Approche évolutionniste pour la détection des collisions au sein d'environnements virtuels denses

Lionnel Joussemet

ACADEMIE DE MONTPELLIER II

UNIVERSITE MONTPELLIER II

- Sciences et Techniques du Languedoc -

THESE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE MONTPELLIER II

Discipline : **Génie Informatique, Automatique et Traitement du Signal**
Formation doctorale : **Systèmes Automatiques et Microélectroniques**
Ecole doctorale : **Informatique, Structures, Systèmes**

présentée et soutenue publiquement par

Lionnel JOUSSEMET

le 14 décembre 2006

Titre :

Approche évolutionniste pour la détection des collisions au sein d'environnements virtuels denses

JURY

M. Claude Andriot	Expert senior CEA / LIST	Examinateur
Mme. Sabine Coquillart	Directeur de recherches / INRIA	Examinateur
M. André Crosnier	Professeur Univ. Montpellier II / LIRMM	Directeur de thèse
M. Georges Dumont	Maître de conférences / ENS-Cachan	Rapporteur
M. Abderrahmane Kheddar	Professeur, co-directeur AIST / ISRI	Rapporteur
M. Jérôme Perret	Directeur général HAPTION	Examinateur

Table des matières

Introduction générale

Position du problème et contexte

L'évolution considérable des systèmes informatiques à laquelle nous avons assisté au cours de cette dernière décennie, tant au niveau de la puissance des unités de calcul proprement dites que des systèmes de restitution sensorielle, a permis l'introduction des technologies numériques au sein d'un grand nombre d'applications et de métiers. Initialement fondées sur des techniques et modes opératoires plus classiques, bon nombre de spécialités voient ainsi mis à leur disposition de nouveaux outils générant bien souvent des gains notables, tant au niveau de la productivité que de la qualité des résultats obtenus.

Au vu de cet engouement pour les technologies numériques, désormais à la porté de tous, la réalité virtuelle, initialement cantonnée aux mondes du cinéma et du jeu vidéo, s'est ainsi vu attribuer un nouveau statut d'***outil de simulation***. Le fait de simuler au sein d'un environnement numérique des scènes réelles présente en effet bien des avantages. Tout d'abord, le fait de ***dématérialiser*** les modes opératoires est susceptible d'engendrer une diminution notable des coûts, notamment dans le cadre d'application mettant en scène des éléments coûteux et/ou encombrants dans la réalité. De plus, contrairement à une réalité dont de nombreuses composantes sont souvent mal voir non maîtrisées, une scène virtuelle présente l'avantage d'être fortement ***paramétrable*** et ***répétable***, tant au niveau de sa définition géométrique que de son comportement. Enfin, la réalité virtuelle permet de faciliter les tâches de l'opérateur via l'introduction de ***nouvelles interfaces*** entre celui-ci et l'environnement simulé avec lequel il interagit.

Cependant, certaines limites sont encore observables quant à l'utilisation de la réalité virtuelle dans le cadre de certains types d'applications. Ces limitations peuvent être de plusieurs natures et vont de restrictions de la complexité des scènes simulées au niveau de réalisme offert par les différents types d'interfaces proposés. Aussi, une activité de recherche importante, menée par une large communauté de scientifiques, vise à réduire l'ensemble de ces limitations au vu de permettre une utilisation de plus en plus large de ces nouvelles technologies, qui donneront naissance à une partie des outils de demain. Cette thèse s'inscrit dans cette dynamique.

Contributions principales

Dans le cadre des travaux présentés au travers de ce manuscrit, nous nous intéressons aux problématiques inhérentes aux modules algorithmiques mis en œuvre pour la simulation du comportement des objets, et, plus précisément, aux processus visant à la détection des collisions entre ces derniers. La principale contribution issue de ces travaux

réside dans l'élaboration et l'étude d'un nouvel algorithme ayant pour fonction première l'accélération de ce processus de détection des collisions.

Fondé sur l'application de techniques inhérentes à l'algorithmique évolutionniste, cet algorithme, baptisé ESPIONS, peut se voir utilisé de différentes manières. On peut cependant le définir comme une procédure visant à identifier des champs de distances minimales entre les objets simulés. Il présente, entre autres, l'avantage d'être utilisable dans le cadre de simulations mettant en scènes des objets de différentes natures, que ce soit au niveau de leur définition géométrique (polyèdres, nuages de points, etc.), de leurs propriétés topologiques (convexité, échelle, etc.) ou encore de leur capacité à se déformer. Des résultats expérimentaux semblent également démontrer l'aptitude d'ESPIONS à accélérer le processus de détection des collisions, particulièrement dans le cadre de simulations mettant en scène des objets présentant une complexité géométrique importante.

Au delà d'une description détaillée du fonctionnement de l'algorithme proposé, nous nous intéressons également à la manière de l'intégrer au sein d'un processus de simulation (on introduira, dans la suite, la notion de moteur physique). Pour ce faire, nous proposons différents modes opératoires permettant, au delà d'une cohabitation entre les différents modules algorithmiques mis en œuvres, une exploitation optimale des plateformes matérielles munies de plusieurs unités de calcul.

L'une des problématiques inhérentes aux approches évolutionnistes réside dans la complexité du paramétrage des algorithmes basés sur ces techniques. Aussi, nous proposons également dans ce rapport une étude théorique de l'algorithme ESPIONS visant à caractériser l'influence du paramétrage de celui-ci sur certaines de ses caractéristiques.

Organisation du mémoire

Dans un premier temps, nous proposons au travers du chapitre 1 une présentation du contexte dans lequel cette thèse s'inscrit. Nous y décrivons plusieurs applications concrètes de la réalité virtuelle dans deux domaines : l'industrie et la médecine. Nous proposons également une présentation générale des différentes composantes qui composent une plateforme de réalité virtuelle, en nous attardant particulièrement sur les principes fondamentaux qui régissent le fonctionnement d'un moteur physique. Enfin, nous discutons des problématiques inhérentes à la détection des collisions en temps réel.

Le chapitre 2 constitue un état de l'art au travers duquel nous nous intéressons aux techniques de détection des collisions proposées dans la littérature scientifique. Celui-ci s'articule autour des différentes problématiques auxquelles nous nous sommes particulièrement intéressés dans le cadre de cette thèse : complexité de la scène simulée, propriétés

géométriques des objets manipulés, …. Nous y traitons, entre autre, des techniques visant à accélérer la détection des collisions au sein d'environnements complexes, des approches adaptés à la manipulation d'objets déformables, et de l'influence des propriétés géométriques des objets simulés (natures, convexité, …) sur les choix algorithmiques réalisés quant à la mise en œuvre du processus de détection des collisions.

Dans le chapitre 3, nous proposons une présentation des concepts fondamentaux inhérents aux approches évolutionnaires (ou évolutionnistes). Nous y abordons les principes généraux qui régissent ces approches avec l'introduction des différentes entités manipulées dans ce contexte (individus, etc.). Nous nous intéressons également aux modes opératoires mis en œuvre quant à la manipulation de ces entités. Enfin, des notions plus avancées telles que la parallélisation de telles approches sont abordées.

Dans le chapitre 4, une présentation détaillée de l'algorithme ESPIONS est proposée. Nous y décrivons l'ensemble des composantes de l'algorithme, tant au niveau des entités élémentaires utilisées dans ce contexte, que des opérateurs permettant leur manipulation. Plusieurs formalismes sont d'ailleurs introduits afin d'illustrer l'aptitude d'ESPIONS à fonctionner au sein d'environnements composés d'éléments géométriques de différentes natures. Le schéma de fonctionnement général de l'algorithme est également introduit, avec une description précise de chacune des étapes qui le composent. Enfin, différentes architectures sont proposées quant à l'intégration d'ESPIONS au sein de processus de simulation, avec une réflexion sur l'exploitation de l'éventuelle multiplicité des unités de calcul mises à disposition.

Le chapitre 5 se décompose en deux parties. La première consiste en une étude expérimentale visant à analyser le comportement de l'algorithme en fonction des paramètres qui lui sont spécifiés. Cette étude a pour but de caractériser l'influence de chacun de ces paramètres sur différentes caractéristiques de l'algorithme. Dans la deuxième partie du chapitre, nous présentons plusieurs implémentations de notre algorithme dans différents contextes. Ainsi, plusieurs modes d'intégration d'ESPIONS au sein de processus de simulation sont mis en œuvre. Les résultats obtenus dans le cadre de chacune de ces expérimentations sont analysés et discutés afin d'évaluer la pertinence d'une utilisation d'ESPIONS dans chacun des cas abordés. Une implémentation visant à illustrer la parallélisation de l'algorithme est également présentée.

En conclusion, nous dressons un récapitulatif des principaux éléments abordés dans ce mémoire et proposons plusieurs perspectives et voies d'investigation susceptibles de donner lieu à une poursuite des travaux réalisés dans le cadre de cette thèse.

Chapitre 1

La réalité virtuelle : Enjeux et Limites

1.1 Introduction

Dans le cadre de l'introduction générale, nous avons proposé une présentation succincte du contexte dans lequel s'inscrivent les travaux réalisés dans le cadre de cette thèse. Nous y avons introduit le fait que les technologies inhérentes à la réalité virtuelle pouvaient se voir exploitées dans bon nombre d'applications et métiers initialement fondés sur des modes opératoires plus traditionnels.

Au travers de ce premier chapitre, nous souhaitons détailler d'avantages le contexte dans lequel se situe l'étude à laquelle nous nous intéressons dans ce mémoire. Pour cela, nous proposons, dans un premier temps, de nous intéresser à deux des principaux domaines d'application de la réalité virtuelle que sont l'industrie et la médecine. Notons que nous ne considérons pas là la réalité virtuelle dans sa globalité, mais seulement ses applications dans le cadre de ***simulations interactives*** mettant en scène des acteurs humains, que nous qualifierons d'***opérateurs***, et des environnements virtuels complexes. Dans un deuxième temps, nous définissons plus précisément le fonctionnement d'une simulation interactive via une présentation de chacune des entités qui la composent. Nous introduisons par la même les principes fondamentaux qui régissent le fonctionnement d'un moteur physique. Enfin, nous nous attardons sur les problématiques inhérentes à la détection des collisions en temps réel, ces mêmes problématiques auxquelles nous souhaitons apporter notre contribution au travers de ce mémoire.

1.2 Domaines d'application et enjeux

Comme il l'a été dit en introduction, nombreux sont les domaines dans lesquels des outils de simulation interactive ont été mis en place. Aussi, dans cette partie, nous n'aurons pas pour ambition d'énumérer l'ensemble des applications de simulation interactive utilisées à ce jour, mais simplement de dresser une liste non exhaustive des différents domaines d'application et de l'apport attendu quant à la mise en place de ces nouveaux outils.

1.2.1 L'industrie

Ce premier domaine d'application est très certainement le plus prédisposé à la mis en place d'outils de simulation interactive. En effet, c'est à plusieurs niveaux que ces outils peuvent être introduits et s'avérer efficaces. Ci-dessous, nous dressons une liste des différentes tâches concernées.

Conception

La phase de prototypage consiste à contrôler la validité des ensembles de pièces constituant le produit final en termes de (dé)montabilité. Jusqu'à présent, cette phase nécessite la fabrication de ***maquettes physiques*** fidèles aux côtes spécifiées au niveau de la conception (CAO), afin de pouvoir valider manuellement la pertinence de l'ensemble et élaborer les trajectoires de montage. Ce mode opératoire s'avère souvent coûteux et contraignant, que ce soit en termes de temps ou de coûts, la fabrication des maquettes étant souvent sous-traitée à des sociétés annexes. Dans ce cas précis, la simulation interactive va s'avérer être un outil particulièrement adapté, puisqu'elle va permettre à l'opérateur de vérifier la validité de l'ensemble via l'utilisation d'une ***maquette numérique*** au sein de laquelle seront directement injectées les informations issues de la CAO. Ainsi, l'opérateur va pouvoir manipuler à sa guise les différents éléments d'un ensemble en prenant en considération les contraintes liées à la géométrie et la cinématique de celui-ci. De plus, dans le cas où des modifications s'avèreraient nécessaires, la maquette numérique pourra dans tous les cas être directement modifiée afin de correspondre au nouvel ensemble ; contrairement à une maquette physique sur laquelle des opérations de rectification vont être à effectuer, voir un ré usinage complet.

La figure 1.1 illustre ce type d'application. On y observe un opérateur interagissant avec un environnement virtuel via deux interfaces haptiques. La plateforme présentée permet une immersion visuelle de ce dernier grâce à une restitution stéréoscopique active réalisée sur un système de type bi-plan. L'opérateur, dont la position de la tête est traquée en temps réel, peut à sa guise modifier son point de vue afin d'apprécier au mieux la pertinence des gestes qu'il élabore quant à la réalisation de la tâche étudiée (Dans ce cas précis : l'assemblage du système de motorisation de la vitre à l'intérieur d'une portière de *Laguna II RENAULT*)

Formation et entraînement

La formation est une composante nécessaire à la mise en place d'une chaîne de fabrication, particulièrement dans le cadre de grandes séries. Elle s'avère également nécessaire dans d'autres domaines tels que la maintenance. Le but de la formation est de permettre aux différents intervenants de réaliser au mieux les tâches qui leur sont imparties afin d'améliorer non seulement leur productivité, mais également leur confort. La réalité virtuelle va permettre de simuler virtuellement ces phases de fabrication ou de maintenance. Ainsi, sur un site équipé des calculateurs et des interfaces sensorielles adéquates, les opérateurs vont pouvoir être formés à tous les gestes techniques sans avoir à immobiliser le site *réel* d'intervention. De plus, le formateur va bénéficier d'outils lui permettant d'agir

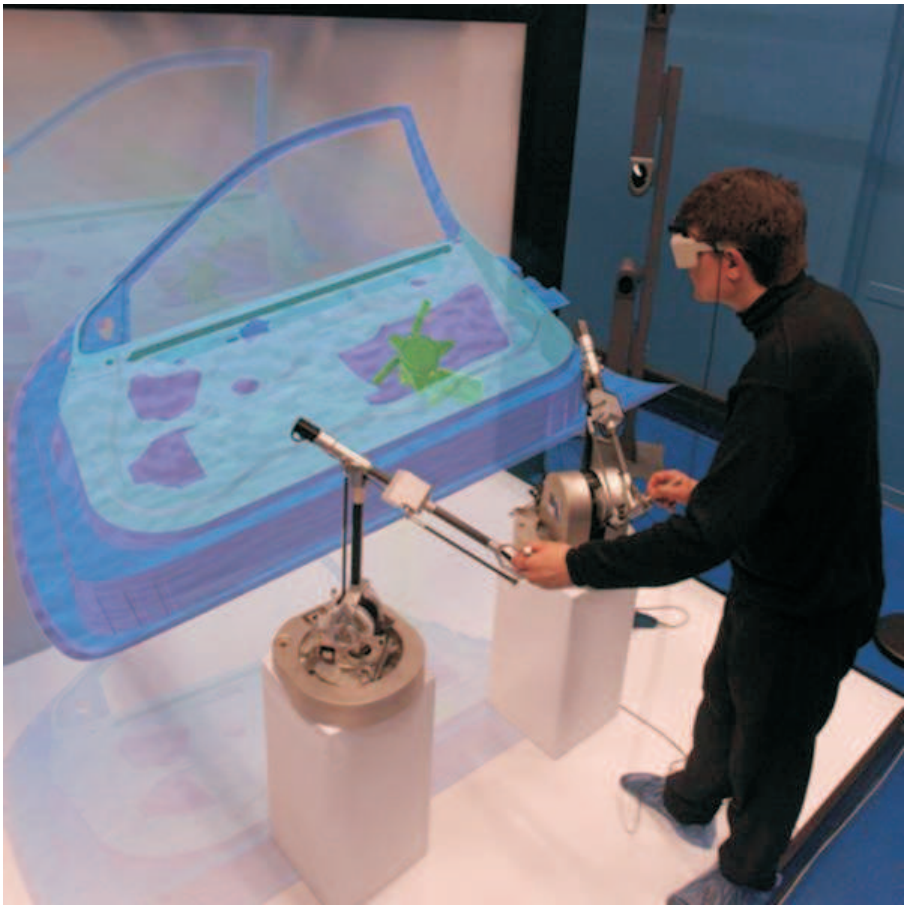


FIG. 1.1 – Plateforme FAR (CEA-LIST 2002)

sur l'environnement virtuel et ainsi faciliter l'apprentissage de ses élèves.

Plus généralement, la réalité virtuelle va permettre l'élaboration de simulations interactives visant à enseigner ou parfaire les gestes et techniques nécessaires à un opérateurs dans un contexte donné. Aussi, plusieurs simulateurs sont déjà opérationnels quant à la formation de pilotes d'avions ou encore d'astronautes.

Ergonomie

Des critères ergonomiques sont souvent considérés dans le cadre de l'élaboration de produits industriels. Le but de ces derniers est de faciliter l'utilisation du produit en prenant en considération des propriétés propres à l'utilisateur final. Là encore, les simulations interactives peuvent s'avérer très utiles, particulièrement en mode immersif. En effet, le concepteur va pouvoir directement prendre en considération l'ergonomie du produit via une représentation naturelle de celui-ci. Il n'est plus contraint d'utiliser des maquettes physiques.

La figure 1.2 illustre ce type d'applications. On y observe, grâce à une restitution infographique, l'intérieur d'un véhicule *RENAULT* alors en cours de conception, tel qu'il sera vu par le conducteur. Cette simulation permet non-seulement d'évaluer l'accessibilité des commandes mises à disposition, mais également d'apprécier le champs de vision offert par l'habitacle quant à l'observation de l'environnement extérieur.

FIG. 1.2 – Simulateur de conduite présentant l'ensemble du tableau de bord (RENAULT)

Démonstration de maintenance

Cette dernière application est un besoin inhérent à l'industrie aéronautique. En effet, contrairement à l'industrie automobile où les principaux apports de la réalité virtuelle se situent au niveau de l'élaboration du produit, les industriels de l'aéronautique ont de plus besoin de démontrer à leur clientèle les prédispositions à la maintenance des produits qu'ils proposent afin d'assurer des temps d'immobilisation réduits et, par conséquent, une meilleure rentabilité de ces derniers. Des outils de simulation interactive vont permettre à l'industriel de démontrer au client la maintenabilité des composants de ses appareils sans avoir à immobiliser l'un d'entre eux.

1.2.2 La médecine

En ce qui concerne la médecine, des outils de simulation opérationnels commencent à voir le jour. De par la difficulté de simuler l'ensemble des caractéristiques géométriques et mécaniques du corps humain, ces outils sont généralement spécialisés pour un type d'intervention particulier (opérations du foie, chirurgie dentaire, ...). Une activité de recherche très active autour de ces problématiques laisse entrevoir la mise en place, dans un futur proche, d'outils de simulations génériques, permettant la manipulation de l'ensemble des composantes du corps humain. Dans la suite, nous présentons deux types d'utilisation des outils de simulation dans ce contexte : la formation et la rééducation.

Formation

L'apprentissage du geste chirurgical constitue une part importante de la formation des futurs chirurgiens. Hors, ce type d'exercices nécessite l'utilisation de corps humains, ce qui pose plusieurs problèmes, tant au niveau de la mise à disposition de ces derniers que sur le plan étique. Un outil permettant de réaliser des gestes chirurgicaux sur des êtres humains virtuels suffisamment réalistes, tant au niveau de leur mécanique que de la représentation de leurs tissus, serait un réel apport dans ce contexte. De plus, un tel outil pourrait offrir des fonctionnalités fort utiles, telles que l'intégration de pathologies, éventuellement orphelines, au sein de l'être simulé, afin de proposer aux étudiants des cas d'études spécifiques, souvent difficiles à reproduire dans la réalité. La figure 1.3 est une illustration de ce type d'application. On y observe un simulateur chirurgical dédié à la formation des médecins aux gestes inhérents aux échographies tels que l'insertion d'aiguilles.

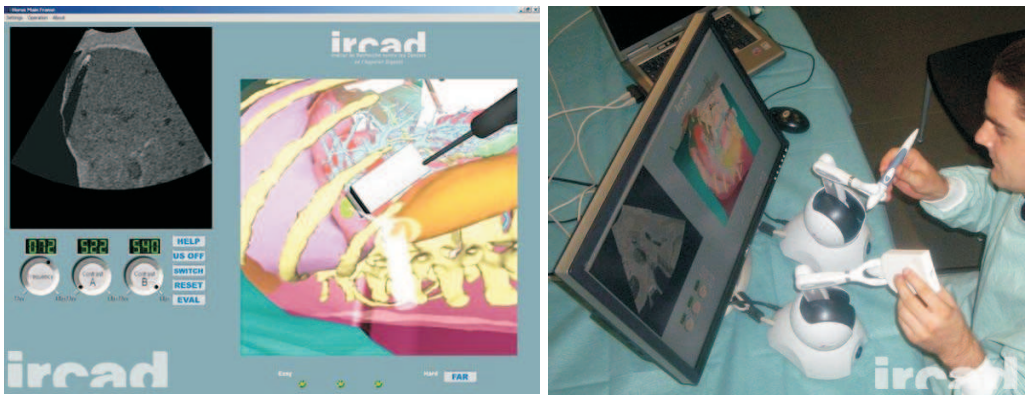


FIG. 1.3 – Simulateur pour la formation des médecins aux gestes inhérents aux échographies tels que l'insertion d'aiguilles (IRCAD - Projet HORUS)

Rééducation

Des outils de simulations interactives peuvent également s'avérer utiles dans le cadre de la rééducation de personnes présentant des déficiences physiques. En effet, plusieurs projets de recherche visent à introduire des systèmes de simulation interactive, généralement équipés de systèmes à retour d'effort, au sein de programmes de rééducations. L'idée est de mettre à disposition du rééducateur un ensemble d'atelier permettant la sollicitation, et ainsi la stimulation, des membres à rééduquer chez le patient. La forte paramétrabilité de l'environnement virtuel permet de mettre en œuvre des exercices spécifiques parfaitement adaptés à la pathologie et à la morphologie du patient.

1.3 Concepts de base

1.3.1 Principes de fonctionnement d'une simulation interactive

La réalité virtuelle immersive (RV) consiste en la simulation interactive du monde réel au sein d'un environnement numérique. Elle est fonctionnellement décomposable en deux parties (Figure 1.4). La première, que nous nommerons *moteur physique*, est assimilable à un logiciel. Elle a pour fonction de régir en temps réel l'évolution des entités de l'environnement virtuel selon les lois de la physique, ainsi que leurs interactions. Deux types d'interactions peuvent être identifiés : celles liées aux contraintes cinématiques inter-objets, assimilables à des liaisons mécaniques (pivot, prismatique,...), et celles inhérentes aux collisions entre ces derniers. La seconde composante d'une plate-forme RV consiste en l'interfaçage de l'environnement virtuel avec le ou les opérateurs participant à la simulation. Il existe plusieurs types d'interfaces, chacune destinée à la stimulation d'un des sens du ou des opérateurs, on parle d'interfaces visuelles, auditives et haptiques.

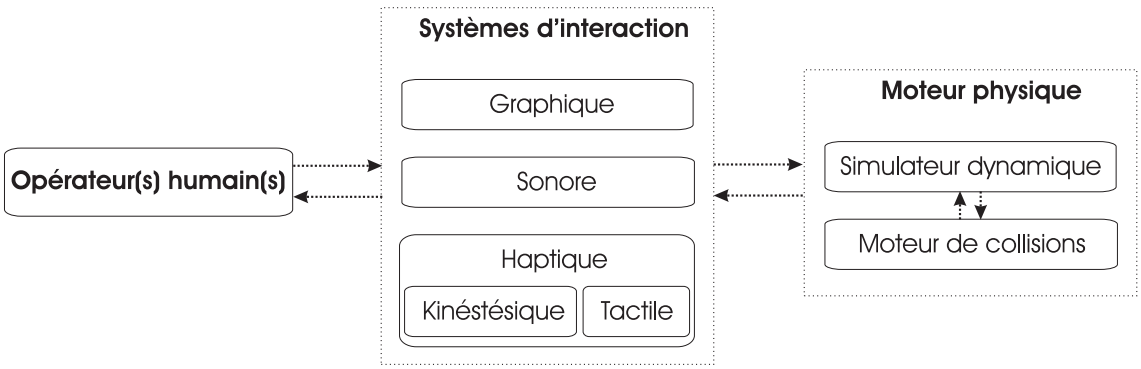


FIG. 1.4 – Architecture fonctionnelle d'une plateforme RV

1.3.2 Fonctionnement d'un moteur physique

Dans cette architecture, le moteur physique a pour rôle de régir l'évolution d'un monde virtuel composé d'objets ayant des propriétés dynamiques et géométriques. Cette tâche est décomposable en deux modules : la gestion du mouvement des objets au sein du monde virtuel et la gestion des collisions entre ces derniers.

Simulateur dynamique

Le but de ce premier module est de régir l'évolution individuelle des objets contenus dans le monde virtuel selon des lois comportementales généralement calquées sur la physique du monde réel. Le schéma de fonctionnement est illustré au travers de la figure 1.5 dans le cas d'un seul objet.

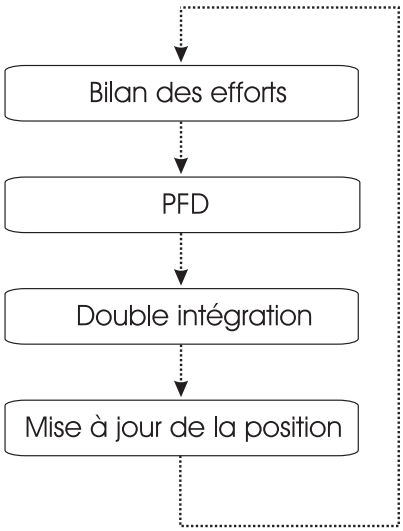


FIG. 1.5 – Etapes permettant la gestion du mouvement d'un objet au sein d'un simulateur dynamique

Dans ce schéma, la première étape consiste à faire un bilan des efforts appliqués sur l'objet virtuel. Ces efforts auxquels est soumis l'objet peuvent être de plusieurs origines : gravité, contraintes articulaires (également appelées liaisons bilatérales), efforts de contact liées aux collisions inter-objets, manipulation de l'opérateur. Leur accumulation va permettre d'obtenir un torseur statique d'effort. Durant la deuxième étape vont être déterminées les accélérations linéaire et angulaire de l'objet par application du principe fondamental de la dynamique. Les nouvelles position et orientation sont alors directement

déterminées par double intégration durant la troisième étape. Il parait intéressant d'observer que le fait d'évoluer en temps discrets engendre un échantillonnage du mouvement des corps (i.e. le mouvement d'un corps est assimilable à une succession de sauts). L'intégration peut se faire selon plusieurs approches mathématiques, implicites ou explicites (Euler, Tustin, Runge-Kutta, ...). La dernière étape consiste simplement à mettre à jour les position et orientation de l'objet considéré au sein de la scène.

Bien entendu, ce schéma présente un cas très simplifié de la simulation de la dynamique au sein d'un environnement numérique. En effet, la résolution du système visant à assurer le respect de l'ensemble des contraintes inhérentes à la simulation à l'instant considéré s'avère souvent complexe et nécessite la mise en place de techniques adaptées permettant la résolution de tels systèmes. Cette problématique, qui engendre à elle seule bon nombre de travaux au sein de la communauté scientifique, n'est pas abordée dans ce mémoire où seules les problématiques inhérentes à la détection des collisions sont présentées.

Gestion des collisions

Ce deuxième module est décomposable en deux tâches : la détection des collisions et la génération des informations de contact. La détection des collisions peut être effectuée selon plusieurs techniques en fonction du type de géométrie utilisé et des performances escomptées. Dans le cadre des méthodes discrètes, qui constituent l'immense majorité des techniques utilisées, cette première phase consiste à détecter soit des interpénétrations au sein des différentes paires d'objets, soit une grande proximité entre ces derniers (on parle de champs de distances). *A contrario*, les méthodes dites continues détectent l'instant exact du premier contact entre les objets. Nous exposerons par la suite un état de l'art présentant l'ensemble des méthodes les plus utilisées. Quelle que soit l'approche choisie, cette première phase s'avère la plus lourde des tâches à effectuer par le gestionnaire des collisions. En effet, la quantité de calculs à effectuer dans le cadre de cette détection est directement liée à la nature des objets évoluant au sein du monde virtuel, particulièrement en termes de complexité géométrique.

Après cette première phase de détection des collisions, il reste à constituer l'ensemble des informations nécessaires au simulateur dynamique quant à la détermination des efforts résultant des collisions entre les paires de solides concernées. Pour ce faire, les contacts peuvent être formalisés suivant plusieurs approches dont les plus communément utilisées sont présentées dans la suite de ce mémoire.

1.4 Conclusion et discussion

Au travers de ce premier chapitre, nous avons proposé une présentation de plusieurs domaines d'application de la réalité virtuelle en tant qu'outil de simulation interactif. Dans un deuxième temps, nous avons également introduit les principes fondamentaux qui régissent le fonctionnement d'un moteur physique. Aussi, avant de nous lancer dans la présentation de l'état de l'art relatif à la détection des collisions, nous proposons dans la suite une discussion visant à mettre en évidence les problématiques inhérentes au fonctionnement de ce processus dans un contexte temps réel.

Problématiques liées à la contrainte temps réel

Le bon fonctionnement d'une simulation interactive impose une fréquence de fonctionnement élevée quant au rafraîchissement du moteur physique, et cela particulièrement lorsque la manipulation est effectuée via un système haptique. Il est en effet communément admis que, dans ce contexte, une fréquence d'au moins 1 KHz est nécessaire afin de garantir la stabilité et la qualité du rendu haptique (en pratique, une fréquence de l'ordre de 500 Hz s'avère généralement suffisante). Hors, les moteurs physiques actuels, tant au niveau des produits commerciaux que des travaux exploratoires proposés par certains laboratoires, ne permettent pas de satisfaire à ces contraintes de performances dès lors que la scène simulée contient des géométries nombreuses et/ou complexes. Cette incapacité des moteurs physiques est principalement liée à la gestion des collisions. Ces limitations sont d'autant plus importantes dans le cas ou la scène contient des objets déformables, les techniques d'accélérations proposées à ce jour s'avérant souvent inefficaces dans ce cadre.

Aussi, la mise en place d'une simulation interactive nécessite souvent une ***simplification de la scène***, soit par une diminution du nombre d'objets simulés, soit via une simplification géométrique de ces derniers. Bien que bon nombre d'outils mathématiques permettent de réduire la complexité des géométries simulées (décimation de maillages, ...), ce genre d'approximations n'est pas toujours acceptable dans le cadre d'applications nécessitant un niveau de réalisme important, contrainte susceptible de compromettre l'utilisation d'outils de simulation interactifs dans certains contextes.

Chapitre 2

Détection des collisions : Etat de l’art

2.1 Introduction

Le coût de la détection de collisions, que ce soit en termes de temps de calcul ou d'utilisation des ressources mémoire, est généralement bien plus important que celui attribuable à la gestion du mouvement des objets. Aussi, bon nombre d'équipes de recherche à travers le monde travaillent à imaginer de nouvelles solutions algorithmiques visant à réduire le coût de cette tâche [MKF03]. Les travaux réalisés au sein de cette communauté s'articulent autour de plusieurs problématiques inhérentes à certaines des caractéristiques des scènes simulées, tant au niveau de la représentation géométrique des objets que du niveau de performance escompté.

Dans le cadre de cette thèse, nous nous concentrons particulièrement sur certaines de ces caractéristiques :

- Complexité géométrique de la scène : nombre d'objets contenus dans celle-ci.
- Complexité géométrique des objets : nombre de primitives élémentaires qui les composent.
- Nature des géométries simulées : nature des primitives élémentaires qui constituent la définition géométrique des objets simulés (triangles, points, ...).
- Convexité des géométries.
- Rigidité des objets : leur aptitude à se déformer.

Ce chapitre constitue un état de l'art au travers duquel nous abordons les principales problématiques liées à la détection des collisions. Nous y proposons une présentation non-exhaustive des concepts et techniques les plus communément utilisés dans le cadre de simulations répondant aux caractéristiques précédemment introduites. Pour ce faire, nous nous intéressons dans un premier temps aux approches visant à répondre aux problématiques inhérentes à la gestion de la complexité de la scène et des objets simulés. Pour cela, nous nous concentrons particulièrement sur les principales approches visant à accélérer le processus de détection des collisions. Dans un deuxième temps, nous nous intéressons aux approches proposées dans la littérature quant à l'adaptation des techniques d'accélération au cas de la simulation d'objets déformables. Nous présentons également, dans ce même contexte, des techniques de détection des collisions fondées sur des approches probabilistes (on parlera d'approches stochastiques). Puis, dans une troisième partie, nous discutons de l'influence de la nature des objets simulés et de leurs propriétés de convexité sur les approches mises en œuvre pour la détection des collisions. Nous nous intéressons entre autre dans cette partie à la manière de représenter les contacts inter-objets dans le cadre d'approches discrètes. Enfin, nous introduisons la notion de

cohérence spatio-temporelle, l'algorithme proposé dans cette thèse exploitant cette caractéristique des simulations pseudo-continues.

2.2 Gestion de la complexité géométrique

Dans cette section, nous nous intéressons aux techniques visant à réduire la complexité du processus de détection des collisions dans le cadre de simulation d'environnements denses. Avant de passer à la description proprement dite des principaux algorithmes et concepts visant à accélérer la détection des collisions, nous présentons ici le schéma de fonctionnement générique d'un cycle de détection des collisions afin de pouvoir, par la suite, clairement classifier les solutions présentées. En effet, la tâche de détection des collisions, qui vise rappelons-le à fournir au solveur dynamique les informations géométriques matérialisant les contacts entre objets, est généralement décomposée en trois sous-tâches (figure 2.1).

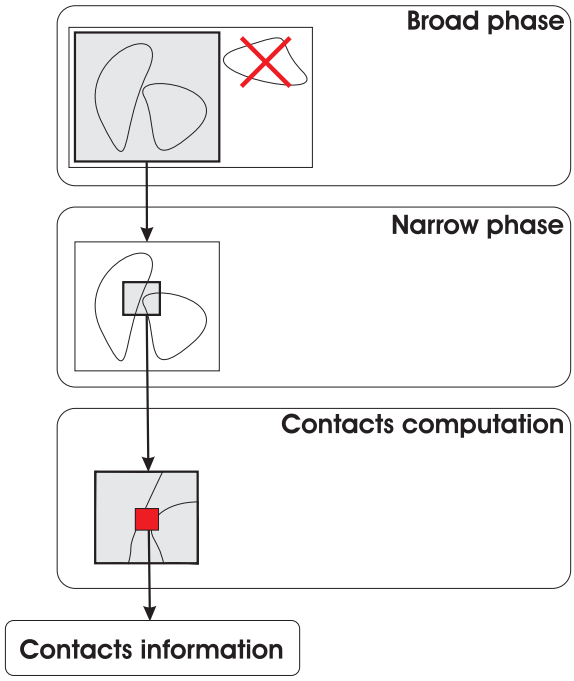


FIG. 2.1 – Schéma standard d'un cycle de détection des collisions

La figure 2.1 illustre le pipeline en question. Les deux premières tâches de ce cycle sont assimilables à des filtres plus ou moins grossiers destinés à réduire l'espace géométrique considéré lors de la phase de détermination des informations de contact. Elles vont ainsi permettre une diminution importante de la combinatoire à considérer en identifiant, dans un premier temps, les paires de modèles susceptibles d'entrer en collisions (*broad phase*), puis, dans un deuxième temps, les zones géométriques (*narrow phase*) à prendre en compte pour la détermination des informations de contacts.

2.2.1 Subdivision spatiale

Cette technique consiste à subdiviser l'espace géométrique en un ensemble de régions afin d'être en mesure d'identifier les paires d'objets susceptibles d'entrer en collision. En pratique, les algorithmes basés sur ces approches vont dresser, pour chaque région, la liste des éléments géométriques présentant une intersection non-nulle avec le volume défini par celle-ci. Ainsi, seules les paires constituées d'éléments présents dans une même liste (*i.e.* dans une même zone de l'espace) seront prises en considération lors de la détection de collision. La subdivision de l'espace peut être réalisée de différentes manières. Dans la suite, nous présentons deux types de représentations.

Décomposition spatiale régulière

Cette première approche consiste à réaliser la décomposition spatiale suivant une grille régulière allignée sur les axes principaux (figure 2.2) [Tur89]. Les zones obtenues, assimilables à des cubes, sont communément appelées *voxels*.

Bien que dans le cadre de cette première approche la décomposition spatiale soit triviale, plusieurs difficultés sont observables quant à la mise en œuvre de l'algorithme. La première, d'ordre purement algorithmique, est de mettre à jour les listes d'appartenance des objets aux différentes zones de l'espace de façon suffisamment performante pour remplir les contraintes de temps inhérentes à une simulation temps réel. Une deuxième difficulté réside dans le dimensionnement des voxels. En effet, une décomposition voxélique trop fine risquerait d'engendrer un nombre de tests d'appartenance trop important et, *a contrario*, une décomposition trop grossière ne s'avérera pas suffisamment sélective pour réduire sensiblement le nombre de paires d'objets à considérer. Aussi, le choix des paramètres de discrétisation sera généralement conditionné par différents critères tels que le nombre d'objets contenus dans la scène, leur taille, leur répartition et leur cinématique.

Approches hiérarchiques

Afin d'identifier plus efficacement les zones d'intérêt de l'espace, plusieurs approches basées sur une hiérarchisation de la décomposition spatiale ont été proposées. Ces dernières consistent en une décomposition récursive de l'espace prenant en considération l'occupation des régions considérées à chaque niveau hiérarchique (*i.e.* une région est subdivisée si et seulement si elle contient des éléments géométriques). Cette technique permet une décomposition plus *intelligente* de l'espace et a souvent pour effet de réduire sensiblement la complexité de l'algorithme.

Plusieurs formalismes existent quant à la forme des régions obtenues après décomposition (*k-d trees* [BF79], ...). Le plus utilisé est très probablement l'*octree* [BT95]. Avec ce type de formalisme, les régions obtenues sont de forme cubique et leur décomposition hiérarchique consiste en un découpage en huit sous-boîtes identiques, découpage réalisé selon les axes principaux de la scène (figure 2.2).

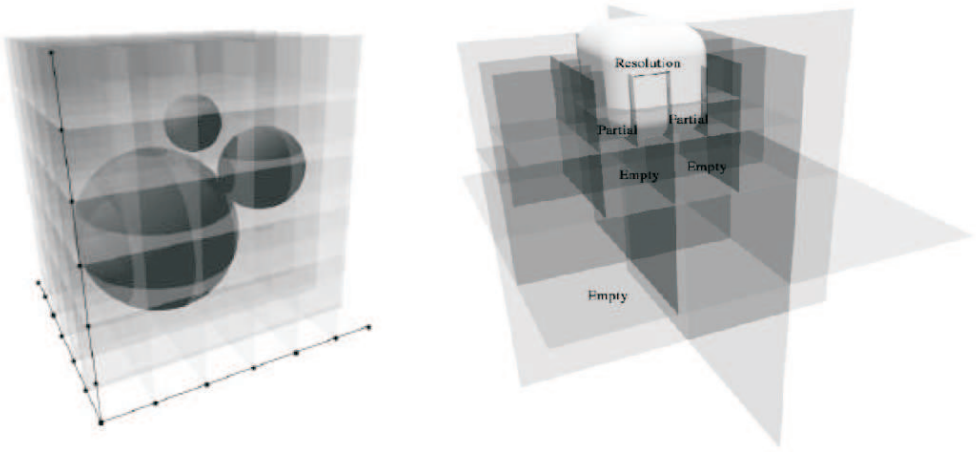


FIG. 2.2 – Exemples de subdivisions de l'espace : à gauche via une grille régulière et à droite via une décomposition hiérarchique de type *octree*

2.2.2 Critères topologiques et cinématiques

Critères topologiques :

Les algorithmes basés sur ce type de critères consistent en l'établissement d'une carte de proximité contenant l'ensemble des distances minimum *globales* séparant les objets. Aussi, les tests inhérents à la détection des collisions seront uniquement effectués au sein des paires d'objets présentant une proximité importante. Afin d'établir efficacement une pseudo-carte de proximité, il est envisageable d'appliquer des algorithmes basés sur *les domaines de Voronoï* (figure 2.3). Les éléments géométriques contenus dans le domaine de Voronoï d'un objet sont plus proches de celui-ci que de tout autre objet au sein de la scène.

L'une des limites de ce type d'approches réside dans la mise à jour de la carte de proximité à chaque cycle de calcul (figure 2.4). Les techniques permettant la mise à jour des domaines de Voronoï sont généralement de complexité $O(n \log n)$ avec n égal au nombre d'objets contenus dans la scène. Plusieurs approches visant à simplifier le processus peuvent cependant être appliquées afin de réduire les temps de mise à jour des cartes de proximité. Elle vont le plus souvent consister en une approximation géométrique des

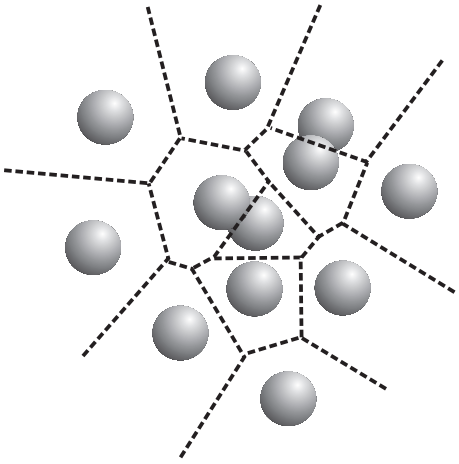


FIG. 2.3 – Exemple d'application des *domaines de Voronoï*. Chaque domaine définit une zone de proximité autour d'un objet.

objets visant à ne considérer non plus la géométrie exacte des modèles mais des volumes englobant ces derniers, tels que des sphères ou encore des enveloppes convexes.

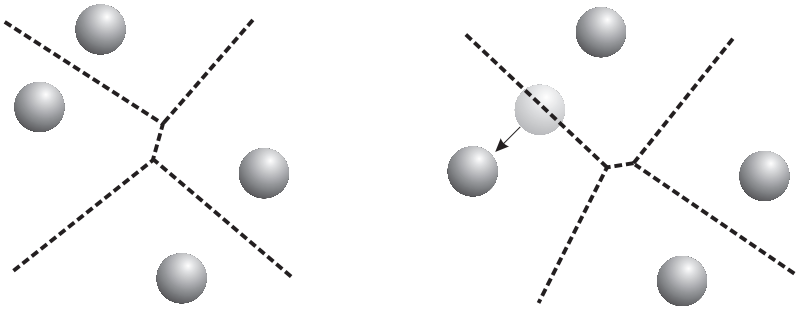


FIG. 2.4 – Exemple de mise à jour des domaines de Voronoï suite au déplacement de l'un des objets de la scène.

D'autres approches, également dites topologiques, consistent en un classement des objets au sein d'une hiérarchie de volumes englobants généralement formalisés sous forme de boîtes isothétiques (*i.e.* alignées selon les axes principaux).

Critères cinématiques :

Les approches basées sur ce type de critères exploitent les vitesses relatives des objets. Elles sont basées sur le simple postulat selon lequel deux objets qui s'éloignent ne peuvent entrer en collision [Van94]. Ces techniques sont souvent citées sous l'appellation *test du recul* [RKC02].

2.2.3 L'approche projective

Cette approche est basée sur le postulat suivant : *Si deux objets sont en intersection, alors toute projection de ces derniers présente également une intersection.* Aussi, son implémentation va généralement consister à réaliser des tests d'intersection entre les intervalles occupés par les projections de chacun des objets selon les axes principaux. Ainsi, pour une paire d'objets, l'opération comprend trois tests d'intersection portant sur les intervalles occupés par les projections. La principale difficulté rencontrée dans le cadre de cette approche va être de déterminer dans un temps restreint les intervalles occupés par les projections. Afin de réduire le temps nécessaire à la phase de projections, il est envisageable d'utiliser des approximations des modèles telles que des boîtes englobantes orientées.

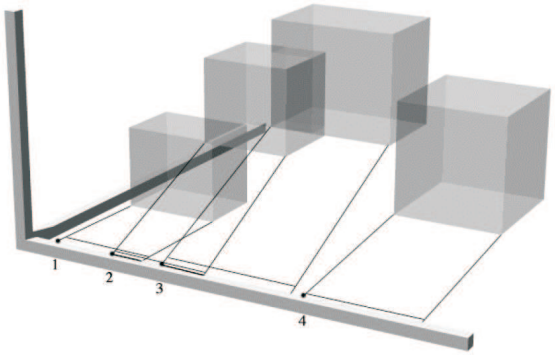


FIG. 2.5 – Illustration de l'approche projective : Projection des objets selon l'un des axes principaux

2.2.4 Hiérarchies de volumes englobants

Le principe de cette approche, clairement assimilable à une technique de détection approximative est de partitionner les espaces occupés par les objets composant la scène. Ainsi, chaque objet va être représenté par un volume englobant lui-même décomposé en une arborescence de sous-volumes, généralement de topologie similaire au volume englobant *racine*. La stratégie consiste à effectuer une localisation spatiale des zones susceptibles de s'interpénétrer. En pratique, on teste la présence de chevauchement entre les volumes englobants racines d'objets distincts, puis, dans l'hypothèse d'un chevauchement, on effectue un test récursif similaire au niveau des sous-volumes qu'ils contiennent. Cette stratégie permet d'éviter une multitude de tests non-pertinents et se voit très utilisée au sein des moteurs physiques existants. Cependant, ces tests de chevauchement peuvent s'avérer très nombreux dans certains cas (contacts conformant par exemple) et

ainsi compromettre les performances de la simulation. De plus, une mise à jour régulière de ces volumes peut s'avérer nécessaire dans le cadre de simulations faisant intervenir des éléments déformables. Des approches permettant la mise à jour des hiérarchies sont présentées dans la section 2.3.

Types de volumes englobants

Plusieurs types de géométries peuvent être utilisés dans le cadre de cette décomposition de l'objet. Les plus communément utilisés sont les sphères [RKS01], les boîtes isothétiques (AABB : Axis Aligned Bounding Boxes) [LAM01] et les boîtes englobantes orientées (OBB : Oriented Bounding Boxes) [GLM96]. D'autres types de volumes englobant plus sophistiqués ont été proposés tels que les polytopes à orientation discrète (k-dops) [KHM+98], les *tribox* [CR99] ou encore les enveloppes convexes.

Chacun de ces types de volumes englobants se voit plus ou moins adapté en fonction de la géométrie considérée et du niveau de performance escompté. Ainsi, les sphères englobantes ont pour principal avantage la simplicité des tests de chevauchement . En effet, un test de chevauchement entre deux sphères se limite à une comparaison entre la distance séparant leur centre et la somme de leurs rayons respectifs. Cependant, les sphères donnent généralement une approximation moins optimale de la géométrie considérée, en comparaison avec les OBBs par exemple qui, malgré des tests de chevauchement beaucoup plus lourds (théorème des 15 axes séparateurs [GLM96]), limitent l'exécution de tests non-pertinents de par un volume englobé minimal. Avec la figure 2.6, nous proposons une illustration de ce phénomène.

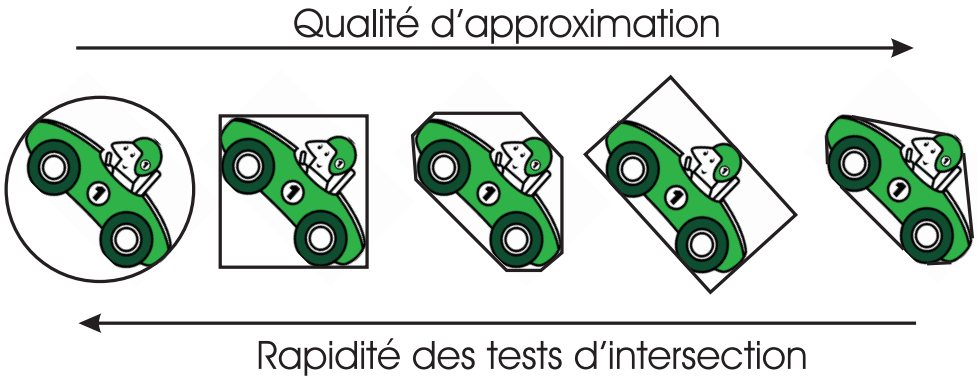


FIG. 2.6 – Comparaison entre différents types de volumes englobants. De gauche à droite : sphère, AABB, k-dop, OBB et domaine convexe

Construction de la hiérarchie

Une hiérarchie de volumes englobants peut se construire selon deux approches :

– Top-down
Cette approche consiste en une construction de la hiérarchie du haut vers le bas. Dans un premier temps, un volume racine englobant l'ensemble du modèle est calculé. Celui-ci est par la suite décomposé en n sous-volumes ($n \geq 2$). La décomposition se poursuit récursivement sur les sous-volumes obtenus jusqu'à obtenir les volumes *feuilles* contenant chacun une primitive élémentaire du modèle (par exemple un triangle dans le cas polyédrique).

– Bottom-up
Cette approche consiste en une construction de la hiérarchie du bas vers le haut. Dans un premier temps, l'ensemble des volumes feuilles contenant les primitives élémentaires du modèle va être calculé. Ces volumes vont par la suite être regroupés selon des critères de proximité afin d'obtenir des volumes englobants de niveau supérieur. Le *regroupement* se poursuit récursivement sur l'ensemble des volumes nouvellement calculés jusqu'à obtenir le volume englobant racine contenant l'ensemble du modèle.

2.2.5 Multirésolution

Dans [OL03], les auteurs proposent de mettre en œuvre une hiérarchisation duale des composantes géométriques des objets. Chaque objet est décomposé en une hiérarchie de volumes englobants dont chaque nœud se voit correspondre plusieurs représentations à résolution variable de la zone géométrique qu'il contient (on parle de *CLODS : Collision Levels of Details*). Aussi, les auteurs proposent d'effectuer la détection de collision de façon plus ou moins précise en fonction de certains critères (planéité, ...) visant à garantir un comportement des objets réalistes, particulièrement dans le cadre de simulations mettant en œuvre un système de manipulation haptique.

FIG. 2.7 – Exemple de représentation multirésolution

2.2.6 Accélération matérielle

En plus des solutions purement algorithmiques présentées précédemment, certaines équipes de recherche ont proposé des approches visant à exploiter des calculateurs hardware, et particulièrement les unités de calculs dédiées à la restitution graphique (on parle de *GPU*).

Les approches basées sur ce type de matériel imposent l'utilisation de géométries de nature polyédrique. Elles sont classifiables en deux catégories :

– *Approches basées sur les triangles*
Ces approches se basent sur la récupération des triangles appartenant au cône de visée (*view volume*) [LCN99]. Elles sont limitées par la géométrie du *view volume* qui, au sein des cartes graphiques actuelles, ne peut être que de deux types : boîte parallélépipédique ou pyramide tronquée à base rectangle.

– *Approches basées sur les pixels*
Ces approches, dites basées sur l'image, exploitent le *pixels shader* [KOLD02]. En d'autres termes, elles ne se basent pas sur les triangles proprement dits, mais sur leur représentation pixélique. Il est à noter que la précision de la détection est ici liée à la taille des pixels. Aussi, un compromis entre niveau de résolution et temps de calcul est à considérer. Des techniques plus récentes également basées sur ce type d'approches utilisent les nouvelles fonctionnalités proposées dans les cartes graphiques les plus évoluées telles que la détection d'occlusions [GRLM03]. Les performances se voient ainsi grandement améliorées.

