R du triplet présenté ci-dessus. Elles sont souvent présentées par des formules de la forme suivante :

$$s_1 + s_2 + ... + s_n \rightarrow s'_1 + s'_2 + ... + s'_m$$

Dans cette formule, s_i et s'_j ($i \in 1..n$, $j \in 1..m$) sont des substrats de l'ensemble S. Ils correspondent aux réactifs (pour les s_i dans la partie gauche de l'équation) et aux produits (les s'_j à droite de l'équation) résultant de l'interaction des réactifs. Une réaction ne peut être déclenchée que si tous les réactifs sont présents en même temps au même endroit.

Comme dans la nature, la quantité de chaque molécule qui sera consommée doit être prise en compte. On obtient alors la formule pour une réaction :

$$a_1 s_1 + a_2 s_2 + ... + a_n s_n \rightarrow a'_1 s'_1 + a'_2 s'_2 + ... + a'_m s'_m$$

où les coefficients $a_i \in \mathbb{N}$ et $a'_j \in \mathbb{N}$ ($i \in 1..n$, $j \in 1..m$) correspondent à la quantité des molécules intervenants dans la réaction.

Dans notre modèle, seule la cellule est capable d'effectuer la transformation chimique. Cela permet de réduire la complexité en limitant la réaction au seul environnement intracellulaire. Cette propriété de notre chimie nous permet de répondre à la question de sa dynamique, c'est à dire l'algorithme de réaction A. En effet, la cellule ne pourra déclencher la réaction chimique si et seulement si tous les composés sont réunis à l'intérieur de sa membrane. Les réactions définissent la quantité d'énergie qu'elles produisent ou consomment. Cette énergie correspond à l'énergie vitale que récupérera ou dépensera la

cellule lors du déclenchement de la réaction chimique.

On obtient finalement la formulation suivante d'une réaction chimique :

$$a_1 s_1 + a_2 s_2 + ... + a_n s_n \rightarrow a'_1 s'_1 + a'_2 s'_2 + ... + a'_m s'_m \quad (\delta_{energy})$$

avec $\delta_{energy} \in \mathbb{R}$, la quantité d'énergie produite (si sa valeur est positive) ou consommée (si sa valeur est négative) par la réaction.

Par exemple, la réaction $2A + B \rightarrow C$ $(+50)$ produira une unité d'un substrat C à l'aide de deux unités de substrat A et d'une de B. Cette transformation émettra 50 unités d'énergie.

3.1.3 La diffusion des substrats

Les substrats de l'environnement peuvent se diffuser sur la grille dans le but d'équilibrer les quantités de substrats dans l'environnement en minimisant les variations des quantités de substrat entre deux points voisins de la grille. Cette diffusion se déroule en deux temps, comme schématisé sur la figure 3.1 :

1. premièrement, le substrat se diffuse vers les quatre points cardinaux ;
2. ensuite, si la quantité de substrat est suffisante, le substrat se diffuse sur les diagonales.

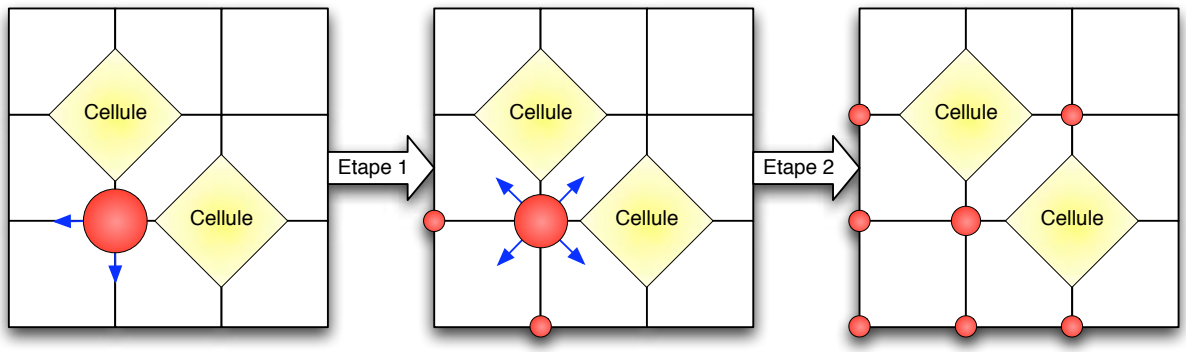


FIGURE 3.1 – *Exemple de diffusion d'un substrat dans l'environnement.*

Si la quantité de substrat n'est pas suffisante pour se diffuser de manière égale, les dernières diffusions de l'étape de diffusion sont choisies aléatoirement. De plus, si un point voisin du point de diffusion contient plus de substrats, la diffusion ne se fera pas dans cette direction étant donné que le but de celle-ci est de minimiser les variations de substrat entre les points voisins de l'environnement. Enfin, aucune diffusion ne se fait à travers une cellule, que ce soit de l'environnement vers l'intérieur de la cellule ou le

contraire. Seule une action déclenchée par la cellule peut permettre un échange de substrat avec l'environnement.

Dans le paragraphe suivant, nous allons étudier les cellules de notre modèle de développement. Il décrit plus particulièrement le fonctionnement interne de nos cellules artificielles.

3.1.4 Les cellules

Les cellules ont différentes capacités et peuvent interagir avec l'environnement et ainsi le modifier. Elles doivent atteindre collectivement un but global décrit par l'utilisateur : récolter un substrat, déplacer un substrat d'un point à un autre, créer une certaine forme ou tout simplement survivre le plus longtemps possible.

Les cellules évoluent dans l'environnement et plus précisément sur la grille de diffusion de celui-ci. Chaque cellule possède un certain nombre de capteurs (voir 3.1.4) et a une liste de capacités (autrement dit, une liste d'actions, voir 3.1.4). Un système de sélection permet à la cellule de sélectionner la meilleure action à effectuer à chaque instant de la simulation (décrit dans la section 3.1.5). Enfin, nos cellules possèdent un réseau d'optimisation d'action basé sur les réseaux de régulation de gènes. Il leur permet de se spécialiser au cours de la division cellulaire (détaillée dans la section 3.1.6). La figure 3.2 présente la structure générale de nos cellules artificielles dans un environnement.

Les capteurs

Chaque cellule contient différents capteurs de densité de substrat. Ceux-ci sont positionnés aux extrémités des cellules. Ils permettent à la cellule de mesurer la quantité de substrats disponible dans son voisinage de Von Neumann. Pour chaque substrat de l'environnement, un capteur spécialisé existe. Seul le capteur correspondant au substrat peut calculer la densité du substrat. La liste des capteurs disponibles ainsi que leur position dans la cellule est décrite dans le génome de la cellule. Par exemple, dans la figure 3.2, la cellule possède dans le coin gauche un capteur pour le substrat B et un pour le substrat D. Les résultats des mesures des densités de substrats sont :

- deux unités de substrat B car deux unités de substrat B sont présents à gauche de la cellule,
- une unité de substrat D.

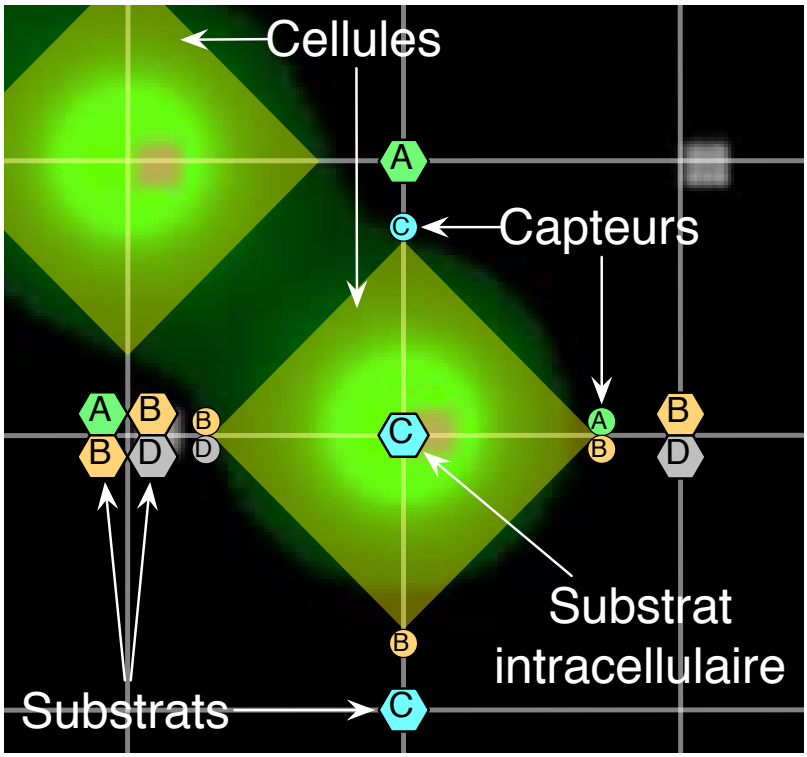


FIGURE 3.2 – *Schéma de la cellule artificielle dans son environnement. Elle contient des substrats (hexagones) et leurs capteurs correspondants (cercles).*

Les actions

Pour interagir avec l'environnement, les cellules peuvent déclencher un certain nombre d'actions :

- La *transformation de substrats* permet à la cellule de déclencher une réaction chimique présente dans l'environnement. Pour qu'elle puisse avoir lieu, tous les substrats de la partie gauche de la réaction doivent être présents dans la cellule, c'est à dire que ces substrats doivent se trouver sur la même intersection que la cellule dans la grille de diffusion. En résultat de cette réaction, le niveau d'énergie vitale de la cellule augmente ou diminue (en fonction des propriétés de la réaction), les substrats de la partie gauche de la réaction sont consommés et ceux de la partie droite sont produits.
- La cellule peut *absorber* ou *rejeter* des substrats dans l'environnement. Ces actions permettent à la cellule de déplacer des substrats d'un point à un autre. Elles sont importantes, et particulièrement la première, pour déclencher une transformation de substrats.
- La *division* permet à la cellule de créer une nouvelle cellule fille. Cette action est déclenchable si les conditions suivantes sont respectées :

  - La cellule mère doit avoir au moins un voisin libre dans l'environnement pour y positionner la nouvelle cellule.
  - La cellule mère doit avoir suffisamment d'énergie vitale pour effectuer l'action de division. Le niveau d'énergie nécessaire est défini lors de la définition de l'environnement.
  - Une liste de préconditions peut être ajoutée lors de la modélisation de l'environnement. Par exemple, une certaine quantité de substrats peut être nécessaire pour produire la nouvelle cellule.
- La *survie* permet à la cellule d'attendre un signal de l'environnement en ne faisant que maintenir son métabolisme.
- L'*apoptose* permet à la cellule de s'autodétruire. Cette action peut être utile pour libérer de l'espace dans l'environnement pour y placer une cellule plus spécialisée par exemple.

La précédente liste d'actions est une liste ouverte. L'implantation de notre modèle doit permettre l'ajout facile d'une ou plusieurs actions. Tout comme pour les capteurs, toutes les actions ne sont pas disponibles pour une cellule : le génome donne la liste de ses actions possibles.

3.1.5 Sélection de l'action

Les cellules contiennent également un système de sélection d'action. Ce système est basé sur les règles des systèmes de classeurs [Wilson et al., 1998]. Il utilise les données fournies par les capteurs de la cellule pour sélectionner la meilleure action à déclencher. La figure 3.3 schématise le fonctionnement du système de sélection. Le système de sélection est en fait une base de règles dans laquelle chaque règle est composée de trois parties :

- La *précondition* décrit l'état de la cellule pour lequel une action doit être déclenchée. Elle est composée d'une liste d'intervalles de valeurs qui correspondent aux densités des substrats du voisinage de la cellule.
- L'*action* donne l'action à déclencher si la précondition est respectée.
- La *priorité* permet un classement des règles. Dans le cas où plusieurs règles seraient déclenchables, la priorité permet de déterminer quelle action doit être déclenchée. Plus la priorité est élevée et plus la règle a de chance d'être sélectionnée.

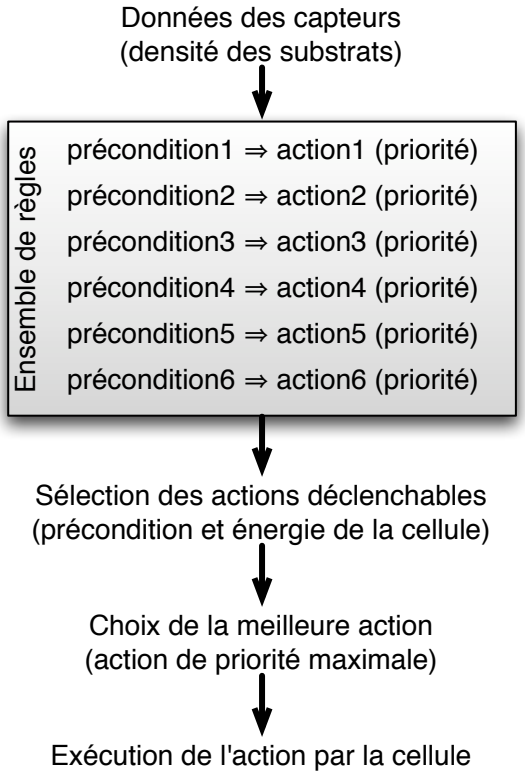


FIGURE 3.3 – *Fonctionnement du système de sélection d'action.*

Un exemple de système de sélection de règle pourrait être :

$$\begin{aligned}(CapteurA = 1)\ and\ (3 < CapteurB < 7)& \\ and\ (CapteurC = 0) \quad &\rightarrow \quad (Action1)\ (23) \\ (CapteurC = 3) \quad &\rightarrow \quad (Action2)\ (13) \\ \emptyset \quad &\rightarrow \quad (Action3)\ (17)\end{aligned}$$

Dans cet exemple, l'action 1 sera déclenchée si et seulement si la valeur du capteur associée au substrat A est égale à une unité, la valeur du capteur B est comprise entre 3 et 7 et la valeur du capteur C est nulle. L'action 3 pourra toujours être déclenchée, la liste de ses préconditions étant vide. La priorité des règles ordonne les actions dans l'ordre $Action1 > Action3 > Action2$ si plus d'une action est déclenchable à un instant t de la simulation.

Dans un premier temps, pour simplifier ce système de sélection et pour réduire la complexité de la simulation, nous n'utiliserons que des booléens pour détecter la présence ou l'absence de substrats dans l'environnement.

3.1.6 Optimisation des actions

La division cellulaire est une action qui peut être déclenchée par la cellule si les préconditions de l'action sont remplies (en plus des préconditions du système de sélection d'actions). La nouvelle cellule créée après la division cellulaire est totalement indépendante et interagit elle aussi avec l'environnement. Pendant la division, la nouvelle cellule se spécialise pour optimiser un groupe d'actions au détriment des autres.

Dans la nature, cette spécialisation est principalement due au réseau de régulation de gènes (GRN) de la cellule. Il utilise les récepteurs protéiques présents sur la membrane de la cellule pour activer ou inhiber la transcription de certaines parties du génome de la cellule. Cette transcription produira une protéine qui déterminera les fonctions de la cellule fille. Le réseau de régulation est lui même codé dans le génome de la cellule. Plus de détails sur le fonctionnement de ce réseau de régulation sont donnés dans le chapitre 2.

Dans notre modèle, nous avons imaginé un mécanisme qui joue le rôle du GRN. Chaque action possède un coefficient d'efficacité qui correspond au niveau d'optimisation de l'action. En d'autres termes, il décrit la capacité de la cellule à effectuer cette action en minimisant son coût : plus ce coefficient est élevé et moins l'action consomme d'énergie. Si ce coefficient est nul l'action n'est plus disponible dans la cellule fille. La somme des coefficients d'efficacité reste constante lors de la spécialisation. En d'autres termes, si une action est optimisée en augmentant son coefficient d'efficacité, un ou plusieurs autres coefficients devront être réduits. On obtient ainsi une spécialisation vers certaines actions au détriment de certaines autres.

La cellule se spécialise donc durant la division cellulaire en faisant varier les coefficients d'efficacité. Un réseau de coefficients permet de donner les règles de transfert. Il est construit de la façon suivante :

- les noeuds du réseau représentent les actions avec leur coefficient d'efficacité,
- les arcs du réseau sont pondérés. Le poids des arcs (un nombre réel compris entre zéro et un) représente la quantité du coefficient d'efficacité qui sera transférée d'une action à l'autre pendant la division.

La figure 3.4 représente un exemple de notre réseau de régulation. $(A, 35\%)$, $(B, 25\%)$, $(C, 17\%)$, $(D, 23\%)$ sont des actions de la cellule avec leur coefficient d'efficacité associé. Les arcs entre deux actions représentent les quantités de coefficients d'efficacité qui sont transférées lors de la division. Par exemple, l'arc pondéré entre A et B signifie que 30% du coefficient d'efficacité de l'action A sera transféré sur l'action B. Ainsi, la valeur du coefficient d'efficacité associé à l'action A devient alors $35 - 35 * 30\% = 24.5$ et pour l'action B $25 + 35 * 30\% + 23 * 13\% - 25 * 5\% = 37.24$. Pour cette dernière action, 25 correspond au

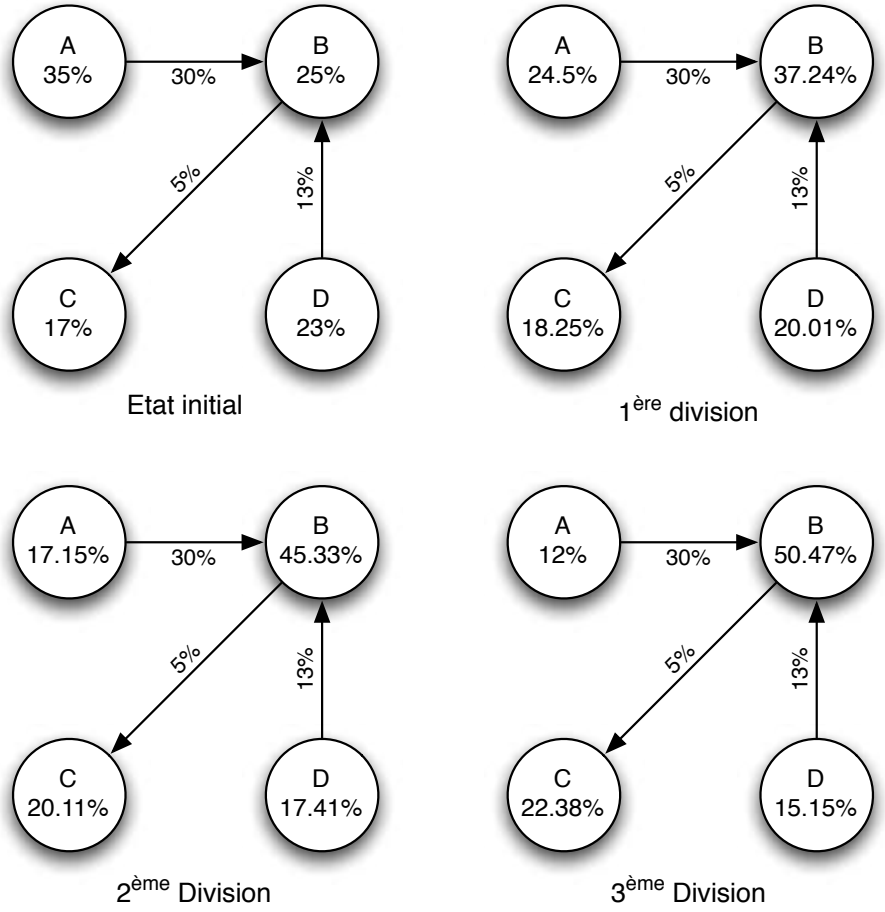


FIGURE 3.4 – *Modélisation d'un exemple du réseau de régulation de gènes. A, B, C et D sont 4 actions avec leur coefficient d'efficacité. Les coefficients de transfert sont donnés par les arcs pondérés.*

coefficient initial de l'action B, $+35 * 30\%$ correspond au transfert de 30% du coefficient de l'action A vers l'action B, $+23 * 13\%$ correspond au transfert du coefficient de l'action D vers l'action B et $-25 * 5\%$ correspond quant à lui au transfert de 5% du coefficient de l'action B vers celui de l'action C.

Après quatre divisions, on peut remarquer que les actions B et C ont été optimisées au détriment des actions A et D. A partir de cet exemple simple, nous pouvons dire que la cellule s'est spécialisée pour effectuer en particulier les actions B et C.

3.2 Le génome de la cellule

Chaque organisme possède un génotype décrivant les différents paramètres des cellules afin de trouver l'individu le mieux adapté à un problème spécifique. Un algorithme géné-

tique, dont les détails seront donnés dans le chapitre 4, fera évoluer ce génome. Le génome des organismes est composé de trois chromosomes comme présenté par le schéma 3.5 : la liste des actions possibles, le système de sélection d'action et le réseau d'optimisation.

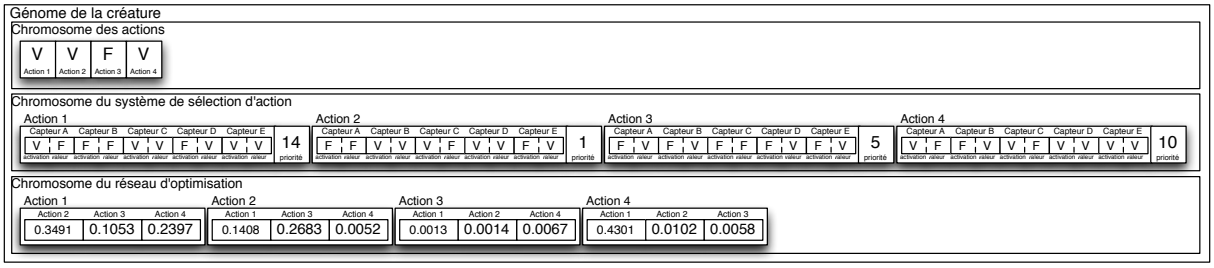


FIGURE 3.5 – *Le génome d'une créature artificielle est composé de trois chromosomes : un premier codant la liste des actions possibles, un second pour le système de sélection d'action et le dernier code le réseau d'optimisation.*

3.2.1 Liste des actions

La liste des actions possibles pour la cellule est en fait un sous-ensemble de la liste des actions possibles décrite dans l'environnement. Cette liste permet à la cellule d'activer ou d'inhiber certaines actions. Ce chromosome est codé comme une chaîne de booléens.

3.2.2 Sélecteur d'action

Le système de sélection d'action contient la liste des règles pour appliquer une action. Le chromosome est codé à l'aide d'une paire *(liste de gènes de capteurs, priorité)* pour chaque action possible dans laquelle :

- la *liste de gènes de capteurs* est une liste de couples de booléens pour chaque capteur de la cellule. Le premier booléen exprime l'utilisation ou la non-utilisation du premier capteur de la cellule pour la règle courante. Le second booléen exprime la fonction du capteur : il peut détecter la présence ou l'absence de substrat dans son environnement.
- la *priorité* représente la priorité de la règle courante dans l'ensemble des règles. Elle est codée à l'aide d'un entier.

La figure 3.6 est un exemple de chromosome de sélecteur d'action. L'environnement contient ici quatre actions $Action1$, $Action2$, $Action3$ et $Action4$ et cinq capteurs $CapteurA$, $CapteurB$, $CapteurC$, $CapteurD$ et $CapteurE$. Ce chromosome produirait le système de sélection d'actions suivant :

$$
\begin{aligned}
(CapteurA = faux)\ et\ (CapteurC = vrai) & \\
et\ (CapteurE = vrai) \quad &\rightarrow \quad (Action1)\ (14) \\
(CapteurB = vrai)\ et\ (CapteurC = faux) & \\
et\ (CapteurD = vrai) \quad &\rightarrow \quad (Action2)\ (1) \\
\emptyset \quad &\rightarrow \quad (Action3)\ (5) \\
(CapteurA = faux)\ et\ (CapteurC = faux) & \\
et\ (CapteurD = vrai)\ et\ (CapteurE = vrai) \quad &\rightarrow \quad (Action4)\ (10)
\end{aligned}
$$

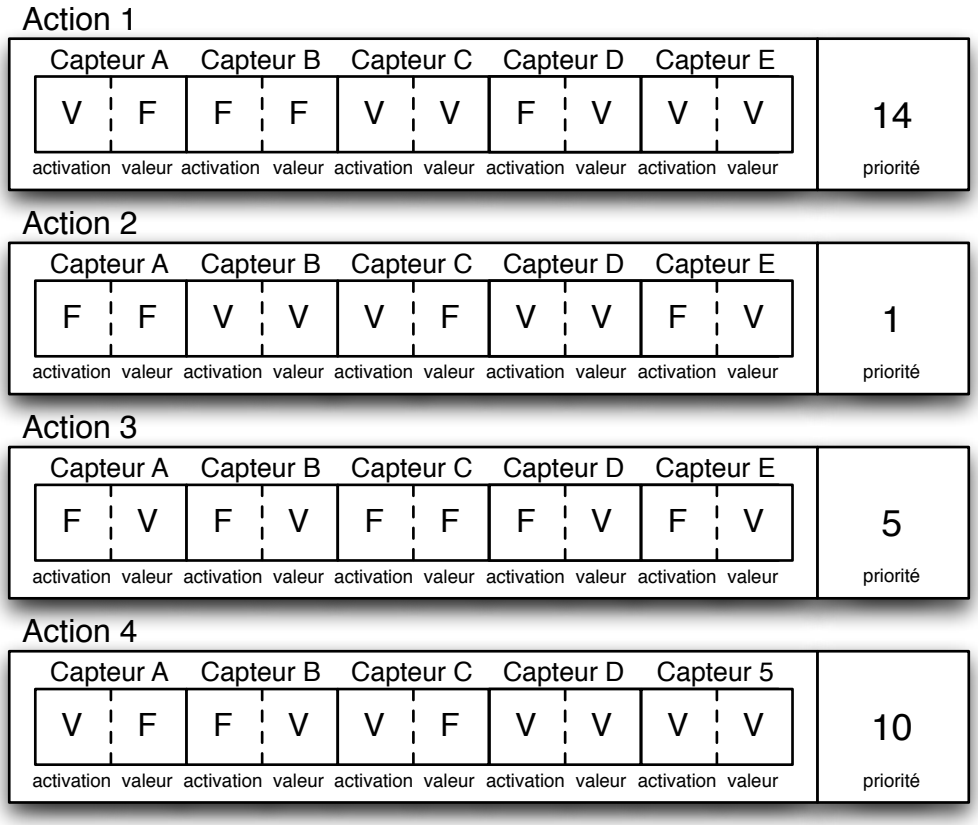


FIGURE 3.6 – *Exemple de chromosome de système de sélection d'actions.*

3.2.3 Réseau d'optimisation

Le réseau d'optimisation génétique permettant la spécialisation cellulaire est codé à l'aide d'une liste de réels appartenant à l'ensemble [0,1] : chaque action possède une liste de coefficients. Chaque coefficient donne la pondération de l'arc entre deux actions

du réseau de régulation. Le réseau de régulation est initialisé avec la même valeur pour toutes les actions (on suppose que la première cellule est capable d'effectuer toutes les actions équitablement).

La figure 3.7 représente un exemple de chromosome de réseau d'optimisation. L'environnement contient quatre actions *Action*1, *Action*2, *Action*3 et *Action*4. Le chromosome produit le réseau d'optimisation décrit par la figure 3.8. La simplification indiquée sur le schéma consiste seulement à effectuer la soustraction des poids des deux arcs entre deux actions dans le but de n'en avoir plus qu'un.

FIGURE 3.7 – *Exemple de codage du chromosome d'un réseau d'optimisation.*

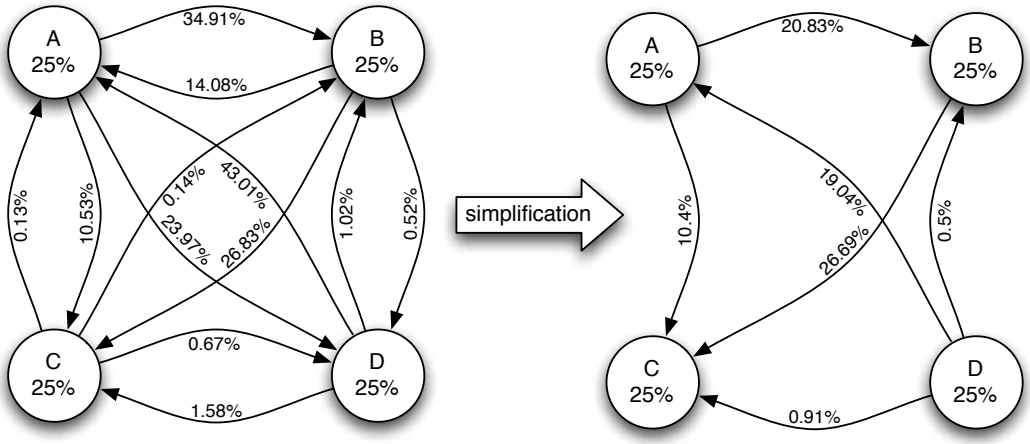


FIGURE 3.8 – *Réseau d'optimisation produit par le chromosome de la figure 3.7.*

Les détails du fonctionnement des organismes produits par ce modèle seront donnés au travers des différentes expérimentations du chapitre 5.

3.3 Principes d'implantation du modèle

Nous avons implémenté ce modèle en utilisant le langage de programmation Java et une architecture multi-threadée :

- un thread pour chaque cellule : les cellules communiquent en utilisant l'environnement et les échanges de substrats,
- un thread pour le système de diffusion.

Nous avons fait ce choix d'une architecture hautement parallèle au vu du fort développement des ordinateurs massivement parallèles tels que les machines multi-processeurs et de plus en plus connectés en grille. Cette parallélisation permet d'augmenter le nombre de tâches exécutées simultanément.

Il est important de noter qu'aucune synchronisation entre les threads n'a été mise en place. Les cellules sont complètement indépendantes et fonctionnent de manière totalement asynchrone. Seul des verrous ont été posés au niveau des différentes ressources de l'environnement afin de garantir la cohérence des données.

Cette architecture multi-threadée a aussi nécessité la mise en place d'un mécanisme de simulation temporelle. Afin de pouvoir accélérer ou ralentir une simulation, les temps indiqués dans la définition des actions dans l'environnement ne sont pas donnés afin d'avoir une cohérence globale entre les actions. Un système d'accélération permet de multiplier ou de diviser ce temps de façon identique pour toutes les actions et un chronomètre calcule le temps de simulation en fonction des fluctuations de l'accélération temporelle. Ainsi, à titre d'information pour les temps d'actions que nous donnerons dans les prochains chapitres pour les différentes expérimentations, la simulation est accélérée entre 2 et 5 fois pour une visualisation correcte et entre 100 et 150 fois pour une simulation sans visualisation (lors de la recherche génétique en particulier). Nous parlerons par la suite de *temps simulé* pour le temps interne de la simulation (donc celui qui peut être accéléré ou ralenti) et de *temps réel* pour le temps défini par l'utilisateur pour les actions.

L'implantation a été faite pour supporter facilement les possibles extensions du modèle :

- l'ajout d'une nouvelle action est facilité grâce à l'utilisation d'actions *génériques*,
- une *interface* capteur permet d'ajouter facilement un nouveau type de capteur,
- les conditions du système de sélection de l'action sont elles aussi interfacées dans le même but,
- nous avons utilisé une méthode de programmation par "listener" pour pouvoir accéder facilement aux données sans interférer avec la simulation.

L'interface graphique est complètement découplée de la simulation. Elle utilise les "listener" de la simulation pour accéder aux données dont elle a besoin. Elle n'interfère donc pas avec la simulation et il est possible de la changer facilement.

Un diagramme de classes de l'implantation de notre modèle est disponible en Annexe B.

4

Parallélisation et paramétrage de l'algorithme génétique

Les créatures que nous allons développer avec ce modèle sont d'une grande complexité. En effet, en plus d'avoir à développer une fonctionnalité décrite par l'utilisateur, le désir d'ajouter un métabolisme utilisant les constituants de l'environnement et qui devra être découvert par la créature elle-même rend la recherche d'un bon génome pour obtenir une créature intéressante difficile. Pour cela, il va nous être nécessaire d'optimiser l'algorithme génétique qui permettra de trouver les meilleures solutions à nos problèmes. Ce chapitre présente trois techniques que nous avons implantées et testées afin de gagner du temps pour la convergence de l'algorithme. Tout d'abord, nous allons étudier une méthode de parallélisation sur une grille de calcul en utilisant le middleware ProActive. Nous verrons ensuite la mise au point d'une technique d'adaptation des paramètres (taux de mutation et de croisement) de l'algorithme génétique. Enfin, nous terminerons par l'implantation d'une méthode d'aide à la convergence basée sur la décomposition de la fonction d'évaluation en sous-objectifs.

4.1 Parallélisation des algorithmes génétiques

La recherche d'un génome pour des créatures aptes à satisfaire des contraintes morphologiques et fonctionnelles impliquées par le problème à résoudre va consommer un temps de calcul important. Du fait du temps nécessaire pour une seule simulation (de quelques secondes en début de recherche jusqu'à environ 120 secondes lorsque la créature commence à répondre à sa fonction), le calcul d'une génération composée d'en moyenne 750 créatures peut prendre jusqu'à $750 * 120 = 90000\ sec = 1500\ min = 25\ heures$ par génération. Sachant qu'il faut 50 à 75 générations à l'algorithme génétique pour conver-

ger, on arrive rapidement à plusieurs semaines de calcul pour trouver le génome recherché. Pour réduire le temps de calcul global, nous avons décidé de paralléliser l'algorithme génétique. Les calculs sont déployés sur la grille de calcul expérimentale française Grid5000 [Cussat-Blanc et al., 2008c].

La suite de la section est organisée de la manière suivante. Tout d'abord, nous allons passer en revue les techniques de parallélisation des algorithmes génétiques existantes. La deuxième sous-partie présente le middleware d'abstraction de grille développé par l'équipe OASIS à l'INRIA Sophia-Antipolis : *ProActive*[16]. Puis, nous testerons la parallélisation obtenue à l'aide de cette méthode à travers deux expérimentations simples. Enfin, nous essaierons également d'évaluer le gain possible sur notre modèle.

