ORSAY
N° d'ordre : 6845

UNIVERSITE PARIS XI
U.F.R. SCIENTIFIQUE D'ORSAY

THESE

Présentée

Pour obtenir

Le GRADE de DOCTEUR EN SCIENCES

DE L'UNIVERSITE PARIS XI ORSAY

PAR

Benoît LEBLANC

Simulations Moléculaires de Monte Carlo : amélioration de l'efficacité statistique de l'échantillonnage grâce aux algorithmes d'évolution artificielle.

Soutenue le 27 mars 2002 devant la Commission d'examen

Mr. Bertrand Braunschweig	
Mr. Jin Kao Hao	rapporteur
Mme. Evelyne Lutton	directeur de thèse
Mme. Michèle Sebag	président de jury
Mr. Berend Smit	rapporteur
Mr. Hervé Toulhoat	

Remerciements

Mes remerciements vont en tout premier lieu à mes trois directeurs de thèse :
- Evelyne Lutton : ma directrice de thèse auprès de l'Université d'Orsay. En m'engageant comme stagiaire au projet FRACTALES en 1996, elle m'a donné la chance de découvrir le monde de la recherche et plus particulièrement celui de l'Evolution Artificielle.
- Bertrand Braunschweig : mon correspondant auprès de l'Institut Français du Pétrole. J'ai profité auprès de ce Directeur Expert d'une expertise dépassant largement celle du cadre de ma thèse.
- Hervé Toulhoat : mon directeur "officieux" de à l'IFP, qui a "manigancé" avec Evelyne et Bertrand ce diabolique sujet de thèse. Il a eu le courage de faire confiance à un informaticien pour aborder le problème de la simulation moléculaire.

Je souhaite leur dire combien je leur suis redevable de leurs compétences et surtout de la confiance et du soutien constant qu'ils m'ont apporté. Travailler avec eux a été un grand plaisir et j'espère bien trouver encore des occasions de le renouveler.

Ensuite je remercie mes rapporteurs, les professeurs Jin Kao Hao et Berend Smit. Ils ont eu la gentillesse, grâce à leurs compétences exceptionnelles dans leurs domaines, de porter des regards croisés et complémentaires sur cette thèse. Grâce à l'excellente qualité pédagogique de l'ouvrage dont Berend Smit est co-auteur [23], j'ai pu me familiariser rapidement avec les techniques de la simulation moléculaire. J'ai également pu profiter de son expérience lors du tutoriel qu'il a organisé avec Thijs Vlugt au CECAM à Lyon en octobre 2000, et je les remercie encore à cette occasion. Toujours dans le domaine de la simulation moléculaire je remercie les membres de l'équipe du professeur Theodorou de l'Université de Patras (Grèce), qui m'ont aussi fait profiter de leur expertise et m'ont apporté leur aide.

Je remercie Michèle Sebag, du Laboratoire de Recherche en Informatique à Orsay. Elle a eu la gentillesse d'accepter d'être membre du jury. Je tiens également à la remercier en tant que membre active de la communauté évolutionnaire française, fort conviviale et stimulante[1]. J'en profite pour remercier chaleureusement toutes les personnes touchant de près ou de loin à ce domaine et avec lesquelles j'ai eu un moment ou à un autre de fructueuses discussions : Jean Paul Allouche, Pierre Collet, Cyril Fontlupt, Olivier François, Pierre Liardet, Jean Louchet, Sophie Rochet, Marc Schoenauer ... et bien d'autres encore.

Je souhaite remercier Alain Bamberger pour m'avoir donné la chance de pouvoir effectuer ma thèse à l'Institut Français du Pétrole, au sein de la Division Informatique Scientifique et Mathématiques Appliquées. Mes remerciements vont également à tous les membres de la DISMA que j'ai côtoyé pour leur bon accueil, stagiaires, thésards, secrétaires, ingénieurs ...J'en profite pour remercier chaleureusement Philippe Ungerer, également de l'IFP, pour l'attention qu'il porte à mon travail et pour m'avoir fait rencontrer l'équipe du Laboratoire de Chimie Physique (LCP) d'Orsay, que je salue également.

Je réitère ici mon affection particulière à tous les membres du projet FRACTALES de l'INRIA Rocquencourt que j'ai eu le plaisir de côtoyer quotidiennement depuis 1996, et auprès desquels

[1] Au passage j'attire l'attention sur les Journées Evolutionnaires Trimestrielles (JET), http ://www.eeaax.polytechnique.fr/JET.html, où tous ceux qui s'intéressent de près ou de loin à l'Evolution Artificielle sont invités à venir se rencontrer.

j'ai eu l'occasion de me cultiver dans des domaines variés. Grâce à Jacques Lévy-Véhel et Evelyne Lutton j'y ai non seulement appris à apprécier l'analyse fractale, mais aussi celle-ci a pu m'être utile à diverses occasions. J'adresse de chaleureux remerciements à Nathalie Gaudechoux notre super assistante de projet. Enfin j'adresse des remerciements tous particuliers à mon condisciple depuis 1992, cothésard, copain, comélomane ... Frédéric Raynal.

Je remercie le laboratoire ID (Informatique Distribuée, http ://www-id.imag.fr/) de Grenoble pour m'avoir permis d'utiliser pour mes simulations la ferme de 225 PC de l'IMAG (Institut d'Informatique et Mathématiques Appliquées de Grenoble), cf section 4.2.1.

Enfin je finirai par remercier Hélène qui m'a apporté une soutien quotidien durant ma thèse et dont les adorables petits lutins, Ophélie et Gabriel, m'ont distrait de mes préocuppations moléculo-évolutionnaires de la façon la plus joyeuse qui soit. En eux je place mes plus belles espérances.

A la mémoire de mon père.

Table des matières

Chapitre 1

Introduction.

Avant l'utilisation de l'informatique, le seul moyen de prédire les propriétés d'une substance moléculaire était de considérer une description approximative de celle-ci afin de pouvoir appliquer les résultats théoriques (gaz parfaits par exemple). L'outil informatique permet maintenant de mettre en jeux des modèles atomistiques affinés qui autorisent une voie d'expérimentation numérique dont les résultats peuvent être directement opposables à ceux d'une expérimentation réelle, à bien moindre coût, bien meilleur délai. Il offre de plus à l'analyse une information d'une richesse inégalable. Et cette tendance se renforce naturellement par l'augmentation de la pratique informatique, de la puissance de calcul des ordinateurs et enfin et surtout par le développement des algorithmes de simulation. Ces derniers sont donc un sujet de recherche actif car tout gain d'efficacité accroît le champs d'application des simulations.

Il existe donc, depuis les débuts de l'informatique, un domaine appelé simulation moléculaire ayant pour but de simuler un ensemble de particules en interaction représentant un système physico-chimique. Le problème est d'obtenir pour cet ensemble une évaluation des propriétés macroscopiques caractéristiques par exemple de l'équilibre thermodynamique (densité, pression, composition des phases, ...). Ces propriétés s'obtiennent par des prises de moyennes étendues à l'ensemble des configurations possibles, pondérées par une densité d'état en fonction, par exemple, de l'énergie. Deux techniques sont possibles alors, la première étant la dynamique moléculaire ([1]), technique déterministe qui a pour principe d'appliquer à chacune des particules composant ce système les lois de la dynamique newtonienne. On simule ainsi l'évolution temporelle du mouvement des molécules, obtenant après un temps suffisant, un ensemble d'états qui constitue un échantillonnage valide des configurations possibles, et à partir desquelles les moyennes désirées peuvent être calculées. Cette technique montre toutefois ses limites lorsque l'on simule des systèmes composés de molécules complexes, et pour lesquelles les contraintes d'échantillonnages du temps simulé, la durée de ce temps simulé nécessaire pour obtenir un échantillon valide, mises en regard du temps de calcul d'un changement d'état, rendent prohibitif le temps de simulation totale.

Pour remédier à cette difficulté majeure, un second champ a fait son apparition dans la simulation moléculaire. Faisant cette fois-ci usage du hasard, la technique de Monte Carlo est maintenant appliquée à de nombreux domaines de simulation. Une telle simulation consiste en un échantillonnage aléatoire de l'espace des configurations, résultant d'une marche aléatoire gouvernée par une chaîne de Markov ergodique, définie sur l'espace des configurations et ayant pour distribution limite la distribution souhaitée. L'algorithme de Metropolis [51] met en oeuvre ce principe en faisant évoluer le système par des mouvement aléatoires des particules (chacun étant

suivi par une phase d'acceptation/rejet), le tout permettant de réaliser cette chaîne de Markov. Dans ce cadre également, on se heurte aux mêmes difficultés qu'en dynamique moléculaire pour la simulation des molécules complexes, mais une possibilité d'amélioration subsiste dans le fait que l'on dispose d'une relative liberté pour effectuer les changements d'états. En particulier ces "mouvements de Monte Carlo" font l'objet de nombreuses recherches ([62], [50], [65]), dans la mesure où l'on attend d'eux qu'ils puissent aider à fournir plus rapidement des états décorrélés, constituant un échantillonnage statistiquement valide. Il s'agit en d'autres termes d'améliorer l'efficacité statistique de l'échantillonnage.

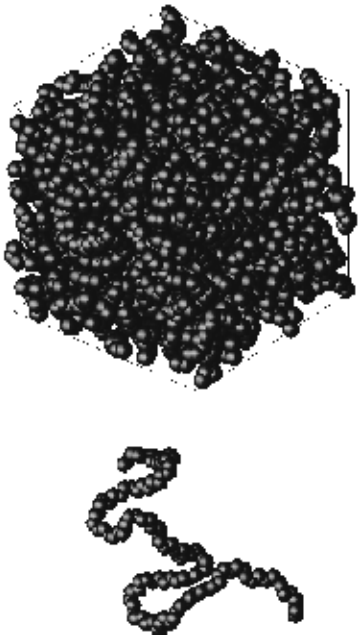
Parallèlement aux développements de ces applications, et suite à des travaux en biologie visant à simuler les mécanismes de l'évolution selon la théorie génétique moléculaire le domaine de l'évolution artificielle (EA) a fait son apparition, s'inspirant de ces premiers travaux pour ses mécanismes et sa terminologie. Ce domaine a connu un essor important depuis les années 1970 ([35], [56]) et 1980 ([27]), produisant des algorithmes d'optimisation stochastique, appliqués dans bien des domaines. De manière intéressante, et non sans un certain effet boomerang, ces algorithmes sont de plus en plus utilisés pour des problèmes d'optimisation en chimie ([38], [9], [15]). Citons brièvement l'identification des conformations géométriques d'énergie minimales ([5]), plus spécifiquement les problèmes de *docking* [16] (recherche des sites actifs sur des protéines), le problème de repliement de protéine [64] (*protein folding*), la conception de molécules *De novo* [26]... Par leur relative facilité d'emploi les algorithmes évolutionnaires (AE) ont donc déjà fait leurs preuves pour tous ces problèmes d'optimisation d'énergie.

Nous nous intéresserons dans ce mémoire aux possibilités offertes par l'évolution artificielle pour contribuer à améliorer l'efficacité statistique des simulations moléculaires de Monte Carlo, évoquée précédemment. Nous commencerons dans le chapitre 2 par présenter plus précisément à la fois les domaines dans lesquels s'inscrit notre travail et son objet. En particulier nous verrons les principes de la simulation des polymères amorphes en état dense, molécules complexes pour la simulation desquelles on rencontre précisément un problème d'inefficacité statistique, et qui serviront de modèle pour nos simulations. Puis au chapitre 3, nous détaillerons les paramètres susceptibles d'être optimisés pour améliorer l'efficacité de telles simulations afin d'en tirer des critères numériques explicites. Au chapitre 4, nous définirons un premier algorithme évolutionnaire (ayant fait l'objet d'une publication [44]) optimisant les fréquences relatives des différents mouvements de Monte Carlo utilisés conjointement lors de la simulation des polymères, et nous commenterons nos premières séries de simulations numériques. A la suite de ces premiers résultats nous proposerons au chapitre 5 une amélioration de cet algorithme en définissant l'Evolution Artificielle d'Individus Immortels (EAII, ayant fait l'objet d'une publication [43]). Le chapitre 6 fait alors une synthèse en appliquant l'EAII à l'optimisation des fréquences tout en explorant les possibilités offertes par tous les critères détaillés au chapitre 3. Enfin, nous verrons au chapitre 7 qu'il est également possible d'appliquer cet algorithme à la technique de la *thermalisation parallèle* (traduction de *parallel tempering*, décrite en section 2.4), pour optimiser non seulement les fréquences des mouvements, mais aussi les fréquences relatives des sous-systèmes simulés à différentes températures.

Chapitre 2

Algorithmes de Monte Carlo, Simulations Moléculaires et Evolution Artificielle.

Ce chapitre est destiné à présenter à la fois les domaines dans lesquels s'inscrit notre travail et son objet. Nous introduirons les algorithmes de Monte Carlo Markoviens et plus particulièrement la méthode de Metropolis en section 2.1. Nous verrons comment ces techniques sont utilisées dans le cadre de la simulation moléculaire, en commençant par le cas des systèmes particulaires (section 2.2) avant de présenter la simulation des polymères (section 2.3), avec un aperçu des différents mouvements de Monte Carlo existants ainsi qu'une présentation de la technique de la thermalisation parallèle en section 2.4. Puis nous introduirons les algorithmes d'Evolution Artificielle en section 2.5 et 2.6, afin d'exposer en section 2.7 la manière dont nous pensons qu'ils peuvent être utiles pour améliorer l'efficacité d'une simulation moléculaire.



2.1 Fondements des algorithmes de Monte Carlo Markoviens.

2.1.1 Monte Carlo

Les algorithmes de Monte Carlo peuvent être utilisés dans leur généralité comme méthode d'échantillonnage stochastique, et dans certains cas comme méthode d'optimisation stochastique (recuit simulé).

On considère un espace paramétrique Ω (discret ou continu) muni d'une distribution de probabilité π et une fonction f définie sur Ω. Le problème est alors d'obtenir un échantillon $(x_i)_{i=1...N}$ distribué suivant π pour estimer la quantité suivante :

$$F = \int_{x \in \Omega} f(x)\pi(x)dx$$

par la quantité :

$$\hat{F} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \ , \quad (x_i)_{i=1...N} \in \Omega$$

Lorsque l'on dispose d'un algorithme rapide permettant de tirer directement une valeur aléatoire de Ω suivant π, il suffit de tirer indépendamment et successivement les x_i. On parle alors de Monte Carlo *statique*. L'exemple trivial est le calcul d'une intégrale définie sur un segment $[a, b]$ de $\mathbb{R}$: il suffit alors de prendre N valeurs tirées aléatoirement uniformément sur $[a, b]$. Par contre dès que la dimension de l'espace devient trop élevée ces algorithmes sont inefficaces.

2.1.2 Chaînes de Markov

Lorsque, pour quelque raison que ce soit, π n'est pas "simulable" directement et simplement ou que la complexité de l'espace est trop élevée pour des tirages aléatoires directs, on peut alors recourir à un algorithme de *Monte Carlo Markovien* ou MCM (voir par exemple [37], [36]), aussi appelé Monte Carlo *dynamique*.

Nous rappelons donc succinctement qu'une *chaîne de Markov* est un processus stochastique, c'est-à-dire une suite de variables aléatoires, définies sur un même espace Ω, indexées par le temps. On dit qu'à chaque instant la chaîne se trouve dans un *état* x, de l'espace d'états Ω. On appelle alors *transition* le fait de passer à un nouvel état y qui ne dépendra alors que de l'état précédent, suivant une probabilité notée $P(x, y) = Pr(X_{t+1} = y | X_t = x)$. Si Ω est fini $P(x, y)_{x,y \in \Omega}$ est alors la matrice de transition, ou matrice markovienne. Par extension la probabilité de passer d'un état x à un état y en t transitions est notée $P^t(x, y)$, $(P^t(x, y)_{x,y \in \Omega}$ correspond alors effectivement à la t^{ieme} puissance de la matrice P). Lorsque que ces probabilités sont indépendantes du temps, on dit que la chaîne de Markov est homogène, cas que nous considérerons comme implicite par la suite.

Les chaînes de Markov permettent de modéliser beaucoup de systèmes dynamiques, et nous nous intéresserons ici à certaines d'entre elles qui sont dites *ergodiques*. Pour cela une telle chaîne doit être :

- *irréductible* : $\forall x, y \in \Omega, \exists t$ tel que $P^t(x, y) > 0$, autrement dit tout état est accessible à partir de n'importe quel état.
- *apériodique* : $pgcd \{t : P^t(x, y) > 0\} = 1$.

Dans ce cas il existe alors une unique *distribution stationnaire* π associée à la chaîne :

$$\forall x, y \in \Omega, \quad lim_{t \to \infty} P^t(y, x) = \pi(x) \tag{2.1}$$

Autrement dit, quelle que soit la distribution initiale, la chaîne de Markov tendra vers une unique distribution limite.

Le principe du MCM est alors de construire une chaîne de Markov ergodique de distribution stationnaire prescrite π sur l'espace d'états Ω. On échantillonne alors Ω en effectuant T transitions en partant d'un état x_0, tel que $\pi(x_0) > 0$.

L'ergodicité garantit d'avoir, pour un T assez grand et n'importe quel état initial x_0, une distribution observée arbitrairement "proche" de π. Bien sûr cette assertion n'est vérifiable que dans la mesure où l'on dispose d'une mesure entre distributions. Nous nous contenterons de dire ici que ce temps "minimal" d'atteinte de la distribution limite est appelé *temps de mélange*, et qu'il est accessible en raisonnant sur les propriétés de la chaîne (seconde valeur propre de la matrice markovienne, méthode des *chemins canoniques*, ...). Evidemment ce temps est crucial pour l'efficacité de l'algorithme, car plus il sera court moins il sera coûteux d'obtenir un échantillon valide.

2.1.3 Metropolis

La problématique

En simulation moléculaire ([1], [23]) les algorithmes de MCM ont été utilisés régulièrement comme méthode alternative à la dynamique moléculaire, dans le cas où celle-ci est insuffisamment efficace. La méthode fondatrice est celle de Metropolis ([51]). Le but de la simulation moléculaire correspond tout à fait au cadre des simulations de Monte Carlo, car il s'agit de calculer la valeur moyenne d'un observable $< A >$ sur l'ensemble des états d'un système physico-chimique :

$$< A >= \frac{\int_{x \in \Omega} A(x) exp(-\beta U(x)) dx}{\int_{y \in \Omega} exp(-\beta U(y)) dy} \tag{2.2}$$

où Ω décrit l'espace des phases du système, en l'occurrence uniquement les coordonnées spatiales de ses constituants, contrairement à la dynamique moléculaire où les moments sont aussi pris en compte. La distribution associée est celle de Boltzmann-Gibbs :

$$\forall x \in \Omega, \quad \pi(x) = \frac{exp(-\beta U(x))}{\int_{y \in \Omega} exp(-\beta U(y)) dy} \tag{2.3}$$

où $\beta = \frac{1}{k_B T}$ (k_B : constante de Boltzmann, T : température du système) et $U(x)$ est l'énergie du système.

Le problème qui se pose alors est que si l'énergie d'un état d'un système est accessible par le calcul, le terme normalisateur nécessite de connaître l'énergie de tous les états. On ne peut donc générer simplement et rapidement un échantillon valide.

La méthode

Pour appliquer un algorithme de MCM, l'idée est alors de contrôler explicitement les transitions pour respecter la distribution de Boltzmann sans avoir à en calculer le terme normalisateur (la méthode est applicable et appliquée en dehors du cadre de la simulation moléculaire). Pratiquement on peut décrire l'algorithme de la manière suivante :

1 Choisir un point de départ $x \in \Omega$.
2 Choisir aléatoirement un nouvel état y dans le voisinage de x. Pratiquement, lorsque Ω est un espace continu, ce nouvel état est généré à partir de l'ancien en appliquant une perturbation aléatoire à l'état courant : on parle de *mouvement de Monte Carlo*. C'est alors cette manière de "perturber" qui définit le voisinage d'un état.
3 Accepter y comme nouvel état courant avec une probabilité $acc(x, y)$ (égale à 1 si $U(y) < U(x)$, moins sinon) et retourner en 2 jusqu'à un critère d'arrêt (nombre de valeurs pour un échantillon, minimum d'énergie atteint dans le cas de l'optimisation).