2.3 Détection des collisions entre corps déformables

Les techniques d'accélération du processus de détection des collisions dans le cadre de simulations mettant en scène des éléments déformables sont, pour la plupart, fondées sur les mêmes concepts que ceux mis en œuvre dans le cas rigide. Aussi, dans la littérature scientifique, on trouve plusieurs adaptations au cas déformable d'une partie des approches et algorithmes précédemment introduits. Dans la suite, nous traitons de l'application des hiérarchies de volumes englobants et de la subdivision spatiale dans ce contexte. Nous introduisons également en fin de section des approches probabilistes dites *stochastiques* proposées dans ce même contexte de simulation d'objets déformables.

2.3.1 Hiérarchies de volumes englobants

Comme expliqué précédemment, les approches utilisant des hiérarchies de volumes englobants consistent en une décomposition hiérarchique des objets de la scène. En se basant sur les décompositions ainsi obtenues, plusieurs tests d'intersections donnant lieu à des descentes récursives des hiérarchies permettent d'isoler les paires de zones au sein desquelles des collisions sont susceptibles d'avoir lieu. Ces techniques permettent ainsi de réduire de manière importante la combinatoire considérée lors de la phase d'identification des contacts, améliorant sensiblement les performances du processus.

Dans le cas rigide, le positionnement relatif des primitives qui composent les objets au sein de leur repère de construction n'évoluent pas durant la simulation. Dans ce contexte, il n'est pas nécessaire de mettre à jour les hiérarchies de volumes englobants. Ceci n'est pas vrai dans le cas déformable. En effet, une déformation est assimilable au déplacement de tout ou partie des composantes d'un objet dans son repère de construction. Aussi, dans le cadre de simulation d'objets déformables, l'utilisation d'approches basées sur des hiérarchies de volumes englobants nécessite une mise à jour régulière de ces dernières, en fonction de l'évolution de la forme des objets simulés. Il est à noter que le temps nécessaire à ces mises à jour vient s'additionner à la durée du processus global de détection des collisions et qu'il doit, par conséquent, se voir minimisé.

Mise à jour des hiérarchies

Deux techniques peuvent être envisagées pour la mise à jour des hiérarchies : la reconstruction, et le ré-ajustement. La première consiste en une reconstruction totale de la hiérarchie à chaque pas de simulation. Cette méthode s'avère souvent incompatible avec les contraintes de temps inhérentes à des simulations interactives, et ne peut se voir mise en œuvre que dans des cas où la scène simulée est constituée de corps peu complexes (i.e. composés de peu de primitives géométriques élémentaires). Le ré-ajustement d'une hiérarchie consiste, comme son nom l'indique, en une modification de la configuration actuelle de la hiérarchie visant à permettre un englobement optimal des primitives nouvellement positionnées. Cette deuxième méthode est beaucoup plus rapide qu'une reconstruction totale. Dans [VDB97], l'auteur s'intéresse à la détection de collisions entre objets déformables complexes. Pour ce faire, il met en œuvre un algorithme basé sur des hiérarchies de volumes englobants de type AABB (boîtes isothétiques). Dans ce contexte, il observe qu'un ré-ajustement des hiérarchies s'avère environ dix fois moins coûteux en termes de temps qu'une reconstruction totale de celles-ci. Il semble cependant intéressant de remarquer qu'une reconstruction totale des hiérarchies, éventuellement non-

systématique, peut parfois s'avérer pertinente, notamment dans le cadre de situations présentant de grandes déformations des objets simulés.

Le réajustement peut être réalisé de différentes manières : Soit en partant du nœud racine et en réalisant un réajustement au travers d'une descente récursive, soit à partir des nœuds feuilles, le réajustement étant alors effectué du bas vers le haut de la hiérarchie. On retrouve là une problématique très similaire à celle abordée précédemment dans le cadre de la construction des hiérarchies (méthodes top-down et bottom-up. Cf. paragraphe 2.2.4).

Dans [LAM01], les auteurs comparent les deux méthodes. Ainsi, il ressort de leurs expériences que dans le cas où de nombreux nœuds profonds dans la hiérarchie sont atteints, la méthode bottom-up donne de meilleurs résultats. *A contrario*, si peu de nœuds profonds sont atteints, l'approche top-down est plus intéressante. Aussi, au vu de ces observations, les auteurs proposent une méthode hybride consistant dans un premier temps en un réajustement bottom-up du haut de la hiérarchie (il s'agit en fait de la moitié supérieure), puis, dans un deuxième temps, en un réajustement top-down à partir des nœuds présentant de nouvelles déformations. Cette méthode permet de réduire sensiblement le nombre de traitements, bon nombre des nœuds non-pertinents étant exclus du traitement à l'issue de la première phase. Cependant, elle entraîne l'allocation d'un espace mémoire supplémentaire, éventuellement important, destiné au stockage des informations inhérentes aux primitives élémentaires au niveau des nœuds internes à la hiérarchie.

Une autre approche visant à minimiser le coût du maintient des hiérarchies dans des configuration pertinentes consiste à réduire le nombre de procédures de réajustement. Dans [MKE03], les auteurs proposent de réaliser un gonflement des volumes englobants. Cette technique permet de n'effectuer les réajustements que lorsque les primitives élémentaires ont parcouru une distance supérieure à la distance de gonflement (on parle là de déplacements des primitives dans le repère de construction de l'objet).

Dans certains contextes (animation, ...), il est envisageable que certaines des formes successivement prises par une partie des objets au cours de la simulation soient connues à l'avance ; On parle généralement de *morphing*. Il est alors possible, certaines configurations cibles étant connues, de reconstruire de façon performante les configurations intermédiaires des hiérarchies, en procédant par interpolation. Dans [LAM03], les auteurs proposent une telle technique dans le cas particulier de volumes englobants de types DOP. Ils observent un gain important de performances en comparaison avec une procédure ne prenant pas en considération les configurations géométriques connues. Plus généralement, il est parfois possible d'exploiter les champs de déplacements inhérents au

mouvement des primitives lors de déformations pour le réajustement des volumes englobants qui composent les hiérarchies [JP04].

Nature des hiérarchies

Contrairement au cas rigide où les volumes de type OBB sont généralement privilégiés, de par une approximation plus fine des composantes des objets, ce sont les AABBs et les k-DOPs qui se voient le plus utilisées dans le cas déformable. Ce choix est motivé par le fait que la mise à jour de volumes de cette nature est moins lourde que pour les OBBs.

Au delà de la nature des volumes qui les composent, les hiérarchies mises en œuvre dans le cas déformable ont une arité différente de celles utilisées dans le cas rigide. En effet, dans un contexte déformable, des arbres d'arité 4 ou 8 s'avèrent mieux adaptés que des arbres binaires [MKE03, LAM01].

2.3.2 Subdivision spatiale

Comme il l'a été expliqué précédemment (Cf. paragraphe 2.3.2), la subdivision spatiale consiste à partitionner l'espace géométrique en un ensemble de régions afin d'être en mesure d'identifier les paires d'éléments susceptibles d'entrer en collision. Cette technique peut se voir utilisée dans le cas déformable, la mise à jour des listes d'objets contenus dans chaque région de l'espace prenant implicitement en compte la forme des objets.

L'espace peut être subdivisé de différentes manières. Dans le cas rigide, les découpages de types *octrees* et *kd-trees* sont généralement mis en œuvre. Ces méthodes, bien qu'elles permettent souvent une meilleure localisation des objets avec une sollicitation mémoire réduite, dépendent de la forme des objets, contrairement à un découpage de type voxélique (i.e. selon une grille 3D alignée). Aussi, dans le cadre de simulation faisant intervenir des éléments déformables, on privilégiera un tel partitionnement. Dans [THM+03], les auteurs se base sur une telle approche. Ils proposent également d'utiliser une structure de type *table de hachage* afin d'optimiser, non-seulement le taux d'occupation mémoire, mais également les phases de mises à jour des listes d'occupation des différentes régions de l'espace.

2.3.3 Approches stochastiques

L'exploration de l'ensemble de la combinatoire inhérente aux algorithmes de détection de collisions *déterministes* rend souvent impossible la mise en œuvre d'une simulation

interactive répondant aux contraintes de temps imposées par l'utilisation d'un système à retour d'effort. Aussi, depuis peu, plusieurs algorithmes basés sur des *approches probabilistes* (dites *stochastiques*) ont été proposés. Ces derniers se sont principalement vus proposés dans le cadre particulier des simulations mettant en œuvre des objets déformables, les algorithmes d'accélération présentés précédemment (Cf. paragraphe 2.2) s'avérant souvent inefficaces dans ce contexte (la mise à jour des informations pré-calculées est souvent très lourde) [TKZ+04].

Comme dit précédemment la mise en œuvre de techniques stochastiques pour la détection des collisions est très récente. Aussi, peu d'algorithmes de ce type ont été proposés dans la littérature. Dans la suite, nous introduisons deux techniques fondées sur des approches probabilistes. Nous introduisons également en fin de section les travaux de Carretero et Nahon qui proposent dans leur travaux une approche basée sur des techniques évolutionnistes pour le calcul de la distance minimum globale (GMD) séparant deux objets.

Évaluation de la probabilité d'intersection

Dans [JG03], les auteurs proposent une approche consistant en l'évaluation de la probabilité d'intersection entre les éléments géométriques constituant les objets. Pour ce faire, l'algorithme se base sur les nœuds de hiérarchies de volumes englobants (de type AABB dans l'implémentation présentée). La technique proposée dans ce contexte consiste à ne réaliser les tests d'intersections inhérents aux approches basées sur hiérarchies de volumes englobants en ne prenant en considération que les paires de noeuds présentant une probabilité d'intersection suffisamment élevée. Notons que cette approche ne permet pas de garantir la détection de l'ensemble des collisions et que l'établissement d'un compromis entre la qualité du résultat obtenu (soit l'aptitude de l'algorithme à détecter l'ensemble des collisions) et le temps d'exécution s'avère nécessaire. La régulation de ce compromis se fait via deux paramètres assimilables à des seuils de probabilité au delà desquels la collision entre des éléments géométriques est considérée comme probable. Grâce à cette technique, les auteurs observent, dans le cadre d'une implémentation utilisant des hiérarchies de boîtes isothétiques, des gains de temps sensibles (gain d'un facteur 3 à 6) pour un taux d'erreur relativement réduit (de l'ordre de 4%).

Descentes de gradients à partir de positions aléatoires

Un autre type d'approches dites stochastiques consiste à réaliser des descentes de gradients locales à partir de positions choisies aléatoirement à la surface des objets. Les entités alors manipulées sont assimilables à des segments ayant pour extrémités des points

situés à la surface des objets (on parle de *paires*). Aussi, la sortie de tels algorithmes est assimilable à des champs de distances minimum locales, tel qu'introduit dans 2.4.1.

Plusieurs algorithmes proposés dans la littérature se basent sur une telle approche. Dans [GD04], les auteurs proposent un algorithme visant à détecter les collisions entre des objets, rigides ou déformables, sous des contraintes temps réel au sens de la fréquence de rafraîchissement graphique (On parle de simulations cadencées à environ 30 Hz). Pour ce faire, une approche multi-résolution visant à utiliser des représentations plus ou moins fines des régions géométriques composant les objets en fonction de leurs niveaux de proximité est mise en œuvre. Nous n'abordons pas ici l'aspect multi-résolution de cette algorithme mais seulement le procédé mis en œuvre pour l'identification des paires de régions présentant une grande proximité. Pour ce faire, les auteurs manipulent une liste de *paires* correspondant à des couples de points positionnés respectivement à la surface des objets dont on cherche à évaluer la proximité. Les *paires* sont générées selon la méthode dite de Monte-Carlo. Celle-ci se décompose en trois phases : 1. Un point, dit *graine* et noté g, est choisi *aléatoirement* à la surface de l'un des objets, 2. Le point le plus proche de g appartenant au deuxième objet est calculé, il est noté P_{g_1}, 3. Le point le plus proche de P_{g_1} à la surface du premier objet, noté P_{g_2}, est calculé, 3. La paire obtenue à l'issue de la procédure est définie par le couple $\{P_{g_1}, P_{g_2}\}$. La figure 2.8 illustre ce mode opératoire.

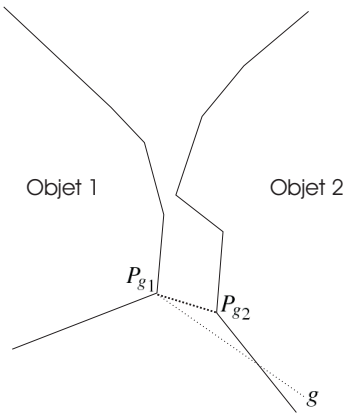


FIG. 2.8 – Méthode de Monte-Carlo pour la génération d'une paire. Une graine g est d'abord choisie aléatoirement. A partir de celle-ci, les points P_{g_1} et P_{g_2} sont calculés selon des critères de proximité.

Ce procédé, mis en œuvre pour la création de chaque paire, a pour complexité $n+n'$, n et n' correspondant aux nombres d'entités élémentaires composant respectivement les deux objets dont on cherche à évaluer la proximité. Aussi, afin de réduire la complexité du processus global d'identification des couples de régions proches, le nombre de paires

créées au cours d'un même cycle est limité. Les auteurs proposent par ailleurs de réguler ce nombre de créations de paires en fonction de la configuration courante de manière à garantir la satisfaction de la contrainte temps réel (au sens graphique rappelons-le). Il est à noter que le fait de limiter le nombre de paires entraîne la non-considération de certaines zones de l'espace combinatoire. Aussi, afin d'assurer la prise en considération des zones les plus susceptibles d'entrer en collision, le positionnement des graines peut être réalisé en se basant sur plusieurs critères tels que la direction du mouvement relatif des objets considérés.

Afin d'exploiter la cohérence spatio-temporelle (Cf. paragraphe 2.5), les paires s'avérant les plus pertinentes (i.e. les plus petites) sont mises à jour à chaque cycle au travers de descentes de gradient locales. Ce processus utilise des informations de connexité pré-calculées (i.e. voisinage des faces). Les paires considérées comme pertinentes sont stockées dans une liste mise à jour à chaque cycle. Pour être ajoutée à la liste, une paire doit avoir une taille d répondant à la contrainte $d < k \times d_{min} + \varepsilon$, où d_{min} correspond à la taille de la plus petite des paires présentes dans la liste, k est un coefficient de réglage (de l'ordre de 1.2) permettant de dimensionner la liste, et ε est un coefficient assimilable à une peau d'orange permettant la prise en considération de plusieurs paires lorsque d_{min} s'avère très petite (condition nécessaire à la gestion d'objets concaves).

Dans [RGF+04], les auteurs proposent une adaptation de l'approche précédemment présentée au cas de la simulation du comportement d'intestins dans un contexte chirurgical. Dans ces travaux, les intestins sont définis comme des colliers de cylindres. Aussi, le processus de détection des collisions consiste ici en une comparaison entre la distance séparant les axes des cylindres considérés et la somme de leurs rayons. Il est à noter que les auteurs proposent d'utiliser directement les paires pour le calcul de la réponse aux contacts. Dans [KNF04], les auteurs proposent d'utiliser une approche mixte mettant en œuvre une approche stochastique basée sur des paires telles que celles présentées précédemment, et une approche déterministe basée sur des hiérarchies de volumes englobants. Aussi, les positions initiales des paires sont ici conditionnées par l'identification préalables de zones de proximité. Avec cette méthode, les auteurs observent une réduction sensible du temps de convergence de l'algorithme par rapport à l'approche stochastique initiale (i.e. n'utilisant pas les volumes englobants).

Discussion sur la qualité des résultats obtenus

Il est à noter que dans le cadre de ces approches stochastiques, il n'est pas garanti d'identifier l'ensemble des contacts apparaissant lors d'une collision (voir de simplement détecter la collision). En effet, le résultat de l'algorithme est dépendant d'une configura-

tion initiale conditionnée de façon aléatoire et ne garantissant pas l'exploration de l'ensemble des zones intervenant dans la collision. La figure 2.9 illustre ce phénomène dans le cadre de l'algorithme proposé dans [RGF+04]. Comme nous le verrons dans la suite de ce mémoire, l'algorithme que nous proposons est assimilable à une approche stochastique. Aussi, nous rediscuterons en détails des problématiques inhérentes à ce type d'approches, notamment des propriétés de convergence.

FIG. 2.9 – Exemple de non-détection des collisions dans le cas des approches stochastiques. (à gauche) L'ensemble des collisions est détecté et le comportement des objets est cohérent. (à droite) Deux zones en collision s'interpénètrent de par une non-convergence de l'algorithme.

Approche évolutionniste pour le calcul de GMD

Dans [CN01], les auteurs proposent un algorithme basé sur une approche évolutionniste visant à déterminer une approximation de la *distance minimum globale* (GMD) séparant deux objets rigides dans une configuration donnée. Les individus manipulés dans ce contexte sont assimilables à des bi-points 3D. Aussi, ils sont définis comme des vecteurs réels de dimension 6. L'algorithme proposé dans le cadre de ces travaux consiste à manipuler une population d'individus de cette nature, la finalité étant d'obtenir, à la fin du processus, un individu définissant la GMD entre les objets considérés. Les individus peuvent se voir manipulés de différentes manières : soit au travers de mutations (variation des positions cartésiennes des points composant les individus), soit via des croisements entre individus (des individus enfants sont obtenus par des combinaisons linéaires d'individus dits parents). Le schéma présenté sur la figure 2.10 décrit le fonctionnement de l'algorithme. Celui-ci se décompose en 6 étapes :

– 1. Une population initiale est créée. Les points composants les individus sont positionnés aléatoirement au sein de zones englobant les objets considérées (OBB).

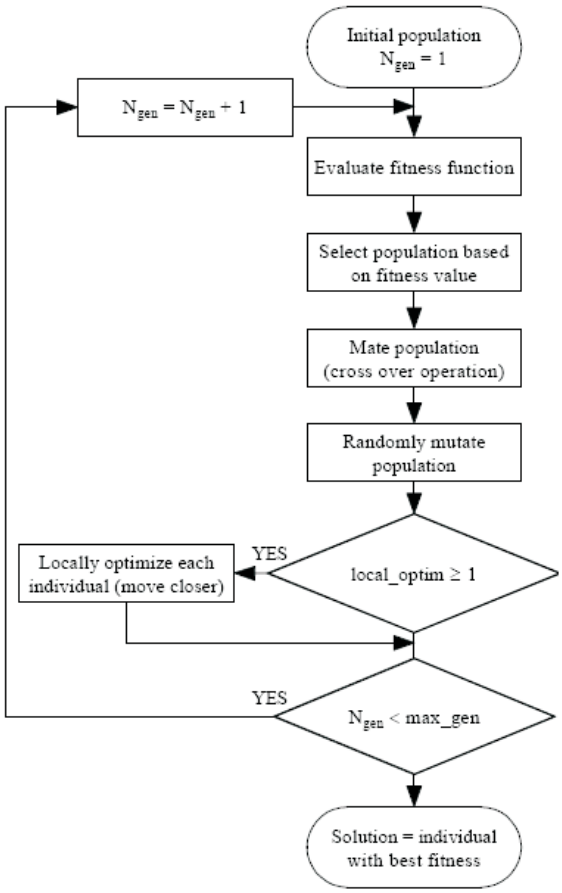


FIG. 2.10 – Schéma de fonctionnement de l'algorithme proposé dans [CN01].

– 2. La qualité (*fitness*) de chacun des individus de la population est évaluée. La *fitness* d'un individu est ici définie de la façon suivante : $f = d + k_p p$, où d correspond à la taille de l'individu, p est la somme des distances séparant les points de la surface des objets au sein desquels ils évoluent respectivement, et k_p est un coefficient permettant de pondérer l'influence de p sur la qualité de l'individu. Notons que plus la valeur de la *fitness* d'un individu est faible, plus la qualité de celui-ci est grande.
– 3. A partir de l'évaluation précédente, une sous-population d'individus ayant de bonnes caractéristiques est extraite par une méthode de sélection stochastique à la roulette (Cf. section 3.5.2).
– 4. Une partie des individus présents dans la sous-population identifiée lors de l'étape précédente sont pris pour géniteurs afin de générer de nouveaux individus dits enfants.
– 5. Une partie des individus subit des mutations (déplacement des points les composant dans le voisinage de leurs positions courantes).

– 6. On réalise une optimisation locale d'une partie des individus.
– On répète les étapes de 2 à 6 un certain nombre de fois (on parle de générations). A l'issue de ce processus, le meilleur individu est pris pour solution. Dans l'hypothèse où l'algorithme ai convergé, cet individu matérialise la GMD séparant les objets considérés.

Afin de pouvoir prendre en considération des objets non-convexes, les auteurs proposent de réaliser une décomposition des objets en éléments convexes en utilisant une méthode basée sur les CSG (Constructive Solid Geometry).

La manipulation d'individus tels qu'ils sont définis dans le cadre de ces travaux entraîne une prise en considération explicite de la contrainte d'appartenance des points aux surfaces des objets qui les portent. Cette caractéristique de l'algorithme complexifie grandement le problème d'optimisation qu'il vise à résoudre. Elle est d'ailleurs critiquée par les auteurs eux-mêmes dans le cadre d'une autre de leur publication [CN05]. Dans celle-ci, les auteurs proposent d'exprimer le génotype des individus selon une représentation paramétrique des points qui les composent. Aussi, dans ce contexte, les points se déplacent, non-plus dans l'espace cartésien délimité par les volumes englobant les objets, mais de sommet en sommet à la surface de ces derniers. Cette approche permet une simplification importante du problème à optimiser, puisque le but de l'algorithme se limite désormais à l'identification d'une paire de sommets au sein d'un nombre fini de paires potentielles. Ainsi, tout en se basant sur le même schéma que celui présenté précédemment 2.10, les auteurs proposent des opérateurs de mutations consistant à déplacer les points de sommet en sommet en se basant sur une carte de proximités inter-points établie lors d'une phase de pré-calcul. Notons cependant que, même si elle tend à réduire sensiblement la complexité de l'algorithme, cette formalisation des individus ne permet pas de garantir la détermination de la GMD, mais seulement une approximation de celle-ci.

L'algorithme ESPIONS, qui constitue la principale contribution de cette thèse, présente plusieurs similitudes avec l'approche proposée par Carretero et Nahon. Il est en effet également fondé sur une approche évolutionniste assimilable à une stratégie d'évolution (c'est à tord que Carretero et Nahon disent mettre en œuvre un algorithme génétique dans le cadre de leurs travaux, puisqu'ils ne manipulent pas des individus à composantes binaires, mais bien des vecteurs réels. Cf. paragraphe 3.3.1). Il semble cependant important de souligner que, comme le lecteur pourra s'en rendre compte dans la suite de l'exposé, notre approche est très différente de celle proposée par ces auteurs. En effet, au delà du fait qu'ESPIONS a pour vocation de déterminer des champs de LMDs et non pas seulement la GMD séparant les objets considérés, son mode de fonctionnement s'avère très différent : schéma de fonctionnement perpétuel (exploitant la cohérence spatio-temporelle),

sélection par tournois (bien plus performante que la sélection à la roulette), possibilité d'identifier les LMDs exactes (et non pas des approximations), définition des individus permettant la gestion d'objets de différentes natures géométriques, opérateurs de mutation conçus pour répondre à la contrainte temps réel, ...

2.4 Adaptation du processus de détection des collisions en fonction des propriétés géométriques des objets simulés

Dans cette section, nous nous intéressons à l'influence des propriétés géométriques des objets simulés sur les solutions algorithmiques à mettre en œuvre pour la détection des collisions. Nous entendons là par propriétés des objets : leur nature (i.e. le type de primitives élémentaires qui les composent), et leurs propriétés topologiques, notamment leur convexité. Pour ce faire, nous rappelons dans un premier temps quelques éléments inhérents à la détection de collisions dans le cadre de simulations fondées sur une formalisation discrète du mouvement des objets simulés (Détection d'intersection vide, détection de l'instant de contact, représentation des contacts). Puis, nous proposons une présentation succincte des algorithmes proposés dans le cas de simulations mettant en scène des éléments polyédriques convexes, et mettons en évidence les problèmes liés à la présence d'objets concaves au sein de la scène simulée. Enfin, nous présentons différentes approches visant à détecter les collisions dans le cadre de simulations faisant intervenir d'autres types d'objets tels que des surfaces paramétriques.

2.4.1 Détection des collisions en mode discret

Dans le cadre de la simulation de scènes réelles au sein d'environnements numériques virtuels, le mouvement d'un objet peut être formalisé de différentes manières. Au travers de cette étude, nous nous intéressons particulièrement au cas de simulations où la dimension temporelle est discrétisée. Dans ce contexte, le mouvement d'un objet est assimilable à une succession de sauts dont l'amplitude est conditionnée par plusieurs paramètres tels que la vitesse de déplacement de l'objet (au sens instantané) et le pas d'échantillonnage du processus. Aussi, une collision entre deux objets est alors naturellement assimilable à la recherche d'une intersection entre ces derniers. Cependant, l'immense majorité des approches proposées à ce jour utilise le raisonnement inverse. En effet, ces approches tentent généralement de détecter l'absence d'intersection (i.e. pas d'intersection $\Rightarrow$ pas de collision). Pour ce faire, ces techniques vont consister en une séparation des objets :

soit en identifiant un plan séparant l'espace en deux zones contenant respectivement l'ensemble des entités géométriques constituant les objets concernés, soit par le calcul de la distance minimale les séparant.

Représentation des contacts en mode discret

Comme expliqué précédemment, dans le cadre de simulations évoluant en temps discret, le mouvement d'un objet mobile est assimilable à une succession de sauts en translation et rotation. Dans ce contexte, l'obtention d'un contact exact surface/surface à l'issue d'un déplacement constitue un cas numériquement dégénéré. Aussi, le contact entre deux objets peut être défini selon plusieurs formalismes. Deux approches sont principalement identifiables dans la littérature.

La première, majoritairement utilisée, consiste à assimiler la collision de deux objets à une interpénétration entre ces derniers. Dans ce premier cas, un contact est généralement défini comme un vecteur d'interpénétration (Figure 2.11). Il est à noter que, pour des objets concaves, plusieurs intersections non-connexes peuvent être présentes. Dans ce cas, la collision n'est plus décrite par un vecteur d'interpénétration unique, mais par un ensemble de vecteurs. Nous revenons sur cette problématique dans la suite de cette section.

La deuxième approche consiste à assimiler une collision à une trop grande proximité entre les objets (Figure 2.11). Dans ce deuxième cas, l'ensemble des contacts est représenté comme un champ de distances minimum locales. Plusieurs travaux présentés dans la littérature sont fondés sur cette approche. Une partie des techniques proposées dans ce contexte sont décrites dans ce chapitre (Voir les approches stochastiques en 2.3.3). On peut également citer les travaux de Johnson et al. [JC01, JW04] qui proposent un algorithme déterministe permettant l'identification de champs de distances minimum locales (LMDs) dans le cas polyédrique. Pour ce faire, ils utilisent des hiérarchies de sphères englobantes afin d'isoler les paires de primitives susceptibles de porter les entités constituant le champ. Il est à noter que, au delà de la décomposition *spatiale* qu'elles constituent, les hiérarchies utilisées dans ce contexte permettent également une prise en considération de l'orientation des primitives élémentaires qui composent les objets. En effet, chaque noeud d'une hiérarchie contient non seulement une sphère (considération spatiale) mais également un cône décrivant la zone de l'espace susceptible de contenir des éléments géométriques plus proches d'une des primitives contenues dans le nœud que de toute autre au sein de la géométrie considérée. Une présentation plus détaillée de cette approche est proposée dans le chapitre 5 du présent mémoire, l'une des implémentations de notre algorithme ayant consisté en un couplage de celui-ci avec une telle approche.

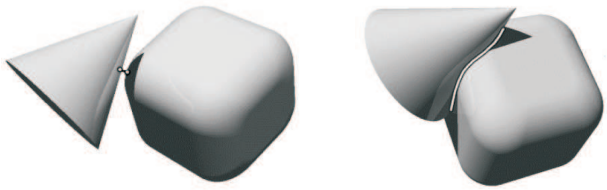


FIG. 2.11 – Représentation de la collision en mode discret : par proximité (à gauche) et par interpénétration (à droite)

Détection de l'instant de contact

Plutôt que d'assimiler la collision à une interpénétration ou une grande proximité, certaines approches s'attachent à identifier l'instant exact de la collision en vue d'obtenir une *définition exacte du contact*. Comme il l'a été expliqué précédemment, dans un contexte discret le mouvement des objets est assimilable à une succession de sauts en position. Aussi, afin d'obtenir l'instant de collision dans ce contexte, il est nécessaire de reconstituer l'information de déplacement inter-sauts.

Pour ce faire, deux types d'approches sont identifiables :

– La *détection spatio-temporelle discrète* : Elle consiste en une décomposition du déplacement effectué lors d'un saut en un ensemble de sous-déplacements en vue d'identifier, à un niveau de résolution donné, l'instant quasi-exact du contact (on parle de sur-échantillonnage). Pour ce faire, plusieurs techniques sont présentées dans la littérature. On trouve par exemple l'algorithme dit de *backtracking*, qui se base sur une mesure de la profondeur d'interpénétration entre les objets ainsi que sur une mesure de leur vitesse relative [Hah88]. D'autres techniques fonctionnent par anticipation, l'idée étant d'identifier l'intervalle durant lequel aucune collision n'est possible en se basant sur une borne inférieure minorant la distance séparant les objets et une borne supérieure majorant la vitesse relative des objets. Pour se faire, dans [GH89], les auteurs proposent de se baser sur la projection des objets sur la ligne portant la distance minimale globale les séparant.

– La *détection spatio-temporelle continue* : Elle consiste à reconstituer l'information de mouvement entre deux pas de temps. Pour cela, la trajectoire de l'objet est interpolée et paramétrée en temps. On parle parfois de *méthodes 4D* en référence à cette composante temporelle. La détection d'une collision consiste alors en l'identification du moment auquel cette dernière a lieu. Ce type d'approches est fondée

sur l'hypothèse selon laquelle la détection d'une collision basée sur une interpolation du mouvement signifie qu'il y a effectivement collision. Hors, cette hypothèse s'avère juste uniquement si la trajectoire interpolée est proche de la trajectoire réelle de l'objet (au sens continu). Les pas de temps utilisés dans le cadre de simulations interactives s'avérant généralement très petits et les vitesses de déplacement relativement réduites, cette hypothèse est dans la majorité des cas vérifiée. De plus, il est à noter que l'approximation du mouvement liée à l'interpolation n'a pas d'effet sur le comportement global des objets lors des déplacements en mode libre et que seuls les configurations post-collisions sont influencées par l'approximation.

Parmi les approches proposées dans la littérature, on peut citer, pour ce qui est du cas polyédrique, les travaux de S. Redon qui propose dans sa thèse une approximation du déplacement effectué entre deux pas de temps successifs basée sur un mouvement géodésique (i.e. un vissage) [RKS02]. Une autre approche, proposée dans [Can86], utilise une paramétrisation des trajectoires basée sur les quaternions. Ces deux approches ramènent l'identification de l'instant de collision à la résolution d'un polynôme. D'autres approches ont été proposées dans le cadre d'algorithmes utilisant d'autres types de géométries. Dans [Bar90], l'auteur propose par exemple une formulation des équations présentées précédemment dans le cas des surfaces implicites au vu d'identifier l'instant du contact.

2.4.2 Problématique des objets concaves

Dans cette partie, nous nous intéressons à l'influence des propriétés de convexité des objets simulés sur les choix algorithmiques effectués quant au processus de détection des collisions. Pour ce faire, nous nous intéressons aux algorithmes visant à la détection d'intersections vides dans le cadre de simulations mettant en scène des éléments polyédriques. Aussi, nous présentons dans un premier temps les approches les plus communément utilisées dans le cas convexe. Puis, nous discutons de l'adaptation de ces techniques dans le cas concave.

Cas des polyèdres convexes

Le cas des polyèdres convexes a grandement été exploré par la communauté scientifique travaillant sur la problématique de la détection des collisions. Aussi, il nous serait bien difficile de proposer une présentation exhaustive de l'ensemble des algorithmes proposés à ce jour et seules les approches les plus communément utilisées seront abordées.

– *Calcul de distance*

Parmi les approches de calcul de distance entre polyèdres convexes, trois techniques majeures peuvent être identifiées.

L'algorithme dit *GJK*, du nom de ses inventeurs, est probablement le plus connu. Ce dernier algorithme permet le calcul de la distance globale exacte séparant deux polyèdres convexes. Il est basé sur la différence de Minkowski avec optimisation convexe [CC86]. Afin d'éviter une construction explicite de cette dernière, l'algorithme *GJK* [GJK88] utilise une approximation par simplexes (Figure 2.12). L'algorithme *GJK* s'est vu apporter bon nombre d'améliorations, donnant lieu à l'apparition de plusieurs variantes telles que l'algorithme *EGJK* [Cam97].

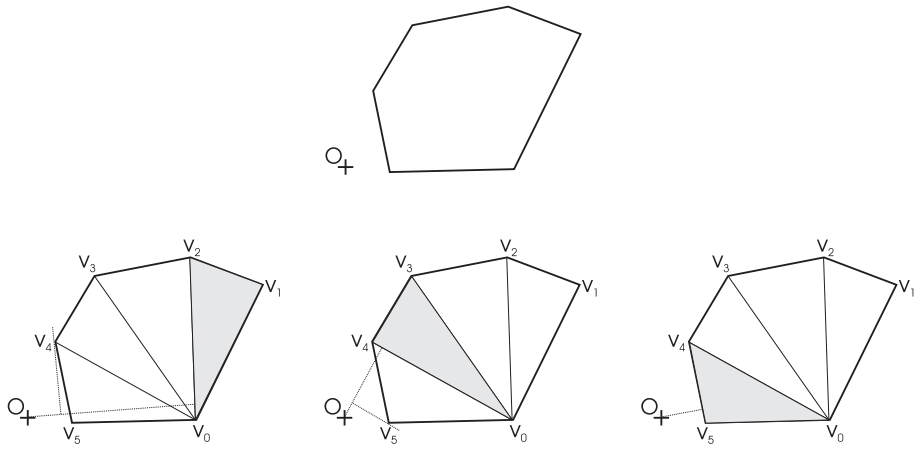


FIG. 2.12 – Fonctionnement de l'algorithme *GJK* (Cas point / polyèdre) : A chaque itération, la différence de Minkowski est calculée entre le point O et une portion du polyèdre (le simplexe). La détermination du point support du polyèdre selon cette direction permet l'identification d'un nouveau simplexe sur lequel l'opération est répétée jusqu'à convergence de l'algorithme.

L'algorithme *DK* [DK90] constitue également une technique intéressante. Il consiste en une approximation itérative des polyèdres. Il est à noter que, bien qu'il soit performant, cet algorithme ne fourni qu'une évaluation de la distance séparant les objets, puisque calculée sur des approximations de ces derniers.

Enfin, la dernière technique est fondée sur le découpage de l'espace selon des critères de proximité entre entités géométriques. Plus connue sous le nom de *Voronoï Marching* [LC91], cette technique consiste à identifier, pour chaque élément de la géométrie (arêtes et sommets), la zone de l'espace contenant les éléments dont il est le plus proche. On parle de *régions de Voronoï* (Figure 2.13).

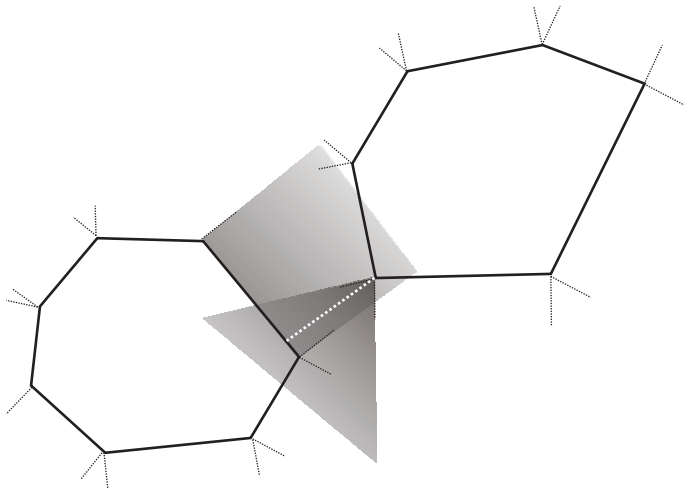


FIG. 2.13 – Fonctionnement de la technique de *Voronoï Marching* : Grâce aux informations de normales aux surfaces, la région de Voronoï de chaque entité est déterminée. La distance minimum séparant les deux polyèdres est nécessairement contenue dans les régions de Voronoï correspondant respectivement aux entités géométriques qui la délimitent.

– *Identification d'un plan séparateur*

  Comme expliqué précédemment, la deuxième approche visant à contrôler qu'il n'y a pas d'intersection entre deux objets consiste en l'identification d'un plan séparateur délimitant deux zones de l'espace contenant respectivement l'un des objets.

  Pour ce faire, deux techniques sont proposées dans la littérature. La première consiste à calculer la borne minimale de la distance de séparation [Ber99]. La deuxième propose d'identifier ce plan via des produits vectoriels [CW96].

  Comme nous le verrons par la suite, le déplacement pseudo-continu des objets engendre une certaine cohérence quant au positionnement relatif des objets d'un pas de temps à l'autre. Aussi, dans [Bar90], l'auteur propose d'utiliser les plans séparateurs calculés lors des pas précédents (si ces derniers restent pertinents bien entendu), afin de ne pas avoir à les recalculer systématiquement. Cette technique est fondée sur la notion de cohérence spatio-temporelle introduite par la suite (Cf. paragraphe 2.5).

Cas des polyèdres concaves

La détection de collisions entre objets concaves est bien plus complexe que dans le cas convexe. Ce phénomène découle du fait que, dans le cas concave, plusieurs intersections

non-connexes peuvent être présentes au sein d'une même paire d'objets. Aussi, lorsque l'on s'intéresse aux approches proposées dans la littérature, on trouve de nombreux algorithmes visant à ramener le problème à des cas convexes : soit par décomposition des objets en composantes convexes, soit par l'utilisation de leur enveloppe convexe. Ces modes opératoires s'avèrent dans certains cas complexes, voir inefficaces. En effet, la décomposition optimale d'une géométrie en éléments convexes impose que celle-ci réponde à certaines contraintes de connexité et d'orientation des normales. De plus, pour certaines formes géométriques, parfois simples, une telle décomposition génère un grand nombre d'éléments convexes (le tore en est un bon exemple), ce qui a pour effet d'augmenter la complexité géométrique de la scène. Pour ce qui est de l'utilisation de l'enveloppe convexe des objets, cette technique implique d'effectuer les tests de collisions sur des approximations de ces derniers. Aussi, cette approche peut s'avérer inadaptée pour certaines formes géométriques (un fer à cheval par exemple).

Cependant, il existe des algorithmes spécialisés fonctionnant dans le cas concave. En ce qui concerne le calcul de distance globale entre objets, les approches proposées utilisent généralement des hiérarchies de volumes englobants [Qui94, EL01]. Des approches ont également été proposées quant à l'identification de plans séparateurs [HDLM96]. Ces dernières sont généralement assimilables à des problèmes d'optimisation [Zac01].

La simulation performante de scènes contenant des éléments non-convexes complexes constitue encore aujourd'hui l'une des problématiques les plus traitées par la communauté scientifique du domaine. Dans le cadre des travaux présentés dans la suite de ce mémoire, nous proposons une solution visant, entre autre, à répondre à cette problématique.

2.4.3 Détection de collisions au sein d'environnements hétérogènes

Bien que les objets polyédriques (particulièrement à faces triangulaires) se voient massivement utilisés en réalité virtuelle, d'autres types de géométries peuvent également être mis en œuvre dans le cadre de simulations interactives. Aussi, la manipulation de nuages de points peut s'avérer très intéressante de par le fait que ce type de géométries se voit très utilisé dans différents domaines tels que l'imagerie médicale. Plusieurs approches ont été proposées dans la littérature pour la détection des collisions entre de telles entités [KZ04]. Les surfaces paramétriques constituent également un formalisme très utilisé, par exemple dans la conception mécanique assistée par ordinateur (CAO). Là aussi, plusieurs approches sont proposées dans la littérature, notamment pour la détermination de la distance minimum globale séparant deux éléments de ce type. Dans [Ber99] par exemple, une approche consistant en une adaptation de l'algorithme GJK est proposée. Dans [LM95], les auteurs proposent de ramener ce problème à la résolution d'un système

d'équations basé sur les propriétés géométriques inhérentes à l'entité recherchée (i.e. un segment formé par deux points particuliers appartenant respectivement à chacune des surfaces et matérialisant la distance minimum globale). Le problème de la mesure de distance entre surfaces implicites a également été traité dans [LM95] selon un mode opératoire similaire à celui adopté dans le cas des surfaces paramétriques. Sans rentrer dans le détail de ces approches, il semble intéressant de constater que ces dernières ne permettent pas la mise en œuvre de simulation mettant en scène des environnements hétérogènes constitués de géométries de types différents. Comme nous le verrons dans la suite du mémoire, nous nous sommes attachés à proposer une solution algorithmique fonctionnant dans un tel contexte.

2.5 Notion de cohérence spatio-temporelle

Lors de la présentation des techniques visant à la détermination de l'instant de contact (Cf. paragraphe 2.4.1), nous avons implicitement introduit la notion de cohérence spatio-temporelle. Cette dernière se fonde sur le caractère pseudo-continu des déplacements effectués par les objets simulés lors de leur mouvement. En effet, dans le contexte des simulations interactives auquel nous nous intéressons ici, les sauts réalisés par les objets entre deux pas successifs d'intégration du mouvement sont de faible amplitude, de par une fréquence de rafraîchissement élevée et des vitesses de déplacement généralement réduites. Aussi, le concept de cohérence spatio-temporelle implique, dans ce contexte, que les zones de proximité observables entre les objets à un cycle donné se situe dans le proche voisinage des zones de proximité observées lors des cycles précédents.

Ce concept de cohérence spatio-temporelle se voit utilisé dans le cadre de plusieurs travaux proposés dans la littérature. On peut par exemple citer l'algorithme présenté dans [LC91], qui permet la détermination, en temps linéaire, des deux points les plus proches de deux maillages non-déformables convexes, ou encore celui proposé dans [RGF+04] et qui consiste en une approche stochastique visant à simuler le comportement d'intestins, approche déjà introduite dans le paragraphe 2.3.3.

2.6 Conclusions

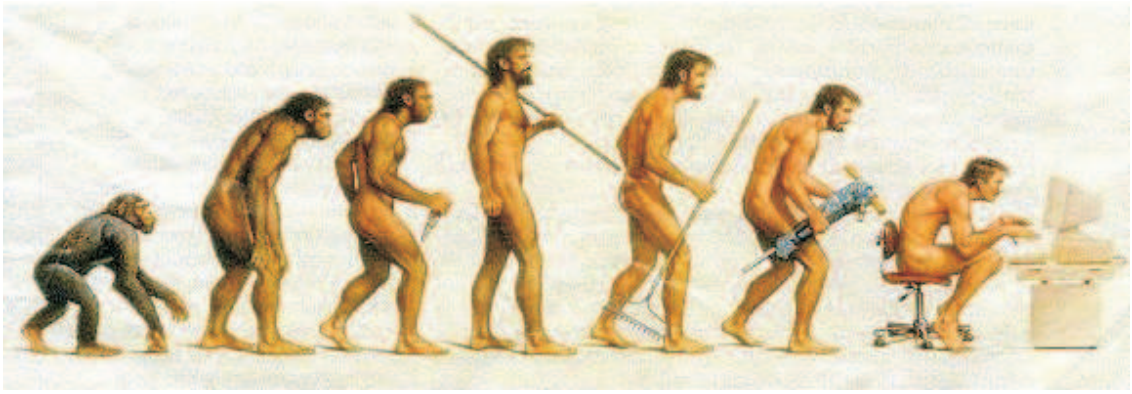
A travers ce chapitre, nous avons proposé une présentation des techniques et concepts les plus communément utilisés dans le cadre de la problématique qu'est la détection des collisions au sein d'environnements virtuels. Malgré le caractère non-exhaustif de cet état de l'art, nous avons pu apprécier l'intense activité de recherche qu'engendre cette

problématique. Cette volonté de faire sans cesse évoluer les solutions algorithmiques inhérentes à la détection des collisions est fortement motivée par la diversité des applications qui pourraient découler de l'élaboration de moteurs physiques permettant la simulation d'environnements complexes, restituant fidèlement l'ensemble des composantes de scènes observables dans le monde réel. Aussi, on observe souvent une spécialisation des algorithmes proposés, l'idée étant de répondre à la problématique de la détection des collisions dans le cadre d'applications spécifiques, présentant des caractéristiques particulières telles que celles autour desquelles cet état de l'art s'est articulé (Complexité, convexité des objets simulés, ...).

Dans le cadre des travaux menés durant cette thèse, nous avons cherché à réduire ces contraintes inhérentes à la définition des scènes manipulables dans le cadre de simulations interactives, notamment en ce qui concerne la nature et la complexité des géométries simulées. Nous avons pour cela travaillé à l'élaboration d'un algorithme fondé sur une approche évolutionniste dont une présentation et une analyse détaillée sont proposées dans la suite de ce mémoire.

Chapitre 3

Les algorithmes évolutionnaires : Concepts fondamentaux



3.1 Introduction

C'est en 1859 que Charles Darwin expose pour la première fois sa théorie de l'évolution au travers de son désormais célèbre ouvrage *L'Origine des espèces* [Dar59]. Selon lui, les êtres vivants participent à une perpétuelle compétition pour la vie qu'il nomme la *lutte pour l'existence (struggle for life)*. Partant des postulats suivant :

- chaque espèce est constituée d'un ensemble d'individus aux caractéristiques variables,
- les ressources nécessaires à la survie des individus se présentent en quantité limitée au sein de l'environnement qui les entoure,

Darwin indique que, parmi ces individus, ce sont les mieux adaptés à leur environnement qui ont le plus de chances de survivre. De plus, au delà d'une simple exposition de leur aptitude à subsister, Darwin émet l'hypothèse que ces mêmes individus vont transmettre à leur descendance leurs caractéristiques et, par conséquent, leur capacité à survivre au sein de leur environnement, assurant ainsi la pérénité de leur espèce.

Au cours des années 1930, la combinaison de la théorie de Darwin et des avancés scientifiques dans le domaine de la génétique donne naissance à une théorie synthétique de l'évolution baptisée néo-darwinisme. Cette dernière est désormais communément admise au sein de la communauté scientifique mondiale.

Disons que la finalité de cet exceptionnel algorithme qu'est celui de la vie est d'aboutir inéxorablement à des ensembles d'individus adaptés à leur environnement. Ainsi, il est assimilable à un algorithme d'optimisation visant à garantir la viabilité de la population globale via une gestion élitiste mais rationnelle de l'immense combinatoire que consitue l'ensemble des individus de notre monde. C'est au vu de cette observation que, dès le début des années 1960, plusieurs scientifiques se lancent dans l'application de la théorie néo-darwinienne à la résolution de problèmes informatiques d'optimisation. Parmi eux, citons John Holland [Hol75], communément reconnu comme l'un des pionniers de l'algorithmique génétique, dont une partie des travaux est présentée dans la suite de ce mémoire. De par leur efficacité et leur simplicité de mise en œuvre, les algorithmes évolutionnaires se voient désormais utilisés dans un grand nombre de domaines et l'engouement pour ce type de techniques ne cesse de croître au sein de la communauté scientifique (économie, théorie du contrôle optimal, théorie des jeux répétés et différentiels, ...).

De nos jours, quatre grandes familles d'algorithmes évolutionnaires sont clairement identifiables :

- Les algorithmes génétiques

– Les stratégies d'évolution
– La programmation évolutionnaire
– La programmation génétique

Rarement concurrentes au sein d'un même domaine, ces différentes approches ont chacune leurs problématiques de prédilection. Elle se différencient principalement par la nature des individus qu'elles traitent (Cf. paragraphe 3.3).

Au travers de ce chapitre, nous proposons une présentation non-exhaustive des principaux concepts inhérents à l'algorithmique évolutionnaire. Pour ce faire, nous présentons dans un premiers temps les principes généraux qui régissent ces approches, avec l'introduction des différentes entités manipulées dans ce contexte (individus, espaces de recherches, ...) et la présentation du schéma de fonctionnement classiquement mis en œuvre dans le cadre de tels algorithmes. Puis, dans un deuxième temps, nous nous intéressons aux différents formalismes adoptés quant à la représentation des éléments manipulés. Nous introduisons ainsi la notion de représentation duale des individus, au travers de leurs expressions génotypique et phénotypique. Dans une troisième partie, nous proposons une énumération des différents modes opératoires mis en œuvre quant à la manipulation des individus pour chacun des formalismes précédemment introduits. Enfin, nous nous intéressons à des notions plus avancées, telles que la parallélisation de telles approches, ou encore le maintien de la diversité génétique.

3.2 Principes généraux et notations

Un algorithme évolutionnaire consiste à faire évoluer une *population d'individus* correspondant à des solutions potentielles au problème qu'il vise à résoudre. La *qualité d'un individu* est évaluée grâce à un critère objectif visant à mesurer la pertinence de la solution à laquelle il correspond au vu du problème à résoudre. Ce critère est généralement assimilable à une fonction de l'individu dite *fonction objectif* ou encore *fonction fitness* (appellation utilisée dans la suite du document). Aussi, le but de l'algorithme consiste généralement à maximiser cette fonction *fitness* afin de converger vers une solution de qualité.

3.2.1 Notations

Ici, nous présentons les différentes entités utilisées dans la suite du mémoire ainsi que les notations utilisées.

Espace de recherche

L'espace de recherche, ou espace génotypique dans ce contexte, correspond à l'espace des solutions (*bonnes et mauvaises*) du problème à résoudre. Il est noté Ω.

Individu

Un individu est noté X_i. Il est assimilable à un vecteur appartenant à Ω. Dans cette notation, l'indice i correspond au rang de l'individu au sein de la population à laquelle il appartient.

Il nous sera parfois nécessaire de raisonner sur une composante particulière de l'individu. Pour cela, nous utiliserons la notation suivante $X_i(k)$, où k correspond au rang de la composante considérée au sein du génome de l'individu.

Population

Une population d'individus est notée Π_t. Elle est assimilable à un ensemble d'individus. Dans cette notation, l'indice t correspond au rang chronologique de la population. Le nombre d'individus contenus dans la population est noté p, soit $\Pi = \{X_0, \ldots, X_p\}$.

Afin de raisonner sur l'évolution d'un individu au cours du temps, il nous sera parfois nécessaire d'exprimer l'état d'un individu à un instant donné. Pour cela, nous adopterons la notation suivante : X_i^t où t correspond au rang chronologique de la population contenant cette instance particulière de X_i.

Fonction *fitness*

Comme expliqué précédemment, la fonction *fitness* a pour vocation d'évaluer la qualité des individus. Elle est notée $\mathscr{F}$, et la *fitness* de l'individu de rang i se note $\mathscr{F}(X_i)$. On peut remarquer que, *a priori*, $\mathscr{F}(X_i) \notin \Omega$.