4.1.1 Les méthodes de parallélisation déjà existantes

Les algorithmes génétiques, par leurs structures, peuvent être aisément parallélisés [Cantù-Paz, 1997, Herrera et al., 2005, Branke et al., 2004]. Pour ce faire, trois méthodes principales existent. Nos travaux utilisent des grilles de calcul, plus extensibles et moins onéreuses. Dans cette partie, nous allons voir les algorithmes de parallélisation existants et la façon dont nous les avons utilisés pour notre problème de vie artificielle.

Un rapide rappel sur les algorithmes génétiques

Cette section est une rapide introduction aux algorithmes génétiques. Elle montre le fonctionnement de cette méthode d'optimisation inspirée de l'évolution naturelle et les différents points qui peuvent être parallélisés. Un algorithme génétique est un algorithme d'optimisation stochastique qui s'inspire de l'évolution darwinienne pour parcourir l'espace de recherche [Holland, 1975]. Il suit le schéma présenté dans la figure 4.1.

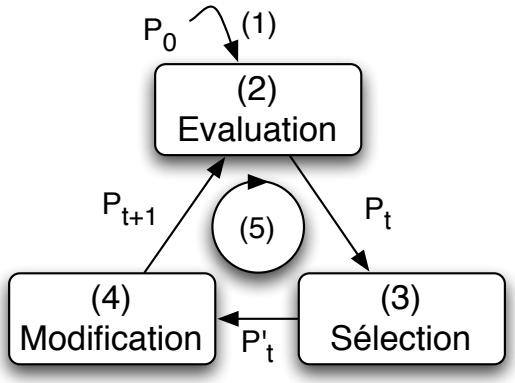


FIGURE 4.1 – *Schéma du fonctionnement général d'un algorithme génétique.*

16. Sources et documentations disponible sur le site http://proactive.inria.fr/

L'algorithme génétique est initialisé avec une population P_0 constituée, généralement aléatoirement, de génomes de l'espace de recherche (1). En utilisant une fonction d'évaluation, chaque solution est évaluée (2). Si une solution suffisamment intéressante est trouvée ou que le temps attribué à la recherche est écoulé (critère d'arrêt), l'algorithme se termine ; sinon, un sous-ensemble de la population P'_t est sélectionné à l'aide des notes précédemment attribuées à la population P_t (3). Différentes méthodes de sélection existent. En voici plusieurs parmi les plus connues :

- la sélection par *rang* : on ne choisit que les meilleurs individus,
- la sélection par *roulette* : la probabilité de sélection d'un individu est proportionnelle à sa note,
- la sélection par *tournoi* : les individus sont réunis en groupes de n génomes et le meilleur individu (au vu de son évaluation) a la plus grande chance d'être sélectionné,
- la sélection *steady-state* : on choisit les individus aléatoirement dans la population,

On peut aussi introduire de l'*élitisme* [De Jong, 1975] afin de s'assurer que la note maximale augmente. Pour compléter la population précédemment sélectionnée, les génomes sont combinés en utilisant principalement les deux opérateurs suivants :

- le *croisement* ou *crossover* permet de croiser deux génomes en un ou plusieurs points,
- la *mutation* permet de changer aléatoirement une valeur du génome.

Une nouvelle population P_{t+1} est alors créée et peut être à nouveau évaluée (5).

Bien que le choix des paramètres de l'algorithme génétique (taille de la population, taux de mutation et de croisement, qualité de la fonction d'évaluation, etc.) ait une importance clé dans la réduction du temps de calcul, une solution complémentaire très efficace est de paralléliser l'algorithme. Deux points peuvent être parallélisés facilement : la sélection des génomes et leur modification. Alors que la parallélisation des opérations de modification des génomes est rarement profitable car elle demande souvent peu de ressources de calcul,

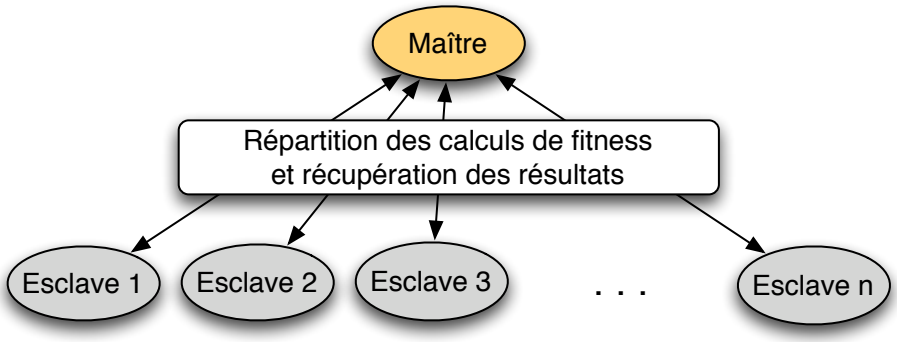


FIGURE 4.2 – *Un algorithme qui calcule en parallèle l'évaluation des individus d'une population répartit la charge de calcul sur différentes unités de calcul.*

la parallélisation de l'évaluation donne de meilleurs résultats. Dans les sections suivantes, nous présentons différentes approches de ce type de parallélisation.

Algorithmiques de parallélisation : Algorithme génétique Maître/Esclaves

En utilisant une architecture Maître/Esclaves, le calcul de l'évaluation peut être réparti sur un ensemble de processeurs (figure 4.2). Dans une approche mono-processeur classique, l'algorithme aurait modifié la population (grâce aux opérateurs de mutation et de croisement) en utilisant les notes données par une évaluation séquentielle des génomes. Dans cette approche, la modification des génomes est effectuée par l'unité maître de l'architecture en utilisant les notes calculées par les esclaves déployés sur un ensemble d'unités de calcul. Etant donné que l'évaluation des génomes sont complètement indépendantes, le résultat donné par cet algorithme est exactement le même qu'avec un algorithme génétique classique.

Algorithmes de parallélisation : Méthode par îlots

Dans cette méthode, la population initiale de l'algorithme génétique est divisée en plusieurs sous-populations. Des algorithmes génétiques classiques sont ensuite appliqués sur chaque sous-population. Les différentes sous-populations sont indépendantes les unes des autres et sont traitées sur des unités de calculs séparées. Pour aider le mélange des sous-populations entre elles, des migrations d'individus peuvent se produire. Ceci permet d'éviter le confinement des sous-populations dans des optima locaux. Le taux de migration des individus est alors un nouveau paramètre de l'algorithme génétique. La figure 4.3

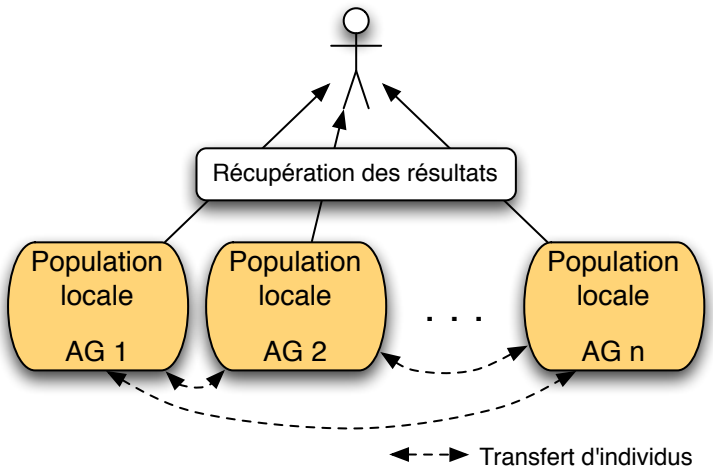


FIGURE 4.3 – *Un algorithme génétique par îlots crée des sous-populations et applique un algorithme génétique classique sur chacune d'entre elles.*

schématise le fonctionnement d'un algorithme génétique par îlots. Différentes architectures de migration sont possibles, telles que la grille, l'anneau, etc [Sekaj, 2004]

Cette méthode réduit grandement la communication entre les différentes unités de calculs du fait du peu d'échange entre les noeuds. Cependant, il est impossible de garantir les propriétés de l'algorithme génétique classique (la vitesse de convergence en particulier) car la division en sous-populations augmente la probabilité de confinement dans un optimum local. Beaucoup d'expériences montrent que la qualité des solutions obtenues par un algorithme par îlots est très proche d'un algorithme génétique classique, mais l'algorithme génétique par îlots a souvent besoin de plus de générations pour converger [Bianchini and Brown, 1993, Tanese, 1989, Neuhaus, 1991, Kommu and Pomeranz, 1992].

Algorithmes de parallélisation : Algorithme génétique hiérarchique

En combinant les deux méthodes précédentes, il est possible d'obtenir un nouvel algorithme parallèle. La population initiale est divisée en sous-populations comme dans la méthode par îlots, et chaque sous-population est évaluée en parallélisant le calcul de l'évaluation, comme dans la première méthode [Sekaj, 2004, Lim et al., 2007]. On obtient alors une architecture hiérarchique dans laquelle chaque sous-population possède un maître avec

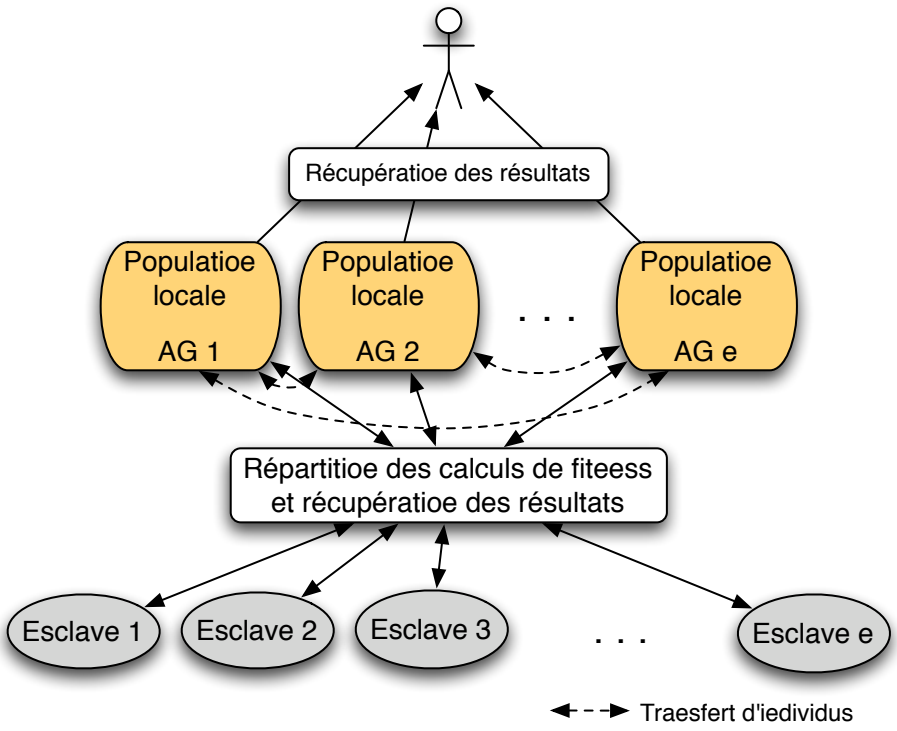


FIGURE 4.4 – *Un algorithme génétique hiérarchique crée initialement des sous-populations et applique un algorithme génétique maître/esclaves sur chacune d'entre elles.*

différents esclaves qui évaluent les individus. La figure 4.4 montre l'architecture de l'algorithme génétique hiérarchique.

Les résultats de cette méthode sont exactement les mêmes que ceux obtenus par un algorithme génétique par îlots. En effet, chaque population est parallélisée grâce à la méthode Maître/Esclaves qui donne exactement les mêmes résultats qu'un algorithme génétique classique, le résultat général restant identique à celui obtenu sans la parallélisation des sous-populations.

Déploiement d'un algorithme génétique sur une grille de calcul

Différentes implantations de ces trois algorithmes de parallélisation existent sur des supercalculateurs. Pour nos travaux, nous avons décidé d'utiliser une grille de calcul pour des raisons de disponibilité dans notre laboratoire de recherche et pour des raisons de capacités d'évolutions (une grille peut être hétérogène et permet donc une augmentation de sa puissance plus aisée). La différence la plus importante entre une grille et un supercalculateur est que les unités de calculs sont séparées par un réseau haute performance dans le premier cas et par un bus de données haute performance dans le second (figure 4.5). Du fait de l'utilisation du réseau, le délai de transfert des données entre les unités de calcul est important à prendre en compte.

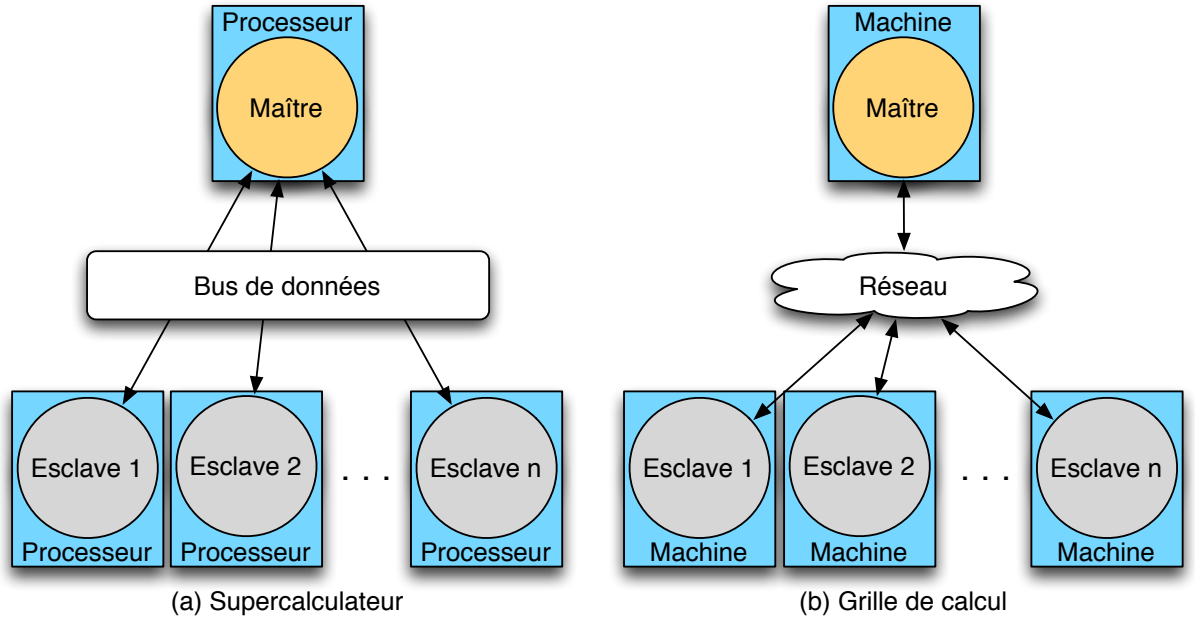


FIGURE 4.5 – *Dans l'exemple d'un algorithme génétique Maître/Esclaves, un supercalculateur (a) communique en utilisant un bus de données alors qu'une grille de calcul (b) utilise un réseau hautes performances.*

Le but premier de cette parallélisation est de réduire le temps de recherche du génome de nos créatures artificielles. Nous avons décidé d'appliquer un algorithme génétique Maître/Esclaves pour paralléliser notre algorithme génétique. Cette méthode est bien adaptée aux créatures artificielles car la taille de leur génome est souvent petite (quelques kilos octets) et le temps de calcul de leur évaluation (dans le simulateur de leur environnement) important (plusieurs secondes voir minutes). Du fait de la petite taille des génomes, leur transfert sur le réseau imposée par l'algorithme Maître/Esclaves déployé sur une grille de calcul n'augmentera pas outre mesure le temps de calcul global. De plus, étant donné qu'un tel algorithme préserve les propriétés d'un algorithme génétique classique, le nombre de générations nécessaire pour trouver une bonne créature et la qualité de la créature obtenue sera la même, avec ou sans la parallélisation. Nous allons montrer par la suite que le temps de transfert d'un génome de petite taille est insignifiant face au gain apporté par la parallélisation du calcul d'un grand nombre d'évaluations nécessitant un temps de calcul important.

Dans ces travaux, nous disposions déjà d'une bibliothèque d'algorithmes génétiques, OO_GA (Object Oriented_Genetic Algorithm), développée dans notre équipe. Elle contient la plupart des algorithmes de sélection et de modification des génomes existants. La parallélisation s'est faite grâce au middleware ProActive [Caromel, 1993] qui permet une abstraction logicielle des mécanismes propres aux grilles de calcul. La section suivante décrit succinctement ProActive ainsi que l'interface Maître/Esclaves que nous avons utilisés.

4.1.2 ProActive : Un middleware d'abstraction de grille

Les grilles de calcul rassemblent en une seule organisation virtuelle de nombreuses ressources hétérogènes à travers différents sites géographiquement distribués. Ces ressources sont souvent organisées en clusters qui sont gérés par différentes organisations administratives (des laboratoires de recherche, des universités, etc.). Grâce à la grande quantité de ressources que les grilles de calcul fournissent, elles sont adaptées pour résoudre de très grands problèmes. Néanmoins, les grilles introduisent aussi de nouveaux challenges tels que le déploiement des calculs, la prise en compte de l'hétérogénéité de la grille, de la tolérance aux pannes, de la communication entre les unités de calcul et de son extensibilité. Pour masquer toutes ces difficultés, des middlewares comme ProActive permettent de s'abstraire de l'infrastructure de la grille. Ils permettent ainsi l'utilisation d'une grille comme étant une seule et même unité de calcul. [17]

17. Plus en détail, ProActive est un middleware de gestion de grille qui permet, entre autre, l'abstraction de l'infrastructure d'une grille de calcul en utilisant un "descripteur" [Baude et al., 2002], et un modèle "d'objets actifs" [Caromel, 1993]. Un objet actif est un objet distant accessible par différentes méthodes.

ProActive contient un ensemble d'outils qui permet de cacher ses concepts internes du logiciel. Il fournit ainsi un grand nombre d'interfaces de programmation de haut niveau bien connues dans le domaine de la parallélisation telles que l'interface Maître/Esclaves ou l'interface Branch & Bound.

Abstraction de l'infrastructure de la grille grâce à ProActive

Le système de déploiement ProActive extrait tous les détails de l'infrastructure de la grille du code source [Baude et al., 2002]. La figure 4.6 montre l'architecture générale de ProActive.

Le principe de base est d'éliminer complètement du code source les éléments suivants :

- le nom des machines,
- les protocoles de création,
- les protocoles de recherche et de découverte de services,
- les protocoles de communication.

Le but d'un outil de déploiement est de déployer des applications n'importe où sans avoir à modifier le code source du programme. Les ressources acquises à travers le processus de déploiement sont appelées des *noeuds*. Ce sont les conteneurs des objets actifs. Ils sont créés lors de l'exécution de ProActive sur les différentes ressources de l'infrastructure.

Le deuxième principe de base est la capacité de décrire une application (ou une partie de celle-ci) en terme d'activités conceptuelles abstraites. Dans le but d'abstraire la plate-forme d'exécution sous-jacente et de permettre un déploiement de codes sources indépendants, l'application à paralléliser doit fournir :

- une description abstraite des entités distribuées d'un programme ou d'un composant parallèle,
- une association externe de ces entités à des machines réelles en utilisant en fait des protocoles de création, de découverte et de recherche.

Pour répondre à ces principes, ProActive utilise un fichier XML de déploiement, le *descripteur*, qui décrit la configuration réelle de l'infrastructure de la grille. Le descripteur introduit pour ce faire la notion de *noeud virtuel* :

- Un noeud virtuel est défini grâce à un nom (une simple chaîne de caractères).
- Un noeud virtuel est utilisé dans le code source du programme à paralléliser.

En particulier, un *noeud virtuel*, après le déploiement de l'application, est *associé à* une

Chaque objet contient un thread de contrôle qui permet d'ordonnancer les tâches qui lui sont soumises. Les méthodes appelées sur l'objet actif sont asynchrones avec une synchronisation automatique en utilisant des "futurs" comme résultats des méthodes et une synchronisation par un mécanisme d'"attente par nécessité" [Caromel et al., 2006].

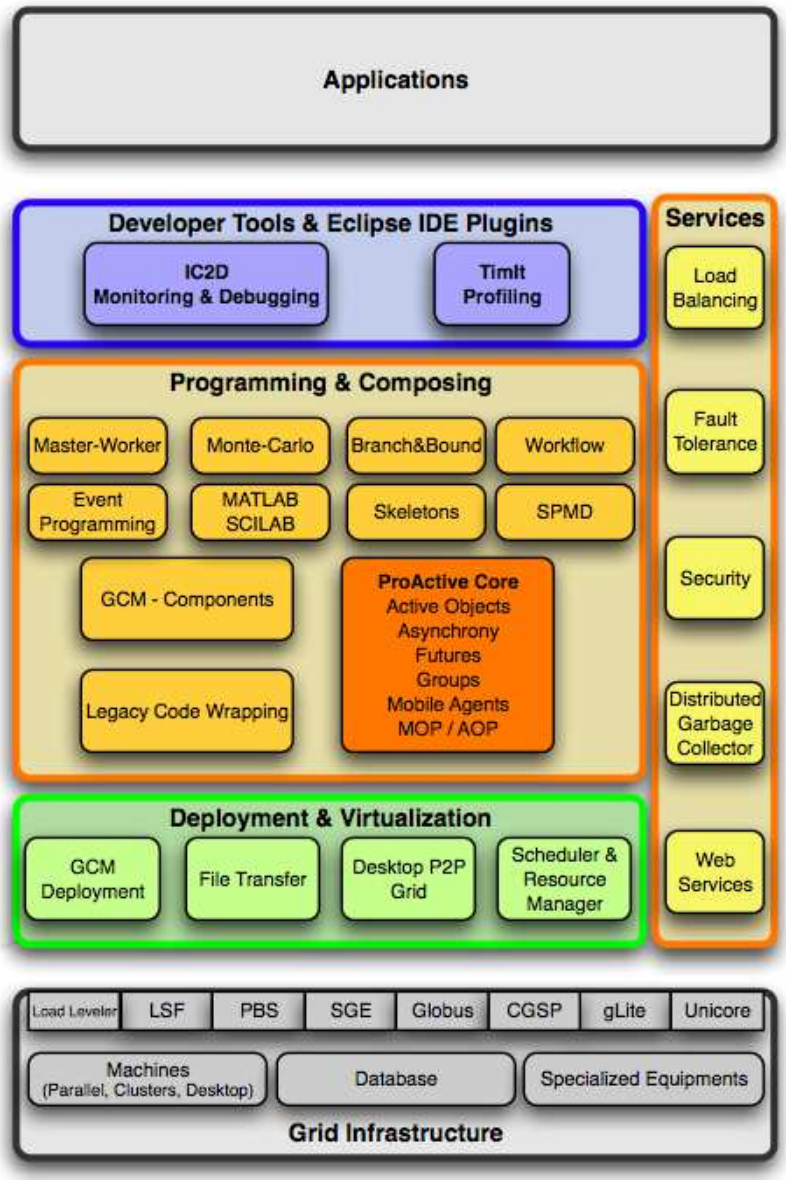


FIGURE 4.6 – *L'architecture de ProActive permet une abstraction complète de l'infrastructure de la grille en fournissant un ensemble de services.*

ou plusieurs *machines réelles* en suivant l'architecture décrite par le descripteur. Un noeud virtuel est un concept de la distribution d'un programme ou d'un composant, comme le noeud est un concept de déploiement qui accueille les objets actifs. Le descripteur décrit l'association entre les noeuds virtuels et les noeuds réels de la grille de calcul. Il n'y a pas d'association directe entre les noeuds virtuels et les objets actifs. Les objets actifs sont déployés par l'application à l'intérieur des noeuds apparentés à un noeud virtuel. Par définition, les opérations suivantes peuvent être configurées par le descripteur de déploiement :

- l'association des noeuds virtuels aux différents noeuds de la grille et aux machines virtuelles Java,
- les mécanismes (protocoles) de création ou d'acquisition des machines virtuelles Java tels que : local, ssh, rsh, rlogin, lsf, glite, etc.,
- les mécanismes (protocoles) de découverte et de recherche de machines virtuelles Java tels que : RMI, HTTP, RMI-ssh, Ibis et SOAP.

Dans le contexte du middleware ProActive, les noeuds désignent des ressources de l'infrastructure de la grille. Ils peuvent être créés ou acquis. L'outil de déploiement est chargé de fournir les noeuds associés aux noeuds virtuels de l'application. Les noeuds doivent être créés en utilisant une connexion distante et les protocoles de création. Les noeuds peuvent aussi être acquis grâce à des protocoles de recherche qui permettent notamment l'accès à l'infrastructure pair à pair de ProActive.

L'interface de programmation Maître/Esclaves

Le paradigme Maître/Esclaves est une approche fondamentale et communément utilisée dans les applications parallèles et distribuées. Dans une application Maître/Esclaves, un unique processus *maître* contrôle la distribution des calculs vers un ensemble de processus esclaves identiques. Ce paradigme a été utilisé précédemment pour une grande diversité d'applications parallèles [Pruyne and Livny, 1996, Everaars and Koren, 1997, Silva et al., 1999] et est parfaitement adapté au modèle de programmation pour une application visant à être distribuée sur une grille hétérogène [Berman, 1999].

L'approche de ProActive pour les applications Maître/Esclaves est de fournir une interface de programmation de haut niveau qui :

- permet aux utilisateurs de définir simplement des tâches qui seront exécutées par les esclaves,
- répartit intrinsèquement les tâches vers les différents esclaves,
- fournit un interface simple pour la collecte des résultats,
- s'occupe de la tolérance aux pannes en réaffectant les tâches si l'esclave tombe en panne,
- est implémentée en utilisant ProActive et le modèle d'objets actifs.

4.1.3 Utilisation de la méthode Maître/Esclaves pour paralléliser notre problème

Pour notre problème, nous avons décidé d'utiliser, dans un premier temps, l'algorithme génétique Maître/Esclaves avec l'interface de programmation Maître/Esclaves fournie avec

ProActive. Comme nous l'avons vu précédemment, les génomes étant de petite taille (quelques centaines de kilos octets), la quantité d'information à transférer ne sera pas trop importante. Ainsi, même si une architecture demande une grande quantité de communication à travers un réseau, le temps de calcul important et la faible taille des génomes nous permettent de penser que le temps de transfert des génomes ne réduira pas énormément les espérances de gain apportées par la parallélisation.

La partie suivante montre les gains apportés grâce à la méthode Maître/Esclaves que nous avons implémentée sur notre bibliothèque d'algorithme génétique, OO_GA.

4.1.4 Mesure de gain de la parallélisation

L'expérimentation suivante a pour but d'étudier le comportement de l'architecture Maître/Esclaves dans différentes situations :

1. quand le temps de calcul de l'évaluation est court et que la taille des génomes est grande,
2. quand le temps de calcul de l'évaluation et la taille des génomes sont moyens,
3. quand le temps de calcul de l'évaluation est grand et que la taille des génomes est petite.

Pour ce faire, nous allons utiliser un problème différent pour chacune des situations précédentes :

1. le problème OneMax pour la première situation,
2. le problème de détection d'une matrice non-inversible,
3. notre modèle de développement présenté dans le chapitre 3.

Les sous parties suivantes montrent les résultats obtenus pour ces différents problèmes. Ils nous permettront de conclure sur les capacités de l'approche que nous voulons utiliser dans notre problème d'embryogenèse artificielle.

Première situation : le problème OneMax

Pour commencer cette série d'expériences, nous allons résoudre le problème académique du OneMax. Le but est de créer une matrice binaire contenant un maximum de un. Pour cette expérimentation, nous prendrons une matrice de 30 par 30. Pour augmenter artificiellement la taille des génomes, nous avons choisi des caractères pour coder les valeurs binaires. La taille du génome de la matrice est alors de 900 octets. Le temps de calcul de l'évaluation pour un tel problème est très faible puisque la fonction d'évaluation

consiste simplement à compter le nombre de un dans la matrice. A l'opposé, la taille du génome est grande : le génome contient toute la matrice. De plus, le faible temps de calcul de l'évaluation augmente la sollicitation du réseau. En effet, le maître doit envoyer en continu des nouvelles données à ses esclaves et récupérer les résultats en parallèle.

Pour cette expérimentation, les paramètres de l'algorithme génétique sont les suivants :

- Taille de la population : 100
- Méthode de sélection : Tournoi à 7 avec élitisme
- Taux de croisement : 60%
- Taux de mutation : 5%

La figure 4.7 montre les résultats obtenus avec une grille de calcul composée de 1, 4, 8 et 16 processeurs. Chaque point d'une courbe représente le temps de calcul d'une génération.

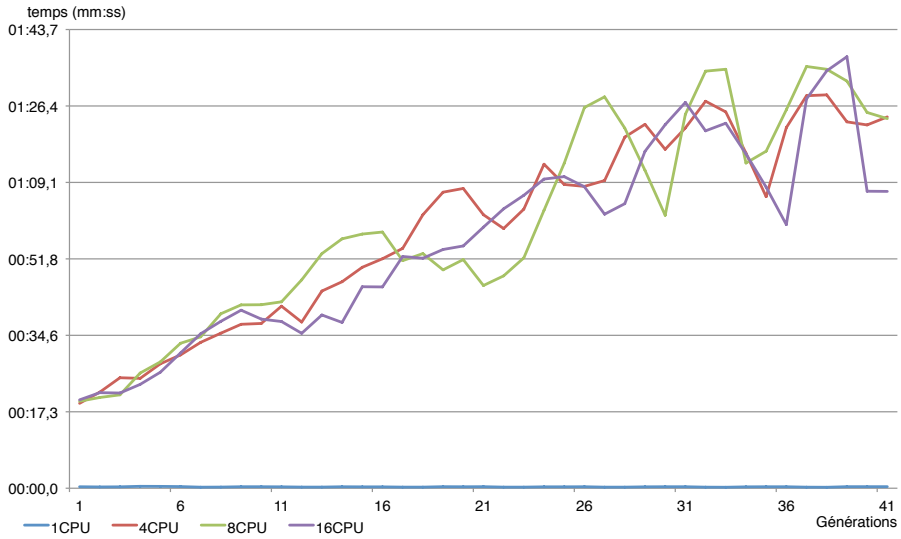


FIGURE 4.7 – *Temps de calcul nécessaire pour calculer 100 générations du problème One-Max en utilisant 1, 4, 8 et 16 processeurs. Du fait du court temps de calcul nécessaire pour calculer la fonction d'évaluation, le réseau ralentit le calcul général.*

La courbe la plus basse (en bleu, proche de l'axe des abscisses) correspond à une architecture mono-processeur. Les trois autres représentent la version parallèle de l'application en utilisant 4 processeurs (en rouge), 8 processeurs (en vert) et 16 processeurs (en violet). Du fait du peu de temps nécessaire au calcul de l'évaluation, la parallélisation sur une grille de calcul n'est pas rentable dans ce cas.

Les courbes montrent aussi les perturbations dues au réseau. En effet, le temps d'envoi des génomes via le réseau est largement supérieur au temps d'évaluation de ces génomes. Ceci est prouvé par l'équivalence des courbes pour 4, 8 et 16 processeurs. L'augmentation

du nombre de processeurs ne réduisant pas significativement le temps de calcul global, ce sont donc les temps de transfert qui le ralentissent.

Deuxième situation : Matrice non-inversible

Pour augmenter le temps de calcul de l'évaluation, nous avons essayé notre méthode de parallélisation sur un nouveau problème. Le but de cette expérimentation est de trouver une matrice non-inversible en calculant le déterminant de celle-ci. En effet, une matrice est non inversible si son déterminant est nul. La fonction d'évaluation de notre problème correspond donc à la distance du déterminant de la matrice à zéro.

Deux méthodes principales existent pour calculer le déterminant :

- la formule de Laplace : le calcul du déterminant d'une matrice carrée de taille n est équivalent au calcul de n déterminants d'une matrice de taille n-1. La complexité de cette méthode est d'ordre $O(n!)$.
- l'élimination de Gauss : le but est de combiner les lignes et les colonnes pour créer une matrice avec un maximum de zéro. Le calcul final est effectué grâce à la formule de Laplace. Grâce aux zéros précédemment introduits dans la matrice, le calcul est accéléré. La complexité de l'algorithme d'élimination de Gauss est de l'ordre de $O(n^4)$.

Pour augmenter le temps de calcul de l'évaluation "artificiellement", nous avons décidé d'implémenter la méthode de Laplace avec une matrice de taille 11x11. Le temps moyen nécessaire pour le calcul d'un déterminant d'une telle matrice est d'environ 18 secondes sur notre configuration [18].

Pour trouver notre matrice non-inversible avec un algorithme génétique, les génomes seront constitués d'une liste d'entiers appartenant à l'ensemble $[-10, 10]$. Les paramètres de l'algorithme génétique ont été trouvés en testant différentes valeurs sur un algorithme génétique classique (non parallélisé). Ils correspondent au plus petit nombre de générations possible pour que l'algorithme converge. Les paramètres sont ainsi les suivants :

- Taille de la population : 2000
- Méthode de sélection : tournoi à 7 avec élitisme
- Taux de croisement : 55%
- Taux de mutation : 7%

La figure 4.8 représente les courbes lissées pour cette expérimentation. Il est important

18. Configuration :
    - Processeurs : Dual-Core Intel Xeon 2.66GHz
    - Mémoire vive : 4Go cadencée à 1.33GHz

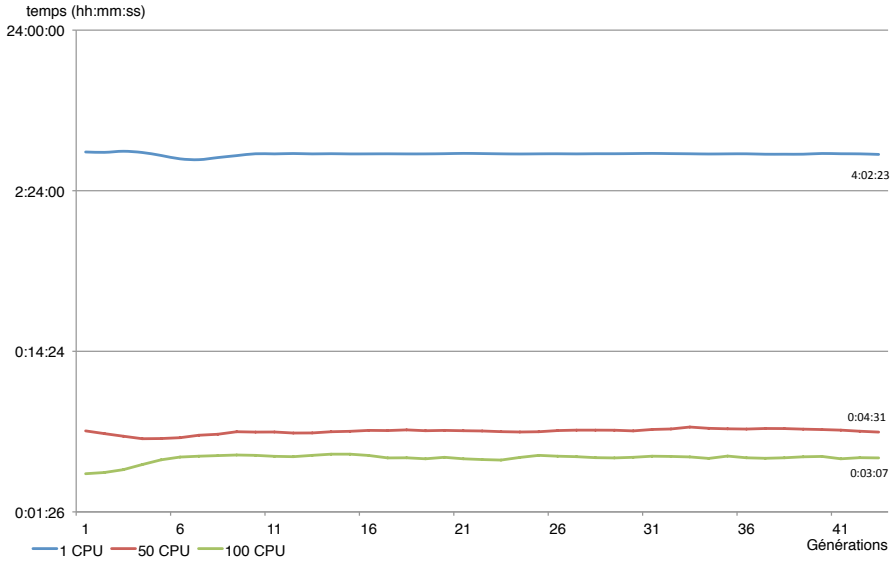


FIGURE 4.8 – *Temps de calcul nécessaire pour chaque génération (courbes lissées). La parallélisation réduit proportionnellement le temps de calcul de chaque génération.*

de noter que l'échelle du temps (en ordonnées) est une échelle logarithmique. Nous avons utilisé trois configurations de grille avec 1, 50 et 100 processeurs. Dans les trois cas, le temps de calcul de chaque génération est quasiment constant. Le temps de calcul est ici suffisant pour compenser la durée des échanges occasionnée par le réseau. De plus, le temps de calcul global est inversement proportionnel au nombre de processeurs. En effet, avec un processeur, le calcul d'une génération prend en moyenne 243 minutes, avec 50 processeurs, il ne faut que 4.5 minutes (54 fois moins qu'avec un processeur) et 3 minutes avec 100 processeurs (soit 81 fois moins qu'avec un processeur). Le temps pris par le réseau pour effectuer les échanges de génomes pour cette expérimentation n'est donc pas important en comparaison du temps gagné par la parallélisation de l'algorithme.