Donc la transition est décomposée en construction-acceptation et si l'on note $\alpha(x, y)$ la probabilité de générer y à partir de x alors on a :

$$P(x, y) = \alpha(x, y) * acc(x, y) \tag{2.4}$$

Si la méthode de génération α rend tous les états accessibles entre eux, la chaîne est irréductible et l'apériodicité est garantie s'il est toujours possible de générer un nouvel état qui soit rejeté, ce qui implique que pour tout $x \in \Omega, P(x, x) > 0$. Dans ces conditions l'ergodicité de la chaîne est garantie, mais il reste à s'assurer que la distribution limite π est celle souhaitée, et pour cela la condition dite de *micro-réversibilité* est suffisante :

$$P(x, y)\pi(x) = P(y, x)\pi(y) \tag{2.5}$$

Si α est symétrique ($\alpha(x, y) = \alpha(y, x)$) on a alors :

$$acc(x, y)\pi(x) = acc(y, x)\pi(y) \quad \text{soit} \quad \frac{acc(x, y)}{acc(y, x)} = \frac{\pi(y)}{\pi(x)} \tag{2.6}$$

Et dans le cas de la distribution de Boltzmann-Gibbs, en notant $\Delta U = U(y) - U(x)$:

$$\frac{acc(x, y)}{acc(y, x)} = exp(-\beta(U(y) - U(x))) = exp(-\beta\Delta U) \tag{2.7}$$

La solution proposée par Metropolis est alors de prendre :

$$acc(x, y) = min(1, exp(-\beta\Delta U)) \tag{2.8}$$

Ajoutons tout de suite que l'on peut utiliser dans ce cadre une procédure de construction α non symétrique (voir le *preferential sampling* dans [1] p 220), ce qui implique alors :

$$\frac{\alpha(x, y) * acc(x, y)}{\alpha(y, x) * acc(y, x)} = \frac{\pi(y)}{\pi(x)} \tag{2.9}$$

avec toujours comme solution possible :

$$acc(x, y) = min\left(1, exp(-\beta\Delta U) * \frac{\alpha(y, x)}{\alpha(x, y)}\right) \tag{2.10}$$

La quantité $\left(\frac{\alpha(y,x)}{\alpha(x,y)}\right)$ peut être vue comme un *biais de construction* car elle biaise la règle d'acceptation classique. Cette méthode est d'ailleurs appliquée dans les mouvements de Monte Carlo (transitions de la chaîne de Markov correspondante) suivants (expliqués plus en détail à le section 2.3.3) :

- Recroissance de chaîne (*Configurational bias Monte Carlo*, [62]) : ce mouvement utilisé pour la simulation moléculaires de polymères amorphes essaie de faire croître une chaîne monomère par monomère en cherchant localement une position de basse énergie. Il introduit donc un biais de construction connu sous le nom de facteur de Rosenbluth ([58]).
- Repontage inter-chaînes (*End-Bridging Monte Carlo*, [55], [50]) : utilisé dans le même contexte ce mouvement revient à couper une chaîne en son milieu et à en raccorder un bout à une autre chaîne dont l'extrémité est proche. Pour cela il est nécessaire de passer par une résolution géométrique qui implique de prendre en compte un biais (rapport de Jacobien de passage des coordonnées cartésiennes à des coordonnées contraintes).

Enfin, dans beaucoup de problèmes et en particulier en simulation moléculaire, on peut utiliser plusieurs mouvements différents pour générer de nouveaux états. Il est possible de les utiliser simultanément, mais il est alors nécessaire de ne pas les appliquer dans un ordre particulier mais de les choisir aléatoirement afin de respecter la condition de micro-reversibilité. En effet un ordre déterminé impliquerait une matrice α non symétrique.

2.2 Simulation moléculaire de systèmes de particules.

Nous résumons ici les principes essentiels de la simulation moléculaire en prenant exemple sur la simulation d'un systèmes de molécules rigides non orientées (en ne considérant qu'une seule espèce) qui peuvent être modélisées par des particules.

2.2.1 Généralités sur les ensembles.

Avant de pouvoir effectuer une simulation, il est important de définir clairement l'espace que l'on cherche à échantillonner. Dans la cadre de la mécanique statistique, où l'on souhaite calculer les propriétés macroscopiques d'un système à partir des informations microscopiques sur ses constituants, cet espace est défini par les conditions thermodynamiques que l'on choisit de lui imposer. Les configurations possibles définissent alors un *ensemble*, qui n'est autre que l'espace à échantillonner, qui est alors nommé en fonction de ces paramètres imposés. Ces paramètres sont typiquement :

- N : le nombre de molécules
- V : le volume
- P : la pression
- T : la température
- E : l'énergie totale
- μ : le potentiel chimique
- ρ : la densité

Les ensembles sont définis en fixant certaines de ces quantités, les autres devant être calculées lors de la simulation. Les ensembles classiques sont les suivants :

- Ensemble NVE ou ensemble *microcanonique*. Généralement considéré lors de la dynamique moléculaire. Par définition il ne peut être utilisé pour une simulation de MCM.
- Ensemble NVT ou ensemble *canonique*. Dans ce cas, la pression et l'énergie font partie des quantités mesurées.
- Ensemble NPT ou ensemble *isothermal-isobarique*. Ici l'énergie et le volume occupé par le système sont mesurés.
- Ensemble μVT, ou ensemble *grand canonique*. On mesure en général l'énergie, la pression et la densité.

Pour chacun des ces ensembles, les probabilités associées aux configurations possibles s'exprimeront différemment en fonction des quantités fixées, comme nous le verrons par la suite.

2.2.2 Les composants de la simulation.

La boîte de simulation

Le point de départ d'une simulation moléculaire par algorithme de Metropolis consiste donc à choisir une configuration initiale. Pratiquement on considère une boîte cubique dans laquelle sont placées les molécules. Ne prenant pas en compte les moments de molécules (comme lors d'une simulation par dynamique moléculaire), mais uniquement les coordonnées spatiales, on a alors $3N$ degrés de liberté pour spécifier complètement une telle configuration. Pour simuler l'étendue de la matière, on définit des images périodiques de cette boîte dans les 3 directions (dans lesquelles sont placées des images périodiques des molécules de la boîte de référence) comme l'illustre la figure 2.1

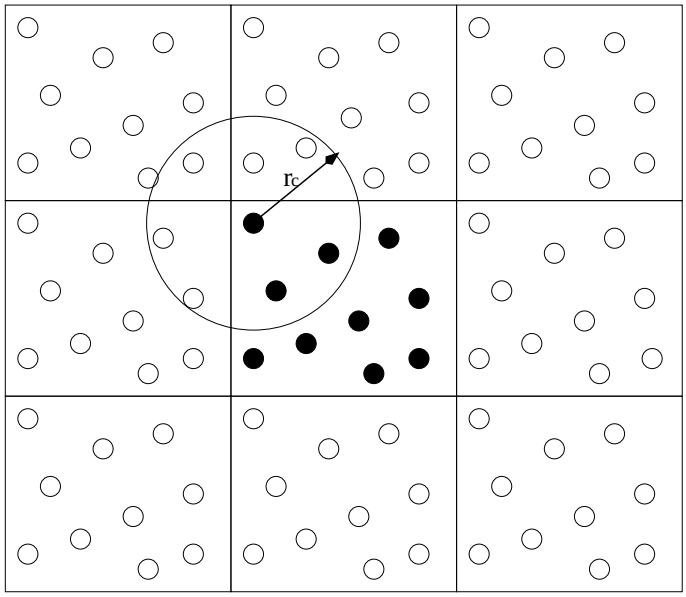


FIG. 2.1 – Images périodiques de la boîte de simulation en 2 dimensions. Les ronds noirs désignent les particules (dites particules originelles) dont les coordonnées sont les coordonnées de référence, réellement gardées en mémoire. Les ronds blancs désignent les images périodiques de ces particules, dont les coordonnées sont obtenues par des décalages dont l'amplitude sur un axe est un multiple du côté de la boîte de simulation. Lors du calcul de l'énergie d'interaction entre les particules, seules les distances courtes (inférieures à un rayon de coupure r_c) sont réellement prises en compte. Comme on le voit alors pour la particule sur laquelle un cercle est centré, il faut à la fois tenir compte des particules originelles et des images périodiques pour calculer l'énergie d'interaction.

L'énergie du système

L'énergie potentielle des particules est alors la somme de l'énergie d'un éventuel champ externe appliqué à chacune des particules et de l'énergie d'interaction entre les particules. Cette dernière est en général bien approximée si on se limite aux interactions de paires de particules, qui sont elles-mêmes modélisées par le potentiel de Lennard Jones, fonction de la distance r entre 2 particules :

$$u_{LJ}(r) = 4 \times \epsilon_{LJ} \times \left(\left(\frac{\sigma_{LJ}}{r} \right)^{12} - \left(\frac{\sigma_{LJ}}{r} \right)^{6} \right) \tag{2.11}$$

On voit sur le graphe de la figure 2.2 qu'en dessous de la distance σ_{LJ}, il y a une répulsion très forte avec le terme $\left(\frac{1}{r^{12}}\right)$, un puits de potentiel de profondeur $-\epsilon_{LJ}$ situé à $r = \sqrt[6]{2}$, puis un terme d'attraction à longue distance en $\left(-\frac{1}{r^6}\right)$. Les paramètres σ_{LJ} et ϵ_{LJ} sont évidemment caractéristiques de l'espèce considérée.

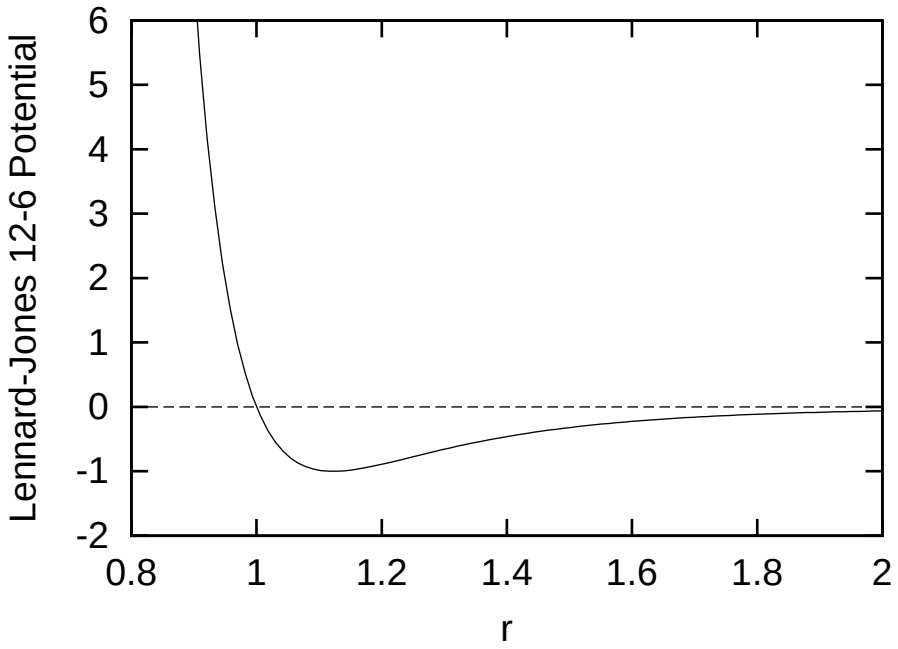


FIG. 2.2 – Potentiel de Lennard Jones en unités réduites. En abcisses : 1 unité = σ_{LJ}, en ordonnées : 1 unité = ϵ_{LJ}

Du fait de la propriété de périodicité de la boîte de simulation (figure 2.1), lors du calcul de l'énergie potentielle d'interaction d'une particule i parmi les N comprises dans la boîte de simulation, il faut tenir compte des interactions avec toutes les autres particules $j \neq i$. L'énergie potentielle est en effet fonction de la distance r_{ij}, mais également de toutes leurs images périodiques avec lesquelles les distances sont de la forme $(r_{ij} + nL)$ où L désigne le diamètre de la boîte de simulation. L'énergie potentielle d'interaction totale est alors donnée par :

$$U_{LJ}^{tot} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{n=-\infty}^{+\infty} u_{LJ}(r_{ij} + n \times L) \tag{2.12}$$

En raison du comportement en $\frac{-1}{r^6}$ du potentiel de Lennard Jones lorsque que r est grand, seules les interactions entre particules proches sont importantes. Pratiquement on ne calcule alors que les interactions entre les molécules ou images de molécules séparées d'une distance inférieure à une distance de coupure r_c. On ajoute finalement un terme correcteur représentatif des interactions à longue distance qui se détermine à partir de la densité ρ de particules par unité de volume ([23], page32) :

$$\begin{aligned} u_{cor}(r_c,\rho) &= \frac{1}{2}\ 4\ \pi\ \rho\ \int_{r_c}^{+\infty} r^2\ u_{LJ}(r)\ dr \\ &= \frac{1}{2}\ 16\ \pi\ \rho\ \epsilon_{LJ} \int_{r_c}^{+\infty} r^2 \left[\left(\frac{\sigma_{LJ}}{r_c}\right)^{12} - \left(\frac{\sigma_{LJ}}{r_c}\right)^{6} \right]\ dr \\ &= \frac{8}{3}\ \pi\ \rho\ \epsilon_{LJ}\ \sigma_{LJ}^3 \left[\frac{1}{3} \left(\frac{\sigma_{LJ}}{r_c}\right)^{9} - \left(\frac{\sigma_{LJ}}{r_c}\right)^{3} \right] \end{aligned}$$

Et en définissant l'ensemble fini :

$$v_{r_c}(r^N) = \{i,j \in \{1,...N\}, n \in \mathbf{Z}\ \ /\ \ j>i,\ \ (r_{ij}+n\times L) < r_c\}$$

on a finalement :

$$U_{LJ}^{tot} = \sum_{(i,j,n)\in v_{r_c}(r^N)} u_{LJ}(r_{ij}+n\times L)\ \ +\ \ N \times u_{cor}(r_c,\rho) \tag{2.13}$$

L'énergie cinétique des particules n'est pas prise en compte lors d'une simulation de Monte Carlo étant donné que les moments des particules ne sont pas spécifiés.

Le mouvement de translation

Dans le cas d'un système de particules le mouvement de Monte Carlo le plus évident revient à choisir une particule aléatoirement et à la déplacer dans une direction aléatoire d'une distance aléatoire. Pour rendre ce mouvement symétrique (la probabilité de déplacer une particule donnée d'un point à un autre doit être égale à celle du mouvement inverse) il suffit de choisir la particule à déplacer avec une probabilité uniforme ($\frac{1}{N}$) et d'effectuer un déplacement aléatoire symétrique, par exemple en se fixant un paramètre Δ_{max}. Une particule est déplacée d'un point $r=(x,y,z)$ à un point $r'=(x',y',z')$ déterminé de la manière suivante :

$$r' = \left\{ \begin{array}{rcl} x' &=& x + U_{[-1:1]}(\omega_x) \times \Delta_{max} \\ y' &=& y + U_{[-1:1]}(\omega_y) \times \Delta_{max} \\ z' &=& z + U_{[-1:1]}(\omega_z) \times \Delta_{max} \end{array} \right.$$

où $U_{[-1:1]}(\omega)$ désigne un tirage aléatoire suivant une loi uniforme sur l'intervalle $[-1:1]$.

Dans le cas d'une simulation dans l'ensemble NVT, la probabilité de trouver N particules dans une configuration donnée r^N est proportionnelle à :

$$exp\left(-\beta U(r^N)\right)$$

et dans l'ensemble NPT, la probabilité de trouver N particules dans une configuration donnée occupant un volume donné V est proportionnelle à :

$$V^N exp(-\beta PV) \times exp\left(-\beta U(r^N)\right)$$

La translation n'impliquant pas de changement de volume, la probabilité d'acceptation de la nouvelle position sera alors la même dans les deux ensembles :

$$acc(r^N, r'^N) = exp\left(-\beta\left(U(r'N) - U(r^N)\right)\right)$$

La fluctuation de volume

Dans un ensemble NPT, il est nécessaire de laisser fluctuer le volume de la boîte de simulation afin que celui-ci corresponde aux conditions imposées. Cette fluctuation est alors un mouvement de Monte Carlo proprement dit. Pratiquement une dilatation ou une contraction de la boîte de simulation est effectuée, en déplaçant les molécules proportionnellement au changement de volume de manière à ce que leur disposition générale demeure inchangée. Pratiquement si l'on considère que la largeur de la boîte est multipliée par un taux donné, alors toutes les distances entre molécules seront multipliées par ce même taux.

Pour le mouvement de fluctuation volumique, passant d'un état (r_0^N, V_0) à (r_1^N, V_1) par un changement de volume symétrique en V :

$$V_1 = V_0 + \Delta V \tag{2.14}$$

avec ΔV tiré uniformément sur $[-\Delta_{max}, \Delta_{max}]$ on a :

$$\frac{\rho(r_1^N, V_n)}{\rho(r_0^n, V_o)} = exp\left(-\beta\left[U(r_1^N, V) - U(r_0^N, V) + P(V_1 - V_0) + N\beta^{-1}ln(V_1/V_0)\right]\right) \tag{2.15}$$

En revanche si le changement de volume se fait sur une échelle logarithmique (méthode que nous emploierons ici) :

$$ln(V_1) = ln(V_0) + \Delta lnV \tag{2.16}$$

avec ΔlnV tiré uniformément sur un intervalle $[-\Delta_{max}, \Delta_{max}]$, la densité de probabilité de trouver le système dans une configuration spatiale r à un volume V devient proportionnelle à :

$$\rho(r, V) = V^{N+1}exp(-\beta PV)exp[-\beta U(r, V)] \tag{2.17}$$

et la probabilité d'acceptation devient alors :

$$\begin{aligned} acc\left((r_0^N, V_0), (r_1^N, V_1)\right) \quad &= min\left\{1, exp\left(-\beta \quad \left[U(r_1^N, V_1) - U(r_0^N, V_0)\right.\right.\right. \\ &\qquad \left.\left.\left. + P(V_1 - V_0) + (N+1)\beta^{-1}ln(V_1/V_0)\right]\right)\right\} \end{aligned} \tag{2.18}$$

2.2.3 Récapitulatif.

Pour un système de molécules pouvant être décrit par un système particulaire, une simulation moléculaire par algorithme de Monte Carlo markovien suivant la méthode de Metropolis, peut se résumer ainsi :

- Un état du système correspond à une configuration spatiale donnée d'un ensemble de particules réparties dans une boîte de simulation pour laquelle on définit une propriété de périodicité, comme décrit précédemment.
- La probabilité d'existence d'un état est fonction de l'énergie du système (équation 2.3).

- L'algorithme consiste en une marche aléatoire sur l'espace des configurations en accord avec la distribution de Boltzmann-Gibbs. L'ensemble des états visités constitue alors un échantillonnage de cet espace en accord avec la distribution prescrite.
- Etant donné que les quantités de mouvement ne sont pas spécifiées, seule l'énergie d'interaction (équation 2.13) différencie les énergies de deux configurations différentes.
- La marche aléatoire proprement dite s'effectue, partant d'une configuration initiale, par une succession de choix de nouvelles configurations à partir d'une configuration courante et d'acceptation/rejet de celles-ci suivant la méthode de Metropolis (section 2.1.3) .
- Selon les conditions thermodynamiques imposées au système, on définit différents types d'espaces de configuration, ou *ensembles* dénommés selon les variables thermodynamiques fixées.
- Les types de perturbations aléatoires applicables dépendront de ces ensembles. Le plus simple pour un système de particules est le déplacement aléatoire borné d'une particule. Lorsque qu'une pression fixe est imposée au système, une fluctuation aléatoire de la taille de la boîte de simulation permet de faire varier son volume, et donc sa densité.

2.3 Simulations des polymères.

Maintenant que le principe de simulation moléculaire pour des molécules simples a été exposé, nous pouvons aborder la simulation des polymères [63]. Le principe de simulation ne change pas, en revanche les polymères nécessitent une modélisation plus fine. Nous verrons donc qu'il en résulte de nouveaux termes d'énergie à prendre en compte, et surtout nous verrons qu'il existe une grande variété de mouvements de Monte Carlo. Cette diversité de mouvements confrontée avec les critères d'évaluation habituels nous amènera à nous poser la question de l'efficacité d'une telle simulation.

2.3.1 Modèle moléculaire.

La diversité des types de polymères interdit l'existence d'un unique modèle simple applicable à tous. Nous nous limiterons donc ici à décrire un modèle de polymères linéaires pour lesquels on peut faire l'hypothèse que chaque monomère présente des propriétés identiques. Par ailleurs, pour les simulations numériques qui seront présentées, nous utiliserons un modèle du polyéthylène déjà décrit par Theodorou et al ([55], [50]), ainsi que ses valeurs numériques.

- **Modèle d'atome unifié** : chaque monomère est considéré comme un seul site actif décrit par ses coordonnées spatiales. Ici chaque groupe CH_2, CH_3, monomères du polyéthylène, sera considéré comme tel.
- Le potentiel d'interaction de **Lennard Jones** (LJ, voir l'équation 2.11) est utilisé pour les sites séparés par plus de trois liaisons sur la même chaîne et pour les interactions intermoléculaires. Pour le cas du polyéthylène les valeurs numériques suivantes sont utilisées :
  - Le diamètre est $\sigma_{LJ} = 3.94$ Å
  - Le puits de potentiel $\epsilon_{LJ} = 0.098\ kcal/mol$
- Les **longueurs de liaisons** entre deux monomères sont fixes. Pour le polyéthylène on a $l = 1.54$ Å.
- Le potentiel de tension de **Van der Ploeg et Berendsen** : pour un angle de liaison θ pouvant fluctuer autour de la valeur moyenne $\theta_0 = 112^o$, est le suivant ($k_\theta = 57950K.rad^{-1}$) :

$$\frac{\vartheta_{VPB}(\theta)}{k_B} = \frac{1}{2} \times k_\theta \times (\theta - \theta_0)^2 \tag{2.19}$$

- Le potentiel de torsion de **Ryckaert et Bellmans** pour un angle de torsion Φ (défini pour 4 monomères consécutifs sur la même chaîne) avec $c_0 = 1116\ K$, $c_1 = 1462\ K$, $c_2 = -1578\ K$, $c_3 = -368\ K$, $c_4 = 3156\ K$, $c_5 = -3788\ K$:

$$\frac{\vartheta_{tor}}{k_B} = \sum_{k=0}^{5} c_k cos^k(\phi) \tag{2.20}$$

L'énergie potentielle totale est alors la somme de l'énergie d'interaction, de l'énergie de tension et de l'énergie de torsion.