Entités mathématiques diverses

- $N(a,b)$: Variable aléatoire gaussienne centrée et réduite sur l'intervalle $[a,b]$
- $U_F(a,b)$: Variable aléatoire à valeur réelle choisie avec une probabilité uniforme sur l'intervalle $[a,b]$
- $U_I(a,b)$: Variable aléatoire à valeur entière choisie avec une probabilité uniforme sur l'intervalle $[a,b]$

3.2.2 Schéma de fonctionnement

Le schéma de fonctionnement classique d'un algorithme évolutionnaire est le suivant :

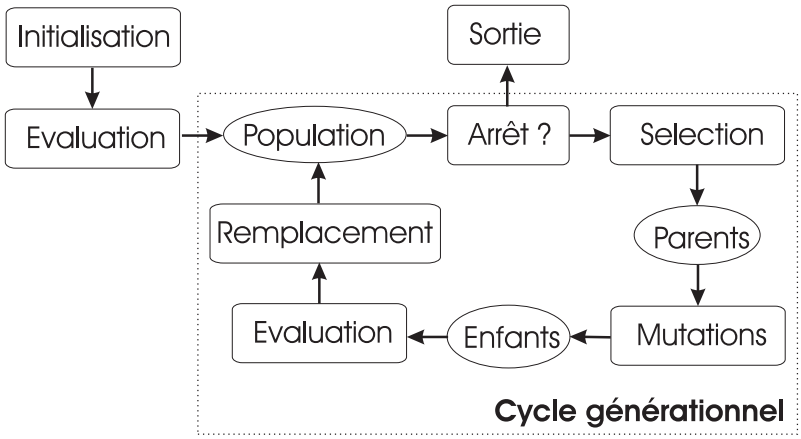


FIG. 3.1 – Schéma de fonctionnement classique d'un algorithme évolutionnaire

Durant la phase d'initialisation, la population initiale Π_0 est créée. Elle est constituée de p individus répartis aléatoirement sur l'ensemble de l'espace génotypique Ω, généralement de façon uniforme.

Une génération se décompose en 4 phases :

- **Sélection :**
  Un groupe de *géniteurs* (*i.e.* de bons individus) est sélectionné au sein de la population.

– **Mutations :**
  Les opérateurs de mutation (cf. paragraphe 3.4) sont appliqués aux individus *géniteurs* afin de générer les individus *enfants*.

– **Evaluation :**
  La qualité des individus est réévaluée via la fonction *fitness*.

– **Remplacement :**
  Les individus sont introduits dans la population. Le nombre d'individus contenus dans la population est généralement fixe ; Aussi, les nouveaux individus viendront souvent remplacer des individus de mauvaise qualité au sein de la population.

L'arrêt de l'algorithme est conditionné par un *critère d'arrêt* pouvant être de différentes natures : niveau de qualité du meilleur individu, stagnation de la *fitness*, nombre de générations maximum, ...

3.3 Représentation des individus

La recherche de la solution à un problème, quel qu'il soit, peut être assimilée à l'identification d'une entité particulière au sein d'un espace dit de recherche. Cependant, dans la plupart des cas, une multitude d'espaces de recherche peut être utilisée dans le cadre de la résolution d'un problème donné. Le choix de cet espace influe sensiblement sur la formalisation de la solution ainsi que sur l'efficacité du processus de résolution quant au temps nécessaire à sa convergence.

Les approches évolutionnaires sont particulièrement sensibles à la pertinence des choix réalisés quant à l'espace de recherche utilisé. En effet, ces algorithmes sont assimilables à des techniques d'optimisation qui, par définition, ne consistent pas en une approche déterministe fondée sur un raisonnement, mais en une approche comparative fondée sur l'exploration d'un espace de solutions potentielles.

Dans le cadre des approches évolutionnaires, c'est au sein de l'espace de recherche que vont être définis les individus, chacun d'entre eux correspondant à une solution potentielle. L'espace de recherche est d'ailleurs généralement nommé *espace génotypique*, en référence au génotype des individus. Trois grandes familles de génotypes (et donc d'espaces de recherche) peuvent être identifiées parmi les approches proposées à ce jour : les représentations binaires, réelles et par arbres. Nous en proposons un descriptif dans la suite de cette section. Sera également introduite la notion de qualité des individus.

3.3.1 Génotype d'un individu

Par analogie à la génétique des êtres vivants, le génotype d'un individu est assimilable à un ensemble de propriétés le décrivant. Comme dit précédemment, ces propriétés peuvent être formalisées de différentes manières. Ici, nous présentons les plus communément utilisées.

Représentation binaire

La représentation binaire utilise un alphabet binaire. Le génome d'un individu est assimilable à une chaîne, généralement de taille fixe, constituée de 0 et de 1 (figure 3.2). Aussi, dans ce types de formalisme, l'espace de recherche est défini comme suit :

$$\Omega = \{0,1\}^n$$

n correspondant à la taille des individus.

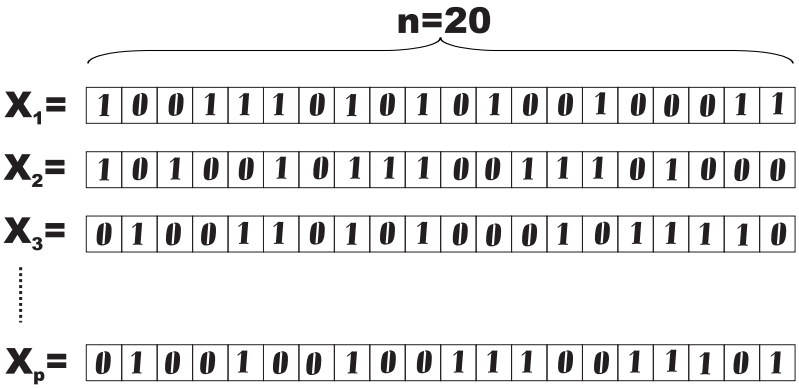


FIG. 3.2 – Exemple de représentation binaire. $n = 20$

Ce type de représentation se voit principalement utilisé dans le cadre des algorithmes génétiques qui furent introduits par Holland [Hol75] dès les années 70, et qui constituent l'une des principales catégories d'approches basées sur l'évolution artificielle.

Représentation réelle

La représentation réelle, comme son nom l'indique, utilise des composantes réelles. Le génome d'un individu est assimilable à un vecteur réel de dimension *n* (figure 3.3), et l'espace de recherche est défini comme suit :

$$\Omega = \mathbb{R}^n$$

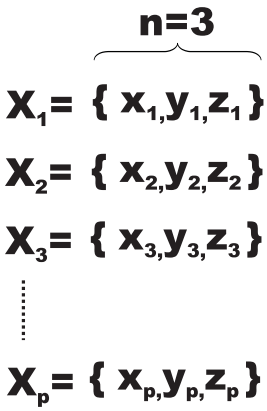


FIG. 3.3 – Exemple de représentation réelle. $n = 3$

n correspondant à la dimension de l'espace de recherche et, par conséquent, à celle des individus.

Ce type de représentation se voit principalement utilisé dans le cadre des stratégies d'évolution qui furent introduites par Rechenberg et Schwefel dans [Rec64, Sch65].

Représentation par arbres

La représentation par arbre, également connue sous le nom de *représentation fonctionnelle* consiste en une représentation des individus sous la forme de *graphes acycliques* (figure 3.4). Dans ce contexte, 3 alphabets vont généralement être utilisés : *un alphabet fonctionnel* tel que $\{+,-,\times,\div\}$, *un alphabet de variables* tel que $\{x,y,z\}$, et enfin *un alphabet de constantes* tel que $\{A,B,C\}$. Les nœuds possédant une descendance vont se voir attribuer pour valeur des fonctions, et les nœuds feuilles des variables et constantes.

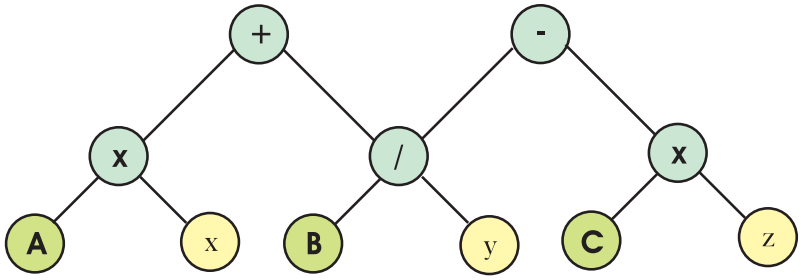


FIG. 3.4 – Exemple de représentation fonctionnelle. L'expression correspondante est la suivante : $A \times x + B/y - C \times z$

Ce type de représentation se voit principalement utilisé dans le cadre de la *programmation génétique* qui fût introduite par John Koza [Koz89, Koz92] et qui consiste en un processus visant à l'élaboration automatique de programmes.

3.3.2 Phénotype et qualité d'un individu

Comme expliqué précédemment, le génotype d'un individu est assimilable à une représentation intrinsèque de ce dernier fondée sur certaines de ses caractéristiques. Il porte l'information des gènes. C'est d'ailleurs au sein de l'espace génotypique Ω que l'ensemble des individus constituant la population est défini.

Représentation phénotypique d'un individu

Le phénotype d'un individu décrit l'ensemble des caractéristiques exprimées par ses gènes. Prenons pour exemple un être humain ; L'information génotypique d'un tel individu est constituée d'un nombre important de gènes répartis au sein d'un ensemble de 56 chromosomes. Aussi la représentation phénotypique de cet individu correspond en fait aux propriétés de l'être humain qui le matérialise, telles que la couleur de ses yeux.

En ce qui concerne les notations adoptées dans la suite du mémoire, le phénotype d'un individu X_i est noté Y_i (en fait, X_i correspond à la représentation génotypique de l'individu considéré). Remarquons que le phénotype d'un individu est déterminé à partir de sa représentation génotypique. Aussi, on peut définir la représentation phénotypique d'un individu comme une fonction de son génotype, soit :

$$Y_i = F(X_i)$$

F étant une fonction de $\mathbb{G}$ dans $\mathbb{P}$, a priori bijective ; $\mathbb{G}$ et $\mathbb{P}$ correspondant aux espaces au sein desquels sont respectivement définis les génotype et phénotype de l'individu considéré. Cette fonction est communément appelée : ***fonction de morphogénèse***.

Evaluation de la qualité d'un individu

C'est au sein de l'espace phénotypique que la qualité des individus est évaluée. Dans le cadre des stratégies d'évolution, la *fitness* d'un individu est un réel. Ainsi, dans ce contexte, la fonction *fitness* est de la forme :

$$\mathscr{F} : \mathbb{P} \rightarrow \mathbb{R}$$

$$\mathbb{P} = \mathbb{R}^{n'}$$

Le choix de la fonction *fitness* a une très grande influence sur l'efficacité de l'algorithme. Aussi, deux critères de choix vont principalement être à considérer :

– La pertinence de la fonction : soit son aptitude à fournir un résultat réellement représentatif de la qualité de l'individu
– Le coût de la fonction : soit la charge de temps et de ressources nécessaires à l'évaluation de la *fitness*

Exemple : Plutôt marathonien ou sprinter ?

Ici, nous proposons un cas d'école visant à illustrer la notion d'évaluation de la qualité d'un individu. L'idée est de proposer, pour un même algorithme, deux expressions différentes de sa fonction *fitness* visant à résoudre deux problèmes distincts. Dans notre exemple, la représentation phénotypique d'un individu correspond à un athlète dont les caractéristiques vont se résumer à son endurance et sa vitesse. Aussi, le génotype de l'individu est défini comme un vaste ensemble de gènes (ADN). Afin d'évaluer la qualité d'un individu, nous nous baserons sur deux critères inhérents à sa représentation phénotypique : Un coefficient d'aptitude à l'endurance noté C_e et un coefficient d'aptitude à atteindre de grandes vitesses noté C_v. Ainsi, nous considérons ici un sous-ensemble de la représentation phénotypique globale de l'individu : $Y_i = \{C_e, C_v\}$.

Afin d'évaluer la qualité d'un individu dans ce contexte, on peut proposer une expression générique de la fonction *fitness* sous la forme suivante :

$$\mathscr{F}(X_i) = \alpha \cdot C_e + \beta \cdot C_v$$

où α et β sont deux coefficients visant à privilégier la prise en compte de l'une ou l'autre des caractéristiques de l'individu considéré.

– Marathonien : dans le cas où le but est d'évaluer l'aptitude des individus à pratiquer le marathon, on choisira les coefficients α et β tels que $\alpha > \beta$ afin de privilégier les individus ayant une bonne endurance.
– Sprinter : dans le cas où le but est d'évaluer l'aptitude des individus à pratiquer le 100 mètres sprint, on choisira les coefficients α et β tels que $\alpha < \beta$ afin de privilégier les individus capables de courir vite.

3.4 Opérateurs de variation

Les *opérateurs de variation*, également appelés *opérateurs génétiques* en référence à la biologie, sont des opérations applicables aux individus. Ils visent à provoquer des variations au sein de la population afin de faire évoluer l'ensemble des solutions potentielles

qu'elle représente. Les opérateurs de variation sont généralement classés selon deux catégories : les *opérateurs de mutation* et les *opérateurs de croisement*. Dans cette section, nous présentons les opérateurs les plus communément utilisés en fonction des différents types d'espaces génotypiques ainsi que les notions qui s'y rattachent.

3.4.1 Notions d'exploration et d'exploitation

Les différents opérateurs de variation ont des effets variables sur les individus et, par conséquent, sur l'évolution globale de la population. Aussi, l'effet d'une variation peut être de deux natures :

- Exploration : dans ce premier cas, l'individu subit une modification importante de son patrimoine génétique. Ce type de variations va permettre une *exploration globale* de l'espace combinatoire et ainsi éviter une convergence vers les *minima locaux* inhérents aux espaces concaves.
- Exploitation : dans ce deuxième cas, l'individu subit une modification faible de son patrimoine génétique. Ce type de variations va permettre une *exploitation locale* de l'espace combinatoire à la périphérie de l'individu initial. Cette opération vise à accélérer la convergence de l'algorithme.

Ainsi, l'une des clés de l'efficacité d'un l'algorithme évolutionnaire réside dans une utilisation pertinente des différents types d'opérateurs ; le but étant d'obtenir le meilleur compromis entre rapidité de convergence et évitement des minima locaux.

3.4.2 Opérateurs de mutation

Un opérateur de mutation est assimilable à une application de Ω vers Ω visant à associer à un individu X_i^t une descendance X_i^{t+1} plus ou moins proche de l'individu initial au sens génotypique (Cf. paragraphe 3.4.1). Ce type de mutation est généralement intrinsèque à l'individu (i.e. le reste de la population n'est pas pris en compte).

Dans le cadre d'une exploitation basée sur ce type d'opérateur, il est possible de revenir à la configuration précédente de l'individu considéré si celle-ci s'avère plus pertinente que celle engendrée par la mutation. On parle alors d'*opérateur élitiste* :

$$\mathscr{F}(X_i^t) > \mathscr{F}(X_i^{t+1}) \Rightarrow X_i^{t+1} := X_i^t$$

Cas des représentations binaires

Dans le cas de la mutation d'une séquence de bits, l'opération consiste en une inversion d'une partie des bits constituant la chaîne selon une probabilité fixe, généralement

fonction de la taille de la séquence. Aussi, dans ce contexte, deux types de mutations peuvent être identifiés :

– Les mutations stochastiques : dans ce premier cas, chacun des bits de la séquence peut potentiellement être inversé selon une probabilité donnée [Bäc93] (Figure 3.5).

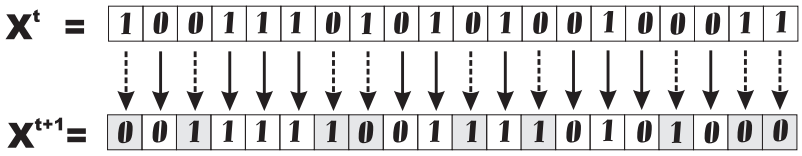


FIG. 3.5 – Exemple de mutation stochastique d'une chaîne de bits.

– Les mutations déterministes : dans ce deuxième cas, un groupe de n bits (n fixe) sélectionnés aléatoirement au sein de la séquence est inversé [Gol89] (Figure 3.6).

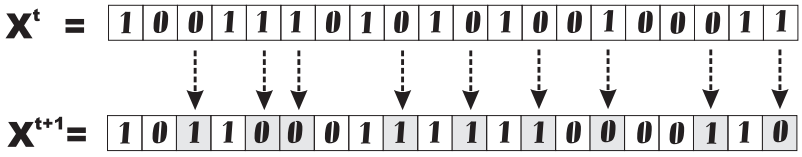


FIG. 3.6 – Exemple de mutation déterministe d'une chaîne de bits.

Cas des représentations réelles

La mutation d'un vecteur réel consiste généralement en l'ajout d'un bruit gaussien de variance σ^2 à l'individu X_i concerné [Rec64, Sch65].

Le choix du coefficient σ a une influence importante quant à l'effet de la mutation sur l'individu. Aussi, plusieurs approches sont identifiables :

– σ fixe : Dans ce premier cas, plus la valeur du coefficient σ est élevée, plus la mutation est exploratoire. *A contrario*, plus la valeur du coefficient σ est faible, plus la mutation favorise une convergence locale de l'individu concerné.

– Approche adaptative : Dans ce deuxième cas, la valeur du coefficient σ évolue au cours de l'exécution. Plusieurs heuristiques peuvent être implémentées quant à la manière de faire évoluer le coefficients σ. L'une des plus classiques consiste à choisir des valeurs de σ inversement proportionnelles à la qualité des individus. Aussi plus la *fitness* d'un individu donné va s'améliorer, plus son amplitude de variation va réduire. On parle d'un *refroidissement* de l'individu.

– Approche auto-adaptative : Dans ce dernier cas, le génotype des individus contient les paramètres de variation. Aussi les individus transmettent à leur descendance non plus seulement les caractéristiques des solutions potentielles qu'ils représentent, mais également les paramètres inhérents à leur manière d'évoluer.

Dans la majorité des cas, l'amplitude de variation n'est pas la même pour l'ensemble des composantes du vecteur décrivant l'individu. Aussi, par opposition aux opérateurs de mutation isotropes dont la variance est conditionnée par un coefficient σ unique tels que

$$X_i^{t+1}(k) = X_i^t(k) + \sigma\, N(0,1), \forall k \in [0,n],$$

les opérateurs anisotropes vont consister en une variation dont l'amplitude varie selon la composante (Cf. figure 3.7) :

$$X_i^{t+1}(k) = X_i^t(k) + \sigma_k\, N(0,1), \forall k \in [0,n].$$

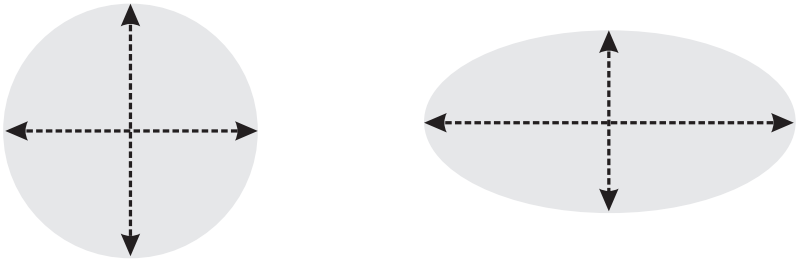


FIG. 3.7 – Illustration dans un cas 2D de l'espace de mutation dans les cas isotrope (à gauche) et anisotrope (à droite).

Cas des représentations par arbres

Comme expliqué précédemment (Cf. paragraphe 3.3.1), les arbres utilisés dans ce contexte sont constitués de trois types de nœuds : fonctions, variables et constantes. Aussi, les opérateurs de mutations utilisés dans le cadre d'algorithmes de programmation génétique, principales approches basées sur ce type de représentations, peuvent être de différentes natures [Ang96] :

– Remplacement de nœuds : Il consiste à remplacer l'un des nœuds de l'arbre par un nœud de même nature. Dans le cas particulier des nœuds fonctionnels, la nouvelle opération doit avoir la même arité que celle correspondant au nœud initial afin d'assurer la validité de la nouvelle configuration.
– Remplacement de sous-arbres : Il consiste à remplacer un sous-ensemble de l'arbre par un nouvel arbre généré aléatoirement.

– Insertion d'un nœud ou d'un sous-arbre : Elle consiste en l'ajout d'un nœud ou d'un sous arbre. Afin d'être valide, la greffe doit être réalisée au niveau d'un nœud fonctionnel non-saturé (i.e. d'arité suffisante pour supporter le nouvel élément).

– Promotion d'un nœud : Elle consiste à remplacer l'un des nœuds de l'arbre par l'un de ses enfants (le promu). Cette opération n'est pas réalisable au niveau des feuilles de l'arbre, le promu devant nécessairement être de type fonctionnel.

– Variation des constantes : Elle consiste à faire évoluer la valeur d'un nœud *constante*. La variation est généralement réalisée par l'ajout du variable de loi normale à la valeur initiale de la constante.

3.4.3 Opérateurs de croisement

Les opérateurs de croisement consistent à combiner les propriétés de plusieurs individus, dits parents ou géniteurs, afin d'obtenir de nouveaux individus dits enfants. Bien que les opérateurs de ce type soient bien souvent définis comme des applications de $\Omega \times \Omega$ vers $\Omega \times \Omega$ (i.e. deux parents donnent deux enfants), on peut plus généralement les assimiler à des applications de Ω^k vers $\Omega^{k'}$.

Dans le cas général, les opérateurs de croisement prennent pour géniteurs des individus de bonne qualité (i.e. ayant une valeur de *fitness* élevée). Aussi, les individus issus des croisements vont posséder de bonnes propriétés héritées de leurs parents ce qui va contribuer à accélérer la convergence globale de l'algorithme. En ce sens, ces opérateurs sont clairement assimilables à des *opérateurs d'exploitation* (Cf. paragraphe 3.4.1).

Comme pour les opérateurs de mutations abordés dans la partie précédente, les opérateurs de croisement peuvent prendre plusieurs formes en fonction du type de représentation utilisé. Nous présentons ici les procédés les plus communément utilisés.

Cas des représentations binaires

Dans le cas de chaînes de bits, deux approches sont principalement identifiables quant à l'implémentation des opérateurs de croisement (Figure 3.8). Le premier type d'approches consiste à choisir aléatoirement un ou plusieurs points délimitant les sous-chaînes à commuter au sein des séquences de bits initiales [Hol75, DS92]. La deuxième approche consiste en une commutation uniforme [SS89] (chaque rang a une chance sur deux d'être commuté).

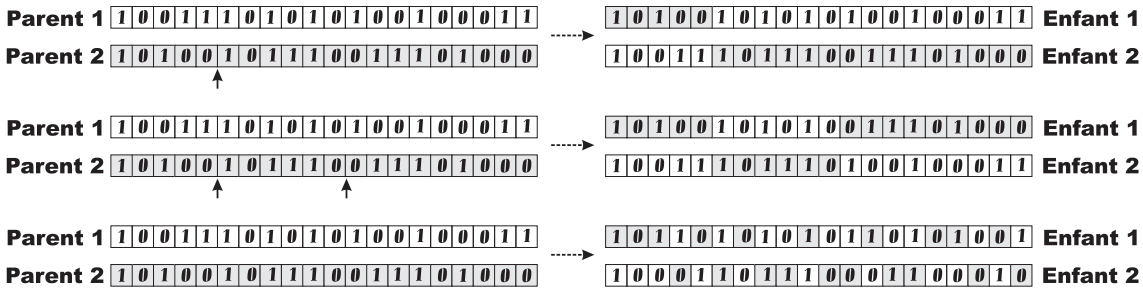


FIG. 3.8 – Exemple d'opérateurs de croisement dans le cas de représentations binaires : (De haut en bas) croisement en 1 point, croisement en 2 points et croisement uniforme.

Cas des représentations réelles

Le croisement de vecteurs réels est généralement réalisés entre deux parents et a pour résultat un enfant unique (soit une application de $\Omega \times \Omega$ vers Ω). Le plus communément utilisé est le croisement dit *barycentrique*. Ce dernier consiste en une combinaison linéaire des deux individus géniteurs, soit de manière homogène sur l'ensemble des composantes tel que

$$\forall k \in [0,n], \quad X_{enfant}(k) = \alpha X_{parent1}(k) + (1-\alpha) X_{parent2}(k) | \alpha = U_F([0,1]),$$

soit de manière hétérogène, composante à composante, tel que,

$$\forall k \in [0,n], \quad X_{enfant}(k) = \alpha_k X_{parent1}(k) + (1-\alpha_k) X_{parent2}(k) | \alpha_k = U_F([0,1]).$$

Cas des représentations par arbres

Dans le cas des représentations par arbres, le croisement consiste généralement en un échange de sous-arbres entre deux parents. Comme dans le cas des opérateurs de mutations inhérents à ce type de représentations, le croisement doit être soumis à certaines contraintes afin d'assurer la validité des individus enfants issus de l'opération. La figure 3.9 présente un exemple d'implémentation de ce type d'opérateur.

3.5 Processus de sélection

La sélection des parents au sein de la population est une étape décisive quant au comportement général de l'algorithme. Le choix du mode de sélection ramène à la problématique du compromis exploration/exploitation présentée dans §3.4.1. En effet, le niveau d'exigence inhérent au mode de sélection va fortement influer sur la tendance générale de l'algorithme : explorer l'ensemble de la combinatoire (niveau d'exigence faible) ou

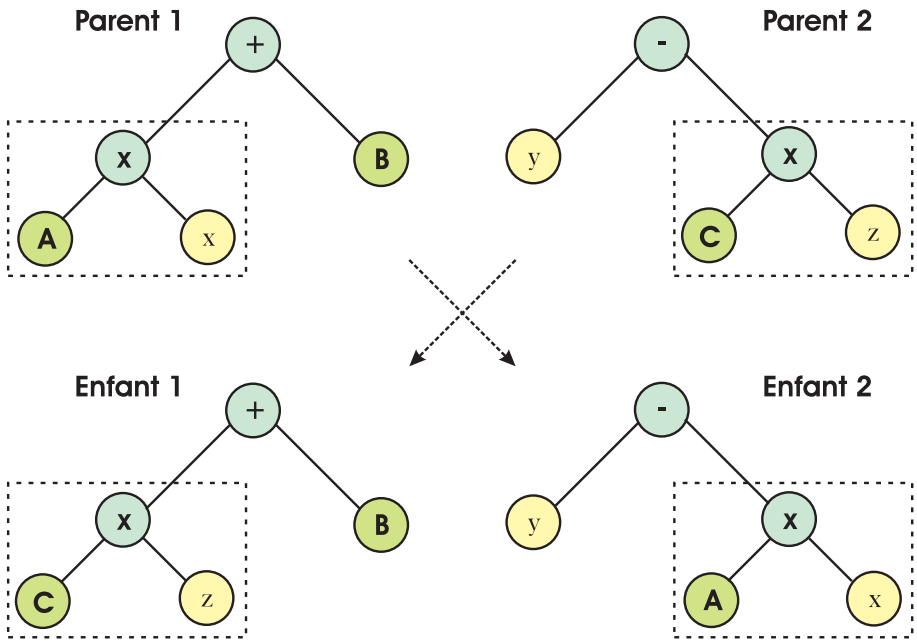


FIG. 3.9 – Exemple d'opérateurs de croisement dans le cas de représentations par arbres

privilégier la rapidité de convergence (niveau d'exigence élevé). On parle de *pression sélective*.

Les méthodes de sélection des individus parents sont généralement classées en deux catégories : les méthodes déterministes d'une part, les méthodes stochastiques d'autre part. Dans la suite, nous présentons brièvement les principales approches proposées dans la littérature.

3.5.1 Sélection déterministe

La sélection déterministe consiste simplement à choisir pour parents les individus possédant les meilleurs *fitness*. Ainsi, les moins bons individus sont exclus et remplacés par les enfants des meilleurs. Cette approche, très élitiste, s'avère souvent inadaptée car elle est susceptible d'entraîner des convergences locales sans prise en compte de l'ensemble de la combinatoire.

3.5.2 Sélection stochastique

Les processus de sélection basés sur des approches stochastiques consistent à choisir les parents selon des critères probabilistes en attribuant à chacun des individus une probabilité de sélection. Afin de privilégier les individus présentant de bonnes propriétés, la probabilité de sélection attribuée à un individu donné est généralement fonction de

sa *fitness*. Bien que privilégiant certains individus, les approches stochastiques sont bien moins élitistes que les approches déterministes. En ce sens, elles s'avèrent souvent mieux adaptées aux problèmes présentant des espaces de recherches concaves (i.e. contenant des minima locaux).

Sélection à la roulette

Ce mode de sélection fait référence à la roulette de casino. L'idée est de « lancer une bille » sur une roulette possédant autant de cases qu'il y a d'individus au sein de la population. Afin de privilégier les individus possédant les meilleurs caractéristiques, les tailles des cases sont généralement ajustées en fonction des *fitness* des individus (i.e. plus la fitness d'un individu est grande, plus sa case est large). Cette technique de sélection est illustrée par la figure 3.10.

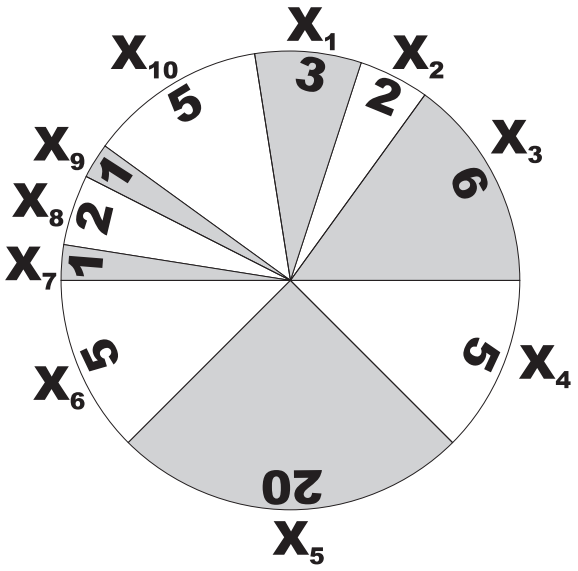


FIG. 3.10 – Mode de sélection basé sur la roulette

Sélection par rang

Très semblable au mode de sélection à la roulette, la sélection par rang consiste à attribuer à chaque individu une probabilité de sélection proportionnelle à son rang au sein de la population.

Sélection par tournoi

Ce dernier mode de sélection consiste en l'organisation de *confrontations inter-individus* au sein de la population. L'idée est d'isoler des individus présentant de bonnes caractéris-

tiques via une approche comparative. Pour se faire, des *combats* entre individus, consistant en une comparaison de leur *fitness*, sont organisés. Deux types d'implémentations sont identifiables :

- Tournoi classique : Ici, l'individu présentant la meilleure *fitness* au sein d'un sous-ensemble d'individus choisis de façon aléatoire est isolé au vu d'être utilisé comme parent dans la suite de la génération. L'opération est répétée autant de fois qu'il est nécessaire afin de constituer le groupe d'individus parents.
- Tournoi stochastique : Ici, chaque individu est confronté à un ou plusieurs adversaires. A chaque victoire d'un individu, celui-ci se voit attribuer un point supplémentaire. Les individus ayant obtenu le plus de points à l'issue du tournoi constituent le groupe des parents.

3.6 Notions avancées

Dans ce qui précède, nous avons proposé une présentation générale des techniques et concepts fondamentaux de l'algorithmique évolutionniste afin d'offrir au lecteur non-initié les éléments nécessaires à la compréhension des approches exposées dans la suite de ce mémoire. Dans cette dernière section, nous introduisons quelques notions supplémentaires qui, bien que quelque peu avancées, seront pour partie également exploitées par la suite.

3.6.1 Composition de la solution : L'approche parisienne

Dans l'introduction du paragraphe dédié à la représentation des individus (Cf. paragraphe 3.3), nous avons assimilé la recherche de la solution d'un problème donné à l'identification d'une entité particulière au sein d'un espace de recherche. Dans le contexte des algorithmes évolutionnaires, cette pseudo-définition sous-entend que la solution du problème peut être donnée par un seul et même individu (*a priori* le meilleur individu de la population).

L'approche parisienne [CLRS99] consiste à considérer chacun des individus de la population comme une partie de la solution. Dans ce contexte, la solution ne se résume plus à un individu unique, mais à un sous-ensemble de la population (voir son intégralité). Ce type d'approches s'avère particulièrement intéressant lorsque la solution du problème est difficilement formalisable en une seule entité (i.e. un individu unique). Il entraîne en effet une décomposition de celle-ci en un ensemble d'entités plus simples, et par conséquent plus facilement manipulables.

3.6.2 Maintien de la diversité génétique

Comme il l'a été dit précédemment, le bon fonctionnement d'un algorithme évolutionnaire est conditionné par une implémentation pertinente des sous-processus qui le composent au vu d'obtenir un bon compromis entre temps de convergence et aptitude à l'exploration de l'ensemble de la combinatoire (Cf. paragraphe 3.4.1, §3.5). Cependant, dans bon nombre de cas, la convergence de l'algorithme vers des minima locaux est inévitable tant ces derniers se trouvent en nombre important au sein de l'espace de recherche.

Pour cela, plusieurs approches visant à préserver la diversité génétique au sein de la population ont été proposées dans la littérature. L'immense majorité de ces techniques découle du *principe de nichage* introduit par John Holland dans le cadre de ses travaux portant sur l'algorithmique évolutionnaire [Hol75]. Ce dernier principe s'inspire du principe des *niches écologiques* tel qu'il est défini en biologie. Il consiste à considérer que les individus présentant d'importantes similarités (i.e occupant une même zone de l'espace de recherche) partagent les ressources disponibles localement. En d'autres termes, la qualité d'un individu n'est plus évaluée de manière intrinsèque, mais en prenant en compte la notion de partage des ressources. Aussi, plus les individus vont s'amasser en un lieu de l'espace de recherche, plus la quantité de ressources attribuée à chaque individu va être moindre.

Ce principe peut être implémenté de différentes manières. La plus communément utilisée est sans nul doute *le partage*, plus connu sous le nom de *fitness sharing* [GR87]. Cette première approche consiste en une modification de la fonction *fitness* des individus. L'idée est de prendre en considération la densité d'individus présents dans le voisinage direct d'un individu lors de l'évaluation de sa qualité. Aussi, dans ce contexte, la fonction *fitness* d'un individu est de la forme suivante :

$$\mathscr{F}'(X_i) = \frac{\mathscr{F}(X_i)}{\sum_{j=0, j\neq i}^{p} sh(X_i, X_j)}$$

avec

$$sh(X_i, X_j) = \begin{cases} \left[\frac{d(X_i,X_j)}{\sigma_{share}}\right]^{\alpha} & si\ d(X_i, X_j) < \sigma_{share} \\ 0 & sinon \end{cases}$$

où σ_{share} est assimilable à un seuil de ressemblance au delà duquel deux individus sont considérés comme éloignés (on parle de rayon de partage), α est un réel permettant d'ajuster l'effet du *sharing* sur la fonction *fitness*, et d est une *fonction distance* permettant d'évaluer le niveau de ressemblance de deux individus (fonction dépendante de type de représentation).

La technique dite d'*éclaircissement* [Pet96] est également largement utilisée. Contrairement au *fitness sharing*, l'éclaircissement consiste à ne préserver qu'un seul individu au

sein de chaque niche. Aussi, dans la pratique, cette technique va consister à annuler la *fitness* de l'ensemble des individus appartenant à une même niche à l'exception du meilleur d'entre eux. Selon son initiateur, la technique d'éclaircissement a pour principal avantage une diminution notable de la complexité de l'algorithme qui passe de $O(p^2)$ à $O(k \times p)$ où k est le nombre de niches (généralement bien inférieur au nombre d'individus).

Bien d'autres approches visant à assurer le maintient de la diversité génétique ont été proposées dans la littérature. On peut citer par exemple *le surpeuplement* [DeJ75] ou encore *le nichage séquentiel* [BBR93].

3.6.3 Parallélisation : Méthodes de sous-populations

Du concept de niches écologiques introduit précédemment découle naturellement la notion de *parallélisme* entre les évolutions respectives de plusieurs *sous-ensembles d'individus* répartis au sein de l'espace de recherche. On parle de *sous-populations*.

Là encore, plusieurs approches ont été proposées quant à l'implémentation d'algorithmes visant à paralléliser l'exploration de l'espace de recherche. Le modèle en îlots [GWA92] est l'une de ces approches. Il consiste à faire évoluer indépendamment plusieurs sous-populations au sein d'un même espace de recherche avec pour seuls liens inter-populations des échanges ponctuels d'individus (on parle de migration).

Au delà des aspects algorithmiques, ce type d'approches permet l'exploitation d'architectures matérielles distribuées. Aussi, cette prédisposition à la parallélisation est une propriété intéressante des approches évolutionnaires qui visent, rappelons-le, à résoudre des problèmes d'optimisation complexes justifiant souvent l'utilisation de *clusters de calculateurs* (génétique, météorologie, ...).

3.7 Conclusions

Les approches basées sur l'évolution artificielle, telles qu'elles ont été présentées dans ce chapitre, sont de puissants outils d'optimisation, permettant la résolution de problèmes difficiles (voir impossibles) à formaliser dans le cadre des approches déterministes classiques.

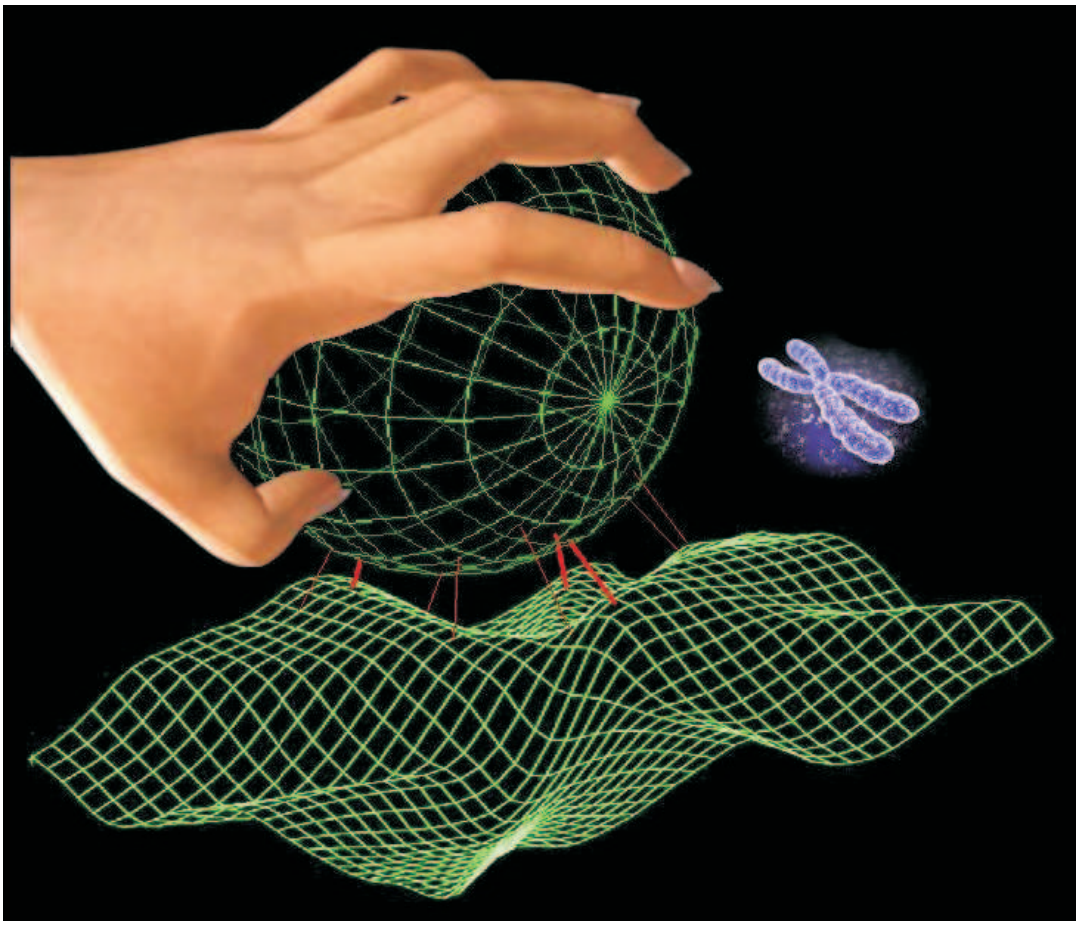
La popularité de ces approches n'a de cesse de croître au sein de bon nombre de communautés scientifiques de par la facilité d'implémentation de ces techniques. Cependant, il est à noter que, bien qu'un algorithme de ce type soit généralement simple à mettre en œuvre (en termes de développements), les choix inhérents à la nature et au paramétrage des opérateurs composant le processus global vont s'avérer cruciaux quant à l'efficacité et

la robustesse de ce dernier. De plus, nous avons vu que, pour un problème donné, plusieurs formalisations sont généralement possibles, chacune s'avérant plus ou moins adaptée.

Dans la suite du mémoire, nous proposons une approche basée sur l'algorithmique évolutionnaire, et plus particulièrement sur les stratégies d'évolution, visant à accélérer la détection des collisions au sein d'environnements virtuels denses. Le lecteur pourra alors juger de la généricité d'une partie des techniques que nous avons présentées dans ce chapitre ainsi que de leur puissance au vu de la résolution du problème complexe et difficilement formalisable qu'est celui de la détection des collisions.

Chapitre 4

Application des algorithmes évolutionnaires à la détection de collisions



4.1 Introduction : De Darwin à la détection de collisions

Dans le cadre du chapitre consacré à la présentation des approches et concepts inhérents à la détection des collisions, et particulièrement dans la section introduisant les techniques visant à son accélération, nous avons mis en évidence plusieurs limitations de ces dernières. Comme il l'a été expliqué, ces techniques ont pour principe de réduire la combinatoire à considérer au sein de l'espace géométrique formé par les modèles dont le comportement est simulé. Ainsi, elles vont permettre une réduction plus ou moins importante de la complexité inhérente au processus ultime qu'est l'identification de l'ensemble des informations nécessaires à la gestion des contraintes unilatérales.

Toutes les limitations relatives à l'utilisation des algorithmes proposés dans ce contexte sont intimement liées au fait que ces derniers sont fondés sur des raisonnements basés sur la représentation géométrique des objets. Nous entendons par là que la complexité de ces algorithmes est fonction de plusieurs caractéristiques de la scène. En effet, mêmes les approches fondées sur une approximation géométrique des éléments constituant la scène dépendent de la complexité globale de celle-ci et de la nature des entités qui la composent, et cela tant au niveau de l'occupation mémoire que des performances observables en cours de simulation.

A partir de cette observation, il parait naturel de chercher à raisonner dans un autre espace que celui formé par les objets qui constituent la scène. L'idée est de trouver une formalisation du problème utilisant des entités autres que celles constituant les géométries. Dans le cadre des approches stochastiques présentées précédemment, nous avons abordé des algorithmes mettant en œuvre de telles entités [RGF+04, KNF04]. En effet, les entités, dites *paires*, utilisées dans ce contexte permettent la détermination de tout ou partie de l'ensemble des contacts à considérer dans le cadre de la gestion des contraintes unilatérales. Ainsi, le processus consiste en la réalisation de plusieurs descentes de gradient locales, visant à l'identification de minima locaux ; les paires matérialisant les plus petites distances étant conservées d'un cycle à l'autre afin de prendre en considération la cohérence spatio-temporelle. Dans [JW04], les auteurs proposent également l'implémentation d'un tel processus dans le cadre d'une boucle haute fréquence permettant de satisfaire aux contraintes inhérentes à une manipulation haptique. Les positions initiales des *paires* sont ici conditionnées par les informations issues d'une boucle basse fréquence mettant en œuvre un processus basé sur des hiérarchies de volumes englobants.

Application des approches évolutionnaires dans ce contexte

Raisonnons désormais en nous basant sur un formalisme tel que celui abordé précédemment. La recherche du champ de distances constituant la solution au problème (i.e. l'identification des contacts) est ainsi assimilable à la mise en évidence d'un ensemble fini de paires au sein de l'espace combinatoire que constitue l'ensemble des paires possibles. Aussi, cette tâche peut être assimilée à un problème d'optimisation ; d'où l'idée d'élaborer un algorithme basé sur les approches évolutionnaires qui, comme il l'a été expliqué précédemment, s'avèrent bien souvent être des outils très efficaces quant à la résolution de problèmes de cette nature.

Dans la suite de ce chapitre, nous proposons un nouvel algorithme basé sur les stratégies d'évolution visant à résoudre ce problème. Cet algorithme, baptisé ESPIONS, constitue la principale contribution quant aux travaux réalisés dans le cadre de cette thèse. ESPIONS peut se voir utilisé de différentes manières (Cf. chapitre 5). Cependant, on peut le définir comme un processus visant à identifier des *champs de distances minimum locales* entre des objets virtuels de natures diverses. Aussi, dans ce chapitre, nous présentons ESPIONS d'une manière générale, sans rentrer dans le détail d'applications spécifiques. Il y est abordé comme un algorithme générique visant à l'accélération de la détection des collisions au sein de simulation de scènes complexes, tant au niveau de la complexité intrinsèque des objets que de leur nombre, et éventuellement hétérogènes quant à la nature géométrique des objets simulés. Aussi, différents modes opératoires visant à son intégration au sein de simulateurs dynamiques, et plus particulièrement de pipelines dédiés à l'accélération de la détection des collisions, sont présentés.

4.2 Représentation des individus

Dans cette section, nous définissons les individus mis en œuvre dans le cadre de l'algorithme ESPIONS. Dans un premier temps, nous en proposons une description générique visant à bien identifier la nature de ces entités. Puis, nous traitons de leur représentation génotypique. Pour ce faire, nous proposons plusieurs formalismes rendant possible l'utilisation de l'algorithme ESPIONS dans le cadre de simulations faisant intervenir divers types de géométries. Par la suite, nous définissons l'expression phénotypique d'un individu, qui vise, rappelons-le, à matérialiser l'ensemble des caractéristiques exprimées par les gènes de ce dernier. A ce niveau sera introduite la fonction de morphogénèse permettant de déterminer le phénotype d'un individu à partir de son expression génotypique. Enfin, nous présentons la fonction *fitness* retenue au vu d'évaluer la qualité des individus.

4.2.1 Définition générique

Avant de passer à la définition des différents formalismes retenus quant à la représentation génotypique des individus, il semble intéressant d'identifier la nature de ces entités que nous allons manipuler afin de répondre au problème qui nous intéresse (i.e. l'identification de champs de distances minimum locales). Comme nous l'avons vu précédemment, un algorithme évolutionnaire consiste à faire évoluer une *population d'individus* correspondant à des solutions potentielles au problème qu'il vise à résoudre. A également été abordé dans 3.6.1 le concept d'*approche parisienne*, introduisant l'idée d'une composition de la solution (i.e la solution est donnée par un sous-ensemble de la population et non pas un individu unique). ESPIONS est fondé sur une telle approche.

Ainsi, les individus utilisés dans le cadre d'ESPIONS sont assimilables à des paires de points 3D évoluant respectivement à la surface de géométries distinctes. En ce qui concerne le vocabulaire, les individus ont été baptisés *pions*, en référence aux particules réalisant l'interaction forte entre les nucléons à l'échelle atomique [Yuk35]. Aussi, les points constituant les individus seront nommés *nucléons*.

ESPIONS** = **E**volution **S**trategy on **PIONS

4.2.2 Représentation génotypique

D'une façon générale, le génotype d'un individu X_i se définit comme un couple de deux vecteurs tel que :

$$X_i = \left\{ \begin{array}{c} \mu_1^i \\ \mu_2^i \end{array} \right\}$$

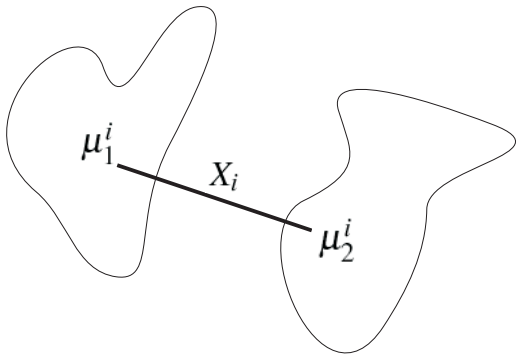


FIG. 4.1 – Représentation générique d'un pion

où μ_1^i et μ_2^i sont deux vecteurs, de dimensions éventuellement différentes, correspondant aux coordonnées paramétriques des nucléons constituant le pion considéré au sein des géométries à la surface desquelles ils sont respectivement contraints d'évoluer.

Adaptation à la nature des géométries

Contrairement à l'immense majorité des algorithmes proposés à ce jour pour l'accélération de la détection des collisions, ESPIONS présente la capacité de fonctionner sur des données géométriques de natures diverses. Cette particularité de l'algorithme permet son implémentation dans le cadre de simulations mettant en scène des objets de différentes natures, tels que des polyèdres ou encore des nuages de points.

Dans ce contexte, il s'avère nécessaire d'exprimer la position des nucléons au sein des géométries à la surface desquelles ils évoluent (au sens paramétrique). Pour ce faire, différents formalismes peuvent être adoptés en fonction de la nature géométrique de l'objet considéré. Dans la suite, nous exposons ceux utilisés dans le cadre de notre implémentation de l'algorithme pour les polyèdres et les nuages de points. Est également proposée une formalisation permettant l'utilisation d'ESPIONS dans le cadre de scène contenant des surfaces paramétriques. Il est à noter que cette présentation n'est pas exhaustive. Nous entendons par là que d'autres types de géométries pourraient être gérés dans le cadre d'ESPIONS, la seule contrainte étant d'être en mesure d'exprimer la position d'un point à la surface de l'objet considéré.

Cas polyédrique :

Dans le cas polyédrique, le formalisme adopté pour la définition d'un nucléon est le

suivant :

$$\mu_k^i = \left\{ \begin{array}{c} G_{id} \\ T_{id} \\ \alpha \\ \beta \end{array} \right\}$$

avec

$$0 \leq \alpha \leq 1, \;\; 0 \leq \beta \leq 1 \;\; et \;\; \alpha + \beta \leq 1$$

où G_{id} est un indice identifiant la géométrie visitée, T_{id} un indice identifiant la face sur laquelle le nucléon se situe, α et β deux scalaires donnant les coordonnées du nucléon exprimées dans la base formée par deux des arêtes de la face considérée. Les conditions relatives aux coordonnées α et β permettent de garantir l'appartenance du nucléon à la face. La figure 4.2 illustre ce formalisme.

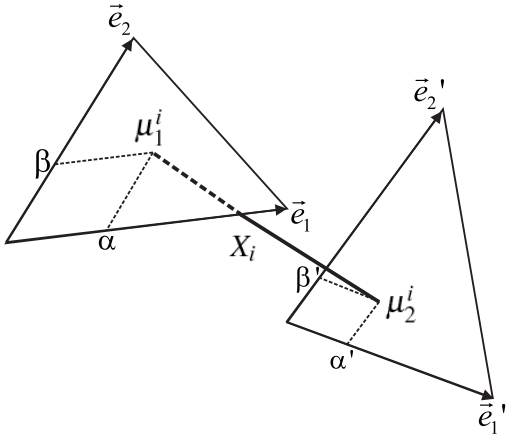


FIG. 4.2 – Représentation d'un pion dans le cas polyédrique

Sur la figure 4.3, on peut observer des captures d'écran réalisées au cours d'une simulation utilisant ESPIONS au sein d'un environnement composés de polyèdres issus de la CAO (modèles proposés par PSA-Peugeot-Citroën).

Notons que le formalisme proposé ici considère des objets polyédriques composés de faces triangulaires. Il est envisageable de généraliser celui-ci aux cas d'objets polyédriques d'autres natures tels que les polyèdres composés de quadrilatères (ou *quads*). En effet, dans ce dernier cas, seules les contraintes visant à garantir l'appartenance du nucléon à la face considérée seront à reformuler.