Troisième situation : Application à notre modèle

Nous allons maintenant appliquer notre méthode de parallélisation à notre problème de génération de créatures artificielles. Pour cela, nous allons utiliser une des créatures décrites dans la partie 5.1.2. Cette créature est en fait un organe capable de déplacer un substrat d'un point de l'environnement à un autre. Nous verrons précisément les détails de cet organe dans la partie 5.1.2. Toutes les caractéristiques de la cellule initiale sont codées dans un génome. Sa taille est d'environ 19 kilos octets et le temps de simulation pour développer un organisme varie fortement, de quelques secondes (dans les premières générations principalement, lorsque l'organisme n'est capable de rien faire) à quelques minutes (quand l'organisme se développe et répond à la fonctionnalité qui lui est demandée). Du

fait de cette grande disparité du temps de simulation, la répartition de la charge fournie par ProActive est extrêmement utile puisqu'elle permettra de répartir les individus de la population de l'algorithme génétique sur tous les esclaves indépendamment de leur temps de simulation.

Dans cette expérimentation, les paramètres de l'algorithme génétique sont les suivants :

- Taille de la population : 750 individus,
- Méthode de sélection : Tournoi à 7 participants avec élitisme,
- Taux de croisement : 65%,
- Taux de mutation : 5%.

Les courbes de la figure 4.9 représentent le temps de calcul pour chaque génération pour un et pour 50 processeurs. Il est important de noter que l'axe des ordonnées (temps de calcul d'une génération) est gradué en utilisant une échelle logarithmique.

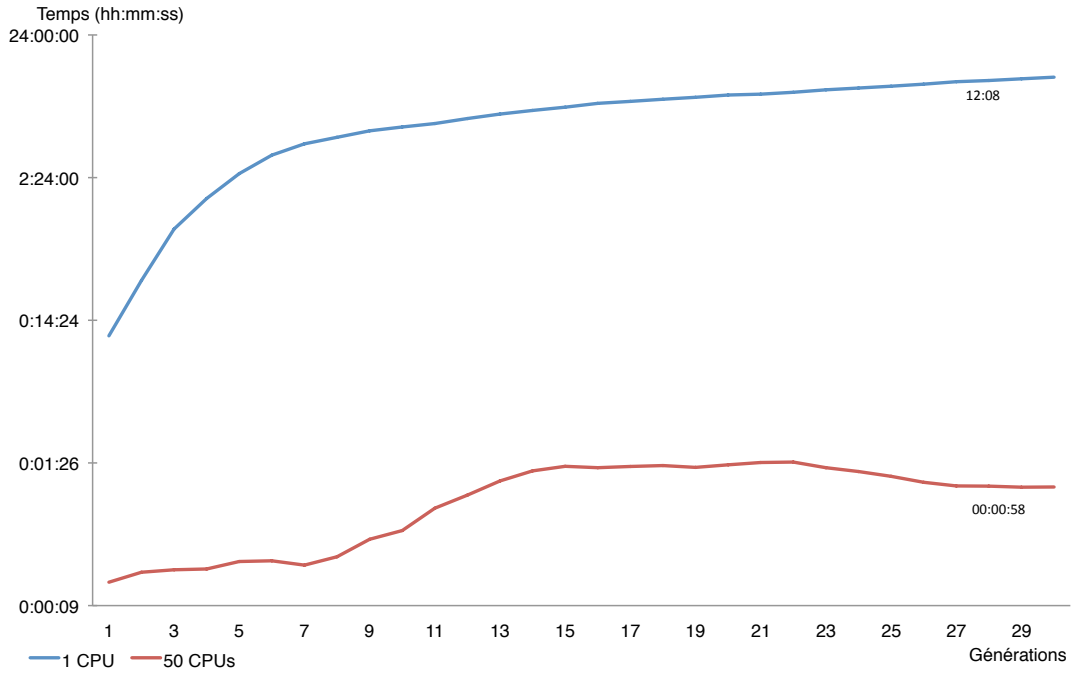


FIGURE 4.9 – *Temps de calcul nécessaire pour le calcul de chaque génération (courbe lissée). La parallélisation à l'aide d'une grille de calcul réduit l'explosion du temps de calcul due à l'évolution des créatures.*

Premièrement, les deux courbes croissent génération après génération étant donné que le temps moyen de calcul pour calculer la fonction d'évaluation d'une créature augmente. En effet, dans la première génération, une grande partie des génomes ne produiront que des créatures incapables de survivre plus de quelques secondes dans leur environnement. Ne possédant pas de métabolisme, ou alors peu performant, elles meurent quasiment immédiatement. La simulation s'arrête alors rapidement (le critère d'arrêt d'une simula-

tion dans ce cas est l'absence de cellule dans l'environnement). Mais, génération après génération, les créatures évoluent et utilisent les ressources de l'environnement pour se développer. Finalement, le temps de développement d'une créature complètement évoluée et capable de répondre à notre problème est d'environ 120 secondes de recherche.

La seconde observation intéressante est que le temps de calcul global a été fortement réduit grâce à la parallélisation. La sollicitation du réseau est moins importante que pour le problème du OneMax et le temps de calcul pour chaque créature est suffisant pour obtenir un bon résultat de la parallélisation même dans les premières générations où le temps de calcul d'un individu est pourtant très court. La différence entre les courbes est plus importante en avançant dans l'évolution des génomes. Ceci est possible grâce à la répartition fournie par ProActive qui répartit parfaitement le calcul des fonctions d'évaluation de manière asynchrone entre les différents esclaves. Ainsi, un esclave ayant une évaluation courte à calculer recevra dès la fin du calcul de celle-ci un nouveau génome à évaluer.

Le gain de la parallélisation pour notre problème est donc bien celui attendu. Il permet une grande réduction de temps de calcul nécessaire à l'évaluation d'une grosse population de génomes très hétérogènes. Il y a cependant quelques inconvénients à cette parallélisation.

4.1.5 Inconvénients de la parallélisation

Bien que la parallélisation réduise grandement le temps de calcul nécessaire pour générer des créatures complexes, cette parallélisation a aussi quelques limites.

Dans notre méthode de parallélisation, nous avons utilisé une grille de calcul. Nous utilisons Grid5000 [19], grille de calcul expérimentale française composée de 5000 processeurs. L'utilisation d'une telle structure impose la réservation de ressources et la cohabitation avec d'autres utilisateurs qui effectuent des calculs en même temps. Ceci implique une restriction : il est quasiment impossible de refaire à l'identique une expérimentation car il est difficile de réunir exactement la même configuration expérimentale. En effet, il est très difficile d'obtenir exactement les mêmes ressources d'une réservation à l'autre et l'utilisation d'un réseau impliquera toujours des perturbations qu'il est impossible de maîtriser.

Le second inconvénient de la parallélisation est dû à l'architecture que nous avons utilisée pour notre algorithme génétique. L'architecture Maître/Esclaves ne répartit pas la population gérée par le maître sur les différents esclaves. Déjà que la population d'un algorithme génétique centralisé utilise une quantité de mémoire importante à cause de la

19. https://www.grid5000.fr

taille et du nombre de génomes, l'utilisation du middleware ProActive ne fait qu'accroître le problème. En effet, chaque génome est encapsulé dans une tâche ProActive qui sera envoyée à l'esclave par la suite. L'explosion mémoire induit par le middleware peut être réduite en utilisant la méthode de parallélisation par îlots ou la méthode hiérarchique qui permet de répartir la charge du maître sur plusieurs esclaves, mais le temps de convergence d'un tel algorithme est supérieur à un algorithme classique [Bianchini and Brown, 1993]. Il pourrait toutefois être intéressant d'implémenter cette version de l'algorithme et de la comparer sur Grid5000 avec la version actuelle de la parallélisation. La répartition de la population en îlots permettra la répartition du problème de mémoire sur plusieurs machines et supprimera le goulot d'étranglement constitué par le maître.

4.2 Ajustement automatique des taux de croisement et de mutation

4.2.1 Principe général

Les taux de mutation et de croisement de l'algorithme génétique sont les principaux paramètres à régler pour améliorer la vitesse de convergence de celui-ci. Différents travaux ont été réalisés pour essayer de supprimer ces deux paramètres souvent difficiles à configurer. Il existe trois approches principales :

- L'approche *centralisée* consiste à adapter la valeur des paramètres des opérateurs génétiques à partir d'une règle d'apprentissage globale qui prend en compte la productivité de l'opérateur génération après génération [Davis, 1989, Eiben et al., 1999, Julstrom, 1997]. La productivité de l'opérateur est mesurée en regardant la qualité des solutions issues de l'opérateur en fonction des solutions de la génération précédentes. Ainsi, si un opérateur génère un grand nombre de bonnes solutions, la probabilité qu'il soit utilisé augmente. Cette approche est plus onéreuse en mémoire qu'un algorithme génétique du fait de la nécessité de stocker toute la lignée d'un individu afin de trouver quel opérateur a été le plus bénéfique pour le produire.
- Dans la stratégie *décentralisée*, les paramètres des opérateurs génétiques sont directement codés dans le génome de l'individu [Srinivas and Patnaik, 1994, Eiben et al., 1999]. Ils sont ainsi soumis à l'évolution génétique tout comme le reste du génome. Des opérateurs génétiques particuliers sont appliqués sur ces paramètres afin de les ajuster correctement. La principale difficulté de cette approche réside dans la définition de ces méta-opérateurs.
- Une approche hybride, présentée dans [Gomez, 2004], consiste à coder pour chaque

individu ces paramètres dans son génome et de définir les méta-opérateurs aléatoirement dans une base de règles d'apprentissage et à l'appliquer individuellement à chaque solution de la population. Ce choix est cependant guidé par la productivité des règles : une règle produisant de meilleures solutions aura plus de chance d'être sélectionnée qu'une autre.

Dans nos travaux, nous avons adapté le taux de mutation et le taux de croisement au cours de la simulation. Leur valeur sera redéfinie à chaque nouvelle génération en fonction de la note maximale obtenue dans la population. Le but de la modification de ces paramètres n'est pas de les supprimer mais plutôt d'essayer d'améliorer la vitesse de convergence de l'algorithme en modifiant le comportement de ce dernier en fonction de la génération courante. Ainsi, nous sommes partis de l'hypothèse qu'au début de la recherche, l'algorithme devait couvrir un maximum de l'espace de recherche en modifiant fortement les génomes de la population. Pour cela, nous avons choisi de donner un fort taux de mutation et un faible taux de croisement au début de la recherche. Au fur et à mesure de l'amélioration des solutions trouvées, il faut basculer vers une recherche plus locale en privilégiant la modification et la combinaison des bonnes solutions trouvées. Ainsi, plus les générations passent et plus le taux de mutation va décroître alors que le taux de convergence augmentera.

4.2.2 Application à notre modèle

Dans notre algorithme génétique, nous adaptons les taux de mutation et de croisement en deux étapes. Au début de la simulation, on utilise un fort taux de mutation (de l'ordre de 90%) et un faible taux de croisement (environ 10%). Ces valeurs seront utilisées jusqu'à ce qu'un "bon" organisme apparaisse dans la population (avec des fonctions de bases telles que le métabolisme et la division déjà acquises). Une fois cet organisme obtenu, on applique la formule suivante pour ajuster le taux de mutation mr et le taux de croisement cr :

$$\begin{aligned} mr(fit) &= 0.85 - \frac{0.65}{1 + e^{\frac{10}{3} - \frac{10fit}{maxfit}}} \\ cr(fit) &= 0.95 - mr(fit) \end{aligned}$$

avec

- mr et cr sont respectivement les taux de mutation et de croisement,
- fit est la valeur de la meilleure évaluation de l'individu de la population courante,
- $maxfit$ est une évaluation de la note maximum qu'il est possible d'obtenir.

L'équation présentée ci-dessus donne la courbe présentée dans la figure 4.10 avec $maxfit = 5000$. Quand l'évaluation maximum de la population est proche de zéro, le taux de mutation est alors égal à 0.8276 et le taux de croisement est égal à 0.1224. Quand l'évaluation croît, les valeurs sont inversées : pour une évaluation égale à 5000, le taux de mutation est égal a 0.2008 et le taux de croisement à 0.7492.

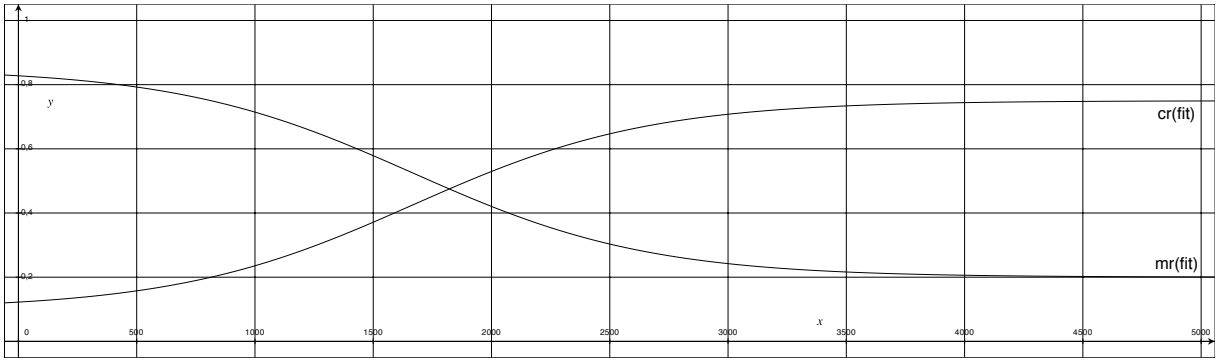


FIGURE 4.10 – *Variation des taux de mutation et de croisement (ordonnées) en fonction de la valeur de l'évaluation maximum (abscisse).*

4.2.3 Expérimentation – Validation de l'approche

Pour tester la validité de cette hypothèse, nous avons fait des expérimentations en utilisant les créatures que nous avons développées grâce à notre modèle. Dans ce cas, nous utiliserons à nouveau le système de transfert présenté en détail dans le chapitre 5.1.2.

La courbe 4.11 présente les résultats en activant (courbe en bleu) ou en désactivant (courbe en rouge) la méthode d'ajustement des taux de mutation et de croisement. Contrairement à ce que nous attendions, la vitesse de convergence en utilisant cette méthode n'est pas améliorée. Au contraire, la convergence globale est décalée de 35 générations sur cette expérimentation.

Afin de confirmer ce résultat, nous avons testé à nouveau cette méthode sur un organe plus complexe, développé dans le but de créer une structure auto-alimentée. Tous les détails de cet organe et de l'organisme entier sont présentés dans la section 5.1.3. La courbe 4.12 montre la convergence de ce nouvel organisme, en activant (courbe en bleu) ou en désactivant (courbe en rouge) comme précédemment l'ajustement automatique des taux de mutation et de croisement. Une fois encore, cette méthode ne donne pas les résultats attendus. Bien que l'écart entre les deux courbes soit moins important que dans l'expérimentation précédente, l'ajustement des paramètres altère la convergence de l'algorithme en le retardant d'environ 6 générations.

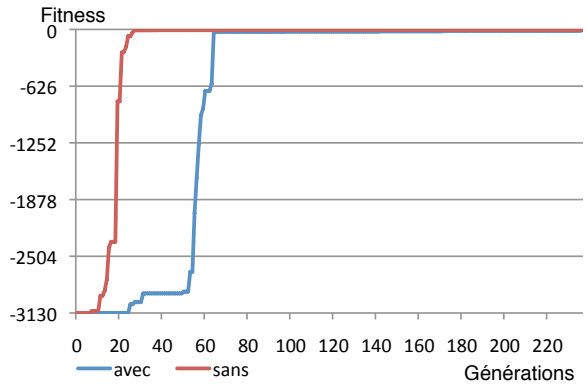


FIGURE 4.11 – *Comparaison de la convergence de l'algorithme génétique avec ou sans ajustement des taux de mutation et de croisement sur l'expérimentation du système de transfert.*

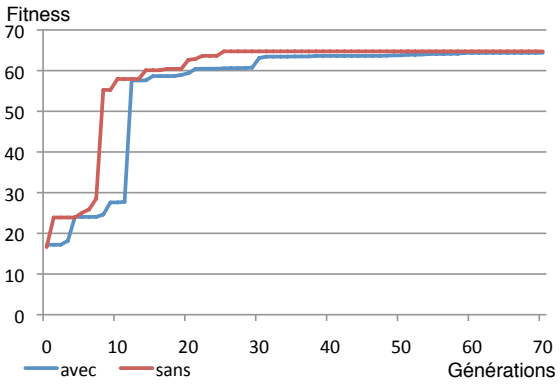


FIGURE 4.12 – *Comparaison de la convergence de l'algorithme génétique avec ou sans ajustement des taux de mutation et de croisement appliqué à un des organes de la structure auto-alimentée.*

Bilan de l'expérimentation

Cette méthode d'ajustement des paramètres de l'algorithme génétique ne semble donc pas profitable à la convergence de la recherche. C'est principalement le début de la convergence qui est affecté par la modification des paramètres de l'algorithme génétique. La mutation ne semble pas être le meilleur opérateur pour cribler convenablement l'espace de recherche dans la phase d'initialisation de l'algorithme génétique. Une recherche purement aléatoire pourrait être plus profitable à cette recherche initiale. Cependant, pour aider l'algorithme à converger, nous avons préféré tester une autre technique basée sur la décomposition de la fonction d'évaluation en sous-fonction afin de guider la convergence en évaluant des points importants du développement des créatures.

4.3 Evaluation multi-objectifs

4.3.1 Principe général

Chaque individu de la population de l'algorithme génétique est évalué indépendamment des autres. La fonction d'évaluation est définie par l'utilisateur et permet de noter l'individu par rapport au but global qu'il lui est demandé d'accomplir. Dans notre problème, le but est souvent difficile à atteindre étant donné la complexité du comportement global demandé ainsi que du métabolisme que la créature doit développer. Pour réduire la difficulté, nous allons décomposer la fonction d'évaluation en plusieurs sous-objectifs décrivant les différents stades d'évolution de la créature. Cette approche, ap-

pelée *évolution incrémentale* [**?**], a été utilisée plusieurs fois dans différents domaines tels que la simulation comportementale [Kodjabachian and Meyer, 1998, Parker, 2001, Mouret and Doncieux, 2008] ou la programmation génétique [Winkeler and Manjunath, 1998]. Plusieurs études sur son efficacité ont été menées [Urzelai and Floreano, 1999, Walker, 2004] et il en résulte que le temps de calcul n'est globalement pas supérieur à une version classique des algorithmes génétiques mais permet cependant de trouver des solutions plus efficientes. De plus, l'élitisme [20] utilisé avec cette méthode permet de ne pas détruire ce qui a été précédemment produit étant donné que le meilleur individu pour un objectif sera toujours sélectionné dans l'évaluation de la population suivante.

Dans notre problème, dans la plupart des cas, on peut décomposer la fonction d'évaluation en au moins trois sous objectifs :

- le métabolisme, qui est la fonction de plus bas niveau nécessaire à la survie de la créature dans l'environnement,
- la quantité de cellules créées durant la simulation qui détermine la capacité d'un organisme à se développer,
- la note globale de l'évolution qui donne la performance d'un organisme à résoudre le problème qui lui est posé (la fonctionnalité que doit assurer l'organisme créé).

Le dernier objectif de la simulation peut lui aussi être décomposé afin d'aider au maximum la convergence de l'algorithme génétique. On peut par exemple ajouter des critères de déplacement de substrats, de valeur d'énergie de l'organisme...

La valeur finale de la fonction d'évaluation est calculée à l'aide d'une agrégation des différentes sous-fonctions. Au début de l'évolution, seule la note correspondant au métabolisme de l'organisme sera considérée. Quand la première division se produit, le deuxième critère est à son tour pris en compte et ainsi de suite.

4.3.2 Expérimentation – Validation de l'approche

Pour tester la validité de cette hypothèse, nous avons développé le système de transfert déjà utilisé dans les expérimentations précédentes et présenté dans la section 5.1.2.

Pour cette expérimentation, nous avons préféré garder des taux de mutation et de croisement constants étant donné les résultats précédemment obtenus. Nous avons choisi une valeur du taux de mutation de 10% et un taux de croisement de 80%. La population est constituée de 750 individus. La sélection se fait là aussi par tournoi à 7 avec élitisme, et le remplacement des individus se fait en remplaçant les plus mauvais d'abord.

20. Méthode qui consiste à toujours garder le meilleur individu d'une population pour l'évaluation suivante

Les courbes 4.13 montrent les courbes de convergence de l'algorithme génétique appliqué au système de transfert en activant le multi-objectifs (en rouge) ou en le désactivant (en bleu). On voit clairement que l'ajout d'objectifs aide l'algorithme à trouver les meilleures solutions plus rapidement.

La méthode avec multi-objectifs converge beaucoup plus rapidement, l'algorithme étant guidé pour sélectionner des solutions viables. Nous l'utiliserons donc au maximum dans nos expérimentations futures.

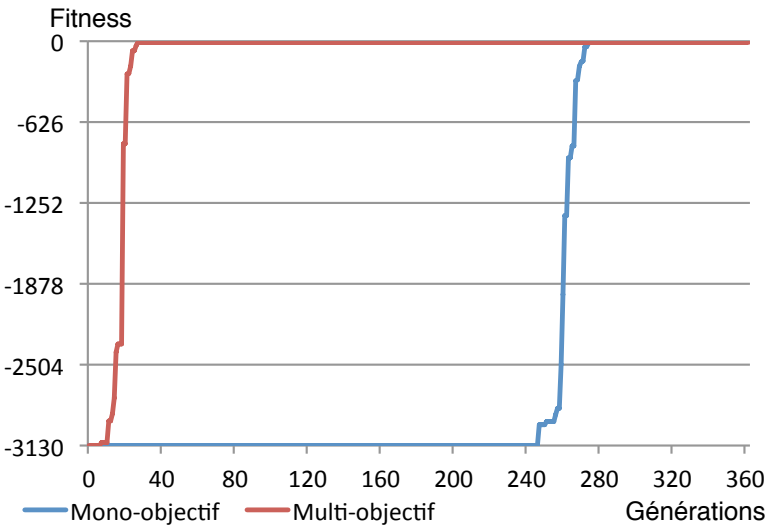


FIGURE 4.13 – *Courbes de convergence du système de convergence en activant (en rouge) et en désactivant (en bleu) le multi-objectifs dans l'algorithme génétique*

4.4 Bilan

Au vue des résultats des expérimentations que nous avons faites précédemment, nous allons utiliser par la suite la méthode de parallélisation Maître/Esclaves qui est suffisante pour notre modèle. Il pourra être cependant intéressant de tester la méthode par îlots afin de réduire les contraintes mémoires du maître. La méthode d'ajustement des paramètres ne sera pas utilisée étant donné les résultats peu satisfaisants obtenus lors des tests. Enfin, nous utiliserons la méthode d'aide à la convergence précédemment présentée qui guide parfaitement et réduit énormément le temps de convergence de l'algorithme génétique.

5

Expérimentations

Après avoir présenté notre modèle et les techniques que nous avons employées au niveau de l'algorithme génétique pour réduire le temps de convergence, ce chapitre montre les capacités du modèle. Pour ce faire, nous allons présenter divers organismes montrant la complexité des comportements pouvant être obtenus.

Nous montrerons tout d'abord la capacité de notre modèle à produire des organes. Un organe, par définition, est un ensemble de cellules spécialisées qui possède une fonction spécifique. Nous présentons ainsi un organe, appelé le moissonneur, capable de dégrader un substrat particulier de l'environnement. Nous verrons ensuite un organe capable de déplacer un substrat d'un point à un autre de l'environnement. Suite au développement de ces deux organes, nous avons voulu tester les capacités d'assemblage de notre modèle. En partant du système de transfert et en développant deux nouveaux organes, nous avons pu produire un organisme plus complexe en assemblant quatre organes différents et en obtenant un organisme auto-alimenté utilisant au mieux les ressources de l'environnement.

Dans un second temps, nous avons étudié les capacités de notre modèle à générer des formes. Nous avons commencé par produire une forme d'étoile de mer. Le principal défi dans ce problème était encore une fois de pouvoir allier le métabolisme des cellules et le but qu'il leur est demandé d'atteindre. Nous verrons aussi qu'avec un même génome, il est possible de produire n'importe quelle forme désirée, seulement en déplaçant les morphogènes dans l'environnement. La génération de telles formes a pour principal intérêt la possibilité de fournir un modèle de croissance à des créatures de plus grande échelle telles que les créatures présentées dans les travaux de Karl Sims [Sims, 1994] ou ceux récemment repris dans notre équipe par Nicolas Lassabe [Zykov et al., 2008].

Nous verrons dans une troisième partie que tous les organismes que nous avons générés possèdent une propriété très intéressante : ils sont capable de se régénérer en cas de dommage. Cette propriété semble être inhérente au codage du modèle puisque cette

spécification n'a pas été pensée lors de sa conception.

Finalement, nous présenterons une étude préliminaire de la création de deux simulateurs supplémentaires autour de ce modèle : un simulateur physique permettant de plonger nos créatures dans un monde aux lois newtoniennes et un simulateur hydrodynamique permettant la simulation des flux de matière de l'environnement. Les premiers résultats présentés dans cette partie sont très encourageant et montrent la complexité des futurs créatures que nous allons pouvoir obtenir grâce à l'utilisation conjointe des trois simulateurs.

5.1 Les organes développés

5.1.1 Expérimentation 1 : Le moissonneur

Conditions expérimentales

Cette créature a été la première développée à l'aide de ce modèle. Elle avait pour but de tester les capacités de convergence du mécanisme d'évolution. Cette expérimentation a été réalisée avant même d'avoir toutes les améliorations décrites dans le chapitre précédent. Présentée dans [Cussat-Blanc et al., 2007], son but est de dégrader un substrat particulier en le transformant en énergie vitale et en rejetant les déchets de la transformation.

Plus en détail, les substrats disponibles dans l'environnement sont les suivants :

- Un substrat A, représenté en rouge, est le substrat à dégrader. Il se diffuse dans l'environnement à la vitesse simulée d'une étape de diffusion toutes les deux secondes.
- Un substrat B, représenté en bleu, sert de matériel de division à la cellule. Il est nécessaire pour produire chaque nouvelle cellule. Lui aussi se diffuse mais à la vitesse simulée d'une étape de division toutes les 1,5 seconde.
- Un substrat C, représenté en rose, représente un déchet produit par la réaction chimique de dégradation du substrat A. Il doit être évacué par la cellule qui ne peut contenir que 7 substrats à l'intérieur de sa membrane.

La dégradation du substrat A se fait à l'aide de la réaction chimique $2A \rightarrow B+C$ (-20). Ainsi, avec deux unités de substrats A, une cellule peut produire une unité de B, substrat nécessaire à la division cellulaire, et une unité de déchet C qui devra être rejeté dans l'environnement.

L'objectif de la créature étant de dégrader un maximum de substrat A, l'environnement initial en contient une grande quantité (1900 unités). Ce substrat est réparti selon le tableau 5.1.

Quantité	750	150	1000
Coordonnées	(3,2)	(10,9)	(18,1)

TABLE 5.1 – *Répartition des 1900 unités de substrat rouge dans l'environnement. La diffusion permettra une répartition homogène dans l'environnement.*

Afin de réaliser son travail, l'organisme possède les actions présentées dans le tableau 5.2. Ces actions lui permettent de :

- se diviser dans n'importe quelle direction,
- déclencher la dégradation du substrat A,
- absorber du substrat A,
- absorber ou rejeter du substrat B,
- attendre une condition particulière de l'environnement.

Action	Coût énergétique	Durée	Exigences particulières
Divide to NorthEast	40	10000	Une division consomme 2 unités de B
Divide to NorthWest	40	10000	
Divide to SouthEast	40	10000	
Divide to SouthWest	40	10000	
Transform $2A \rightarrow B + C$	-20	5000	2 unités de A
Absorb A from North	2	2000	La cellule doit contenir moins de 7 substrats
Absorb A from South	2	2000	
Absorb A from East	2	2000	
Absorb A from West	2	2000	
Absorb C from North	3	2000	La cellule doit contenir moins de 7 substrats
Absorb C from South	3	2000	
Absorb C from East	3	2000	
Absorb C from West	3	2000	
Evacuate C to North	2	2000	La cellule doit contenir au moins 1 unité de C
Evacuate C to South	2	2000	
Evacuate C to East	2	2000	
Evacuate C to West	2	2000	
Do nothing	1	250	-

TABLE 5.2 – *Liste des actions possibles des cellules pour développer le moissonneur.*

La cellule possède tous les capteurs possibles dans ses quatre coins nord, sud, est et ouest.

Fonction d'évaluation et paramètres de l'algorithme

La fonction d'évaluation de l'algorithme génétique permettant de trouver la créature dégradant un maximum de substrat A est très simple. La méthode de décomposition de

la fonction d'évaluation n'est pas utilisée dans ce cas, l'organisme ayant peu d'étapes à franchir durant son évolution. En effet, le métabolisme de l'organisme est secondaire dans cette expérimentation puisque la dégradation du substrat A produit suffisamment d'énergie pour effectuer quelques actions par la suite.

L'évaluation consiste donc simplement à laisser l'organisme se développer et attendre la mort de la totalité des cellules et à calculer la quantité de substrat A restant dans l'environnement. On peut ainsi en déduire par soustraction la quantité de substrat A consommée.

L'objectif de l'algorithme génétique est donc de maximiser cette fonction d'évaluation. Pour cela, on utilise les paramètres suivants pour l'algorithme génétique :

- sélection : tournoi à 7 avec élitisme,
- taux de mutation : 5% ; taux de croisement : 65%,
- remplacement : individus les plus mauvais,
- taille de la population : 500 individus.

Résultats

A la suite de l'évolution génétique de la population d'organismes, le meilleur d'entre eux, présenté par la colonne de droite de la figure 5.1, est capable de dégrader plus de 1700 unités de substrat A, soit près de 90% de la quantité totale. La stratégie déployée par cet organisme est intéressante à étudier. A partir d'une cellule unique, il commence tout d'abord à parcourir l'environnement sur sa diagonale en utilisant une série de division dans la direction nord-est. Une fois qu'il a engrangé suffisamment d'énergie, il commence à prendre de l'amplitude en se divisant à la fois vers le nord-ouest et le sud-est. Cette caractéristique est possible grâce au réseau d'optimisation des actions qui permet de retarder le déclenchement d'une action en augmentant le coût de celle-ci. Il en résulte un organisme déployé sur une colonne entière de l'environnement, se déplaçant de gauche à droite tout en ramassant une grande quantité de substrats [21].

L'étude de l'évolution du comportement génération après génération est tout aussi intéressante. Les organismes présentés par la figure 5.1 montrent l'émergence de la stratégie précédemment décrite tout au long de l'évolution du génome de la créature. Ainsi, dans les premières générations, l'organisme n'est pas capable de se déplacer et parvient tout juste à développer un métabolisme. De ce fait, son temps de survie dans l'environnement est très court. Après quelques générations supplémentaires, l'organisme commence à se déplacer dans une seule direction en développant une stratégie de divisions successives. A

21. Pour rappel, l'environnement étant torique, l'organisme a la possibilité de passer de gauche à droite et de bas en haut.

FIGURE 5.1 – *Evolution du comportement du moissonneur, génération après génération. Lors de la première génération, il meurt très rapidement après s'être quelque peu développé. A la génération 5, il est capable de se déplacer sur la diagonale grâce à une suite de divisions cellulaires. Après 15 générations, il se déploie sur une colonne et se déplace horizontalement afin de couvrir un maximum de surface sur l'environnement torique.*

la fin de l'évolution, l'organisme se déploie en plus de se déplacer à l'aide de division et de mort cellulaire, permettant ainsi d'agrandir son champ d'action.

Bilan

Le but de la conception de cet organisme était de vérifier les capacités de convergence de notre modèle et ainsi de vérifier qu'il est capable de produire des créatures. Le développement de celui-ci ne comportait pas de difficulté majeure mais il a permis de vérifier que le modèle fonctionne et qu'il est possible d'utiliser un algorithme génétique pour la recherche des génomes de nos créatures.

5.1.2 Expérimentation 2 : Le système de transfert

Conditions expérimentales

L'objectif de cette créature, présentée dans [Cussat-Blanc et al., 2008b] est de développer une structure cellulaire capable de transporter un substrat particulier d'un endroit de l'environnement de coordonnées (5,6) à un point final de coordonnées (18,18). Pour cela, l'environnement, de taille 25x25, contient deux substrats :

- Un substrat rouge nommé A qui doit être transféré par l'organisme. Ce substrat a la particularité de ne pas se diffuser dans l'environnement.
- Un substrat gris, appelé B qui est utilisé par les cellules pour développer leur métabolisme et comme matériel de division. Ce substrat se diffuse dans l'environnement à la vitesse d'une étape de diffusion toutes les 2000 millisecondes en temps simulé.