2.3.2 Les ensembles.

Pour décrire les ensembles représentant les espaces de configurations possibles il faut désormais tenir compte du nombre de monomères n et du nombre de chaînes N. Lorsque toutes les chaînes ont le même nombre de monomères $\frac{n}{N}$, les ensembles sont analogues au cas particulaire.

Par contre si l'on choisit simuler un système de distribution de masse moléculaire de support non ponctuel, autrement dit dont l'indice de polydispersité [1] est supérieur à 1, indiquant de masses différentes, il faut alors tenir compte dans la simulation d'un spectre de potentiels chimiques réduits μ^* [55] contrôlant les flux entre les chaînes de masses différentes.

Theodorou et al. ont alors défini l'ensemble $nN\mu^*PT$ pour simuler un tel système de polymères. On considère alors la coexistence de plusieurs espèces chimiques correspondant aux chaînes de masses différentes, et on notera N_k le nombre de chaînes contenant k monomères (nous dirons alternativement chaîne de longueur k, cette longueur mesurant le nombre de monomères de la chaîne et non la distance séparant les deux extrémités de la chaîne) et μ_k leurs potentiels chimiques. Si les longueurs de chaînes sont comprises entre k_{min} et k_{max}, on a alors par définition de l'ensemble les contraintes suivantes :

$$\sum_{k=k_{min}}^{k_{max}} N_k = N \quad \text{et} \quad \sum_{k=k_{min}}^{k_{max}} kN_k = n \tag{2.21}$$

Les potentiels chimiques réduits ([55]) sont alors définis en fixant deux espèces de longueur (i,j) comme espèces de référence :

$$\mu_k^* = \mu_k - \left(\frac{k-i}{j-i}\right)\mu_j - \left(\frac{k-j}{i-j}\right)\mu_i \tag{2.22}$$

La densité de probabilité (établie en [55]) pour cet ensemble est alors proportionnelle à la quantité suivante :

$$\rho(r,V) = V^N exp\left[\beta \sum_{k=k_{min}}^{k_{max}} \mu_k^* N_k - \beta PV - \beta U(r,V)\right] \tag{2.23}$$

On remarque donc que si un mouvement dans cet ensemble n'implique pas de changements dans la distribution des longueurs de chaînes, sa probabilité d'acceptation est identique à celle de l'ensemble $nNPT$. Par contre dans le cas contraire, il est nécessaire de tenir compte du spectre (μ_k^*).

2.3.3 Quelques mouvements.

Nous proposons ici un aperçu des principaux mouvements de Monte Carlo utilisables pour la simulation des polymères linéaires.

[1] L'indice de polydispersité caractérise la largeur de la distribution de masse d'un polymère, il est au minimum de 1 lorsque toutes les chaînes ont la même masse, soit le même nombre de monomères. Il est défini par le rapport $\frac{M_n}{M_w}$, où M_n et M_w sont respectivement les masses moyennes en nombre et en masse :

$$M_n = \frac{\sum_i N_i M_i}{\sum_i N_i} \quad \text{et} \quad M_w = \frac{\sum_i N_i M_i^2}{\sum_i N_i M_i}$$

où les (N_i, M_i) désignent les fractions des composants distincts et leurs masses, considérant deux polymères de masses différentes comme des composants distincts

La translation, figure 2.3 gauche

L'ensemble de la molécule est déplacé de manière rigide sans changer d'orientation. Là encore ce déplacement se fait pour une molécule choisie aléatoirement sans biais, et d'une distance inférieure à un Δ_{max} habituellement réglé dynamiquement au cours de la simulation de manière à ce que le taux de succès observé pour ce mouvement se rapproche d'un taux prescrit. Il est évident que plus les polymères sont longs, plus ce mouvement est coûteux à effectuer [2], et moins il a de chance de réussir, du fait qu'il est fréquent qu'un recouvrement [3] en résulte. Dans ce dernier cas, un Δ_{max} adaptatif baisse rapidement vers une valeur très petite, ce qui se traduit par de tout petits déplacements, qui, lorsqu'ils sont acceptés, changent assez peu la configuration tout en nécessitant un temps de calcul important.

La rotation, figure 2.3 droite

Un monomère en bout de chaîne subit une rotation sur la sphère centrée sur le monomère le précédant, et dont le rayon est la distance de liaison, qui reste fixe. La variation d'énergie intervient sur les 3 potentiels. Ce mouvement simple ne déplace qu'un seul monomère et est donc un des plus rapides à exécuter.

La reptation, figure 2.4 gauche

Un monomère en bout de chaîne est supprimé et est ajouté à l'autre extrémité de cette même chaîne de manière similaire à une rotation. Ici également la variation d'énergie intervient sur les 3 potentiels.

Le flip, figure 2.4 droite

Un monomère interne à une chaîne subit une rotation autour de l'axe des deux monomères l'encadrant. Ce qui implique, outre le déplacement du site, le changement de deux angles de tension et de quatre angles de torsion. L'angle de rotation est tiré aléatoirement uniformément dans l'intervalle $]0, \phi_{max}[$, ϕ_{max} étant classiquement réglé dynamiquement pour obtenir un taux de succès prescrit.

La fluctuation de volume, figure 2.5 droite

Ce mouvement devient moins trivial que dans le cas particulaire car les distances entre monomères d'une même chaîne ne peuvent être contractées ou dilatées à cause de la contrainte de liaison. Une solution est de déplacer de manière rigide toutes les chaînes de manière à ce que les rapports des distances de leur centre de masse à l'origine sur la largeur de la boîte de simulation reste constant. La transformation est homothétique pour les centres de masse mais ne l'est pas pour chacun des monomères (du fait de la contrainte de la géométrie interne) ce qui rend nécessaire de recalculer l'ensemble de l'énergie d'interaction, celle-ci dépendant justement des distances relatives des sites d'interaction qui sont les monomères.

[2] Tous les monomères d'une chaîne sont déplacés simultanément, ce qui implique de recalculer pour chacun le potentiel de Lenard-Jones avec tous les monomères des autres chaînes

[3] Deux monomères se chevauchant, c'est-à-dire dont la distance est inférieure à σ_{LJ}.

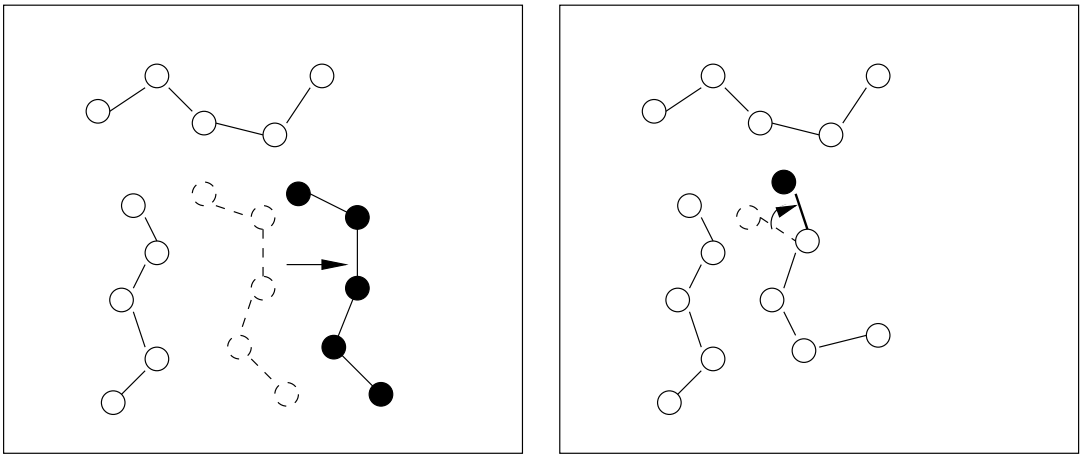


FIG. 2.3 – Translation (gauche) et Rotation (droite)

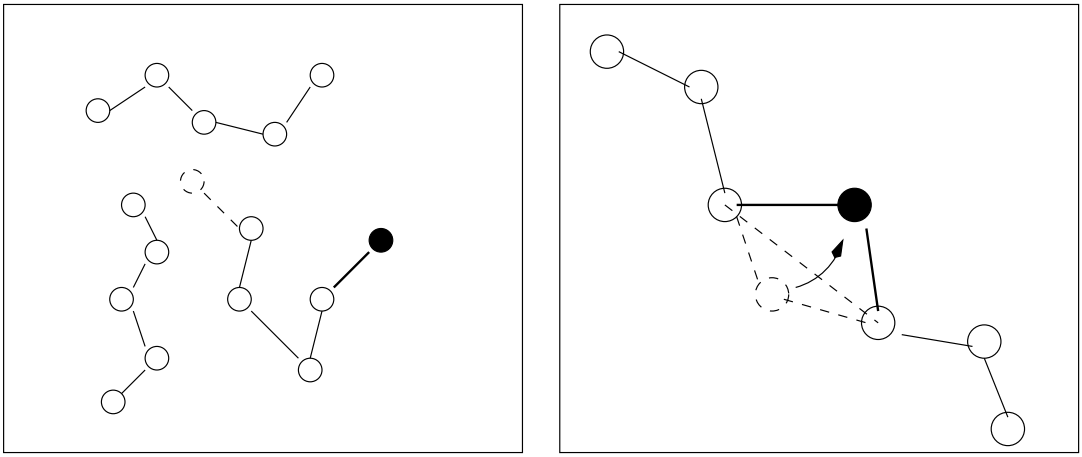


FIG. 2.4 – Reptation (gauche) et Flip (droite)

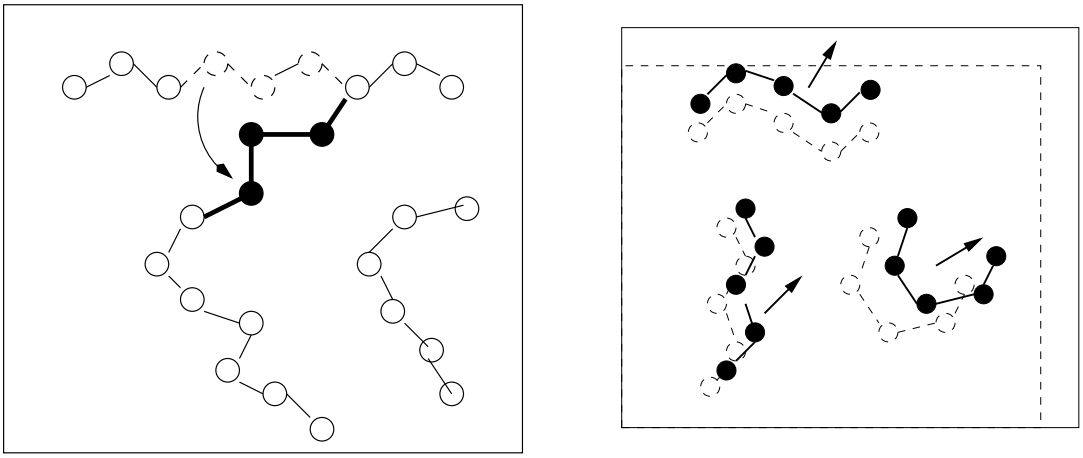


FIG. 2.5 – Pontage IC (gauche) et Fluctuation de volume (droite)

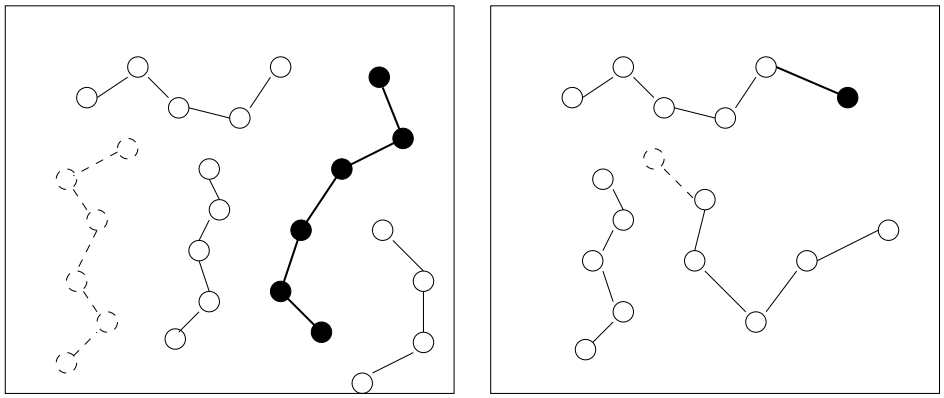


FIG. 2.6 – Recroissance de chaîne (gauche) et Reptation inter-chaînes (droite)

La recroissance, ou biais configurationnel, figure 2.6 gauche

Ce mouvement consiste à enlever une chaîne complète de la boîte puis de la faire croître à partir d'une nouvelle position choisie. La technique de croissance introduite par Rosenbluth et Rosenbluth [58] peut se résumer ainsi :

1 Tirer aléatoirement plusieurs positions dans la boîte et en choisir une aléatoirement avec un biais favorisant la plus stable énergétiquement comme position de départ.

2 Chaque monomère suivant est ajouté en générant plusieurs positions possibles à partir du monomère précédent (contrainte de longueur de chaînes) et en en choisissant une aléatoirement avec encore un biais favorisant la plus stable énergétiquement.

Cette technique a donc été utilisée pour construire un mouvement de Monte Carlo par Siepmann et Frenkel [62]. Dans ce but il est nécessaire de tenir compte du biais de construction (appelé facteur de Rosenbluth) de la nouvelle configuration ainsi que de l'ancienne dans la probabilité d'acceptation du mouvement (voir l'équation 2.10). On constate alors que plus la chaîne à faire croître est longue, plus la probabilité qu'elle occasionne un recouvrement [4] augmente et donc moins le mouvement a de chances d'être accepté. Une méthode alternative pour les longues chaînes consiste alors à n'en faire recroître qu'une partie, soit uniquement l'étape 2.

Améliorant l'efficacité de cette démarche, le mouvement de *Recoil Growth* ([10], [11]) réduit les chances que la chaîne ainsi construite se trouve prise dans un cul-de-sac, ce qui arrive lorsqu'à partir d'une position donnée il n'est plus possible de rajouter de monomères sans qu'il y ait un recouvrement. La solution proposée est alors de faire une recherche arborescente : faire croître la chaîne comme précédemment, puis si un cul-de-sac se présente revenir en arrière en essayant un autre. Encore une fois cette construction particulière entraîne un biais qui doit alors figurer dans la probabilité d'acceptation de ce mouvement.

Le Pontage Inter-chaînes, figure 2.5 gauche

Ce mouvement sera appelé *Pontage IC* par la suite (traduction de *end-bridging*), mouvement décrit par [55]et [50]. Il s'agit alors de couper une chaîne et d'en raccorder une moitié à l'extrémité d'une autre chaîne proche du point de coupure. Pour ce faire 3 monomères sont déplacés, leur

[4] Deux particules en répulsion très forte, autrement dit dont la distance est inférieure à la distance caractéristique du potentiel de Lennard Jones σ_{LJ}.

nouvelles positions étant calculées en résolvant un problème géométrique complexe : les longueurs de liaisons et les angles de liaisons sont conservés. Ce mouvement, lorsqu'il réussit, altère donc la connectivité des chaînes, et permet de simuler une système de distribution de masse moléculaire de support non ponctuel, autrement dit dont l'indice de polydispersité est supérieur à 1. On peut par exemple considérer une distribution uniforme dans un intervalle centré autour de la moyenne, décrite par le couple $(\bar{X}, \Delta)$, soit $[\bar{X}(1-\Delta), \bar{X}(1+\Delta)]$. Comme nous l'avons vu l'ensemble considéré doit alors inclure un spectre de potentiels chimiques réduits μ^* contrôlant les flux entre les chaînes de longueur différentes. Comme il est montré en [50], le spectre correspondant à une distribution uniforme bornée est alors le suivant :

$$\mu_k^* = \begin{cases} -\infty & \text{pour } k < k_{min} \text{ et } k > k_{max} \\ 0 & \text{pour } k_{min} \leq k \leq k_{max} \end{cases} \tag{2.24}$$

Afin de déterminer le biais de construction du mouvement il est nécessaire de toujours effectuer le mouvement inverse. La résolution géométrique du positionnement des 3 monomères résulte en l'identification de plusieurs solutions, notées $N_{geom}(r_o, r_n)$ et $N_{geom}(r_n, r_o)$, parmi lesquelles une seule est retenue par un tirage aléatoire selon une loi α_{select} (biaisée en fonction de l'énergie de torsion par exemple). De plus cette résolution se fait sous contraintes (longueurs de liaisons et angles de liaisons constants) ce qui nécessite alors de tenir compte des Jacobiens (notés $J(r_n, r_o)$ et $J(r_o, r_n)$) de transformation des coordonnées contraintes en coordonnées cartésiennes. La probabilité d'acceptation $acc((r_o, V), (r_n, V))$ est alors :

$$min\left[1, \frac{\frac{1}{N_{geom}(r_n, r_o)}\alpha_{select}(r_o)J(r_o, r_n)}{\frac{1}{N_{geom}(r_n, r_o)}\alpha_{select}(r_n)J(r_n, r_o)} \; exp\left(-\beta\left[U(r_n, V) - U(r_o, V)\right]\right)\right] \tag{2.25}$$

Grâce à ce mouvement les auteurs montrent qu'ils ont ainsi pu simuler efficacement des systèmes de polymères jusqu'à une longueur moyenne de 5000 monomères, et ceci malgré un taux de succès du mouvement très faible ($\simeq 0.1\%$).

La reptation inter-chaînes, figure 2.6 droite

Cette variante de la reptation (dénommée ultérieurement reptation IC) est adaptée aux ensembles $nN\mu^*PT$ et $nN\mu^*VT$ car elle a pour effet de changer la distribution de masse du système : un monomère est enlevé de l'extrémité d'une chaîne et est ajouté à une extrémité d'une autre chaîne. Son utilisation est rapportée dans [55] conjointement au mouvement de Pontage IC, car la reptation IC ayant plus de chance de réussite que ce dernier, elle permet de mieux simuler la distribution de masse.

2.3.4 L'efficacité statistique.

Le problème essentiel qui se pose en simulation moléculaire est l'erreur statistique due à l'insuffisance d'échantillons décorrélés. En effet la valeur $< A >$ (équation 2.2) est estimée par la quantité $< A_{run} >$:

$$< A_{run} >= \frac{1}{n_e} * \sum_{i=0}^{n_e - 1} A_i \tag{2.26}$$

où n_e est le nombre de mesures prises de la fonctionnelle A au cours de la simulation, notées $(A_i)_{i\in\{0,\ldots,(n-1)\}}$. Dans le cas idéal où les (A_i) sont décorrélés on a :

$$\sigma^2(< A_{run} >) = \frac{\sigma^2(A)}{n_e} \tag{2.27}$$

Evidemment si les (A_i) sont corrélés, la variance en sera plus élevée, et l'estimation plus mauvaise. Le problème est donc de prendre un "pas d'échantillonnage" suffisamment grand.

En terme de chaînes de Markov cela revient à attendre un temps appelé *temps de mélange* $\tau(\delta)$ de la chaîne qui indique le temps de simulation nécessaire pour que la distribution estimée soit suffisamment proche de la distribution limite π. Cette formalisation, pour les espaces discrets, est faite en [36] :

$$\tau(\delta) = max\{x \in \Omega : \tau_x(\delta)\} \quad avec \quad \tau_x(\delta) = min\{t : \ \delta_x(t') \leq \delta, \ \forall \ t' \geq t\} \tag{2.28}$$

où :

$$\delta_x(t) = \frac{1}{2}\sum_{y\in\Omega} |P^t(x,y) - \pi(y)| \tag{2.29}$$

En fait $\delta(t)$ est une distance entre les distributions $P^t(x,.)$ (distribution de la loi conditionnelle $Pr(X_t|X_0 = x)$) et π. Parfois il est possible de déterminer analytiquement un borne supérieure pour $\tau(\delta)$ en fonction de la taille du problème pour un δ fixé, mais cela reste généralement ardu et réservé aux cas où l'on a suffisamment d'information sur la structure de la chaîne. A notre connaissance, une telle mesure n'a pas encore été établie pour la simulation moléculaire avec des modèles non triviaux.

Pratiquement, on calcule alors l'autocorrélation de certaines propriétés caractéristiques du système, pour avoir une idée de la fréquence maximale d'échantillonnage permettant d'obtenir des configurations échantillonnées décorrélées. Si cette propriété est supposée gaussienne il est même possible d'estimer explicitement a posteriori le pas d'échantillonnage. Par exemple pour mesurer l'efficacité du mouvement de Pontage IC dans [55] et [50], les auteurs calculent l'autocorrélation des orientations des vecteurs des chaînes. Nous aurons l'occasion de revenir plus longuement au chapitre 3 sur les différentes possibilités qui s'offrent dans ce genre de simulations.