Cas des nuages de points :

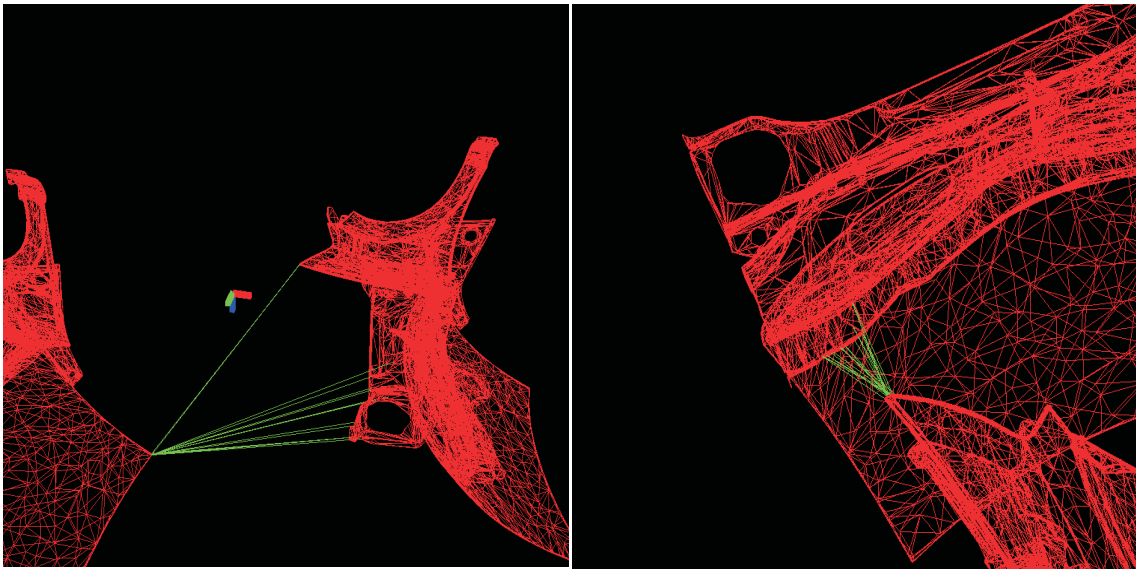


FIG. 4.3 – Simulation utilisant des données polyédriques. Les modèles sont issus de données CAO proposées par PSA-Peugeot-Citroën (Capot et aile d'une Citroën C3).

Dans le cas des nuages de points, un nucléon se définit de la façon suivante :

$$\mu_k^i = \left\{ \begin{array}{c} G_{id} \\ P_{id} \end{array} \right\}$$

où G_{id} est un indice identifiant la géométrie visitée et P_{id} un indice identifiant le point sur lequel le nucléon se situe au sein du nuage considéré. La figure 4.4 illustre cette définition.

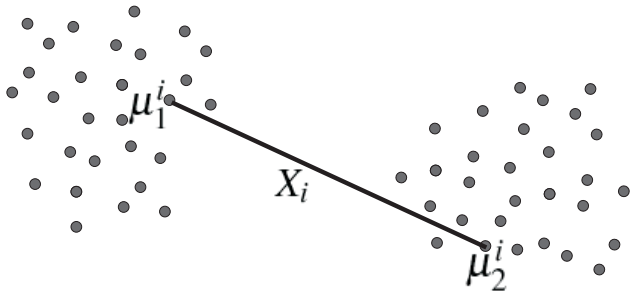


FIG. 4.4 – Représentation d'un pion dans le cas des nuages de points

Les nuages de points sont souvent obtenus via des techniques d'imagerie 3D utilisant des scanners spécifiques. Aussi, la figure 4.5 illustre l'utilisation d'ESPIONS dans un tel contexte. Dans la simulation présentée, un modèle géométrique de col de fémur issu d'un scanner 3D est traité.

Il est à noter que la représentation des individus proposée dans ce contexte permet seulement la détermination de distances minimum locales supportées par les points constituant les objets. Nous ne proposons pas là de solution de reconstruction des surfaces approximées par ces points telle que proposée dans le cadre d'autres algorithmes tels que *GJK*. La détermination d'informations de contact pertinentes avec ce mode opératoire sous-entend que la résolution des nuages utilisés, en termes de densité de points, s'avère suffisamment élevée.

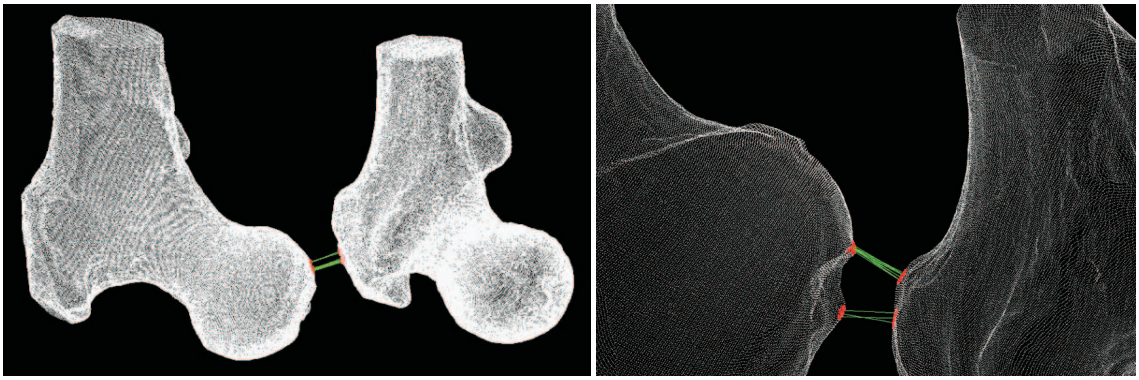


FIG. 4.5 – Simulation utilisant des nuages de points. Ici les modèles sont deux cols de fémur. On peut observer des pions (en vert) ainsi que les nucléons qui les composent (en rouge).

Cas des surfaces paramétriques :

Dans le cas des surfaces paramétriques, la définition d'un nucléon est donnée par :

$$\mu_k^i = \left\{ \begin{array}{c} G_{id} \\ C_{id} \\ u \\ v \end{array} \right\}$$

où G_{id} est un indice identifiant la géométrie visitée, C_{id} un indice identifiant le carreau sur lequel le nucléon se situe (dans l'hypothèse où l'objet considéré est composé de plusieurs carreaux), u et v deux scalaires donnant les coordonnées du nucléon au sein du carreau considéré. La figure 4.6 illustre cette définition. Il est à noter que, pareillement aux paramètres α et β dans le cas polyédrique, les coordonnées u et v doivent être saturées de manière à garantir l'appartenance du nucléon au carreau considéré. On a généralement une normalisation croisée du paramétrage (i.e. $0 \leq u \leq 1$ et $0 \leq v \leq 1$).

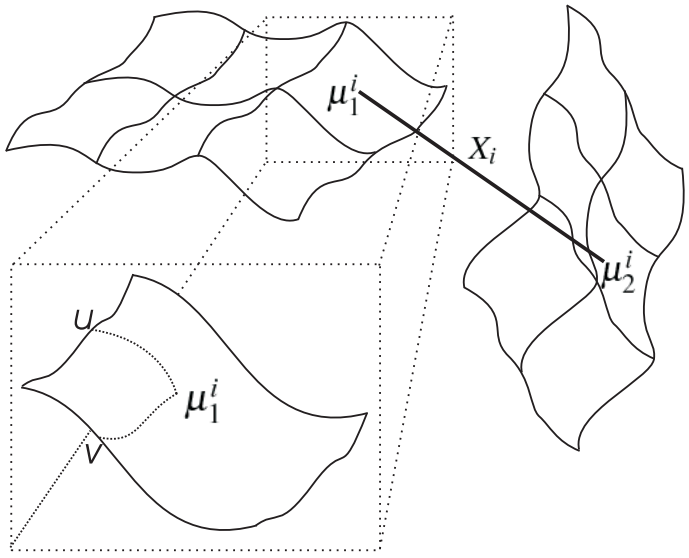


FIG. 4.6 – Représentation d'un pion dans le cas des surfaces paramétriques

4.2.3 Représentation phénotypique

Comme expliqué dans le paragraphe 3.3.2, le phénotype d'un individu correspond à une expression de l'ensemble des caractéristiques exprimées par ces gènes. Dans le cadre de l'algorithme ESPIONS, la fonction de morphogénèse se définit comme suit :

$$\begin{aligned} F: \quad \mathbb{G}_{\mu_1^i} \times \mathbb{G}_{\mu_2^i} &\rightarrow \mathbb{R}^3 \times \mathbb{R}^3 \\ X_i = \{\mu_1^i, \mu_2^i\} &\rightarrow Y_i = \{\eta_1^i, \eta_2^i\} \end{aligned}$$

où Y_i est assimilable à la représentation phénotypique de l'individu X_i, et où $\mathbb{G}_{\mu_1^i}$ et $\mathbb{G}_{\mu_2^i}$ définissent respectivement les espaces paramétriques actuellement explorés par les nucléons μ_1^i et μ_2^i (en fait, $\Omega = \mathbb{G}_{\mu_1^i} \times \mathbb{G}_{\mu_2^i}$). Aussi, η_1^i et η_2^i sont assimilables à deux points de $\mathbb{R}^3$ et ont respectivement pour coordonnées $(\eta_{1x}^i, \eta_{1y}^i, \eta_{1z}^i)$ et $(\eta_{2x}^i, \eta_{2y}^i, \eta_{2z}^i)$, coordonnées exprimées dans le repère absolu de la scène.

Dans ce contexte, la fonction de morphogénèse F est décomposable en deux sous-fonctions f_1 et f_2 ayant toutes deux valeur dans $\mathbb{R}^3$. Ces dernières permettent la détermination des coordonnées euclidiennes des nucléons dans le repère absolu à partir des coordonnées paramétriques définissant leur position au sein des objets sur lesquels ils évoluent (i.e. leur représentation génotypique). Ainsi, on a :

$$Y_i = \left\{ \begin{array}{l} \eta_1^i = f_1(\mu_1^i) \\ \eta_2^i = f_2(\mu_1^i) \end{array} \right\}$$

Dans la suite nous définissons ces fonctions pour chacun des types de géométries abordés précédemment.

Cas polyédrique :

Dans le cas polyédrique, la fonction, notée f_P, se définit comme suit :

$$f_P(\mu_k^i) = H_o^w(\vec{O_{T_{id}}} + \alpha\vec{e_1} + \beta\vec{e_2})$$

où H_o^w correspond à une matrice de transformation homogène donnant la position du polyèdre dans le repère absolu (on parle là de la position du repère relatif de construction de l'objet au sein duquel sont définies les positions des primitives élémentaires qui le composent), $\vec{O_{T_{id}}}$ est un vecteur donnant les coordonnées du sommet du triangle sur lequel se situe le nucléon dans le repère local de l'objet (on parle là du sommet v_0 dans la description du triangle), les vecteurs $\vec{e_1}$ et $\vec{e_2}$ prennent respectivement pour valeur $v_0\vec{v}_1$ et $v_0\vec{v}_2$. Comme précédemment, les coefficients α et β correspondent aux coordonnées du nucléon dans la base ainsi définie. (Voir figure 4.2)

Cas des nuages de points :

Dans le cas des nuages de points, la fonction est notée f_N et se définit comme suit :

$$f_N(\mu_k^i) = H_o^w\vec{O_{P_{id}}}$$

où H_o^w correspond à une matrice de transformation homogène donnant la position du nuage de point dans le repère absolu (on parle là de la position du repère relatif de construction du nuage au sein duquel sont définies les positions des points qui le composent), $\vec{O_{P_{id}}}$ est un vecteur donnant les coordonnées du point du nuage sur lequel se situe le nucléon dans le repère local de l'objet. (Voir figure 4.4)

4.2.4 Evaluation de la qualité

La fonction *fitness* d'un individu X_i se base sur la représentation phénotypique de ce dernier. Aussi, nous définissons la fonction *fitness* d'un individu X_i ayant pour représentation phénotypique Y_i comme suit :

$$\mathscr{F}(Y_i) = (\eta_1^i - \eta_2^i)^2$$

Cette prise en considération de la taille de l'individu (au sens distance euclidienne) paraît en effet être le critère le plus naturel quant à l'évaluation de la qualité de ce dernier. Il est à noter que nous ne considérons pas la taille de l'individu proprement dite, mais le carré de celle-ci. Ce choix se justifie par le coût de la fonction racine carré en termes de temps de calcul dans un contexte informatique. L'information considérée reste cependant pertinente de par la relation : $a < b \Rightarrow a^2 < b^2$ étant donné $\{a,b\} \in \mathbb{R}_+^2$.

Aussi, l'optimisation de la population réalisée par ESPIONS va consister en l'identification de configurations visant à minimiser la *fitness* des individus.

4.3 Opérateurs de variation

Comme il l'a été présenté dans le chapitre dédié aux approches évolutionnaires, l'évolution de la population au sein de son environnement est le résultat de l'application de divers opérateurs de variation ayant pour effet de modifier, de façon plus ou moins importante, le patrimoine génétique des individus qui la composent. Ont également été introduites les notions d'exploration et d'exploitation au travers desquelles nous avons mis en évidence l'existence de plusieurs familles d'opérateurs de mutation ayant des effets différents sur les individus.

Dans cette section, nous présentons les opérateurs de mutations développés dans le cadre de l'algorithme que nous proposons. Leurs effets sont d'abord définis dans un cadre général, sans considérations quant à la nature des objets manipulés, puis dans les cas particuliers de chacun des types géométriques abordés précédemment.

En ce qui concerne les notations, dans la suite de cette section, l'entité $\hat{\mu}_k^i$ correspond au nucléon μ_k^i après repositionnement.

4.3.1 Opérateurs d'exploration

Dans le cadre général des algorithmes évolutionnaires, les opérateurs d'exploration ont pour vocation de provoquer une exploration globale de l'espace combinatoire. Ils ont pour effet une modification importante du patrimoine génétique des individus sur lesquels ils sont appliqués. L'utilisation de tels opérateurs s'avère essentielle dans le cadre de la résolution de problèmes nécessitant l'exploration d'espaces de recherche présentant des minima locaux.

Principes des opérateurs proposés

Dans le cadre de l'algorithme ESPIONS, plusieurs techniques ont été implémentées pour explorer l'espace géométrique au sein duquel la recherche est effectuée. Elles ont pour effet de modifier de façon sensible la position des nucléons composant les pions affectés. Aussi, deux modes opératoires ont été mis en place. Le premier, dit *opérateur de repositionnement aléatoire*, consiste, comme son nom l'indique, en un repositionnement arbitraire de chacun des nucléons constituant le pion affecté en des lieux choisis aléatoirement au sein de l'espace géométrique. Il est à noter que, durant cette opération, les nucléons peuvent effectuer des sauts inter-géométries, la seule contrainte étant de ne pas positionner les nucléons composant un même pion à la surface d'un même objet. Le deuxième type d'opérateurs proposé dans ce contexte consiste en un déplacement des

nucléons dans un large voisinage de leur position courante dans l'espace paramétrique. Ce type d'opérateur, bien que permettant une réelle exploration de l'espace de recherche, s'avère moins différenciant que celui présenté précédemment. Il est en effet conditionné par la configuration initiale de l'individu affecté.

Implémentations en fonction des types de géométries

Ici, nous présentons les implémentations des opérateurs décrits précédemment dans les cas spécifiques des divers types de géométries abordés.

Cas polyédrique :

Opérateur de repositionnement aléatoire

Dans ce contexte, l'opérateur de repositionnement aléatoire introduit précédemment se définit comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} U_I(1, n_G) \\ U_I(1, n_F) \\ U_F(0.0, 1.0) \\ U_F(0.0, 1.0) \end{array} \right\}$$

où n_G correspond au nombre d'objets contenus dans la scène et n_F le nombre de faces contenus dans l'objet à la surface duquel le nucléon est désormais positionné. Rappelons que $U_F([a,b])$ correspond à une variable aléatoire *à valeur réelle* choisie avec une probabilité uniforme sur l'intervalle $[a,b]$, et que $U_I([a,b])$ correspond à une variable aléatoire *à valeur entière* choisie avec une probabilité uniforme sur ce même intervalle.

Une saturation des coordonnées du nucléon au sein de la face nouvellement considérée peut s'avérer nécessaire afin de garantir le positionnement de celui-ci à la surface de l'objet. Pour ce faire, les conditions suivantes sont imposées.

$$\begin{array}{rcl} \alpha < 0.0 & \Rightarrow & \alpha = 0.0 \\ \alpha > 1.0 & \Rightarrow & \alpha = 1.0 \\ \alpha + \beta > 1.0 & \Rightarrow & \alpha = \dfrac{\alpha}{|\alpha + \beta|} \end{array}$$

et *idem* pour β.

Bien entendu, cette expression de l'opérateur sous-entend que l'objet dont l'indice est donné par la première composante est de type polyédrique. Par ailleurs, il est à noter qu'aucune contrainte n'est imposée quant au type de la géométrie d'origine du nucléon (i.e. la géométrie à la surface de laquelle le nucléon était positionné avant application de l'opérateur). Cette caractéristique de l'opérateur de repositionnement aléatoire, que l'on

retrouve pour chacun des types de représentations géométriques proposés, rend possible l'utilisation d'ESPIONS dans un contexte hétérogène. La description de l'architecture logicielle proposée en annexe illustre le mode opératoire mis en œuvre quant à l'implémentation informatique des nucléons en vue de permettre la gestion d'environnements hétérogènes.

Opérateur d'exploration du voisinage

En ce qui concerne le deuxième type d'opérateurs d'exploration, qui consiste à explorer le voisinage des positions courantes des nucléons constituant le pion considéré, l'implémentation proposée dans le cas polyédrique est la suivante :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ V^{n_S}(T_{id}(\mu_k^i)) \\ U_F(0.0, 1.0) \\ U_F(0.0, 1.0) \end{array} \right\}$$

Ici, la fonction notée $V(T_{id})$ renvoie l'indice de l'une des faces adjacente à celle spécifiée en paramètre (i.e. possédant une arête commune). Elle permet de réaliser une succession de sauts (leur nombre est ici exprimé par l'exposant n_S), chaque saut correspondant à un déplacement du nucléon considéré vers l'une des faces voisines à celle sur laquelle il se situe. Il est à noter que, dans le cadre de cet opérateur, le nucléon ne change pas de géométrie. Les deux dernières composantes du génotype doivent être saturées comme présenté précédemment dans le cadre de l'opérateur de repositionnement aléatoire. La figure 4.7 illustre le fonctionnement de l'opérateur d'exploration du voisinage dans le cas polyédrique.

Dans un contexte temps réel, l'utilisation d'un tel opérateur nécessite de calculer les informations inhérentes au voisinage de chacune des faces avant le lancement de la simulation. Aussi, ces informations sont stockées sous la forme de n_F tableaux d'entiers, chaque tableau correspondant à l'une des faces, et contenant les indices des faces présentant au moins un élément commun avec celle-ci (i.e. un sommet ou une arête).

Cas des nuages de points :

Opérateur de repositionnement aléatoire

Dans ce contexte, l'opérateur de repositionnement aléatoire introduit précédemment se définit comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} U_I(1, n_G) \\ U_I(1, n_P) \end{array} \right\}$$

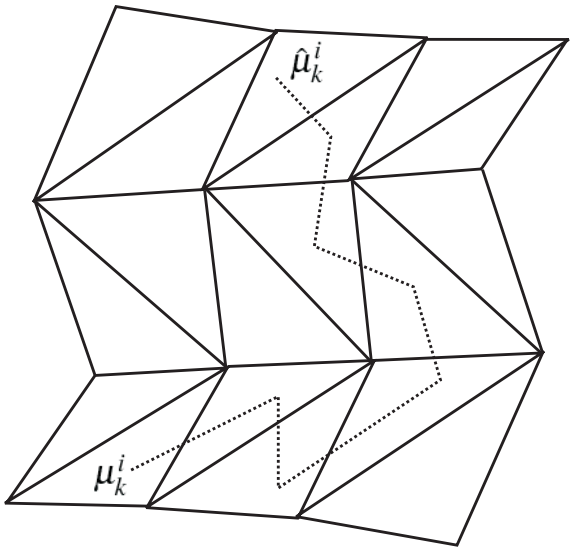


FIG. 4.7 – Opérateur d'exploration du voisinage appliqué au cas polyédrique (avec $n_S = 7$).

où n_G et n_P correspondent respectivement au nombre d'objets contenus dans la scène et au nombre de points contenus dans le nuage au sein duquel le nucléon est désormais positionné. En effet, comme précédemment dans le cas polyédrique, cette expression de l'opérateur sous-entend que la géométrie sur laquelle le nucléon est positionné à l'issu de l'opération est un nuage de points (sans contrainte sur la nature de la géométrie d'origine).

Opérateur d'exploration du voisinage

Dans le cas des nuages de points, l'opérateur d'exploration du voisinage a été implémenté de la façon suivante :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ V_\sigma^{n_S}(P_{id}(\mu_k^i)) \end{array} \right\}$$

Ici, la fonction notée $V(P_{id})$ renvoie l'indice d'un des points présents dans le voisinage de celui spécifié en paramètre. Elle permet de réaliser une succession de sauts (leur nombre est ici exprimé par l'exposant n_S), chaque saut correspondant à un déplacement du nucléon considéré vers l'un des points situés dans le voisinage du point courant. La figure 4.8 illustre cet opérateur.

Il est à noter que, contrairement au cas polyédrique, il n'existe *a priori* pas de relations de connexité entre les points qui constituent le nuage. Aussi, les informations inhérentes au voisinage de chacun des points seront ici basées sur un critère de proximité. La zone contenant les points dits voisins d'un élément donné est assimilable à une sphère ayant

pour centre le point considéré, et pour rayon un seuil de proximité σ. L'opérateur présenté ici est à vocation exploratoire. Aussi, la valeur du seuil de proximité σ doit être relativement élevée afin de permettre des déplacements suffisamment importants.

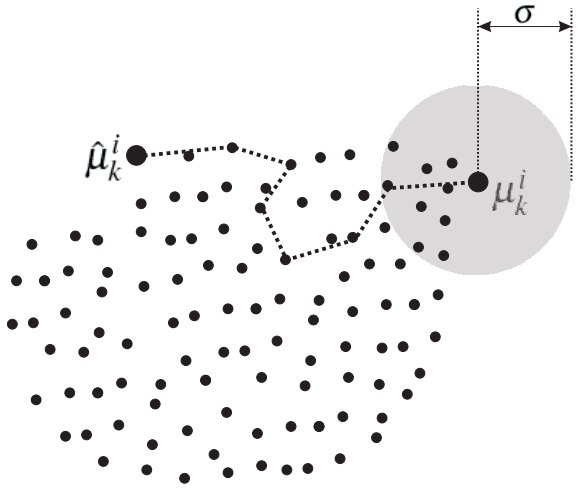


FIG. 4.8 – Opérateur d'exploration du voisinage appliqué au cas des nuages de points (avec $n_S = 7$). La zone de voisinage du point initial est également représentée.

Cas des surfaces paramétriques :

Opérateur de repositionnement aléatoire

Dans ce contexte, l'opérateur de repositionnement aléatoire introduit précédemment se définit comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} U_I(1, n_G) \\ U_I(1, n_C) \\ U_F(0.0, 1.0) \\ U_F(0.0, 1.0) \end{array} \right\}$$

où n_G correspond au nombre d'objets contenus dans la scène et n_C le nombre de carreaux contenus dans l'objet à la surface duquel le nucléon est désormais positionné.

Une saturation des coordonnées du nucléon au sein du carreau nouvellement considéré peut s'avérer nécessaire afin de garantir le positionnement de celui-ci à la surface de l'objet. Cette saturation dépend alors du formalisme utilisé quant au bornage des paramètres u et v.

Bien entendu, comme précédemment, cette expression de l'opérateur sous-entend que l'objet dont l'indice est donné par la première composante est une surface paramétrique.

Opérateur d'exploration du voisinage

Dans le cadre des surfaces paramétriques, l'opérateur d'exploration du voisinage se définit comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ C_{id}(\mu_k^i) \\ u(\mu_k^i) + \sigma_u \, U_F(-1.0, 1.0) \\ v(\mu_k^i) + \sigma_v \, U_F(-1.0, 1.0) \end{array} \right\}$$

On retrouve là un formalisme très semblable à celui introduit dans le chapitre 3 (Cf. section 3.4.2) dans le cadre de la définition des opérateurs de mutations anisotropes. Aussi, dans la pratique, les valeurs des coefficients σ_u et σ_v seront élevées afin de garantir le caractère exploratoire de l'opérateur. Encore une fois, les coordonnées nouvellement obtenues devront être saturées afin de garantir l'appartenance du nucléon au carreau considéré. La figure 4.9 illustre le fonctionnement de l'opérateur d'exploration du voisinage dans le cas des surfaces paramétriques.

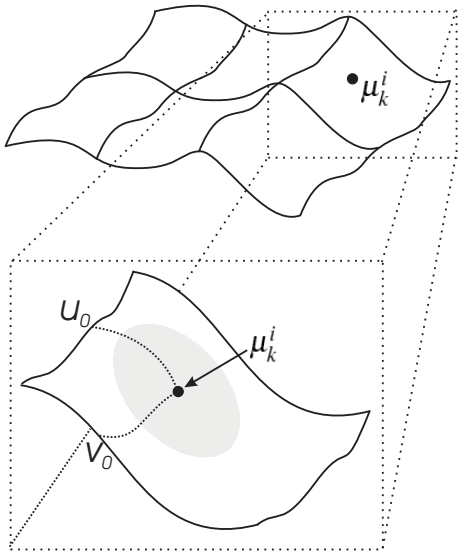


FIG. 4.9 – Opérateur d'exploration du voisinage appliqué au cas des surfaces paramétriques. La zone grisée contient l'ensemble des positions atteignables durant la mutation.

4.3.2 Opérateurs d'exploitation

Dans le cadre général des algorithmes évolutionnaires tels qu'ils sont présentés dans le chapitre 3, les opérateurs dits d'exploitation ont pour fonction de permettre un affinage des individus présentant des propriétés intéressantes. Aussi, un opérateur de ce type va généralement consister en une exploration locale de l'espace combinatoire dans le proche

voisinage de l'individu sur lequel il est appliqué. L'utilisation de tels opérateurs permet d'accélérer la convergence de l'algorithme.

Principes des opérateurs proposés

Dans le cadre de l'algorithme ESPIONS, plusieurs techniques ont été implémentées quant à l'exploitation des individus présentant de bonnes caractéristiques (i.e. individus ayant une bonne *fitness*). La plupart des opérateurs que nous proposons dans ce contexte consiste en une exploration du voisinage des nucléons constituant les individus exploités. Le but de ces opérations étant d'optimiser localement les individus concernés, une approche élitiste est mise en œuvre afin de garantir l'amélioration des propriétés des éléments exploités (Cf. paragraphe 3.4.1). Pour ce faire, la configuration d'un pion après exploitation n'est conservée que si la *fitness* de celui-ci s'est améliorée au cours de la mutation. Dans le cas contraire, l'individu est repositionné dans sa configuration initiale.

Un opérateur déterministe visant à identifier la configuration locale optimale est également proposé dans le cas particulier des modèles polyédriques.

Implémentations en fonction des types de géométries

Ici, nous présentons les implémentations de ces opérateurs d'exploitation pour les différents types de géométries abordés précédemment.

Cas polyédrique :

Dans le cas polyédrique, trois modes opératoires ont été proposés quant à l'exploitation des pions.

Opérateur d'exploitation du voisinage

Ce premier opérateur consiste en un déplacement des nucléons constituant le pion exploité dans le proche voisinage de leur position courante. Il se définit comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ V^1(T_{id}(\mu_k^i)) \\ \alpha(\mu_k^i) + \sigma\, U_F(-1.0, 1.0) \\ \beta(\mu_k^i) + \sigma\, U_F(-1.0, 1.0) \end{array} \right\}$$

L'opérateur considéré ayant pour seule vocation d'affiner l'individu exploité, la valeur du coefficient σ doit être suffisamment faible, afin de limiter le déplacement des nucléons. On retrouve là la fonction V introduite précédemment dans le cadre des opérations d'exploration du voisinage. Cependant, le nombre de sauts effectués est ici réduit à 1 afin de

ne pas trop s'éloigner de la configuration courante de l'individu que l'on vise à exploiter. Cette capacité à sauter sur une face voisine a pour vocation de ne pas *emprisonner* le nucléon considéré à l'intérieur d'une face, lui permettant ainsi d'évoluer sans restriction à la surface de l'objet qui le supporte.

Par ailleurs, on peut remarquer que, bien qu'il soit défini comme un *opérateur de mutation isotrope* (même σ pour α et β. Cf. section 3.4.2), dans l'immense majorité des cas l'opérateur proposé est géométriquement assimilable à un *opérateur de mutation anisotrope*, de par le fait que la base formée par les arêtes du triangle n'est pas orthonormée. La figure 4.10 illustre cet opérateur. Il est à noter que, comme précédemment, les coefficients α et β sont saturés de manière à garantir le positionnement du nucléon à la surface de l'objet à l'issue de la mutation.

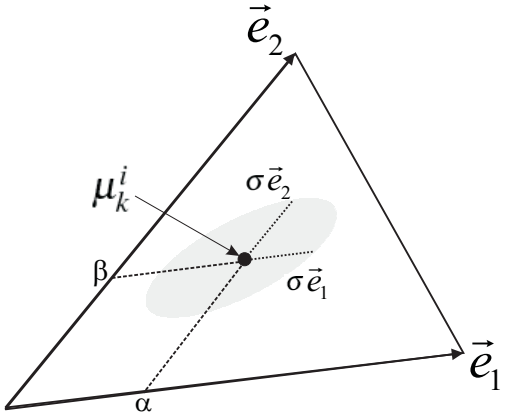


FIG. 4.10 – Opérateur d'exploitation du voisinage appliqué au cas des polyèdres. La zone grisée contient l'ensemble des positions atteignables durant la mutation.

Opérateur d'exploitation stochastique

Ce deuxième opérateur consiste en un repositionnement aléatoire des nucléons constituant le pion exploité au sein des faces sur lesquelles ils évoluent et de leur voisinage direct :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ V^1(T_{id}(\mu_k^i)) \\ U_F(0.0,1.0) \\ U_F(0.0,1.0) \end{array} \right\}$$

La configuration de l'individu après mutation n'est maintenue que si la *fitness* de ce dernier s'est vu améliorée par l'opération. Dans le cas contraire, le pion retrouve sa position initiale.

Opérateur d'exploitation déterministe

Ce dernier opérateur consiste en une optimisation locale de l'individu. Pour ce faire, les nucléons sont positionnés de manière à ce que le pion qu'ils constituent matérialise le segment minimum séparant les triangles sur lesquels ils évoluent.

Une variante de cet opérateur consiste en une application de l'optimisation en prenant en considération, non pas seulement la paire constituée par les triangles supportant les nucléons, mais également les paires formées par les triangles voisins. Comme on pourra l'observer dans la suite, cette variante permet une accélération sensible quant à la convergence de l'algorithme.

La figure 4.11 illustre les effets de l'opérateur d'exploitation déterministe avec et sans considération du voisinage.

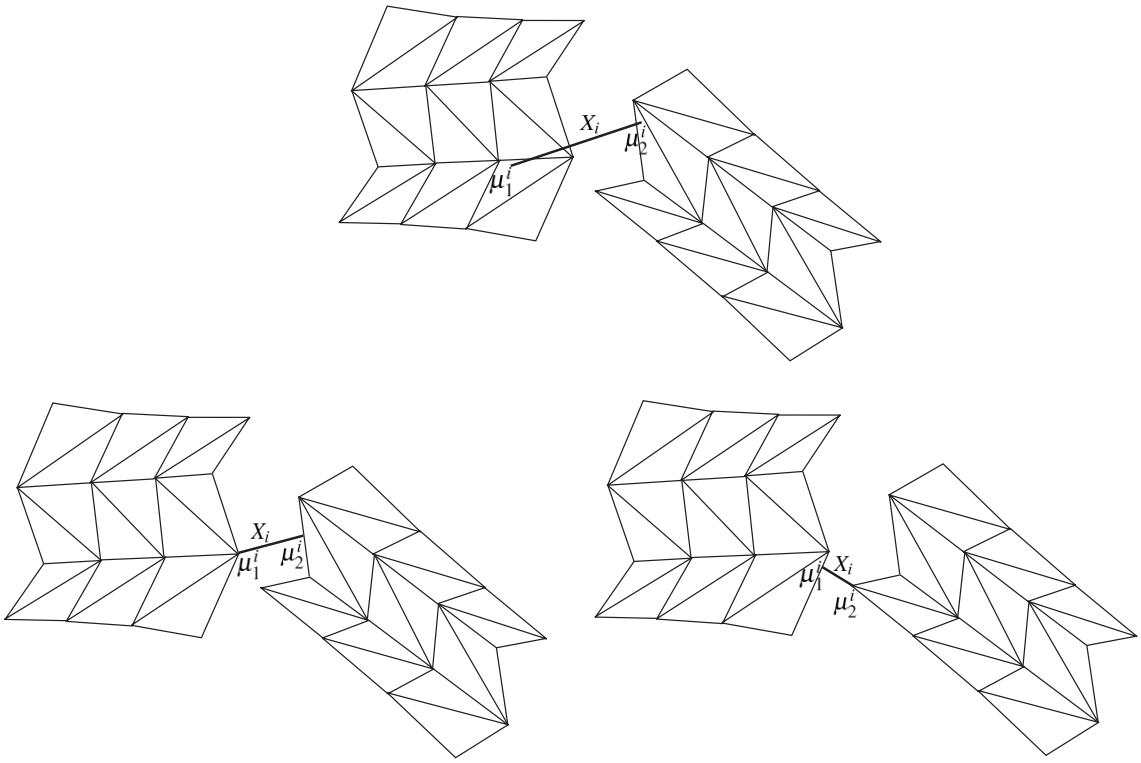


FIG. 4.11 – Opérateur d'exploitation déterministe. Position initiale de l'individu (en haut), position après application de l'opérateur sans considération des paires de triangles voisines (à gauche) et avec considération des paires de triangles voisines (à droite).

Cas des nuages de points :

L'opérateur d'exploitation mis en œuvre dans le cas des nuages de points est basé sur le même principe que l'opérateur d'exploration du voisinage proposé précédemment

pour ce même type de géométries. Comme ce dernier, il consiste en une exploration du voisinage des points occupés par les nucléons. Nous rappelons ici sa définition :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ V_\sigma^{n_S}(P_{id}(\mu_k^i)) \end{array} \right\}$$

Afin de permettre une exploitation des individus traités, deux changements sont apportés à cet opérateur initialement présenté comme exploratoire. Le premier consiste en une minimisation de la valeur du coefficient σ qui permet, rappelons-le, le dimensionnement des zones contenant les entités dites voisines pour chacun des points du nuage. Le deuxième changement consiste en l'adoption d'une approche élitiste semblable à celle présentée pour les opérateurs précédents.

Cas des surfaces paramétriques :

Comme dans le cas des nuages de points, la mise en place d'un opérateur d'exploitation adapté aux surfaces paramétriques va consister en une adaptation de l'opérateur d'exploration précédemment introduit pour ce type de géométries. Celui-ci est défini comme suit :

$$\hat{\mu}_k^i = \left\{ \begin{array}{c} G_{id}(\mu_k^i) \\ C_{id}(\mu_k^i) \\ u(\mu_k^i) + \sigma_u \, U_F(-1.0, 1.0) \\ v(\mu_k^i) + \sigma_v \, U_F(-1.0, 1.0) \end{array} \right\}$$

Afin de permettre une exploitation des individus traités, les valeurs des paramètres σ_u et σ_v sont réduites en comparaison avec celles utilisées dans le cadre de la variante exploratoire de l'opérateur. Comme précédemment, une approche élitiste est adoptée afin de garantir l'amélioration de la *fitness* des individus lors de mutations de cette nature.

4.3.3 Opérateurs de croisement

Comme il l'a été expliqué précédemment dans le chapitre consacré aux approches évolutionnaires, les opérateurs de croisement consistent en la combinaison des propriétés de plusieurs individus dits parents afin d'obtenir de nouveaux individus dits enfants. Les parents sont généralement choisis parmi les bons individus de la population. Aussi, en héritant d'une partie des caractéristiques *a priori* intéressantes de leurs géniteurs, les nouveaux individus vont contribuer à l'accélération de la convergence de l'algorithme.

Dans l'algorithme que nous proposons, l'opérateur de croisement est assimilable à une application de Ω^2 dans Ω (i.e. deux parents donnent un enfant). Il se définit comme suit :

$$X_k = X_i \otimes X_j \Rightarrow X_k = \{\mu_1^i, \mu_2^j\}$$

Ainsi, il consiste à attribuer à l'individu enfant le premier nucléon du premier parent et le deuxième nucléon du deuxième parent.

Afin de contribuer encore d'avantages à l'accélération de la convergence de l'algorithme, nous avons choisi de retenir la configuration la plus intéressante parmi les deux combinaisons réalisables à partir des individus parents. Aussi, la formulation finale de l'opérateur de croisement, tel qu'il est implémenté dans le cadre d'ESPIONS, est la suivante :

$$X_k = X_i \otimes X_j \Rightarrow X_k = Min_{\mathscr{F}}(\{\mu_1^i, \mu_2^j\}, \{\mu_1^j, \mu_2^i\})$$

où la fonction $Min_{\mathscr{F}}$ permet l'identification de la combinaison présentant la meilleure *fitness*. La figure 4.12 illustre le fonctionnement de cet opérateur.

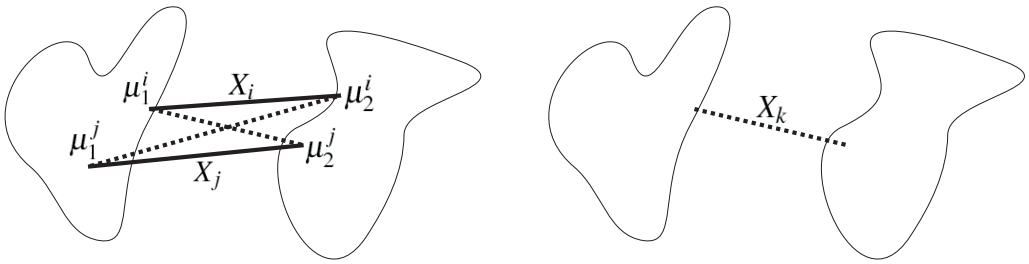


FIG. 4.12 – Opérateur de croisement. Dans le cas présenté ici, on a $X_k = \{\mu_1^i, \mu_2^j\}$.

Il semble intéressant de noter que dans cette définition de l'opérateur de croisement, les individus pris pour parents ne doivent pas nécessairement relier la même paire d'objets. La seule contrainte quant à la mise en œuvre de cet opérateur réside dans le fait qu'il n'est pas possible que l'individu enfant soit composé de deux nucléons évoluant à la surface d'un même objet (contrainte issue de la définition même d'un individu dans le cadre d'ESPIONS. Cf. paragraphe 4.2.1).

4.4 Maintien de la diversité génétique

Dans le chapitre dédié à la présentation des principes généraux inhérents aux approches évolutionnaires, nous avons introduit la notion de maintien de la diversité génétique (Cf. section 3.6.2). Dans ce contexte ont été présentées plusieurs techniques visant à préserver une bonne répartition des individus au sein de l'espace à explorer. Parmi celles-ci, intéressons-nous à l'approche dite de *partage*, ou *fitness sharing*. Cette dernière consiste en une modification de la fonction *fitness* par la prise en compte de la densité d'individus présents dans le voisinage de l'élément à évaluer. Dans ce contexte, la fonc-

tion *fitness* d'un individu prend la forme suivante :

$$\mathscr{F}'(X_i) = \frac{\mathscr{F}(X_i)}{\sum_{j=0, j\neq i}^{p} sh(X_i, X_j)}$$

avec

$$sh(X_i, X_j) = \begin{cases} \left[\frac{d(X_i,X_j)}{\sigma_{share}}\right]^{\alpha} & si\ d(X_i, X_j) < \sigma_{share} \\ 0 & sinon \end{cases}$$

où σ_{share} est assimilable à un seuil de ressemblance au delà duquel deux individus sont considérés comme éloignés (on parle de rayon de partage), α est un réel permettant d'ajuster l'effet du *sharing* sur la fonction *fitness*, et d est une *fonction distance* permettant d'évaluer le niveau de ressemblance de deux individus (fonction dépendante de type de représentation).

4.4.1 Le *partage* appliqué à ESPIONS

Dans la formulation de la fonction *fitness* présenté ci-dessus, il s'avère nécessaire de mettre en place une *fonction distance* permettant l'évaluation du niveau de ressemblance de deux individus. Cette fonction doit ensuite se voir appliquée sur l'ensemble de la combinatoire formée par les individus, soit $p-1$ fois pour chaque évaluation de la qualité d'un unique élément. L'implémentation d'un tel processus ne s'avère nullement problématique dans le cadre de la résolution de problèmes n'imposant pas de contraintes temporelles importantes quant à l'identification des solutions pertinentes. Cependant, dans le cadre de la problématique qui nous intéresse ici, des contraintes de cette nature sont à respecter afin de permettre l'utilisation de l'algorithme dans un contexte temps réel, rendant possible une manipulation interactive des objets composant la scène par l'utilisateur. Dans la suite, nous présentons les modes opératoires que nous avons mis en œuvre afin de préserver la diversité génétique sans porter préjudice aux performances de l'algorithme.

4.4.2 Carte de densité

Afin d'obtenir une information de densité sans pénaliser les performances de l'algorithme, nous proposons de définir une *carte de densité* contenant les informations nécessaires à l'évaluation de la densité d'individus présents dans chaque région de l'espace. Cette approche sous-entend une décomposition de l'espace d'exploration (i.e. les objets simulés) en régions. Une telle décomposition des objets peut se faire selon différentes techniques. Dans cette partie, nous ne traitons pas de cette problématique. Retenons cependant que, dans le cadre des expériences proposées dans la suite du mémoire, cette

segmentation est réalisée par décomposition des objets en hiérarchies de volumes englobants (sphériques) ; Les régions correspondant aux éléments de surface contenus dans les volumes englobants dont la taille est inférieure à un seuil donné. La procédure permettant l'identification de ces volumes au sein de la hiérarchie est présentée dans l'algorithme 1. Une illustration d'une telle décomposition est proposée avec la figure 4.13.

Il est à noter que le choix de ce mode de décomposition fût simplement motivé par des raisons d'ordre pratique (utilisation de librairies préalablement développées au sein du laboratoire) et que d'autres modes opératoires auraient pu être choisis. Il serait par exemple possible d'utiliser d'autres types de volumes englobants (OBB, k-dops, ...) ou encore de procéder à une décomposition des objets en éléments convexes. Ce dernier type de décomposition mériterait d'ailleurs d'être exploré dans ce contexte. En effet, les propriétés de convexité des zones ainsi obtenues pourraient probablement se voir exploitées par ailleurs, notamment pour le paramétrage de l'algorithme. Rappelons cependant que ce type de décomposition est susceptible d'engendrer un nombre important d'éléments (cas du tore par exemple).

Algorithme 1 (Procédure d'extraction des régions sur une hiérarchie de sphères englobantes)

ExtractRegionsFromNode *(node, threshold, regionsList)*
(1) si *(node* $\rightarrow$ *radius* $<$ *threshold* ***ou*** *node* $\rightarrow$ *children* $= 0$*)*
    regionsList $\leftarrow$ *node*
    terminer
(2) sinon
    ExtractRegionsFromNode *(node* $\rightarrow$ *childRight, threshold, regionsList)*
    ExtractRegionsFromNode *(node* $\rightarrow$ *childLeft, threshold, regionsList)*
    terminer

ExtractBSTreeRegions *(tree, threshold, regionsList)*
(1) Empty*(regionsList)*
(2) ExtractRegionsFromNode *(tree* $\rightarrow$ *rootNode, threshold, regionsList)*

La fonction *ExtractBSTreeRegions* permet l'extraction à partir de la hiérarchie *tree* des indices des nœuds dont les sphères englobantes présentent un rayon inférieur à *threshold*. Pour cela, elle fait appel à une fonction récursive *ExtractRegionsFromNode* permettant une exploration en profondeur de la hiérarchie considérée. Bien entendu, seules les sphères de plus haut niveau répondant à ce critère sont retenues. Nous entendons par là que les sphères contenues dans celles-ci ne sont pas ajoutées à la liste.

Ainsi, une carte de densité donne le taux d'occupation de chaque paire de régions pour un couple d'objets donné. Elle est assimilable à une matrice $n_{R_1} \times n_{R_2}$, n_{R_1} et n_{R_2} corres-

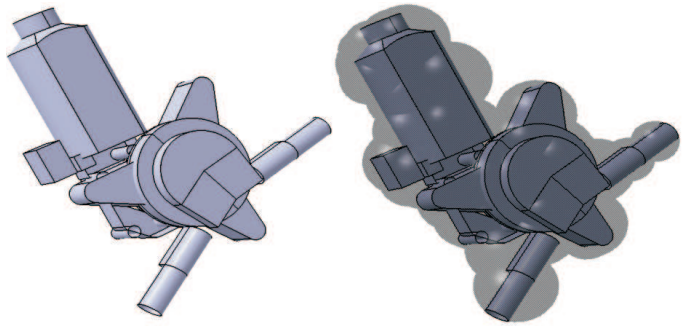


FIG. 4.13 – Exemple de décomposition en régions à partir d'une hiérarchie de sphères englobantes : A gauche l'objet considéré nu, à droite le même objet décomposé (rayon seuil = 25mm).

pondant respectivement au nombre de régions composant chacun des objets du couple considéré. Chaque élément de la matrice donne le nombre d'individus reliant les régions correspondantes. Dans la suite, nous noterons $\rho_i^j(r_1, r_2)$ le nombre d'individus reliant les régions r_1 et r_2 appartenant respectivement aux objets i et j. La figure 4.14 donne une illustration d'une carte de densité pour une paire d'objets tous deux décomposés en une vingtaine de régions. On peut y observer une grande hétérogénéité quant à la fréquentation de l'espace, les paires de zones les plus fréquentées (i.e. les pics), correspondant à des minima locaux.

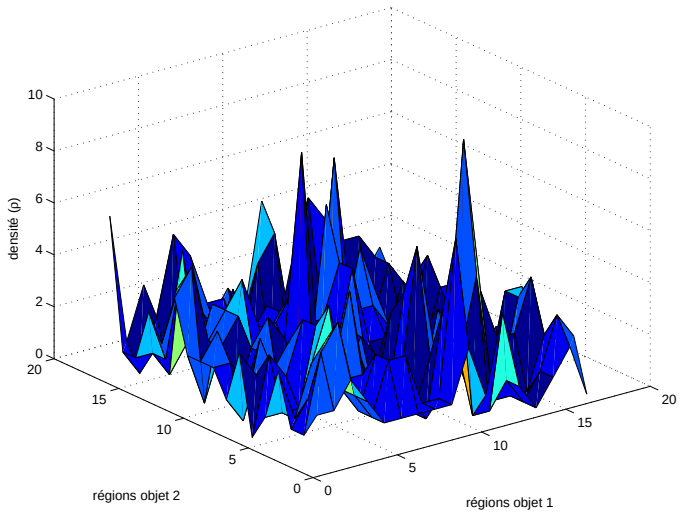


FIG. 4.14 – Illustration d'une carte de densité prenant en considération un couple d'objets tous deux décomposés en une vingtaine de régions.

Mise à jour de la carte de densité

Les variations de la carte de densité sont issues des déplacements des individus, ou plutôt des nucléons qui les composent, au sein de l'espace d'exploration. Aussi, le maintien à jour de la carte de densité va simplement consister à effectuer, pour chaque mutation d'un individu, une incrémentation de l'élément de la matrice correspondant à la paire de régions nouvellement fréquentée, et une décrémentation de l'élément de la matrice correspondant à la paire de régions initialement fréquentée par l'individu considéré. Ce mode opératoire nécessite un marquage des primitives géométriques constituant les objets afin d'être en mesure de déterminer l'appartenance d'un nucléon à telle ou telle région. La figure 4.15 illustre ce marquage dans le cas polyédrique.

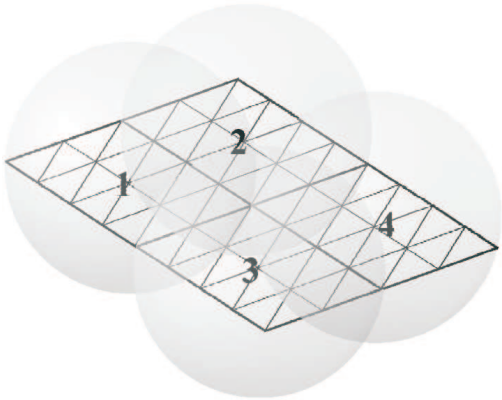


FIG. 4.15 – Exemple de marquage des primitives composant un objet (ici un polyèdre) en fonction de leur région d'appartenance.

Il est à noter que cette procédure n'engendre que peu de calcul, et que l'occupation mémoire nécessaire au stockage de la matrice reste raisonnable dès lors que les régions ne s'avèrent pas sous-dimensionnées.

Validité de l'approche dans le cas déformable : discussion

La vocation de la carte de densité, telle qu'elle est introduite dans ce contexte, est de rendre possible une mesure de la répartition des pions au sein de l'espace de recherche afin de permettre la mise en oeuvre d'un processus de maintien de la diversité. L'utilisation de telles cartes n'est pas seulement possible dans le cas rigide. En effet, il est tout à fait envisageable de fonctionner de la même manière dans le cadre de simulation mettant en scène des objets déformables. Dans l'hypothèse où les déformations des objets s'avèrent élastiques et d'amplitude relativement faible, il n'est pas nécessaire de mettre à

jour le marquage des primitives réalisé en phase d'initialisation, de par le fait que, dans ce contexte, les éléments géométriques proches (on parle des primitives composant les objets) restent proches tout au long de la manipulation. Dans le cadre de déformations persistantes et de grandes amplitudes, il peut s'avérer nécessaire de mettre à jour le marquage des primitives. Ce processus n'a pas été mis en œuvre dans le cadre de cette thèse. On peut cependant définir les étapes nécessaires à cette mise à jour : 1 - réaliser un nouveau marquage des primitives, 2 - mettre à jour la carte de densité en fonction de ce nouveau marquage. Ce processus pourrait s'avérer coûteux en termes de temps. Une étude plus poussée devra certainement être menée dans ce contexte.

4.4.3 Nouvelle expression de la fonction *fitness*

Afin de contribuer au maintien de la diversité génétique au sein de la population, nous proposons donc d'utiliser la nouvelle expression de la fonction *fitness* suivante :

$$\mathscr{F}'(X_i) = \mathscr{F}(X_i) \cdot \rho_{o_1}^{o_2}(r_1, r_2)^{\gamma}$$

où o_1 et o_2 correspondent aux indices des objets respectivement fréquentés par chacun des nucléons composant l'individu, et r_1 et r_2 les indices des régions fréquentées par ces mêmes entités. Comme α dans la formulation classique, γ est un paramètre réel permettant d'ajuster l'effet du *partage* sur la fonction *fitness*.

4.5 Schémas de fonctionnement

A ce niveau de la présentation d'ESPIONS nous avons introduit plusieurs formalismes permettant l'utilisation de l'algorithme dans le cadre de simulations faisant intervenir divers types de géométries, et un ensemble d'opérateurs destinés à faire évoluer les individus au sein de l'espace géométrique que constituent les éléments de la scène. Dans cette section, nous décrivons, dans un premier temps, le schéma de fonctionnement interne de l'algorithme et les différents processus qui le composent. Puis, nous présentons les architectures proposées quant à l'intégration d'ESPIONS au sein d'un processus global de détection de collisions.

4.5.1 Schéma de fonctionnement classique

Dans le chapitre consacré à la présentation des approches basées sur l'évolution artificielle, nous avons introduit le schéma de fonctionnement classique de ce type d'algorithmes tel qu'il est présenté dans la littérature. La figure 4.16 illustre ce processus.