Nous plaçons dix unités de substrat rouge à l'endroit où le système de transfert doit commencer le transfert (coordonnées (5,6) dans l'environnement) et nous ajoutons en plus les quantités de substrat gris suivantes :

- 180 unités en (14,15),
- 180 unités en (4,8),
- 150 unités en (9,11),
- 180 unités en (18,18),
- 130 unités en (13,11),
- 200 unités en (13,14),
- 200 unités en (11,12).

La propriété de diffusion du substrat permettra de répartir rapidement ce substrat dans l'environnement, avant même que le système de transfert n'ait fini de se développer.

Les cellules peuvent effectuer les actions suivantes :

– se diviser dans les quatre diagonales, ce qui permet un développement complet de la créature dans l'environnement si elle en a besoin,
– absorber ou rejeter du substrat dans les quatre directions cardinales,
– transformer une unité de substrat gris en énergie vitale,
– ne rien faire.

La table 5.3 indique les paramètres des actions précédentes (coût énergétique, durée en millisecondes et exigences supplémentaires) :

Action	Coût énergétique	Durée	Exigences
Divide to NorthEast	30	10000	1 unité de B
Divide to NorthWest	30	10000	
Divide to SouthEast	30	10000	
Divide to SouthWest	30	10000	
Transform B into Energy	-30	5000	1 unité de B
Absorb A from North	0.5	2000	La cellule doit contenir moins de 7 substrats
Absorb A from South	0.5	2000	
Absorb A from East	0.5	2000	
Absorb A from West	0.5	2000	
Absorb B from North	2	2000	La cellule doit contenir moins de 7 substrats
Absorb B from South	2	2000	
Absorb B from East	2	2000	
Absorb B from West	2	2000	
Evacuate A to North	0.5	2000	La cellule doit contenir au moins 1 unité de A
Evacuate A to South	0.5	2000	
Evacuate A to East	0.5	2000	
Evacuate A to West	0.5	2000	
Do nothing	1	250	-

TABLE 5.3 – *Liste des actions possibles des cellules pour développer le système de tranfert.*

Chacun des quatre coins de la cellule (nord, sud, est et ouest) possède un capteur pour le substrat A et un capteur pour le substrat B.

Evaluation de l'organisme et paramètres de l'algorithme génétique

La fonction d'évaluation de l'organisme est divisée en trois sous-fonctions :
– le métabolisme de l'organisme *meth* (durée de la simulation en millisecondes),
– le nombre de cellules produites $nbCells$,
– la somme *goal* des carrés des distances de chaque unité de substrat A par rapport à la position finale exigée.

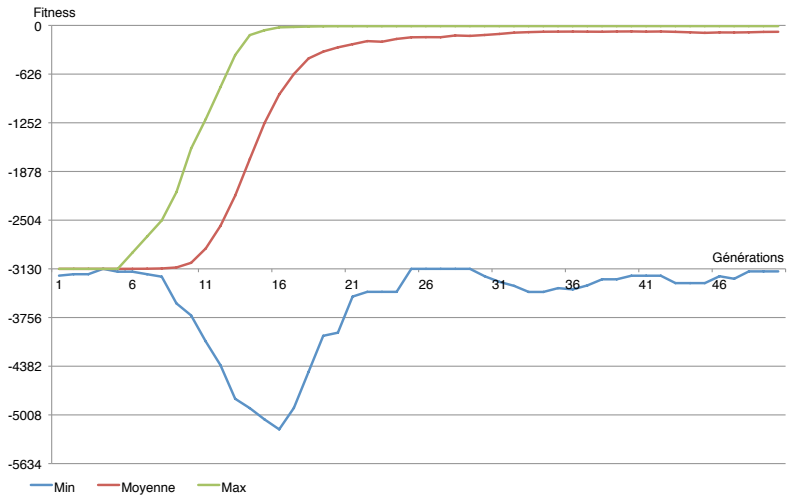


FIGURE 5.2 – *Courbes lissées des valeurs des évaluations minimum, moyenne et maximum de l'organisme. L'algorithme génétique maximise l'opposé de la somme des carrés des distances de la position des substrats rouges à leur but.*

La fonction d'évaluation finale est donnée par la formule :

$$fit = \alpha.metha + \beta.nbCells + \gamma.goal \quad avec\ \alpha = \frac{1}{100\ 000\ 000},\ \beta = \frac{1}{10\ 000}\ et\ \gamma = -1.$$

Les coefficients α, β et γ de la formule précédente sont affectés pour prendre plus en considération le but final de l'organisme que son métabolisme et sa capacité de développement. Ces derniers ne sont nécessaires qu'au début de l'évolution pour guider l'organisme vers un organisme viable.

Etant donné les résultats obtenus pour l'évolution des paramètres de l'algorithme génétique, nous avons utilisé pour le développement des créatures un jeu de paramètres fixes. Nous avons ainsi utilisé les valeurs suivantes :

- sélection : tournoi à 7 avec élitisme,
- taux de mutation : 5% ; taux de croisement : 65%,
- remplacement : individus les plus mauvais,
- taille de la population : 500 individus.

La figure 5.2 montre la courbe de convergence de l'algorithme génétique. Pour chaque génération, elle montre les variations des évaluations minimum, moyenne et maximum de la population. Le but de l'algorithme génétique est de maximiser la valeur de l'évaluation, c'est à dire de trouver l'organisme le plus adapté à notre problème de transport de substrat.

Il est intéressant d'observer l'amélioration du comportement de l'organisme au cours de l'évolution (figure 5.3). Dans un premier temps (génération 5), les premiers organismes développent leur métabolisme afin de simplement survivre pendant les premiers instants

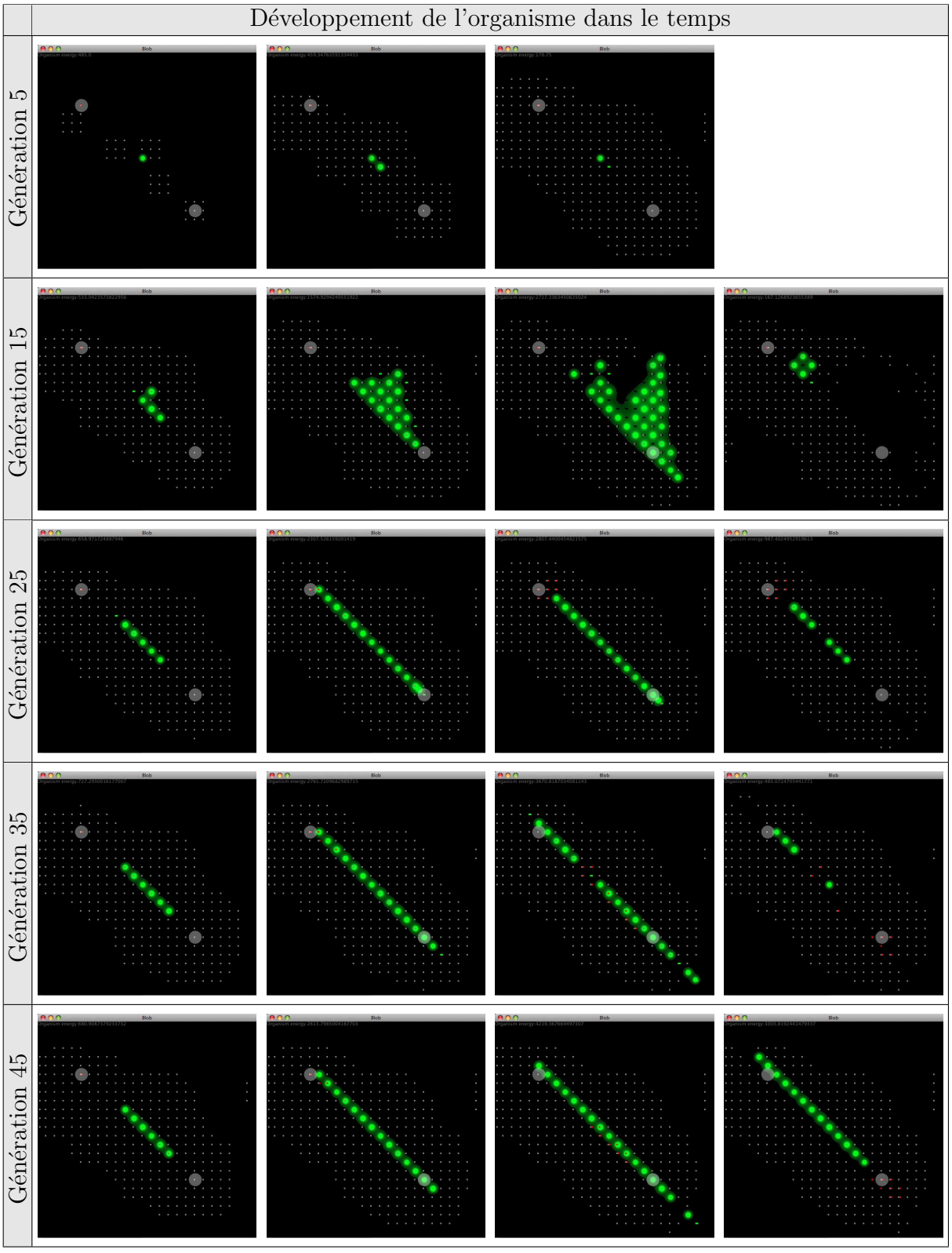


FIGURE 5.3 – *Développement du comportement et de la morphologie du système de transfert génération après génération.*

de la simulation. Pour cela, les organismes absorbent une unité de substrat gris puis le transforment en énergie. L'étape suivante est d'apprendre à se diviser (génération 15) puis à le faire dans la bonne direction (génération 25). Il faut donc désactiver les actions de division dans les mauvaises directions afin de former la structure qui permettra le transfert. Dès que cette structure est en place, les organismes déplacent du substrat, pas toujours dans la bonne direction, jusqu'à optimiser leur comportement afin de produire la fonction escomptée. La figure 5.4 montre le développement et le transfert produit par le meilleur organisme obtenu.

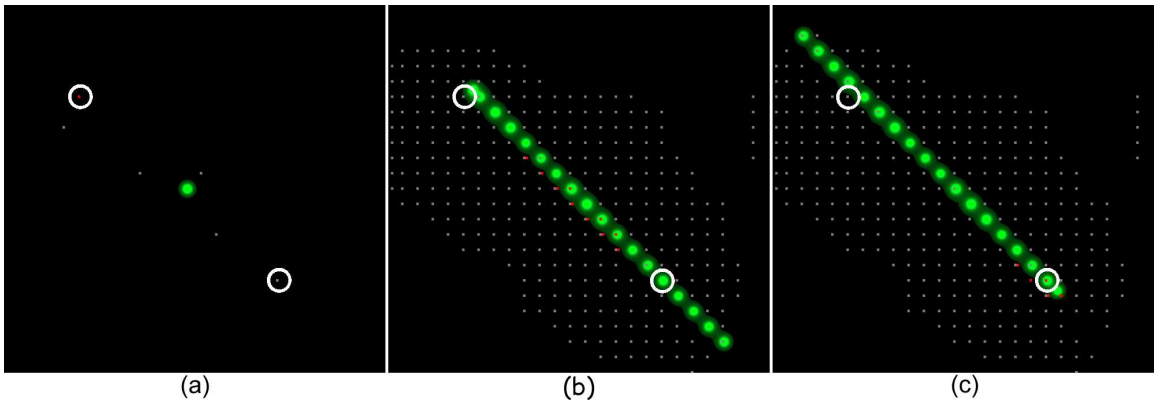


FIGURE 5.4 – *Développement du système de transfert. (a) Début de la simulation. (b) L'organisme se développe afin de créer sa structure et commence le transfert du substrat. (c) La créature transfère le substrat du point initial (cercle blanc dans le coin supérieur gauche) vers le point final (cercle dans le coin inférieur droit).*

Analyse du meilleur organisme obtenu

La figure 5.4 montre le développement du meilleur organisme. Premièrement, il est intéressant de noter que la division ne se fait que dans la direction du transfert. Les actions *Divide to NorthEast* et *Divide to SouthWest* sont correctement désactivées dans le chromosome de la liste des actions (voir table 5.4). Nous pouvons également observer que le développement de l'organisme s'arrête exactement au point de dépôt du substrat rouge. C'est le système de sélection d'action, présenté dans la table 5.5, et particulièrement la règle (A) qui produit ce comportement. En effet, l'action de division vers le nord-est sera effectuée si et seulement s'il n'y a pas de substrat rouge à l'est de la cellule. En arrivant sur le point de dépôt, la cellule détecte la présence de substrat rouge et arrête donc le développement de la structure dans cette direction.

Une fois cette partie de la structure développée, le transfert du substrat commence. Comme montré sur la figure 5.5, l'organisme utilise une série d'absorptions et d'évacuations du substrat rouge pour le déplacer de proche en proche. Là encore, c'est le système

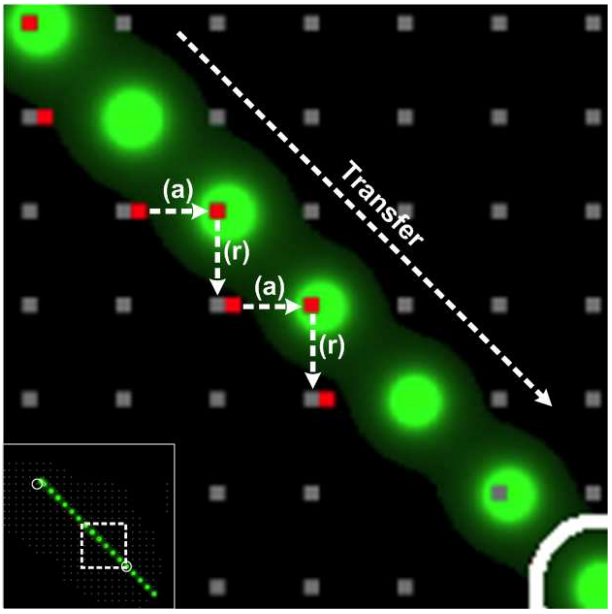


FIGURE 5.5 – *Le système de transfert en action : le substrat rouge est transféré de proche en proche du coin supérieur gauche au coin inférieur droit. L'organe utilise une série d'absorptions (a) et de rejets du substrat (r) pour déplacer le substrat.*

Action	Activation	Action	Activation
Divide to NorthEast	Désactivée	Absorb B from North	Désactivée
Divide to NorthWest	Activée	Absorb B from South	Activée
Divide to SouthEast	Activée	Absorb B from East	Activée
Divide to SouthWest	Désactivée	Absorb B from West	Désactivée
Transform B into Energy	Activée	Evacuate A to North	Activée
Absorb A from North	Désactivée	Evacuate A to South	Activée
Absorb A from South	Désactivée	Evacuate A to East	Activée
Absorb A from East	Désactivée	Evacuate A to West	Désactivée
Absorb A from West	Activée	Do nothing	Activée

TABLE 5.4 – *Chromosome de l'activation des actions du meilleur système de transfert. Les actions de division qui ne correspondent pas à la direction de transfert sont désactivées par ce chromosome.*

de sélection d'action qui définit le comportement de l'organisme. Ce sont les règles (D) et (H) qui produisent le déplacement. Pour ne pas dépasser le point final des substrats rouges, le réseau d'optimisation présenté dans la table 5.6 régule la distance à parcourir par les substrats. Pour ne pas dépasser le point final, le réseau augmente le coût de l'absorption nécessaire au transfert (ligne en jaune dans le tableau). La cellule dépassant le point final ayant un coût énergétique fort pour l'action d'absorption ne peut pas la déclencher et stoppe donc le transfert du substrat.

#	Règle			Priorité
(A)	(A_East=false) and (B_East=true)	→	(Divide NW)	(449603)
(B)		→	(Divide SE)	(131903632)
(C)	(A_North=false) and (A_East=false)	→	(Transform B->En)	(0)
(D)	(B_South=true) and (B_East=true)	→	(Absorb_A_West)	(0)
(E)	(A_North=false) and (A_East=false) and (A_West=false)	→	(Absorb_B_South)	(0)
(F)	(A_North=false) and (A_South=fase) and (A_East=false) and (B_East=false)	→	(Absorb_B_East)	(0)
(G)	(A_East=false) and (A_West=true) and (B_North=false) and (B_East=false)	→	(Evacuate_A_North)	(0)
(H)	(B_North=true) and (B_West=true)	→	(Evacuate_A_South)	(71304017)
(I)	(A_South=false) and (B_South=true) and (B_East=true) and (B_West=true)	→	(Evacuate_A_East)	(0)
(J)	(A_South=false) and (A_East=false)	→	(DoNothing)	(0)

TABLE 5.5 – *Système de sélection d'action produit par le génome du meilleur système de transfert obtenu grâce à l'algorithme génétique. Le transfert est dû aux règles (D) et (H) qui permettent un transfert de proche en proche. La règle (A) permet quant à elle la régulation de la longueur d'une moitié de l'organisme.*

	(Divide NW)	(Divide SE)	(Transform $B \rightarrow En$)	(Absorb_A_West)	(Absorb_B_South)	(Absorb_B_East)	(Evacuate_A_North)	(Evacuate_A_South)	(Evacuate_A_East)	Do Nothing	Gain/Perte
(Divide NW)	-	0.37	-0.56	-0.40	0.48	0.27	-0.42	0.22	-0.48	-0.30	-0.82
(Divide SE)	-0.37	-	0.28	0.53	0.02	0.09	-0.29	0.18	0.15	-0.04	0.55
(Transform $B \rightarrow En$)	0.56	-0.28	-	-0.04	-0.14	-0.17	-0.28	-0.32	-0.03	-0.45	-1.16
(Absorb_A_West)	0.40	-0.53	0.04	-	-0.69	0.16	-0.04	0.21	0.38	-0.59	-0.66
(Absorb_B_South)	-0.48	-0.02	0.14	0.69	-	0.15	0.01	0.44	-0.25	-0.18	0.51
(Absorb_B_East)	-0.27	-0.09	0.17	-0.16	-0.15	-	-0.22	-0.22	0.41	0.36	-0.17
(Evacuate_A_North)	0.42	0.29	0.28	0.04	-0.01	0.22	-	-0.20	0.14	-0.05	1.13
(Evacuate_A_South)	-0.22	-0.18	0.32	-0.21	-0.44	0.22	0.20	-	0.16	0.38	0.23
(Evacuate_A_East)	0.48	-0.15	0.03	-0.38	0.25	-0.41	-0.14	-0.16	-	-0.29	-0.76
(Do Nothing)	0.30	0.04	0.45	0.59	0.18	-0.36	0.05	-0.38	0.29	-	1.16

TABLE 5.6 – *Réseau de régulation du meilleur système de transfert obtenu par algorithme génétique. La ligne en jaune correspond à la régulation de l'action d'absorption du substrat A dans le but de ne pas transférer le substrat plus loin que nécessaire.*

5.1.3 Expérimentation 3 : L'organisme auto-alimenté et ses organes

Dans le but d'augmenter la complexité dans le développement de nos organismes, nous avons créé un organisme auto-alimenté. Le but de cette expérimentation est de mettre en exergue les capacités d'assemblage des organes dans leur environnement. Bien qu'encore à son stade primitif, nous verrons que nous pouvons positionner quatre organes différents dans l'environnement et les faire fonctionner conjointement afin de produire un organisme auto-alimenté.

Pour réaliser cette expérimentation, nous avons développé deux types d'organes : un système de transfert très proche de l'organe présenté dans la section 5.1.2 et un organe capable de transformer un substrat en énergie et de déposer le déchet de la réaction en un point particulier de l'environnement (à l'entrée d'un système de transfert). Le fonctionnement global de l'organisme est présenté dans la figure 5.6.

Dans cette partie, nous allons tout d'abord décrire les différents organes de cette créature puis nous analyserons la structure globale de l'organisme obtenu.

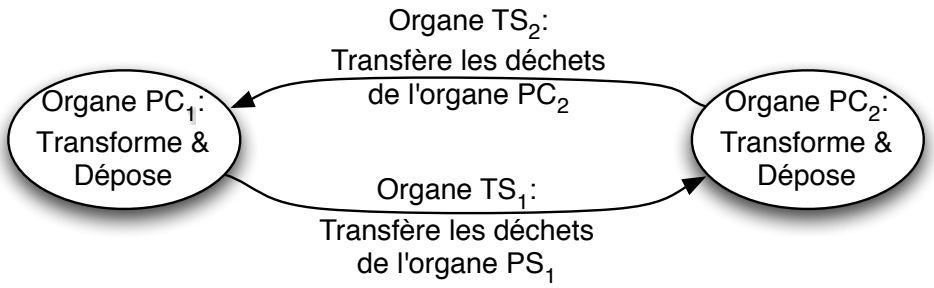


FIGURE 5.6 – *Schéma du fonctionnement de l'organisme auto-alimenté. Il est composé de deux types d'organes : les organes Producteurs-Consommateurs PC_1 et PC_2 capables de transformer des substrats et de les positionner en un point particulier de l'environnement ; les organes TS_1 et TS_2 (Transfert System) capables de transférer des substrats d'un point à un autre de l'environnement.*

Conditions expérimentales

L'environnement est composé de trois substrats :

- M (représenté en bleu sur les figures suivantes) qui sera utilisé par l'organisme pour son métabolisme initial,
- A et B (respectivement représentés en rouge et en jaune sur les figures suivantes) qui seront utilisés par l'organisme pour créer la structure auto-alimentée.

Trois transformations de substrat sont possibles :

Action	Coût	Durée	Exigences	Organes			
				TS_1	TS_2	PC_1	PC_2
Divide to NorthEast	30	10000	1 unité de M	X	X	X	X
Divide to NorthWest				X	X	X	X
Divide to SouthEast				X	X	X	X
Divide to SouthWest				X	X	X	X
Transform $M \rightarrow energie$	-30	5000	1 unité de M	X	X	X	X
Transform $A \rightarrow B$	-50	2000	1 unité de A			X	
Transform $B \rightarrow A$	-50	2000	1 unité de B				X
Absorb A from North	-2	2000	La cellule doit contenir moins de 7 substrats		X	X	X
Absorb A from South	-2	2000			X	X	X
Absorb A from East	-2	2000			X	X	X
Absorb A from West	-2	2000			X	X	X
Absorb B from North	-2	2000	La cellule doit contenir moins de 7 substrats	X		X	X
Absorb B from South	-2	2000		X		X	X
Absorb B from East	-2	2000		X		X	X
Absorb B from West	-2	2000		X		X	X
Evacuate A from North	-0.5	3000	La cellule doit contenir au moins 1 unité de A		X	X	X
Evacuate A from South	-0.5	3000			X	X	X
Evacuate A from East	-0.5	3000			X	X	X
Evacuate A from West	-0.5	3000			X	X	X
Evacuate B from North	-0.5	3000	La cellule doit contenir au moins 1 unité de B	X		X	X
Evacuate B from South	-0.5	3000		X		X	X
Evacuate B from East	-0.5	3000		X		X	X
Evacuate B from West	-0.5	3000		X		X	X
Do Nothing	1	500	-	X	X	X	X

TABLE 5.7 – *Table des actions de l'organisme auto-alimenté. Tous les organes n'ont pas la possibilité de faire toutes les actions.*

- $M \rightarrow Energie$ produit de l'énergie vitale à partir de M,
- $A \rightarrow B + Energie$ produit de l'énergie vitale et un déchet B à partir d'un substrat A,
- $B \rightarrow A + Energie$ produit de l'énergie vitale et un déchet A à partir d'un substrat B.

Les actions disponibles pour l'organisme sont présentées dans la table 5.7. Toutes les actions ne sont pas disponibles pour tous les organes. Pour réduire la complexité de la recherche du génome, seules les actions nécessaires pour l'organe sont gardées. Par exemple, les organes producteurs-consommateurs n'ont besoin qu'une seule des deux transformations de substrat et les systèmes de transfert n'en ont pas besoin. Pour chaque action, un "X" dans la colonne de l'organe signifie que l'action est disponible pour cet organe.

Substrat du capteur	Position dans la cellule	Présence dans l'organe			
		TS_1	TS_2	PC_1	PC_2
M	North	X	X	X	X
M	South	X	X	X	X
M	East	X	X	X	X
M	West	X	X	X	X
A	North		X	X	X
A	South		X	X	X
A	East		X	X	X
A	West		X	X	X
B	North	X		X	X
B	South	X		X	X
B	East	X		X	X
B	West	X		X	X

TABLE 5.8 – *Table des capteurs de l'organisme auto-alimenté*

La table 5.8 donne la liste des capteurs pour les différents organes de la structure. Comme pour les actions, tous les capteurs ne sont pas disponibles pour tous les organes dans le but de réduire la complexité de la recherche du génome.

50 unités de substrat A sont positionnées près de l'organe PC_1 (coordonnées (8,5) dans l'environnement) et 50 unités de substrats B sont déposées près de l'organe PC_2 (coordonnées (13,14)). L'organe doit transformer le substrat A en B et doit positionner ce dernier substrat à l'entrée de TS_1 (coordonnées (6,7)) qui transfère ce substrat jusqu'à l'entrée de PC_2 (coordonnées (13,14)). PC_2 effectue le travail inverse de PC_1 : il transforme le substrat B en A, récupérant l'énergie de la transformation et déposant le substrat A, déchet de la réaction à l'entrée de TS_2 (coordonnées (15,12)), qui remonte ensuite le substrat A à l'entrée de PC_1 (coordonnées (8,5)). Etant donné que toutes ces actions produisent de l'énergie (coûts négatifs), la structure obtenue devrait théoriquement s'auto-alimenter et vivre indéfiniment.

Résultats

Pour produire cette structure, chaque organe est d'abord développé indépendamment, dans un environnement similaire. Lorsque l'algorithme génétique a trouvé les quatre organes indépendants, les quatre cellules initiales sont positionnées en même temps dans le même environnement et on lance la simulation. La figure 5.7 montre le développement et le comportement de l'organisme. Il est intéressant de noter que pour des organes similaires, différentes stratégies ont émergé afin d'obtenir la fonction désirée. C'est par exemple le cas pour les producteurs-consommateurs. Alors que l'organe PC_1 transfère d'abord le

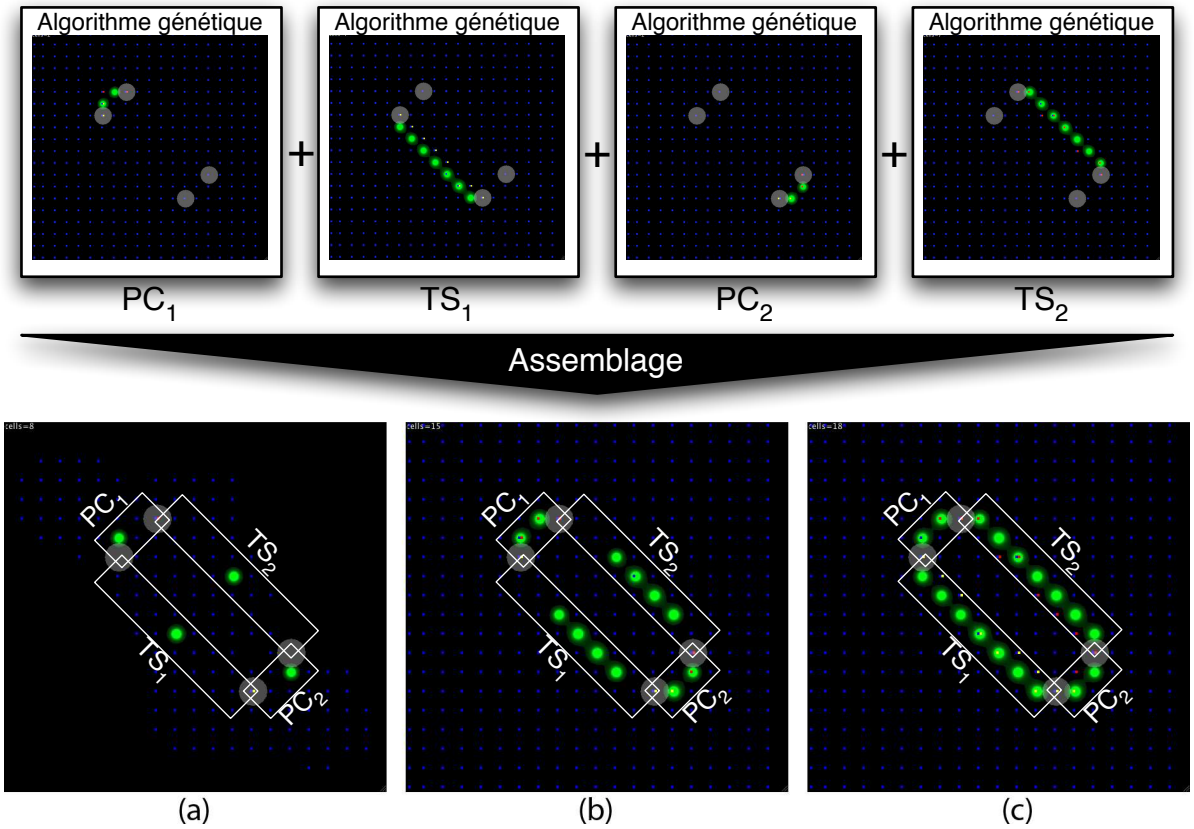


FIGURE 5.7 – *Résultat de la coopération des 4 différents organes. (a) Début de la simulation : 4 cellules qui contiennent le code génétique de chaque organe sont positionnées dans l'environnement. (b) Croissance de l'organe : alors que les 2 organes producteurs-consommateurs ont fini leur phase de développement et ont déjà commencé leur travail, les systèmes de transfert continuent leur croissance. (c) Tous les organes ont terminé leur développement et la structure auto-alimentée est en place. Alors que les organes producteurs-consommateurs continuent leur travail, les systèmes de transfert commencent le déplacement de substrat pour alimenter les autres organes.*

substrat A avant de le dégrader en énergie et placer le déchet là où il le fait, l'organe PC_2 fait le contraire. Il transforme le substrat à sa source et transfère le déchet à l'entrée du système de transfert.

L'organisme obtenu est proche de celui attendu. Le réseau d'optimisation régule correctement la longueur des systèmes de transfert et les organes producteurs-consommateurs développent différentes stratégies pour atteindre leur but. Le fonctionnement détaillé de chaque organe de l'organisme est donné par la figure 5.8. L'organe PC_1, en haut à gauche, transfère le substrat A (en rouge) à sa deuxième cellule avant de le transformer en B qui est rejeté à la bonne position. L'organe PC_2 adopte la stratégie opposée : il absorbe le substrat B, le transforme dans la première cellule et transfère le substrat A résultant de

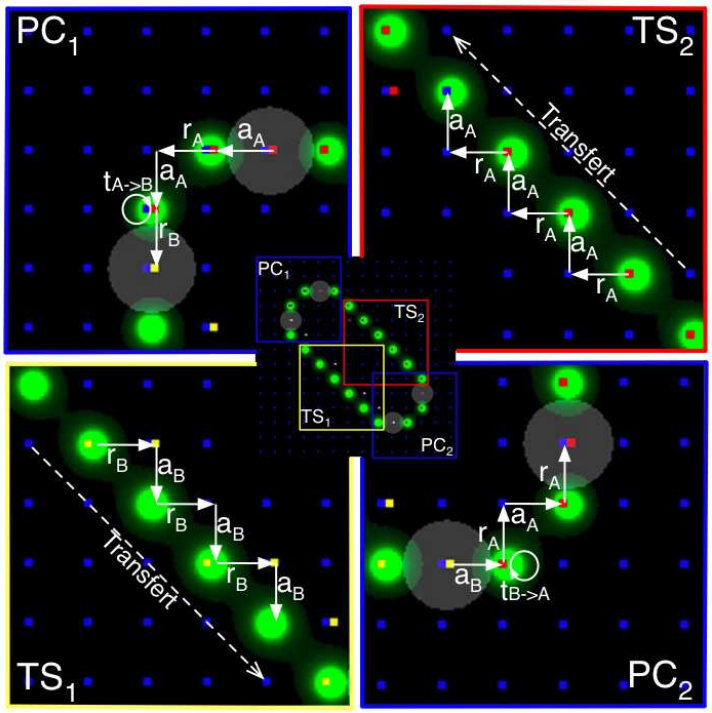


FIGURE 5.8 – *Détail du fonctionnement de l'organisme. Les acronymes sur les flèches correspondent aux actions effectuées par les cellules : r_A signifie "reject de A", a_B "absorb B" et $t_{A\to B}$ "transform A into B".*

la transformation à la position désirée. Pour plus de détails, les génomes des organes PC_1 et PC_2 se trouvent en Annexe C. Les organes TS_1 et TS_2 utilisent une série d'absorptions et de rejets afin de déplacer le substrat de la sortie d'un premier organe producteur-consommateur vers l'entrée du second.

Les courbes présentées dans la figure 5.9 montrent l'évolution du nombre de cellules et de la quantité de substrat M dans l'environnement. La quantité de M diminue fortement au début de la simulation, avant l'initialisation du cycle (phase 1). Les différents organes utilisent du substrat M pour produire un premier métabolisme. Une fois que le cycle d'auto-alimentation démarre (phase 2), les organes utilisent ce cycle comme métabolisme. Ils continuent cependant à consommer du substrat M au lieu d'utiliser l'action d'attente. Cette énergie est alors stockée pour une utilisation future.

La courbe présentée dans la figure 5.10 montre la proportion de substrats A et B. La quantité de substrat B diminue lentement. Ceci prouve une différence d'efficacité entre les organes : l'organe PC_1 met plus de temps à convertir A en B que l'organe PC_2 pour transformer B en A. Même si la différence est petite, la courbe montre que le cycle n'est pas infini : après une longue période de temps, le substrat B va venir à manquer et le cycle sera rompu.