Finalement, quelle que soit la manière dont on mesure ce temps de mélange, il est évident qu'il représente une mesure d'efficacité de l'algorithme d'échantillonnage : plus il est court, plus l'algorithme est efficace. Et pour le cas des polymères, en dehors de la température (plus elle est froide plus le système évolue lentement), le paramètre limitant est la masse moléculaire, soit la longueur des chaînes. En effet, il est facile de comprendre que plus une chaîne est longue plus il faut, par exemple, de mouvement de reptations réussis pour changer sensiblement son orientation ou déplacer son centre de masse d'une distance proportionnelle à son rayon de giration. De même la translation d'une chaîne plus longue est l'occasion de plus de recouvrements, réduisant ainsi les probabilités de réussite de ce mouvement. Enfin si l'on veut garder un nombre suffisant de chaînes dans la boîte de simulation, l'augmentation de leurs longueurs augmente le nombre de monomères composant le système simulé, augmentant d'autant la charge de calcul globale.

2.4 La thermalisation parallèle.

2.4.1 Le principe général.

Face au problème de l'efficacité d'une simulation de Monte Carlo, des méthodes comme le *simulated tempering* et le *parallel tempering* ([24], [25]) (que nous traduisons par thermalisation parallèle) ont été proposées. Le principe est de définir une famille de distributions $(\pi_i)_{i\in 1...n_t}$ sur l'espace que l'on souhaite échantillonner, où π_1 est la distribution souhaitée et $(\pi_i)_{i\in 2...n_t}$ sont en quelque sorte des distribution secondaires servant à améliorer l'efficacité de l'échantillonnage. Lorsqu'une simulation est inefficace, c'est en général parce que les états les plus probables selon π_1 sont rares et disséminés dans l'espace des états. C'est pour cela que l'on utilise alors d'autres distributions $(\pi_i)_{i\in 2...n_t}$ qui sont choisies de telle sorte que les états probables soient de plus en plus nombreux, rendant ainsi la chaîne de Markov plus *mélangeante*. La marche aléatoire effectuée lors d'un Monte Carlo par chaîne de Markov sera faite alors classiquement selon une des distributions, et en effectuant des changements de distribution. Evidemment seules les trajectoires effectuées avec la distribution π_1 seront prises en compte pour l'échantillonnage, les trajectoires effectuées avec les autres distributions ne servant qu'à permettre de visiter d'autres régions de l'espace des états. On fait ainsi un sacrifice en terme de nombre d'états retenus, mais si les distributions $(\pi_i)_{i\in 2...n_t}$ sont correctement choisies, elle permettront de visiter des états moins corrélés qui augmenteront alors l'efficacité globale de l'échantillonnage.

2.4.2 Son utilisation en simulation moléculaire.

La méthode de la thermalisation parallèle a donc été adaptée pour le cadre de la simulation moléculaire ([18], [66]) en considérant une famille de distributions de Boltzmann sur l'espace des phases du système considéré, en définissant une famille de températures croissantes $T_1 < T_2 ... < T_{n_t}$. On considère en fait un ensemble étendu qui est le produit cartésien des ensembles correspondant à chacune des températures, et dont la fonction de partition est :

$$Q = \prod_{i=1}^{n_t} Q_i \tag{2.30}$$

où Q_i est la fonction de partition correspondant à la température T_i :

$$Q_i = \sum_{x_i} exp\left(-\frac{1}{k_B T_i} U(x_i)\right) \tag{2.31}$$

Pratiquement un système est composé de n_t sous-systèmes, simulés chacun à une température différente, avec une phase de changement de température comme suit :

- Avec une probabilité p, choisir un des systèmes au hasard et effectuer un mouvement de Monte Carlo classique. Lors du choix d'un des systèmes, il est possible d'associer à chacun une probabilité différente λ_i (avec évidemment $\sum_{i=1}^{n_t} \lambda_i = 1$).
- Avec une probabilité $(1-p)$, choisir deux systèmes s_1 et s_2 dont les températures se suivent, pour effectuer un mouvement d'*échange* de températures. Ce changement doit évidemment être sujet à acceptation ou rejet pour obéir au principe de Metropolis. Si on note T_i et $T_{(i+1)}$ les deux températures concernées et (s_1, T_i) le fait que le système s_1 est à la température T_i, le rapport des probabilités d'acceptation est :

$$\frac{acc(x \to y)}{acc(y \to x)} = \frac{acc((s_1,T_i),(s_2,T_{(i+1)}) \to (s_1,T_{(i+1)}),(s_2,T_i))}{acc((s_2,T_i),(s_1,T_{(i+1)}) \to (s_2,T_{(i+1)}),(s_1,T_i))}$$

$$
\begin{aligned}
&= exp\left[\left(\frac{1}{k_B T_i} - \frac{1}{k_B T_{(i+1)}}\right) \times (U(s_1) - U(s_2))\right] \\
&= exp\left(-\Delta\beta\Delta U\right)
\end{aligned}
$$

Les principaux facteurs identifiés pour l'efficacité de cette méthode sont :

- **Le choix des températures**. La température T_1 étant donnée, il reste possible de choisir à la fois le nombre n_t de températures et leurs valeurs. Trois contraintes doivent guider ce choix. D'abord T_{n_t}, la température la plus haute, doit être assez élevée pour permettre au système de franchir les barrières de potentiel. Ce fait restant souvent difficile à appréhender, on choisit alors une température à laquelle on observe une décorrélation très rapide entre les états visités (ou un taux de succès élevé pour des mouvements qui ont un taux de succès faible à basse température). Ensuite les températures intermédiaires doivent être suffisamment proches pour permettre un taux d'échange non nul, ce qui a lieu lorsque les distributions d'énergies observées aux différentes températures se chevauchent (voir la figue 2.7). De plus n_t ne doit pas être trop élevé car le nombre d'échantillons pris à T_1 deviendrait trop faible pour le temps de calcul investi. Enfin du point de vue pratique, cela peut aussi poser des problèmes de taille mémoire.
- **Le choix des probabilités λ_i.** Là encore ce choix reste neutre en regard de la validité de la simulation. Par contre on comprend qu'il puisse affecter son efficacité, dans une mesure qui reste à déterminer. La règle suivie est en général d'avoir une suite décroissante de manière à ce que le plus d'efforts soient apportés au sous-système le plus froid. Ce qui se justifie d'une part par le fait que c'est à cette température que l'on souhaite obtenir des échantillons, et d'autre part par le fait que plus un sous-système est chaud, plus il se mélange vite, moins il est nécessaire de lui consacrer de temps de calcul.

Finalement, on voit que si cette technique peut apporter une amélioration sensible de la qualité de l'échantillonnage, il reste certains paramètres de sa mise en oeuvre qui sont cruciaux pour son efficacité et qui se règlent encore souvent de manière empirique. L'optimisation explicite de ceux-ci par évolution artificielle fait l'objet du chapitre 7.

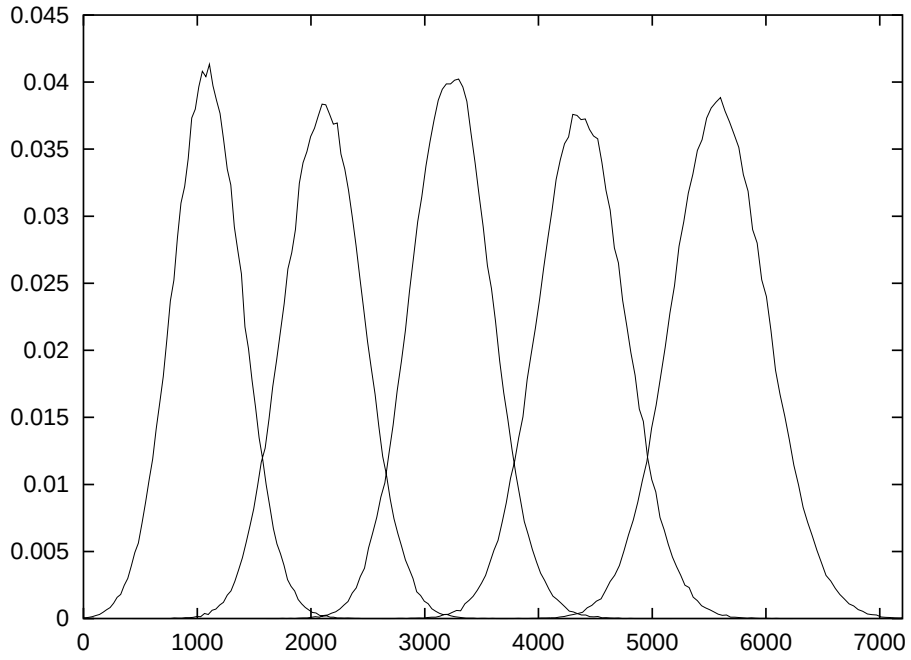


FIG. 2.7 – Histogrammes d'énergie (énergie totale des molécules simulées exprimée en multiple de ϵ_{LJ}) d'une simulation de thermalisation parallèle effectuée à $nNPT$ constants (mêmes paramètres que la série B). Les températures sont $T_0 = 450\ K$, $T_1 = 495\ K$, $T_2 = 544\ K$, $T_3 = 599\ K$, $T_4 = 659\ K$. L'ordre des histogrammes en fonction de l'énergie correspond à l'ordre des températures : à T_0 correspond l'histogramme de gauche, à T_4 l'histogramme de droite.

2.5 Les algorithmes d'évolution artificielle.

2.5.1 Le principe

L'évolution artificielle (EA) regroupe sous un même terme une classe d'algorithmes d'optimisation stochastique inspirés du principe biologique de l'évolution darwinienne et lui doivent donc non seulement leurs noms mais aussi leurs terminologies. Partant de travaux initiaux sur la simulation de modèles de l'évolution naturelle, étendus à l'optimisation en général, ces algorithmes ont émergé à peu près simultanément sous le terme *Algorithmes Génétiques* ([35], [27]) et *Stratégies d'Evolution* en Allemagne ([56]). Pour une synthèse introductive du domaine on pourra se référer à [61].

En se plaçant dans un contexte d'optimisation, le principe commun est de considérer simultanément un ensemble de points de l'espace paramétrique S sur lequel est définie une fonction F (à valeurs réelles positives) dont on cherche un optimum. Cet ensemble est alors appelé *population* Π, constituée d'*individus* (points de S). Un individu possède une valeur de *fitness* (appelée aussi *utilité*[5]) qui est la valeur de F pour le point qu'il représente. On peut schématiquement décrire le squelette d'un algorithme évolutionnaire comme suit :

1 Génération d'une *population initiale*.
2 Calcul des valeurs de fitness des individus.
3 *Sélection* des individus les plus adaptés (ceux ayant les plus hautes valeurs de fitness dans le cas d'une maximisation, les plus basses pour une minimisation), pour constituer un groupe de *parents*.
4 Application d'*opérateurs génétiques* à l'ensemble des parents pour générer un ensemble d'*enfants*.
5 Incorporation des enfants dans la population (remplacement total ou partiel) pour créer une nouvelle génération.
6 Arrêter si un critère d'arrêt est atteint, sinon retour en 2.

Il est important de remarquer qu'il n'est pas fait usage de la dérivée ou de la continuité de la fonction de fitness : seul compte le fait de pouvoir la calculer en tout point. A ce titre, on peut qualifier ce type d'algorithme, d' "algorithme d'optimisation de degré 0".

2.5.2 Le codage ou représentation.

On entend par codage la manière de représenter les paramètres du problème. C'est d'ailleurs la différence principale qu'il y a entre les algorithmes d'EA existant :

- Les Algorithmes Génétiques (AG) utilisent souvent un codage binaire : chaque paramètre est codé par une chaîne binaire plus ou moins longue.
- Les Stratégies d'Evolution (SE) utilisent un codage dit réel : chaque paramètre réel est codé par un "réel machine" (flottant). En général la correspondance est plus directe car on cherche souvent à résoudre de cette manière des problèmes dans $\mathbb{R}^n$.
- La Programmation Génétique (PG, voir [39]) utilise un codage par arborescence et est donc plus adaptée aux problèmes dont les paramètres s'expriment mieux sous cette forme, en particulier lorsque la représentation est de taille variable. Un exemple classique est la régression symbolique, où l'on recherche une expression formelle de fonction (représentation

[5] Malgré l'existence du terme français, l'usage le plus répandu dans les articles francophones est d'utiliser le terme fitness, ce que nous ferons ici également.

arithmétique arborescente) qui puisse générer au mieux un ensemble de données. On peut également représenter des petits "programmes" (d'où le nom) dans certains langages (LISP par exemple) et les faire évoluer pour accomplir au mieux une tâche donnée.

Ajoutons encore que l'on fait la distinction entre le codage d'un individu et l'ensemble des paramètres qu'il représente, car la correspondance n'est pas toujours bijective. On parle alors de *génotype* pour le codage et de *phénotype* pour son expression. Par exemple les expressions $(f(x) = x + 2)$ et $(f(x) = 2 + x)$, que l'on peut considérer comme des génotypes, représentent la même fonction, que l'on peut considérer comme leur phénotype. Comme nous allons le voir par la suite, les opérateurs génétiques sont directement liés au codage. Enfin, s'il est vrai que ces trois codages sont les plus répandus, il est en existe bien d'autres (listes ordonnées, codages mixtes, ...), pour lesquels l'essentiel est de définir des opérateurs adaptés.

2.5.3 La sélection

La sélection a pour but de choisir au sein de la population courante les individus pouvant participer à la création de la nouvelle population. Des individus (des *parents*) sont sélectionnés en rapport avec leur adaptation à l'environnement (leur *fitness*). La figure 2.8 illustre une technique classique de sélection appelé *Roulette Wheel Selection*, où l'on considère que l'on fait un tirage aléatoire sur un disque, où chaque individu occupe un secteur d'aire proportionnelle à sa valeur de fitness ; c'est un genre de loterie biaisée : les meilleurs individus ont plus de chances d'être sélectionnés. On effectue alors autant de tirages que l'on souhaite sélectionner d'individus.

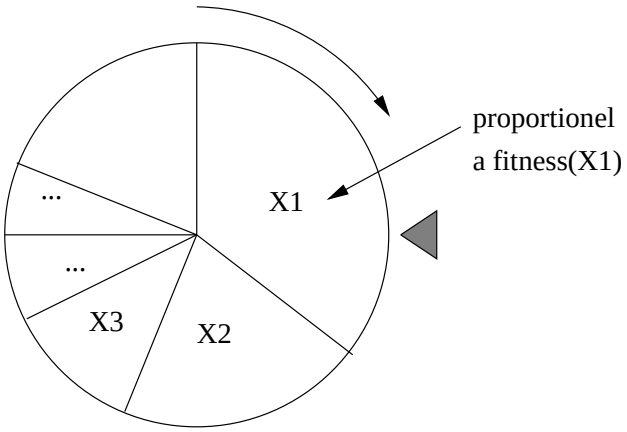


FIG. 2.8 – Sélection des parents : une loterie biaisée

De cette manière la probabilité qu'un individu soit sélectionné est :

$$p_s(x) = \frac{f(x)}{\sum_{y \in \Pi} f(y)} \tag{2.32}$$

Une variante de la roue de la fortune a été proposée par J.E. Baker dans [4]. Ici, tous les individus sont sélectionnés en une seule fois (un seul tirage aléatoire). Encore une fois la probabilité $p_s(x)$ est respectée. En reprenant l'image de la "roue de la fortune", chaque individu se voit attribuer un espace proportionnel à $p(i)$ sur la roue. Si nous nommons N' le nombre d'individus devant être sélectionnés pour la création d'une nouvelle population (qui peut être inférieur à la

taille de la population N, il y a un seul tirage de roue avec N' pointeurs également répartis (voir la figure 2.9). Pour un individu donné x, le nombre de descendants attendus est alors compris dans l'intervalle :

$$[\ \lfloor p(x) * N' \rfloor, \lceil p(x) * N' \rceil\ ] \tag{2.33}$$

Comme Baker l'a montré, cette méthode présente l'avantage d'avoir, pour le nombre de descendants d'un individu, une variance inférieure à celle des autres méthodes de sélection étudiées jusqu'alors.

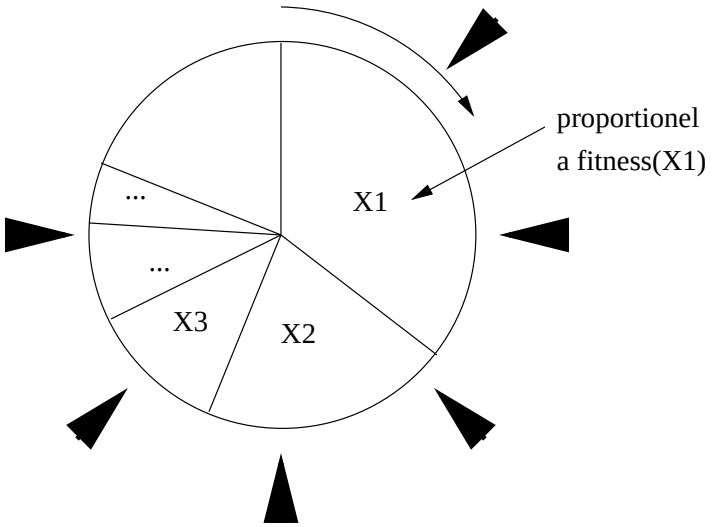


FIG. 2.9 – Roue de la fortune avec N' pointeurs

2.5.4 Le remplacement.

Une fois que l'ensemble des enfants a été créé, ils remplacent généralement tous les parents. Il est toutefois possible de faire de *l'élitisme*, c'est-à-dire de copier directement les meilleurs individus de la population précédente dans la population courante. Le remplacement de la population est alors plus progressif et permet de garder la meilleure solution découverte jusqu'alors au sein de la population. En contrepartie il peut y avoir un risque de stagnation de la population dans le bassin d'attraction d'un optimum secondaire (convergence prématurée).

2.5.5 Les opérateurs génétiques.

Les opérateurs génétiques ont pour but de produire de nouveaux individus à partir de ceux sélectionnés. On peut les diviser en deux catégories, le *croisement*, qui "mélange" les génotype de deux individus (ou plus) et la *mutation* qui est un changement aléatoire du génotype d'un seul individu. Bien sûr ils peuvent être définis de manière non unique pour chaque type de codage.

Le croisement le plus répandu pour le codage binaire est le **croisement à un point** (figure 2.10) qui fonctionne en coupant les deux chromosomes parents en un point aléatoirement choisi et échange alors les fractions résultantes pour obtenir deux nouveaux chromosomes. Le même principe peut s'appliquer pour des codages réels où les chromosomes sont des vecteurs de

$\mathbb{R}^n$. Toujours dans le cas réel, le **croisement barycentrique** permet de produire deux nouvelles valeurs (x', y') à partir des anciennes valeurs (x, y) de la manière suivante :

$$\begin{cases} x' = \gamma x + (1-\gamma)y \\ y' = (1-\gamma)x + \gamma y \end{cases} \tag{2.34}$$

Le paramètre γ est alors tiré aléatoirement uniformément sur l'intervalle $[0, 1]$, ou si l'on souhaite pouvoir produire des valeurs sortant du segment $[x, y]$ on peut aussi faire un tirage aléatoire sur l'intervalle $[-\epsilon, 1+\epsilon]$ avec $\epsilon > 0$. On peut aussi définir γ comme un vecteur aléatoire de $\mathbb{R}^n$, dont chaque composante est tirée aléatoirement uniformément sur l'intervalle $[0, 1]$, et l'utiliser pour appliquer le principe du barycentre composante par composante.

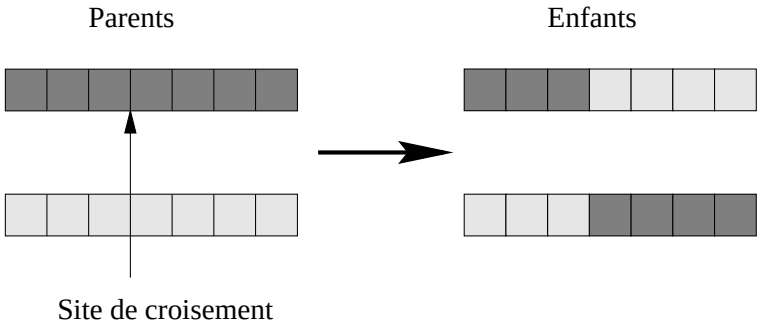


FIG. 2.10 – Croisement à un point

La mutation d'un chromosome binaire peut se faire en inversant chaque bit avec une probabilité p_m petite, ou alors on peut faire le choix d'inverser un seul bit de la chaîne avec une probabilité plus grande. Le même principe peut s'appliquer à un vecteur réel où l'on ajoute une perturbation gaussienne :

$$x' = x + N(0, \sigma_{mut}) \tag{2.35}$$

où la variance σ_{mut} est soit fixée en fonction de l'intervalle considéré soit variable et codée elle-même dans le chromosome pour laisser l'évolution déterminer quel taux de mutation est plus adapté au problème [3] et à la zone considérée. Là encore, on peut aussi utiliser un vecteur de variances applicables à chacune des composantes si l'on se place dans $\mathbb{R}^n$.