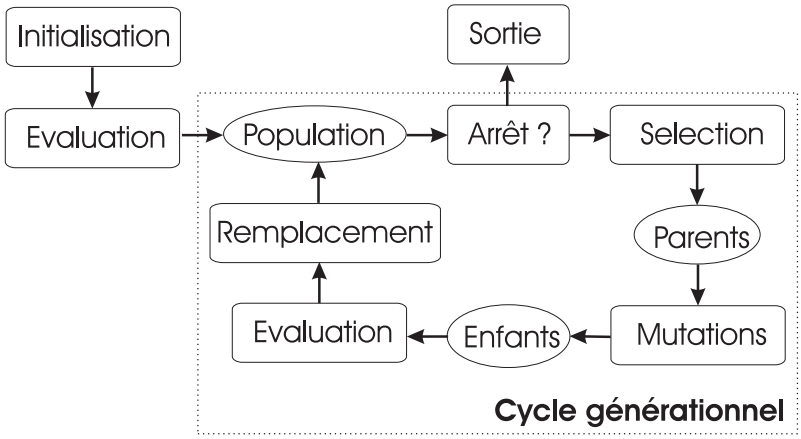


FIG. 4.16 – Schéma de fonctionnement classique d'un algorithme évolutionnaire

Ce schéma se décompose en deux phase. Une phase d'initialisation, durant laquelle la population initiale Π_0 est créée, et une phase d'évolution, elle-même décomposable en quatre étapes que sont : la sélection des meilleurs individus de la population, l'évolution de la population par application des opérateurs de mutation, l'évaluation de la qualité des individus nouvellement obtenus, et enfin le remplacement par les bons individus obtenus des éléments s'avérant de plus mauvaise qualité. Dans ce schéma, l'arrêt de l'algorithme est conditionné par un *critère d'arrêt* pouvant être de différentes natures : niveau de qualité du meilleur individu, stagnation de la *fitness*, nombre de générations maximum.

4.5.2 Fonctionnement d'ESPIONS : une évolution perpétuelle

Le schéma de fonctionnement d'ESPIONS diffère du schéma classique présenté précédemment, et cela à plusieurs niveaux. Sur la figure 4.17, nous proposons une illustration du fonctionnement d'ESPIONS.

Comme dans le schéma classique, on peut distinguer une phase d'initialisation et une phase d'évolution. Dans la suite, nous proposons de décrire en détail chacune des étapes de l'algorithme.

Initialisation

Durant la phase d'initialisation, l'ensemble des individus constituant la population est créé. Le nombre d'individus est l'un des paramètres d'entrée de l'algorithme. Noté p, il n'évolue pas en cours de simulation.

En ce qui concerne le positionnement initial des pions, ou plutôt des nucléons qui les composent, chaque individu se voit appliquer un opérateur de repositionnement aléatoire

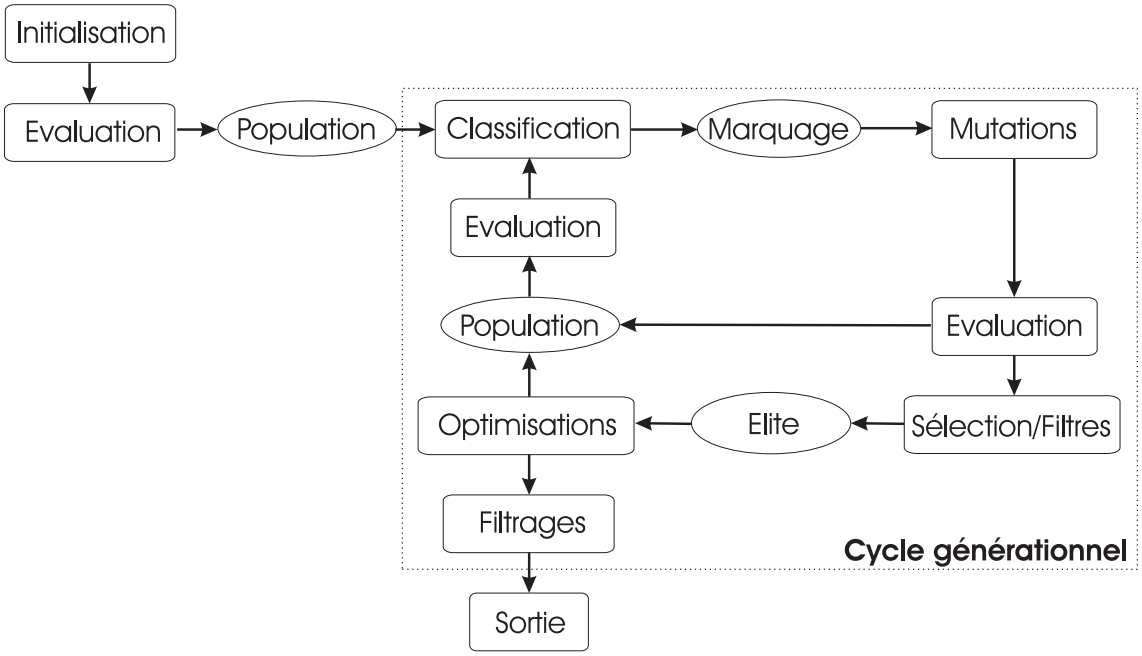


FIG. 4.17 – Schéma de fonctionnement d'ESPIONS

après sa création. Ce mode opératoire permet une répartition uniforme de la population au sein de l'espace géométrique à explorer.

Cycle générationnel

Nous avons vu précédemment que, dans le cadre du fonctionnement classique d'un algorithme évolutionnaire, le cycle générationnel se décompose en plusieurs phases. Sans réitérer l'explication proposée dans 3.2.2, on peut résumer le mode opératoire adopté dans ce contexte de la façon suivante. Dans un premier temps, une partie de la population est isolée via un processus de sélection visant à identifier les meilleurs individus (i.e. les parents). Puis, les individus constituant cette sous-population se voient appliquer des opérateurs de mutations. Enfin, après évaluation de leur qualité, les individus nouvellement obtenus (i.e. les enfants) sont introduits au sein de la population initiale, venant éventuellement remplacer des individus présentant de mauvaises caractéristiques.

Dans le cadre de l'algorithme que nous proposons, le cycle générationnel se décompose de la manière suivante :

1. **Classification :**
   Comme son nom l'indique, cette phase est assimilable à une classification (ou marquage) des individus selon des critères qualitatifs. Pour ce faire, un tournoi est organisé au sein de la population. Ce tournoi consiste en une succession de combats

opposant les individus deux à deux. A l'issue de chaque combat, le pion gagnant (i.e. celui qui a la meilleure *fitness*) se voit attribuer un statut d'individu vainqueur. Celui-ci sera exploité lors de la phase de mutations. Quant au pion perdant, un type d'opération lui est attribué. Celui-ci correspond à l'opérateur, *a priori* exploratoire, qui lui sera appliqué durant la phase de mutations. L'individu perdant pourra également se voir remplacé par le résultat du croisement de deux individus présentant des propriétés intéressantes. L'attribution de l'un ou l'autre des opérateurs d'exploration à l'individu perdant est conditionnée par un jeu de trois paramètres correspondant aux pourcentages d'application de chacun des opérateurs. Aussi, dans la suite du mémoire, nous noterons $\%_E$ le pourcentage d'individus perdants devant subir un opérateur d'exploration du voisinage, $\%_R$ le pourcentage d'individus perdants devant subir un opérateur de repositionnement aléatoire et $\%_X$ le pourcentage d'individus perdants devant être remplacés par le résultat d'un croisement.

La façon la plus intuitive pour implémenter une telle procédure consiste en une succession de tirages au sort permettant d'obtenir des couples d'individus à confronter, tel que présenté dans l'algorithme 2.

Algorithme 2 (Procédure de classification des individus : tournoi stochastique)

(I) $currentTask \leftarrow ExplorMut$
$NbExplorMut \leftarrow p \times \%_E \div 100$
$NbCrossMut \leftarrow p \times \%_X \div 100$
$NbRandMut \leftarrow p \times \%_R \div 100$

(II) ***pour*** $(k \leftarrow 0, k < p)$

(1) ***si*** $(k = NbExplorMut)$
$currentTask \leftarrow CrossMut$

(2) ***si*** $(k = NbExplorMut + NbCrossMut)$
$currentTask \leftarrow RandMut$

(3) ***si*** $(k = NbExplorMut + NbCrossMut + NbRandMut)$
terminer

(4) $i \leftarrow U_I(0, p-1)$

(5) $j \leftarrow U_I(0, p-1)$

(6) ***si*** $(\mathscr{F}(X_i) < \mathscr{F}(X_j))$
$(X_i \rightarrow task) \leftarrow ExploitMut$
$(X_j \rightarrow task) \leftarrow currentTask$
sinon
$(X_i \rightarrow task) \leftarrow currentTask$

$(X_j \rightarrow task) \leftarrow ExploitMut$

Cependant, afin de garantir la considération de l'ensemble des individus et ainsi contribuer à une meilleure dynamique de l'algorithme (i.e. plus d'entités se voient évoluer), nous avons implémenté le processus de classification tel que présenté dans l'algorithme 3. Cette procédure présente également l'avantage de n'effectuer qu'une génération de valeur aléatoire, contrairement à celle présentée précédemment. Or, ce type de génération s'avère coûteux en termes de temps de calcul (au sens informatique). C'est pourquoi nous parlons là d'une procédure de classification optimisée.

Algorithme 3 (Procédure de classification des individus optimisée)

(I) $start \leftarrow U_I(0, p-1)$
$currentTask \leftarrow ExplorMut$
$NbExplorMut \leftarrow p \times \%_E \div 100$
$NbCrossMut \leftarrow p \times \%_X \div 100$
$NbRandMut \leftarrow p \times \%_R \div 100$

(II) ***pour*** $(k \leftarrow 0, k < p)$

(1) ***si*** $(k = NbExplorMut)$
$currentTask \leftarrow CrossMut$

(2) ***si*** $(k = NbExplorMut + NbCrossMut)$
$currentTask \leftarrow RandMut$

(3) ***si*** $(k = NbExplorMut + NbCrossMut + NbRandMut)$
terminer

(4) $i \leftarrow start$

(5) $j \leftarrow start + p - (2k+1)$
si $(j \geq p)$
$j = j - p$

(6) ***si*** $(\mathscr{F}(X_i) < \mathscr{F}(X_j))$
$(X_i \rightarrow task) \leftarrow ExploitMut$
$(X_j \rightarrow task) \leftarrow currentTask$
sinon
$(X_i \rightarrow task) \leftarrow currentTask$
$(X_j \rightarrow task) \leftarrow ExploitMut$

(7) $start = start + 1)$

si *(start = p)*
start = 0

Cette phase de *classification* se distingue ainsi clairement de la *sélection* opérée dans le cadre des algorithmes classiques et dont le rôle consiste en l'extraction d'une sous-population d'individus.

2. **Mutations :**
   Comme dit précédemment, durant la phase de classification chacun des individus s'est vu attribuer un type d'opérateur. Aussi, la phase de mutations va simplement consister en l'application des opérateurs de mutation sur les individus auxquels ils ont respectivement été attribués.

3. **Evaluation :**
   Dans le schéma proposé ici, on trouve deux phases d'évaluation. La première, réalisée à l'issue du processus de mutations, permet une prise en compte des variations subies par les individus lors de l'application des opérateurs génétiques.

   La deuxième phase d'évaluation présentée sur ce schéma est réalisée juste avant l'étape de classification précédemment décrite. Cette réévaluation de la qualité des individus se justifie par le fait que l'algorithme ESPIONS a pour vocation de fonctionner dans le cadre de la simulation de scènes en mouvement. Aussi, au delà des déplacements induits par l'application des divers opérateurs de mutation, la position (au sens phénotypique) des nucléons dans l'espace s'avère également dépendante de la position des objets qui les supportent. On touche là à l'une des originalités de la problématique abordée. En effet, nous cherchons ici à optimiser un problème dont la solution ne cesse d'évoluer dans le temps, d'où le terme d'*évolution perpétuelle*. Ce phénomène induit une prise en considération implicite de la cohérence spatio-temporelle dont nous discutons dans la suite de ce chapitre (Cf. paragraphe 4.6). Il est cependant à noter que, durant cette phase, seuls les pions dont au moins l'un des nucléons appartient à la surface d'un objet s'étant déplacé sont réévalués ; Ceci pour des raisons évidentes d'économie de temps, l'évaluation d'un pion ayant un coup.

4. **Sélection/Filtres :**
   Cette étape a pour but d'extraire de la population un sous-ensemble d'individus présentant des caractéristiques intéressantes. Contrairement à un processus de sélection classique, généralement fondé sur des seuls critères qualitatifs, d'autres paramètres sont ici pris en compte. Aussi, dans un premier temps, une étape préliminaire de sélection va consister en l'identification des pions dont la taille est inférieure à un

seuil spécifié par l'utilisateur, noté ζ dans la suite de l'exposé. Puis, diverses étapes de filtrages vont permettre d'exclure de la sous-population les individus n'apportant pas de valeur ajoutée quant à la résolution du problème considéré. Un premier niveau de filtrage exclut les individus doublons, soit tous les individus ayant la même *fitness* sauf un. Ce filtre sous-entend que deux individus présentant la même *fitness* sont nécessairement superposés. Nous justifions cette affirmation par le fait que la similarité entre deux pions reliant des éléments géométriques distincts correspond à une situation numériquement dégénérée. Dans le cadre de simulations mettant en scène des objets polyédriques, un deuxième niveau de filtrage élimine les doublons par paires de triangles. Ainsi, pour chaque paire de triangles occupée, seul un individu est conservé. Ces exclusions sont justifiées par le fait que, lors de l'étape ultérieure d'*optimisation* des pions constituant cette *élite* que nous cherchons ici à isoler, l'opérateur d'exploitation déterministe présenté précédemment dans 4.3.2 va être appliqué sur les individus constituant ce sous-ensemble. Or, pour une paire de triangles donnée, il n'existe qu'une configuration minimum, ce qui justifie l'étape de filtrage ainsi introduite.

5. **Optimisations :**
   Lors de cette étape, les individus issus de la phase de sélection (i.e. l'*élite* de la population) sont optimisés. Cette optimisation est assimilable à une exploitation supplémentaire des pions concernés. Dans le cas polyédrique, le processus consiste en l'application de l'opérateur d'exploitation déterministe à chacun de ces individus (Cf. section 4.3.2). Il est à noter que cette phase d'optimisation agit directement sur une partie de la population globale, ce qui justifie le bouclage observable sur le schéma proposé (Cf. figure 4.17).

6. **Filtrages et Sortie :**
   Comme nous le verrons dans la suite du mémoire, l'algorithme ESPIONS peut se voir utilisé de différentes manières. Aussi, la sortie de l'algorithme n'est pas *a priori* identifiable à ce niveau de l'exposé. Cependant, on peut d'ores et déjà définir la sortie de l'algorithme comme une fonction de la population que constitue l'ensemble des pions. Dans la suite, on notera cette fonction $\omega(\Pi)$. Cette caractéristique de l'algorithme en fait clairement une approche parisienne, tel que présenté dans 3.6.1. Prenons un exemple ; Admettons que le problème à résoudre consiste en l'identification de la distance minimum globale séparant les objets. Dans ce cas, l'étape de filtrage va consister à ne retenir, pour sortie du processus, que l'individu présentant la meilleure *fitness*.

   Cette dernière étape du cycle générationnel va également permettre une formali-

sation des informations inhérentes aux individus. Nous entendons par là que la description d'un individu (génotypique ou phénotypique) ne constitue pas nécessairement une information directement utilisable par le processus dit *client*. Aussi, comme nous le verrons dans le cadre des couplages présentés dans le chapitre 5, une formalisation des informations portées par les individus sera généralement réalisée à ce niveau du processus.

4.5.3 Schémas d'intégration d'ESPIONS

Après avoir introduit le schéma de fonctionnement interne d'ESPIONS, nous proposons ici deux architectures visant à intégrer l'algorithme au sein d'un processus global de simulation.

Schéma séquentiel

La première architecture proposée consiste en un *couplage séquentiel* d'ESPIONS avec un autre algorithme dit *client*. La figure 4.18 illustre cette architecture. On y observe les échanges entre les deux modules algorithmiques. Ainsi, ESPIONS fournit à l'algorithme client un ensemble d'informations géométriques inhérentes à la description des contacts, et, en retour l'algorithme client fournit à ESPIONS les positions des éléments géométriques qui constituent la scène. Dans le cas rigide, ces dernières positions se limitent à celles des objets. Dans le cas déformable, d'autres informations devront être échangées, telles que les positions des sommets dans le cas polyédrique (ou du moins les positions des sommets s'étant déplacés). On peut également remarquer que, dans le cadre du schéma proposé ici, il est envisageable de réaliser plusieurs générations d'ESPIONS à chaque cycle du processus global. Nous verrons en effet dans la suite du mémoire qu'il peut s'avérer pertinent d'effectuer plusieurs cycles générationnels d'ESPIONS afin d'être en mesure de garantir la pertinence des informations transmises à l'algorithme *client*.

Schéma parallèle

La deuxième architecture proposée consiste en un *couplage parallèle* d'ESPIONS avec l'algorithme *client*. La figure 4.19 illustre cette architecture. Contrairement au schéma séquentiel présenté précédemment, les algorithmes sont ici assimilables à deux *processus distincts et non-synchronisés*. Bien qu'elles soient de même nature que dans le cas séquentiel, les informations géométriques sont désormais échangées via des *espaces mémoire partagés*. Cette architecture permet de répartir la charge de calcul sur plusieurs unités. Elles offrent ainsi la possibilité d'exploiter pleinement les plateformes matérielles munies

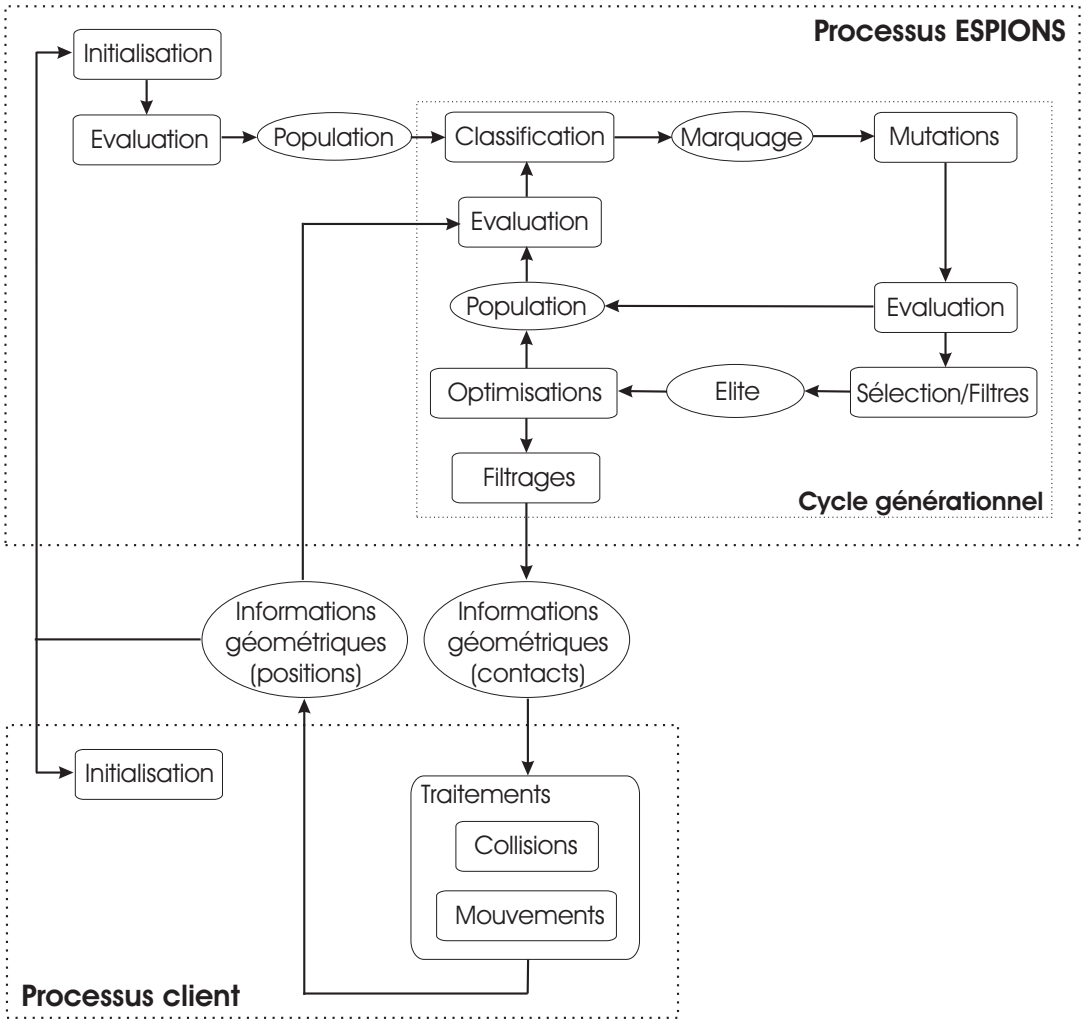


FIG. 4.18 – Architecture permettant une intégration d'ESPIONS selon un schéma séquentiel.

de plusieurs processeurs. On peut également remarquer que les espaces mémoire partagés ne sont respectivement modifiés que par l'un des modules algorithmiques, le second accédant à l'espace mémoire en lecture seule. Aussi, cette architecture ne nécessite pas de datation des informations partagées.

4.5.4 Distribution du processus ESPIONS

Dans la section 3.6.3 du chapitre dédié à la présentation des concepts fondamentaux inhérents aux algorithmes évolutionnaires, nous avons introduit la notion de distribution des processus basés sur de telles approches. Dans le cadre de l'algorithme ESPIONS, nous proposons de mettre en œuvre une technique proche de celles proposées dans la littérature

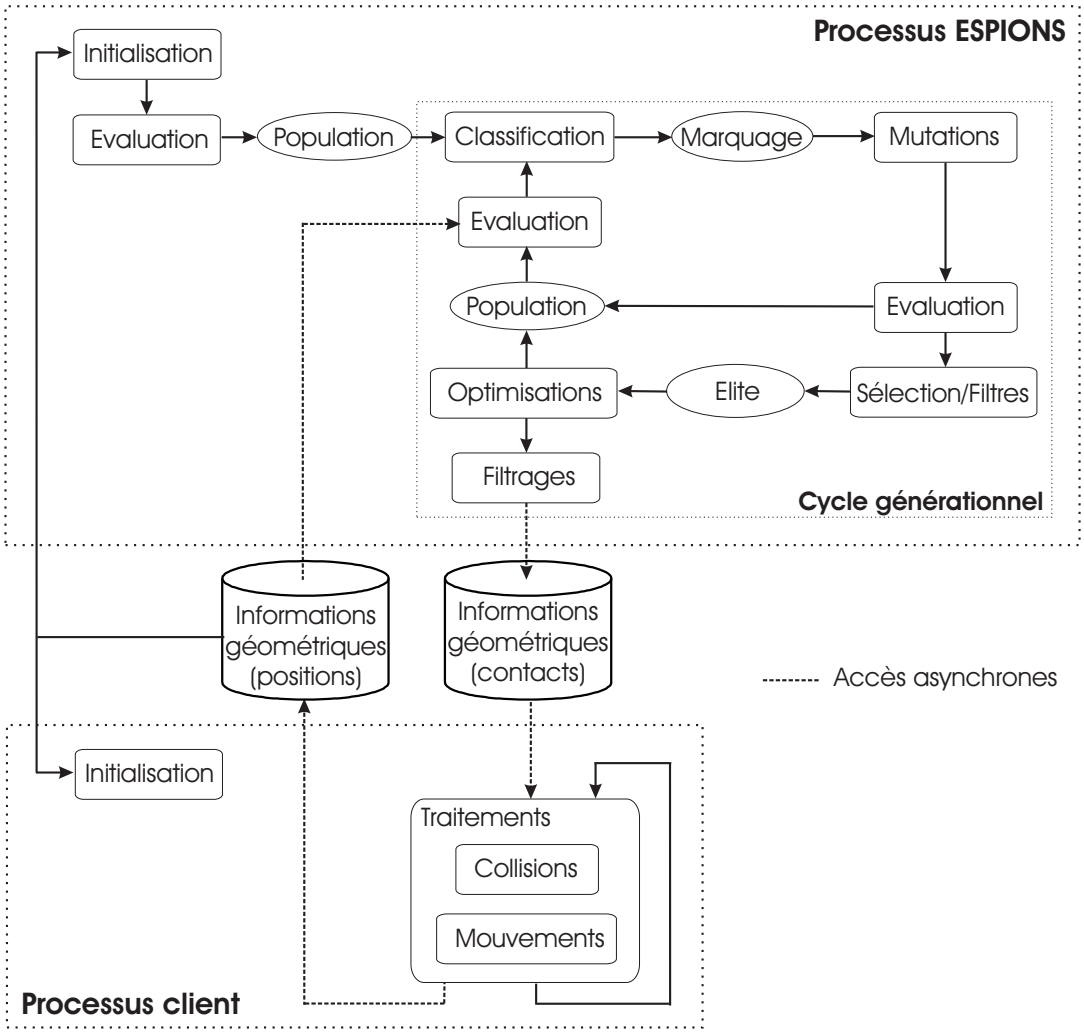


FIG. 4.19 – Architecture permettant une intégration d'ESPIONS selon un schéma parallèle.

et précédemment introduites dans ce contexte. L'idée est de créer plusieurs populations et d'attribuer à chacune d'entre elles une portion de l'espace de recherche. On voit apparaître là les notions de sous-populations et de sous-espaces. Aussi, nous proposons de limiter les déplacements des nucléons constituant les individus de ces sous-populations au sein d'espaces de recherche réduits à des paires d'objets (i.e. les sous-espaces) ; L'union de ces paires constituant l'ensemble de la combinatoire de la scène considérée. Pour ce faire, on opère une restriction au niveau de l'opérateur de repositionnement aléatoire, seul opérateur à permettre des sauts inter-géométries (i.e. deux nucléons qui composent un individu évoluent nécessairement à la surface de géométries dont la paire est incluse dans le sous-espace attribué à la population qui contient l'individu).

La figure 4.20 illustre le schéma mis en œuvre dans ce contexte. On retrouve les

mêmes composantes que dans le cadre du couplage parallèle précédemment introduit. Cependant, on voit apparaître un second processus ESPIONS partageant les mêmes espaces mémoires avec le processus *client*. Aussi, chacun des deux processus ESPIONS se voit attribuer un sous-ensemble des objets constituant la scène. Il est à noter que d'autres processus ESPIONS peuvent être introduits dans ce schéma, l'idée étant de créer autant de processus qu'il n'y a d'unités de calcul attribuables au processus ESPIONS global (l'une des unités devant être attribuée au processus client).

4.6 Discussion concernant la cohérence spatio-temporelle

Dans le cadre de l'état de l'art dédié à la présentation des approches inhérentes à la détection des collisions, nous avons abordé la notion de cohérence spatio-temporelle (Cf. paragraphe 2.5). Rappelons que ce concept se fonde sur l'hypothèse selon laquelle, dans le cadre de simulations impliquant un mouvement pseudo-continu des éléments de la scène, les zones de proximité observables entre des objets à un cycle donné se situe dans le proche voisinage des zones de proximité observées entre ces mêmes objets lors des cycles précédents.

Les schémas mis en œuvres quant à l'intégration de notre algorithme au sein de processus de simulation (Cf. section 4.5.3) permettent une prise en considération implicite de ce phénomène. En effet, le fait que la population évolue de façon perpétuelle tout au long d'une simulation entraîne que, lors de chaque déplacement des objet, les nucléons qui composent les individus reliant des zones de proximité sont nécessairement situés dans le proche voisinage des zones portant les distances minimum inhérentes à la nouvelle configuration. Aussi, les nucléons n'auront que peu de chemin à parcourir pour atteindre ces zones et l'identification des nouvelles positions optimales sera réalisée dans un temps réduit.

Il est à noter que cette caractéristique constitue l'un des atouts majeurs de l'approche que nous proposons. En effet, sans garantir l'identification systématique de l'ensemble des informations de contact, elle va engendrer des temps de convergence réduits vers les zones de proximité, et ainsi contribuer au bon fonctionnement du processus globale.

4.7 Conclusion

Au travers de ce chapitre, nous avons présenté l'algorithme ESPIONS qui constitue la principale contribution quant aux travaux réalisés dans le cadre de cette thèse. Au delà

des principes généraux qui régissent cet algorithme, nous avons introduit plusieurs formalismes permettant son utilisation dans le cadre de simulations mettant en scène des objets de différents types : polyèdres, nuages de points, surfaces paramétriques. Nous avons également présenté le schéma de fonctionnement général de l'algorithme en prenant soins de détailler l'ensemble des étapes qui le composent. Enfin, plusieurs modes opératoires visant à son intégration au sein de processus de simulations ont été proposés.

Discussion concernant la détection d'interpénétrations

Dans ce chapitre, et plus généralement dans le cadre de cette thèse, ESPIONS peut être défini comme un processus visant à identifier des *champs de distances minimum locales* entre des objets. Ainsi, il peut se voir utilisé dans le cadre de simulations fondées sur le formalisme selon lequel, dans un contexte discret, la collision entre deux objets peut être assimilée à une trop grande proximité entre ces derniers (Cf. paragraphe 2.4.1). Le couplage avec l'algorithme LMD++ proposé en section 5.3 illustre bien cette aptitude.

Cependant, comme il l'a également été expliqué (Cf. paragraphe 2.4.1), les approches proposées dans la littérature consistent souvent à considérer qu'il y a collision entre deux objets lorsqu'une intersection entre ces derniers est observable, auquel cas les informations de contact sont souvent définies comme des vecteurs d'interpénétration. En l'état actuel de l'avancement de nos travaux, ESPIONS ne peut être utilisé dans ce contexte. En effet, le fait de tendre à minimiser la taille des individus, sans prise en considération du signe de la distance séparant les nucléons qui les composent, compromet la pertinence des informations issues de l'algorithme lors d'interpénétrations. Lorsque qu'il y a interpénétration entre les objets, les pions tendent à s'agglutiner au niveau des intersections entre les objets au niveau desquelles leur *fitness* s'annule.

Une voie d'investigation pour répondre à cette problématique pourrait consister à redéfinir la fonction *fitness* en prenant en considération les normales aux surfaces des objets considérés. La *fitness* serait ainsi assimilable à une distance signée et ESPIONS continuerait à fonctionner en cas d'intersection. L'algorithme tendrait alors à identifier des distances maximum d'interpénétration, assimilables aux plus grandes diagonales des enveloppes convexes englobant chacune des intersections inter-objets. Ce type d'informations ne sera cependant pas exploitable en l'état, un vecteur d'interpénétration étant généralement défini comme le déplacement minimum à réaliser pour ramener les objets à une situation de contact exact. De plus une telle approche imposerait des contraintes fortes quant à la définition géométrique des objets simulés : connexité, orientation des normales aux surfaces, ...

Aussi, une perspective intéressante aux travaux présentés dans cette thèse pourrait

consister en une adaptation d'ESPIONS au cas de simulations dans le cadre desquelles les collisions entre les objets sont définies comme des interpénétrations entre ces derniers.

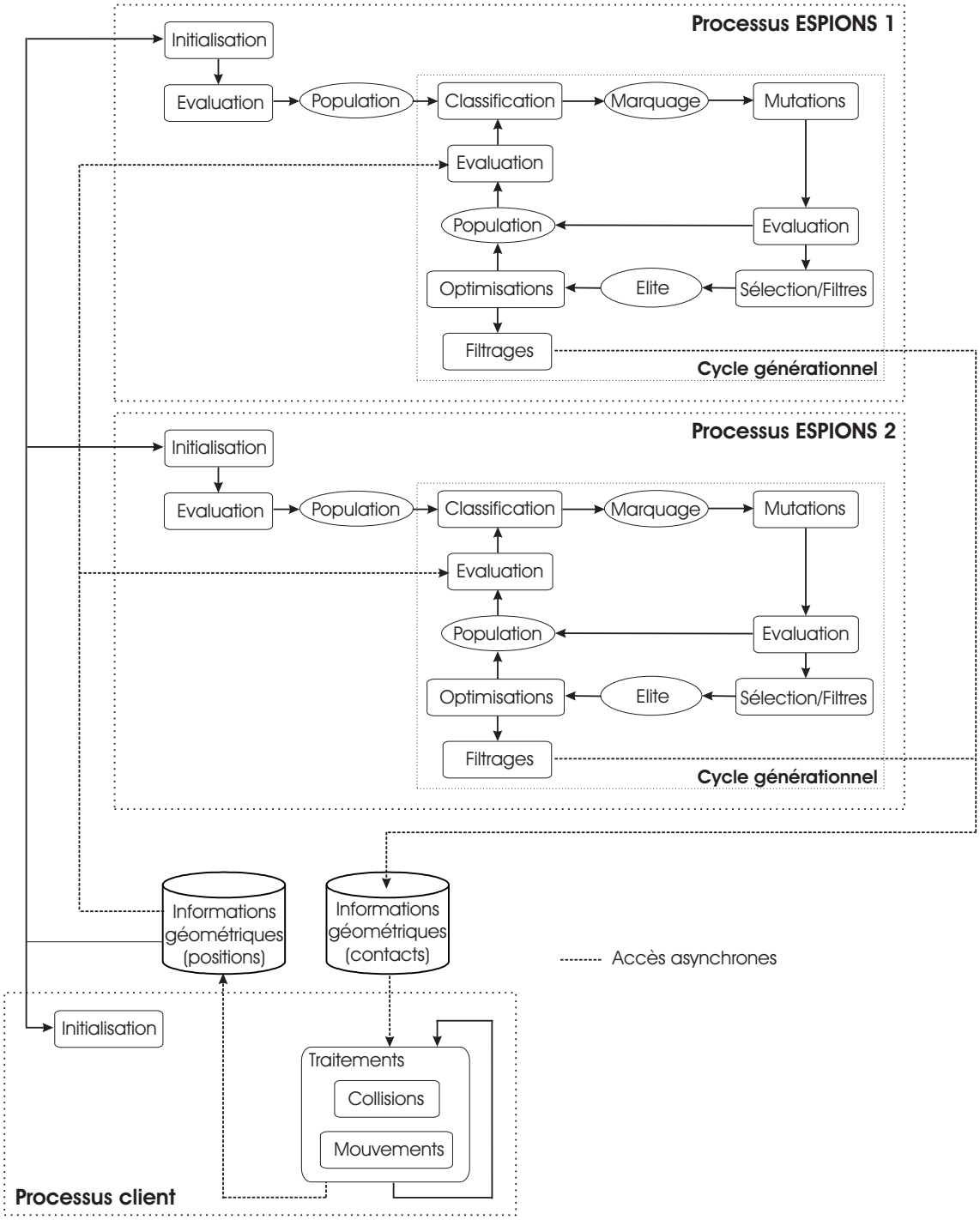
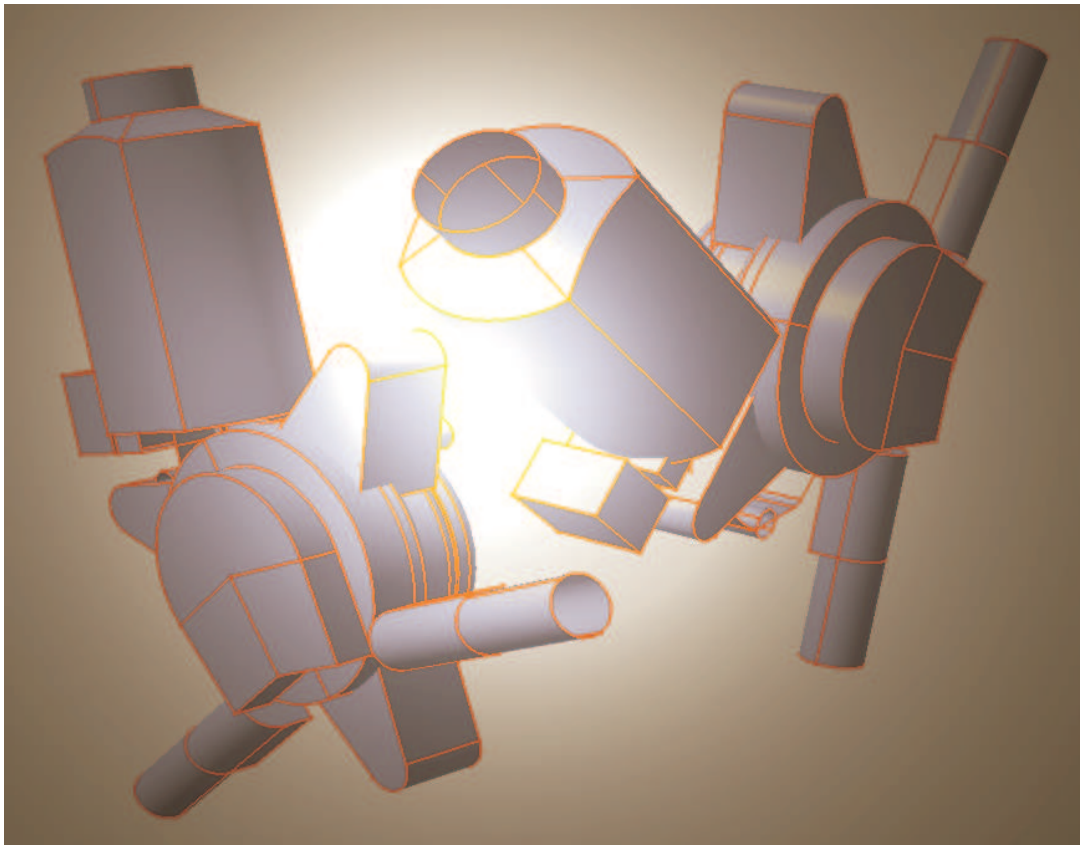


FIG. 4.20 – Architecture permettant une distribution du processus ESPIONS. L'idée est de partager l'espace à explorer entre plusieurs threads afin de répartir la charge sur plusieurs unités de calcul.

Chapitre 5

Analyse expérimentale d'ESPIONS et implémentations dans un contexte applicatif



5.1 Introduction : ESPIONS, de la théorie à la pratique

Dans le chapitre précédent, nous avons proposé une présentation générale de l'ensemble des composantes de l'algorithme ESPIONS. Nous y avons introduit les concepts fondamentaux de l'algorithme et les différents procédés et modes opératoires qui le composent. Au travers de ce nouveau chapitre, nous proposons une analyse du fonctionnement d'ESPIONS. Pour ce faire, nous procédons en deux phases.

Dans un premier temps, nous réalisons une étude expérimentale de l'algorithme dans un cadre général (i.e. sans prise en compte de l'aspect applicatif). Cette première étape a pour but de caractériser l'effet des différents paramètres de l'algorithme sur son comportement général. La finalité de cette démarche, fondée sur l'évaluation de certaines caractéristiques de l'algorithme, consiste en la mise en place de critères visant à faciliter l'élaboration de jeux de paramètres pertinents en fonction de la nature du problème à résoudre.

Dans un deuxième temps, nous proposons d'étudier le fonctionnement d'ESPIONS en nous basant sur des implémentations concrètes de ce dernier. Nous souhaitons ainsi valider les hypothèses et affirmations présentées dans le chapitre précédent, et mettre en évidence les apports et limites de notre algorithme dans des contextes spécifiques. Pour ce faire, nous exposons différentes applications concrètes de notre algorithme dans le cadre de deux couplages visant à son intégration au sein de moteurs physiques. Nous souhaitons ainsi illustrer deux modes d'utilisation d'ESPIONS : Le premier consistant en son intégration au sein d'un pipeline de détection de collisions en vue d'améliorer les performances du processus, et le deuxième en une utilisation d'ESPIONS comme un module de détection des collisions à part entière, dans le cadre de simulations mettant en scène des objets déformables. Nous proposons également une implémentation visant à illustrer la parallélisation du processus ESPIONS, afin d'exploiter pleinement les plateformes matérielles pourvues de plusieurs unités de calcul.

5.2 Etude expérimentale de l'algorithme ESPIONS

Dans cette première partie du chapitre, nous proposons une étude détaillée de l'effet des différents paramètres de l'algorithme ESPIONS sur son comportement général. Nous souhaitons ainsi mettre en évidence l'influence des différents *leviers* dont nous disposons :

- La proportion d'application des différents opérateurs de mutation : Elle est spécifiée au travers des paramètres conditionnant le marquage réalisé lors des phases de classification décrite dans 4.5.2 ($\%_R$, $\%_X$ et $\%_E$).
- Le choix de l'opérateur d'exploitation : Stochastique ou déterministe (Cf. section 4.3.2).
- Le maintien de la diversité : Il est le résultat du processus de partage. Son effet est conditionné par le paramètre d'ajustement γ (Cf. paragraphe 4.4.3).
- La taille de la population.

Afin de mesurer les effets de ces éléments de réglage, nous raisonnons sur deux caractéristiques de l'algorithme : le temps de convergence et le maintien de la diversité. Nous retrouvons là la notion de compromis entre rapidité de convergence et évitement des minima locaux telle qu'elle a été introduite lors de la présentation des concepts fondamentaux inhérents aux approches évolutionnaires (Cf. paragraphe 3.4.1).

En fin de section, nous proposons une synthèse des résultats obtenus dans le cadre de cette étude et discutons de la mise en place d'une aide méthodologique pour le paramétrage d'ESPIONS.

5.2.1 Caractéristiques de l'algorithme

Dans la suite, nous décrivons les modes opératoires mis en œuvre pour l'évaluation des caractéristiques de l'algorithme que sont : le temps de convergence et le maintien de la diversité.

Temps de convergence

A ce niveau de l'exposé, nous en sommes encore à une présentation générique de l'algorithme ESPIONS, sans prise en considération de la nature des informations échangées entre celui-ci et le processus avec lequel il interagit. Aussi, nous proposons ici de raisonner sur le temps nécessaire à ESPIONS pour identifier la distance minimale globale séparant les objets. Ce critère n'est pas pleinement représentatif de la convergence de l'algorithme de par le fait que, comme il l'a été expliqué précédemment, le couplage

d'ESPIONS avec un autre processus va, dans un cadre général, se faire selon une approche dite *parisienne*, consistant en une composition de la solution (i.e. chaque individu retenu constitue seulement une partie de la solution). La sortie d'ESPIONS est alors assimilable à un champ de distances minimum locales et non à la seule distance minimum globale observable entre les objets concernés. Cependant, cette aptitude de l'algorithme à identifier la distance minimum globale nous semble être une critère pertinent pour l'étude de l'effet des opérateurs sur la rapidité de convergence d'ESPIONS. Aussi, nous proposons donc d'étudier l'évolution de la valeur de la fitness du meilleur individu en fonction du nombre de générations réalisées. Cette évolution sera également exprimée en fonction du temps écoulé, la proportion d'utilisation des différents opérateurs ayant un effet, parfois sensible, sur la durée moyenne d'une génération.

Note importante : Il semble important de remarquer que, dans la suite de cette section, nous mesurons le temps nécessaire à l'algorithme pour identifier la distance minimum globale avec pour conditions initiales une répartition aléatoire des individus au sein de l'espace de recherche sans cycles générationnels préalables. Ces expériences ne permettent donc pas l'exploitation implicite de la cohérence spatio-temporelle telle qu'elle a été décrite dans le chapitre précédent (Cf. paragraphe 4.6), ce qui explique des temps de convergence relativement long au vu de la tâche pour laquelle ESPIONS a été conçu. Il est également à noter que, afin de bien découpler les effets des différents éléments étudiés, nous avons choisi d'exploiter systématiquement les individus vainqueurs par application de l'opérateur d'exploitation déterministe, tel que défini dans le chapitre précédent (Cf. section 4.3.2). Or, comme nous le verrons dans la suite de cette section, ce type d'opérateurs a un coût important en termes de temps de calcul. Nous verrons d'ailleurs dans les parties dédiées à la présentation de couplages réalisés entre ESPIONS et d'autres algorithmes que, dans un contexte temps réel, l'utilisation de cet opérateur doit se faire dans une moindre mesure, afin de réduire le temps nécessaire à l'accomplissement d'une génération d'ESPIONS.

Maintien de la diversité

Dans la partie de l'état de l'art dédiée à la présentation des approches évolutionnaires, nous avons introduit la notion de compromis entre rapidité de convergence et évitement des minima locaux (Cf. paragraphe 3.4.1). Dans le cas de l'algorithme auquel nous nous intéressons ici, la gestion de ce compromis s'avère particulièrement complexe de par une grande diversité des espaces à explorer (objets convexes, concaves, etc.). Aussi, afin d'être en mesure d'évaluer l'effet des différents opérateurs sur l'aptitude de l'algorithme à considérer l'ensemble des régions potentiellement intéressantes au sein de l'espace de

recherche, nous proposons ici de définir un indicateur visant à mesurer la diversité génétique au sein de la population.

Dans le chapitre précédent, nous avons introduit la notion de densité dans le cadre de notre algorithme. Rappelons que cette dernière s'exprime en nombre de pions par paires de régions et que son évaluation est fondée sur la mise en place d'une carte de fréquentation des différentes paires de régions. Aussi, afin de permettre une comparaison entre les différentes configurations de la population obtenues lors des expériences présentées dans la suite, nous proposons d'évaluer la diversité génétique en calculant l'écart type sur la répartition des individus au sein de l'espace de recherche. Ainsi, dans le cas d'une simulation contenant deux géométries, la diversité observable pour une configuration donnée s'exprime de la façon suivante :

$$\delta = \frac{1}{\sigma}$$

avec

$$\sigma = \sqrt{\frac{1}{n_{reg_1} n_{reg_2} - 1} \sum_{i=1}^{n_{reg_1}} \sum_{j=1}^{n_{reg_2}} (\rho_i^j - \bar{\rho})^2}$$

où la diversité et l'écart type sont respectivement notés δ et σ. n_{reg_1} et n_{reg_2} correspondent au nombre de régions constituant chacun des objets considérés. ρ_i^j correspond au nombre de pions reliant les régions i (du premier objet) et j (du deuxième objet). Enfin, $\bar{\rho}$ correspond à la moyenne du nombre de pions par paire de régions.

Interprétation des caractéristiques en termes de performances

Comme il l'a été dit, dans le cadre de cette étude expérimentale, nous souhaitons fonder notre raisonnement sur deux caractéristiques de l'algorithme : le temps de convergence et le maintien de la diversité. Aussi, avant de poursuivre, il semble intéressant de faire le lien entre ces caractéristiques et les performances proprement dites de l'algorithme. Nous entendons là par performances :

L'aptitude d'ESPIONS à produire, *dans le temps qui lui est imparti*, l'*ensemble des informations* nécessaires à la résolution du problème considéré.

L'étude du temps de convergence permet de raisonner sur le premier critère issu de cette formulation, soit l'aptitude de l'algorithme à converger rapidement vers des zones d'intérêt. En effet, une augmentation du temps nécessaire à la convergence de la population vers une configuration optimale est susceptible de compromettre le bon fonctionnement de l'algorithme. Prenons le cas d'une utilisation d'ESPIONS comme accélérateur

d'un module de détection des collisions selon un schéma séquentiel, telle que celle proposée dans la suite de ce chapitre (Cf. section 5.3 expérience 2). Dans ce contexte, ESPIONS bénéficie d'un nombre fini de générations pour identifier l'ensemble des zones de proximité nécessaires au module client avec lequel il est couplé. Aussi, dans l'hypothèse où le nombre de générations nécessaires à la convergence s'avère élevé, ESPIONS est susceptible de ne pas identifier, *dans le temps qui lui est imparti*, tout ou partie des éléments recherchés. Au delà de cette constatation relativement triviale, on peut également observer qu'une augmentation du temps de convergence porte préjudice à la prise en considération de la cohérence spatio-temporelle précédemment introduite (Cf. paragraphe 4.6). En effet, le fait que la population n'atteigne pas une configuration optimale dans le cadre d'un cycle donné va directement influer sur le comportement de l'algorithme au cycle suivant, de par un mauvais conditionnement initial de la population. On voit ainsi apparaître un phénomène que l'on pourrait qualifier d'*inertie de l'algorithme ESPIONS*. La figure 5.1 illustre ce phénomène.

L'étude de la deuxième caractéristique retenue dans le cadre de cette analyse, soit la diversité génétique observable au sein de la population, a pour but d'évaluer l'aptitude de l'algorithme à identifier *l'ensemble des informations* nécessaires à la résolution du problème considéré. Elle prend tout son sens lorsque l'on s'intéresse au cas de simulations mettant en scène des éléments concaves. En effet, dans le cadre de telles simulations, qui constituent l'immense majorité des cas à traiter, l'espace de recherche est susceptible de présenter une multitude de minima locaux. Aussi, plusieurs zones de proximité non-connexes peuvent apparaître entre les objets simulés, et il devient alors essentiel d'explorer l'espace de recherche dans son ensemble afin d'être en mesure de garantir l'identification de l'ensemble de ces zones d'intérêt.

5.2.2 Déroulement des expériences

Une expérience est ici assimilable à l'exécution d'un certain nombre de générations d'ESPIONS dans le cadre d'une scène statique constituée de deux objets. Une expérience est réalisée pour un jeu de paramètres donné (i.e. les paramètres d'ESPIONS). Afin de garantir la pertinence des informations issues des expériences, chacune est répétée à plusieurs reprises (en l'occurrence 200 fois), et les résultats obtenus sont moyennés. Bien entendu, chaque réalisation d'expérience est précédée d'un repositionnement aléatoire de l'ensemble des individus constituant la population ; Ceci afin de garantir l'absence d'une éventuelle prise en considération de la cohérence spatio-temporelle d'une expérience à l'autre (Cf. paragraphe 4.6).

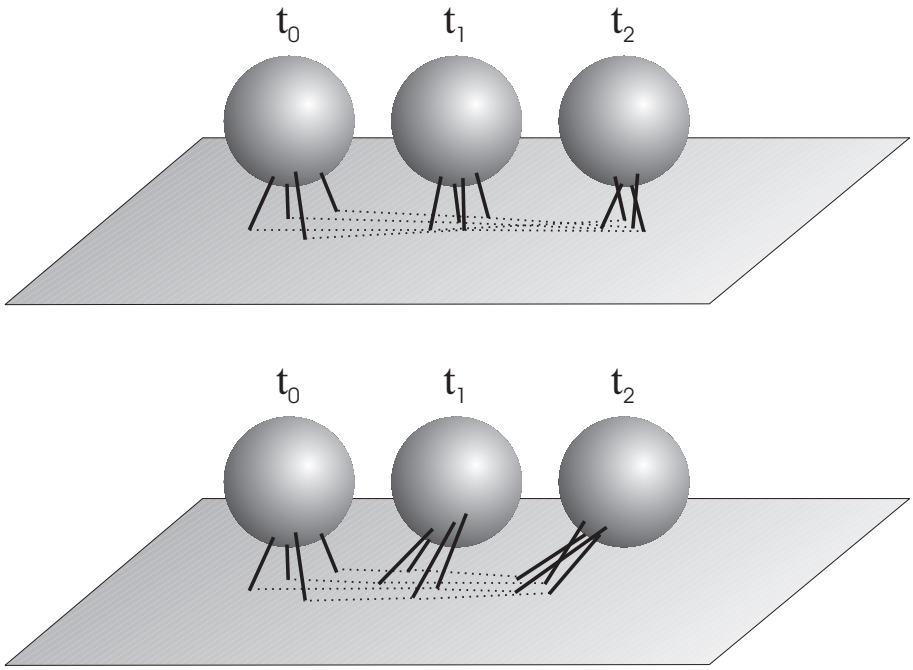


FIG. 5.1 – Illustration des effets du temps de convergence sur la cohérence spatio-temporelle. En haut, le temps de convergence est réduit. On observe que, d'un cycle à l'autre, les individus sont en mesure d'atteindre des configurations pertinentes. En bas, le temps de convergence est trop important pour permettre à l'algorithme de converger suffisamment. On observe une détérioration progressive de la configuration de la population. Une latence apparaît entre le déplacement des objets et la convergence de l'algorithme. C'est ce phénomène que nous assimilons à une inertie.