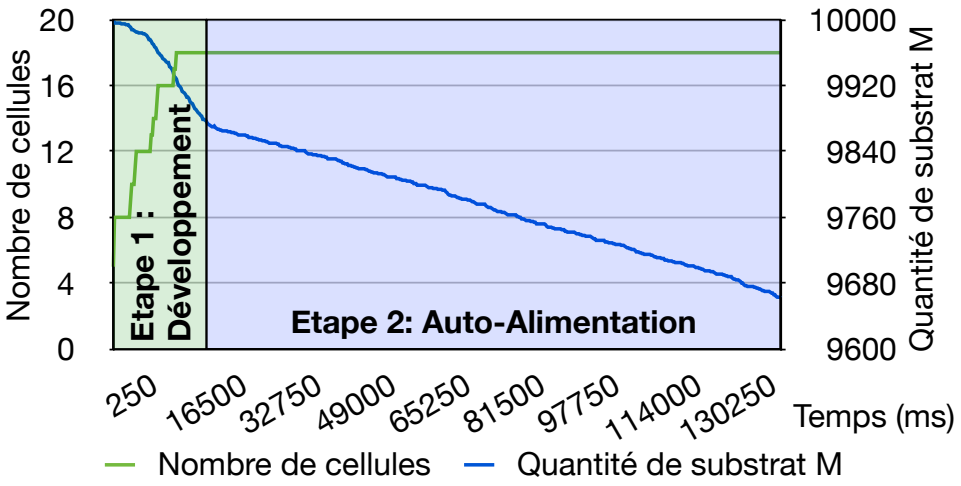


FIGURE 5.9 – *Nombre de cellules (axe des ordonnées à gauche) et quantité de substrat M (axe des ordonnées à droite) dans l'environnement en fonction du temps (axe des abscisses).*

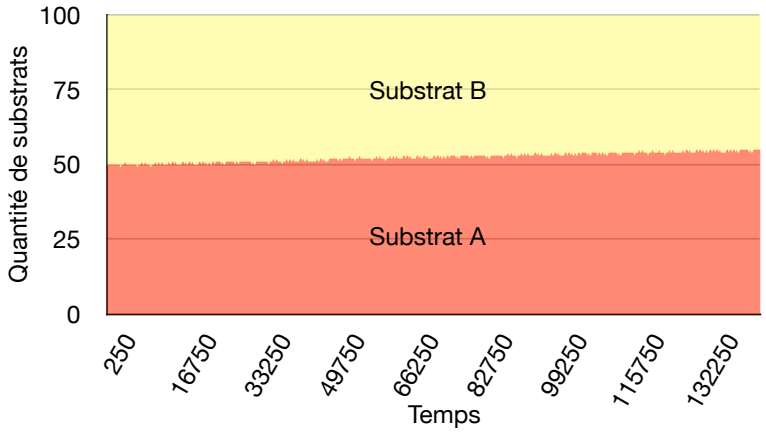


FIGURE 5.10 – *Evolution du rapport entre les quantités de substrat A et de substrat B au cours du temps. La quantité de substrat B diminue : l'organe PC_2 est plus efficace que l'organe PC_1.*

5.2 Expérimentation 4 : La génération de formes

5.2.1 Conditions expérimentales

Dans cette expérience présentée dans [Cussat-Blanc et al., 2008b], nous voulons générer des créatures prenant une forme décrite par l'utilisateur. Le but de cette expérimentation est de simuler la croissance de la morphologie de créatures similaires à celles de Karl Sims [Sims, 1994]. Elles pourront ensuite être plongées dans un simulateur physique et posséder un module comportemental de haut niveau leur permettant d'évoluer dans leur environnement.

Cinq substrats différents sont nécessaires à la génération de ces formes :

- Le substrat M qui donne de l'énergie aux cellules grâce à la transformation ($M \rightarrow (+30)$). Ce substrat se diffuse dans l'environnement.
- Quatre substrats différents servant de morphogènes, ici nommés NW, NE, SW and SE, désignent les directions de division des cellules. Ces substrats ne se diffusent pas dans l'environnement pour ne pas interagir avec le travail des cellules. La disposition des substrats donnera la forme globale de la créature.

Action	Coût énergétique	Durée	Exigences
Divide to NorthEast	30	10000	1 unité de M
Divide to NorthWest	30	10000	
Divide to SouthEast	30	10000	
Divide to SouthWest	30	10000	
Transform B into Energy	-30	5000	1 unité d'M
Absorb M from North	0.5	2000	La cellule doit contenir moins de 7 substrats
Absorb M from South	0.5	2000	
Absorb M from East	0.5	2000	
Absorb M from West	0.5	2000	
Evacuate M to North	0.5	2000	La cellule doit contenir au moins 1 unité d'M
Evacuate M to South	0.5	2000	
Evacuate M to East	0.5	2000	
Evacuate M to West	0.5	2000	
Apoptosis	30	7500	-
Do nothing	1	250	-

TABLE 5.9 – *Liste des actions possibles des cellules pour développer la forme.*

Quatre actions sont associées à ces substrats :

- la *division* qui consomme de l'énergie vitale et une unité de substrat M,
- la *transformation de M* qui permet à la cellule de déclencher sa transformation en

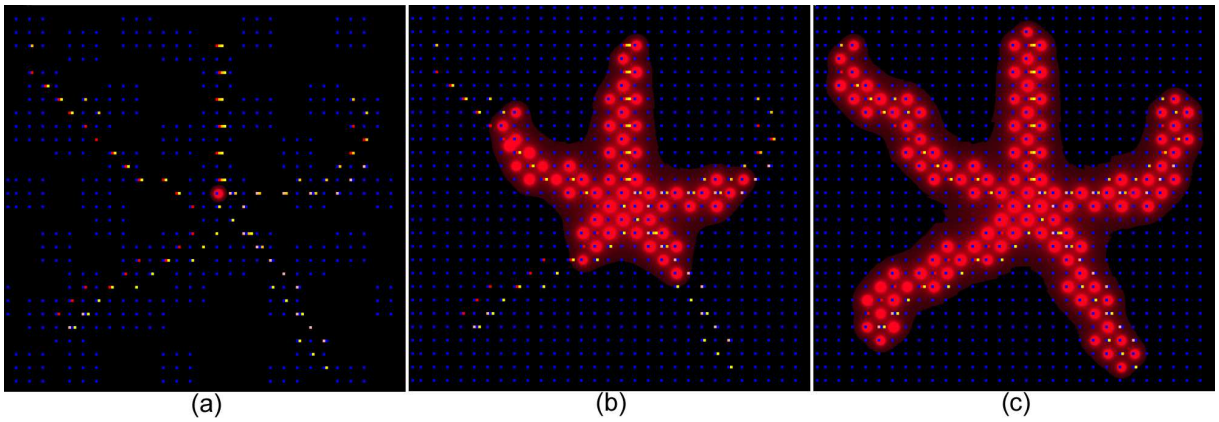


FIGURE 5.11 – *Croissance de l'étoile de mer. (a) Début de la simulation. (b) L'étoile de mer se développe en suivant les morphogènes. (c) L'organisme arrête de se développer quand la forme désirée est obtenue.*

énergie vitale,

- l'*absorption de M* qui permet à la cellule d'ingérer une unité de M (consomme de l'énergie vitale),
- l'*apoptose* qui permet à une cellule de s'autodétruire si elle le désire. Cette action pourra être utilisée si le développement de l'organisme ne respecte pas la forme qui lui est demandée.

Le tableau 5.9 donne les paramètres de chaque action.

Pour obtenir la morphologie désirée, la fonction d'évaluation fit de l'algorithme génétique est calculée à la fin d'un temps de simulation constant (équivalent à 30 secondes réelles en accélérant la simulation 100 fois) et est donnée par la formule suivante :

$$fit = 2card(c \in S) - card(c \notin S)$$

avec c, une cellule de l'organisme et S la forme désirée.

5.2.2 Détail de l'organisme obtenu

La forme que nous allons utiliser pour l'apprentissage du développement de la morphologie ressemble à une étoile de mer. Pour ce faire, nous avons placé les morphogènes selon la forme désirée dans un environnement de taille 30x30. La cellule est ensuite placée au centre de l'environnement, aux coordonnées (16,14).

La figure 5.11 montre le résultat de la meilleure créature produite par l'algorithme génétique. Nous pouvons observer que la forme désirée est obtenue. Il est intéressant d'étudier le système de sélection d'action produit par l'algorithme génétique :

$$
\begin{aligned}
(SensorNE = 1) &\rightarrow (DuplicateNE)\ (6)\\
(SensorNW = 1) &\rightarrow (DuplicateNW)\ (5)\\
(SensorSE = 1) &\rightarrow (DuplicateSE)\ (4)\\
(SensorSW = 1) &\rightarrow (DuplicateSW)\ (3)\\
&\rightarrow (TransformM)\ (2)\\
(SensorM = 1) &\rightarrow (AborbM)\ (1)\\
&\rightarrow (DoNothing)\ (0)
\end{aligned}
$$

Ce système de sélection montre que l'algorithme génétique utilise correctement les informations données par l'environnement pour suivre le schéma de croissance donné par l'utilisateur. De plus, les actions de division cellulaire sont toujours prioritaires sur les autres actions pour ne pas accumuler l'énergie inutilement. La dernière remarque que nous pouvons faire est que l'action d'apoptose n'est pas utilisée durant la croissance. L'organisme suppose que les morphogènes donnés par l'utilisateur sont correctement positionnés.

5.2.3 Expérimentation 5 : Plasticité phénotypique

En biologie, la plasticité phénotypique est la capacité d'un organisme à s'adapter aux modifications de l'environnement [Larsen, 2004]. Ce phénomène émerge du processus de développement. L'adaptation de l'organisme vient de la capacité de l'organisme à modifier sa morphologie et son comportement en fonction des stimuli de son environnement proche. Cette plasticité permet la production d'organismes différents avec un génome unique.

Tufte présente dans [Tufte and Haddow, 2007, Tufte, 2008] un modèle de développement mettant en scène la plasticité phénotypique. Il utilise un réseau de régulation génétique basé sur un ensemble de règles. Il développe ainsi différents organismes en plongeant un génome unique dans différents environnements. Le but de l'organisme est d'augmenter le nombre de cellules actives à chaque pas de simulation. De ce fait, l'organisme produit un compteur. La plasticité issue du modèle est montrée en partant de ce que l'auteur appelle un "environnement initial". L'ajout de données sur l'environnement dans le génome de l'organisme aide l'organisme à maintenir sa fonction si l'environnement change durant son développement.

Dans notre modèle, les génomes produits possèdent aussi cette propriété de plasticité

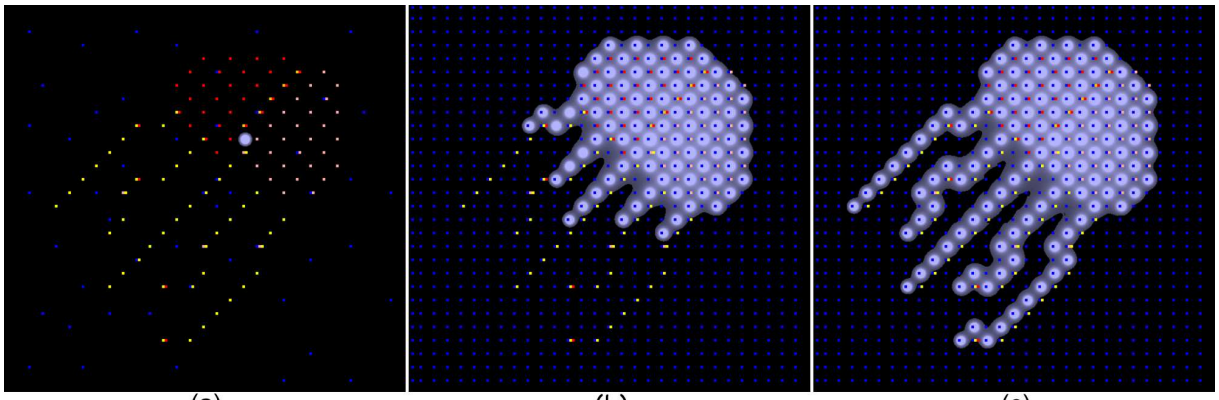


FIGURE 5.12 – *Croissance de la méduse. (a) Début de la simulation. (b) La méduse se développe en suivant les morphogènes. (c) L'organisme arrête de se développer quand la forme désirée est obtenue.*

phénotypique [Cussat-Blanc et al., 2009a]. On peut s'en rendre compte en étudiant le système de sélection de l'action obtenu pour l'étoile de mer. En observant ces règles, nous pouvons remarquer qu'il est possible de produire n'importe quelle forme à partir de ce génome. En effet, les règles découvertes par l'organisme permettent de répondre à n'importe quelle configuration de morphogènes. Pour vérifier cette hypothèse, nous avons développé un second organisme : une méduse. Pour cela, nous utilisons exactement le même environnement, avec les mêmes substrats et les mêmes actions. Seule la disposition des morphogènes dans l'environnement est modifiée. En utilisant le génome de l'étoile de mer, nous lançons une simulation et nous obtenons la créature représentée par la figure 5.12.

Ce système de sélection d'action nous indique aussi que la forme produite sera toujours la même quel que soit le positionnement de la cellule initiale (qui doit tout de même se situer à l'intérieur de la forme désirée).

La possibilité de produire différentes formes de créatures en utilisant un génome unique répond à la définition de la plasticité phénotypique. Quelle que soit la configuration des morphogènes dans l'environnement, l'organisme s'adaptera pour produire la forme désirée.

Une autre preuve de plasticité phénotypique est la capacité d'un organisme à s'auto-réparer en cas de blessure. En effet, une blessure infligée à un organisme par son environnement (par une autre créature, par un substrat agissant comme un poison, etc.) oblige l'organisme à modifier son comportement afin de continuer à réaliser la fonction pour laquelle il a été créé. Les capacités de réparation de notre modèle sont décrites dans la partie suivante à l'aide de deux expériences : la régénération de forme et la récupération de fonction.

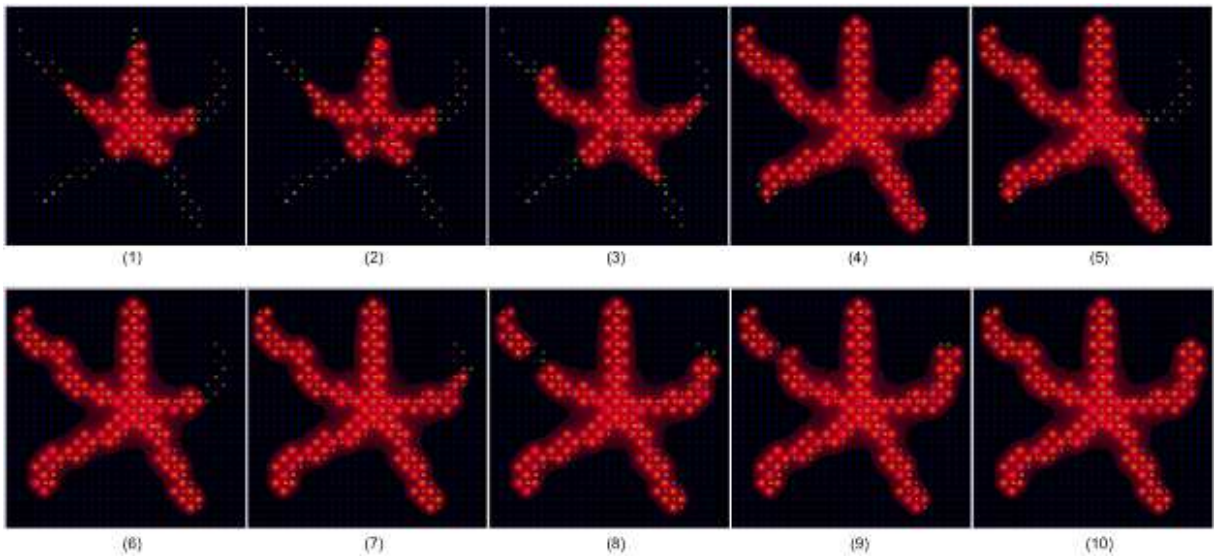


FIGURE 5.13 – *Capacité d'auto-réparation de l'étoile de mer. (1) Avant d'infliger une blessure dans le milieu, la créature est en phase de développement. (2-4) Régénération du trou au centre de la créature. (5-10) Régénération de la branche droite de l'étoile de mer. (8-10) Régénération simultanée de la branche droite et du trou infligé dans la branche gauche.*

5.3 Propriété d'auto-réparation

5.3.1 Expérimentation 6 : Régénération de formes

Pour montrer les capacités d'auto-réparation de notre modèle, nous allons infliger aux créatures différentes blessures durant leur développement et leur phase fonctionnelle [Cussat-Blanc et al., 2009a]. La première créature à être testée est l'étoile de mer. Durant son développement, nous allons agir sur la créature de trois façons différentes (la figure 5.13 illustre cette expérimentation) :

- en créant un trou dans le centre de la créature,
- en amputant la branche gauche de la créature,
- en faisant un trou dans la branche droite de l'étoile.

Dans ces trois expérimentations, l'organisme répond convenablement et reconstruit la forme désirée. Dans le premier cas, l'organisme met du temps pour combler le trou. Les cellules servant à combler le trou sont positionnées trop rapidement par la cellule mère. Ainsi, les cellules filles meurent rapidement par manque d'énergie vitale pour initialiser leur processus métabolique. Il faut attendre que la cellule mère ait suffisamment d'énergie pour produire une cellule fille viable avant que le trou ne se comble. La reconstruction de la branche droite semble pour sa part plus aisée. Dès que la branche est amputée,

de nouvelles cellules apparaissent rapidement et reconstruisent la branche sans problème. La branche reconstruite est exactement la même que la précédente. L'organisme a donc parfaitement régénéré la partie manquante. Il en est de même lors de la création d'un trou dans la branche gauche de l'organisme. Comme lors de l'amputation de la branche droite, la régénération est très rapide et la branche est complètement reconstituée.

Ces trois expérimentations montrent les capacités du modèle à regénérer la forme des organismes lorsque ces derniers sont altérés par l'environnement. La prochaine partie va montrer les capacités de récupération de fonction de nos organismes.

5.3.2 Expérimentation 7 : Récupération de la fonction : le système de transfert

Expérimentation 7a : Destruction contrôlée de cellules

Dans le but d'évaluer les capacités de récupération de fonction de notre modèle, nous avons imaginé deux expériences effectuées sur le système de tranfert présenté dans la partie 5.1.2 [Cussat-Blanc et al., 2009a].

Tout d'abord, nous allons faire un trou de trois cellules dans le centre de la structure durant sa phase de développement. Les cellules du voisinage du trou régénèrent la structure du système de tranfert et la fonction de l'organe est reconstituée : le système, et principalement les cellules régénérées, est toujours capable de déplacer des substrats comme avant la blessure.

La seconde expérience consiste à détruire les mêmes trois cellules mais, cette fois, du-

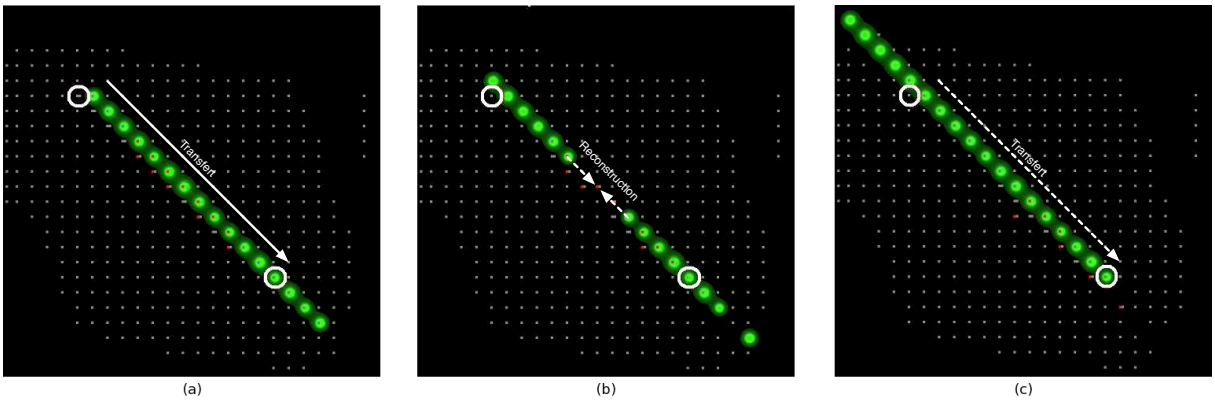


FIGURE 5.14 – *Récupération de la fonction du système de transfert. (a) Avant la blessure de l'organe. (b) Résultat de la blessure : le système de transfert est coupé en son centre et commence à se reconstruire. (c) Continuation du transfert de substrat après la régénération de la structure : la fonction de l'organe est récupérée.*

rant le transfert lui même. Cela va ainsi produire une "fuite" dans le transfert. L'expérience est illustrée par la figure 5.14. Le trou créé arrête le transfert. Comme dans l'expérience précédente, les cellules voisines se divisent pour remplir le trou. Dès que les nouvelles cellules sont créées, le transfert reprend graduellement, à chaque fois qu'une nouvelle cellule est produite. Le résultat du transfert est très proche du résultat obtenu par un système de tranfert n'ayant subi aucune blessure.

La figure 5.15 montre la courbe du niveau d'énergie du système de tranfert (plus précisemment, la somme des niveaux d'énergie des cellules de l'environnement) durant la simulation. La courbe rouge, en haut sur le graphique, correspond au niveau d'énergie d'un organe normal, n'ayant pas subi de blessure. La seconde, en bleu est celle d'un organe ayant subi les deux blessures décrites précédemment. Les éclairs sur la courbe montrent les moments où les blessures sont infligées à l'organe. Nous pouvons observer que le niveau d'énergie du système décroît fortement lors des blessures. Ceci est dû à la mort de 3 cellules. Il est aussi intéressant de noter que la courbe d'énergie est globalement inférieure à la courbe de l'organe normal. L'organe blessé ne se remet pas complètement de la blessure et possède un niveau d'énergie inférieur à l'organe normal. La crête de la courbe est aussi différente : l'organe blessé semble être moins stable et son fonctionnement semble être plus difficile à maintenir. Enfin, la durée de vie de l'organisme est plus courte que celle de l'organe normal. Cette durée de vie peut être importante car elle pourrait permettre de transférer plus ou moins de substrat en cas de nouvelle arrivée de substrat.

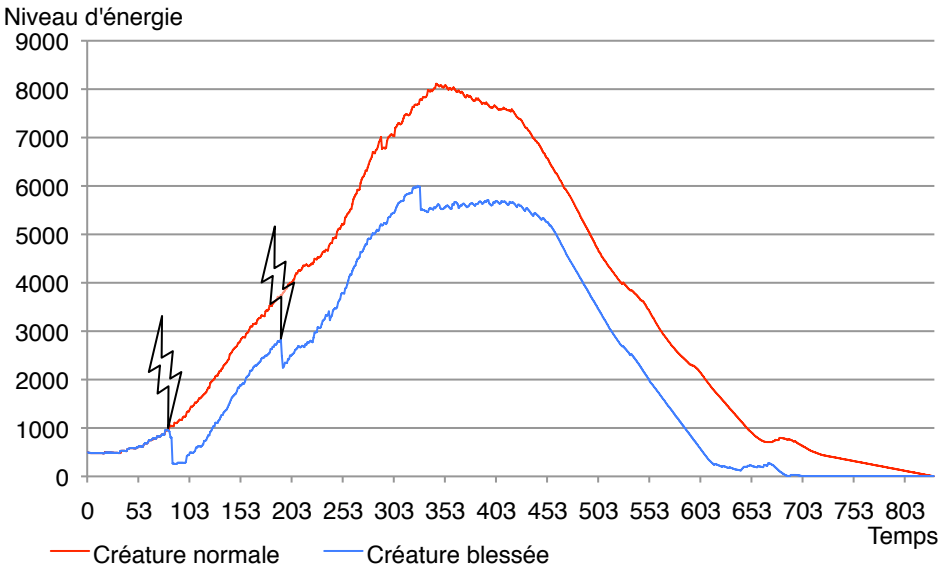


FIGURE 5.15 – *Courbes des niveaux d'énergie de l'organe sain et de l'organe blessé.*

Expérimentation 7b : Blessure continue

Nous voulons maintenant étudier la réaction de l'organisme face à un stress continu. Pour ce faire, en utilisant le même organisme que précédemment, nous allons tuer périodiquement une cellule choisie aléatoirement. Une cellule sera tuée toute les x millisecondes, avec $x \in [5, 2000]$. Dans le but de donner à l'organisme la possibilité de se régénérer, la dernière cellule dans l'environnement ne sera jamais tuée.

Pour chaque valeur de x, 50 simulations sont lancées. Dans cette expérience, la fonction d'évaluation varie de 0 à 3130 (elle est inversement proportionnelle à la somme des carrés des distances des substrats à leur but) et le but est de maximiser cette évaluation. Le résultat des notes obtenues par les organismes est présenté dans la figure 5.16. Les boîtes à moustaches représentent les valeurs de la fonction d'évaluation (en ordonnée) pour chaque valeur de x (en abscisse). Ils représentent :

- l'*évaluation minimum* en bas de la ligne,
- l'*évaluation maximum* en haut de la ligne,
- le *premier et le troisième quartile* en bas et en haut de la boîte,
- la *médiane* au centre de la boîte et représentée grâce à la courbe.

L'organe répond globalement bien au stress qui lui est infligé : le nombre de cellules tuées doit être important avant que l'organe arrête de répondre à sa fonction. La courbe (a) montre la réponse globale de l'organisme alors que la courbe (b) se concentre sur l'effondrement de la courbe (les écarts de temps entre 5 et 300 millisecondes). La dégradation de l'évaluation a lieu lorsque la fréquence de destruction des cellules est importante, lorsqu'on tue environ une cellule toutes les 200 millisecondes, pour une durée moyenne d'expérience de 7192 millisecondes. Cela correspond à environ 15 cellules tuées aléatoirement durant la simulation dans un organisme composé d'environ 16 cellules. Pour les plus petites fréquences (valeurs plus grandes de x), l'organisme continue à se développer et à répondre à sa fonction très correctement, voire même parfaitement. Les écarts de valeurs de la fonction d'évaluation diminuent fortement avec la diminution de la fréquence du stress. Pour les plus grandes fréquences (plus petites valeurs de x), l'organisme ne peut pas se développer et ne peut ainsi pas répondre à sa fonction.

5.4 Vers la construction d'un ensemble de simulateurs

Pour compléter le modèle de croissance que nous avons présenté, nous avons commencé à développer un simulateur physique et un simulateur hydrodynamique dans lesquels les

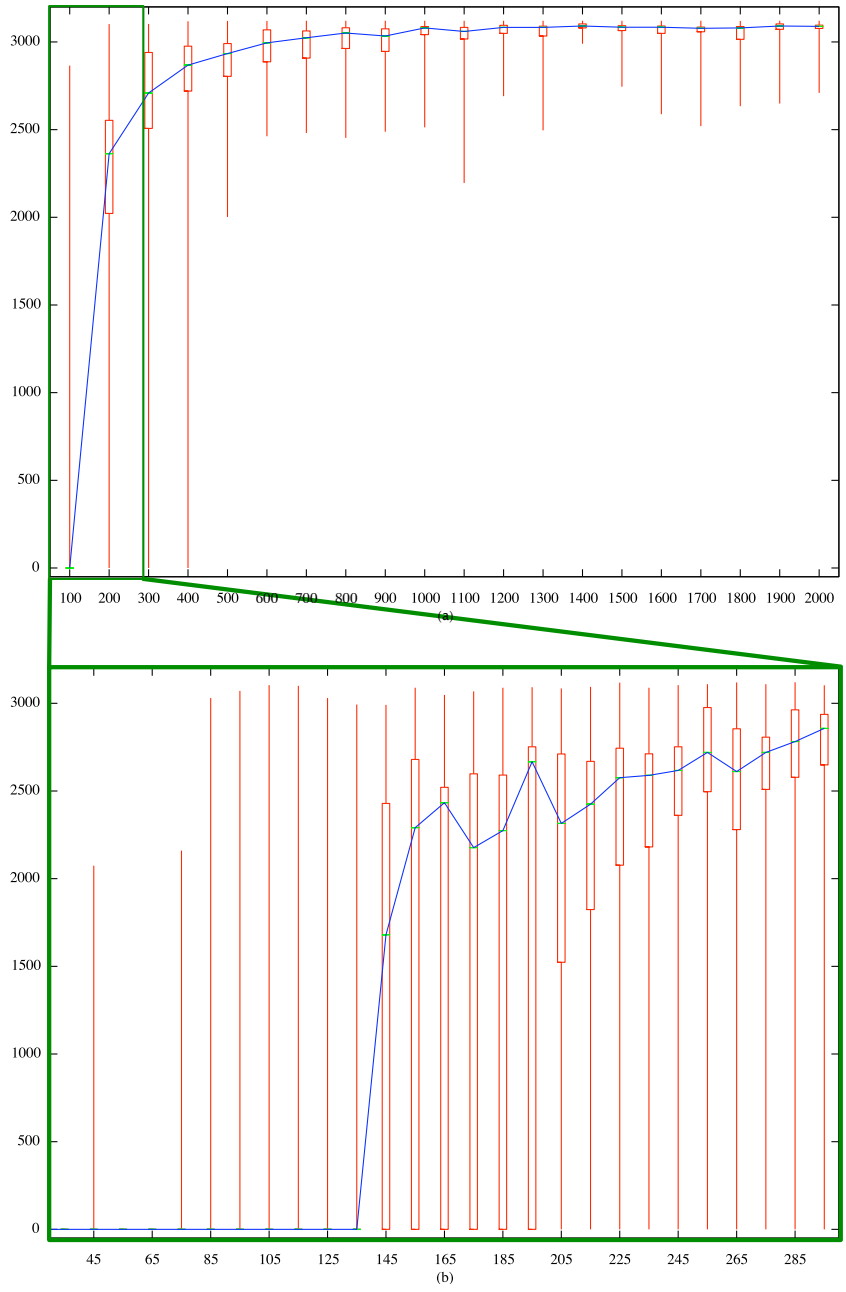


FIGURE 5.16 – *Valeurs de la fonction d'évaluation (ordonnées) du système de transfert blessé. Les boîtes à moustaches représentent le minimum, le premier quartile, la médiane, le troisième quartile et le maximum des valeurs des évaluations pour les 50 tests, pour chaque pas de temps (l'abscisse représente le temps en millisecondes entre deux suppressions de cellule). (a) Courbe globale. (b) Zoom sur l'effondrement de la courbe, entre 5 et 300 millisecondes avec un pas de temps de 10 millisecondes..*

créatures pourront être plongées dans le but d'accroître leur capacité et de pouvoir simuler le comportement des créatures à différents niveaux. Même si ces travaux sont encore en

cours, les premiers résultats obtenus, présentés dans cette section, sont encourageants.

L'architecture basée sur le paradigme de programmation des listeners de notre modèle permet l'ajout facile de nouvelles fonctionnalités. Notre idée est de créer un ensemble de simulateurs dans lequel les cellules évolueront. Le modèle que nous avons présenté jusqu'ici permet un développement chimique des organismes, permettant particulièrement le développement d'un métabolisme et de capacités d'auto-réparation.

Afin de compléter cette approche, nous avons décidé d'ajouter un simulateur physique et un simulateur hydrodynamique à notre simulateur chimique. Ces simulateurs sont encore en cours de développement mais quelques résultats encourageants nous laissent à penser que certaines limites du modèle chimique pourront être dépassées.

La suite de cette section décrit brièvement les deux simulateurs et leur intégration au modèle. Elle donne aussi les premières images des résultats des expérimentations préliminaires.

5.4.1 Le simulateur physique

Quelques simulateurs physiques existants

Etant donnée la complexité de développement d'un simulateur physique et la qualité de certains simulateurs actuels, nous avons préféré utiliser un simulateur physique existant.

Un certain nombre de simulateurs physiques existe avec leurs défauts et leurs qualités. Adrain Boeing et Thomas Bräunl présentent dans [Boeing and Bräunl, 2007] un très bon état de l'art comparatif des différents moteurs physiques existants, tous gratuits. Ils étudient les différents aspects des moteurs tels que le réalisme, la précision, la complexité combinatoire des principaux algorithmes propres à la simulation physique (détection de collision, joints, propriétés des matériaux, etc). Dans leur article, ils testent sept simulateurs : AGEIA PhysX (anciennement Novodex), Bullet, JigLib, Newton, Open Dynamics Engine (ODE), Tokamak et True Axis.

Les auteurs en concluent que chaque moteur possède un domaine de prédilection particulier dans lequel il surpasse tous les autres. Bullet est cependant le seul moteur qui a de bons résultats dans la majorité des tests et dont les capacités dépassent certains moteurs commerciaux. De plus, ce moteur consomme très peu de ressources processeur : l'augmentation du temps de résolution de contraintes reste linéaire avec l'augmentation du nombre de ces contraintes. Un module PhysX^{TM} by NVIDIA [22] est aussi en cours de développement dans Bullet et permettra une grande amélioration des performances quand

22. technologie NVidia permettant d'accélérer les calculs physiques directement sur les cartes graphiques compatibles.

il sera complètement opérationnel. Enfin, une autre propriété importante de ce modèle est le développement parallèle d'une version C++ et d'une version Java du moteur, ce qui est une caractéristique rare dans les moteurs physiques. La version Java nous intéresse plus particulièrement étant donné que notre modèle est lui-même implémenté dans ce langage de programmation. Nous avons donc décidé d'utiliser ce moteur physique pour plonger nos cellules dans un modèle physique virtuel.

Le rendu graphique 3D est donné en utilisant un moteur de rendu OpenGL. Il permet l'affichage direct des objets du moteur physique Bullet. En effet, Bullet possède un mécanisme permettant d'afficher les objets directement. Leur forme est définie par l'utilisateur et est utilisée par Bullet pour donner le rendu au moteur graphique.

Intégration

Le paradigme de programmation des listeners utilisé pour implanter notre modèle de développement cellulaire nous permet une communication aisée entre les différents éléments de la simulation. Ainsi, l'intégration du moteur physique ne nécessite aucune modification du code du modèle chimique. Le moteur est seulement enregistré au niveau des listeners lors de l'initialisation de la simulation. Il sera alors informé de toutes les actions des cellules et de toutes les modifications de substrat dans l'environnement.

Les cellules sont représentées par des différentes formes 3D collées les unes aux autres en utilisant les joints du moteur physique. Des contraintes physiques pourront ainsi être appliquées aux cellules et aux liaisons cellulaires. Si elles sont trop fortes, le moteur physique pourra détruire une cellule via l'environnement en lui demandant de la tuer dans le monde chimique. Le moteur physique est ensuite informé d'une mort cellulaire et peut la détruire à son tour dans le monde physique.