2.6 Panorama des sujets de recherche en EA.

Ayant donné un aperçu des principes essentiels de l'EA, nous voulons présenter ici certains des aspects du domaines qui en constituent les principaux axes de recherche. Le découpage peut en paraître un peu artificiel, mais il faut garder à l'esprit que ces sujets sont souvent mêlés dans nombre de travaux.

2.6.1 Schémas et Déceptivité

Les premiers travaux concernant la dynamique des AG (codage binaire) ont porté sur la notion de schémas. Un schéma représente un sous-ensemble de l'espace des chaînes binaires. On les représente classiquement en ajoutant à l'ensemble $\{0, 1\}$ le symbole '*', signifiant indifféremment

0 ou 1. Par exemple, pour une longueur de chaîne $l = 4$, les deux chaînes $i_1 = 0101$ et $i_2 = 1101$ appartiennent au même schéma $H = *101$. On définit l'ordre $O(H)$ d'un schéma H par le nombre de bits fixés dans ce schémas ($O(*101) = 3$), et la distance caractéristique $\delta(H)$ par la distance entre le premier et le dernier bit fixé du schéma ($\delta(*101) = 3 - 1 = 2$).

Ajoutons que l'évolution de la proportion des individus appartenant à tel ou tel schéma au sein d'une même population lors du déroulement d'un AG (en appliquant la sélection par roue de la fortune, la mutation avec un probabilité p_m et le croisement à un point avec une probabilité p_c) a été formalisée par Holland [35] avec le célèbre théorème suivant :

Théorème 1 (Théorème des schémas) *En utilisant les notations suivantes :*

- *m(H,t) : fréquence relative d'un schéma H au sein la génération t.*
- *f(H) : la valeur fitness moyenne des éléments de H.*
- *$\bar{f}$ la valeur de fitness moyenne de la population courante.*

Pour un schéma H on a :

$$E\left(m(H,t+1)\right) \geq m(H,t) * \frac{f(H)}{\bar{f}} * \left[1 - p_c \frac{\delta(H)}{l-1} - O(H)p_m\right]$$

La conclusion la plus souvent tirée du théorème des schémas est que les petits schémas ($\delta(H)$ et $O(H)$ petits) avec une valeur de *fitness* supérieure à la moyenne on tendance à voir la proportion de leurs représentants augmenter au sein de la population : de tels schémas sont classiquement appelés *blocs de construction* ("building blocks"). Cette remarque permet alors de dire que si un optimum de la fonction de *fitness* correspond à l'intersection de tel blocs, l'AG le trouvera facilement et que si au contraire ils mènent vers un optimum secondaire, l'AG convergera avec difficultés vers un optimum global.

Goldberg a formalisé cette idée en introduisant la notion de *déceptivité* ("deception") ([28], [29]) : la sélection entraîne une augmentation de la valeur de *fitness* moyenne entre la population courante et l'*ensemble parental* qu'elle permet de construire. Toutefois l'application des opérateurs génétiques sur cet ensemble va en changer la valeur de fitness moyenne. On peut alors considérer que la population aura tendance à être attirée par les maxima d'une fonction f', définie pour l'individu i comme l'espérance de la valeur de *fitness* de i après qu'il ait subi les opérateurs génétiques. L'AG sera alors qualifié de déceptif si les maxima globaux de f' ne correspondent pas aux maxima globaux de f.

L'analyse des schémas-déceptivité, a également servi d'outil pour analyser la difficulté d'optimisation d'une fonction en tenant compte de sa non-linéarité, que ce soit avec le concept d'épistasie ([13] [14]), ou le concept de régularité de la fonction de fitness ([46], [47], [41], [42]). Toutefois les limites d'une analyse de convergence d'un AG uniquement à partir de propriétés statiques ont fait l'objet de critiques [32], car elle ne prennent pas en compte l'aspect dynamique de l'évolution d'un population.

2.6.2 Modélisation markovienne et par processus dynamique.

La modélisation markovienne des algorithmes d'EA a servi à étudier leur propriétés de convergence ([19], [31], [54], [59]). La démarche générale est de considérer une chaîne de Markov dont l'espace d'états est l'espace des populations possibles, et non l'espace de recherche lui-même. Comme la sélection et les opérateurs génétiques ne s'appliquent qu'à la population courante, seul l'état courant conditionne les probabilités des états suivants : le processus est bien markovien. Mais en conséquence de cette démarche la chaîne de Markov n'aura pas nécessairement

toujours les mêmes propriétés en fonction des paramètres de l'algorithme, il peut y avoir des états absorbants, et ce sont justement ces propriétés qui font l'objet de ces études. Sophie Rochet a d'ailleurs établi dans sa thèse [57] une formalisation de la démonstration du théorème des schémas dans un cadre probabiliste rigoureux, en travaillant sur une modélisation markovienne de l'espace des schémas (en ayant d'ailleurs élargi ce concept dans un formalisme algébrique). Citons également les travaux Raphael Cerf [7] adaptant à l'EA les travaux de Freidlin et Wentzell [22] sur les perturbations stochastiques de processus dynamiques : l'algorithme génétique canonique avec la seule sélection est considéré comme le système dynamique, les perturbations aléatoires étant dues à la mutation (le croisement n'est pas inclus dans le modèle). Sous conditions de décroissance de la probabilité de mutation (de manière analogue à un recuit simulé), il a pu alors établir la preuve de la convergence de ce modèle d'algorithme. L'extension de ces résultats a de plus permis d'obtenir, toujours sous ces mêmes conditions, des estimations des temps de convergence ([8], [21], [40]).

2.6.3 Diversité génétique.

Le problème

De manière complémentaire à la notion de déceptivité où l'on s'attache essentiellement à analyser le lien entre la régularité de la fonction de fitness et le codage des individus, la notion de *diversité génétique* est également un facteur important pour la compréhension de la dynamique d'un AE. Cette notion peut d'ailleurs se quantifier plus explicitement à l'aide d'une distance définie sur l'espace de recherche : par exemple la distance de Hamming entre deux chaînes binaires (décompte des positions où les bits diffèrent), distance euclidienne pour des points appartenant à $\mathbb{R}^n$, ou tout autre distance appropriée au codage employé. Comme nous l'avons vu en section 2.5.3, la sélection est l'opérateur d'optimisation proprement dit car il multiplie les instances des individus jugés les meilleurs au fur et à mesure de l'évolution. Or il peut arriver, comme nous l'avons évoqué en section 2.5.4, que si cette tendance est trop marquée, un individu possédant une très bonne valeur de fitness parmi la population initiale aura très vite des descendants occupant la majorité de la population, et cette tendance sera renforcée si une stratégie d'élitisme est également employée. Or s'il s'avère que cet individu ne représente qu'un optimum secondaire, l'ensemble de la population est alors attiré vers lui provoquant une convergence prématurée, et entraînant une plus grande difficulté à générer de nouveaux individus "différents" de leurs parents.

La contrepartie évidente à cet effet d'uniformisation est l'utilisation des opérateurs génétiques dont le but est explicitement de créer de nouveaux individus. Mais généralement leur action sera d'autant plus efficace qu'une certaine diversité est déjà présente dans la population : l'opérateur de mutation n'opérant généralement que des changements mineur ne peut à lui seul produire des individus très différents (au sens de la distance appropriée) de leurs parents. De même les opérateurs de croisement ne peuvent produire de nouveaux individus différents si les deux parents sont identiques. En fait l'équilibre entre l'action d'optimisation locale de la sélection et l'action de dispersion des opérateurs génétiques est un facteur essentiel de la réussite d'un algorithme d'EA. Face au risque de convergence prématurée, il existe plusieurs méthodes que nous allons voir. La plus évidente est d'assigner à la mutation une probabilité élevée et un effet plus important, mais à l'extrême cette solution finira par rendre l'EA identique à une recherche purement aléatoire.

La pression sélective.

Un moyen de mesurer la dominance des meilleurs individus est la *pression sélective* définie comme le rapport de probabilité de sélection du meilleur individu sur la moyenne des probabilités de sélection de tous les individu. Pour une probabilité de sélection exactement proportionnelle à la valeur de fitness (cas de la *roue de la fortune*) cette valeur correspond exactement au rapport de la valeur de fitness du meilleur individu sur la valeur de fitness moyenne de la population. On peut alors limiter une domination excessive en contrôlant explicitement cette pression sélective. On peut par exemple effectuer un *échelonnage* (traduction de "scaling") des valeurs de fitness. Une méthode pour le faire (en désignant les valeurs de fitness maximales et moyennes de la population courante Π par F_{max} et F_{moy}) est de prescrire une pression sélective C et d'effectuer la transformation linéaire suivante :

$$f'(i) = a * fitness(i) + b$$

$$\text{où } a = \frac{(C-1)*F_{moy}}{F_{max}-F_{moy}} \quad \text{et } b = \frac{Fmoy*(F_{max}-C*F_{moy})}{F_{max}-F_{moy}}$$

On a bien alors :

$$C = \frac{max\{f'(i); i \in \Pi\}}{\sum_{i\in\Pi} f'(i)}$$

On peut encore utiliser des méthodes de sélection se basant uniquement sur le classement des individus au sein de la population en fonction des valeurs de fitness. La méthode dite justement du classement ("ranking") consiste à appliquer une distribution de probabilité de sélection prédéfinie à la population classée. La méthode du tournoi consiste à choisir aléatoirement plusieurs individus dans la population (généralement 2) et à en retenir le meilleur ; ainsi le choix se fait uniquement en fonction du classement.

Le partage.

Enfin les méthodes dites de *partage* ou *nichage* ("sharing" ou "niching" [30]) ont également pour but de maintenir une certaine diversité dans la population. Qui plus est, si le problème est multimodal, elles sont souvent utilisées comme le moyen d'identifier plusieurs optima simultanément au terme de l'évolution. Le but est de forcer l'AE à explorer simultanément toutes les zones prometteuses de l'espace de recherche qu'il découvre. L'espoir est alors de créer autant *d'espèces* (ou de sous-populations) qu'il y a d'optima à identifier. Le concept de partage des ressources vient de ce besoin : si les individus d'une même sous-population ont à partager des ressources, la croissance de cette population est limitée, et lorsqu'il y a surpopulation, les individus tendent à chercher de nouvelles régions à coloniser. Le moyen principal de contrôler un algorithme d'EA étant la fonction de fitness, c'est sur elle que le partage agit. L'idée est de dégrader la valeur de fitness initiale en fonction de la "proximité" des autres individus de la population. On peut se reporter à [48], qui présente un cadre formel pour la mesure de la diversité ainsi qu'un large panorama des méthodes utilisées pour sa conservation.

2.6.4 Extensions du domaine.

En plus de travaux sur le comportement dynamique des algorithmes d'EA qui se sont souvent faits au prix de restrictions voire de simplifications des méthodes utilisées, la pratique de l'EA a amené nombre de diversifications de ses méthodes. Les opérateurs génétiques (sélection, mutation

et recombinaison) en sont évidemment les principaux sujets. Toutefois nous voulons finir en attirant l'attention sur les travaux qui ont amené l'EA à étendre ses domaines d'application initiaux.

Optimisation sous contraintes.

Pour faire de l'optimisation sous contraintes, nombre d'adaptations ont été proposées (on pourra par exemple se référer à [52] pour une présentation sur ce sujet). On peut dire succinctement que dans ce cadre, l'espace de recherche se décompose en sous-espace faisable (ensemble de points satisfaisant les contraintes) et sous-espace infaisable (ne satisfaisant pas les contraintes). La solution la plus simple et la plus radicale consistant à rejeter tous les individus dits "infaisables" (ne satisfaisant pas les contraintes) donne généralement de mauvais résultats. Un technique alternatives est la *réparation* : un individu "infaisable" est transformé en un individu faisable, qui soit le plus "proche" de lui. Une autre est l'application de *pénalités* : tous les individus sont gardés sans changement mais on applique une pénalité (diminution de la valeur de fitness) aux individus infaisables. Par rapport à la précédente cette technique permet d'éviter de concentrer la population sur les "frontières" des régions faisables.

Optimisation multi-critères

L'optimisation multi-objectif en EA [20] est aussi un sujet de recherche à part entière, dans la mesure où l'évaluation étant alors vectorielle, les méthodes de sélection habituelles ne sont plus applicables directement. L'outil principal est alors l'optimalité au sens de Pareto : un individu sera dit non-dominé si il n'existe au sein de la population aucun autre individu qui le domine, c'est-à-dire aucun individu dont toutes les composantes du vecteur d'évaluation soient meilleures. On désigne l'ensemble de ces individus par *front de Pareto* (illustré par la figure 2.11), et le but est alors de ne pas en favoriser arbitrairement, mais plutôt de maintenir une certaine diversité sur ce front.

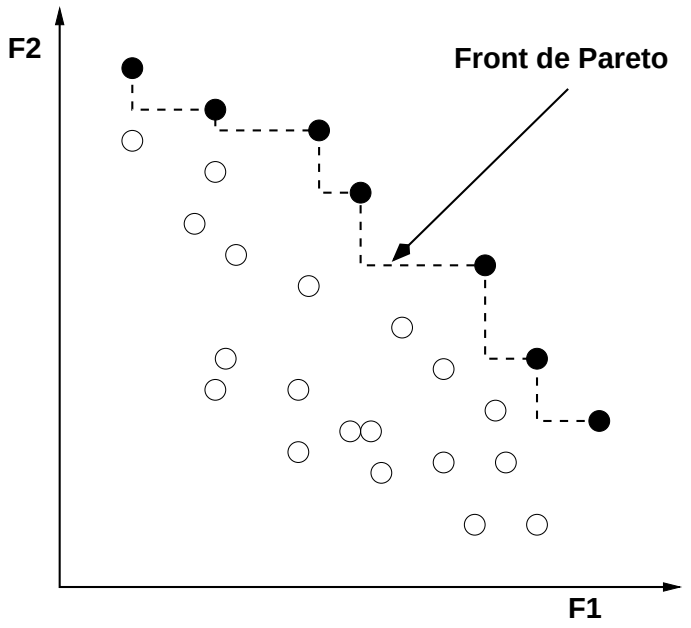


FIG. 2.11 – Front de Pareto pour deux critères. Les ronds représentent un ensemble de points pour lesquels ces deux critères sont calculés. Les rond noirs sont les individus non-dominés, si on considère une maximisation.

Environnements bruités

Dans certains problèmes d'optimisation, les évaluations des performances peuvent être sujettes à des fluctuations aléatoires. Les manières habituelles de faire face à un environnement bruité ([53], [34], [2]) sont soit d'augmenter la taille de la population, ce qui peut suffire parfois, soit d'évaluer plusieurs fois chaque individu et de prendre la moyenne de ces évaluations comme valeur de fitness (la variance est divisée par le nombre d'évaluations). Nous serons confrontés par la suite à ce problème, et proposerons alors une méthode plus adaptée à notre type d'application (voir la section 4.1.5).

2.7 Confrontation des problématiques.

Mêmes si les problématiques de l'échantillonnage de Monte Carlo et de l'optimisation stochastique par algorithme d'évolution artificielle diffèrent dans leur finalité, ces deux familles d'algorithmes sont modélisables par des processus markoviens. Dans le premier cas, cette modélisation est explicite car c'est à travers les propriétés d'une chaîne de Markov que l'on conditionne l'échantillonnage et dans le deuxième cas, comme nous l'avons évoqué en section 2.6.2, elle permet d'étudier les propriété de convergence.

De plus, ces deux familles d'algorithmes se rejoignent encore sur une qualité commune que l'on requiert d'eux : effectuer une marche aléatoire sur un espace de recherche tout en couvrant efficacement toutes les "régions intéressantes" de cet espace. Dans le cadre du MCM, la notion de région désigne un ensemble de points dans l'espace d'états accessibles avec des probabilités non négligeables et en un nombre de transitions limité. L'adjectif "intéressant" désigne alors

les régions contenant des états dont la densité de probabilité est importante (par exemple en simulation moléculaire, ils correspondent aux configurations d'énergie les plus basses). En EA, le terme région désigne généralement un ensemble de points proches dans l'espace de recherche selon une distance appropriée à cet espace. Une région intéressante sera alors une région contenant beaucoup d'individus avec des valeurs de fitness hautes (respectivement basses) dans le cas d'un problème de maximisation (respectivement minimisation). Dans le cadre du MCM, si le système reste confiné dans une région de second ordre, on peut parler de *régime métastable* et en EA si la population se concentre dans une région de second ordre, on peut parler de *convergence prématurée* vers un *optimum local* (voir la section 2.6.3).

	EA	MCM
Utilisation	Optimisation	Echantillonnage
Chaîne de Markov	Induite par la modélisation de l'algorithme Pas de propriétés imposées	Ergodique, contrainte par la distribution prescrite
Espace d'états	Espace des populations	Espace de recherche
Faiblesse	Convergence prématurés vers un optimum local	Régime métastable

Compte tenu de la bonne aptitude des algorithmes d'EA à visiter efficacement un espace de recherche multimodal (comme nous l'avons vu en section 2.6.3 il existe plusieurs moyens de maintenir la diversité de la population), il serait tentant alors de vouloir utiliser un algorithme d'EA dans le but d'effectuer un échantillonnage rigoureux de cet espace. Mais il semble qu'un certain nombre de difficultés nous en empêchent encore. Tout d'abord s'il est parfois possible d'avoir une idée de la chaîne de Markov représentative de tels algorithmes, les propriétés requises d'ergodicité et de contrôle de la distribution limite ne sont pas établies pour le cas général à ce jour. De plus, comme nous venons de le voir, l'espace d'états considérés n'est pas l'espace de recherche, mais l'espace des populations (voir la section 2.6.2) et en l'absence de résultats théoriques nouveaux nous ne pouvons pas contrôler explicitement la distribution des individus sur l'espace de recherche en fonction des paramètres de l'algorithme (types de sélection, opérateur génétiques...).

Par contre les algorithmes d'EA peuvent être utilisés comme des méta-heuristiques efficaces lors de l'utilisation d'algorithmes dont certains des paramètres restent difficiles à régler autrement qu'empiriquement [33]. A cet égard les méthodes de Monte Carlo appliquées à la simulation moléculaire présentent nombre de paramètres qu'il est possible d'optimiser en vue d'améliorer leurs performances et par conséquent l'efficacité statistique de l'échantillonnage. C'est donc dans cet axe que nous avons orienté notre travail.

Chapitre 3

Critères d'efficacité pour la simulation de Monte Carlo des polymères.

Ayant vu les principes de la simulation moléculaire de polymères par algorithme de Monte Carlo Markovien, et évoqué quelques paramètres susceptibles d'être optimisés pour améliorer l'efficacité de cette simulation, il nous faut maintenant trouver des mesures numériques de cette efficacité. Nous nous attacherons donc à passer en revue dans ce chapitre les diverses possibilités déjà proposées dans la littérature, desquelles nous tirerons des critères numériques. Puis nous introduirons une nouvelle mesure fondée sur le principe de la lacunarité de la distribution de masse.

3.1 Propriétés des chaînes

Comme nous l'avons évoqué en section 2.3.4, il n'a pas été établi à ce jour de moyen de mesurer directement le temps de mélange[1] de la chaîne de Markov sous-jacente à la marche aléatoire dans l'espace des configurations pour le type de simulations qui nous intéresse. Or c'est précisément à partir de la connaissance de cette valeur que l'on peut savoir le temps de simulation nécessaire pour effectuer un échantillonnage correct.

Néanmoins en l'absence de cette connaissance, on retrouve dans la littérature le souci de contrôler la justesse des mesures effectuées en s'appuyant sur des critères "physiques" de mélange. Le principe général est de mesurer l'autocorrélation d'une propriété caractéristique du système et de considérer le temps de décorrélation de cette propriété. Si la décroissance de cette autocorrélation indique réellement une décorrélation des configurations, alors les configurations, prises régulièrement avec un pas de temps égal au temps de décorrélation, pourront être considérées comme indépendantes. Plus particulièrement, pour la simulation des polymères, les propriétés géométriques des chaînes sont souvent prises en compte ([17], [55], [50])

3.1.1 Autocorrélation des vecteurs de chaîne.

Les vecteurs d'orientation des chaînes correspondent concrètement aux vecteurs normalisés reliant les deux extrémités de chacune des N chaînes de la boîte de simulation. On comprend que pour des chaînes longues en état dense, qui sont généralement très imbriquées, les changements d'orientations nécessitent de déplacer un grand nombre de monomères. On peut donc raisonnablement supposer que si au bout d'un certain temps de simulation les orientations de chaînes ne sont plus du tout corrélées avec leur orientations initiales, les configurations ne le sont plus.