Rappel des notations

Ici, nous listons l'ensemble des paramètres conditionnant une expérience. Nous faisons également le lien avec les notations introduites dans le chapitre précédent :

- $\%_R$: Pourcentage d'individus ayant perdu durant le tournoi et devant subir un repositionnement aléatoire au sein de l'espace d'exploration.
- $\%_X$: Pourcentage d'individus ayant perdu durant le tournoi et devant être remplacés par des individus issus de croisements.
- $\%_E$: Pourcentage d'individus ayant perdu durant le tournoi et devant subir un opérateur de type exploration du voisinage.
- n_S : Nombre de sauts à réaliser au cours d'une mutation de type exploration du voisinage.
- γ : Paramètre d'ajustement de la forme de la fonction fitness visant de réguler le

partage des ressources.

– p : Taille de la population.

Description de la scène utilisée

La figure 5.2 illustre la scène utilisée dans le cadre des expériences dont les résultats sont traités dans la suite de cette section. Celle-ci contient en fait deux instances d'un même objet correspondant à un mécanisme de lève-vitre de voiture (modèle Mégane RENAULT). Chaque instance de cette géométrie est constituée d'environ 15000 polygones. Bien entendu, les positions des objets dans la scène ne seront pas modifiées d'une expérience à l'autre afin de préserver l'espace géométrique à explorer. Aussi, dans la configuration proposée, la distance minimum globale séparant les deux objets a pour valeur 14.7 millimètres. En ce qui concerne la décomposition des surfaces en régions, décomposition visant à établir la carte de densité (Cf. paragraphe 4.4), chacune des instances s'est vue décomposée en 78 régions, assimilables à des portions de surfaces appartenant respectivement à 78 sphères d'environ 25 millimètres de diamètre, réparties de façon homogène sur l'ensemble de la géométrie (décomposition réalisée selon le processus décrit dans 4.4).

Plateforme matérielle utilisée

L'ensemble des expériences dont les résultats sont étudiés dans cette section ont été réalisées sur un PC ayant les propriétés suivantes :

– Unité de calcul : Mono-processeur intel bi-cœurs cadencé à 3.6 GHz.

– Mémoire vive : 2.0 Go type DDR2.

– Système d'exploitation : Windows XP.

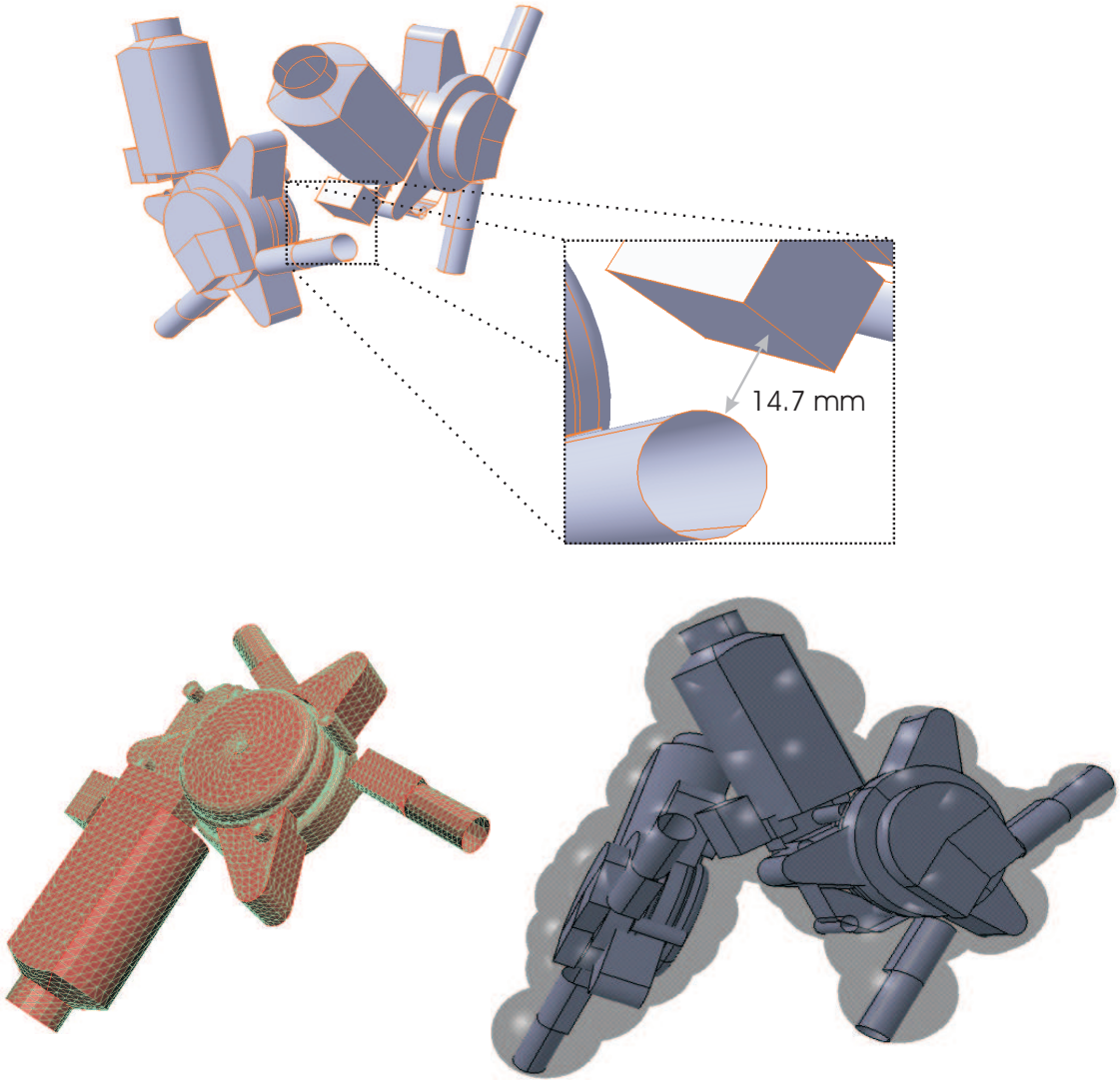


FIG. 5.2 – (En haut) Illustration de la scène servant de référence pour les expériences. Elle contient deux instances d'un même objet correspondant à un mécanisme de lève-vitre de voiture (modèle mégane RENAULT) distantes de 14.7 millimètres. (En bas à gauche) Chaque instance de cette géométrie est assimilable à un polyèdre constitué d'environ 15000 triangles. (En bas à droite) La décomposition des objets visant à établir la carte de proximité est obtenue par une décomposition en volumes englobants telle que présentée dans 4.4. Dans le cadre de cette expérience, le seuil de taille est de 25 millimètres. On obtient ainsi 78 régions pour chaque instance de l'objet.

5.2.3 Effets des opérateurs de variation

Dans cette section, nous proposons une étude des effets des différents opérateurs de mutation sur le comportement général de l'algorithme.

Opérateur de croisement

D'après la théorie relatives aux algorithmes évolutionnaires, les opérateurs de croisement ne sont pas nécessaires à la convergence de la population manipulée vers la solution au problème considéré. Ils ont pour vocation d'accélérer cette convergence par une exploitation de l'amélioration globale de la qualité des individus tout au long du processus d'évolution. Ce type d'opérateur entraîne une concentration des individus dans certaines zones de l'espace de recherche. Ils portent ainsi naturellement atteinte à la diversité génétique observable au sein de la population. Aussi, une utilisation trop importante d'opérateurs de cette nature peut théoriquement avoir pour effet d'empêcher la convergence de l'algorithme vers la solution optimale au problème qu'il vise à résoudre. Les zones de l'espace présentant de meilleures caractéristiques n'étant pas explorées, les meilleurs individus de la population vont *aspirer* le reste de la population vers des minima locaux. On parle de *super individus*.

Expériences réalisées

Au travers de cette première partie de l'étude, nous proposons une analyse visant à caractériser l'effet des opérateurs de croisement dans le cadre de l'algorithme ESPIONS. Trois expériences servent de support à cette analyse. Dans le tableau 5.1, nous présentons les jeux de paramètres spécifiés pour chacune d'entre elles.

	$\%_R$	$\%_X$	$\%_E$	n_S	γ	p
Expérience 1	100.0	0.0	0.0	0	0.0	500
Expérience 2	60.0	40.0	0.0	0	0.0	500
Expérience 3	0.0	100.0	0.0	0	0.0	500

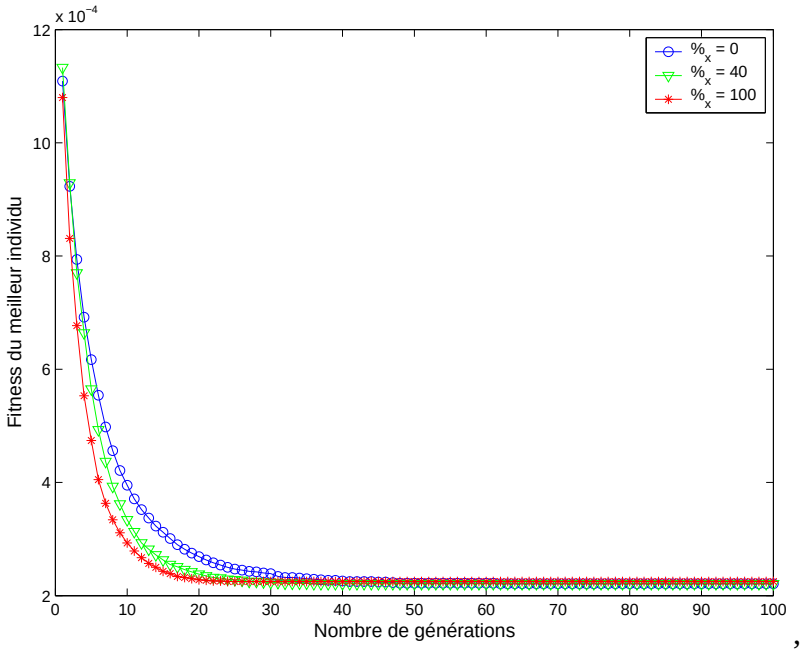
TAB. 5.1 – Paramètres utilisés pour les expériences visant à évaluer l'influence des opérations de croisement.

Résultats obtenus et analyse

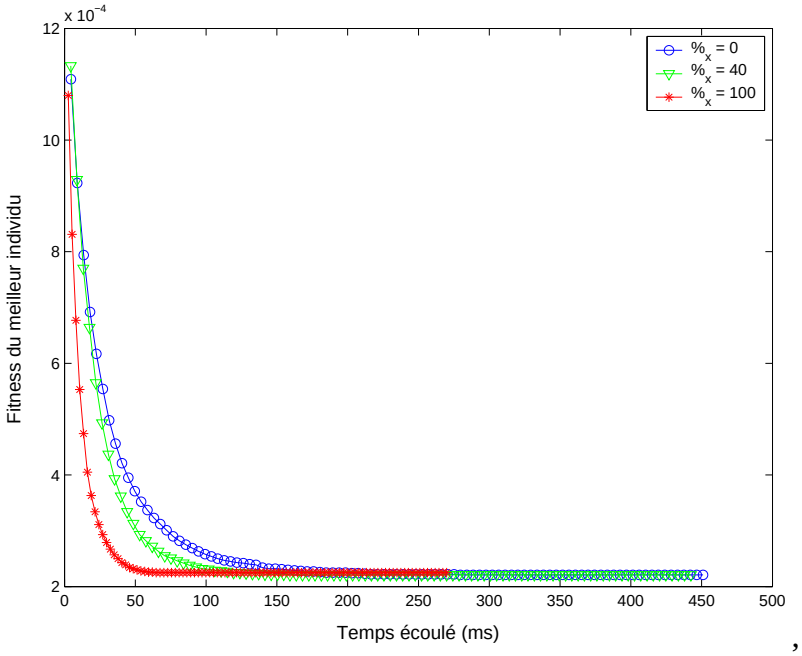
La figure 5.4 illustre les résultats obtenus lors des trois expériences définies précédemment. Sur les figures 5.3(a) et 5.3(b), décrivant respectivement l'évolution de la meilleure

fitness en fonction du nombre de générations et du temps écoulé, on peut observer que, conformément à la théorie, l'augmentation de la proportion d'application de l'opérateur de croisement lors du processus d'évolution engendre une accélération sensible de la convergence de l'algorithme. Cette observation est d'autant plus observable sur la courbe prenant le temps pour abscisse, le coût (au sens temporel) d'une opération de croisement s'avérant moins important que celui inhérent à un repositionnement aléatoire.

Les deux courbes présentées sur les figures 5.4(a) et 5.4(b) permettent d'observer l'effet des opérations de croisement sur la capacité de l'algorithme à explorer l'ensemble de l'espace de recherche. La courbe 5.4(a) décrit l'évolution de la diversité génétique en fonction du nombre de générations réalisées. On y observe une influence importante de la proportion d'application de l'opérateur de croisement. Comme expliqué précédemment, ce phénomène peut en théorie engendrer l'apparition de super individus provoquant une convergence partielle de l'algorithme vers des minima locaux. Sur la courbe 5.4(b), qui permet une observation précise de la qualité de la solution trouvée par l'algorithme, on constate en effet que, dans le cadre de la troisième expérience, où l'opérateur de croisement se voit massivement utilisé, l'algorithme se retrouve dans certains cas piégé dans des minima locaux.

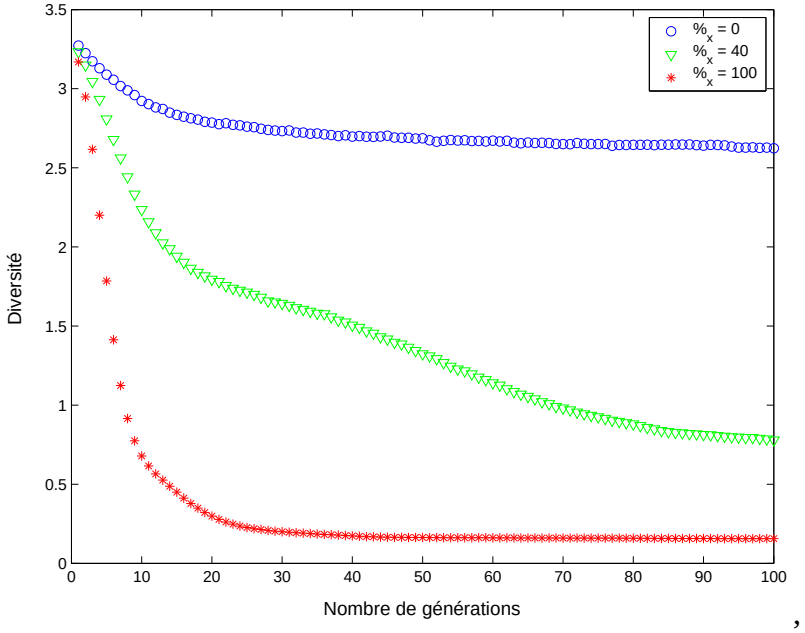


(a) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différentes valeurs de $\%_X$.

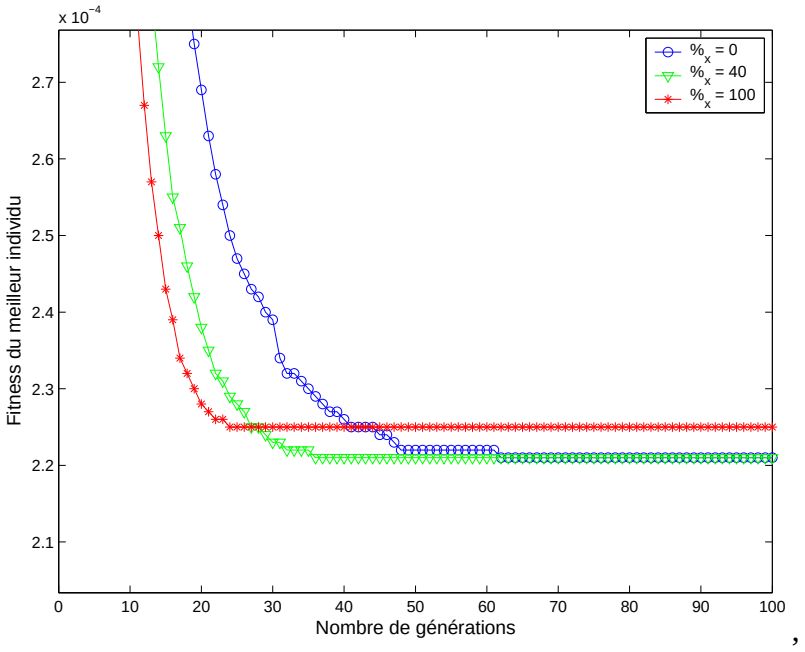


(b) Evolution de la meilleure *fitness* en fonction du temps pour différentes valeurs de $\%_X$.

FIG. 5.3 – Etude des effets des opérations de croisement sur l'évolution de la meilleure *fitness*.



(a) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs de $\%_X$.



(b) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différentes valeurs de $\%_X$.

FIG. 5.4 – Etude des effets des opérations de croisement sur la diversité génétique.

Opérateur d'exploration du voisinage

Dans cette partie, nous nous intéressons à l'effet des opérations d'exploration du voisinage telles que définies, pour ce qui est du cas polyédrique, dans la section 4.3.1.

Expériences réalisées

Afin de caractériser l'effet des opérateurs d'exploration du voisinage sur le comportement de l'algorithme ESPIONS, nous nous basons sur les résultats obtenus dans le cadre de cinq expériences visant à mettre en évidence l'influence des paramètres $\%_E$ et n_S. Dans le tableau 5.2, nous présentons les jeux de paramètres spécifiés pour chacune d'entre elles.

	$\%_R$	$\%_X$	$\%_E$	n_S	γ	p
Expérience 1	100.0	0.0	0.0	0	0.0	500
Expérience 2	60.0	0.0	40.0	10	0.0	500
Expérience 3	0.0	0.0	100.0	10	0.0	500
Expérience 4	60.0	0.0	40.0	5	0.0	500
Expérience 5	0.0	0.0	100.0	5	0.0	500

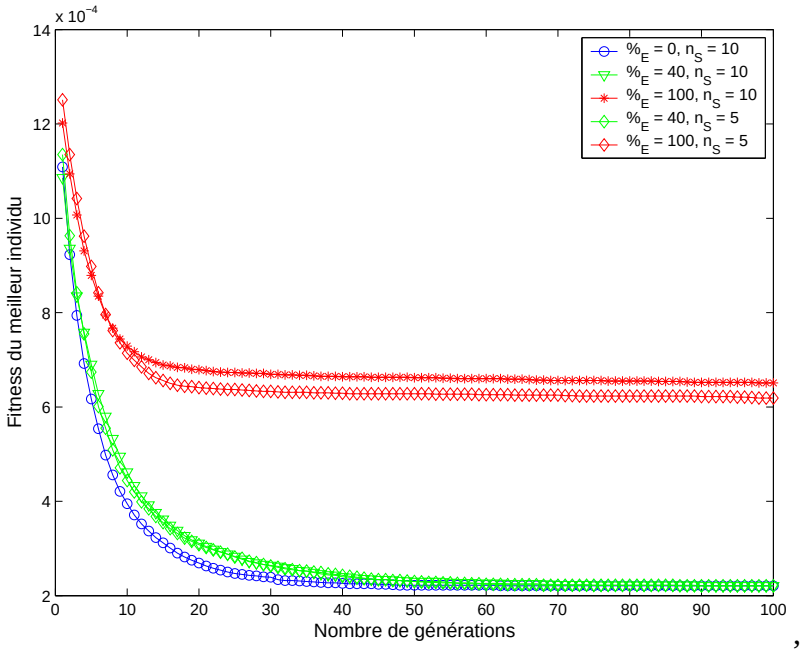
TAB. 5.2 – Paramètres utilisés pour les expériences visant à évaluer l'influence des opérations d'exploration du voisinage.

Résultats obtenus et analyse

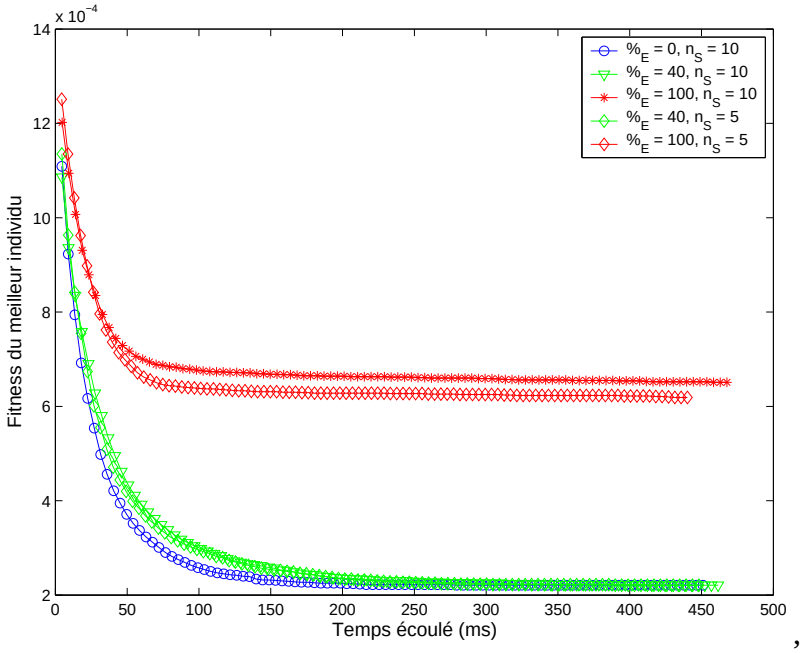
La figure 5.5 présente les courbes illustrant les effets des opérations d'exploration du voisinage sur l'évolution de la meilleure *fitness* au cours du processus d'évolution. Comme on pouvait s'y attendre, la proportion d'utilisation d'un tel opérateur de mutation a une influence sensible sur la capacité générale de l'algorithme à identifier des zones d'intérêt au sein de l'espace de recherche. Le but de ces opérateurs consistant en une exploration locale de la périphérie des individus affectés, on peut constater qu'une utilisation massive de ces derniers entraîne la convergence d'une grande partie de la population vers des zones présentant des minima locaux dont les individus ne peuvent s'échapper à cause d'une amplitude de déplacement réduite. En ce qui concerne les effets du paramètre n_S, qui correspond, rappelons-le, au nombre de sauts réalisés pour chaque mutation, on peut constater qu'il n'a que peu d'influence sur le comportement global de l'algorithme. Il semble cependant intéressant de remarquer que ce paramètre a un effet direct sur la durée des générations qui, même s'il est moindre, justifie une minimisation de ce paramètre

(Cf. 5.5(b)). Cette différence de durée découle naturellement du coût de calcul inhérent à chaque saut (i.e. chaque saut constitue une opération qui a un coût).

Sur la figure 5.6, on peut observer les effets de la proportion d'utilisation des opérations d'exploration du voisinage sur l'évolution de la diversité génétique. Comme on pouvait s'y attendre, les opérations d'exploration du voisinage entraînent une diminution de la diversité génétique. Cet effet est particulièrement observable sur les courbes décrivant l'évolution de la diversité avec $\%_E = 100.0$ (Cf. 5.6(a)). Ce phénomène découle d'une convergence locale des individus vers les minima locaux présents dans les régions qu'ils occupent. En ce qui concerne l'effet du paramètre n_S dans ce contexte, on constate que sa diminution entraîne également une réduction de la diversité génétique. Il semble en effet normal qu'une limitation de l'amplitude de variation lors des mutations entraîne une localisation des individus en des mêmes lieux. Sur la figure 5.6(b) qui propose une illustration de l'évolution de la diversité pour différentes valeurs de n_S et à $\%_E$ constant, on peut également observer, dans une moindre mesure, ce phénomène.

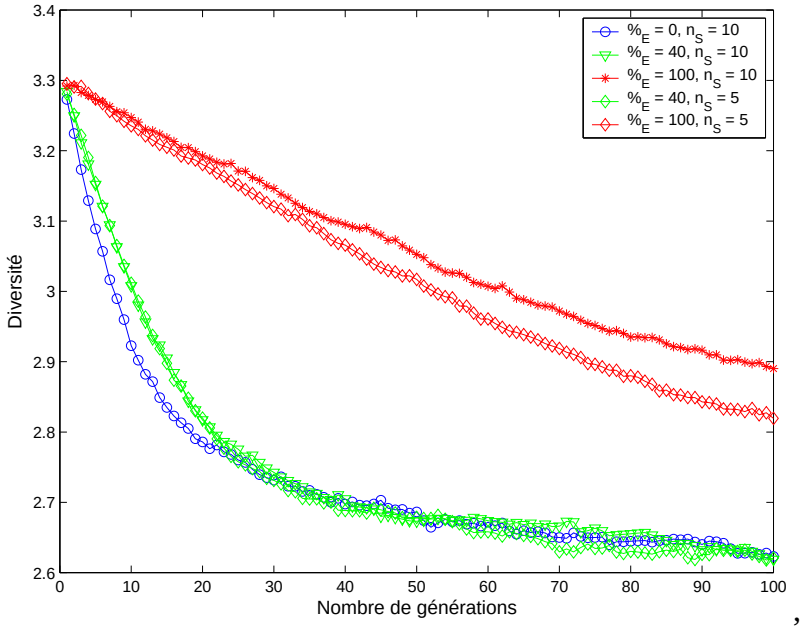


(a) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différentes valeurs des paramètres $\%_E$ et n_S.

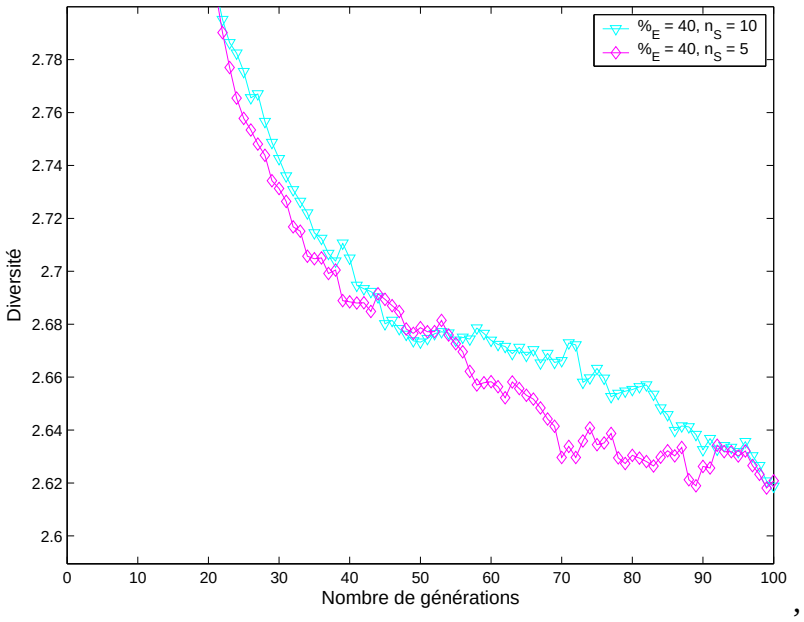


(b) Evolution de la meilleure *fitness* en fonction du temps pour différentes valeurs des paramètres $\%_E$ et n_S.

FIG. 5.5 – Etude des effets des opérations d'exploration du voisinage sur l'évolution de la meilleure *fitness*.



(a) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs des paramètres $\%_E$ et n_S.



(b) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs du paramètre n_S et à $\%_E$ constant.

FIG. 5.6 – Etude des effets des opérations d'exploration du voisinage sur la diversité génétique.

Opérateurs d'exploitation

Dans cette partie, nous nous intéressons à l'influence du type d'opérateur utilisé pour l'exploitation des meilleurs individus. Nous y comparons l'effet de deux des opérateurs présentés dans la section 4.3.2 : l'opérateur d'exploitation stochastique et l'opérateur d'exploitation déterministe avec considération du voisinage.

Expériences réalisées

Afin d'effectuer la comparaison qui nous intéresse ici, nous nous basons sur les résultats obtenus dans le cadre de deux expériences présentant toutes deux le même paramétrage mais utilisant chacune un type d'opérateur d'exploitation différent. Le tableau 5.3 résume les paramètres spécifiés dans ce contexte.

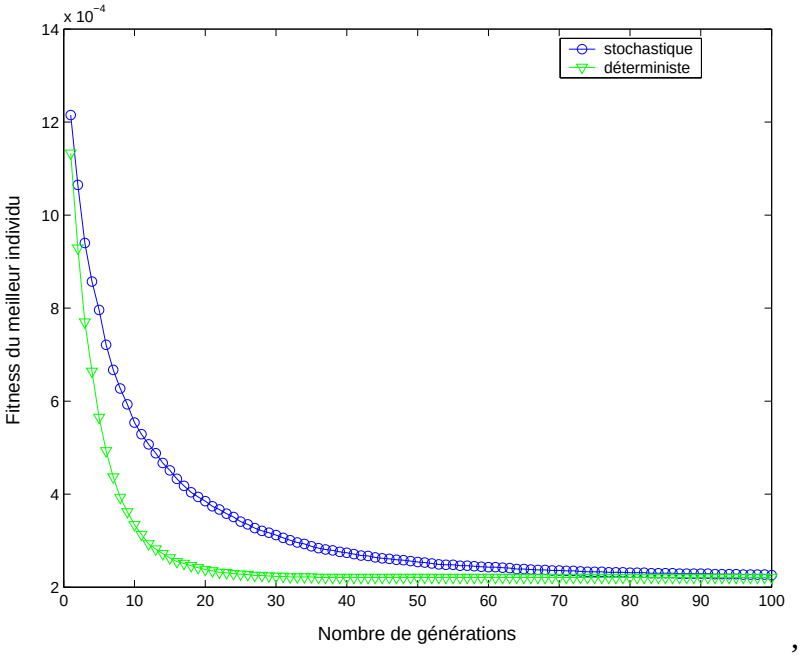
$\%_R$	$\%_X$	$\%_E$	n_S	γ	p
50.0	60.0	40.0	0	0.0	500

TAB. 5.3 – Paramètres utilisés pour les expériences visant à évaluer l'influence du type d'exploitation mis en œuvre.

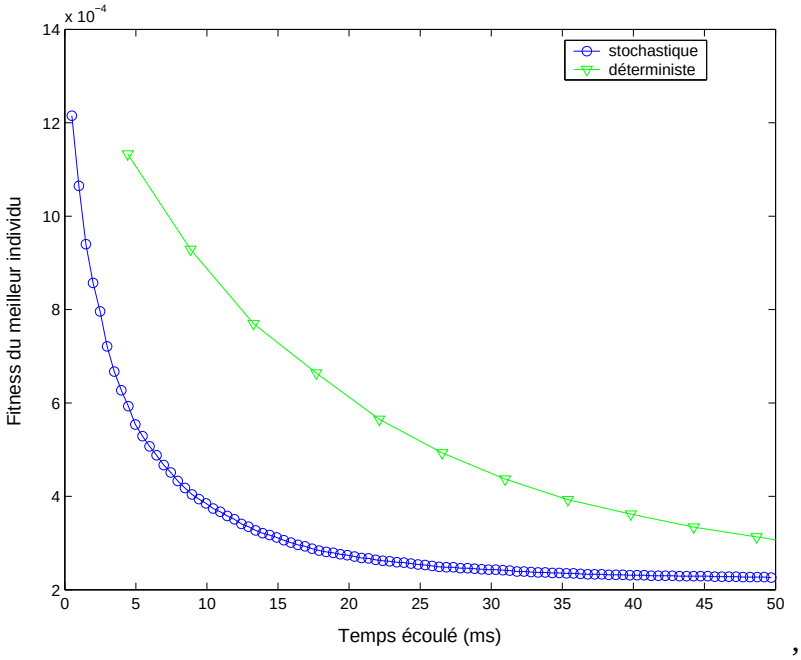
Résultats obtenus et analyse

En observant la figure 5.7(a), on constate que le nombre de générations nécessaires à la convergence de l'algorithme s'avère bien moindre dans le cas où les bons individus sont exploités via l'opérateur déterministe. Ce résultat semble bien naturel, la vocation de cet opérateur étant d'optimiser localement la configuration des individus à exploiter. Cependant, lorsqu'on raisonne en termes de temps, en se basant sur la courbe 5.7(b), on observe que le coût inhérent à une exploitation déterministe s'avère bien plus important que dans le cas d'une exploitation stochastique. Ce phénomène est d'autant plus notable que la convergence globale de l'algorithme se montre sensiblement plus rapide dans ce deuxième cas. Il est cependant à noter que l'on traite là de la convergence globale de l'algorithme et non de son aptitude à identifier la meilleure solution. En effet, dans le cas stochastique, rien ne permet de garantir que la solution optimale constituant la distance minimum globale sera effectivement atteinte en un temps fini, les nucléons pouvant très bien osciller indéfiniment autour des positions constituant la solution optimale.

En ce qui concerne le maintien de la diversité génétique, on observe une faible influence du type d'exploitation sur cette dernière, phénomène probablement lié à l'adoption d'une approche élitiste dans chacun des cas (Cf. 5.8).



(a) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différents types d'opérateurs d'exploitation.



(b) Evolution de la meilleure *fitness* en fonction du temps pour différents types d'opérateurs d'exploitation.

FIG. 5.7 – Etude des effets du type d'exploitation des meilleurs individus sur l'évolution de la meilleure *fitness*.

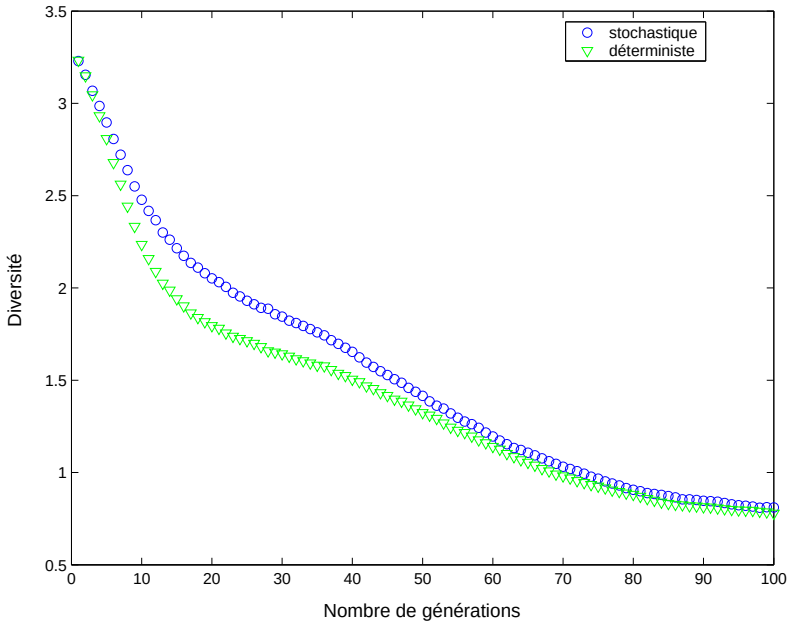


FIG. 5.8 – Etude des effets du type d'exploitation des meilleurs individus sur l'évolution de la diversité génétique.

5.2.4 Effets du partage

Dans la section 4.4, nous avons introduit une formulation de la fonction *fitness* prenant en considération la densité d'individus contenus dans le voisinage de l'élément évalué. Fondée sur la notion de *partage*, cette approche doit permettre une régulation de la densité des individus et ainsi contribuer au maintien de la diversité génétique. Nous rappelons ci-dessous l'expression de la fonction *fitness* ainsi élaborée :

$$\mathscr{F}'(X_i) = \mathscr{F}(X_i) \times \rho_{o_1}^{o_2}(r_1, r_2)^{\gamma}$$

où o_1 et o_2 correspondent aux indices des objets respectivement fréquentés par chacun des nucléons composant l'individu, et r_1 et r_2 les indices des régions fréquentées par ces mêmes entités.

Dans cette formulation, γ est un paramètre réel permettant d'ajuster l'effet du *partage* sur la fonction *fitness*. C'est sur cet élément que nous allons agir afin de caractériser l'effet du partage.

Expériences réalisées

Pour cette partie de l'étude, nous proposons une analyse visant à caractériser l'effet du paramètre γ sur le comportement de l'algorithme ESPIONS. Aussi, trois expériences servent de support à cette analyse. Dans le tableau 5.4, nous présentons les jeux de paramètres spécifiés pour chacune d'entre elles.

	$\%_R$	$\%_X$	$\%_E$	n_S	γ	p
Expérience 1	50.0	50.0	0.0	0	0.2	500
Expérience 2	50.0	50.0	0.0	0	0.6	500
Expérience 3	50.0	50.0	0.0	0	1.0	500

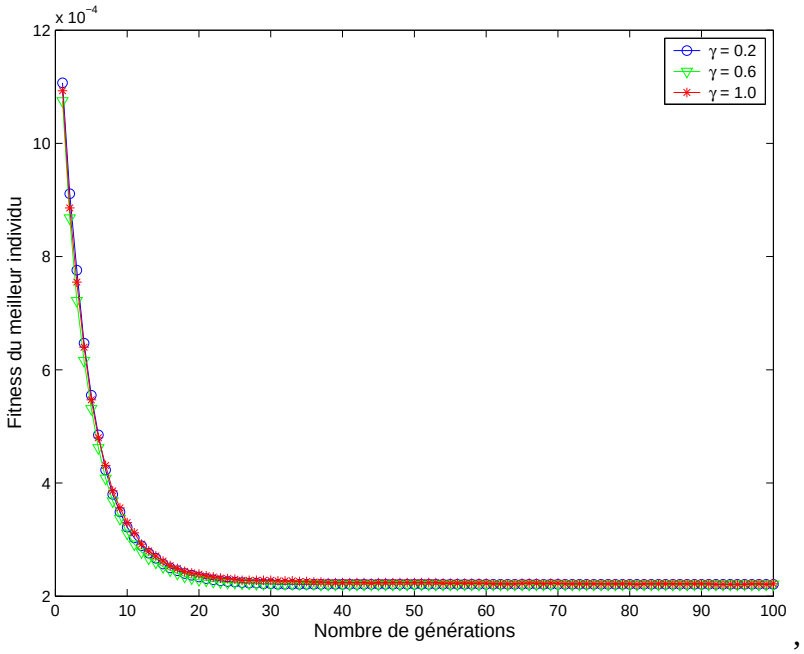
TAB. 5.4 – Paramètres utilisés pour les expériences visant à évaluer l'influence du partage.

Résultats obtenus et analyse

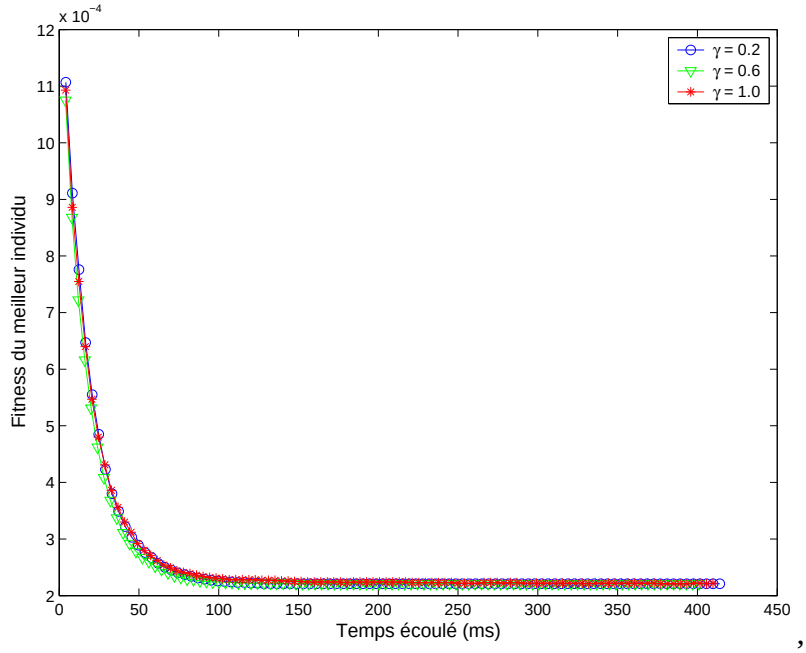
La figure 5.9 illustre les résultats obtenus à l'issue des trois expériences. En ce qui concerne la rapidité de convergence globale de l'algorithme, on peut observer que le partage ne semble pas porter préjudice à la capacité de l'algorithme à identifier les régions pertinentes au sein de l'espace de recherche. En effet, l'augmentation du paramètre γ ne génère que des variations minimes quant à la forme générale des courbes 5.9(a) et 5.9(b) décrivant l'évolution de la meilleur fitness de la population.

Par ailleurs, l'effet du partage sur la diversité génétique est sensible, comme le montre la courbe présentée sur la figure 5.10(a). Conformément à la théorie, le fait de pénaliser les individus situés sur des régions très fréquentées semble en effet contribuer à une meilleure exploration de l'ensemble de l'espace de recherche.

Cependant, on peut également constater qu'une valeur trop élevée du paramètre γ est susceptible d'entraîner la mutation malheureuse des individus représentant les meilleurs solutions (au sens global), dès lors que la densité s'avère trop importante au niveau des paires de régions qu'ils relient. Cet effet pervers du processus de partage est illustré par la figure 5.10(b), où l'on peut clairement observer, au delà d'un ralentissement de la convergence globale, un phénomène d'oscillation de la valeur de la meilleur *fitness* pour $\gamma = 1.0$.

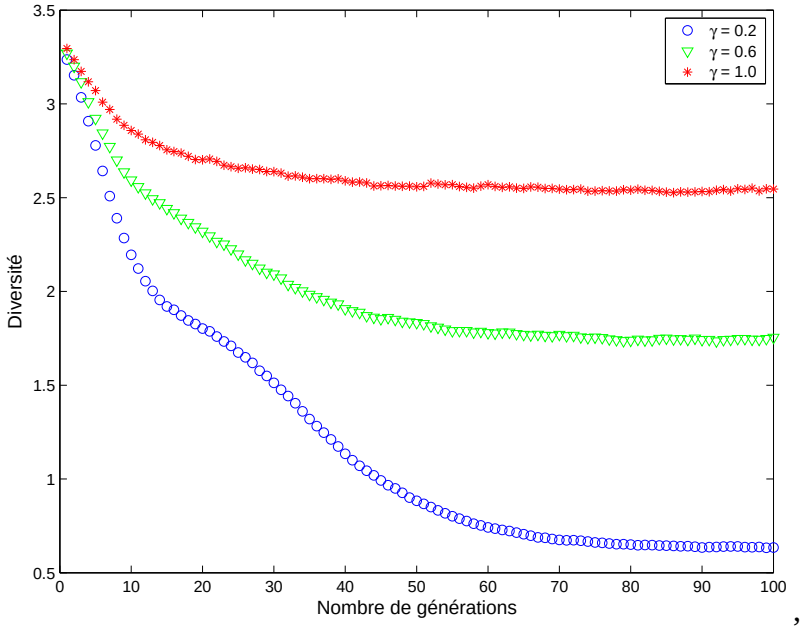


(a) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différentes valeurs de γ.

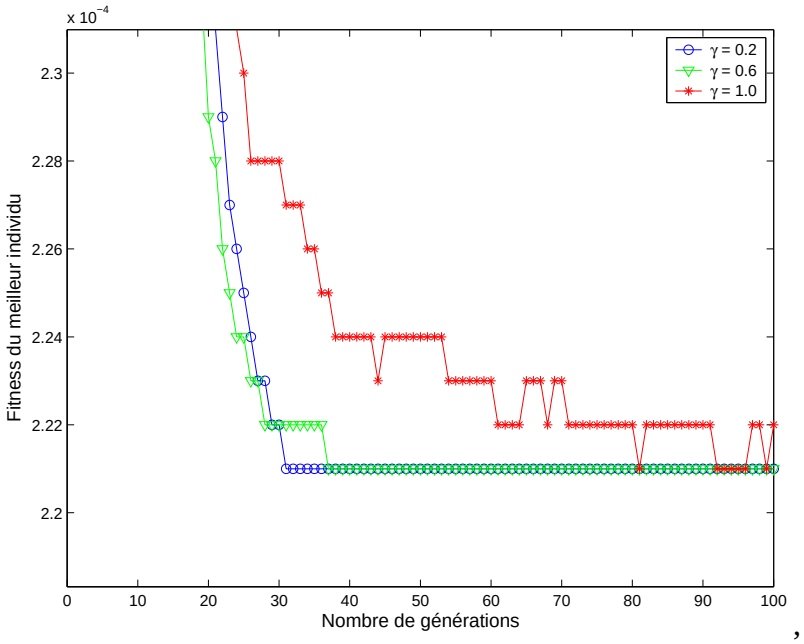


(b) Evolution de la meilleure *fitness* en fonction du temps pour différentes valeurs de γ.

FIG. 5.9 – Etude des effets du partage sur l'évolution de la meilleure *fitness*.



(a) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs de γ.



(b) Evolution de la meilleure *fitness* en fonction du nombre de générations effectuées pour différentes valeurs de γ.

FIG. 5.10 – Etude des effets du partage sur la diversité génétique.

5.2.5 Effets du nombre d'individus

Expériences réalisées

Pour cette partie de l'étude, nous proposons une analyse visant à caractériser l'effet du nombre d'individus manipulés sur le comportement de l'algorithme ESPIONS. Aussi, les résultats issus de six expériences servent de support à cette analyse. Dans le tableau 5.5, nous présentons les jeux de paramètres spécifiés pour chacune d'entre elles.

	$\%_R$	$\%_X$	$\%_E$	n_S	γ	p
Expérience 1	100.0	0.0	0.0	0	0.0	100
Expérience 2	100.0	0.0	0.0	0	0.0	500
Expérience 3	100.0	0.0	0.0	0	0.0	1000
Expérience 4	60.0	40.0	0.0	0	0.0	100
Expérience 5	60.0	40.0	0.0	0	0.0	500
Expérience 6	60.0	40.0	0.0	0	0.0	1000

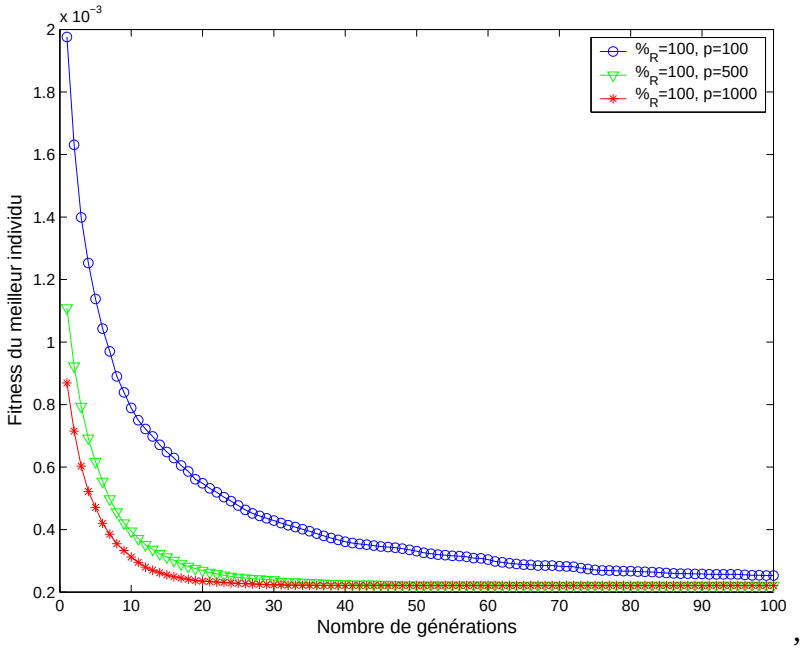
TAB. 5.5 – Paramètres utilisés pour les expériences visant à évaluer l'influence du nombre d'individus.

Résultats obtenus et analyse

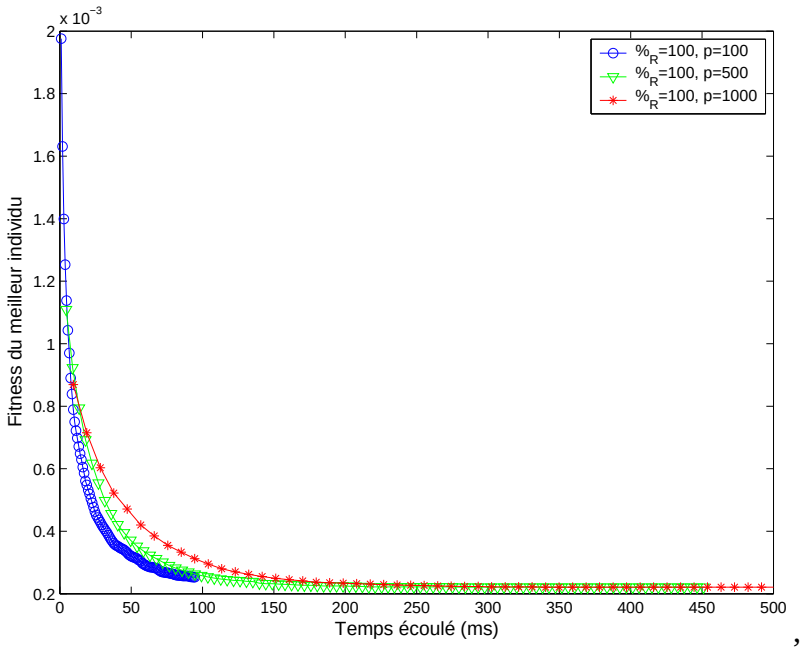
Comme on peut l'observer sur la courbe présentée en 5.11(a) le fait d'augmenter la quantité d'individus constituant la population réduit de façon drastique le nombre de générations nécessaires à la convergence de l'algorithme. Ce phénomène découle d'un postulat simple : Plus on est à chercher, plus on a de chances de trouver ce que l'on cherche. Cependant, le fait d'augmenter le nombre d'individus influe directement sur la durée d'une génération, la charge de traitement nécessaire à la manipulation des individus augmentant par la même (plus on est à chercher, plus on est à manger...). En observant la courbe 5.11(b), on observe en effet que, lorsque l'on raisonne en termes de temps, il apparaît que la convergence globale de l'algorithme ne semble pas être conditionnée par le nombre d'individus (on ne parle pas là de la capacité de l'algorithme à converger vers les bonnes solutions...).

En ce qui concerne l'évolution de la diversité génétique, l'étude des courbes présentées sur la figure 5.12 permet de constater l'influence sensible du nombre d'individus sur la capacité de l'algorithme à explorer l'ensemble de l'espace de recherche. Ce constat paraît naturel, l'augmentation du nombre de pions permettant la prise en compte d'une plus large partie de la combinatoire que constitue l'espace de recherche. Cependant, comme

on peut l'observer sur la courbe 5.12(b), le fait de stimuler la convergence de l'algorithme en augmentant la proportion de croisements entraîne inexorablement une diminution de la diversité, même dans le cas où les individus s'avèrent plus nombreux. Aussi, l'augmentation du nombre d'individus n'exclue pas un maintien artificiel de la diversité génétique tel que celui que nous proposons avec le partage des ressources présenté précédemment.