Expérimentation 8 (préliminaire) : La fibre musculaire

Cette première expérience a pour but de montrer les capacités de l'utilisation d'un moteur physique en parallèle de notre moteur chimique. Dans cette expérimentation, nous avons utilisé le génome de développement de forme présenté dans le chapitre 5.2 pour développer un organisme possédant une "rotule" centrale (une cellule), deux "os" sur le coté (environ 400 cellules chacun) et des "fibres musculaires" les reliant (environ 100 cellules). Dans un premier temps, le but de l'expérimentation étant de montrer les possibilités de l'utilisation d'un modèle physique pour notre modèle, la spécialisation des cellules en rotule, en os ou en fibre se fait de manière manuelle : une matrice de spécialisation donne le type des cellules en fonction de leurs coordonnées.

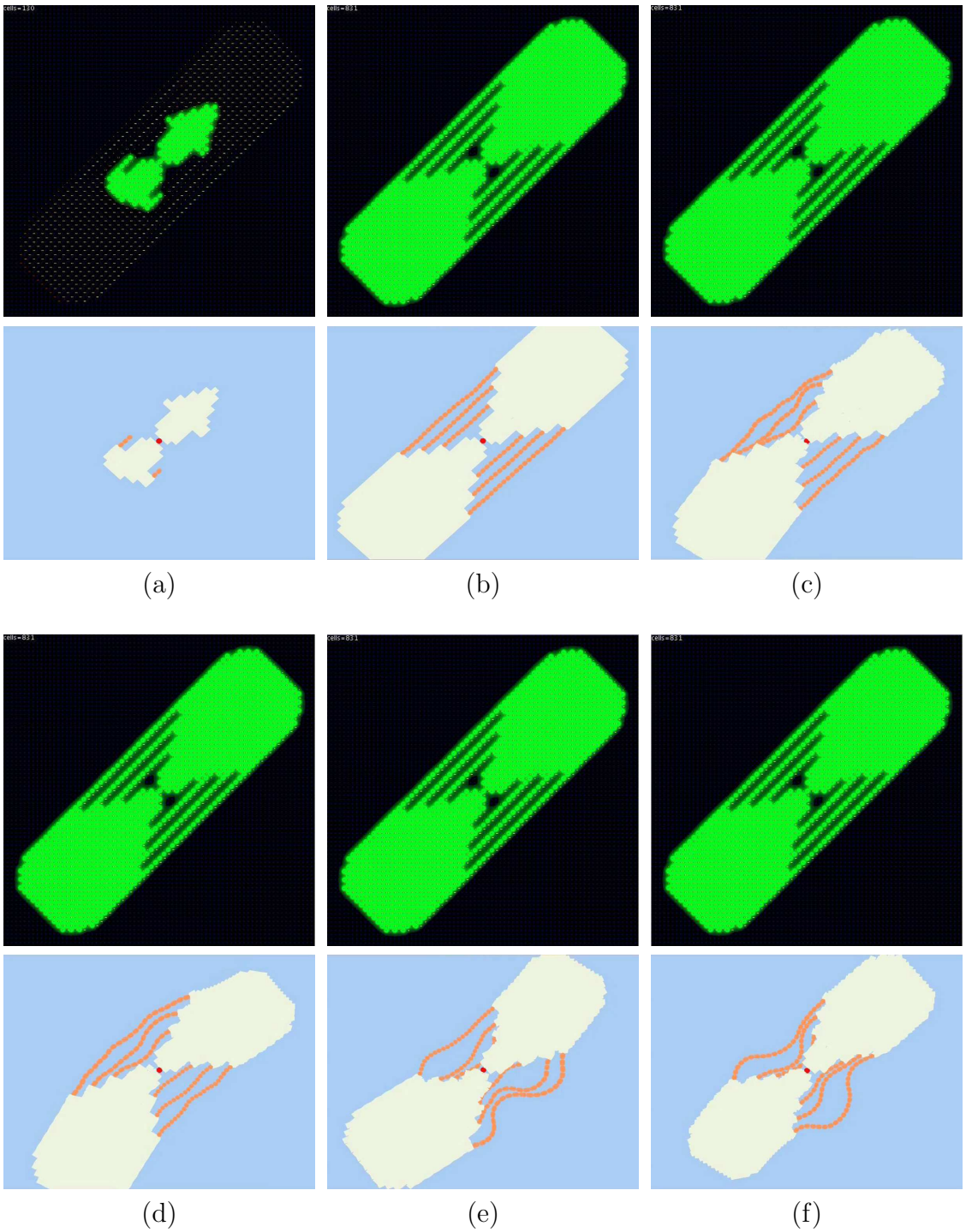


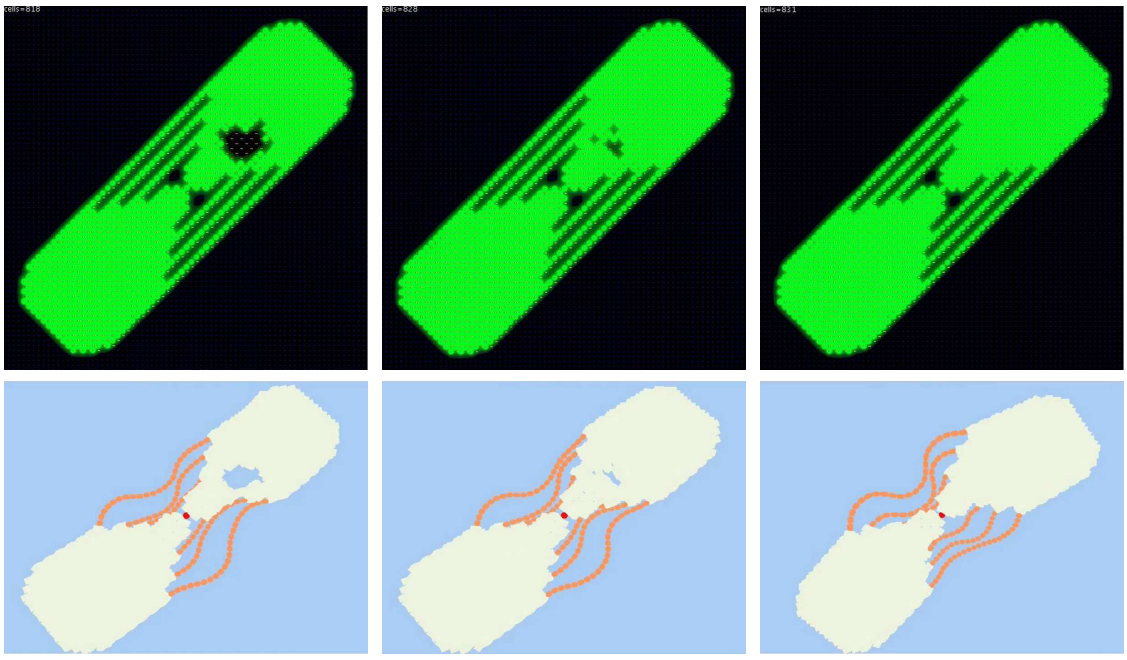
FIGURE 5.17 – *Développement d'une créature dans un monde chimique (en haut) et dans un monde physique (en bas) en parallèle. (a) et (b) Développement de la forme de la créature dans les deux mondes. (c) et (d) Rétrécissement des fibres musculaires inférieures, élargissement des fibres supérieures et torsion de la structure dans le monde physique. (e) Modification structurelle inverse et torsion inverse dans le monde physique. (f) Retour à la position initiale dans le monde physique.*

La forme et les fonctions des cellules varient selon leur type :

- une cellule "rotule" est sphérique et permet un pivotement des cellules de son voisinage,
- une cellule "osseuse" est représentée par un cube et est fermement accrochée à ses voisines,
- une cellule "musculaire" est représentée par une capsule et est capable de modifier sa forme en s'allongeant ou en se contractant.

Bien que non biologiquement plausible, la modification de la forme des cellules musculaires permet un allongement ou un rétrécissement général de la fibre musculaire. On peut ainsi obtenir une rotation des deux os autour de l'axe donné par la rotule présentée par les images de la figure 5.17. En cas de fortes contraintes appliquées sur les cellules, celles-ci peuvent être détruites par le moteur physique et reconstruites par le modèle chimique comme nous l'avons vu dans la section 5.3. La figure 5.18 montre ce phénomène.

Le moteur physique permet donc l'ajout de fonctionnalités qui peuvent être intéressantes dans notre modèle. Il permet aussi un passage à l'échelle pour produire des créatures qui pourront interagir dans un monde virtuel avec d'autres créatures ou bien un humain



(a) (b) (c)

FIGURE 5.18 – *Reconstruction d'une créature dans un monde chimique (en haut) et dans un monde physique (en bas) en parallèle. (a) Destruction de cellules par l'utilisateur en tirant sur la structure. (b) Reconstruction de la créature dans le monde chimique et implication dans le monde physique. (c) Retour à la normale de la créature.*

qui se déplace dans ce monde à l'aide d'un avatar. Ce moteur est cependant actuellement difficile à paramétrer : il est nécessaire de définir toutes les forces d'adhésion entre les cellules afin d'obtenir des résultats réalistes.

5.4.2 Le simulateur hydrodynamique

Intérêt de la simulation hydrodynamique

L'hydrodynamique a été formellement décrite à partir de 1750. En 1822, Navier donna une équation générale permettant de décrire précisément les déplacements d'un fluide. Quelques années plus tard, Stokes améliorait ces équations, donnant naissance aux fameuses équations de Navier et Stokes. Cependant, ces équations, très complexes à résoudre, ne s'appliquent qu'à peu de cas simples. En effet, leur résolution nécessairement analytique demande un temps de calcul encore trop important.

L'un des premiers modèles a été présenté par Hardy *et al.* dans [Hardy et al., 1976]. Nommé HPP, ce modèle simple consiste à discrétiser le milieu en le découpant en un maillage rectangulaire. Le fluide est représenté par des particules et leur déplacement se fait en appliquant des règles de collision à l'intersection de chaque maille. Ce modèle trop simple ne permet pas de simuler de manière réaliste les phénomènes hydrodynamiques. En 1986, Frisch et ses collègues ont amélioré ce modèle en transformant les mailles rectangulaires en mailles triangulaires. Dans leur modèle nommé FHP [Frisch et al., 1986], les particules de fluide ont ainsi six directions de dispersion possibles lors d'une collision.

Ces dernières techniques permettent d'étudier les écoulements des fluides mais ne permettent pas d'observer l'évolution d'une quantité de matière. Dans le but de remédier à ce problème, la méthode de Boltzmann consiste à résoudre les équations de Boltzmann en chaque point du maillage. Cette équation décrit la dynamique d'un gaz parfait. Cette méthode permet particulièrement l'étude de l'écoulement d'un fluide autour d'une forme complexe.

Développé par Jonathan Pascalie [Pascalie, 2009], ce simulateur permet de montrer les interactions hydrodynamiques des substrats de notre modèle. Le but de ce simulateur est de simuler, de manière simplifiée, les mécanismes de la phase de gastrulation qui a lieu durant le développement d'un certain nombre d'êtres vivants. Cette phase permet le placement des morphogènes dans l'embryon et permettra ensuite le développement de ses organes. En utilisant un simulateur hydrodynamique dans notre modèle de développement, on peut ainsi observer l'apparition de courants dans l'environnement, correspondant aux courants créés par l'organisme lorsque celui-ci absorbe ou rejette du substrat dans l'environnement. Ainsi, une cellule peut par exemple expulser un substrat avec une certaine

force dans le but de modifier l'environnement à une certaine distance.

Le modèle choisi

Au vu des contraintes de coûts de calcul induits par la complexité des simulateurs hydrodynamiques, nous avons décidé d'utiliser une méthode tentant de réduire au maximum la consommation de ressources systèmes tout en restant réalistes. Nous avons ainsi choisi d'implanter le solveur de Jos Stam [Stam, 1997, Stam, 2003]. Ce modèle est principalement utilisé dans l'informatique graphique. Ce modèle est intéressant car il est prouvé qu'il résout parfaitement les équations de Navier et Stokes pour un fluide incompressible. L'implantation de ce modèle remplacera complètement l'algorithme de diffusion du modèle de développement que nous avons vu au chapitre 3.1.3.

Dans ce modèle, l'environnement est représenté par une grille sur laquelle les particules de fluides se déplacent en suivant des vecteurs vitesses. Pour l'intégration à notre modèle de développement cellulaire, les cellules de l'environnement seront considérées comme des obstacles infranchissables. Lorsqu'une particule frappe la membrane d'une cellule, le vecteur vitesse correspondant au point de la grille sur laquelle elle se trouve est modifié afin que la particule suive une direction parallèle au bord de la cellule. Dans un premier temps, afin de simplifier la simulation, les différents substrats seront diffusés indépendamment les uns des autres. Aucune interaction entre les substrats ne sera considérée.

Une des limitations principales de ce modèle est la non conservation de la quantité de matière. En effet, durant la simulation, ce simulateur peut engendrer une faible perte de matière qui n'est pas acceptable pour notre modèle de développement. Le but de ce simulateur hydrodynamique est principalement de diffuser du morphogène dans l'environnement dans le but de développer une créature l'utilisant pour former sa morphologie. Une telle perte de matière pourrait alors engendrer un développement non désiré de la créature. Différentes méthodes existent actuellement afin de corriger ce problème. La première consiste à implanter les lois de conservation de l'énergie afin d'équilibrer les pertes de substrat dues à la simplification des équations. Cependant, cette méthode est très coûteuse en ressources de calcul et ne rentre donc pas dans notre cahier des charges. Dans un premier temps, nous avons donc préféré corriger les pertes de matière de façon proportionnelle en calculant la perte de matière globale du système et en répartissant cette perte de manière proportionnelle sur toute la grille.

Afin d'affiner les conditions aux bords, il faut doubler la taille de la grille du simulateur hydrodynamique par rapport à la grille de l'environnement chimique. En effet, plus la subdivision est petite, plus les conditions aux bords pourront être calculées précisément en décrivant le courant du fluide plus finement. Cependant, cette subdivision augmente

la complexité du calcul. Pour prendre en compte la diffusion entre cellules comme décrite dans le chapitre 3.1.3, il a fallu mettre en place un mécanisme le permettant. En effet, la méthode précédente ne permet pas ce passage de fluide, les obstacles représentant les cellules étant supposés coller les uns aux autres. Ainsi, les vecteurs vitesses extérieurs à la membrane de l'organisme peuvent exciter les vecteurs vitesses internes et créer des flux internes.

Les cellules interagissent avec l'environnement, notamment grâce à l'absorption et au rejet de substrat, ce qui crée de nouveaux flux dans l'environnement. Ainsi, alors que précédemment une cellule déposait un substrat dans son voisinage, nous pouvons maintenant simuler une force d'expulsion d'un substrat. Une cellule peut donc éjecter un substrat avec une certaine force et dans une certaine direction pour le positionner à un endroit bien précis dans l'environnement. Le simulateur calculera par la même occasion les flux créés par cette expulsion. L'absorption crée de même une dépression qui entraînera aussi différents flux. Enfin, la division modifie aussi l'environnement en créant un courant dans le sens de division afin de vider de substrat la position de la nouvelle cellule.

Expérimentation 9 (préliminaire) : Création de flux hydrodynamiques

Afin de valider notre simulateur hydrodynamique et particulièrement la cohérence de son rendu visuel, nous avons effectué quelques tests. Dans ces tests, aucune méthode de recherche évolutionniste n'a été utilisée. Les génomes de cellules ont été codés à la main car ils sont souvent très simples. L'émergence d'un comportement par l'utilisation de ce simulateur est un point qui reste encore à étudier même si cela n'est pas difficile à mettre en oeuvre.

La première expérimentation consiste à tester l'évacuation des substrats durant la phase de croissance d'un organisme. Pour cela, nous avons utilisé notre génome de développement de forme présenté dans la section 5.2 en le plaçant dans un environnement saturé en eau. Les morphogènes sont placés par l'utilisateur dans le but de produire la forme qu'il désire et ne seront pas affectés par les flux produits par les divisions cellulaires (dans un premier temps, pour que la forme désirée soit capable de se développer sans évolution de son génome).

La figure 5.19 montre les flux créés durant une division. On peut remarquer la création de multiples vortex autour de la créature dûs aux forts mouvements de fluides et aux collisions des particules. De plus, la liaison entre le simulateur hydrodynamique et le simulateur chimique est observable : lorsqu'une cellule meurt dans le simulateur chimique (qui contrôle la vie et la mort cellulaire), elle est détruite dans le simulateur hydrodynamique. Il en est de même en cas de naissance d'une nouvelle cellule. Le simulateur

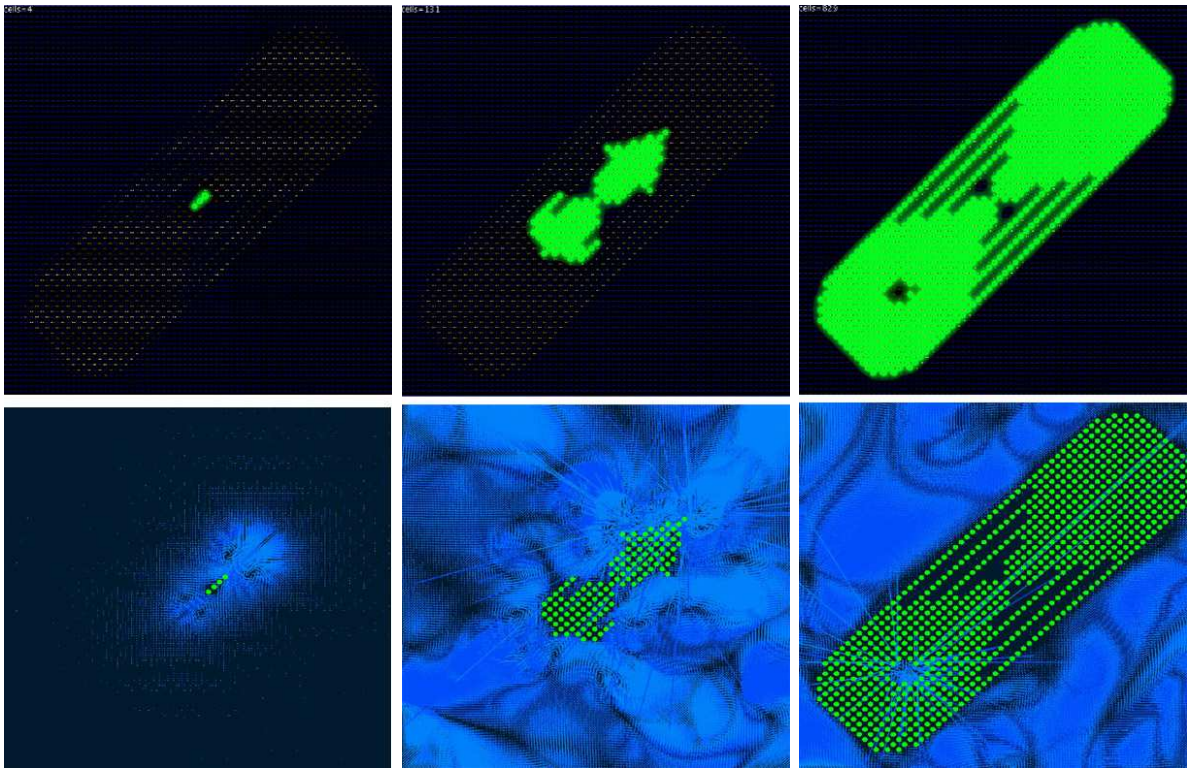


FIGURE 5.19 – *Flux hydrodynamiques créés par la division cellulaire.*

hydrodynamique gère bien la répartition des substrats dans ces cas de modification de l'organisme.

La seconde expérience consiste à déplacer un morphogène dans l'environnement. Cette étape de positionnement est une phase primordiale pour développer la forme d'une créature. Des cellules capables de se développer suivront par la suite ces morphogènes dans le but de créer la forme désirée.

L'organisme est composé de deux types de cellule :

- Un ensemble de cellules inactives qui servent à guider les morphogènes dans l'environnement. Le but de cette expérimentation étant de valider le simulateur hydrodynamique, ces cellules sont placées à la main en utilisant le génome de développement de forme présenté dans l'expérimentation de la section 5.2 et à l'aide de morphogènes positionnés par l'utilisateur. Ces cellules vont nous permettre d'analyser le réalisme des conditions aux bords du modèle hydrodynamique.
- Trois cellules capables d'absorber un substrat rouge d'un côté et de le rejeter de l'autre. Elles ont pour but de créer un courant dans l'organisme.

Les figures 5.20 et 5.21 montrent le résultat obtenu. Les pastilles vertes représentent les cellules du système. Dans la figure 5.20, les traits bleus représentent les vecteurs vitesses des particules de substrats. On remarque que les courants sont très forts à la sortie des

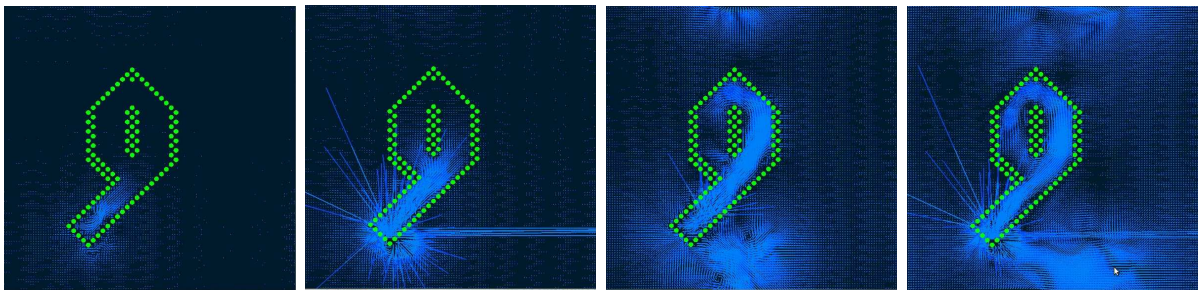


FIGURE 5.20 – *Champs de vecteurs vitesses de la diffusion des morphogènes.*

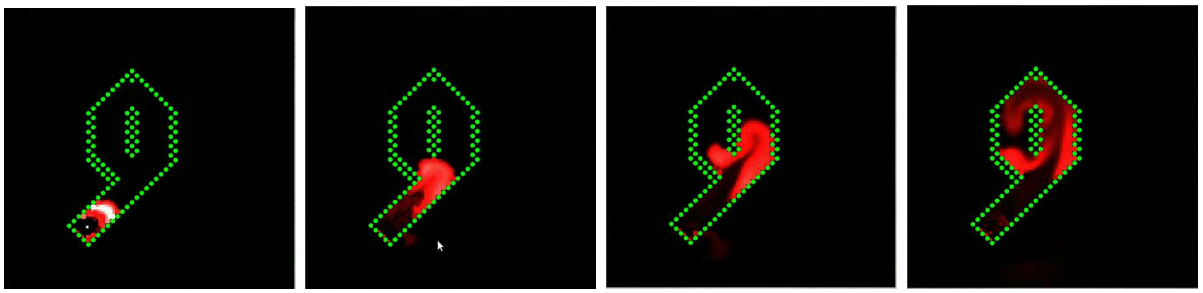


FIGURE 5.21 – *Densité des morphogènes dans l'environnement.*

cellules qui expulsent le substrat et ralentissent de plus en plus tout en suivant parfaitement la forme de l'organisme. La figure 5.21 montre les concentrations du substrat rouge au cours de la simulation. On voit que le fluide suit bien les vecteurs vitesses et se déplace le long de l'organisme tout en créant des vortex lors de la collision avec certaines cellules.

Ces deux petites expérimentations permettent de valider notre simulateur hydrodynamique et plus particulièrement la partie la plus complexe que sont les conditions aux bords des cellules qui peuvent apparaître et disparaître au cours de la simulation.

5.4.3 Expérimentation 10 : Intégration des trois simulateurs

L'intégration des trois simulateurs reste à faire. Comme le montrent les images de la figure 5.22, ils peuvent fonctionner en même temps mais le niveau physique n'interagit pas pour le moment avec le niveau hydrodynamique et inversement. Il serait intéressant de rajouter un lien physique-hydrodynamique afin de créer des mouvements de fluide lors de la torsion de la structure et un déplacement de la structure dû aux flux de substrats.

Cette intégration risque cependant de consommer beaucoup de ressource processeur étant donné le nombre de forces à appliquer dans les deux simulateurs. Nous pouvons évaluer les ressources supplémentaires à un processeur dédié au calcul des simulation physique et hydrodynamique. Il est pour cela nécessaire de réfléchir à une simplification. Elle peut être rendue possible en rassemblant les cellules d'un même type en un seul et même

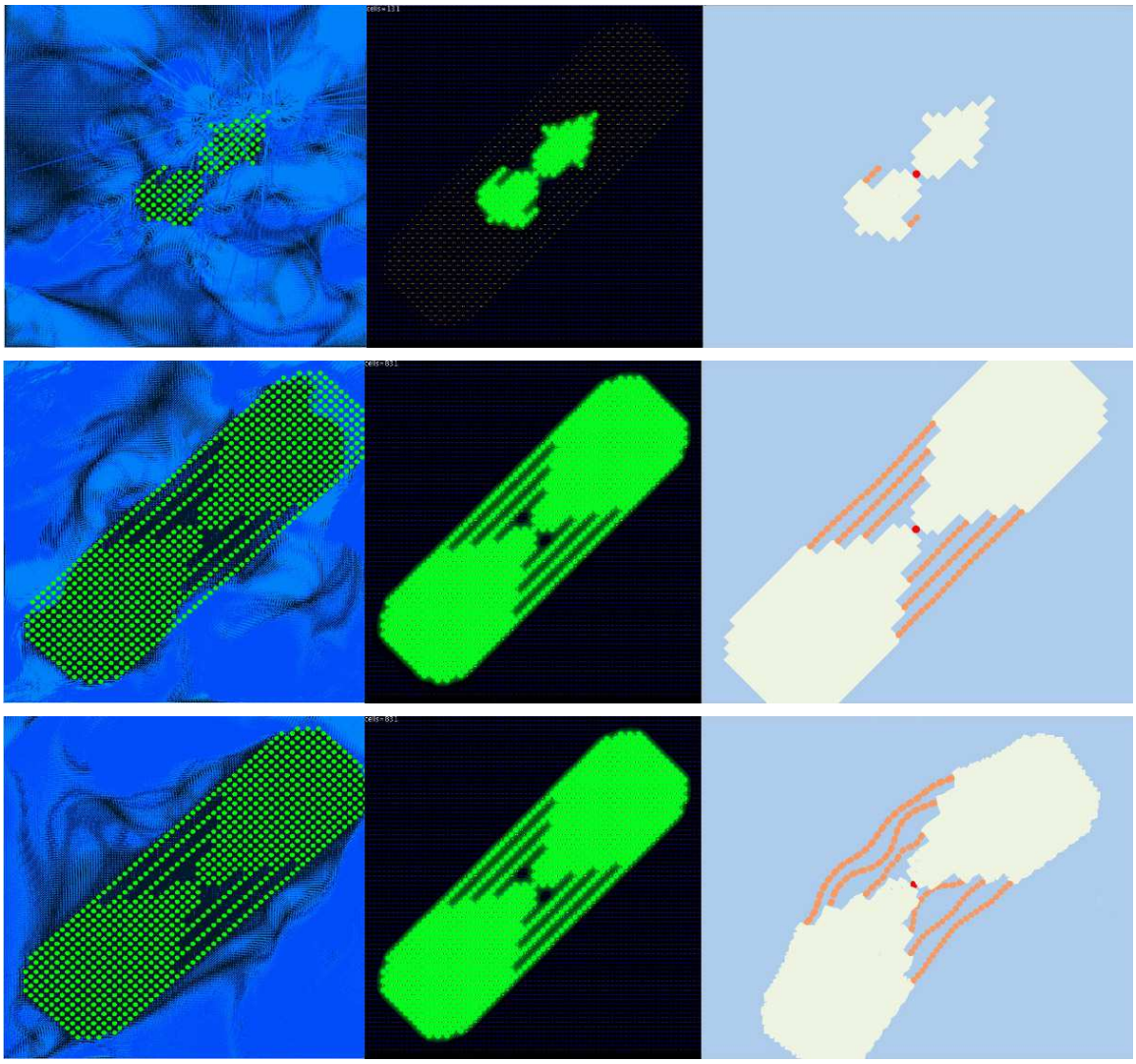


FIGURE 5.22 – *Intégration des 3 niveaux de simulation (de gauche à droite : le niveau hydrodynamique, le niveau chimique et le niveau physique).*

objet. On obtiendrait ainsi un os entier dans les mondes physique et hydrodynamique, bien que constitué d'un grand nombre de cellules dans le monde chimique. Les forces de collision seraient ainsi beaucoup plus simples à calculer.

La spécialisation cellulaire manque encore afin que le code génétique donne au simulateur physique le type et les capacités physiques d'une cellule. Nous avons modifié le code du modèle chimique pour que cette remarque soit prise en compte mais elle induit une augmentation de la complexité du génome. Il nous faut trouver de nouvelles techniques de recherche pour produire des génomes capables de différencier les cellules et non de les optimiser comme le fait actuellement le modèle présenté. Les modèles de réseaux artificiels de régulation de gènes présentés dans le chapitre 2.3.3 tels que ceux utilisés pour

le drapeau français pourraient être intéressants à utiliser ici, un changement de couleur correspondant dans ce cas à un changement de type de cellule.

5.5 Conclusion préliminaire

5.5.1 Récapitulatif des résultats obtenus

Dans ce chapitre, nous avons présenté les capacités de développement de créatures de notre modèle. Nous avons proposé des organismes accomplissant une certaine fonction ou capable de développer une morphologie particulière. Les organismes développés utilisent à la fois le réseau d'optimisation des actions et le système de sélection d'action pour arriver à leurs fins.

Ce modèle est aussi capable de réparer des structures endommagées : nous avons expérimenté les capacités de régénération de notre modèle en tuant aléatoirement ou manuellement des cellules de nos organismes et en observant les capacités de régénération de ceux-ci. Bien que leur espérance de vie s'en trouve réduite, les organismes sont toujours capables de recréer ce qui a été détruit dans le but de continuer à effectuer leur tâche.

Nous avons aussi présenté une version parallèle des algorithmes génétiques qui utilise la grille de calcul française Grid5000 et le middleware ProActive. Cette parallélisation permet de déployer l'algorithme génétique sur plusieurs centaines d'ordinateurs connectés en réseau. Le middleware ProActive simplifie grandement la tâche de parallélisation car il gère tous les échanges réseau et offre des services de retour sur erreur et de répartition des tâches nous permettant d'utiliser au mieux les ressources de la grille de calcul.

Enfin, nous avons présenté les prémices d'un ensemble constitué de trois simulateurs : le simulateur chimique et métabolique sur lequel porte principalement cette thèse, un simulateur physique permettant le déplacement de créatures dans un monde newtonien et enfin un simulateur hydrodynamique dans lequel les flux de substrat sont simulés afin de donner la capacité à nos cellules de modifier l'environnement sur des distances plus importantes. Bien qu'encore à un stade primitif, cet ensemble de simulateurs nous permettra dans le futur de construire une créature complète interagissant avec son environnement à plusieurs échelles.

Les résultats que nous avons présentés tout au long de cette thèse ont cependant quelques limites, présentées dans la section suivante.

5.5.2 Limites du modèle

Complexité

La principale limite de ce modèle est sa complexité. Le temps de convergence de l'algorithme génétique pour trouver un génome reste encore très élevé. Même si d'importantes améliorations ont été apportées durant cette thèse à l'aide de la parallélisation dans un premier lieu et en utilisant le mécanisme d'aide à la convergence présenté dans la section 4.3, les simplifications apportées au modèle ne permettent pas encore de développer des organismes aux fonctions complexes.

Des mécanismes améliorant encore les capacités de recherche doivent être testés. La réutilisation des chromosomes du code génétique de la créature [Acan and Tekol, 2003] permettant à un gène d'être exprimé plusieurs fois dans différentes parties (comportement, morphologie, métabolisme...) de l'organisme pourrait être intéressante dans notre modèle. En effet, diminuer la taille du génome diminue l'espace de recherche et ainsi le temps de convergence de l'algorithme de recherche tout en préservant une important liberté dans le codage des paramètres de l'organisme.

En plus de l'amélioration du codage de l'information, l'algorithme d'optimisation peut être perfectionné. De nouveaux algorithmes d'optimisation tels que les CMA-ES [Auger and Hansen, 2005] pourraient être testés sur notre modèle afin de voir si leur vitesse de convergence est meilleure que les algorithmes génétiques. De plus, la parallélisation de l'algorithme génétique avec le middleware ProActive a entraîné une explosion de la mémoire nécessaire, ce qui perturbe souvent les calculs. Il serait intéressant d'implémenter la version par îlots ou hybride des algorithmes génétiques parallèles présentés dans la section 4.1 afin de réduire cette consommation mémoire. L'algorithme pourra ainsi tourner plus efficacement sur une grille de calcul.

Paramétrage du modèle

Une autre limite du modèle concerne sa difficulté de paramétrage. Il porte sur deux points : le paramétrage de l'algorithme génétique et celui de l'environnement.

Le paramétrage de l'algorithme génétique a beaucoup été étudié dans la littérature dans le but de réduire au maximum le temps de convergence mais également de réduire le nombre de paramètres sur lesquels l'utilisateur doit agir. Les deux améliorations que nous avons apportées sur l'algorithme (parallélisation et aide à la convergence) ont permis une grande réduction du temps de recherche de l'algorithme. Alors que la parallélisation ne complexifie pas le paramétrage de l'algorithme, l'aide à la convergence rend la description de la fonction d'évaluation plus difficile. En effet, il est nécessaire de déterminer des

objectifs secondaires permettant une montée graduelle de la complexité des créatures vers le but final. Ces sous-objectifs doivent être définis avec précision car ils doivent guider la recherche tout en laissant suffisamment de liberté à l'algorithme génétique pour qu'il ne se restreigne pas à un optimum local.