Si pour chaque chaîne, on échantillonne régulièrement les vecteurs d'orientation $\{v_t(i)\}_{i=0...N}$ (i désignant la chaîne et t l'indice de l'échantillon), on peut calculer au bout d'un nombre n_e fixé de pas d'échantillonnage (soit (n_e+1) échantillons en comptant la configuration initiale), les 3 quantités suivantes :

- Une autocorrélation que nous appellerons *autocorrélation instantanée* qui se mesure à tout instant t par :

$$C_v^i(n_e) = \frac{1}{N}\sum_{i=0}^{N} v_{n_e}(i).v_0(i) \tag{3.1}$$

  Cette valeur ne dépend alors que de la configuration initiale et de la dernière configuration des chaînes.
- La deuxième, que nous appellerons *autocorrélation moyenne*, est en fait le véritable estimateur d'autocorrélation. Si n_e échantillons ont été pris, on estime l'autocorrélation des états séparés d'un temps de $\frac{n_e}{2}$ pas d'échantillonnage par :

$$C_v^m(n_e) = \frac{2}{n_e} \sum_{t=(\frac{n_e}{2}+1)}^{n_e} \sum_{i=1}^{N} v_t(i).v_{t-\frac{1}{n_e}}(i) \tag{3.2}$$

[1] Rappelons que le terme temps de mélange pour une chaîne de Markov ergodique, désigne le temps moyen nécessaire pour que la distribution estimée par une marche aléatoire selon cette chaîne soit assez proche (avec un erreur prédéfinie) de la distribution limite de la chaîne.

– Enfin nous définissons une *autocorrélation cumulée* comme étant la moyenne sur tous les échantillons des autocorrélations instantanées :

$$C_v^c(n_e) = \frac{1}{n_e} \sum_{t=1}^{n_e} C_v^i(t) \tag{3.3}$$

Quelle que soit la définition considérée, l'autocorrélation décroît vers 0 lorsque le nombre d'échantillons augmente. En revanche, il est clair que si l'autocorrélation instantanée décroît plus vite, elle aura aussi tendance à osciller beaucoup plus car elle fait intervenir moins de mesures d'orientations alors que les deux autres intègrent l'ensemble des mesures effectuées. Ce qui nous a motivé à considérer l'autocorrélation cumulée est qu'on peut la calculer de manière itérative en ne conservant qu'un seul échantillon $\{v_0(i)\}_{i=0...N}$ de référence (nécessitant donc moins de mémoire lors d'une implantation informatique) :

$$C_v^c(n_e + 1) = \frac{n \times C_v^c(n_e) + C_v^i(n_e + 1)}{n_e + 1} \tag{3.4}$$

La question qui se pose alors est de savoir quelle est la mesure la plus appropriée pour qualifier l'efficacité d'une simulation. Partons d'abord du principe que nous souhaitons maximiser une fonction réelle positive et considérons un critère de la forme

$$c(t) = 1 - C(t)$$

où $C(t)$ désigne l'une des trois autocorrélations. Etant donnée la nature aléatoire de l'algorithme de Metropolis, il faut considérer ces mesures comme des réalisations de variables aléatoires. Il nous paraît donc souhaitable de privilégier la mesure qui, à temps de simulation égal, présente la plus faible variance.

Dans ce but nous avons simulé un système de polyéthylène (suivant le modèle présenté en section 2.3, dans les conditions exactes de simulation précisées en section 4.2.1.) dans l'ensemble nNPT :

- $n = 640$ monomères.
- $N = 20$ chaînes de 32 monomères exactement.
- $P = 1atm$
- $T = 450K$
- Les 4 premiers mouvements définis en section 2.3.3 : rotation, reptation, flip, translation et fluctuation de volume. A cette occasion nous avons donné la même fréquence aux 3 premiers ($\frac{640}{1941}$), une fréquence égale à ($\frac{20}{1941}$) pour la translation (qui implique le déplacement de 32 monomères) et une fréquence égale à ($\frac{1}{1941}$) pour la fluctuation de volume qui implique le déplacement simultané des 640 monomères.
- 32 simulations indépendantes ont été effectuées pour une durée de 22000 secondes (sur des processeurs Pentium III, 733 Mhz).
- Les vecteurs d'orientation des chaînes (vecteurs bout-à-bout normalisés) ont été calculés toutes les 10 secondes
- Pour chaque système, les 3 critères sont mesurés sur des périodes de temps croissantes ($\tau_i = i * 200s, \quad i \in \{1, ...10\}$), la mesure étant répétée 2 fois pour chaque longueur de période (pour un total de $\tau = 2 \times \frac{10*11}{2} \times 200s = 22000s$) donnant pour chaque durée 64 valeurs (2 fois 32 simulations) pour les 3 critères.

La figure 3.1 montre alors les courbes pour les 3 critères (c_v^i noté ci, c_v^m noté cm et c_v^c noté cc) de la moyenne et du rapport moyenne sur écart-type. Comme on s'y attend le critère d'autocorrélation instantanée est plus important que les autres. Par contre on voit que si le rapport moyenne sur écart-type décroît un peu avec le nombre d'échantillons, il se stabilise environ à $\frac{1}{3}$ à partir d'un certain temps et ceci pour les 3 critères. On peut en conclure que si l'autocorrélation instantanée ne présente pas plus de variance que les deux autres mesures, c'est sans doute que cette variance est plus attribuable à la nature aléatoire des simulations qu'à l'imprécision des mesures elles-mêmes.

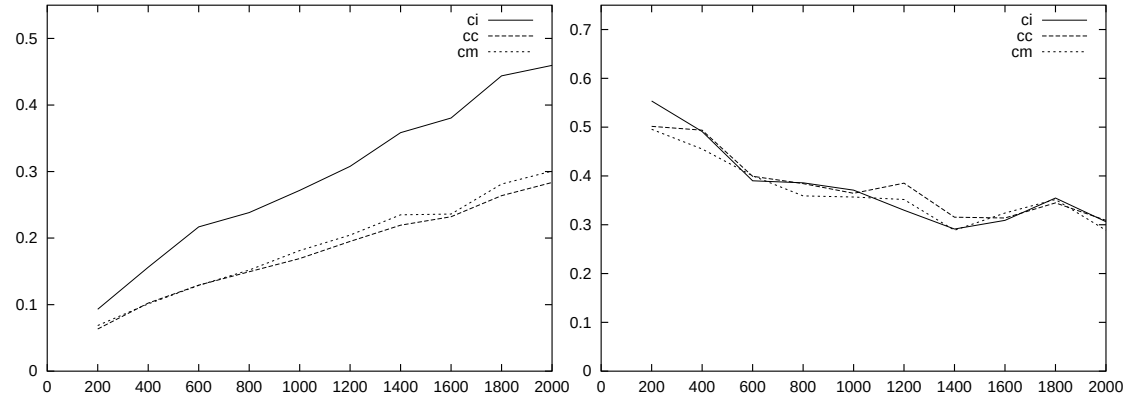


FIG. 3.1 – Gauche : moyenne des critères d'autocorrélation (c_v^i noté ci, c_v^c noté cc et c_v^m noté cm) des vecteurs d'orientation en fonction du temps de simulation. Droite : rapport moyenne sur écart-type

Finalement, si l'on considère la complexité en mémoire nécessaire pour calculer ces critères, c_v^i et c_v^c sont les plus intéressants. Le rapport moyenne sur écart-type ne permettant pas de départager entre ces deux critères, on pourra utiliser l'un ou l'autre. Toutefois, pour un confort de visualisation, on pourra préférer l'autocorrélation cumulée C_v^c pour l'aspect plus lisse de son graphe comme le montre la figure 3.2, ce qui peut aussi rendre plus facile la comparaison visuelle de deux graphes correspondant à deux simulations différentes.

3.1.2 Déplacement des molécules.

Une autre grandeur communément utilisée comme indicateur de mélange est le déplacement des molécules par rapport à leurs positions d'origine. Dans le cas des polymères, on calcule le déplacement des centres de masse des chaînes. Sachant qu'en état dense, les chaînes sont fortement imbriquées, leurs centres de masses se déplacent très lentement par rapport à leur propre taille, ce qui implique que plus ce déplacement est important plus le système se mélange.

Nous proposons donc d'utiliser également les deux critères d'efficacité suivants :

– Le *déplacement carré des centres de masse* des chaînes :

$$d^2(t) = \frac{1}{N}\sum_{i=1}^{N}(rc_t(i) - rc_0(i))^2 \tag{3.5}$$

où $rc()$ désigne les coordonnées du centres de masse d'une chaîne et t le temps de simulation.

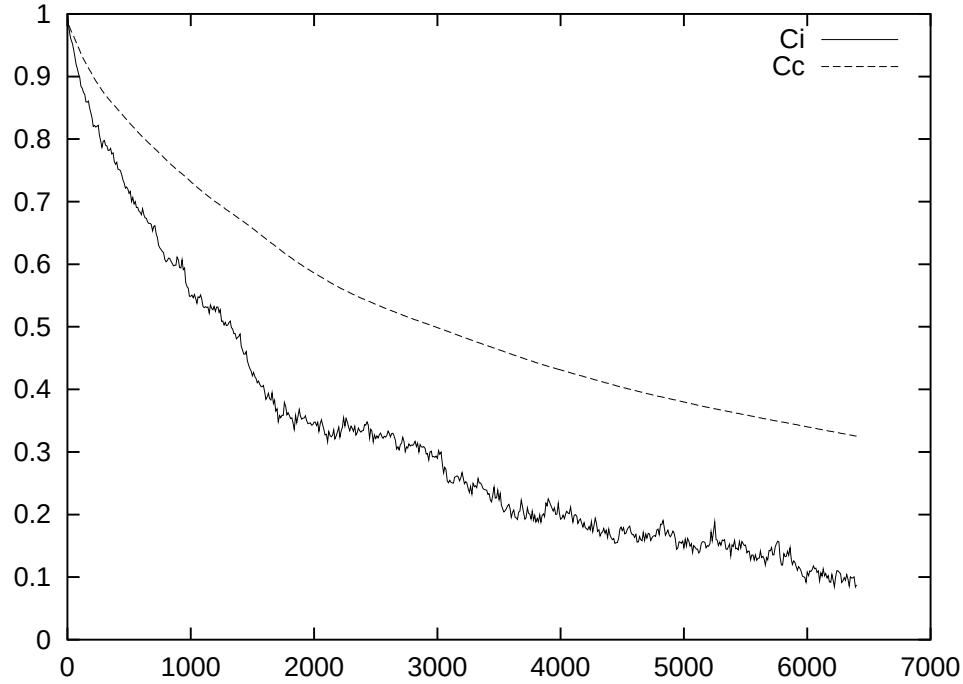


FIG. 3.2 – Comparaison des graphes d'autocorrélation instantanée et cumulée pour la même simulation (C_v^i noté Ci, C_v^c noté Cc).

– Le *déplacement carré cumulé des centres de masse* des chaînes, utilisant là encore tous les déplacements mesurés au cours de la période considérée :

$$d_c^2(n_e) = \frac{1}{n_e} \sum_{t=1}^{n_e} d^2(i) \tag{3.6}$$

Le déplacement cumulé n'a pas de signification sur le plan physique, mais encore une fois, le fait d'utiliser tous les échantillons nous a semblé être un moyen de diminuer la variance de ce critère mesuré. De manière analogue on peut aussi mesurer le déplacement carré moyen des monomères, offrant de la même manière deux critères supplémentaire :

– Le *déplacement carré des monomères* :

$$d_m^2(t) = \frac{1}{n} \sum_{i=1}^{n} (r_t(i) - r_0(i))^2 \tag{3.7}$$

où $r()$ désigne les coordonnées d'un monomère et t le temps de simulation.

– Le *déplacement carré cumulé des monomères* :

$$d_{mc}^2(n_e) = \frac{1}{n_e} \sum_{t=1}^{n_e} d_m^2(i) \tag{3.8}$$

Lors de la simulation présentée dans la section précédente, nous avons également mesuré ces critères et les figures 3.3 et 3.4 en présentent alors les courbes de moyenne et de rapport moyenne sur écart-type. Le rapport moyenne sur écart-type est ici plus favorable aux moyennes dites cumulées.

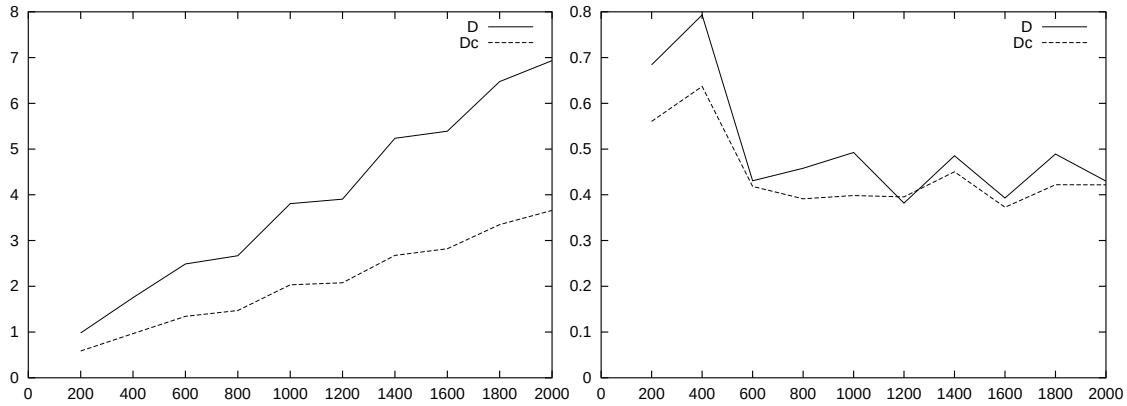


FIG. 3.3 – Gauche : moyenne des critères de déplacement carré des centres de masse des chaînes (d^2 noté D, d_c^2 noté Dc), exprimés en unités réduites (1 unité = σ_{LJ}^2). Droite : rapport moyenne sur écart-type

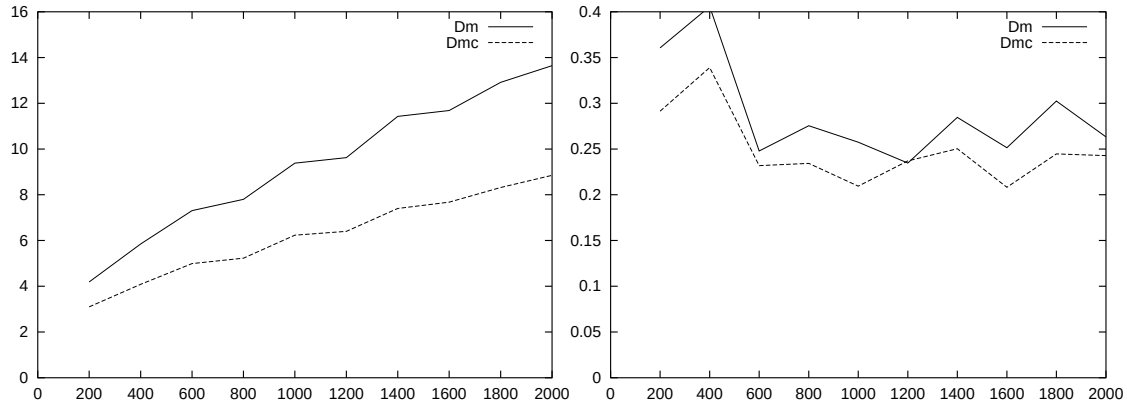


FIG. 3.4 – Gauche : moyenne des critères de déplacement carré des monomères (d_m^2 noté Dm, d_{mc}^2 noté Dmc), exprimés en unités réduites (1 unité = σ_{LJ}^2). Droite : rapport moyenne sur écart-type

Si la notion de déplacement des molécules a un sens évident lorsqu'un système est simulé par dynamique moléculaire, il n'en est pas toujours autant pour tous les mouvements de Monte Carlo. En particulier lorsque l'on utilise un mouvement qui change la connectivité des chaînes (un ou plusieurs monomères passent d'un chaîne à une autre, comme par exemple lors d'un mouvement de pontage inter-chaînes [55], décrit en section 2.3.3), il n'est plus possible de mesurer simultanément le déplacement des centres de masse et des monomères. Pour le concevoir il faut se souvenir que, du fait des conditions de périodicité de la boîte de simulation décrites en section 2.2.2, chaque molécule possède des coordonnées de référence et une infinité d'images dont les coordonnées sont obtenues par décalage. Il est évident qu'un déplacement ne peut être mesuré qu'en prenant en compte les coordonnées de références. Lors d'un mouvement de Pontage inter-chaîne, il se peut qu'une chaîne en attaque[2] une autre qui soit en fait l'image d'une chaîne qui est en réalité distante d'une distance plusieurs fois supérieure au côté de la boîte de simulation. La figure 3.5 illustre cette possibilité, où une chaîne C1 attaque l'image d'une chaîne C2 située près d'une de ses extrémités. Ce faisant, si l'on veut conserver la cohérence des centres de masse des chaînes, il faut que les monomères pris à C2 aient de nouvelles coordonnées de référence cohérentes avec celles des monomères de C1, résultant en un déplacement de ces monomères supérieur au côté de la boîte. De ce fait, la notion la mesure du déplacement des monomères n'a plus de sens. Inversement s'il l'on voulait absolument le calculer, il faudrait ne changer que les coordonnées des 3 monomères réellement déplacés obtenant une nouvelle chaîne C1, composée de "monomères de référence" et de "monomères image", rendant incohérent le suivi des centres de masse.

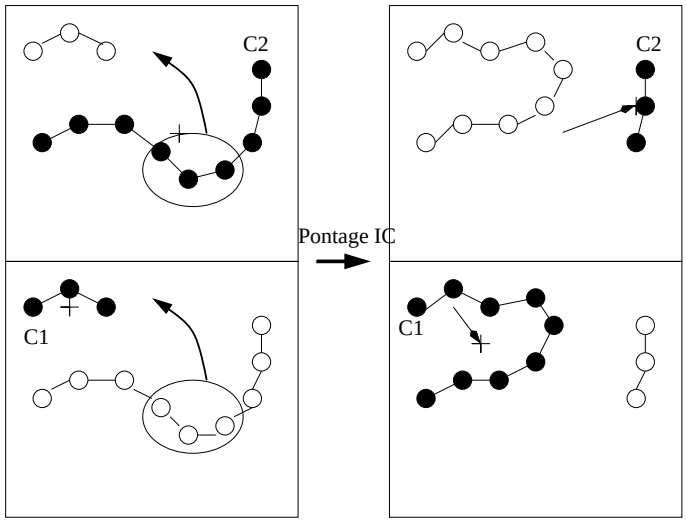


FIG. 3.5 – Rupture de connectivité et déplacement des centres de masse. Les ronds noirs indiquent les coordonnées de référence, et les rond blancs des images périodiques dont les coordonnées sont obtenues par décalage. Les croix indiquent la position des centres de masse des chaînes et les flèches (sur la partie droite) leurs déplacements.

En conclusion il n'est pas possible d'utiliser simultanément les déplacements de monomères

[2]On dit que la chaîne qui attaque est celle au bout de laquelle on ajoute une partie de l'autre, soit celle qui se trouve rallongée au détriment de l'autre.

et des centres de masse, lors de simulations dans l'ensemble $nN\mu^*PT$.

3.2 Critères de lacunarité.

Nous avons vu dans la section précédente comment utiliser la géométrie des chaînes pour mesurer le mélange du système. Cependant on peut objecter que, si ces mesures sont parfaitement adaptées à des simulations par dynamique moléculaire, elle peuvent moins l'être pour le cas de certaines techniques de Monte Carlo. Par exemple le suivi des positions des molécules dépend de la manière dont sont mises à jour leurs coordonnées après un mouvement de Monte Carlo. Pareillement l'orientation des chaînes a-t-elle autant de signification lorsqu'elles peuvent être rallongées ou raccourcies ? On constate que ces critères, en fonction du type de mouvement de Monte Carlo, peuvent paraître dans une certaine mesure arbitraires, ou du moins leur interprétation n'est-elle pas aussi évidente que dans le cas de la dynamique moléculaire.

C'est pourquoi nous avons cherché à identifier un critère qui puisse se mesurer sur un ensemble de configurations sans pour autant dépendre de l'étiquetage des molécules. Une possibilité est d'étudier les variations locales de densité à l'intérieur de la boîte de simulation. En effet, même si le volume global (et donc la densité globale) est fixé, les molécules ne sont pas disposées bien uniformément dans la boîte de simulation, surtout lorsque l'on considère des polymères. Cet écart par rapport à l'uniformité va nous servir à définir un nouveau critère de mélange.

3.2.1 Lacunarité de densité de monomères.

On trouve justement en géométrie fractale la notion de lacunarité ([49], [45]) mesurant la dispersion de la masse d'un objet par rapport à l'homogénéité. Si l'on considère un objet de masse M et un voisinage V de rayon ϵ (par exemple un cube de côté ϵ), on peut calculer la masse m attendue dans V. La lacunarité mesure l'écart de la masse observée m' par rapport à la masse attendue m :

$$L = \left\langle \left(\frac{m'}{m} - 1 \right)^2 \right\rangle \tag{3.9}$$

On peut directement adapter cette mesure à la répartition des monomères. On définit une subdivision de la boîte de simulation en R^3 cubes, où R est un paramètre d'échelle. On s'intéresse à la masse (i.e. le nombre monomères) présente dans chacun de ces cubes que l'on note, $m_R(i), i \in \{1, ...R^3\}$. La masse attendue est $m_e(R) = \frac{n}{R^3}$ si n est le nombre de monomères dans la boîte de simulation. La lacunarité est alors :

$$L(R) = \frac{1}{R^3} \sum_{i=1}^{R^3} \left(\frac{m_R(i)}{m_e(R)} - 1 \right)^2 \tag{3.10}$$

$L(R)$ fournit donc une mesure de lacunarité d'un configuration spatiale donnée à une échelle R donnée.