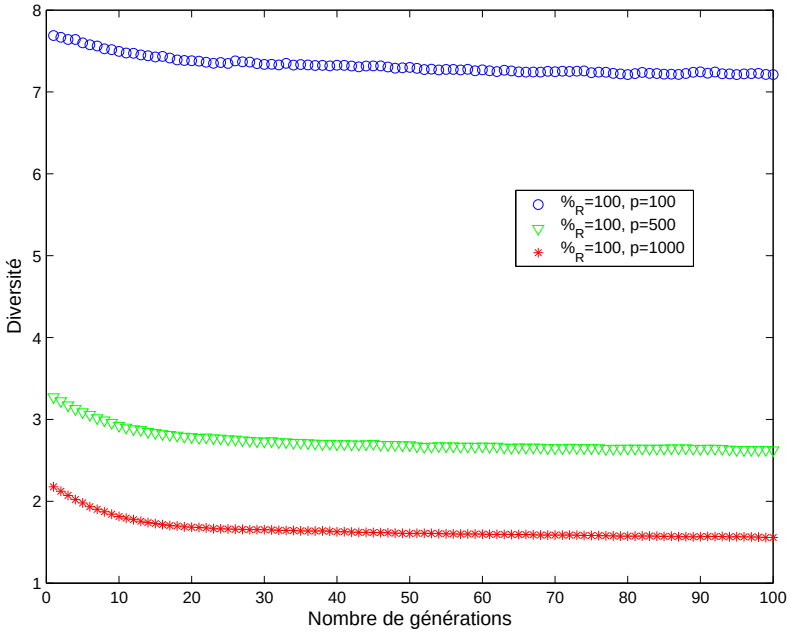


(a) Evolution de la meilleure *fitness* en fonction du nombre de générations réalisées pour différentes valeurs de *p*, le nombre d'individus contenus dans la population.

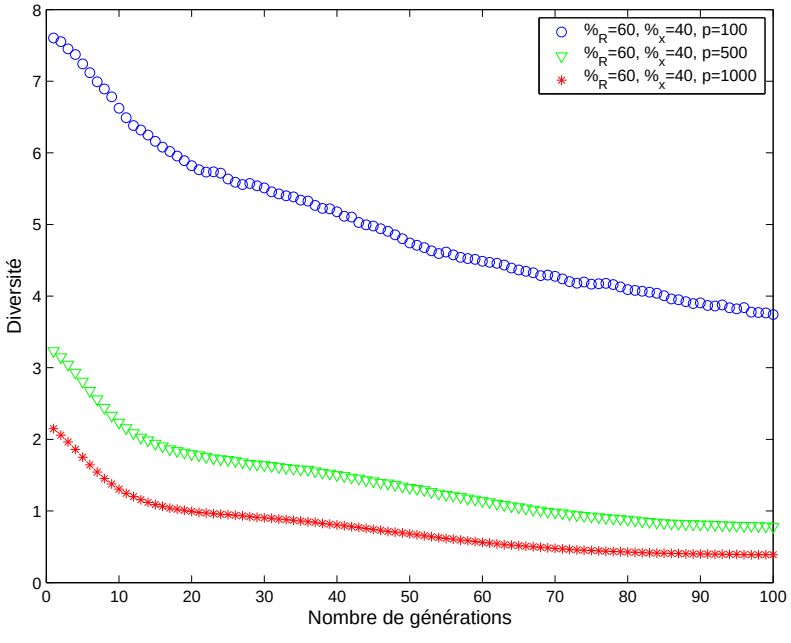


(b) Evolution de la meilleure *fitness* en fonction du temps pour différentes valeurs de *p*, le nombre d'individus contenus dans la population.

FIG. 5.11 – Etude de l'effet du nombre d'individus sur l'évolution de la meilleure *fitness*.



(a) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs de p, avec $\%_R = 100.0$.



(b) Evolution de la diversité génétique en fonction du nombre de générations effectuées pour différentes valeurs de p, avec $\%_R = 60.0$ et $\%_X = 40.0$.

FIG. 5.12 – Etude de l'effet du nombre d'individus sur la diversité génétique.

5.2.6 Discussion et conclusions

Dans cette section, nous avons proposé une étude des effets des différents paramètres d'entrée d'ESPIONS sur son comportement général. Cette caractérisation des *leviers* mis à disposition quant à la gestion des tendances de l'algorithme s'est fondée sur l'analyse de deux caractéristiques de celui-ci que sont la rapidité de convergence et la capacité à explorer l'ensemble de l'espace de recherche considéré. Dans ce contexte, les résultats issus d'une série d'expériences nous ont permis d'apprécier l'influence des paramètres sur les propriétés de l'algorithme.

Cependant, nous n'avons pas répondu à l'ultime question qui est la suivante : *Comment paramétrer ESPIONS ?*

En effet, même si à l'issue de cette analyse nous sommes désormais en mesure de définir l'influence des paramètres de l'algorithme, nous ne sommes pas capables d'identifier précisément un jeu de paramètre idéal au vu de la définition géométrique d'une scène donnée. Certes, les paramètres répondant à certains cas triviaux sont désormais intuitivement identifiables ; Dans le cas d'une scène convexe, par exemple, les éléments précédemment observés nous permettent d'assurer qu'une utilisation massive des opérateurs de croisement (i.e. $\%_X \approx 100.0$) permettra une convergence très rapide de l'algorithme, l'absence de minima locaux permettant de négliger le maintien de la diversité génétique. Mais rappelons que dans l'immense majorité des cas qui nous intéressent, les objets simulés sont de formes quelconques, voir variables dans le cas de simulations mettant en scène des objets déformables.

Dans la pratique, l'établissement de jeux de paramètres adaptés aux problèmes géométriques abordés est actuellement réalisé de façon empirique. Pour ce faire, nous nous basons sur notre connaissance de l'influence de ces paramètres sur les caractéristiques de l'algorithme. Dans le tableau 5.6, nous résumons ces informations issues de l'étude expérimentale présentée précédemment. On peut y lire l'effet de l'augmentation de chacun des paramètres sur les caractéristiques de l'algorithme. Aussi, les signes + et - expriment respectivement une amélioration ou une détérioration de la caractéristique considérée, le signe 0 indique quant à lui une influence négligeable.

Comme nous le présenterons dans le chapitre présentant les conclusions et perspectives, l'une des suites logiques à nos travaux devrait consister en l'élaboration d'heuristiques permettant une identification des paramètres optimaux pour une situation donnée. Ces processus seraient très certainement fondés sur plusieurs critères inhérents à la tâche à réaliser : définition des géométries, complexité de la scène, nature des interactions entre l'utilisateur et l'environnement simulé (niveau de réalisme escompté, type d'interfaces

	Vitesse de convergence	Diversité génétique	Temps de calcul
$\%_R$	- -	+ +	0
$\%_X$	+ +	- -	+
$\%_E$	-	-	0
n_S	0	0	-
Exploitation déterministe	+ +	0	- -
Exploitation stochastique	- -	0	+ +
γ	-	+ +	0
p	+	+	-

TAB. 5.6 – Synthèse de l'influence des paramètres d'ESPIONS sur ses caractéristiques.

utilisées), capacité de calcul de la plateforme sur laquelle la simulation est réalisée, quantité de mémoire vive allouée au processus, etc.

5.3 Couplage d'ESPIONS avec un module déterministe de détection des collisions

Dans cette section, nous décrivons le mode opératoire adopté quant à l'intégration de notre algorithme au sein d'un module visant à détecter des collisions dans le cadre de simulations interactives mettant en scène des objets polyédriques. Pour ce faire, nous proposons dans un premier temps une description générale de l'algorithme avec lequel le couplage doit être réalisé. Puis, nous présentons les modes opératoires adoptés quant à l'intégration d'ESPIONS au sein du processus. Enfin, nous exposons plusieurs résultats obtenus dans le cadre d'expérimentations.

5.3.1 Le module client : LMD++

Le module de détection de collisions auquel nous nous intéressons dans cette section, nommé LMD++, est fondé sur une approche visant à déterminer, selon différents critères, des champs de distances minimum locales séparant les objets simulés. Il a été développé au sein du CEA-LIST afin de répondre à des besoins inhérents à la simulation interactives, besoins exprimés par certains partenaires industriels désireux d'intégrer les technologies de la réalité virtuelle dans leurs cycles de conception. Bien que plusieurs composantes algorithmiques originales, non abordées dans ce mémoire, aient été introduites lors de son élaboration, ce module est fondé sur une implémentation des approches proposées par Johnson et al. dans [JC01, JW04]. Dans cette partie, nous introduisons les concepts fondamentaux inhérents à ces travaux.

Décomposition des objets en hiérarchies de volumes englobants

Dans l'état de l'art dédié aux approches inhérentes à l'accélération de la détections des collisions, nous avons introduit un type de techniques fondées sur la décomposition des objets simulés en des hiérarchies de volumes englobants (Cf. paragraphe 2.2.4). L'algorithme auquel nous nous intéressons ici utilise une telle approche. En effet, chacun des objets traités par le module se voit décomposé en une hiérarchie de sphères englobantes, et le processus de détection des collisions va consister en une descente récursive sur chaque paire de hiérarchies.

Détermination des distances minimum locales au niveau des feuilles de la hiérarchie

Dans ce contexte, chaque feuille de la hiérarchie contient une entité géométrique élémentaire. On entend ici par entité élémentaire : un triangle, une arête ou un sommet.

L'identification d'une distance minimum locale entre des éléments de cette nature est fondé sur deux critères :

- La distance séparant les deux entités :
  Le segment minimum séparant deux entités élémentaires est assimilable à une distance minimum locale si sa taille est inférieure à un seuil de proximité (paramètre utilisateur).
- L'orientation des entités :
  Ce critère vise à s'assurer que chacune des extrémités du segment matérialisant la distance minimum séparant les entités élémentaires considérées s'avère être plus proche de l'entité portant l'autre extrémité que de toute autre entité de la géométrie qui contient cette dernière. Sans décrire la procédure permettant d'établir le respect de ce critère, précisons tout de même que cette dernière se base sur les normales aux triangles (Cf. [JC01]). On retrouve là la notion de *domaines de Voronoï* introduite dans l'état de l'art dédié à la détection des collisions (Cf. paragraphe 2.4.2).

Introduction de cônes de normales au sein de la hiérarchie

Dans [JC01], les auteurs proposent d'enrichir l'information *spatiale* contenue dans chaque nœud de la hiérarchie (i.e. les tests d'intersection sphère/sphère permettent d'identifier si les éléments géométriques qu'elles contiennent sont susceptibles d'être proches), d'une information visant à prendre en considération l'orientation (au sens normal) des entités géométriques englobées. Cette approche consiste en la définition de cônes de normales décrivant, pour chaque nœud de la hiérarchie, la zone de l'espace contenant l'ensemble des éléments susceptibles de former une distance minimum locale avec l'une des entités géométriques qu'il contient (Voronoï marching). Ainsi, lors de la détection des collisions entre deux objets, la poursuite de la descente de leurs hiérarchies respectives à partir d'une paire de nœuds donnée n'est plus seulement conditionnée par un test d'intersection entre deux sphères, mais également par un second test d'intersection entre deux cônes.

5.3.2 Fonctionnement du couplage LMD++ / ESPIONS

Justification et principe de base

Comme expliqué précédemment, la procédure de détection de collision réalisée par LMD++, qui vise rappelons-le à identifier un éventuel champ de distances minimum locales séparant les objets, consiste d'une part en une descente récursive des hiérarchies de volumes englobants correspondant aux objets considérés, et, d'autre part, en la détermina-

tion d'éventuelles distances minimum locales séparant les entités géométriques élémentaires présentes au niveau des feuilles de la hiérarchie atteintes lors de la descente. Bien que les tests inhérents à la poursuite de la recherche à partir d'une paire de nœuds permettent d'exclure une partie importante de la combinatoire, on observe dans bon nombre de cas une augmentation importante du temps nécessaire à l'algorithme pour réaliser la tâche qui lui incombe. Ce phénomène est observable dès lors que les géométries considérées s'avèrent de grande complexité et/ou que les objets présentent de nombreuses zones de proximité (on parle de contacts conformant).

Aussi, l'implémentation décrite dans cette section vise à réduire la combinatoire traitée par LMD++ en identifiant, à l'aide d'ESPIONS, les zones de proximité susceptibles de porter des distances minimum locales. Aussi, le couplage va consister à fournir à LMD++ des paires de nœuds à partir desquelles les descentes doivent être réalisées. Les descentes ne sont alors plus effectuées en partant des nœuds racines des hiérarchies, mais à partir de descendants plus ou moins lointains de ces derniers, réduisant ainsi la combinatoire à explorer.

Phase d'initialisation

Afin de réaliser ce couplage, il est nécessaire d'établir un lien entre les zones fréquentées par les nucléons et les nœuds des hiérarchies. Pour ce faire, une étape de pré-calculs va consister à identifier les indices des nœuds correspondant aux sphères englobantes délimitant des portions composant les objets pour un seuil de taille donné. Ce processus est décrit dans l'algorithme 4, déjà présenté précédemment dans le cadre de l'établissement de la carte de densité (Cf. paragraphe 4.4).

Algorithme 4 (Procédure d'extraction des régions sur une hiérarchie de sphères englobantes)

ExtractRegionsFromNode *(node, threshold, regionsList)*
(1) si *(node* $\rightarrow$ *radius* $<$ *threshold* ***ou*** *node* $\rightarrow$ *children* $= 0$*)*
regionsList $\leftarrow$ *node*
terminer
(2) sinon
ExtractRegionsFromNode *(node* $\rightarrow$ *childRight, threshold, regionsList)*
ExtractRegionsFromNode *(node* $\rightarrow$ *childLeft, threshold, regionsList)*
terminer

ExtractBSTreeRegions *(tree, threshold, regionsList)*
(1) Empty*(regionsList)*

(2) ExtractRegionsFromNode *(tree → rootNode, threshold, regionsList)*

La fonction *ExtractBSTreeRegions* permet l'extraction à partir de la hiérarchie *tree* des indices des nœuds dont les sphères englobantes présentent un rayon inférieur à *threshold* (noté τ dans la suite). Pour cela, elle fait appelle à une fonction récursive *ExtractRegionsFromNode* permettant une exploration en profondeur de la hiérarchie considérée. Bien entendu, seules les sphères de plus haut niveau répondant à ce critère sont retenues. Nous entendons par là que les sphères contenues dans celles-ci ne sont pas ajoutées à la liste.

Une fois établie la liste des régions (et par conséquent des nœuds) dont ESPIONS va chercher à évaluer la proximité, un marquage des triangles va être réalisé afin d'être en mesure d'identifier, pour un nucléon donné, l'indice du nœud correspondant à la région qu'il occupe. La figure 5.13 illustre le résultat de ce processus.

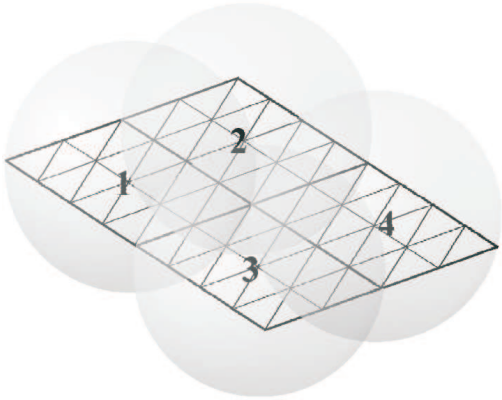


FIG. 5.13 – Exemple de marquage des primitives composant un objet en fonction des nœuds auxquels ils correspondent au sein de la hiérarchie de volumes englobants.

Schéma de fonctionnement et nature des informations échangées

Comme expliqué précédemment, dans le cadre du couplage qui nous intéresse ici, la tâche d'ESPIONS va consister à fournir à LMD++ des paires d'indices correspondant aux nœuds des hiérarchies de volumes englobants à partir desquels les descentes doivent être réalisées. Aussi, un jeu de quatre indices est nécessaire à la description de chaque paire de nœuds. Ci-dessous nous donnons l'expression pour une telle entité :

$$paire_k = \left\{ \begin{array}{c} O1_{id} \\ N1_{id} \\ O2_{id} \\ N2_{id} \end{array} \right\} \tag{5.1}$$

où $N1_{id}$ et $N2_{id}$ donnent les indices des nœuds à considérer au sein des hiérarchies correspondant aux objets ayant respectivement pour indice $O1_{id}$ et $O2_{id}$.

En ce qui concerne le mode de couplage adopté, les deux schémas d'intégration introduits dans 4.5.3 ont été implémentés afin de permettre une validation de chacun d'entre eux. Les expérimentations proposées dans la partie de cette section dédiée à la présentation des résultats obtenus permettrons d'apprécier la validité de ces deux modes opératoires. La figure 5.14 illustre l'architecture mise en œuvre dans le cadre de l'implémentation parallèle du couplage.

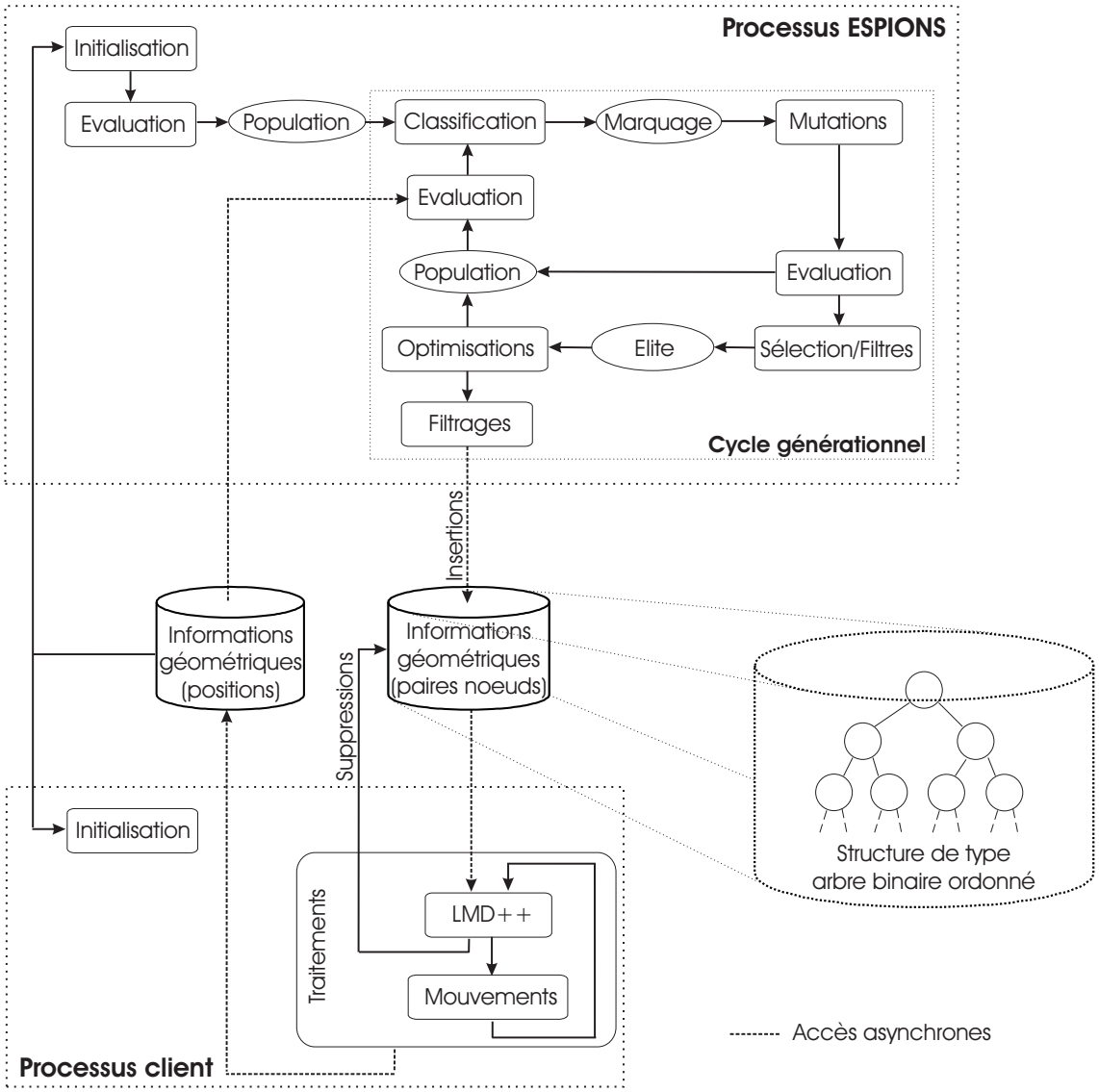


FIG. 5.14 – Schéma présentant l'architecture mise en œuvre dans le cadre du couplage LMD++ / ESPIONS en fonctionnement parallèle.

Il est à noter que, contrairement au schéma parallèle proposé dans un cadre général, le mode opératoire proposé ici permet à LMD++ d'agir sur l'espace mémoire partagé destiné au stockage des informations issues d'ESPIONS. Cette spécificité se justifie par le fait que, dans le cadre de l'implémentation proposée ici, seul LMD++ est apte à juger de la réelle pertinence des paires de nœuds proposées par ESPIONS. En effet, comme il le sera expliqué dans la suite, l'identification des zones potentiellement intéressantes réalisée par ESPIONS est fondée sur des critères ne permettant pas de garantir que des distances minimum locales sont en effet portées par ces dernières. Aussi, c'est au module LMD++ qu'incombe la tâche d'éliminer les paires de nœuds inintéressantes.

Pour ce faire, et afin de garantir un niveau de performances optimal, nous avons choisi de stocker les paires fournies par ESPIONS dans une structure de données de type *arbre binaire ordonné*. Ce choix se justifie par le fait que ce type de structure permet de réduire le temps nécessaire aux opérations visant à l'insertion et à la suppression de nouveaux éléments, en comparaison avec d'autres types de formalismes tels que les listes chaînées par exemple. Il permet également une élimination implicite des doublons. L'implémentation d'un tel type de structure nécessite de mettre en place un ou plusieurs critères permettant de comparer les éléments à stocker, cela afin de rendre possible un ordonnancement de ces derniers. Aussi, nous avons mis en place un opérateur de comparaison simple, basé sur les valeurs des éléments définissant chaque paire (voir expression 5.1).

Identification des zones de proximité par ESPIONS

Dans la section 4.5.2, nous avons décrit l'ensemble des traitements réalisés dans le cadre de l'étape du cycle générationnel dite *sélection/Filtres*. Dans ce contexte, nous avons introduit entre autre une opération de filtrage visant à exclure de l'élite les individus présentant une *fitness* supérieure à un seuil noté ζ (i.e. les individus trop grands pour être intéressants). Ce paramètre va permettre, dans le cadre de ce couplage, de définir la distance au delà de laquelle deux régions sont considérées comme trop éloignées pour justifier une prise en considération de la paire de nœuds à laquelle elles correspondent au sein de la hiérarchie lors du processus de détection des collisions réalisé par LMD++.

En ce qui concerne la dernière étape dite de *filtrages*, cette dernière va simplement consister en une élimination des doublons générés par la procédure d'*optimisation* réalisée en amont. Notons que nous entendons là par doublons les individus reliant les mêmes régions (i.e. décrivant les mêmes paires de nœuds). Comme expliqué dans la section 4.5.2, une formalisation de la sortie d'ESPIONS est également réalisée à ce niveau du processus. Celle-ci va consister à identifier, pour chacun des individus ayant franchi l'ensemble des

procédures de filtrage, la paire de nœuds à laquelle il correspond soit :

$$paire(X_k) = \left\{ \begin{array}{c} G_{id}(\mu_1^k) \\ trianglesRegionsArray[T_{id}(\mu_1^k)] \\ G_{id}(\mu_2^k) \\ trianglesRegionsArray[T_{id}(\mu_2^k)] \end{array} \right\} \tag{5.2}$$

5.3.3 Résultats obtenus

Dans cette partie, nous exposons une partie des résultats obtenus dans le cadre d'expérimentations du couplage LMD++ / ESPIONS introduit précédemment. Pour ce faire, nous comparons les niveaux de performances obtenus avec et sans ESPIONS dans le cadre de simulations consistant à mesurer les temps nécessaires à la détermination des champs de distances minimum locales tout au long de trajectoires préalablement définies. Les données utilisées pour la définition géométrique des objets constituant les scènes sont directement issues d'assemblages CAO industriels fournis par Renault, PSA Peugeot Citroën et HAPTION.

Expérience 1 : Couplage parallèle LMD++ / ESPIONS

Nous présentons ici les résultats obtenus dans le cadre d'une expérimentation de l'implémentation parallèle du couplage LMD++ / ESPIONS. Comme expliqué précédemment, l'expérience a consisté à mesurer les temps nécessaires à l'identification des champs de distance minimum tout au long d'une trajectoire, dans un premier temps en utilisant exclusivement LMD++, puis, dans un deuxième temps en exploitant l'algorithme ESPIONS. La figure 5.15 illustre la scène simulée. Elle est composée de deux objets polyédriques respectivement constitués de 15000 et 60000 triangles. En ce qui concerne la plateforme utilisée dans le cadre de cette expérience, elle répondait au caractéristiques suivantes :

- Unité de calcul : Mono-processeur intel bi-cœurs cadencé à 3.6 GHz.
- Mémoire vive : 2.0 Go type DDR2.
- Système d'exploitation : Windows XP.

Les paramètres spécifiés pour le module ESPIONS dans le cadre de cette expérimentation sont présentés dans le tableau 5.7. Ces derniers, obtenus de façon empirique, garantissent l'aptitude d'ESPIONS à identifier l'ensemble des zones de proximité tout au long de la trajectoire (i.e. le nombre de distances minimum locales trouvées est le même avec et sans utilisation d'ESPIONS).

La figure 5.16 illustre les résultats expérimentaux obtenus dans le cadre de cette expérimentation du couplage LMD++ / ESPIONS. Précisons que le seuil spécifié à LMD++

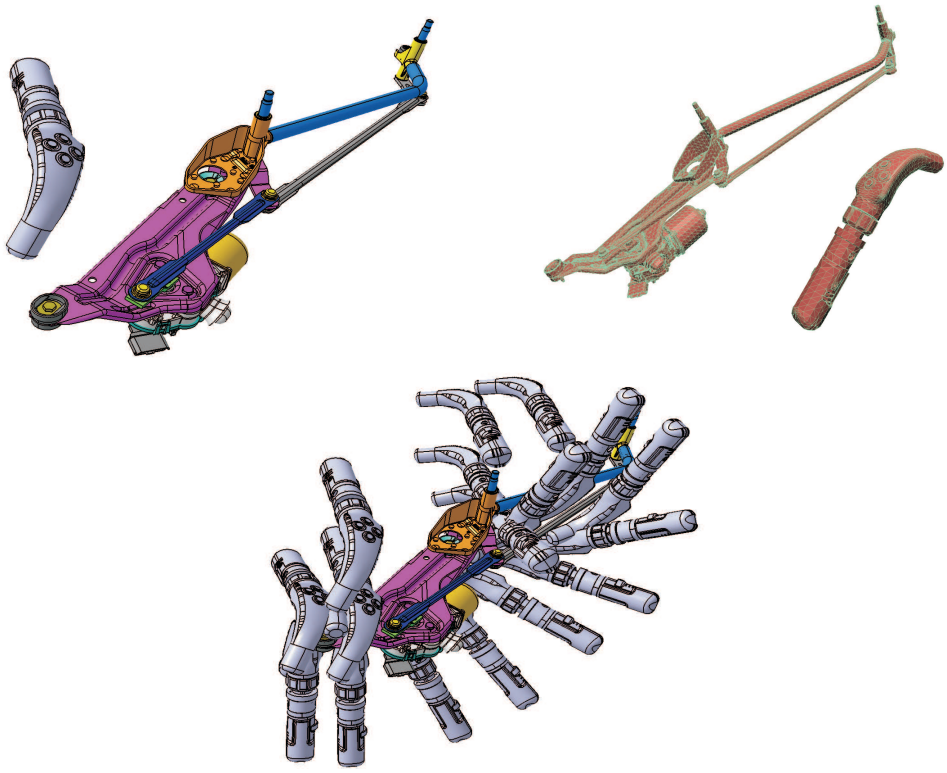


FIG. 5.15 – Expérimentation du couplage LMD++ / ESPIONS en mode parallèle. Les objets constituant la scène sont une poignet de *VIRTUOSE* (*HAPTION*), et un mécanisme d'entraînement des essuie-glace issu d'une *1007* (*PSA*). Leur représentations polyédriques contiennent respectivement 15000 et 60000 triangles (En haut à droite). (En haut à gauche) La scène en configuration initiale. (En bas) Une illustration de la trajectoire réalisée dans le cadre de l'expérimentation proposée.

quant à la taille maximum d'une distance minimum local était de 5*mm* (Cf. paragraphe 5.3.1). On observe une réduction sensible du temps nécessaire à la détermination des champs de distances minimum locales lorsque qu'ESPIONS est mis en œuvre. Le tableau 5.8 propose une petite synthèse de ces résultats.

Au delà d'une réduction sensible des temps dans le cas où le couplage est mis en œuvre, on peut constater sur la figure 5.17 que, conformément à la vocation du mode parallèle, le fait de créer un thread dédié à ESPIONS engendre une répartition du processus sur les deux cœurs qui composent l'unité de calcul. Cette répartition explique que le couplage n'engendre pas de coût supplémentaire en situation de non-collision.

$\%_R$	$\%_X$	$\%_E$	n_S	γ	p	τ	ζ
30.0	60.0	10.0	5	0.4	2000	17mm	7mm

TAB. 5.7 – Paramètres spécifiés dans le cadre du couplage parallèle LMD++ / ESPIONS.

	Temps moyen	maximum	minimum
LMD++ seul	10.53ms	41.45ms	33.0μs
LMD++ / ESPIONS	2.71ms	9.80ms	13.0μs

TAB. 5.8 – Synthèse des résultats obtenus dans le cadre du couplage parallèle LMD++ / ESPIONS.

Expérience 2 : Couplage série LMD++ / ESPIONS avec montée en charge

Dans cette partie, nous proposons un exposé des résultats obtenus dans le cadre d'un *couplage de type série* réalisé sur une machine munie d'un unique processeur modèle *Intel Pentium 4* cadencé à 2.4GHz. Contrairement à l'expérience précédente, qui consistait en une étude de l'évolution du temps de calcul tout au long d'une trajectoire, nous proposons ici de comparer les temps moyens nécessaires à l'identification des champs de distances sur l'ensemble d'une même trajectoire pour différents niveaux de résolution des géométries mises en scène. Ce mode opératoire, que nous qualifierons de montée en charge, a pour but d'évaluer l'effet de l'utilisation d'ESPIONS pour différents niveaux de complexité de la scène considérée.

Les données utilisées dans le cadre de cette expérience nous ont été fournies par la société *Renault*. Elles sont issues d'un véhicule de type *Logan*. L'objet mobile correspond au mécanisme d'entraînement des essuie-glaces. Le reste de l'environnement est constitué des deux ailes avant, de la gouttière et du capot de la voiture. Les représentations polyédriques des objets sont obtenues par tessellation de ces derniers. Un des paramètres du processus de tessellation, assimilable à une erreur de corde, permet la générations des modèles à différents niveaux de résolution (i.e. une diminution de l'erreur de corde tolérée entraîne une augmentation du nombre de triangles nécessaires à la définition de l'objet). La figure 5.18 propose une illustration de la scène et de la trajectoire réalisée lors des simulations.

Les paramètres spécifiés pour le module ESPIONS dans le cadre de cette expérimentation sont présentés dans le tableau 5.9. Comme dans le cadre de l'expérience précédente, ces derniers ont été obtenus de façon empirique et garantissent l'aptitude d'ESPIONS à identifier l'ensemble des zones de proximité tout au long de la trajectoire *pour tous les*

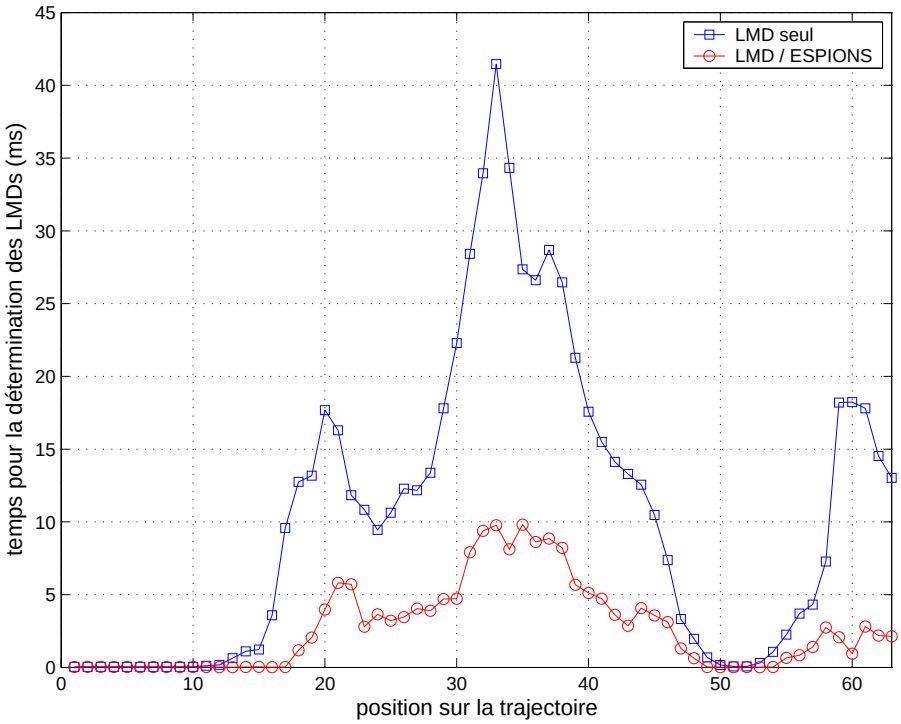


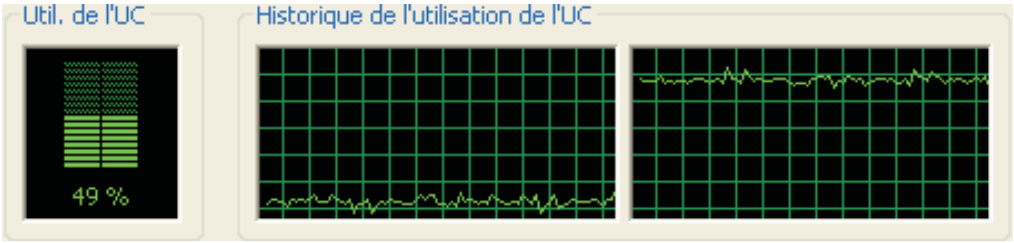
FIG. 5.16 – Expérimentation du couplage LMD++ / ESPIONS en mode parallèle : Résultats expérimentaux. Temps nécessaires à l'identification des distances minimum locales avec et sans utilisation d'ESPIONS. Les temps sont donnés en fonction de la position de l'objet mobile sur la trajectoire. Pour chaque position, la valeur lue en ordonnée correspond à la moyenne des temps de calcul mesurés au cours de l'intervalle précédent.

niveau de résolution proposés (i.e. le nombre de distances minimum locales trouvées est le même avec et sans utilisation d'ESPIONS). Il est à noter que ces valeurs sont pénalisantes dans le cas des faibles niveaux de résolution, particulièrement pour τ et ζ (i.e. de meilleurs performances pourraient être atteintes avec un paramétrage adapté). Mais nous avons choisi, dans le cadre de cette expérience, de découpler les problématiques que sont : le paramétrage d'ESPIONS et son effet sur la complexité du processus global (i.e. LMD++ / ESPIONS).

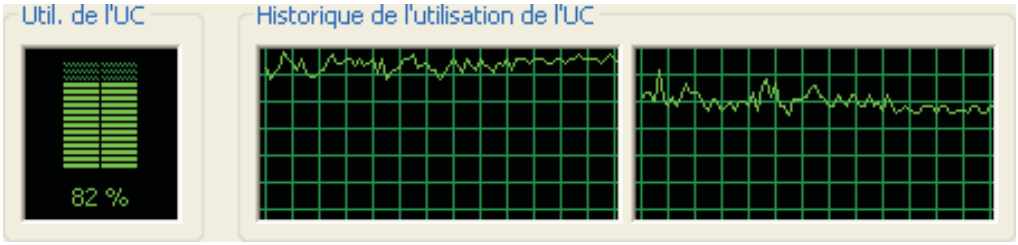
n_G	$\%_R$	$\%_X$	$\%_E$	n_S	γ	p	τ	ζ
8	40.0	60.0	0.0	0	0.0	1000	40*mm*	26*mm*

TAB. 5.9 – Paramètres spécifiés dans le cadre du couplage série LMD++ / ESPIONS.

La figure 5.19 illustre les résultats obtenus dans le cadre de cette expérience. On observe que, au delà d'une certaine complexité, l'utilisation d'ESPIONS engendre une



(a) LMD++ seul : Seul l'unité de calcul de droite est sollicitée.



(b) LMD++ / ESPIONS : le thread ESPIONS s'exécute sur l'unité de gauche, laissant l'autre unité disponible pour la réalisation des traitements inhérents à LMD++

FIG. 5.17 – Etude de l'effet de la parallélisation sur l'exploitation des unités de calcul.

diminution importante du temps nécessaire à l'identification des champs de distances minimum locales. Le fait que les performances observables en dessous d'une certaine complexité s'avèrent moins bonnes dans le cadre d'une utilisation d'ESPIONS découle directement du type de couplage mis en œuvre dans cette expérience. En effet, contrairement au cas parallèle présenté précédemment, le coût de calcul inhérent à ESPIONS est désormais attribué à la même unité de calcul que celle sollicitée par LMD++. Ce phénomène engendre une sollicitation importante du processeur et peut s'avèrer pénaliser les performances du processus globale. Notons tout de même que, comme il l'a été expliqué précédemment, les paramètres d'ESPIONS pour cette expérience ne sont pas optimaux dans le cadre des simulations mettant en scène des objets de faible complexité. Cependant, il est légitime d'affirmer que, dans un cadre générale, le fait d'utiliser ESPIONS selon un schéma série peut être pénalisant pour la simulation de scènes de faible complexité.

Au delà du seuil de complexité dont nous venons de traiter, on observe un gain de performances important lors de l'utilisation d'ESPIONS. La courbe 5.19(b) permet d'apprécier ce phénomène. Elle illustre l'évolution du pourcentage de temps économisé pour chaque niveau de complexité. On observe que, dans le cas où la complexité s'avère maxi-

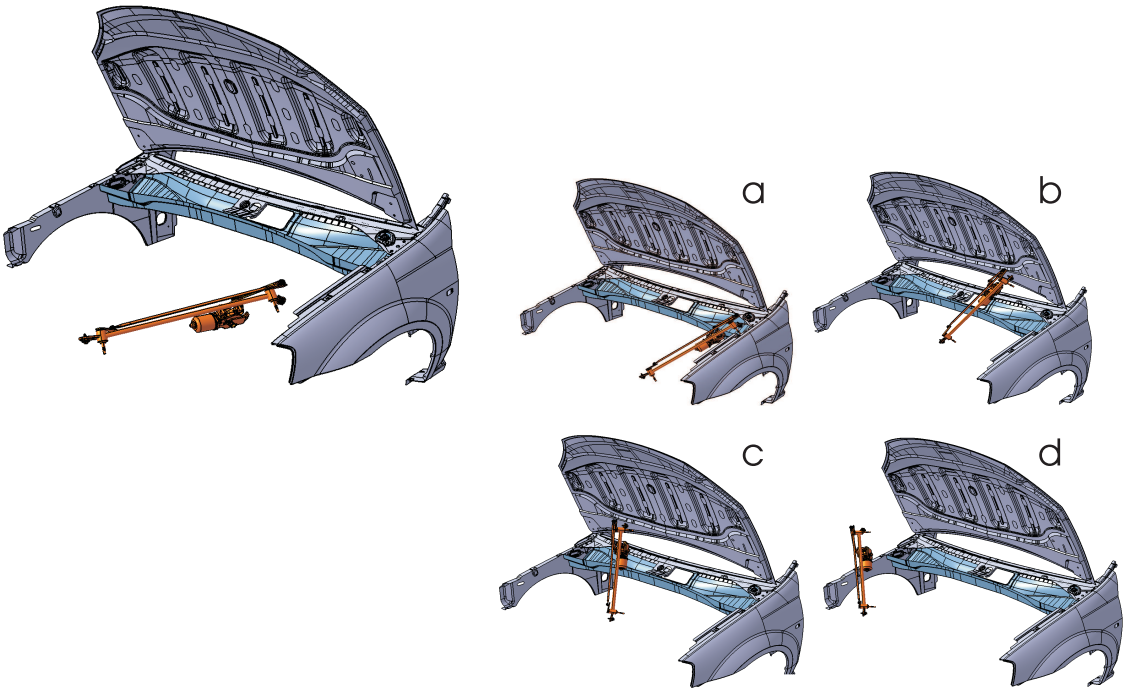


FIG. 5.18 – L'objet mobile correspond au mécanisme d'entraînement des essuie-glaces. Le reste de l'environnement est constitué des deux ailes avant, de la gouttière et du capot de la voiture. Ces données, fournies par *Renault*, sont issues d'une *Logan*. Les quatres images de droite illustrent la trajectoire considérée.

mum (environ 1 200 000 triangles), le gain est de l'ordre de 70%. De plus, à la lecture des courbes présentées sur la figure 5.19(a), on observe que, contrairement au cas LMD++ seul, où la complexité du processus évolue de façon exponentielle, le couplage LMD++ / ESPIONS présente une augmentation du temps de calcul quasi-linéaire.

5.3.4 Discussion

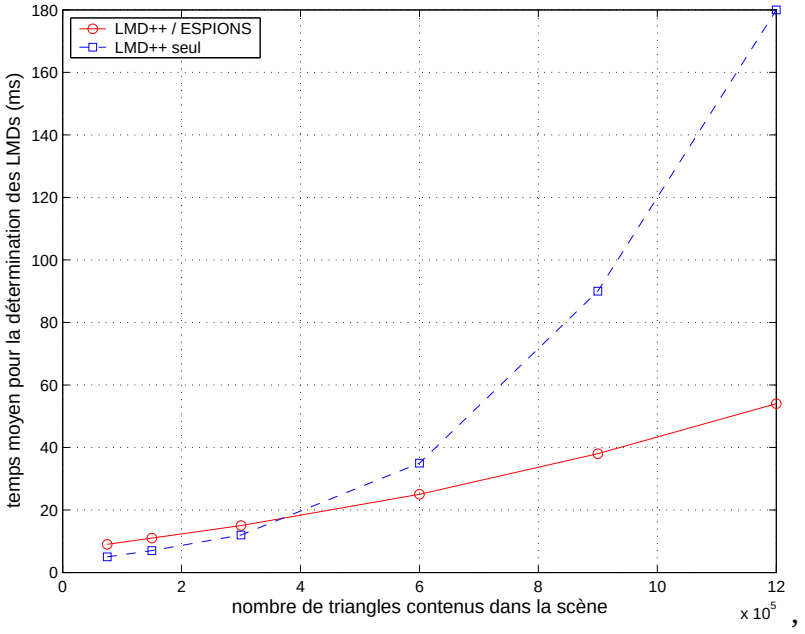
Comme il l'a été introduit précédemment, un des problèmes inhérents à l'utilisation d'algorithmes fondés sur des approches stochastiques, tel qu'ESPIONS, réside dans l'incertitude de trouver, dans la limite du temps imparti, une solution répondant au problème que l'on vise à résoudre. Ce phénomène découle du caractère probabiliste de ces techniques qui sont fondées sur une exploration partielle de la combinatoire. Dans le cadre de l'implémentation à laquelle nous nous intéressons dans cette section, cette problématique nous amène naturellement à la question suivante :

Comment garantir qu'ESPIONS a identifié l'ensemble des zones de proximité néces-

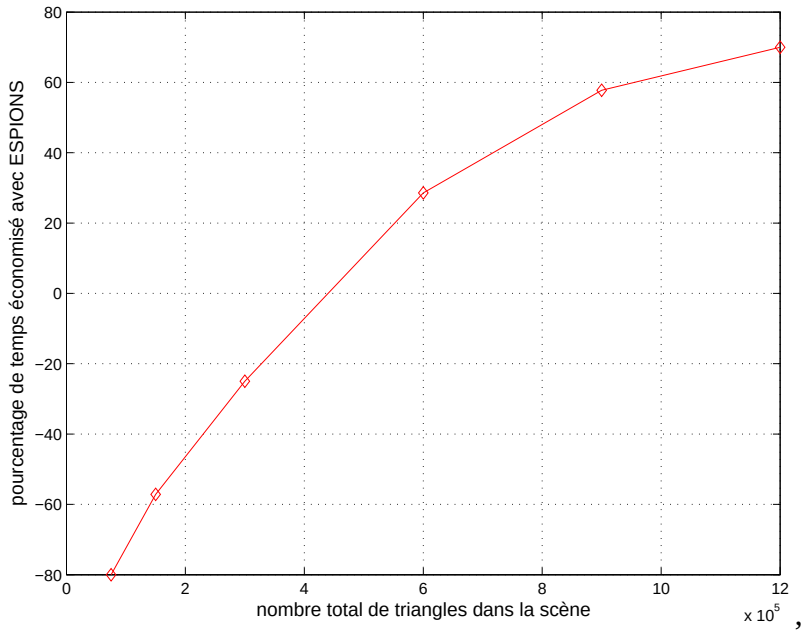
saires à LMD++ pour calculer l'intégralité des distances minimum locales existantes pour une configuration donnée ?

A cette question, il existe une réponse simple : *Il n'est pas possible d'apporter une telle garantie*.

Dans la première partie de ce chapitre, nous avons mis en évidence la notion de compromis entre temps de convergence et maintien de la diversité génétique. Dans le cas qui nous intéresse ici, la bonne gestion de ce compromis s'avère fondamentale. Aussi, la mise au point d'un paramétrage permettant une bonne gestion de cet équilibre pour un problème donné s'avère complexe, et nécessite bien souvent une démarche empirique visant à identifier un jeu de paramètres satisfaisant.



(a) Temps moyen nécessaire à l'identification des champs de distances minimum locales sur l'ensemble de la trajectoire en fonction du niveau de résolution de la scène.



(b) Gains observé, dans le cas où ESPIONS se voit mis en œuvre, en fonction du niveau de résolution de la scène.

FIG. 5.19 – Etude de l'apport d'ESPIONS dans le cadre d'une augmentation de la complexité géométrique de la scène.

5.4 Couplage d'ESPIONS avec un module de simulation d'objets déformables

Dans cette section, nous présentons une implémentation consistant en un couplage d'ESPIONS avec un algorithme permettant la simulation de scènes contenant des objets déformables. Dans un premier temps, nous proposons une description succincte de l'algorithme en question. Puis, nous décrivons le fonctionnement du couplage mis en œuvre quant à l'utilisation d'ESPIONS dans ce contexte. Enfin, nous discutons des résultats obtenus, notamment dans le cadre d'une application visant à la manipulation d'objets déformables via une interface haptique.

5.4.1 Le module client

L'algorithme auquel nous nous intéressons dans cette partie est assimilable à un *simulateur dynamique* tel qu'introduit dans 1.3.2. Dépourvu d'un module de détection des collisions, cet algorithme a pour fonction de régir le comportement d'objets déformables en réaction à des contraintes de types unilatérales (i.e. aux collisions). L'approche mise en œuvre consiste en une définition des contacts fondée sur une extension du *modèle de Signorini*, initialement décrit dans le cadre de la mécanique des milieux continus, au cas d'espaces discrets formalisés selon la méthode des *éléments finis*. Afin de répondre aux contraintes de performances inhérentes à la réalisation de simulations permettant l'utilisation de systèmes à retour d'efforts, une méthode de résolution itérative de type Gauss-Seidel est également mise en œuvre. Pour plus de détail sur l'algorithme en question, nous invitons le lecteur à se référer à [DAK04, Dur04].

5.4.2 Fonctionnement du couplage

Comme dit précédemment, l'algorithme auquel nous nous intéressons dans cette section a pour vocation de régir le comportement d'objets déformables en réaction aux collisions inter-objets. Aussi, il s'avère nécessaire de lui fournir des informations géométriques permettant la définition des contacts (assimilables à des champs de distances). Dans le cadre de l'implémentation décrite dans cette section, nous proposons d'attribuer cette tâche d'identification des informations de contact à ESPIONS.

Phase d'initialisation

Les modèles géométriques manipulés par l'algorithme avec lequel nous souhaitons coupler ESPIONS sont des objets volumiques décrit de façon discrète selon la méthode des éléments finis. Ces derniers sont composés de tétraèdres. Aussi, afin d'utiliser ESPIONS dans ce contexte, les informations fournies à ce dernier quant à la définition géométrique des objets constituant la scène vont se limiter aux maillages polyédriques décrivant la surface de ces objets, sans prise en considération des éléments constituant leur volume. Ainsi, comme dans le cadre de l'implémentation précédente, les modèles fournis à ESPIONS dans ce contexte sont assimilables à des polyèdres à faces triangulaires.

Nature des informations échangées

ESPIONS → *module client* :

Dans le cadre du couplage introduit ici, les informations requises par le module client sont assimilables à des champs de distances. Il est à noter que nous ne faisons pas là allusion à des entités géométriques répondant à des contraintes aussi fortes que celles qualifiées de distances minimum locales, introduites dans le cadre de l'implémentation précédente (domaines de Voronoï, etc.). Aussi, les entités géométriques transférées au module client sont définies comme un sous-ensemble des individus ayant franchi l'ensemble des étapes de filtrage. Ce sous-ensemble corresponds aux n_B individus présentant les meilleurs propriétés au sein de cette élite. La constante n_B est une variable utilisateur. Elle permet une régulation de la complexité des traitements inhérents à la résolution des contacts par le module client. Aussi, elle doit être suffisamment faible pour permettre des temps de résolution compatibles avec une manipulation haptique, tout en garantissant que le nombre d'éléments retenus s'avérera suffisant pour permettre une définition complète des contacts.

module client → ESPIONS :

Contrairement au cas rigide, où seules les positions absolues des objets dans la scène sont à rafraîchir à chaque cycle d'intégration, la manipulation d'objets déformables nécessite également la prise en considération de l'évolution de la forme des objets tout au long de la simulation. Cette tâche consiste en une mise à jour régulière des positions des entités élémentaires qui composent les géométries considérées. Dans le cas polyédrique, ce sont les positions des sommets dans les repères de construction relatifs des objets qui vont ainsi se voir évoluer. Aussi, dans le cadre du couplage auquel nous nous intéressons dans cette section, c'est naturellement le module client qui a la charge de mettre à jour les positions des éléments géométriques décrivant les objets simulés. Ces nouvelles posi-

tions des sommets seront par la suite prises en compte par le processus ESPIONS lors de l'évaluation de la qualité des individus. En effet, la fonction de morphogénèse, qui vise rappelons-le à la détermination de la représentation phénotypique d'un individu, répond, dans le cas polyédrique, à la formulation suivante :

$$f_P(\mu_k^i) = H_o^w(\vec{O_{T_{id}}} + \alpha\vec{e_1} + \beta\vec{e_2})$$

où H_o^w correspond à une matrice de transformation homogène donnant la position du polyèdre dans le repère absolu (on parle là de la position du repère relatif de construction de l'objet au sein duquel sont définies les positions des primitives élémentaires qui le composent), $\vec{O_{T_{id}}}$ est un vecteur donnant les coordonnées du sommet du triangle sur lequel se situe le nucléon dans le repère local de l'objet (on parle là du sommet v_0 dans la description du triangle), les vecteurs $\vec{e_1}$ et $\vec{e_2}$ prennent respectivement pour valeur $v_0\vec{v}_1$ et $v_0\vec{v}_2$. Comme précédemment, les coefficients α et β correspondent aux coordonnées du nucléon dans la base ainsi définie.

Schéma de fonctionnement

Pour cette implémentation, nous proposons un schéma de couplage de type parallèle, tel qu'introduit dans 4.5.3. Ce mode de fonctionnement permet de gérer indépendamment la fréquence de fonctionnement de chacun des processus, caractéristique qui va s'avérer très intéressante dans le cadre de l'application présentée dans la suite de cette section.