La deuxième partie du paramétrage concerne la description de l'environnement. Celui-ci est capital car un organisme ne pourra se développer et atteindre l'objectif qui lui a été fixé que si l'environnement le lui permet. Trois composants de l'environnement sont importants à paramétrer :

- Les *substrats* et plus particulièrement leur vitesse de diffusion et leur placement initial dans l'environnement peuvent influencer la croissance des organismes. Par exemple, si certaines zones de l'environnement sont pauvres en substrats nécessaires au métabolisme des créatures, il y a de fortes chances que l'organisme résultant ne se développe pas dans ces zones. Si un des objectifs d'un tel organisme est de déplacer un substrat dans une de ces zones, l'algorithme génétique aura vraisemblablement du mal à trouver une solution convenable.
- Le nombre de *capteurs* utilisables par la cellule est important aussi : trop de capteurs pourrait entraîner une explosion de la complexité et trop peu ne permettrait pas à la cellule d'utiliser correctement ses actions.
- Les *actions* (type, coût et durée) doivent être paramétrées avec attention afin d'obtenir une sélection d'actions cohérentes pour la cellule. Il s'agit en effet de ne pas avoir une phase de croissance de l'organisme trop importante par rapport à sa phase fonctionnelle : une action de division ayant un coût trop faible entraîne souvent une prolifération trop importante des cellules qui remplissent l'environnement en réduisant considérablement la durée de vie de l'organisme (surconsommation de ressources) et en nuisant souvent à la fonction globale de l'organisme.

Même si un travail a été fait afin de simplifier la définition de la fonction d'évaluation et la création de nouveaux environnements[23], l'aide à la production d'une simulation reste encore sommaire et doit être améliorée afin de pouvoir produire des organismes plus aisément.

23. Nous avons réalisé une application annexe rendant la génération de forme très facile : elle permet de générer l'environnement (les morphogènes et leurs placements) correspondant à une forme directement dessinée par l'utilisateur.

6

Etude comparative

Dans le but de mieux positionner notre modèle par rapport aux travaux de référence cités dans le chapitre 2, nous proposons ici une base de comparaison axée sur les principales caractéristiques des modèles d'embryogenèse artificielle. Nous avons donc réalisé une étude comparative basée sur différentes propriétés des modèles de développement selon les axes suivants :

- la chimie artificielle,
- la spécialisation cellulaire,
- le développement cellulaire,
- les capacités du modèle.

6.1 La chimie artificielle

Comme nous l'avons vu dans l'état de l'art (chapitre 2), les modèles de chimie artificielle [Ono and Ikegami, 1999, Rasmussen et al., 2003] cherchent à simuler le plus fidèlement possible les interactions chimiques qui se déroulent au sein d'une cellule. Ils arrivent ainsi à simuler avec une très grande précision le maintien de la membrane cellulaire et les conditions environnementales nécessaires à la survie de la cellule. Cependant, le nombre de cellules simulées dans ce type de modèle est très faible : un petit groupe d'une demi-douzaine de cellules. Ces modèles ne peuvent ainsi pas produire d'organisme complet dans un temps de simulation raisonnable.

Les autres modèles de développement que nous avons présenté dans l'état de l'art ne possèdent pas à proprement parler de modèle de chimie artificielle. Ils intègrent souvent un mécanisme de diffusion de substrats permettant un positionnement des cellules dans l'environnement mais aucune interaction entre ces morphogènes n'est prise en compte.

Le modèle *Cell2Organ* de développement intègre pour sa part une chimie artificielle,

certes fortement simplifiée. Il ne permet pas la simulation des interactions atomiques entre chaque molécule du milieu mais permet une simulation des échanges possibles entre une cellule et son milieu ainsi que la dégradation de certaines molécules par les cellules. Ceci permet de créer une communication indirecte entre les cellules à l'aide des substrats et de l'environnement. De plus, nous sommes capables de simuler des réactions chimiques formelles. Ceci permet à nos cellules de mieux utiliser l'environnement et ses différentes ressources afin de produire un métabolisme leur permettant de réaliser des fonctions complexes. Cette simplification de la chimie nous permet de développer des organismes de beaucoup plus grande taille en comparaison des organismes produits par la chimie artificielle réaliste puisque notre modèle a montré ses capacités de fonctionnement avec plus de 800 cellules dans l'expérimentation avec le simulateur physique (newtonien) vu en fin de chapitre 5.

Bien que notre chimie soit fortement simplifiée et peu réaliste, elle permet cependant d'imaginer des échanges et des transformations possibles entre les cellules elles-mêmes ou entre les cellules et le milieu. En s'abstrayant des molécules chimiques, on peut imaginer des transferts ou des modifications d'outils, de matériaux ou autres permettant aux cellules de réaliser certains travaux dans l'environnement.

6.2 La spécialisation cellulaire

La spécialisation est un domaine qui intéresse beaucoup de chercheurs du domaine actuellement. Beaucoup de modèles récents l'intègrent mais avec pour seule expression un changement de couleur de la cellule. Un problème commun mettant en jeu les mécanismes de spécialisation est le problème du drapeau français (french flag problem) qui permet un changement de couleur en fonction du positionnement de la cellule dans l'environnement.

Dans la majorité des cas, cette spécialisation est due à un réseau de régulation génétique inspiré du modèle du vivant. Les différents modèles existants utilisent des morphogènes pour positionner les cellules dans l'environnement. Ces morphogènes sont soit placés dans l'environnement par la main de l'homme soit produit de manière artificielle par les cellules de l'organisme, afin de se localiser dans l'environnement. En fonction des densités des différents morphogènes, le réseau de régulation modifie l'expression des gènes de la cellule. Une même cellule pourra ainsi changer sa couleur en fonction de son placement dans l'environnement. Cependant, dans ces modèles, son activité ne sera pas modifiée par cette modification de couleur. Elle continuera notamment à se développer comme avant.

Certains modèles utilisent un réseau booléen pour se spécialiser. C'est par exemple le cas du modèle de Dellaert. Cependant, une fois de plus, seule la couleur de la cellule est

influencée par la spécialisation et non pas son action.

Dans notre modèle, le système de spécialisation est plus proche des réseaux booléens que d'un réseau de régulation inspiré du vivant. Il est basé sur un réseau d'optimisation dans lequel des flux de coefficients d'efficacité permettent l'optimisation d'un groupe d'actions durant une division cellulaire. Ainsi, une cellule capable de faire une certaine action à un instant donné de la simulation peut perdre cette aptitude, l'action devenant trop coûteuse en énergie vitale. De ce fait, bien que notre mécanisme de spécialisation soit moins réaliste que le réseau proposé dans la grande majorité des modèles de développement, il permet une spécialisation au niveau des actions qui entraîne une modification du comportement des cellules. Il est possible de voir son intérêt dans l'expérimentation 2 (le système de transfert) dans laquelle il permet la régulation de la longueur de l'organisme. Ce réseau permet donc d'optimiser l'utilisation des actions par la cellule.

6.3 Le modèle de développement

Basé sur une matrice 2D permettant une diffusion aisée du substrat et surtout une importante réduction de la complexité, notre modèle permet de développer des organismes contenant beaucoup plus de cellules que les autres modèles. En effet, la moitié des modèles existants utilisent un environnement discret (souvent une matrice 2D ou 3D) réduisant grandement la complexité de la diffusion des substrats ainsi que le modèle de divisions cellulaires. Les autres modèles utilisent un environnement continu permettant un plus grand réalisme du développement des organismes. Dans ce type d'environnement, les cellules peuvent avoir différentes formes ou différentes tailles. Un modèle masse-ressort permet alors une croissance plus réaliste de l'organisme en modifiant leur morphologie à la fois lors de division cellulaire mais aussi durant les phases de croissance des cellules.

Il est intéressant de regarder les différences de taille des organismes obtenus. Ceux développés avec les modèles existants possèdent entre 20 et 250 cellules. Notre modèle, grâce aux simplifications que nous avons opérées, particulièrement au niveau du mécanisme de régulation génétique, nous permet de développer des organismes de plusieurs centaines de cellules. Le plus grand obtenu est celui que nous avons plongé dans le simulateur physique newtonien qui contient environ 850 cellules. Nous avons pu intégrer de plus une chimie artificielle simplifiée permettant à l'organisme de construire un métabolisme dépendant des propriétés de son environnement.

Certains modèles, plus particulièrement ceux qui utilisent la troisième dimension et la physique pour se développer, prennent en compte les interactions mécaniques entre les cellules. Ainsi, les propriétés d'adhésion sont prises en compte et permettent d'obtenir

une croissance plus réaliste. Cette propriété que nous avons délibérément omise dans notre modèle pourrait être intéressante à étudier si nous décidions d'étendre notre modèle à la troisième dimension.

6.4 Les capacités du modèle

Dans le chapitre 2, nous avons résumé les capacités des différents modèles étudiés autour de trois principaux axes : la génération de forme, la fonction de l'organisme et le développement d'un métabolisme.

Nous avions remarqué alors que si la quasi totalité des modèles était capable de générer des formes avec différentes couleurs, seuls trois donnaient aux créatures développées une fonction. Nous avions vu que le modèle pionnier de la vie artificielle, le jeu de la vie de Conway, avait produit un automate cellulaire dont les formes étaient capables de se déplacer ou de compter. Plus tard, Fleisher et Barr ont développé un modèle de croissance permettant la génération d'un réseau de neurones. Enfin, Tufte a produit un modèle basé sur les automates cellulaires et obtient ainsi des organismes capables de modifier leur comportement en cours de simulation et ainsi de produire différentes opérations mathématiques. Notre modèle pour sa part est axé sur le développement de fonctions. La morphologie quand à elle émerge de la fonction. Ainsi, la forme en colonne du moissonneur de l'expérimentation 1 a émergé de la fonction qui lui était demandées, la forme linéaire du système de transfert (expérimentation 2) est la meilleure qui ait été retenue car la plus efficace, etc. Nous avons aussi montré les capacités de notre modèle à générer diverses formes en utilisant des morphogènes, pour le moment placés par la main de l'homme mais qui devraient à terme être positionnés par l'organisme grâce au simulateur hydrodynamique.

Enfin, une autre remarque que nous avions faite en fin d'état de l'art est l'absence générale de métabolisme des organismes développés avec les modèles existants. En effet, outre les modèles issus de la chimie artificielle dont le but est de simuler de manière détaillés tous les mécanismes cellulaires, aucun modèle ne prend en compte un idée de dépense énergétique liée à l'accomplissement d'une action (croissance, division, modification du plan de division, etc.). En partant de ce constat, nous avons décidé d'ajouter comme contrainte aux cellules la nécessité de développer un métabolisme à partir de l'environnement. Elles doivent ainsi trouver la façon de dégrader certains substrats afin de conserver un niveau d'énergie suffisant à leur survie pour ensuite effectuer la tache qui leur est demandé.

6.5 Bilan

Notre modèle montre une approche nouvelle du développement de créatures artificielles. Il a pour but de répondre à certains manques des modèles existants, tels que donner une fonction aux organismes, créer les briques d'une chimie artificielle ou les doter d'un métabolisme. Pour atteindre ces objectifs, nous avons simplifié d'autres mécanismes simulés de manière très réaliste dans les autres modèles. Par exemple, notre modèle de spécialisation cellulaire, bien que fonctionnant, n'est pas plausible au niveau biologique. Nous avons aussi préféré un modèle en deux dimensions afin de réduire le nombre d'actions possible par les cellules. Ceci à pour but de se focaliser sur les mécanismes de sélection des actions des cellules afin de produire diverses structures complexes.

Pour résumer les propriétés de notre modèle, nous allons reprendre succinctement les trois tableaux que nous avions présentés dans l'état de l'art en ne gardant que les modèles les plus proches du notre. Ainsi, le tableau 6.1 résume les propriétés des environnements, le tableau 6.2 résume les propriétés des cellules et le tableau 6.3 montre les capacités des organismes.

Modèles	Environnement		
	Représentation graphique	Chimie	Actions mécaniques
Rasmussen	Environnement 2D	Réaliste	Non
Chavoya & Duthen	Matrice 2D & 3D	Diffusion	Non
Knabe	Matrice 2D	Diffusion	Non
Eggenberger Hotz	Environnement 3D	Diffusion	Oui
Joachimczak	Environnement 3D	Diffusion	Oui
Tufte	Matrice 2D	Absente	Non
Cussat-Blanc	Matrice 2D	Simplifiée	Non

TABLE 6.1 – *Table des propriétés des environnements des systèmes de développement.*

Modèles	Cellule				
	Forme	Spécialisation	Régulation génétique	Communication	Evolution génétique
Rasmussen	Polygone	Non	Non	Aucune	Non
Chavoya	Cercle/Sphère	Couleur	Oui	Indirecte	Oui
Knabe	Polygone	Couleur	Oui	Aucune	Oui
Eggenberger Hotz	Sphère	Non	Oui	Indirecte	Oui
Joachimczak	Sphère	Couleur	Oui	Indirecte	Oui
Tufte	Carré	Fonction	Non	Aucune	Oui
Cussat-Blanc	Carré	Fonction	Oui	Indirecte	Oui

TABLE 6.2 – *Table des propriétés des cellules des systèmes de développement.*

Modèles	Capacités			
	Taille de l'organisme	Forme	Fonction	Métabolisme
Rasmussen	$\sim$ 6 cellules	Non	Non	Oui
Chavoya & Duthen	$\sim$ 250 cellules	Oui	Non	Non
Knabe	$\sim$ 100 cellules	Oui	Non	Non
Eggenberger Hotz	$\sim$ 75 cellules	Oui	Non	Oui
Joachimczak	<200 cellules	Oui	Non	Non
Tufte	$\sim$ 100 cellules	Oui	Oui	Non
Cussat-Blanc	$\sim$ 500 cellules	Oui	Oui	Oui

TABLE 6.3 – *Table des capacités des systèmes de développement.*

7

Conclusion

Dans le cadre de notre approche de développement de créatures artificielles pour la synthèse d'image et le peuplement de mondes virtuels, nous avons voulu étudier dans cette thèse une nouvelle approche basée sur un modèle de croissance. Dans la continuité des travaux de Nicolas Lassabe qui a produit des créatures à base de blocs et capables d'accomplir des actions simples tels que marcher, nager ou faire de la planche à roulettes, nous avons voulu poursuivre nos travaux en développant des créatures de plus petite échelle mais constituées de cellules et simulant certaines fonctions du vivant. Pour cela, cette thèse propose un nouveau modèle de simulation de croissance que nous avons nommé *Cell2Organ*. Il est basé sur une forte simplification du modèle de développement cellulaire biologique. Plus inspiré de la biologie que biologiquement plausible, nous avons ignoré dans un premier temps les règles de la physique et les interactions atomiques et moléculaires pour nous focaliser sur les capacités des cellules.

A travers nos différentes expérimentations, nous avons montré que notre modèle est capable de produire une diversité de fonctions et de morphologies. Toutes les créatures que nous avons développées ont la particularité de posséder un métabolisme leur permettant de créer de l'énergie vitale à partir d'éléments présents dans l'environnement. C'est donc un des rares modèles de développement prenant en compte le métabolisme, trop souvent oublié dans les modèles de la littérature.

Pour arriver à cela, nous avons dû dans un premier temps paralléliser la bibliothèque d'algorithme génétique que nous utilisons et créer une méthode de guidage de l'évolution pour aider l'algorithme génétique à trouver des solutions viables avant de pouvoir répondre au but, parfois décomposé en plusieurs sous-buts, qui leur est demandé. Ceci a permis de réduire grandement le temps de convergence de l'algorithme génétique et donc de pouvoir augmenter la complexité de nos créatures.

Nous avons montré qu'une des propriétés importantes de nos créatures est de pouvoir

s'auto-réparer. Sans aucune modification du modèle avant de tester cette propriété et sans qu'elle n'ait été pensée lors du développement du modèle, l'auto-réparation semble être une propriété inhérente au modèle et commune à une grande partie des modèles de développement.

Le but de notre modèle était aussi de pouvoir faire coopérer différents organes afin de produire un organisme entier. Nous avons montré que le modèle répond bien à cette demande en mettant dans l'environnement quatre organes différents et en les faisant coopérer afin de développer un organisme auto-alimenté. Bien que basé sur un assemblage manuel des organes, nous pouvons entrevoir la possibilité de développer un organisme artificiel composé de plusieurs organes coopérants dans le but de bien utiliser les substrats de l'environnement afin de réaliser une tâche de haut niveau et permettant d'augmenter la complexité du métabolisme de l'ensemble de l'organisme. Comme nous le verrons plus en détail dans les perspectives, nous devons cependant imaginer un mécanisme permettant de développer ce type d'organisme à partir d'une cellule unique.

7.1 Application aux bio et nano-systèmes

Les modèles de développement qui considèrent le métabolisme des organismes artificiels pourraient avoir un certain nombre d'applications dans le futur. Leurs propriétés d'auto-assemblage, d'auto-réparation et d'auto-alimentation (ou plus généralement et plus communément appelé le "self-*") sont importantes pour les futurs bio et nano systèmes. Beaucoup de travaux sont actuellement en cours pour développer des cellules artificielles [Service, 2005, Forster and Church, 2006] ou des nano-modules pour les futurs robots modulaires [Christensen, 2007, Yim et al., 2007, Zykov et al., 2008].

En biologie synthétique, domaine qui produira les futurs biosystèmes, les chercheurs travaillent aujourd'hui sur la modification du génome de différentes bactéries afin de leur faire produire des protéines particulières. Ces protéines sont ensuite utilisées pour exprimer une fonction particulière dans la cellule. Certains blocs de base de cette auto-organisation chimique sont déjà en place et beaucoup d'autres sont encore à découvrir [Forster and Church, 2006]. Dans les années à venir, il semble qu'il sera possible de créer des cellules capables de se diviser et d'avoir un ensemble d'actions possibles. Les modèles de développement, particulièrement ceux qui sont capables de prendre en compte le métabolisme des cellules, pourront être utilisés afin de trouver le meilleur génome à introduire dans les cellules. Ces modèles de développement devront être plus biologiquement plausibles que bio-inspirés du fait des contraintes imposées par la Nature. En d'autres termes, un modèle tel que *Cell2Organ* devra être plus précis dans la simulation des différents

mécanismes du vivant (mieux simuler la physique, les réactions chimiques, les interactions moléculaires dans le milieu et les cellules...) afin d'être suffisant pour ce type d'applications.

Notre modèle s'appliquera sans doute plus au domaine des nano-robots ou plus précisément aux méso-systèmes. Plusieurs équipes de recherche sont aujourd'hui capables de travailler au niveau des atomes pour modifier la structure de molécules ou de matériaux. Nous pouvons imaginer que, dans quelques années, la robotique moléculaire comme les robots moléculaires construits dans le laboratoire de Hod Lipson [Yim et al., 2007, Zykov et al., 2008] mesureront quelques nanomètres. Produire des structures composées de centaines de milliers de modules sera possible et permettra de créer des robots dont la structure sera proche des êtres vivants, un module du robot pouvant être considéré comme une cellule de l'être vivant [24]. Ces modules pourraient être capables d'accomplir une ou plusieurs actions voir même de se dupliquer. Ainsi, dans le but d'apprendre à ces robots à s'auto-assembler et à produire une fonction globale complexe, des modèles de développement pourront être là encore une bonne solution pour trouver le comportement de chaque module du robot. Dans ce cas, un modèle bio-inspiré comme présenté dans cette thèse pourrait être suffisant, les modules étant produits par l'homme et pouvant donc être plus ou moins façonnés selon ses désirs. Le métabolisme peut aussi être intéressant à prendre en compte dans ces robots car il leur permettrait d'utiliser les ressources de leur environnement (telles que le glucose contenu dans beaucoup de matières organiques par exemple) afin de produire l'énergie nécessaire à son fonctionnement.

7.2 Perspectives

Un certain nombre d'améliorations sont envisageables dès à présent. Premièrement, pour pallier aux limitations présentées dans le chapitre précédent, nous devons explorer les différentes méthodes qui peuvent aider à trouver une bonne solution pour les créatures que nous développons. Il nous faut pour cela implanter les différents algorithmes d'optimisation existants et les tester sur notre problème.

Il est clair que nous devons continuer à développer de nouveaux organes afin de produire des créatures possédant des fonctions plus complexes. En effet, en développant une bibliothèque importante de génomes divers, nous pourrons les assembler facilement pour obtenir des créatures possédant différents organes et une complexité importante.

24. La définition exacte d'une cellule d'après l'encyclopédie Universalis étant la "plus petite unité du vivant, la cellule est la base de la structure et du fonctionnement de l'organisme", cette définition montre bien l'idée de modularité de la cellule.

Le développement de nouveaux organes pose aussi la question de leur assemblage. Actuellement, le positionnement des cellules est fait à la main. Pour obtenir un organisme complet développé à partir d'une cellule unique, il faudrait construire un premier organisme capable de positionner les quatre cellules initiales de l'organisme. Ce "pré-organisme" se résorberait alors afin de laisser la place à l'organisme suivant et ne pas interférer dans la simulation. L'organisme attendu pourra alors se développer et atteindre le but qu'on lui a fixé.

La génération de formes de créatures peut aussi être grandement améliorée. Un seul morphogène pourrait être utilisé au lieu des quatre nécessaires. Cela augmente cependant la complexité du problème mais simplifie le placement automatique des morphogènes. En effet, nous avons vu que le simulateur hydrodynamique permet aux cellules de propulser un morphogène avec une certaine force dans l'environnement produisant ainsi différents flux. Nous pensons qu'il est possible de trouver un organisme capable de positionner les morphogènes dans l'environnement avant de se résorber et de laisser la place à la créature finale.

Une autre remarque que l'on peut faire sur la génération de forme, sur l'étoile de mer par exemple, est que la créature ne croît pas de manière régulière. En effet, certaines branches se développent plus vite que d'autres. Il en va de même pour la méduse dont l'ombrelle se développe trop vite par rapport au développement des tentacules. Pour résoudre ce problème, nous pensons qu'il serait intéressant de calculer la fitness de la créature à différents moments de son développement. On peut alors imaginer que la créature ait un certain nombre de points de passage à respecter dans le temps, et le meilleur génome serait celui qui respecte un maximum de points de contrôle. Cependant, cette contrainte augmente encore une fois la complexité de la solution.

Enfin, le passage du modèle à la troisième dimension doit aussi être sérieusement étudié. Les perspectives d'une telle amélioration sont nombreuses. Cela permettra de développer d'entières créatures composées de cellules, dotées d'un métabolisme, de capacité de régénération et pouvant se déplacer dans un monde virtuel 3D basé sur un simulateur physique. Malheureusement, il entraînera aussi une importante augmentation de la complexité dans la recherche des génomes. Pour essayer d'évaluer cette augmentation, on peut citer par exemple le problème de la division cellulaire. Actuellement, en deux dimensions, nous avons quatre actions de division (une par direction possible). En trois dimensions, il y en aura six. Il en sera de même pour la quasi totalité des actions telles que les absorptions ou les rejets de substrats. De plus, le système de sélection d'action sera lui aussi soumis à la même augmentation de complexité en intégrant deux nouveaux capteurs par substrat dans ses règles.

Et ce genre de modèle pourrait peut être servir de représentation interne d'un robot constitué de milliers de cellules...

Remerciements

Les expérimentations réalisées dans cette thèse ont été réalisées avec la grille de calculs française Grid5000, une initiative du ministère de la recherche française en motivation de l'action de l'ACI GRID, de l'INRIA, du CNRS, de RENATER et des autres partenaires (voir https://www.grid5000.fr).

La grille de calcul a pu être utilisée au meilleur de ces capacités grâce au middleware ProActive développé par l'équipe OASIS dirigée par Denis Caromel à l'INRIA Sophia-Antipolis. Plus de détails sur ce middleware sont disponible sur le site http://proactive.inria.fr.

Bibliographie

[Acan and Tekol, 2003] Acan, A. and Tekol, Y. (2003). Chromosome reuse in genetic algorithms. *Lecture Notes in Computer Science*, pages 695–705.

[Aristote, 330] Aristote (AC 330). *De l'âme*.

[Auger and Hansen, 2005] Auger, A. and Hansen, N. (2005). A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2.

[Banzhaf, 2003] Banzhaf, W. (2003). Artificial regulatory networks and genetic programming. *Genetic Programming Theory and Practice*, pages 43–62.

[Baude et al., 2002] Baude, F., Caromel, D., Mestre, L., Huet, F., and Vayssière, J. (2002). Interactive and descriptor-based deployment of object-oriented grid applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, pages 93–102, Edinburgh, Scotland. IEEE Computer Society.

[Berman, 1999] Berman, F. (1999). High-performance schedulers. *The Grid : Blueprint for a New Computing Infrastructure, Morgan Kaufmann*, pages 279–309.

[Bianchini and Brown, 1993] Bianchini, R. and Brown, C. (May 1993). Parallel genetic algorithms on distributed-memory architectures. *Technical Report (revised version), University of Rochester*.

[Boeing and Bräunl, 2007] Boeing, A. and Bräunl, T. (2007). Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques (Graphite 2007)*, pages 281–288.

[Bongard and Pfeifer, 2003] Bongard, J. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. *Morpho-functional machines : The new species (designing embodied intelligence)*, pages 237–258.

[Bornhofen, 2008] Bornhofen, S. (2008). *Emergence de dynamiques évolutionnaires dans une approche multi-agents de plantes virtuelles*. PhD thesis, Université Paris-Sud.

[Bowers, 2005] Bowers, C. (2005). Simulating evolution with a computational model of embryogeny : Obtaining robustness from evolved individuals. *Lecture notes in computer science*, 3630 :149.

[Branke et al., 2004] Branke, J., Kamper, A., and Schmeck, H. (2004). Distribution of evolutionary algorithms in heterogeneous networks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 923–934.

[Brooks, 1986] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, 2(1) :14–23.

[Cantù-Paz, 1997] Cantù-Paz, E. (1997). A survey of parallel genetic algorithms. *Technical report 95004, Illinois Genetic Algorithms Laboratory, Urbana, IL.*

[Caromel, 1993] Caromel, D. (1993). Towards a Method of Object-Oriented Concurrent Programming. *Communications of the ACM*, 36(9) :90–102.

[Caromel et al., 2006] Caromel, D., Delbe, C., di Costanzo, A., and Leyton, M. (2006). Proactive : an integrated platform for programming and running applications on grids and p2p systems. *Computational Methods in Science and Technology*, 12.

[Chavoya and Duthen, 2008] Chavoya, A. and Duthen, Y. (2008). A cell pattern generation model based on an extended artificial regulatory network. *Biosystems.*

[Christensen, 2007] Christensen, D. (2007). Experiments on Fault-Tolerant Self-Reconfiguration and Emergent Self-Repair. In *IEEE Symposium on Artificial Life (ALIFE'07)*, pages 355–361.

[Cleland and Chyba, 2002] Cleland, C. and Chyba, C. (2002). Defining Life. *Origins of Life and Evolution of the Biosphere*, 32(4) :387–394.

[Cliff and Miller, 1996] Cliff, D. and Miller, G. (1996). Co-evolution of pursuit and evasion II : Simulation methods and results. In *From Animals to Animats IV : Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior. MIT Press-Bradford Books, Cambridge, MA.*

[Cussat-Blanc et al., 2007] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2007). Using a single cell to create an entire organ. *International Conference on Artificial reality and Telexistance (ICAT).*

[Cussat-Blanc et al., 2008a] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2008a). From single cell to simple creature morphology and metabolism. In *Artificial Life XI*, pages 134–141. MIT Press, Cambridge, MA.

[Cussat-Blanc et al., 2008b] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2008b). From single cell to simple creature morphology and metabolism. In *Artificial Life XI*, pages 134–141. MIT Press, Cambridge, MA.

[Cussat-Blanc et al., 2009a] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2009a). Cell2organ : Self-repairing artificial creatures thanks to a healthy metabolism. In *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2009)*.

[Cussat-Blanc et al., 2009b] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2009b). Digital organ cooperation : Toward the assembly of a self-feeding organism. In *Proceedings of the European Conference on Artificial Life (ECAL'09)*.

[Cussat-Blanc et al., 2009c] Cussat-Blanc, S., Luga, H., and Duthen, Y. (2009c). Making a self-feeding structure by assembly of digital organs. In *Proceedings of the Australian Conference on Artificial Life (ACAL'09), A paraître*.

[Cussat-Blanc et al., 2008c] Cussat-Blanc, S., Viale, F., Luga, H., and Caromel, D. (2008c). Genetic algorithms and grid computing for artificial embryogeny. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 281–282. ACM New York, NY, USA.

[Darwin, 1859] Darwin, C. (1859). *On the origin of species*.

[Davidson, 2006] Davidson, E. H. (2006). The regulatory genome : gene regulatory networks in development and evolution. *Academic Press*.

[Davis, 1989] Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms table of contents*, pages 61–69. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Dawkins, 1986] Dawkins, R. (1986). The blind watchmaker. *Longman Scientific & Technical*.

[de Garis, 1999] de Garis, H. (1999). Artificial embryology and cellular differentiation. In Peter J. Bentley, e., editor, *Evolutionary Design by Computers*, pages 281–295.

[De Jong, 1975] De Jong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. University of Michigan, Logic of Computers Group.

[Deamer and Fleischaker, 1994] Deamer, D. and Fleischaker, G. (1994). *Origins of life : the central concepts*. Jones and Bartlett Boston.

[Dellaert and Beer, 1994] Dellaert, F. and Beer, R. (1994). Toward an evolvable model of development for autonomous agent synthesis. In *Artificial Life IV*, Cambridge, MA. MIT press.

[Devert, 2009] Devert, A. (2009). *Building processes optimization : Toward an artificial ontologeny based approach*. PhD thesis, Université Paris-Sud 11.

[Dittrich et al., 2001] Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries - a review. *Artificial Life*, 7(3) :225–275.

[Donikian and Rutten, 95] Donikian, S. and Rutten, E. (95). Reactivity, concurrency, data-ow and hierarchical preemption for behavioural animation. *Programming Paradigms in Graphics*.

[Doursat, 2008] Doursat, R. (2008). Organically grown architectures : Creating decentralized, autonomous systems by embryomorphic engineering. *Organic Computing*, pages 167–200.

[Doursat, 2009] Doursat, R. (2009). Facilitating evolutionary innovation by developmental modularity and variability. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 683–690. ACM.

[Dyson, 1985] Dyson, F. (1985). *Origin of life*. Cambridge University Press Cambridge.

[Eggenberger Hotz, 2003] Eggenberger Hotz, P. (2003). Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. *On Growth, Form and Computers*, page 302.

[Eggenberger Hotz, 2004] Eggenberger Hotz, P. (2004). Asymmetric cell division and its integration with other developmental processes for artificial evolutionary systems. In *Artificial Life IX*, pages 387–392.

[Eiben et al., 1999] Eiben, A., Hinterding, R., and Michalewicz, Z. (1999). Parameter Control in Evolutionary Algorithms. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 3(2).

[Everaars and Koren, 1997] Everaars, C. T. H. and Koren, B. (1997). Using coordination to parallelize sparse-grid methods for 3D CFD problems. In *219*, page 23. Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-369X.

[Ferguson et al., 1992] Ferguson, I., Laboratory, C., and of Cambridge, U. (1992). *TouringMachines : An architecture for dynamic, rational, mobile agents*.

[Flann et al., 2005] Flann, N., Hu, J., Bansal, M., Patel, V., and Podgorski, G. (2005). Biological development of cell patterns : characterizing the space of cell chemistry genetic regulatory networks. *Lecture notes in computer science*, 3630 :57.

[Fleischer and Barr, 1992] Fleischer, K. and Barr, A. (1992). A Simulation Testbed for the Study of Multicellular Development : The Multiple Mechanisms of Morphogenesis. In *Artificial life III : proceedings of the Workshop on Artificial Life, held June 1992 in Santa Fe, New Mexico*, page 389. Addison-Wesley Longman.

[Forster and Church, 2006] Forster, A. and Church, G. (2006). Towards synthesis of a minimal cell. *Molecular Systems Biology*, 2(1).

[Fraser and Perkel, 1990] Fraser, S. and Perkel, D. (1990). Competitive and positional cues in the patterning of nerve connections. *Journal of Neurobiology*, 21(1).

[Frisch et al., 1986] Frisch, U., Hasslacher, B., and Pomeau, Y. (1986). Lattice-gas automata for the Navier-Stokes equation. *Physical Review Letters*, 56(14) :1505–1508.

[Fukunaga et al., 1994] Fukunaga, A., Marks, J., and Ngo, J. T. (1994). Automatic control of physically realistic animated figures using evolutionary programming. *Proc. of Third Annual Conference on Evolutionary Programming*, pages 76–83.

[Funge et al., 1999] Funge, J., Tu, X., and Terzopoulos, D. (1999). Cognitive modeling : Knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 29–38. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[Garcia Carbajal et al., 2004] Garcia Carbajal, S., Moran, M. B., and Martinez, F. G. (2004). Evolgl : Life in a pond. *Artificial Life XI*, pages 75–80.

[Gardner, 1970] Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game life. *Scientific American*, 223 :120–123.

[Goldberg, 1989] Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[Gomez, 2004] Gomez, J. (2004). Self adaptation of operator rates in evolutionary algorithms. *Lecture Notes in Computer Science*, pages 1162–1173.

[Haddow and Hoye, 2009] Haddow, P. and Hoye, J. (2009). Investigating the effect of regulatory decisions in a development model. In *CEC 2009 : Congress on Evolutionary Computation*.

[Hardy et al., 1976] Hardy, J., De Pazzis, O., and Pomeau, Y. (1976). Molecular dynamics of a classical lattice gas : Transport properties and time correlation functions. *Physical Review A*, 13(5) :1949–1961.

[Heguy et al., 2001] Heguy, O., Rodriguez, N., Luga, H., Jessel, J., and Duthen, Y. (2001). Virtual environment for cooperative assistance in teleoperation. In *Proceedings of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2001). 9 (3) : II9-II12*.

[Herrera et al., 2005] Herrera, J., Huedo, E., Montero, R. S., and Llorente, I. M. (2005). A grid-oriented genetic algorithm. *in Advances in Grid Computing - EGC 2005*, pages 315–322.

[Holland, 1975] Holland, J. (1975). Adaptation in natural and artificial systems. an introductory analysis with applications to biology, control and artificial intelligence. *Ann Arbor : University of Michigan Press, 1975*.

[Holland and Reitman, 1978] Holland, J. H. and Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. *Pattern-Directed Inference Systems*.

[Hopfield, 1982] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8) :2554–2558.

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

[Horvitz and Herskowitz, 1992] Horvitz, H. and Herskowitz, I. (1992). Mechanisms of asymmetric cell division : two Bs or not two Bs, that is the question. *Cell*, 68(2) :237.

[Hutton, 2007] Hutton, T. J. (2007). Evolvable self-reproducing cells in a two-dimensional artificial chemistry. *Artificial Life*, 13(1) :11–30.

[Jaeger, 2001] Jaeger, H. (2001). The echo state approach to analysing and training recurrent neural networks. *submitted for publication*.