3.2.2 Critères d'efficacité.

Si, au cours d'une simulation, on échantillonne régulièrement la distribution de masse pour différentes valeurs d'échelles, en calculant les moyennes on obtient une estimation de la lacunarité

aux différentes échelles. De plus si on accumule les distributions de masses, on peut alors mesurer la lacunarité de manière temporelle, i.e. mesurer la vitesse d'uniformisation de cette masse cumulée (voir la figure 3.6). Une simulation efficace sera celle qui, en un temps de simulation donné, aura la valeur de lacunarité cumulée la plus basse.

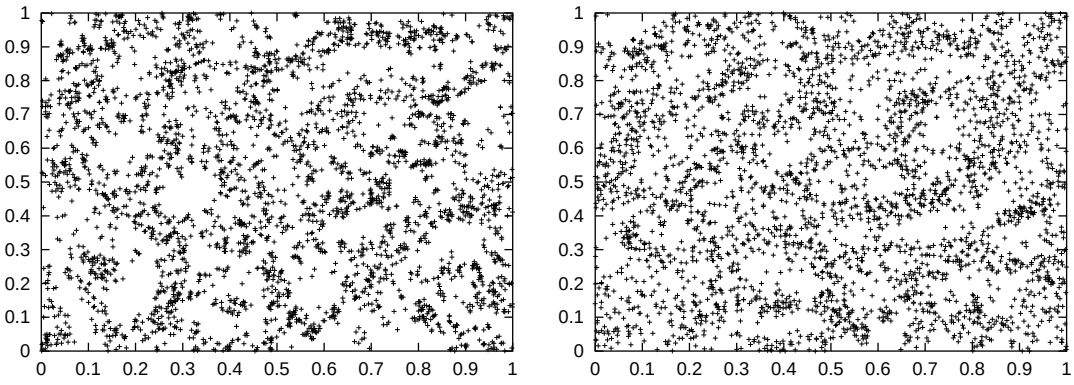


FIG. 3.6 – Cumul de 5 configurations comprenant chacune 640 monomères, projetées dans le plan XY. Ces configurations ont été prises pour une même type de sytème avec un pas de d'échantillonage plus long à droite qu'à gauche, donnant des configurations plus décorrélées à droite. L'image de droite a une distribution de points plus proche de l'uniformité que celle de gauche, elle est donc moins lacunaire.

Pour une échelle donnée R, nous proposons le critère de mélange suivant :

$$c_L^R(n_e) = max\left(0, \frac{\hat{L}(R)}{L_c(R, n_e)} - 1\right) \tag{3.11}$$

où $L_c(R, n_e)$ désigne la lacunarité de la masse accumulée au bout de n_e échantillons et $\hat{L}(R)$ désigne la moyenne des lacunarité des configurations échantillonnées (qui est généralement supérieure à $L_c(R, n_e)$) :

$$\hat{L}(R) = \frac{1}{n_e}\sum_{i=1}^{n_e} L(R, i) \tag{3.12}$$

La figure 3.7 montre la courbe de la moyenne des mesures de ce critère pour différentes échelles, sur des périodes croissantes, obtenues lors de la simulation présentée dans la section précédente. La figure 3.8 montre les courbes correspondantes du rapport moyenne sur écart-type. Il est important de savoir que la longueur moyenne du côté d'une boîte de simulation est dans les conditions présentes d'environ 6.93 fois la distance caractéristique de Lennard Jones σ_{LJ}, celle-ci indiquant la distance de répulsion entre deux monomères n'appartenant pas à la même chaîne. La distance de liaison entre 2 monomères consécutifs étant elle-même $0.39 \times \sigma_{LJ}$, on comprend que pour des cubes ayant un côté d'une longueur légèrement inférieure à σ_{LJ}, il peut y avoir soit 0 monomère, soit 1, soit plusieurs lorsqu'ils appartiennent à la même chaîne (2 ou 3 typiquement). La présence de deux monomères appartenant à deux chaînes différentes y est alors fortement improbable. Dans notre cas ceci correspond au cas où $R > 6$, et on constate justement qu'à ces échelles le critère progresse plus lentement et que la variance est moins élevée.

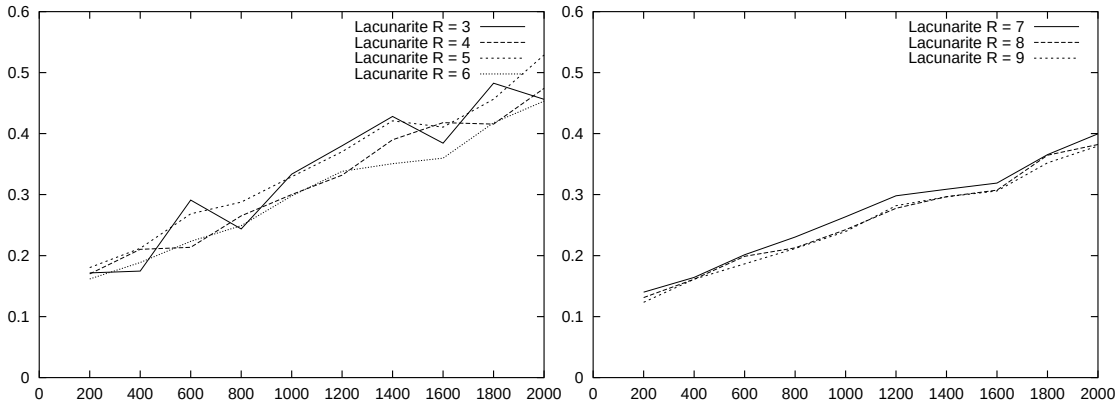


FIG. 3.7 – Moyenne des critères de lacunarité à différentes échelles.

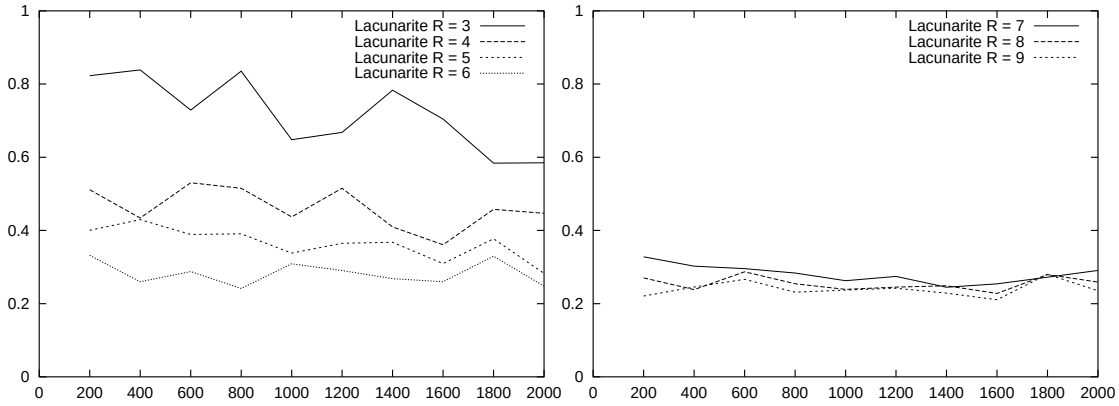


FIG. 3.8 – Rapport moyenne sur écart-type des critères de lacunarité à différentes échelles.

Pour tenter d'expliquer ce comportement, il est utile d'étudier $L(R)$ en l'approximant par :

$$\log(L(R)) = \alpha \log(R) + \beta \tag{3.13}$$

La limite de $L(R)$ en fonction de R peut se calculer en considérant que pour un nombre fini n de points répartis dans un volume fini, il existe une valeur de R pour laquelle la plus petite distance entre les points est supérieure au diamètre des cellules. Dans notre cas cette plus petite distance correspond à la distance de liaison de deux monomères (la distance de répulsion étant supérieure). Considérant qu'à partir d'un moment il n'y a au plus qu'un seul monomère par cellule, si on note $M(i)$ le nombre de monomère dans la cellule i, alors $M(i) \in \{0, 1\}$:

$$lim_{R\to+\infty} L(R) = lim_{R\to+\infty} \sum_{i=1}^{R^3} \left(\frac{R^3 M(i)}{n} - 1 \right)^2$$

$$lim_{R\to+\infty} L(R) = lim_{R\to+\infty} \frac{1}{R^3} lim_{R\to+\infty} \sum_{i=1}^{R^3-n-1} 1 + \sum_{i=R^3-n}^{R^3} \left(\frac{R^3}{n} - 1 \right)^2$$

$$lim_{R\to+\infty} L(R) = lim_{R\to+\infty} \frac{1}{R^3} \left(R^3 - n - 1 \right) + \frac{n}{R^3} \left(\frac{R^6}{n^2} - 2\frac{R^3}{n} - 1 \right)$$

$$lim_{R\to+\infty} L(R) = lim_{R\to+\infty} \frac{1}{R^3} \left(R^3 - n - 1 \right) + \frac{n}{R^3} \left(\frac{R^6}{n^2} - 2\frac{R^3}{n} - 1 \right)$$

$$lim_{R\to+\infty} L(R) = lim_{R\to+\infty} \frac{R^3}{n}$$

$$lim_{R\to+\infty} \alpha(R) = 3 \quad \text{et} \quad lim_{R\to+\infty} \beta(R) = -\log(n)$$

On retrouve sans surprise que la lacunarité se comporte en limite en R^3. Il est toutefois intéressant de considérer l'écart de la lacunarité observée par rapport à cette limite. La figure 3.9 montre la courbe de lacunarité mesurée sur les configurations issues des simulations précédemment citées (640 monomères, 20 chaînes, $R \in [2, 25]$), ainsi que les accroissements logarithmiques :

$$\alpha(R) = \frac{\log(L(R+1)) - \log(L(R))}{\log(R+1) - \log(R)} \tag{3.14}$$

On remarque que le maximum des accroissements est atteint pour $R = 7$. Sachant que la boîte de simulation est d'environ 6.9 fois la distance de répulsion du potentiel de Lennard Jones, on comprend que cette valeur correspond à la distance optimale de discrétisation des "trous intermoléculaires". Sur la figure 3.10 montrant la distribution du nombre de monomères par cellule, on voit bien qu'à $R = 7$ elle forme un plateau entre 0 et 3 (la moyenne est de 1.86). Inversement l'échelle de discrétisation "optimale" des monomères est atteinte lorsqu'une cellule, du fait des longueurs de liaison, ne peut plus contenir qu'un seul monomère. Cette distance de liaison étant $(0.39 \times \sigma_{LJ})$, cela donne $R_{opt} = 6.9/0.39 = 17.2$. On constate bien que pour les valeurs autour de $R = 18$, les accroissements sont moins importants, car à ces échelles il n'y a quasiment plus que des cellules avec 0 ou 1 atome, avec une proportion minimale de cellules vides.

Finalement il apparaît que les variations de la lacunarité de la répartition des monomères en fonction de l'échelle considérée reflètent en partie la structure des configurations. Il reste maintenant à voir si l'utilisation des critères fondés sur cette lacunarité peut s'avérer utile et à quelles échelles, et pour répondre à cette question nous renvoyons le lecteur au chapitre 6.

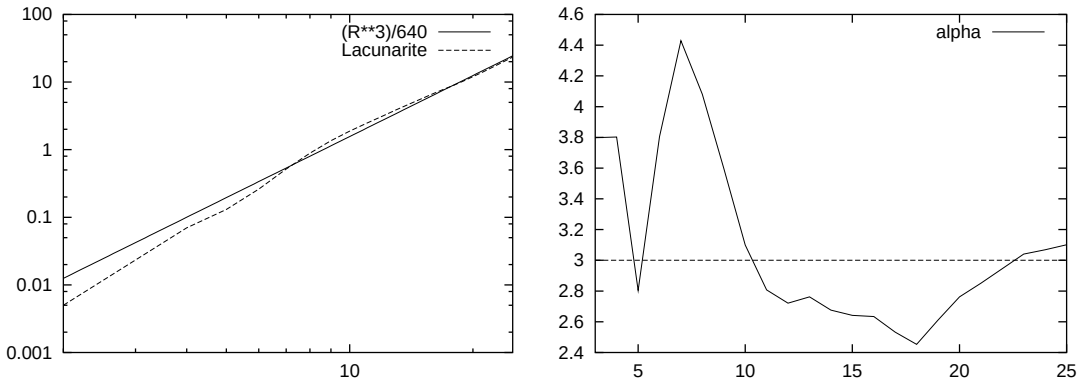


FIG. 3.9 – Gauche : Lacunarité observée, et lacunarité limite en échelle log/log. Droite : accroissements logarithmiques de la lacunarité observée.

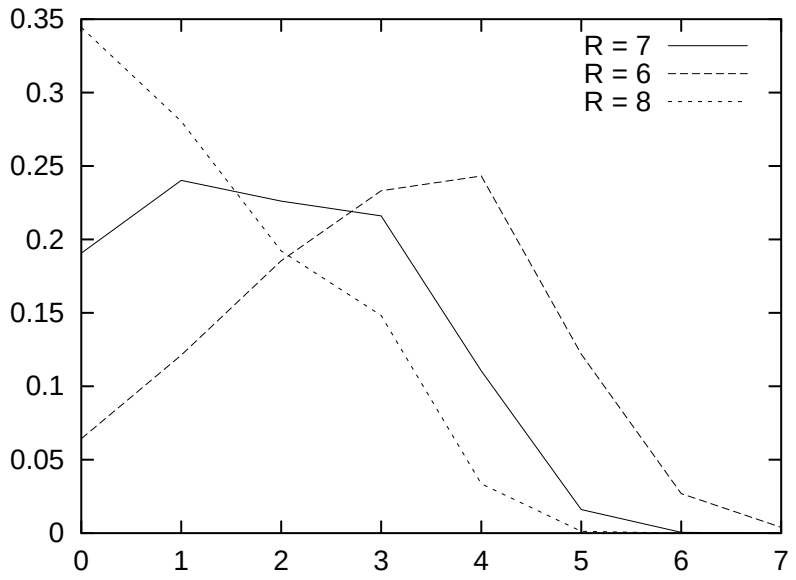


FIG. 3.10 – Distribution du nombre de monomères par cellules, pour $R = 6, 7, 8$.

Chapitre 4

Optimisation des fréquences des mouvements de Monte Carlo.

Face à la possibilité d'appliquer simultanément plusieurs mouvements de Monte Carlo lors de la simulation moléculaire de polymères amorphes en état dense, nous proposons un algorithme évolutionnaire réglant dynamiquement leurs fréquences d'utilisation dans le but d'obtenir la meilleure efficacité possible. La section 4.1 décrit le problème et l'algorithme proposé, et dans les sections suivantes nous présentons les résultats de simulation qui illustrent à la fois la validité de cet algorithme ainsi que le besoin de définir un critère numérique d'efficacité qui soit toujours adapté aux conditions de simulation.

4.1 Définition du problème et premier algorithme proposé.

4.1.1 Cadre d'application.

En section 2.3, nous avons vu comment mettre en oeuvre un algorithme de MCM pour effectuer une simulation moléculaire de polymères amorphes en état dense, et en section 2.3.4 les raisons pour lesquelles une telle simulation peut être inefficace. En outre, nous avons vu en section 2.3.3 que pour faire face à ce problème, de nombreux mouvements de Monte Carlo ont été créés. Nous voulons alors voir s'il est possible d'améliorer encore les performances en se concentrant non plus sur la sophistication d'un type de mouvement mais plutôt en jouant sur la manière d'utiliser conjointement plusieurs d'entre eux.

Nous proposons alors de considérer des simulations parallèles, c'est-à-dire plusieurs simulations parallèles du même système. Dans ce cas, si l'on cherche à obtenir un échantillon de l'espace des phases associé, tous les états parcourus par les différentes simulations contribuent à l'élaboration de l'échantillon. Ceci permet, outre de pouvoir utiliser des architectures parallèles, de comparer différents modèles distribués. Et plus spécifiquement notre but est d'évaluer l'efficacité des algorithmes d'évolution artificielle pour le réglage des fréquences des mouvements de Monte Carlo, habituellement réglés empiriquement. On peut alors énoncer notre problème comme suit :

> Dans le cadre de la simulation moléculaire de polymères amorphes en état dense, étant donné un modèle moléculaire, des conditions physico-chimiques et un ensemble de mouvements de Monte Carlo $(M_1, ..., M_m)$ applicables dans ces conditions, trouver une distribution de fréquences $(\mu_1, ..., \mu_m)$ associées à ces mouvements maximisant un critère d'efficacité de la simulation.

Il reste donc à définir explicitement une mesure de performance à partir de celles que nous avons déjà recensées en section 2.3.4.

4.1.2 L'algorithme de référence

Afin de pouvoir effectuer une comparaison de performances, nous avons besoin d'un algorithme de référence (AR). Il nous a semblé alors logique de considérer n instances d'une simulation classique partant de n configurations initiales différentes et utilisant les mêmes paramètres de Monte Carlo réglés a priori comme on le ferait pour une simulation unique, sans donner de préférence arbitraire à tel ou tel mouvement, soit en fixant $\mu_1 = ... = \mu_m = (\frac{1}{m})$. Ajoutons que les mêmes configurations initiales devront êtres utilisées pour les approches alternatives afin de ne pas biaiser la comparaison.

4.1.3 L'algorithme évolutionnaire

Le codage

La première étape pour définir un algorithme évolutionnaire (AE) est de choisir une représentation génétique. Dans notre cas, notre espace de recherche étant l'ensemble des fréquences possibles applicables à m mouvements différents, il semble donc naturel de considérer un codage génétique dit "réel". Nous définirons donc le génome d'un *individu* par vecteur réel de dimension m, qui correspondra alors explicitement aux fréquences $(\mu_1, ..., \mu_m)$, avec les contraintes :

$$\forall i \in \{1, ..., m\},\ \mu_i \in]0, 1[\ \text{ et } \sum_{i=1}^{m} \mu_i = 1$$

Nous noterons donc S_μ cet espace, correspondant simultanément à l'espace *génotypique* et à l'espace *phénotypique* :

$$S_\mu = \left\{ \mu \in \prod_{i=1}^{m}]0,1[\ ; \ \sum_{i=1}^{m} \mu_i = 1 \right\} \tag{4.1}$$

Cet espace étant contraint, nous faisons le choix d'appliquer une phase dite de *réparation* après l'application des opérateurs génétiques (en l'occurrence le croisement barycentrique et la mutation gaussienne, définis en 2.5.5, sont applicables). Cette phase consiste juste à s'assurer que les nouveaux individus vérifient les contraintes. En l'occurrence, si un vecteur μ^* représentant le génome d'un nouvel individu n'est pas normalisé, la réparation sera simplement sa normalisation :

$$\mu = \left(\frac{1}{\sum_{i=1}^{m} \mu_i^*} \right) \mu^*$$

Evaluation des performances : fonction de fitness

Nous commencerons par définir les conditions dans lesquelles seront évalués les individus, pour cela il est nécessaire de décrire le déroulement global de l'algorithme :

- n_s systèmes de mêmes paramètres physico-chimiques sont simulés en parallèle, .
- Une population contient n_s individus, codant chacun des distributions de fréquences différentes. La première population est générée aléatoirement.
- Une simulation particulière consiste en n_c *cycles*.
- La durée totale de simulation est segmentée en n_g générations durant lesquelles les individus de la population courante sont évalués (sur une période de (n_c/n_g) cycles). On applique la sélection et les opérateurs génétiques pour obtenir une nouvelle population et les nouveaux paramètres correspondants sont attribués aux différentes simulations.

Nous définissons ici un **cycle** de simulation de manière pragmatique. Nous attribuons un temps de simulation (temps du processus, au sens informatique) égal quel que soit le nombre de mouvements effectués. Cette manière de faire rend évidemment la rapidité de simulation dépendante de plusieurs facteurs : la programmation des mouvements, les performances des plates-formes utilisées (vitesse des processeurs, taille de mémoire cache, rapidité d'accès à la mémoire, etc...). Elle privilégie implicitement les mouvements qui sont à la fois rapides en termes de temps de calcul et efficaces pour le mélange du système.

L'évaluation des individus se fait donc sur une période de (n_c/n_g) cycles au cours desquels on mesure la performance de la simulation. Les critères d'efficacité que nous utilisons sont alors un critère d'autocorrélation des vecteurs d'orientation des chaînes et un critère de déplacement des centres de masse, décrits au chapitre précédent (section 3.1). Toutefois les séries de simulations de ce chapitre ont été réalisés avant que nous n'utilisions des critères dits "cumulés", c'est pourquoi nous utiliserons le critère d'autocorrélation instantanée des vecteurs des chaînes $C_v^i(t)$ et le déplacement carré moyen des centres de masse des chaînes $d^2(t)$ (nous abandonnerons alors la variable t étant donné que les performances sont toujours mesurées sur la même durée de simulation). La fonction de fitness d'un individu $\mu \in S_\mu$ est alors le produit de 2 termes que l'on souhaite maximiser :

$$f(\mu, \omega) = \left[1 - C_v^i(\mu, \omega) \right] \times d^2(\mu, \omega) \tag{4.2}$$

Remarquons que la variable ω est représentative de la part aléatoire de la simulation de Monte Carlo durant laquelle l'individu est évalué. On peut donc considérer que notre fonction de fitness est bruitée.