5.4.3 Résultats obtenus

Ici, nous présentons les résultats obtenus dans le cadre du couplage décrit précédemment. Nous ne proposons pas là une étude détaillée des performances d'ESPIONS. En effet, les géométries manipulées au cours des simulations présentées dans la suite s'avèrent d'une complexité moindre (quelques centaines de triangles par modèle) et ne présentent pas le moindre risque de voir ESPIONS dans l'incapacité d'identifier l'ensemble des informations nécessaires au bon fonctionnement du processus de résolution des contacts. Le but de cette partie est simplement d'illustrer la validité du couplage auquel nous nous intéressons dans cette section au travers de quelques expérimentations.

Quelques résultats en images

Sur la figure 5.20 nous présentons plusieurs illustrations de simulations mettant en œuvre le couplage entre ESPIONS et le module de simulation considéré.

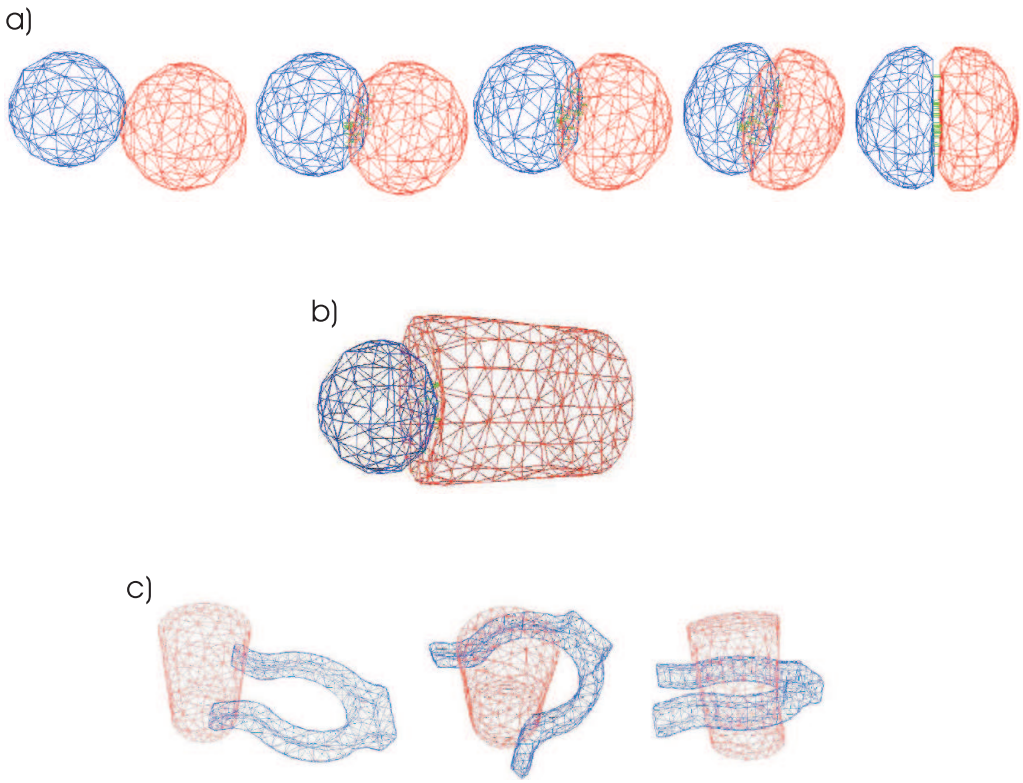


FIG. 5.20 – (a) Deux sphères déformables sont mises en situation de contact. On observe que, malgré une déformation importante des objets, les informations fournies par ESPIONS (représentées en vert) permettent une gestion pertinente du contact. (b) Une sphère contre un cylindre. (c) Une simulation de clipsage. On observe la bonne gestion des contacts conformant.

Manipulation haptique

Une application intéressante du couplage présenté dans cette section a consisté en la réalisation de simulations permettant la manipulation des objets simulés via un système à retour d'efforts. Dans ce contexte, le choix d'un schéma de couplage parallèle, qui n'impose aucune contrainte de synchronisation entre les processus, a montré tout son intérêt. En effet, contrairement au cas rigide, la simulation d'objets déformables entraîne une quantité de traitements importantes quant à la gestion du mouvement des éléments qui composent la scène. Aussi, le fait d'attribuer une unité de calcul à cette seule tâche, ainsi que l'absence de contraintes de synchronisation entre les modules, ont permis un cadencement optimal du processus de simulation. La fréquence de rafraîchissement des positions observée étant de l'ordre de 500Hz (nous parlons là de la fréquence du module de simulation dynamique bien entendu) la manipulation des objets via un système haptique a ainsi pu être réalisée. La figure 5.21 illustre ces résultats.

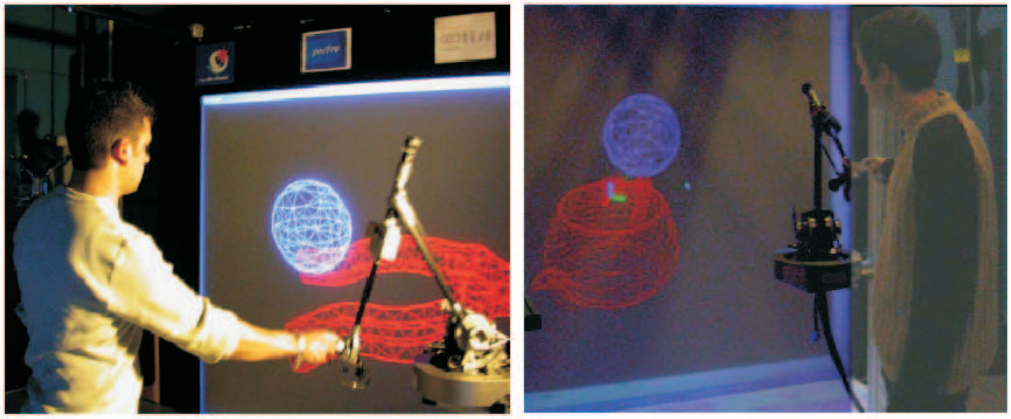


FIG. 5.21 – Deux exemples de manipulations d'objets déformables via un système à retour d'effort de type *VIRTUOSE 6D 35-45 HAPTION*.

5.4.4 Discussion

Comme il l'a été précisé précédemment, les objets simulés dans le cadre des expérimentations de ce couplage présentaient une faible complexité géométrique. Aussi, en l'état actuel des choses, on ne peut raisonnablement revendiquer une aptitude de notre algorithme à identifier l'ensemble des informations nécessaires à la définition des contacts dans le cadre de simulations mettant en scène des objets déformables de grande complexité. Cependant, l'implémentation traitée dans cette section démontre la capacité de notre algorithme à fonctionner dans un tel contexte. De plus, il est à noter que le fait que les objets soient déformables n'entraîne pas d'augmentation de la complexité de notre algorithme, le déplacement des sommets n'engendrant pas de traitements supplémentaires pour ESPIONS. En effet, ces déplacements n'ont d'effet que sur la représentation phénotypique des individus et sont déjà explicitement pris en considération dans l'expression de la fonction de morphogénèse rappelée précédemment. D'autres expériences devront être menées afin de caractériser plus précisément le comportement d'ESPIONS dans le cadre de simulations d'objets déformables.

5.5 Distribution d'ESPIONS

Dans cette dernière section, nous nous intéressons à la distribution du processus ESPIONS. Nous avons déjà introduit cette notion dans le cadre du chapitre dédié à la présentation de l'algorithme (Cf. paragraphe 4.5.4). Rappelons que le mode opératoire sur lequel est fondé cette distribution consiste en un partage de l'espace de recherche entre plusieurs sous-populations évoluant indépendamment. En pratique, des groupes de paires d'objets sont définis et respectivement attribués à chaque sous-population.

Dans la suite, nous proposons une expérimentation visant à illustrer ce mode de fonctionnement. Il est à noter que le but de cette dernière n'est pas de mettre en œuvre un processus de simulation complet, mais simplement de valider le schéma proposé. Aussi, aucun module client n'est couplé à ESPIONS dans ce contexte.

5.5.1 Procédure expérimentale

Le but de la distribution du processus ESPIONS est de permettre une exploitation optimale de l'ensemble des unités de calcul disponibles. Aussi, dans le cadre de cette expérimentation, nous proposons d'observer le niveau de sollicitation des deux cœurs qui composent le processeur dont nous disposons dans ce contexte (intel P4 bi-cœurs cadencé à 3.6 GHz).

Pour ce faire, l'expérimentation se déroule en deux temps. Une première étape consiste à observer le niveau de sollicitation du processeur dans le cadre d'une simulation où la distribution n'est pas mise en œuvre. Dans un deuxième temps, l'expérience est renouvelée dans les mêmes conditions, si ce n'est que, cette fois-ci, le processus de recherche global est décomposé en deux sous-processus (i.e. des *threads*) selon le mode opératoire précédemment défini.

Définition de la scène

La scène utilisée dans le cadre de cette expérimentation est composée de quatre instances d'un même polyèdre constitué d'environ 13000 triangles. Ces derniers sont positionnés de manières à ce qu'une multitude de zones de proximité soient observable au sein de la scène. La figure 5.22 illustre cette scène.

Décomposition de l'espace de recherche

Comme expliqué précédemment, la parallélisation du processus va consister en une décomposition de l'espace de recherche en deux sous-espaces. La scène sur laquelle nous

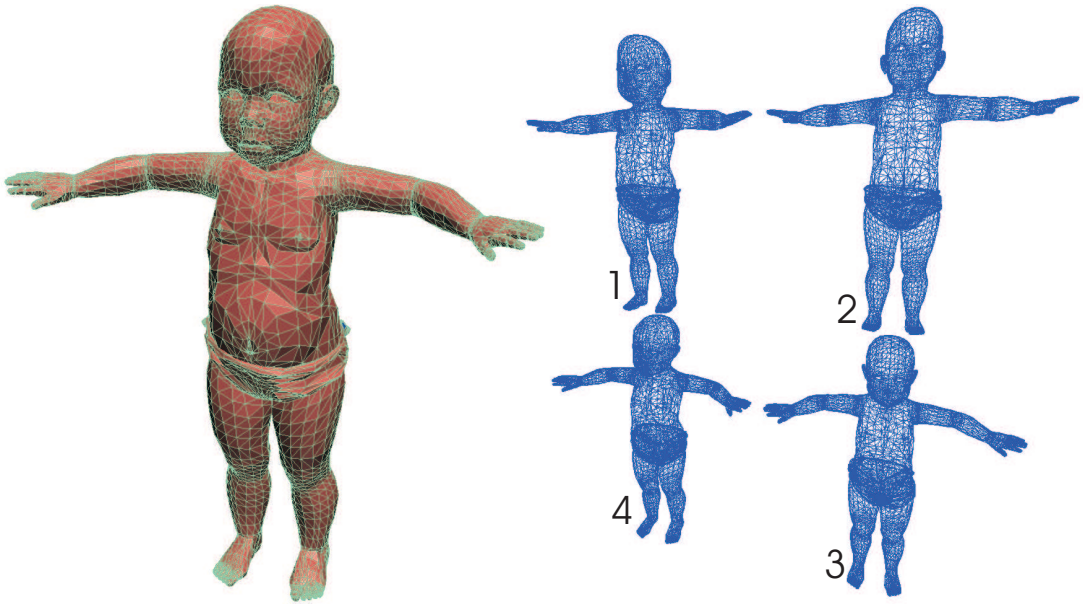


FIG. 5.22 – Scène utilisée dans le cadre de la distribution d'ESPIONS. (A gauche) la géométrie utilisée dans ce contexte. Il s'agit d'un modèle de type polyédrique composé d'environ 13000 triangles. (A droite) Les quatre instances qui composent la scène sont positionnées de manière à faire apparaître plusieurs zones de proximité.

nous basons dans le cadre de ces expériences est composée de quatre objets, soit, en termes de combinatoire, six paires. Pour la distribution, nous proposons d'attribuer à chacune des deux populations trois de ces paires. Aussi, nous choisissons arbitrairement de les répartir de la façon suivante : la première population, notée Π_1, se voit attribuer l'ensemble de paires $\{\{1,2\}\{1,3\}\{1,4\}\}$ (les indices correspondent à ceux spécifiés sur la figure 5.22). La combinatoire formée par les paires restantes, soit $\{\{2,3\}\{2,4\}\{3,4\}\}$, est attribuée à la deuxième population, notée Π_2.

5.5.2 Résultats expérimentaux

Expérience 1

La figure 5.23 illustre les résultats obtenus dans le cadre de la première expérience qui consistait, rappelons-le, en une utilisation d'ESPIONS en mode nominal (i.e. sans distribution). On observe que le processeur n'est exploité qu'à hauteur de 50% de ses capacités.

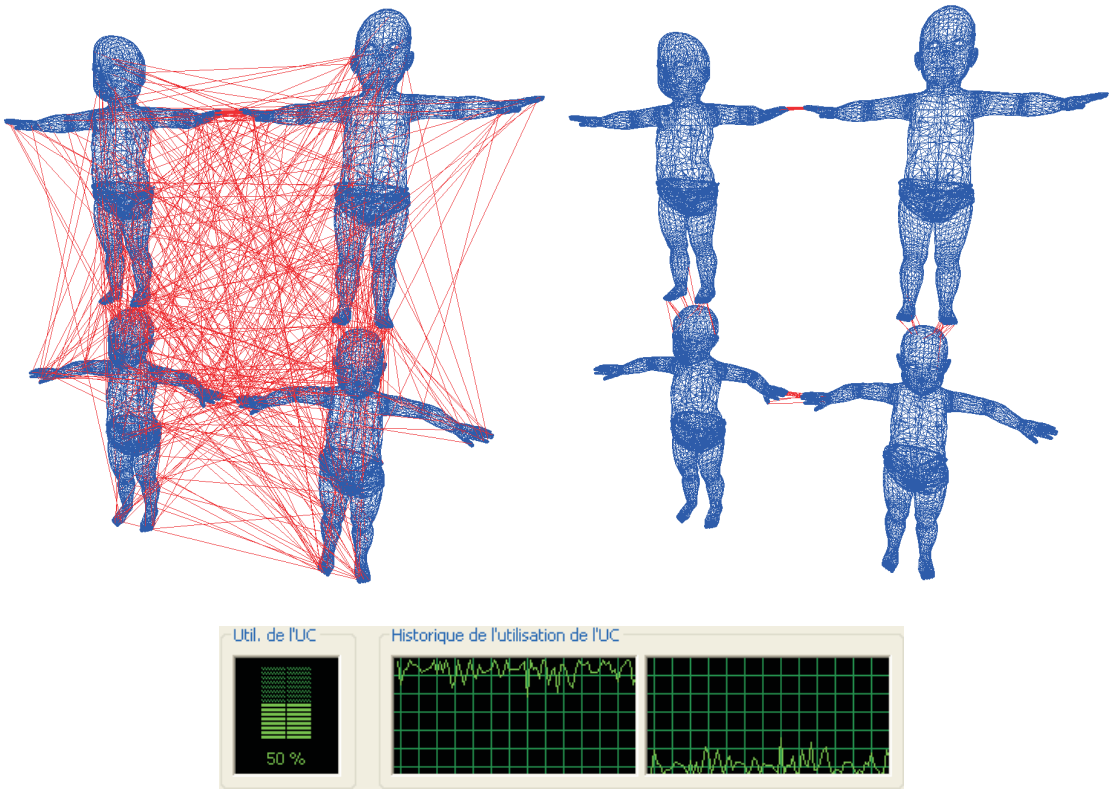


FIG. 5.23 – Expérience 1 : fonctionnement d'ESPIONS en mode non-distribué. En haut à gauche, on observe l'ensemble des individus constituant la population. En haut à droite, seuls les individus dont la taille est inférieure à un seuil donné (ici 30 mm) sont affichés. En bas, une image du moniteur *Windows* permet d'observer le niveau de sollicitation de chacune des unités de calcul.

Expérience 2

La figure 5.24 illustre les résultats obtenus dans le cadre de la deuxième expérience lors de laquelle le processus ESPIONS s'est vu décomposé en deux sous-processus. Contrairement au cas de l'utilisation nominale étudié dans le cadre de l'expérience précédente, on observe que les deux unités qui composent le processeur sont désormais pleinement exploitées. Le fait que chacune des populations soit restituée dans une couleur différente permet d'apprécier la décomposition de l'espace de recherche.

5.5.3 Discussions

Certes, les résultats expérimentaux présentés dans cette section ne permettent pas de caractériser quantitativement l'apport de la distribution d'ESPIONS quant à ses performances. Cependant, plusieurs conclusions peuvent être tirées de ces expériences. Tout

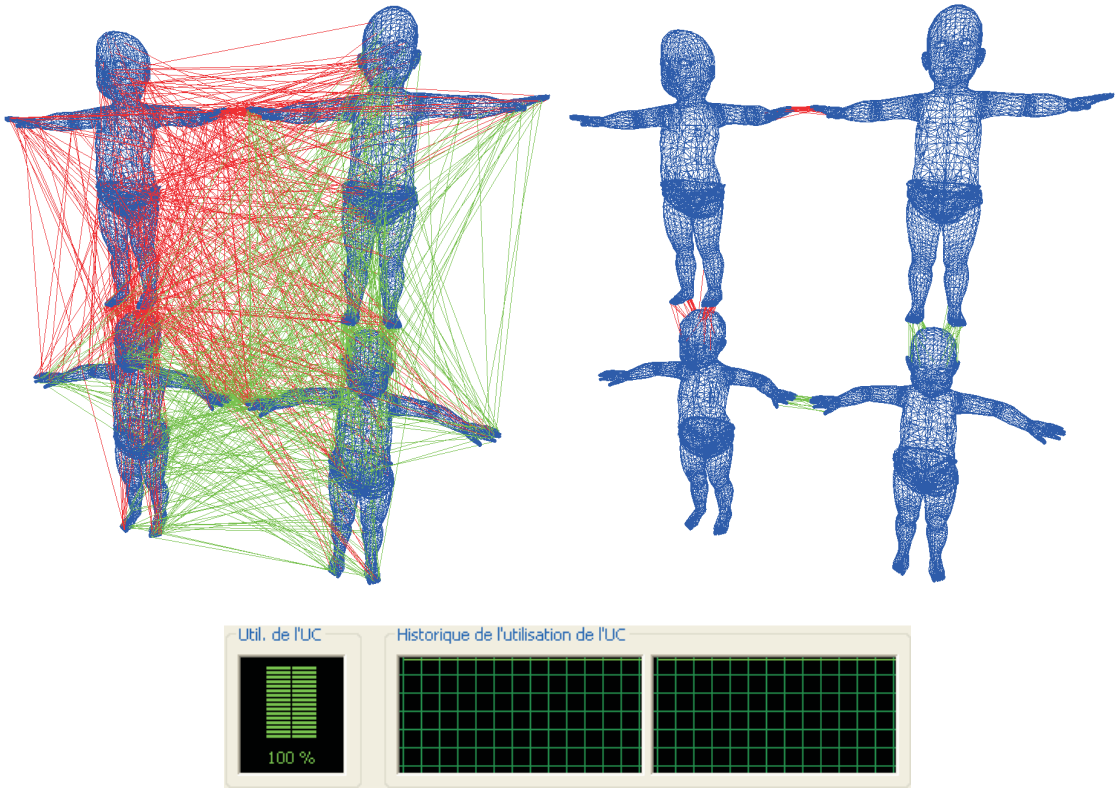


FIG. 5.24 – Expérience 2 : fonctionnement d'ESPIONS en mode distribué. En haut à gauche, on observe l'ensemble des individus constituant les populations. En haut à droite, seuls les individus dont la taille est inférieure à un seuil donné (ici 30 mm) sont affichés. Les individus représentés en rouge appartiennent à Π_1, ceux en vert à Π_2. En bas, une image du moniteur *Windows* permet d'observer le niveau de sollicitation de chacune des unités de calcul.

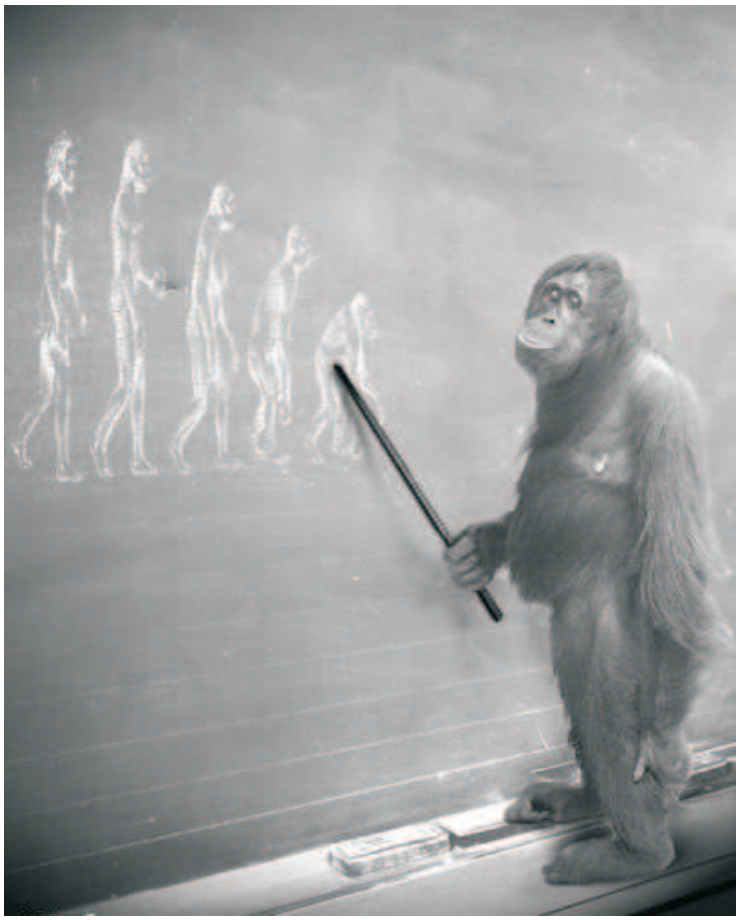
d'abord, cette implémentation a permis de valider le mode opératoire proposé pour décomposer le processus ESPIONS en plusieurs sous-processus. Nous avons également pu observer les effets attendus quant à la répartition de la charge sur l'ensemble des unités de calcul disponibles.

D'autres expériences doivent désormais être menées afin d'évaluer plus précisément l'influence de la distribution d'ESPIONS sur ses performances. Celles-ci devront être réalisées dans le cadre d'une intégration de celui-ci au sein d'un processus de simulation complet, tel que ceux présentés dans les sections précédentes. On peut d'ores et déjà émettre l'hypothèse que le fait de distribuer ESPIONS va engendrer une réduction sensible du temps nécessaire à l'identification des zones de proximité. De plus, le fait de solliciter plusieurs unités de calcul va permettre d'augmenter le nombre d'individus manipulés (même si ces derniers évoluent au sein de population indépendantes) et ainsi

contribuer au maintien de la diversité génétique (Cf. paragraphe 5.2.5).

Plusieurs voies d'investigation s'ouvrent à nous dans le cadre de cette problématique portant sur la distribution de notre algorithme. On pourrait par exemple réfléchir à un mode opératoire permettant une interaction entre les sous-populations, actuellement totalement indépendantes. Pour ce faire, plusieurs approches proposées dans la littérature pourraient être mises en œuvre, telles que la technique des îlots introduite dans 3.6.3. Une autre problématique réside dans la décomposition de l'espace de recherche. Actuellement gérée par l'utilisateur, on pourrait imaginer de réaliser celle-ci grâce à des heuristiques permettant une répartition optimale des paires d'objets simulés en fonction de divers critères tels que leur complexité.

Conclusions et perspectives



Conclusion générale

Au travers de cette thèse, nous nous sommes intéressés à la problématique de la détection des collisions au sein d'environnements virtuels, notamment dans le cadre de simulations interactives consistant en la manipulation de scènes complexes. Nos recherches se sont particulièrement orientées vers la mise en œuvre de techniques d'optimisation dans ce contexte. Aussi, la principale contribution issue des travaux réalisés dans le cadre de cette thèse réside dans l'élaboration d'un algorithme fondé sur une approche évolutionniste. Baptisé ESPIONS, cet algorithme peut se voir utilisé de différentes manières. On peut cependant le définir comme un processus visant à identifier des *champs de distances minimum locales* entre des objets virtuels de natures diverses. Nous avons également proposé une étude expérimentale visant à caractériser l'influence des paramètres d'ESPIONS sur son comportement général. Celle-ci nous a permis d'identifier différents critères destinés à faciliter l'établissement de jeux de paramètres pertinents en fonction des problèmes considérés. Plusieurs implémentations de l'algorithme ont également étaient présentées afin d'illustrer les différents modes opératoires introduits et d'évaluer les apports et limites de l'approche proposée.

Dans la suite, nous proposons un récapitulatif de l'ensemble des sujets traités dans ce mémoire. Nous introduirons également plusieurs perspectives quant à la poursuite des travaux réalisés dans le cadre de cette thèse.

Récapitulatif

Chapitre 1 : Une présentation du contexte dans lequel cette thèse s'est inscrite a été proposée. Nous y avons décrit plusieurs applications concrètes de la réalité virtuelle dans deux domaines : l'industrie et la médecine. Nous avons également proposé une présentation générale des différentes composantes qui composent une plateforme de réalité virtuelle, en nous attardant particulièrement sur les principes fondamentaux qui régissent le fonctionnement d'un moteur physique. Enfin, nous avons discuté des problématiques inhérentes à la détection des collisions en temps réel.

Chapitre 2 : Un état de l'art dédié aux techniques de détection des collisions présentées dans la littérature a été proposé. Celui-ci s'est articulé autour des différentes problématiques auxquelles nous nous sommes particulièrement intéressés dans le cadre de cette thèse : complexité de la scène simulée, propriétés géométriques des objets manipulés, …. Nous y avons abordé, entre autre, les techniques visant à accélérer la détection des collisions, les approches adaptés à la manipulation d'objets déformables, et l'influence

des propriétés géométriques des objets simulés (natures, convexité, ...) sur le choix des approches algorithmiques à mettre en œuvre pour la détection des collisions.

Chapitre 3 : Une présentation générale des concepts fondamentaux relatifs aux approches évolutionnaires a été proposée. Nous y avons abordé les principes généraux qui régissent ces approches avec l'introduction des différentes entités manipulées dans ce contexte (individus, etc.). Nous nous sommes également intéressé aux modes opératoires mis en œuvre pour la manipulation de ces entités. Enfin, des notions plus avancées telles que la parallélisation de telles approches ont été abordées.

Chapitre 4 : Une présentation détaillée de l'algorithme *ESPIONS* a été proposée. Nous y avons décrit l'ensemble des composantes de l'algorithme, tant au niveau des entités élémentaires utilisées dans ce contexte, que des opérateurs permettant leur manipulation. Plusieurs formalismes ont d'ailleurs été introduits, afin d'illustrer l'aptitude d'*ESPIONS* à fonctionner au sein d'environnements composés d'éléments géométriques de différentes natures. Le schéma de fonctionnement général de l'algorithme a également été introduit, avec une description précise de chacune des étapes qui le composent. Enfin, différentes architectures ont été proposées en vue de l'intégration d'ESPIONS au sein de processus de simulation, avec une réflexion sur l'exploitation de l'éventuelle multiplicité des unités de calculs mises à disposition.

Chapitre 5 : La première partie de ce chapitre a consisté en une étude expérimentale visant à analyser le comportement de l'algorithme en fonction des paramètres qui lui sont spécifiés. Cette étude avait pour but de caractériser l'influence de chacun de ces paramètres sur différentes caractéristiques de l'algorithme. Dans la deuxième partie du chapitre, nous avons présenté plusieurs implémentations de notre algorithme dans différents contextes. Ainsi, plusieurs modes d'intégration d'ESPIONS au sein de processus de simulation ont été mis en œuvre. Les résultats obtenus dans le cadre de chacune de ces expérimentations ont été analysés et discutés afin d'évaluer la pertinence d'une utilisation d'ESPIONS dans chacun des cas abordés. Enfin, une implémentation visant à illustrer la parallélisation de l'algorithme a également été présentée.

Perspectives

A l'issue de cette thèse, bien des chemins restent à explorer pour d'améliorer les performances de l'algorithme ESPIONS. Dans ce dernier paragraphe, nous proposons plusieurs voies d'investigation qui nous semblent prometteuses. Celles-ci sont réparties par problématique.

Paramétrage de l'algorithme

Dans le chapitre 5, nous avons proposé une étude expérimentale visant à caractériser l'influence des paramètres d'ESPIONS sur son comportement général. Même si les résultats obtenus dans ce contexte nous ont permis de dégager plusieurs critères a priori pertinents, nous ne sommes pas aujourd'hui en mesure de proposer une méthodologie permettant l'établissement de jeux de paramètres adaptés aux problèmes posés. La phase de réglage d'ESPIONS se fonde en effet sur une démarche empirique, incompatible avec une utilisation de celui-ci par un utilisateur non-spécialiste.

Aussi, l'une des suites logiques à nos travaux devrait consister en l'élaboration d'heuristiques permettant une identification des paramètres optimaux pour une situation donnée. Plusieurs critères inhérents à la tâche à réaliser seront probablement à prendre en compte dans ce contexte : définition des géométries, complexité de la scène, nature des interactions entre l'utilisateur et l'environnement simulé (niveau de réalisme escompté, type d'interfaces utilisées), capacité de calcul de la plateforme sur laquelle la simulation est réalisée, quantité de mémoire vive allouée au processus, etc.

Une autre voie nous semble également prometteuse. Il s'agit de la mise en œuvre de techniques d'auto-adaptabilité des coefficients de variation en fonction de la configuration courante. Introduites dans le chapitre dédié à l'introduction des approches évolutionnaires (Cf. paragraphe 3.4.2), ces processus pourraient produire des résultats intéressants dans le cadre de la problématique qui nous intéresse. En effet, la configuration géométrique de la scène ne cessant d'évoluer, il semble pertinent de modifier automatiquement certains des paramètres en cours de simulation.

Gestion d'autres types d'objets

Au travers de ce mémoire, nous nous sommes particulièrement intéressés au cas des objets polyédriques. Nous avons également proposé des formalismes permettant l'utilisation d'ESPIONS dans le cadre de simulations mettant en scène des nuages de points (validés dans le cadre développements non-exposés dans ce mémoire) et des surfaces paramétriques. D'autres types de géométries pourraient également être gérés par ESPIONS. Nous pensons par exemple au cas des surfaces implicites. Aussi, divers formalismes pourraient être élaborés dans ce but.

Amélioration du processus de parallélisation

Le mode opératoire actuellement mis en œuvre pour la parallélisation d'ESPIONS, bien qu'il réponde à sa fonction première qu'est l'exploitation de l'ensemble des unités de

calcul disponibles, peut très certainement être amélioré. En effet, même si l'architecture proposée semble adaptée, le processus de décomposition de l'espace de recherche n'offre pas un niveau de granularité suffisamment fin pour permettre une répartition optimale de la combinatoire. Pour ce faire, cette décomposition devrait non pas être réalisée au niveau de paires d'objets, mais au niveau de paires de portions d'objets. Plusieurs approches pourraient être utilisées pour le découpage des géométries (domaines de Voronoï, etc.).

Application de techniques de multirésolution

Dans l'état de l'art dédié à la détection des collisions, nous avons introduit des techniques de détection des collisions fondées sur l'utilisation de différents niveaux de résolution pour la définition des objets (Cf. paragraphe 2.2.5). Afin de permettre une amélioration des performances de l'algorithme ESPIONS, particulièrement dans le cadre de simulations mettant en scène des éléments complexes, il serait probablement intéressant d'exploiter de telles techniques. On pourrait en effet imaginer de faire évoluer les nucléons sur différentes représentations des objets, présentant des niveaux de résolution plus ou moins élevés. Dans ce contexte, les nucléons appartenant à des individus de mauvaise qualité se déplaceraient sur les couches basses (i.e. de basse résolution) afin de réduire la combinatoire à explorer, et les nucléons constituant les bons individus évolueraient sur les couches hautes afin d'identifier précisément les informations géométriques nécessaires à la définition des contacts.

Annexe A

Architecture de l'algorithme ESPIONS

Dans cette section, nous présentons l'architecture de l'algorithme *ESPIONS*, ou du moins de son implémentation telle qu'elle a été réalisée dans le cadre des travaux d'expérimentation réalisés au cours de cette thèse. Le but de cette présentation n'est pas de rentrer dans les détails de l'implémentation informatique proprement dite, mais simplement d'identifier d'une façon générale les différentes entités nécessaires à sa réalisation. La figure A.1 illustre l'architecture proposée.

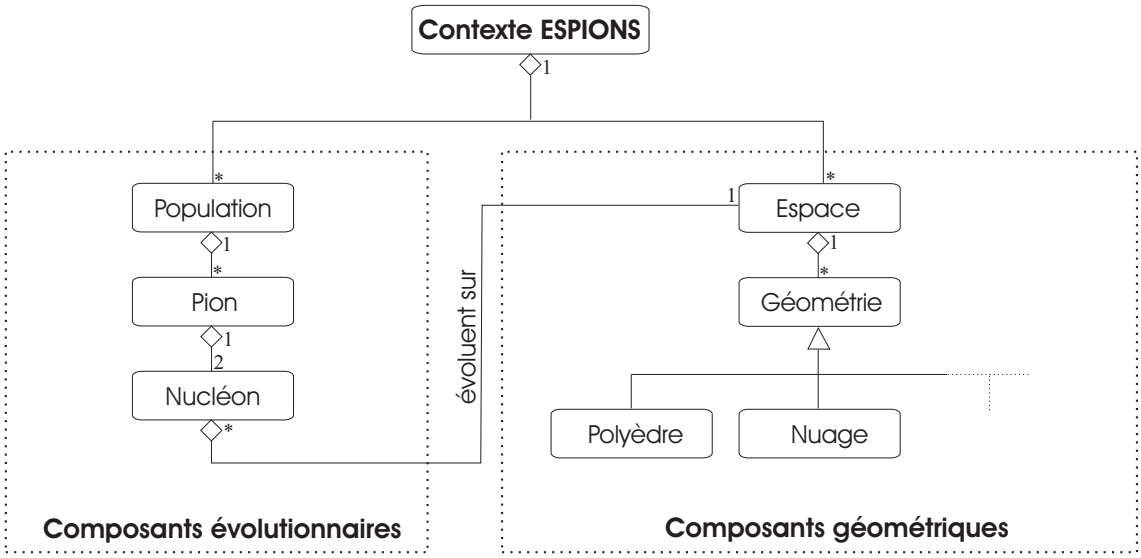


FIG. A.1 – Schéma UML présentant l'architecture de l'algorithme *ESPIONS*

Dans un premier temps, il semble intéressant de constater que les entités présentées ici peuvent être classifiées selon deux catégories : Les composantes dites *évolutionnaires* qui interviennent directement au sein du processus d'évolution, et les composantes dites *géométriques* qui composent l'espace à explorer (i.e. les éléments géométriques contenus

dans la scène). Au sens informatique, elles sont assimilables à des objets. Aussi, dans la suite de l'exposé, nous emprunterons parfois des éléments de vocabulaire inhérents à la programmation orientée objet. On parlera par exemple d'attributs pour définir les éléments qui constituent un objet donné.

Dans la suite, nous proposons une description de chacune de ces entités :

- **Contexte *ESPIONS* :** Comme son nom l'indique, cet objet est en fait assimilable à une instance de l'algorithme. Il contient d'une part une ou plusieurs populations d'individus et, d'autre part, un ou plusieurs espaces d'exploration composés de modèles géométriques. Même si dans la suite de l'exposé nous traitons majoritairement du cas ou le contexte ESPIONS contient une seule *population* pour un seul *espace* d'exploration, ce dernier peut tout à fait considérer simultanément plusieurs instances de telles entités. Nous retrouvons là la notion de parallélisation d'*ESPIONS* introduite dans 4.5.4, et visant à décomposer l'espace de recherche en sous-espaces respectivement explorés par des populations distinctes, cela afin de rendre possible une exploitation optimale des plateformes matérielles présentant plusieurs unités de calcul.

  Attributs :
  - *nbPopulations* : nombre de populations composant le contexte (≥ 1).
  - *populationsArray* : Tableau (au sens informatique) contenant les populations du contexte.
  - *nbSpaces* : nombre d'espaces à explorer (≥ 1).
  - *spacesArray* : Tableau contenant la description des espaces.

- **Population :** Elle contient un certain nombre (fixe) d'individus. C'est en son sein que les processus inhérents au processus d'évolution sont réalisés. Aussi, en plus des entités constituant la population, cet objet se voit affecter des paramètres conditionnant la classification des individus et l'effet des opérateurs de variation.

  Attributs :
  - *nbPions*(p) : nombre de pions contenus dans la population (≥ 2).
  - *pionsArray* : Tableau contenant l'ensemble des pions constituant la population.
  - $\%_R$: Pourcentage d'individus ayant perdu durant la phase de tournoi et devant subir une mutation de type repositionnement aléatoire.
  - $\%_X$: Pourcentage d'individus ayant perdu durant la phase de tournoi et devant être remplacer par un individu issu du croisement de deux individus présentant de bonnes propriétés.
  - $\%_E$: Pourcentage d'individus ayant perdu durant la phase de tournoi et devant

subir une mutation de type exploration du voisinage.
  - n_S : Nombre de sauts à réaliser durant une opération d'exploration du voisinage.
  - γ : Paramètre d'ajustement de la forme de la fonction fitness au vu de réguler le partage des ressources.

- **Pion :** Comme son nom l'indique, cet objet correspond à un individu. Ainsi, comme défini précédemment, il est constitué de deux nucléons. Les opérateurs de variation sont appliqués sur des objets de cette nature. Il contient également une expression qualitative de l'entité géométrique à laquelle il correspond (i.e. sa *fitness*).

  Attributs :
  - *nucleon*1 : Premier nucléon constituant l'individu.
  - *nucleon*2 : Deuxième nucléon constituant l'individu.
  - $fitness$: Expression de la qualité de l'individu, soit, comme exprimé précédemment, le carré de la distance euclidienne séparant les nucléons qui le constituent.
  - $task$: Tâche que l'individu se voit attribuer à l'issue des phases de classification. Elle correspond en fait au type d'opérateur que l'individu va se voir appliqué durant la phase de mutation.

- **Nucléon :** Comme présenté dans le chapitre précédent, un nucléon est assimilable à un point 3D évoluant à la surface de l'une des géométries composant la scène simulée. Cet objet contient en fait une double représentation de l'entité géométrique à laquelle il correspond : Une représentation paramétrique donnant la position de l'élément à la surface de la géométrie sur laquelle il évolue (elle même identifiée par un indice au sein de l'espace géométrique), et une représentation vectorielle au sein de l'espace euclidien (déterminée à partir de la première représentation). Ces deux représentations permettent une définition duale des individus au sein des espaces génotypique et phénotypique tels qu'ils sont définis dans la section 4.2.

  Attributs :
  - *espace* : Espace géométrique au sein duquel le nucléon évolue.
  - *geometryID* : Identifiant de la géométrie sur laquelle le nucléon est positionné au sein de l'espace (élément du génotype).
  - *primitiveID* : Identifiant de la primitive sur laquelle le nucléon est positionné au sein de la géométrie fréquentée (élément du génotype).
  - *localParametersArray* : Tableaux contenant les paramètres à valeurs réelles donnant la position du nucléon au sein de la primitive fréquentée :α et β dans le cas polyédrique par exemple (éléments du génotype).
  - *absolutePosition* : Vecteur 3 dimensions donnant la position euclidienne du nucléon dans le repère absolu (représentation phénotypique).

– **Espace :** Un espace contient tout ou partie des géométries constituant la scène. Il est assimilable à un espace à explorer par les nucléons constituant les individus d'une population. En plus de ces informations géométriques, l'espace a pour attributs les matrices de densité contenant les informations inhérentes au taux de fréquentation des paires de régions. Comme défini dans 4.4, ces informations rendent possible le maintient de la diversité génétique au travers d'une approche fondée sur le *partage des ressources*.

  Attributs :
  – *nbGeometries* : Nombre de géométries contenues dans l'espace (≥ 2).
  – *geometriesArray* : Tableau contenant les géométries contenues dans l'espace.
  – *densityMatricesArray* : Tableau contenant les matrices de densité.

– **Géométrie :** Elle est assimilable à un objet virtuel évoluant au sein de la scène. Une géométrie peut être de différentes natures. Aussi, on voit ici apparaître la notion d'héritage au sens analyse objet. L'objet considéré ici est assimilable à une *classe mère*. Elle a pour attribut la position de l'objet dans la scène et le nombre de régions constituant celui-ci (on ne détaille pas ici les attributs tels que l'état de mobilité de l'objet considéré).

  Attribut :
  – *absolutePosition* : Position de la géométrie dans la scène.
  – *nbRegions* : Nombre de régions composant la géométrie.

– **Polyèdre :** Géométrie de type polyédrique (ici, on parle de polyèdres au sens maillages triangulaires).

  Attributs :
  – *nbVertices* : Nombre de sommets définissant le polyèdre (≥ 3).
  – *verticesArray* : Tableaux contenant la définition des sommets du polyèdre (*nbVertices* vecteurs de position).
  – *nbTriangles* : Nombre de triangles composant le polyèdre (≥ 1).
  – *trianglesArray* : Tableaux contenant la définition des triangles du polyèdre (*nbTriangles* triplets donnant les indices des sommets constituant le triangle).
  – *trianglesRegionsArray* : Tableaux contenant les indices des régions auxquelles les triangles appartiennent (*nbTriangles* indices).

– **Nuage :** Géométrie de type nuage de points.

  Attributs :
  – *nbPoints* : Nombre de points définissant le nuage.
  – *pointsArray* : Tableaux contenant la définition des points du nuage (*nbPoints*

vecteurs de position).

– *pointsRegionsArray* : Tableaux contenant les indices des régions auxquelles les points appartiennent (*nbPoints* indices).

Bibliographie

[Ang96] P. J. Angeline. Genetic programming's continued evolution. *In Peter J. Angeline and K. E. Kinnear, Jr., editors, Advances in Genetic Programming 2. MIT Press, Cambridge, MA, USA*, pages 1–20, 1996.

[Bar90] D. Baraff. Curved surfaces and coherence for non-penetring rigid body simulation. *SIGGRAPH'90 Conference Proceedings, Computer Graphics*, 24(4) :19–28, august 1990.

[BBR93] D. Beasley, D. R. Bull, and Martin R. R. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2) :101–125, 1993.

[Bäc93] T. Bäck. Optimal mutation rates in genetic search. *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993.

[Ber99] G. Van Den Bergen. A fast and robust gjk implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2) :7–25, 1999.

[BF79] J. L. Bentley and J. H. Friedman. Data structures for range searching. *ACM Computing Surveys*, pages 398–409, december 1979.

[BT95] S. Bandi and D. Thalmann. An adaptative spacial subdivision of the object space for fast collision detection of animated rigid bodies. *EUROGRAPHICS'95 Conference. Computer Graphics Forum, Maastricht*, 14(3) :259–270, august 1995.

[Cam97] S. A. Cameron. Enhancing GJK : Computing minimum and penetration distance between convex polyhedra. *IEEE International Conference on Robotics and Automation*, pages 3112–3117, 1997.

[Can86] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2) :200–209, 1986.

[CC86] S. A. Cameron and R. K. Culley. Determining the minimum translational distance between two convex polyhedra. *IEEE International Conference on Robotics and Automation*, pages 591–597, 1986.

[CLRS99] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Individual gp : an alternative viewpoint for the resolution of complex problems. *Genetic and Evolutionary Computation Conference GECCO99, San Francisco, USA*, 1999.

[CN01] J. A. Carretero and M. A. Nahon. A genetic algorithm for calculating minimum distance between convex and concave bodies. *Proceedings of The 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space : A New Space Odyssey, Montreal, Québec, Canada*, pages 18–22, june 2001.

[CN05] J. A. Carretero and M. A. Nahon. Solving minimum distance problems with convex or concave bodies using combinatorial global optimization algorithms. *IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 36, No. 6*, december 2005.

[CR99] A. Crosnier and J. R. Rossignac. Tribox bounds for three dimensional objects. *Computers and Graphics*, 23(3) :429–437, june 1999.

[CW96] K. Chung and W. Wang. Quick elimination of non-interference polytopes in virtual environment. *Third European Workshop on Virtual Environments, Monté-Carlo*, février 1996.

[DAK04] C. Duriez, C. Andriot, and A. Kheddar. Signorini's contact model for deformable objects in haptic simulations. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan*, 2004.

[Dar59] C. Darwin. *L'Origine des espèces*. Larousse, 1859.

[DeJ75] K. A. DeJong. An analysis of the behavior of a class of genetic adaptive systems. *PhD thesis, University of Michigan, Ann Arbor*, 1975.

[DK90] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra : an unified approach. *17th International Conference on Automata Language Programming*, pages 400–413, 1990.

[DS92] K. A. DeJong and W. M. Spears. A formal analysis of the role of multipoint crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence Journal*, 5(1) :1–26, 1992.

[Dur04] C. Duriez. Thèse de doctorat : Contact frottant entre objets déformables dans des simulations temps-réel avec retour haptique. *Université d'Evry*, 2004.

[EL01] S. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *EUROGRAPHICS'01 Conference, Computer Graphics Forum, Manchester*, 20(3), september 2001.

[GD04] S. Guy and G. Debunne. Monte-carlo collision detection. *Rapport de recherche INRIA 5136*, 2004.

[GH89] E. G. Gilbert and S. M. Hong. A new algorithm for detecting the collision of moving objects. *IEEE International Conference on Robotics and Automation*, 1989.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. Keerty. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2)(Annual Conference Series) :193–203, april 1988.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree : a hierarchical structure for rapid interference detection. *SIGGRAPH'96 Conference Proceedings, Computer Graphics annual conference series*, pages 171–180, august 1996.

[Gol89] D. E. Goldberg. Genetic algorithms in search, optimization and machine learning. *Addison Wesley : Reading, MA*, 1989.

[GR87] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 41–49, 1987.

[GRLM03] N. Govindaraju, S. Redon, M. Lin, and D. Manocha. Cullide : Interactive collision detection between complex models in large environments using graphics hardware. *Eurographics Workshop on Graphics Hardware, San Diego, USA*, 19(21), july 2003.

[GWA92] V. Gordon, D. Whitley, and Bohn A. Dataflow parallelism in genetic algorithms. *Proceedings of the 2nd Conference on Parallel Problems Solving from Nature, North Holland*, pages 42–553, 1992.

[Hah88] J. K. Hahn. Realistic animation of rigid bodies. *SIGGRAPH'88 Conference Proceedings, Computer Graphics*, 22(4) :299–308, 1988.

[HDLM96] M. Hughes, C. DiMattia, M. C. Lin, and D. Manocha. Efficient and accurate interference detection for polynomial deformation. *Computer Animation'96 Conference*, 1996.

[Hol75] J. H. Holland. Adaptation in natural and artificial systems. *Univ. of Michigan Press : Ann Arbor*, 1975.

[JC01] D. E. Johnson and E. Cohen. Spatialized normal cone hierarchies. *ACM Symposium on Interactive 3D Graphics, ACM SIGGRAPH*, march 2001.

[JG03] Klein J. and Zachmann G. Adb-trees : Controlling the error of time-critical collision detection. *8th International Fall Workshop Vision, Modeling, and Visualization (VMV), University München, Germany*, 19(21), november 2003.

[JP04] D. L. James and D. K. Pai. Bd-tree : Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, august 2004.

[JW04] D. E. Johnson and P. Willemsen. Accelerated haptic rendering of polygonal models through local descent. *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS 2004)*, 2004.

[KHM+98] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualisation and Computer Graphics*, 4(1), march 1998.

[KNF04] S. Kimmerle, M. Nesme, and F. Faure. Hierarchy accelerated stochastic collision detection. In *Vision, Modeling, and Visualization*, Stanford, California, 2004.

[KOLD02] Y. Kim, M. Otaduy, M. Lin, and Manocha D. Fast penetration depth computation for physically-based animation. *ACM Symposium on Computer Animation, San Antonio, USA*, pages 21–22, july 2002.

[Koz89] J. R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. *N. S. Sridharan, editor, Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, 1 :768–774, 1989.

[Koz92] J. R. Koza. Genetic programming : On the programming of computers by means of natural selection. *MIT Press, Cambridge, MA, USA*, 1992.

[KZ04] J. Klein and G. Zachmann. Point cloud collision detection. *In Proceedings EUROGRAPHICS'03. Computer Graphics forum. Grenoble*, pages 567–576, 2004.

[LAM01] T. Larsson and T. Akenin-Möller. Collision detection for continuously deforming bodies. *EUROGRAPHICS'01 Conference. Computer Graphics Forum, Manchester, England*, 20(3) :325–333, september 2001.

[LAM03] T. Larsson and T. Akenin-Möller. Efficient collision for models deformed by morphing. *The Visual Computer 19*, 2 :164–174, may 2003.

[LC91] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. *IEEE International Conference on Robotics and Automation*, pages 1008–1015, 1991.

[LCN99] J. C. Lombardo, M.-P. Cani, and F. Neyret. Real-time collision detection for virtual surgery. *Computer Animation'99 Conference, Geneve, Switzerland*, pages 82–91, may 1999.

[LM95] M. C. Lin and D. Manocha. Fast interference detection between geometric models. *The Visual Computer*, 11 :542–561, 1995.

[MKE03] J. Mezger, S. Kimmerle, and O. Etzmuss. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG 11*, 2 :322–329, 2003.

[MKF03] P. Meseure, A. Kheddar, and F. Faure. Détection de collisions et calcul de la réponse. *Technical report, Action Spécifique du CNRS N°90*, page 25, decembre 2003.

[OL03] M. A. Otaduy and M. C Lin. Clods : Dual hierarchies for multiresolution collision detection. *Proc. of the Eurographics Symposium on Geometry Processing, Aachen, Germany*, pages 94–101, 2003.

[Pet96] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. *IEEE 3rd International Conference on Evolutionary Computation, Nagoya*, may 1996.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. *IEEE International Conference on Robotics and Automation*, 1994.

[Rec64] I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment, Library Translation No. 1122, Farnborough, England*, 1964.

[RGF+04] L. Raghupathi, L. Grisoni, F. Faure, D. Marchal, M.-P. Cani, and C. Chaillou. An intestine surgery simulator : Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6) :708–716, Nov/Dec 2004.

[RKC02] S. Redon, A. Kheddar, and S. Coquillart. Hierarchical back-face culling for collision detection. *IEEE/RSJ Conference on Intelligent Robots and Systems, Lausanne*, october 2002.

[RKS01] S. Redon, A. Kheddar, and Coquillart S. Contact : In-between motions for collision detection. *IEEE International Workshop on Robot and Human Interactive Communication, RO-MAN, Bordeaux, France*, 2001.

[RKS02] S. Redon, A. Kheddar, and Coquillart S. Fast continuous collision detection between rigid bodies. *EUROGRAPHICS'02 Conference, Saarbruck, Germany*, 21(3), september 2002.

[Sch65] H. P. Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik. *Master's thesis, Technical University of Berlin*, 1965.

[SS89] G. Syswerda and W. M. Spears. Uniform crossover in genetic algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989.

[THM+03] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. *In Proceedings of Vision, Modeling, Visualization VMVŠ03*, 3 :47–54, 2003.

[TKZ+04] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, L Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann, and W. Strasser. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, pages 119–139, 2004.

[Tur89] G. Turk. *Interactive collision detection for molecular graphics*, volume Technical Report TR90-014. 1989.

[Van94] G. Vanecek. Back face culling applied to collision detection of polyhedra. *The Journal of Visualization and Computer Animation*, (5), 1994.

[VDB97] G. Van Den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools 2*, 4 :1–14, 1997.

[Yuk35] H. Yukawa. On the interaction of elementary particles. i. *Proc. Phys.-Math. Soc., Japan*, 17 :48–57, 1935.

[Zac01] G. Zachmann. Optimizing the collision detection pipeline. *1st International Game Technology Conference*, January 2001.