[Joachimczak and Wróbel, 2008] Joachimczak, M. and Wróbel, B. (2008). Evo-devo in silico : a model of a gene network regulating multicellular development in 3d space with artificial physics. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI : Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 297–304. MIT Press, Cambridge, MA.

[Joachimczak and Wróbel, 2009] Joachimczak, M. and Wróbel, B. (2009). Evolution of the morphology and patterning of artificial embryos : scaling the tricolour problem to the third dimension. In *10th European Conference on Artificial Life (ECAL09)*. Springer Verlag.

[Julstrom, 1997] Julstrom, B. (1997). Adaptive operator probabilities in a genetic algorithm that applies three operators. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 233–238. ACM New York, NY, USA.

[Kauffman, 1969] Kauffman, S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theorical Biology*, 22 :437–467.

[Knabe et al., 2008] Knabe, J., Schilstra, M., and Nehaniv, C. (2008). Evolution and morphogenesis of differentiated multicellular organisms : autonomously generated diffusion gradients for positional information. *Artificial Life XI*, 11 :321.

[Kodjabachian and Meyer, 1998] Kodjabachian, J. and Meyer, J. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9(5) :796–812.

[Kommu and Pomeranz, 1992] Kommu, V. and Pomeranz, I. (1992). Effect of communication in a parallel genetic algorithm. In *ICPP (3)*, pages 310–317.

[Komosinski and Ulatowski, 1999] Komosinski, M. and Ulatowski, S. (1999). Towards a simulation of a nature-like world creatures and evolution. In *ECAL '99*, pages 261–265, London, UK. Springer-Verlag.

[Krčah, 2007] Krčah, P. (2007). Evolving virtual creatures revisited. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM New York, NY, USA.

[Larsen, 2004] Larsen, E. W. (2004). A view of phenotypic plasticity from molecules to morphogenesis. In *Environment, Development, and Evolution : Toward a Synthesis*, pages 117–124. MIT Press, Cambridge.

[Lassabe, 2008] Lassabe, N. (2008). *Morphogenèse et Evolution de Créatures artificielles*. Thèse de doctorat, Université des Sciences Sociales, Toulouse, France.

[Lassabe et al., 2007] Lassabe, N., Luga, H., and Duthen, Y. (2007). A New Step for Evolving Creatures. In *IEEE-ALife'07*, pages 243–251. IEEE.

[Le Peuch and Dorée, 2000] Le Peuch, C. and Dorée, M. (2000). Le temps du cycle cellulaire. *médecine/sciences*, 16(4) :461–8.

[Lim et al., 2007] Lim, D., Ong, Y.-S., Jin, Y., Sendhoff, B., and Lee, B.-S. (2007). Efficient hierarchical parallel genetic algorithms using grid computing. *Future Gener. Comput. Syst.*, 23(4) :658–670.

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development. II. Simple and branching filaments with two-sided inputs. *Journal of theoretical biology*, 18(3) :300.

[Lindenmayer, 1971] Lindenmayer, A. (1971). Developmental systems without cellular interactions, their languages and grammars. *Journal of Theoretical Biology*, 30(3) :455.

[Lipson, 2006] Lipson, H. (2006). Evolutionary robotics and open-ended design automation. *Biomimetics : Biologically Inspired Technologies. Ed. Yoseph Bar-Cohen. Boca Raton, FL : CRC Press*.

[Lipson, 2007] Lipson, H. (2007). Curious and Creative Machines. *Lecture Notes in Computer Science*, 4850 :315.

[Mavelli and Ruiz-Mirazo, 2007] Mavelli, F. and Ruiz-Mirazo, K. (2007). Stochastic simulations of minimal self-reproducing cellular systems. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 362(1486) :1789.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4) :115–133.

[Miller, 2003] Miller, J. (2003). Evolving Developmental Programs for Adaptation, Morphogenesis, and Self-Repair. In *Advances in Artificial Life*, pages 256–265. Springer.

[Mouret and Doncieux, 2008] Mouret, J. and Doncieux, S. (2008). Incremental Evolution of Animats' Behaviors as a Multi-objective Optimization. *Lecture Notes in Computer Science*, 5040 :210–219.

[Neuhaus, 1991] Neuhaus, P. (1991). Solving the mapping problem - experiences with a genetic algorithm. In *PPSN I : Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 170–175, London, UK. Springer-Verlag.

[Odell et al., 1981] Odell, G., Oster, G., Alberch, P., and Burnside, B. (1981). The mechanical basis of morphogenesis. I. Epithelial folding and invagination. *Developmental Biology*, 85(2) :446.

[Ono and Ikegami, 1999] Ono, N. and Ikegami, T. (1999). Model of Self-Replicating Cell Capable of Self-Maintenance. In *Advances in Artificial Life : 5th European Conference, Ecal'99, Lausanne, Switzerland, September, 1999 : Proceedings*, page 399. Springer.

[Ontanon et al., 2007] Ontanon, S., Mishra, K., Sugandh, N., and Ram, A. (2007). Case-based planning and execution for real-time strategy games. *Lecture Notes in Computer Science*, 4626 :164.

[Panzoli et al., 2008] Panzoli, D., Luga, H., and Duthen, Y. (2008). A Reactive Architecture Integrating an Associative Memory for Sensory-Driven Intelligent Behavior. *Lecture Notes in Computer Science*, 5208 :528–529.

[Parker, 2001] Parker, G. (2001). The incremental evolution of gaits for hexapod robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1114–1121.

[Pascalie, 2009] Pascalie, J. (2009). *Gastrulation Artificielle pour le Positionnement de Morphogènes de Créatures Artificielles*. Rapport de Master.

[Pollack et al., 2003] Pollack, J., Hornby, G., Lipson, H., and Funes, P. (2003). Computer creativity in the automatic design of robots. *Leonardo*, 36(2) :115–121.

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer-Verlag New York, Inc. New York, NY, USA.

[Pruyne and Livny, 1996] Pruyne, J. and Livny, M. (1996). Interfacing Condor and PVM to harness the cycles of workstation clusters. *Future Generation Computer Systems*, 12(1) :67–85.

[Rasmussen et al., 2003] Rasmussen, S., Chen, L., Nilsson, M., and Abe, S. (2003). Bridging nonliving and living matter. *Artificial Life*, 9(3) :269–316.

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

[Reil, 1999] Reil, T. (1999). Dynamics of gene expression in an artificial genome-implications for biological and artificial ontogeny. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 457–466.

[Reynolds, 1987] Reynolds, C. (1987). Flocks, herds and schools : A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM New York, NY, USA.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386–408.

[Sanchez et al., 2004] Sanchez, S., Balet, O., Luga, H., and Duthen, Y. (2004). Autonomous virtual actors. *LECTURE NOTES IN COMPUTER SCIENCE.*, pages 68–78.

[Sanza, 2001] Sanza, C. (2001). *Evolution d'entités virtuelles coopératives par système de classifieurs*. Ph.D thesis.

[Sekaj, 2004] Sekaj, I. (2004). Robust parallel genetic algorithms with re-initialisation. In Yao, X., Burke, E., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J., Kabán, P. T. A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 410–419, Birmingham, UK. Springer-Verlag.

[Service, 2005] Service, R. (2005). How Far Can We Push Chemical Self-Assembly ?

[Silva et al., 1999] Silva, L. M., Batista, V., Martins, P., and Soares, G. (1999). Using mobile agents for parallel processing. In *DOA '99 : Proceedings of the International Symposium on Distributed Objects and Applications*, page 34, Washington, DC, USA. IEEE Computer Society.

[Sims, 1994] Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial Life IV*, pages 28–39.

[Solga et al., 2007] Solga, A., Cerman, Z., Striffler, B., Spaeth, M., and Barthlott, W. (2007). The dream of staying clean : Lotus and biomimetic surfaces. *Bioinspiration and Biomimetics*, 2(4) :126.

[Srinivas and Patnaik, 1994] Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in geneticalgorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4) :656–667.

[Stam, 1997] Stam, J. (1997). A general animation framework for gaseous phenomena. *ERCIM Research Report*, 47 :369376.

[Stam, 2003] Stam, J. (2003). Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*, volume 18.

[Stanley, 2004] Stanley, K. (2004). *Efficient evolution of neural networks through complexification*. PhD thesis, The University of Texas at Austin.

[Stewart et al., 2005] Stewart, F., Taylor, T., and Konidaris, G. (2005). Metamorph : Experimenting with genetic regulatory networks for artificial development. In *ECAL'05*, pages 108–117.

[Tanese, 1989] Tanese, R. (1989). Distributed genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 434–439, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Taylor, 2000] Taylor, T. (2000). Artificial life techniques for generating controllers for physically modelled characters. In *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000)*.

[Torrel, 2007] Torrel, J. (2007). *Modélisation et Simulation de phénomènes complexes par systèmes multi-agents hiérarchiques : application en cosmologie*. PhD thesis, Université de Paris 5.

[Tu and Terzopoulos, 1994] Tu, X. and Terzopoulos, D. (1994). Artificial fishes : Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50. ACM New York, NY, USA.

[Tufte, 2008] Tufte, G. (2008). Evolution, development and environment toward adaptation through phenotypic plasticity and exploitation of external information. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI : Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 624–631. MIT Press, Cambridge, MA.

[Tufte, 2009] Tufte, G. (2009). Metamorphosis and Artificial Development : An Abstract Approach to Functionality. In *10th European Conference on Artificial Life (ECAL09)*. Springer Verlag.

[Tufte and Haddow, 2007] Tufte, G. and Haddow, P. C. (2007). Extending artificial development : Exploiting information for the achievement of phenotypic plasticity. In *Internaltional Conference on Systems Evolvable : From Biology to Hardware (ICES'07)*, pages 297–308.

[Turing, 1952] Turing, A. (1952). The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641) :37–72.

[Urzelai and Floreano, 1999] Urzelai, J. and Floreano, D. (1999). Incremental evolution with minimal resources. *Proceedings of IKW99*.

[Ventrella, 1998a] Ventrella, J. (1998a). Attractiveness vs efficiency (how mate preference affects location in the evolution of artificial ng organisms). *Artificial Life VI*, pages 178–186.

[Ventrella, 1998b] Ventrella, J. (1998b). Designing emergence in animated artificial life worlds. *Internationnal Conference on Virtual Worlds*, pages 143–155.

[Von Neumann and Burks, 1966] Von Neumann, J. and Burks, A. (1966). Theory of self-reproducing automata.

[Walker, 2004] Walker, M. (2004). Comparing the performance of incremental evolution to direct evolution. In *Second International Conference on Autonomous Robots and Agents*, pages 119–124.

[Wilson, 1994] Wilson, S. (1994). ZCS : A zeroth level classifier system. *Evolutionary Computation*, 2(1) :1–18.

[Wilson et al., 1998] Wilson, S. et al. (1998). Generalization in the XCS classifier system. *Genetic programming*, pages 665–674.

[Winkeler and Manjunath, 1998] Winkeler, J. and Manjunath, B. (1998). Incremental evolution in genetic programming. *Genetic Programming*, pages 403–411.

[Wolpert, 1968] Wolpert, L. (1968). The french flag problem : A contribution to the discussion on pattern development and regulation. *Towards a theoretical biology*, 1 :125–133.

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

[Zykov et al., 2008] Zykov, V., Phelps, W., Lassabe, N., and Lipson, H. (2008). Molecubes Extended : Diversifying Capabilities of Open-Source Modular Robotics. In *International Conference on Intelligent RObots and Systems, Self-Reconfigurable Robots Workshop (IROS 08)*.

A

Expérience de Miller-Urey

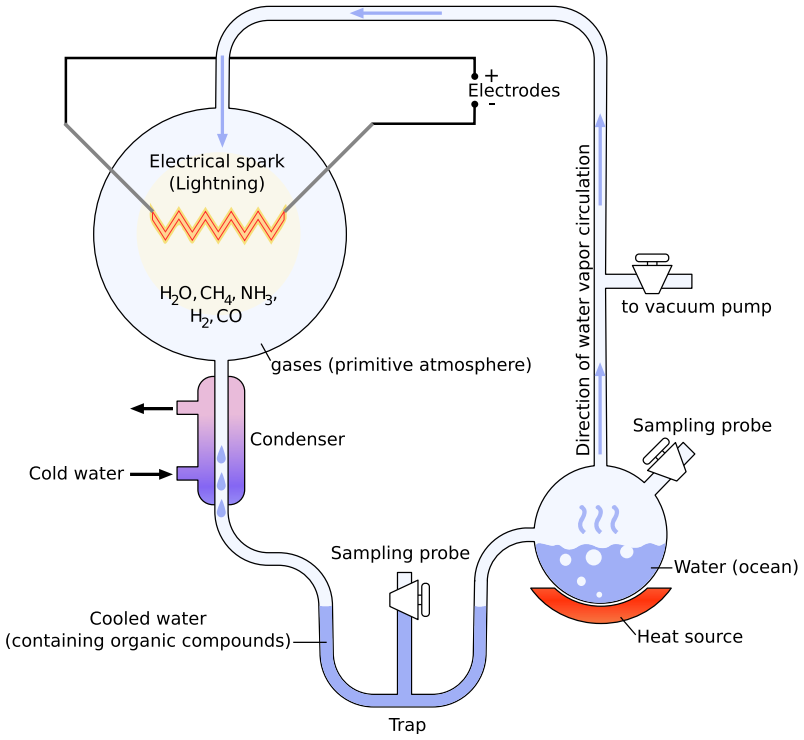


FIGURE A.1 – *L'expérience de Miller-Urey a permis la synthèse de 13 des 22 acides aminés existants en partant des hypothèses sur la constitution de l'atmosphère primitive de la Terre*[25].

L'expérience de Miller et Urey, réalisée par les scientifiques du même nom en 1953, tente de recréer les 22 acides aminés connus sur notre planète en essayant de reconstituer les conditions de la "soupe primitive". L'objectif était de prouver l'hypothèse d'Oparin et de Haldane qui pensaient que les composés organiques étaient apparus à partir de

25. Source : *http://fr.wikipedia.org/wiki/Fichier:Miller-Urey_experiment-en.svg*

composés inorganiques.

Les composés supposés présents dans l'atmosphère primitive (l'eau, le méthane, l'ammoniac et l'hydrogène) sont enfermés dans une boucle de tubes stériles illustrée par la figure A.1. Elle contient les éléments suivants :

- une source de chaleur permettant l'évaporation de l'eau,
- deux électrodes permettant la simulation d'éclairs traversant l'atmosphère,
- un système de refroidissement permettant la condensation de l'eau et la récupération des possibles éléments organiques créés.

Cette expérience permet de recréer le cycle de l'eau avec une phase de condensation, une phase dans laquelle elle est sous forme de vapeur et une condensation qui simule la pluie sur la planète.

Après plusieurs semaines d'expérimentations, Miller et Urey arrivent à recréer 13 des 22 acides aminés existants et qui sont toujours actuellement utilisés par les cellules pour produire des protéines. Ils observent aussi la formation de sucres et de lipides ainsi que certains composants des acides nucléiques mais cependant pas de molécule aussi complexe que l'ADN ou l'ARN.

B

Diagramme de classes du modèle

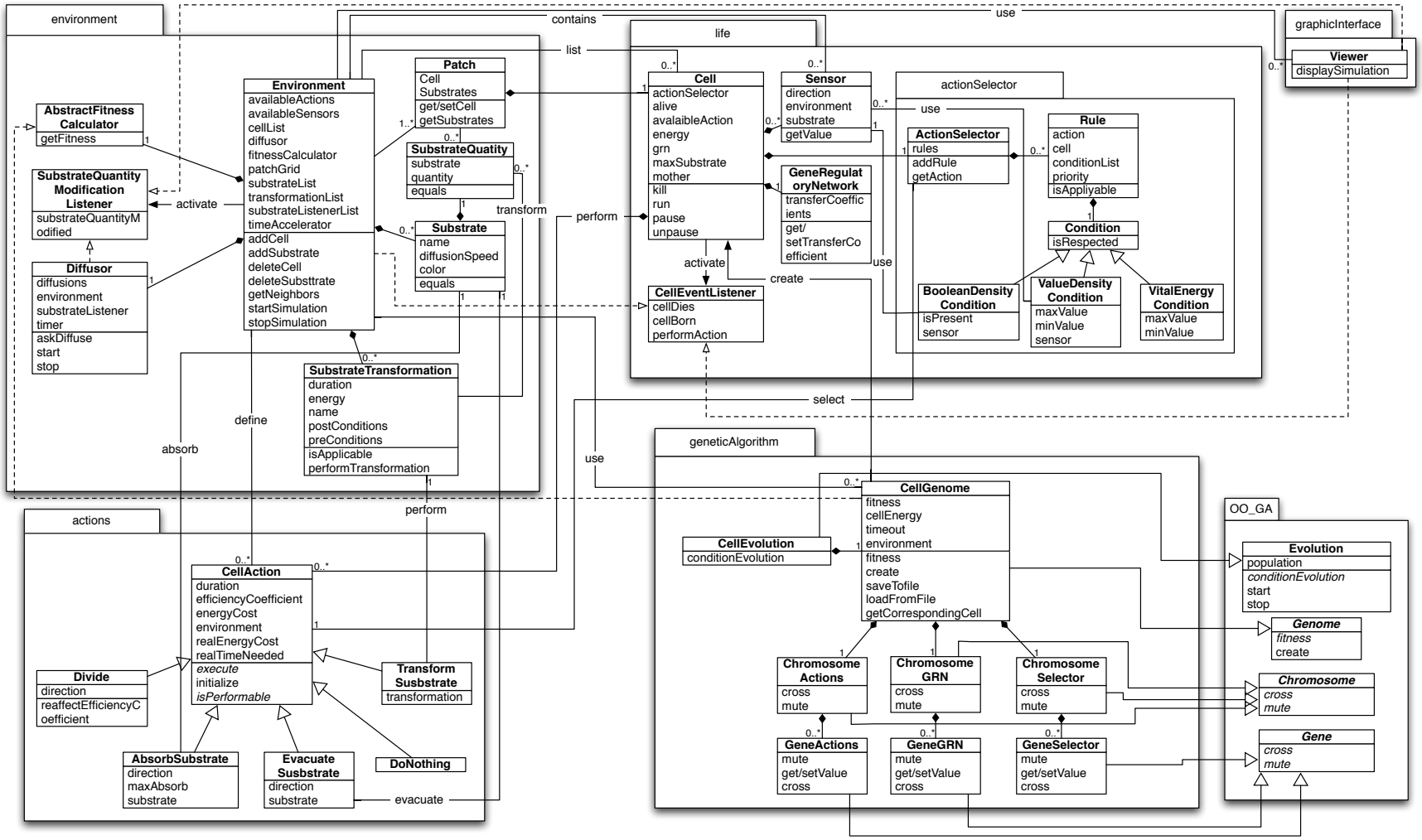


FIGURE B.1 – *Diagramme de classes de l'implantation de notre modèle. L'application est construite pour accueillir le plus facilement possible de nouvelles actions et pour pouvoir brancher une interface graphique facilement et complètement découplée du simulateur.*

C

Génome de l'organisme auto-alimenté

C.1 Organe Producteur-Consommateur PC_1

Action	Activation	Action	Activation
Divide to NorthEast	Activée	Absorb M from North	Désactivée
Divide to NorthWest	Activée	Absorb M from South	Activée
Divide to SouthEast	Activée	Absorb M from East	Activée
Divide to SouthWest	Désactivée	Absorb M from West	Activée
Transform M into Energy	Activée	Evacuate B to North	Activée
Transform $A \rightarrow B$	Activée	Evacuate B to South	Désactivée
Absorb A from North	Activée	Evacuate B to East	Désactivée
Absorb A from South	Désactivée	Evacuate B to West	Activée
Absorb A from East	Désactivée	Evacuate A to North	Activée
Absorb A from West	Désactivée	Evacuate A to South	Activée
Absorb B from North	Activée	Evacuate A to East	Activée
Absorb B from South	Activée	Evacuate A to West	Activée
Absorb B from East	Désactivée	Do nothing	Activée
Absorb B from West	Activée		

TABLE C.1 – *Chromosome de l'activation des actions de l'organe PC_1.*

TABLE C.2 – *Chromosome du système de sélection d'action* PC_1.

#	Règle			Priorité
(A)	(M_North=true) and (M_South=false) and (M_West=true) and (B_South=false)	→	(Divide NE)	(33)
(B)	(A_North=true) and (A_East=false) and (B_South=false) and (B_East=false)	→	(Divide NW)	(8)
(C)	(A_North=true) and (A_East=true) and (A_West=true) and (M_South=true) and (M_East=true) and (B_West=false)	→	(Divide SE)	(0)
(D)	(A_East=false) and (M_North=true) and (B_East=false) and (B_West=false)	→	Transform A->B)	(49)
(D)		→	(Transform M->)	(4)
(E)	(A_South=false) and (A_East=false) and (A_West=false) and (M_North=true) and (M_South=true) and (M_East=true)	→	(Absorb_A_North)	(20)
(F)	(A_West=false) and (M_West=true) and (B_North=false) and (B_South=false) and (B_East=false)	→	(Absorb_A_East)	(18)
(G)	(A_South=false) and (A_East=true) and (M_North=true) and (M_South=true) and (M_West=false) and (B_North=true)	→	(Absorb_B_North)	(46)
(H)	(A_South=false) and (A_East=true) and (M_West=false) and (B_North=false)	→	(Absorb_B_South)	(16)
(I)	(A_North=false) and (A_South=true) and (M_North=true) and (M_South=true) and (B_North=true) and (B_East=true) and (B_West=true)	→	(Absorb_B_West)	(38)
(J)		→	(Absorb_M_South)	(0)
(K)		→	(Absorb_M_East)	(1)
(L)		→	(Absorb_M_West)	(2)
(M)	(A_East=true) and (M_North=false) and (M_East=true) and (B_West=true)	→	(Evacuate_A_North)	(27)
(N)	(B_West=false)	→	(Evacuate_A_West)	(21)
(O)	(A_North=true) and (A_South=true) and (M_North=false) and (M_East=true) and (M_West=false)	→	(Evacuate_B_North)	(18)
(P)	(A_South=false) and (A_East=false) and (A_West=false) and (M_North=true) and (M_South=true) and (B_North=false) and (B_East=false)	→	(Evacuate_B_South)	(27)
(Q)	(A_South=true) and (A_West=true) and (M_East=true) and (M_West=false) and (B_North=false) and (B_South=true)	→	(Evacuate_B_East)	(29)
(R)	(A_North=true) and (A_South=false) and (M_South=true) and (M_West=true) and (B_North=false) and (B_East=true)	→	(Evacuate_B_West)	(1)
(S)	(M_South=false)	→	(DoNothing)	(0)

	(Absorb_B_West)	(Divide NE)	(Absorb_A_East)	(Transform A->B)	(Evacuate_B_East)	(Evacuate_B_North)	(Absorb_B_South)	(Evacuate_A_West)	(Absorb_A_North)	(Absorb_M_East)	(Transform M->)	(Absorb_M_West)	(DoNothing)	(Evacuate_B_South)	(Absorb_M_South)	(Evacuate_A_North)	(Absorb_B_North)	(Divide SE)	(Evacuate_B_West)	(Divide NWT)
(Absorb_B_West)	0	0,16	0,48	0,36	0,28	0,81	0,08	0,84	0,51	0,55	0,17	0,1	0,45	0,65	0,29	0,2	0,42	0,35	0,3	0,76
(Divide NE)	0,68	0	0	0,39	0,79	0,41	0,5	0,62	0,86	0,54	0,94	0,32	0,9	0,42	0,52	0,42	0,47	0,85	0,88	0,61
(Absorb_A_East)	0,49	0,71	0	0,23	0,22	0,64	0,86	0,92	0,82	0,51	0,89	0,8	0,78	0,16	0,47	0,36	0,76	0,86	0,16	0,28
(Transform A->B)	0,66	0,25	0,28	0	0,11	0,69	0,49	0,44	0,34	0,9	0,43	0,42	0,67	0,23	0,06	0,37	0,62	0,54	0,19	0,59
(Evacuate_B_East)	0,62	0,98	0,57	0,56	0	0,7	0,19	0,58	0,76	0,79	0,58	0,81	0,65	0,02	0,55	0,54	0,61	0,42	0,67	0,3
(Evacuate_B_North)	0,92	0,12	0,79	0,46	0,35	0	0,76	0,42	0,45	0,92	0,61	0,24	0,77	0,01	0,77	0,78	0,24	0,98	0,1	0,41
(Absorb_B_South)	0,48	0,13	0,31	0,51	0,7	0,7	0	0,46	0,47	0,78	0,16	0,34	0,57	0,7	0,86	0,82	0,47	0,58	0,3	0,06
(Evacuate_A_West)	0,27	0,75	0,55	0,68	0,05	0,82	0,41	0	0,78	0,63	0,33	0,26	0,51	0,52	0,56	0,21	0,79	0,31	0,27	0,53
(Absorb_A_North)	0,52	0,92	0,56	0,84	0,86	0,05	0,54	0,44	0	0,57	0,11	0,01	0,83	0,84	0,6	0,33	0,76	0,47	0,81	0,96
(Absorb_M_East)	0,62	0,29	0,41	0,84	0,53	0,16	0,69	0,69	0,53	0	0,77	0,14	0,27	0	0,49	0,51	0,08	0,69	0,17	0,58
(Transform M->)	0,25	0,28	0,62	0,43	0,74	0,39	0,35	0,91	0,4	0,76	0	0,13	0,72	0,4	0,5	0,39	0,72	0,41	0,62	0,91
(Absorb_M_West)	0,86	0,17	0,23	0,74	0,79	0,58	0,72	0,7	0,28	0,84	0,44	0	0,78	0,12	0,76	0,7	0,16	0,3	0,72	0,37
(DoNothing)	0,42	0,47	0,4	0,46	0,15	0,65	0,27	0,5	0,22	0,25	0,47	0,24	0	0,76	0,6	0,41	0,54	0,56	0,79	0,72
(Evacuate_B_South)	0,51	0,44	0,95	0,08	0,78	0,72	0,68	0,78	0,21	0,18	0,59	0,22	0,58	0	0,49	0,92	0,97	0,82	0,58	0,6
(Absorb_M_South)	0,1	0,47	0,84	0,99	0,31	0,52	0,5	0,77	0,24	0,98	0,17	0,75	0,11	0,42	0	0,83	0,67	0,43	0,57	0,47
(Evacuate_A_North)	0,32	0,38	0,41	0,9	0,39	0,25	0,58	0,65	0,45	0,83	0,31	0,52	0,46	0,75	0,81	0	0,44	0,6	0,3	0,45
(Absorb_B_North)	0,32	0,81	0,96	0,78	0,96	0,44	0,54	0,26	0,07	0,45	0,42	0,07	0,63	0,42	0,36	0,98	0	0,33	0,4	0,21
(Divide SE)	0,6	0,75	0,6	0,61	0,39	0,75	0,24	0,49	0,48	0,64	0,37	0,66	0,35	0,51	0,73	0,8	0,55	0	0,22	0,81
(Evacuate_B_West)	0,58	0,14	0,77	0,37	0,2	0,63	0,58	0,41	0,42	0,31	0,79	0,31	0,07	0,41	0,38	0,32	0,71	0,75	0	0,62
(Divide NWT)	0,39	0,95	0,78	0,11	0,41	0,06	0,32	0,21	0,63	0,63	0,48	0,6	0,33	0,86	0,83	0,29	0,17	0,7	0,43	0

TABLE C.3 – *Chromosome du réseau de régulation de l'organe* PC_1.

C.2 Organe Producteur-Consommateur PC_2

Action	Activation	Action	Activation
Divide to NorthEast	Désactivée	Absorb M from North	Désactivée
Divide to NorthWest	Désactivée	Absorb M from South	Désactivée
Divide to SouthEast	Activée	Absorb M from East	Activée
Divide to SouthWest	Activée	Absorb M from West	Activée
Transform M into Energy	Activée	Evacuate B to North	Désactivée
Transform $B \rightarrow A$	Activée	Evacuate B to South	Activée
Absorb A from North	Activée	Evacuate B to East	Désactivée
Absorb A from South	Activée	Evacuate B to West	Activée
Absorb A from East	Activée	Evacuate A to North	Activée
Absorb A from West	Activée	Evacuate A to South	Désactivée
Absorb B from North	Désactivée	Evacuate A to East	Désactivée
Absorb B from South	Activée	Evacuate A to West	Activée
Absorb B from East	Désactivée	Do nothing	Activée
Absorb B from West	Activée		

TABLE C.4 – *Chromosome de l'activation des actions de l'organe PC_2.*

#	Règle			Priorité
(A)	(A_North=false) and (A_South=true) and (M_West=false) and (B_South=false)	→	(Divide SE)	(2)
(B)	(M_North=false) and (M_South=true) and (B_West=false)	→	(Divide SW)	(36)
(C)		→	(Transform M->)	(11)
(D)	(A_East=false) and (B_North=false) and (B_East=false)	→	(Transform B->A)	(24)
(E)	(A_North=false) and (A_South=false) and (A_East=true) and (B_East=false) and (B_West=false)	→	(Absorb_A_North)	(24)
(F)	(A_North=true) and (M_South=false) and (M_East=false) and (B_North=false) and (B_West=true)	→	(Absorb_A_South)	(22)
(G)	(A_North=true) and (M_North=false) and (M_West=false) and (B_South=false)	→	(Absorb_A_East)	(37)
(H)	(A_East=false) and (M_North=true) and (M_East=true) and (M_West=true) and (B_East=false) and (B_West=false)	→	(Absorb_A_West)	(49)
(I)	(A_North=true) and (A_East=true) and (M_North=false) and (M_South=true) and (M_West=true) and (B_South=false) and (B_East=false)	→	(Absorb_B_South)	(23)
(J)	(A_South=false) and (A_West=false) and (M_North=true) and (M_South=true)	→	(Absorb_B_West)	(19)
(K)		→	(Absorb_M_East)	(2)
(L)		→	(Absorb_M_West)	(2)
(M)	(A_East=false) and (M_North=true) and (M_South=true) and (M_West=true) and (B_South=false)	→	(Evacuate_A_North)	(21)
(N)	(A_North=false) and (A_West=true) and (M_South=true) and (B_North=true) and (B_South=false) and (B_East=false)	→	(Evacuate_A_West)	(29)
(O)	(A_East=true) and (A_West=false) and (M_East=false) and (B_North=true) and (B_South=false) and (B_East=true) and (B_West=false)	→	(Evacuate_B_South)	(7)
(P)	(A_North=true) and (A_South=true) and (A_East=false) and (A_West=true) and (M_North=true) and (M_West=false) and (B_North=false) and (B_West=true)	→	(Evacuate_B_West)	(26)
(Q)		→	(DoNothing)	(0)

TABLE C.5 – *Chromosome du système de sélection d'action, produit du génome de l'organe* PC_2.

	(Evacuate_B_South)	(Divide SW)	(Transform M->)	(DoNothing)	(Absorb_M_East)	(Absorb_A_West)	(Absorb_M_West)	(Absorb_B_West)	(Absorb_A_East)	(Transform B->A)	(Evacuate_A_North)	(Absorb_A_North)	(Absorb_A_South)	(Evacuate_B_West)	(Divide SE)	(Absorb_B_South)	(Evacuate_A_West)
(Evacuate_B_South)	0	0,94	0,6	0,81	0,62	0,91	0,72	0,08	0,56	0,48	0,73	0,16	0,28	0,81	0,89	0,33	0,58
(Divide SW)	0,93	0	0,52	0,75	0,15	0	0,77	0,5	0,47	0,49	0,85	0,64	0,26	0,18	0,16	0,81	0,45
(Transform M->)	0,9	0,42	0	0,11	0,51	0,39	0,13	0,71	0,62	0,47	0,81	0,79	0,22	0,64	0,26	0,48	0,19
(DoNothing)	0,44	0,41	0,12	0	0,01	0,3	0,72	0,83	0,38	0,23	0,56	0,51	0,44	0,89	0,06	0,98	0,92
(Absorb_M_East)	0,22	0,31	0,47	0,34	0	0,91	0,51	0,5	0,64	0,56	0,62	0,77	0,61	0,05	0,64	0,07	0,75
(Absorb_A_West)	0,49	0,2	0,83	0,53	0,1	0	0,56	0,57	0,84	0,54	1	0,45	0,2	0,76	0,57	0,66	0,35
(Absorb_M_West)	0,52	0,8	0,57	0,96	0,73	0,39	0	0,9	0,55	0,29	0,35	0,43	0,35	0,57	0,34	0,48	0,26
(Absorb_B_West)	0,1	0,11	0,75	0,63	0,1	0,64	0,18	0	0,48	0,99	0,08	0,72	0,74	0,36	0,42	0,67	0,28
(Absorb_A_East)	0,74	0,8	0,48	0,93	0,41	0,49	0,8	0,89	0	0,7	0	0,46	0,4	0,38	0,9	0,25	0,62
(Transform B->A)	0,41	0,38	0,55	0	0,35	0,55	0,38	0,07	0,11	0	0,17	0,27	0,79	0,54	0,57	0,08	0,91
(Evacuate_A_North)	0,17	0,82	0,5	0,86	0,89	0,33	0,04	0,28	0,47	0,11	0	0	0,49	0,65	0,44	0,14	0,72
(Absorb_A_North)	0,8	0,25	0,24	0,33	0,56	0,54	0,72	0,57	0,68	0,9	0,32	0	0,5	0,33	0,52	0,68	0,26
(Absorb_A_South)	0,89	0,9	0,43	0,51	0,63	0,8	0,58	0,57	0,52	0,33	0,58	0,91	0	0,78	0,81	0,41	0,57
(Evacuate_B_West)	0,44	0,93	0,93	0,72	0,4	0,63	0,69	0,97	0,37	0,51	0,82	0,86	0,46	0	0,6	0,35	0,44
(Divide SE)	0,57	0,43	0,19	0,34	0,22	0,84	0,19	0,28	0,85	0	0,49	0,21	0,73	0,05	0	0,06	0,02
(Absorb_B_South)	0,5	0,78	0,98	0,53	0,52	0,99	0,84	0,34	0,63	0,63	0,67	0,62	0,77	0,38	0,54	0	0,7
(Evacuate_A_West)	0,17	0,11	0,61	0,97	0,86	0,17	0,41	0,27	0,62	0,71	0,59	0,61	0,51	0,19	0,46	0,94	0

TABLE C.6 – *Chromosome du réseau de régulation de l'organe PC_2.*