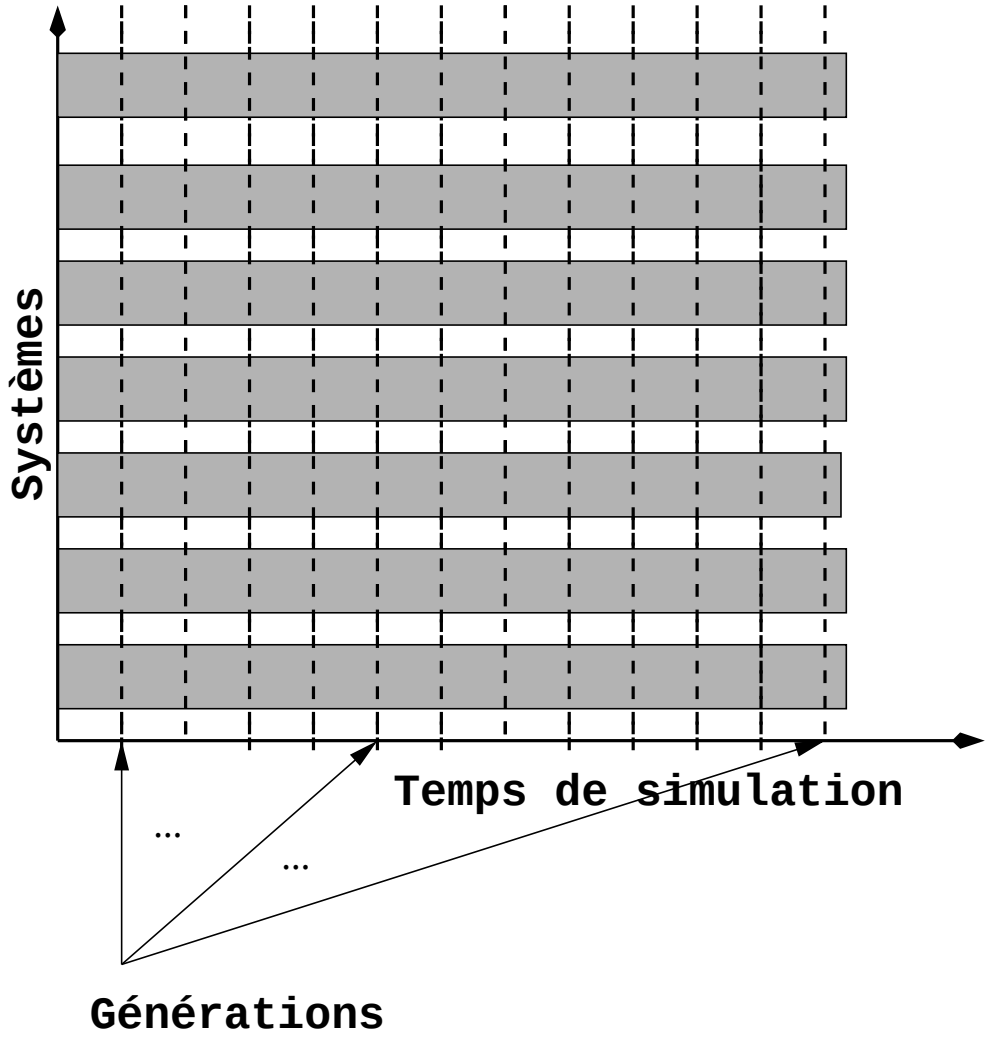


FIG. 4.1 – Représentation schématique de l'algorithme évolutionnaire : les barres grises représentent les simulations des différents systèmes. Le temps de simulation est segmenté en générations au terme desquelles de nouveaux paramètres pour les mouvements de Monte Carlo sont attribués à chacun des systèmes.

4.1.4 Spécifités de cet algorithme.

L'algorithme que nous proposons ici présente les caractéristiques essentielles d'un algorithme classique d'EA : évaluation d'une population de solutions potentielles au moyen d'un fonction de *fitness* permettant son évolution grâce à la sélection et aux opérateurs génétiques. Toutefois certaines spécificités méritent d'être soulignées. En premier lieu, on constate que l'évaluation d'un individu, résultant d'un longue période de simulation, est à la fois longue en termes de temps de calcul et bruitée comme nous venons de le voir dans la section précédente. Si ces conditions ne sont pas nouvelles dans le domaine, la recherche de moyens spécifiques d'en tenir compte est un sujet qui n'a pas été initialement privilégié en EA. Néanmoins la diversité et la nature des applications de l'EA apportant régulièrement de nouveaux types de problèmes, le cas des fonctions de fitness bruitées a commencé par être abordé lui aussi. Et c'est pourquoi nous proposerons en section 4.1.5 une solution pour ce problème.

La seconde spécificité tient alors à la manière dont sont utilisés les résultats de l'évolution. En effet les algorithmes d'EA sont essentiellement utilisés comme algorithmes d'optimisation stochastique, et il est d'usage d'extraire le meilleur individu ou les meilleurs individus produits au terme de l'évolution. Dans notre cas, si cet aspect est également envisageable, la priorité est alors d'obtenir rapidement une population dont tous les individus soient jugés efficaces, dans la mesure où l'optimisation des paramètres est faite pendant la simulation même. En d'autres termes nous ne sommes pas seulement intéressé par le meilleur individu produit au terme de l'évolution mais aussi par la qualité moyenne de tous les nouveaux individus produits. Notre algorithme peut alors être vu comme un outil de contrôle dynamique.

4.1.5 Réduction du bruit de la fonction de fitness

Comme nous l'avons vu en section 2.6.4, les méthodes classiques pour faire face à un environnement bruité résultent généralement en une augmentation significative du nombre d'évaluations. Dans notre cas, il est difficile d'augmenter beaucoup la taille de population, car cela implique d'augmenter d'autant le nombre de simulations simultanées, or nous avons tout intérêt à essayer d'être efficace avec la plus petite population possible. De même, multiplier les évaluations pour une même individu signifie également multiplier le nombre de systèmes simulés, ou alors diviser la taille effective de la population (à nombre de systèmes égal).

Il nous a paru donc plus judicieux d'essayer de tirer parti du fait que l'évolution sélectionne plusieurs fois le même individu, ou ses descendants (via les opérateurs génétiques), en d'autres termes de profiter de l'historique de toutes les populations créées. D'autre part, si l'on considère une distance euclidienne pour notre codage réel, deux individus proches auront des fréquences proches, et on comprend alors que leurs performances le seront aussi. La démarche que nous avons adoptée est d'utiliser à la fois la multiplicité implicite et de la proximité des points évalués pour calculer une valeur de fitness pondérée à la fois temporellement (plusieurs évaluations du même point) et spatialement (distances proches). Cette démarche a d'ailleurs récemment été expérimentée par Sano et Kita [60]. Leur approche est alors fondée sur l'utilisation d'un modèle stochastique de la fonction de fitness qui permet l'utilisation de la technique du maximum de vraisemblance pour l'estimation de la fonction de fitness sous-jacente.

Tout d'abord nous définissons les notations suivantes :

– Distance max sur S_μ :

$$d_\infty(\mu, \nu) = max_{i \in \{1,..,m\}} |\mu_i - \nu_i| \tag{4.3}$$

– Distance *euclidienne* sur S_μ :

$$d_2(\mu,\nu) = \sqrt{\sum_{i=1}^{m} (\mu_i - \nu_i)^2} \tag{4.4}$$

– Le voisinage d'un point est alors défini en utilisant la distance max :

$$\forall \mu \in S_\mu,\ \sigma_\infty \in \mathbb{R}_+^*,\ \ \mathrm{B}_{\sigma_\infty}(\mu) = \ \{\nu \in \mathrm{S}_\mu;\ \mathrm{d}_\infty(\mu,\nu) \leq \sigma_\infty\} \tag{4.5}$$

L'avantage d'utiliser la distance max pour déterminer le voisinage d'un point est qu'il est plus rapide de déterminer si deux points sont assez proches au sens de cette distance. En effet dès que l'on trouve un indice i tel que $|\mu_i - \nu_i| > \sigma_\infty$ on sait que $\nu \notin B_{\sigma_\infty}(\mu)$.

– L'ensemble $H = \left\{ \left(\nu_i, n_{\nu_i}, \tilde{f}(\nu_i)\right), i \in \{1, ...n_H\} \right\}$ des points pour lesquels on a n_{ν_i} évaluations et dont $\tilde{f}(\nu_i)$ est la moyenne : l'historique des évaluations.

Nous utiliserons alors alors une notion de régularité de fonction de fitness qui est la suivante : les valeurs de fitness des individus proches d'un individu μ (appartenant au voisinage $B_{\sigma_\infty}(\mu)$) seront aussi proches de $f(\mu)$. La régularité au sens de Hölder est alors un outil approprié à cet effet (voir l'annexe A).

Pratiquement nous proposons une pondération de la fonction de fitness de la forme suivante :

$$f_H(\mu) = \frac{\sum_{\nu \in H \cap B_{\sigma_\infty}(\mu)} \left(w(\mu,\nu) \times n_\nu \times \tilde{f}(\nu) \right)}{\sum_{\nu \in H \cap B_{\sigma_\infty}(\mu)} \left(w(\mu,\nu) \times n_\nu \right)} \tag{4.6}$$

Où $w(\mu,\nu)$ est un terme pondérateur défini par :

$$w(\mu,\nu) = \left(1 - \frac{d_2(\mu,\nu)}{\sqrt{m} \times \sigma_\infty}\right) \tag{4.7}$$

On a bien $w(\mu,\mu) = 1$ et de plus comme :

$$max(d_2(\mu,\nu),\ \nu \in B_{\sigma_\infty}(\mu)) = \sqrt{m} \times \sigma_\infty \text{ et } d_\infty(.,.) \geq d_2(.,.)$$

On est assuré que :

$$\forall \mu \in S_\mu,\ \forall \nu \in B_{\sigma_\infty}(\mu),\ w(\mu,\nu) \geq 0$$

Ainsi à chaque génération, on ajoute n_s évaluations (valeurs de fitness "instantanée') à l'historique H, puis la valeur de fitness pondérée est calculée en ces points. Ainsi un individu représenté plusieurs fois dans l'historique a une valeur de fitness qui est la moyenne pondérée de ses propres évaluations ainsi que de celles de ses voisins éventuels.

4.2 Simulations numériques : premier algorithme.

4.2.1 Choix de simulation.

Nous reprendrons le modèle moléculaire défini en 2.3.1, ainsi que certains des mouvements décrits en 2.3.3.

Rayon de coupure et cellules de voisinage.

Ajoutons qu'afin d'écourter le temps de calcul du potentiel d'interaction de LJ, nous utilisons un découpage en cellules de la boîte de simulation, méthode classique décrite en détail par exemple dans [1], page 149-152. Pour tout calcul d'interaction inter-sites, où seuls les sites proches (en deçà d'un rayon de coupure σ_{cut}) interviennent, et si la largeur des cellule est supérieure à σ_{cut}, il suffit alors de parcourir la cellule contenant ce site et ses cellules voisines. Bien sûr le gain n'est valable que si le nombre de cellules est au moins $4^3 = 64$, ce qui impose que la largeur de la boîte de simulation soit supérieure à $(4 \times \sigma_{cut})$. De plus nous utilisons le systèmes d'*unités réduites* (voir par exemple [23], pages 36-37) qui accélèrent le calcul des potentiels. Par exemple les distances ne sont plus mesurées en Å mais en multiples de σ_{LJ}, distance caractéristique du potentiel de Lennard-Jones.

Fréquences à charge égale.

En outre il est utile de remarquer que les mouvements n'impliquent pas tous la même charge de calcul, car précisément ils n'impliquent pas tous le même nombre de degrés de liberté. Nous avons donc choisi d'en tenir compte pour le codage des fréquences des mouvements de notre AE. Le génome d'un individu code alors des fréquences "à charge égale" (FCE). Les fréquences effectives (FE) (c'est-à-dire celles utilisées effectivement lors du choix d'un mouvement à appliquer) sont alors calculées en divisant les FCE par le nombre de monomères déplacés (noté *deg*) de chacun des mouvements et en renormalisant :

$$FE(i) = \frac{\frac{FCE(i)}{deg(i)}}{\sum_{j=1}^{m} \frac{FCE(j)}{deg(j)}} \tag{4.8}$$

Nous renvoyons à la section 4.2.5 pour une série d'expérimentations numériques illustrant l'avantage du codage des FCE plutôt que des FE.

Présentation des résultats.

Dans les expérimentations numériques qui suivent, nous comparerons à chaque fois les résultats de l'AE avec un AR dans les mêmes conditions. Les fréquences à charge égale (FCE) des différents mouvements pour les simulations AR seront alors égales. Pour les simulations AE, nous présenterons, pour chaque mouvement, les histogrammes de toutes les FCE représentées dans toutes les populations au cours de toute la simulation. Enfin nous tirerons profit de la construction de l'historique H (voir section 4.1.5), pour présenter les individus les plus performants "en moyenne" (dont le nombre d'évaluations est suffisamment élevé).

Plateforme de simulation.

Les simulations ont été effectués avec des programmes écrits lors de cette thèse, en langage C, parallélisés pour certains avec la librairie PVM (Parallel Virtual Machine), puis en utilisant directement le protocole des "socket" UNIX. Elles ont été réalisées pour la plupart sur une ferme de 220 stations PC Pentium III à $733MHZ$ sous Linux hébergée à l'INRIA Grenoble, dont l'utilisation a fait l'objet d'un accord avec le projet APACHE (Algorithmique, Programmation Parallèle et Partage de Charge). APACHE est un projet de recherche sur le calcul parallèle, développé au sein du laboratoire Informatique et Distribution (ID) de l'Institut d'Informatique

et de Mathématiques Appliquées de Grenoble (IMAG). Il est soutenu de manière conjointe par le CNRS, l'INPG (Institut National Polytechnique de Grenoble), l'INRIA Grenoble et l'UJF (Université Joseph Fourier).

4.2.2 Série A : ensemble $nNPT$, longueur de chaîne 32

Paramètres

Cet ensemble et les conditions de simulations qu'il implique sont décrites en 2.3. Les paramètres sont :

- $n = 640$
- $N = 20$. Les chaînes comptent alors 32 monomères
- $P = 1atm$
- $T = 450K$
- $n_c = 5000$ cycles de simulations (1 cycle = 1seconde).

Les paramètres de l'AE sont les suivants :

- population de taille 16, soit 16 systèmes simulés en parallèle. Cette taille est relativement réduite, mais c'est ici un choix explicite, car le but est de montrer qu'il est possible d'obtenir des résultats satisfaisants avec un nombre restreint de simulations simultanées.
- remplacement complet de la population avec sélection stochastique universelle (roue à N pointeurs, voir section 2.5.3).
- découpage de la simulation en $n_g = 50$ générations, soit 100 cycles pour une génération.
- mutation gaussienne avec $p_m = 0.1$. La mutation n'intervient ici qu'une seule fois par individu (chaque individu est testé, puis un de ses gènes est tiré au hasard et muté). Cette manière de faire rend plus explicite la probabilité qu'un individu ne soit pas modifié par la mutation.
- croisement uniforme avec $p_c = 0.5$. Ce qui correspond à un compromis entre sauvegarde des individus sélectionnés et recombinaison.
- rayon de voisinage pour la *fitness pondérée*. $\sigma_{max} = 0.05$

Les mouvements utilisés sont :

- La Translation ($deg = \frac{n}{N}$)
- La Rotation ($deg = 1$)
- La Reptation ($deg = 1$)
- Le Flip ($deg = 1$)
- La fluctuation de volume ($deg = n$)

Sachant qu'il y a 5 mouvements, toutes les FCE de l'AR sont égales à 0.2 ce qui correspond aux FE suivantes :

Translation	Rotation	Reptation	Flip	Fluctuation volumique
20 / 1941	640 / 1941	640 / 1941	640 / 1941	1 / 1941

Résultats

Les figures 4.2 et 4.3 montrent les performances globales en termes d'autocorrélation des vecteurs de chaînes et déplacement moyen des centres de masse. On remarque que sur ces deux critères, l'AE se comporte mieux. La figure 4.4 montre les variations de volume lors du simulation AE.

Partant du volume moyen on peut retrouver le volume spécifique correspondant en notant V^* le volume de la boîte en unités réduites, M_{mol} la masse molaire et N_A le nombre d'Avogadro :

$$v = \frac{V^* \times \sigma_{LJ}^3}{N} \times \frac{N_A}{M_{mol}(C_{32}H_{66})} \tag{4.9}$$

Dans notre cas avec un volume moyen de 333 on obtient $v = 1.36\ cm^3/g$, ce qui correspond aux résultats présentés en [50], où pour des chaînes de taille 24 dans les mêmes conditions on a $v = 1.41\ cm^3/g$, et $v = 1.27\ cm^3/g$ pour des chaînes de taille 78.

Les histogrammes des FCE de la simulation AE (figures 4.5 et 4.6) montrent une préférence pour le mouvement de Reptation au détriment du Flip. Les figures 4.7 et 4.8 montrent l'évolution des valeurs de fitness au cours des générations : moyenne de la population, valeur minimale, valeur maximale. On remarque en premier lieu l'effet du bruit de la fonction de fitness, non seulement sur la valeur moyenne mais surtout sur la valeur maximale, ce qui reflète bien le fait qu'un individu nouvellement créé hors du voisinage d'autres individus, peut avoir une évaluation "heureuse", créant ainsi ces pics, sa valeur de fitness pondérée étant alors égale à sa valeur de fitness instantanée. Le meilleur individu trouvé dans l'ensemble H dont le poids w est supérieur à 20 a une valeur de fitness pondérée de 0.098 ce qui est supérieur à la moyenne, mais inférieur au moins de moitié au maximum des valeurs de fitness instantanée. La figure 4.9 montre la courbe de la moyenne, du minimum et du maximum d'autocorrélation des vecteurs de chaînes (sur les 16 systèmes simulés) au cours de la simulation AR, qui utilise les mêmes paramètres pour chacun des systèmes. La disparité que l'on constate alors explique la variance élevée de la fitness instantanée.

Nous reprenons alors les FCE du meilleur individu pour effectuer une nouvelle simulation "AR optimisé" dont les performances sont présentées sur les figures 4.10 et 4.11. Ces fréquences sont :

Mouvements	Translation	Rotation	Reptation	Flip	Fluctuation volumique
FCE	19.4 %	18.2 %	43.9 %	13.5 %	5 %
FE	0.8 %	23.9 %	57.6 %	17.6 %	0.01 %

Remarquons d'ailleurs que ces valeurs ne correspondent pas systématiquement au pics des histogrammes des FCE. Les performances observées sont alors les meilleures des 3 simulations, sans pour autant nettement dépasser celles de l'AE.

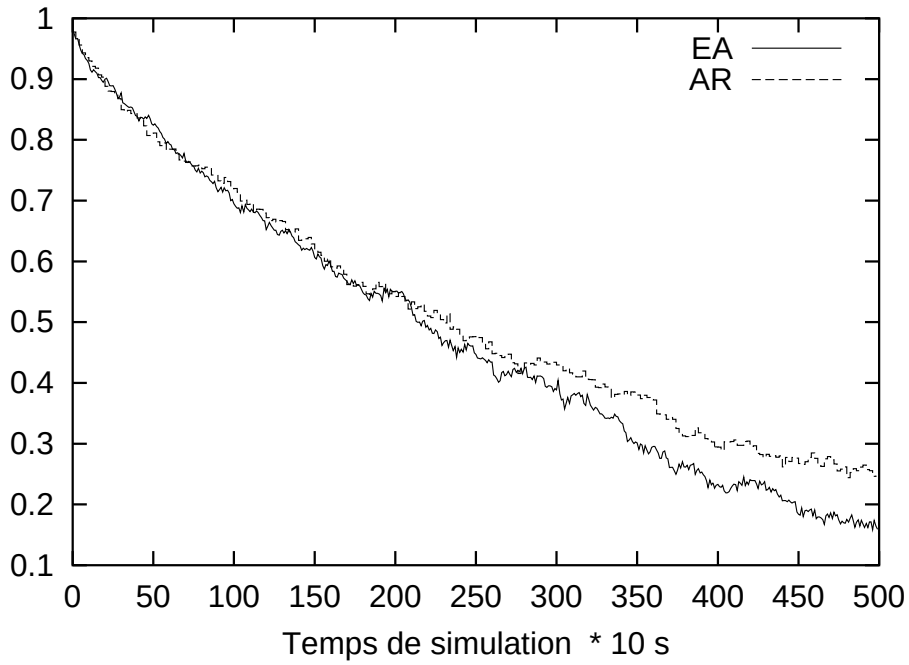


FIG. 4.2 – Série A : Courbe d'autocorrélation instantanée de vecteurs de chaînes.

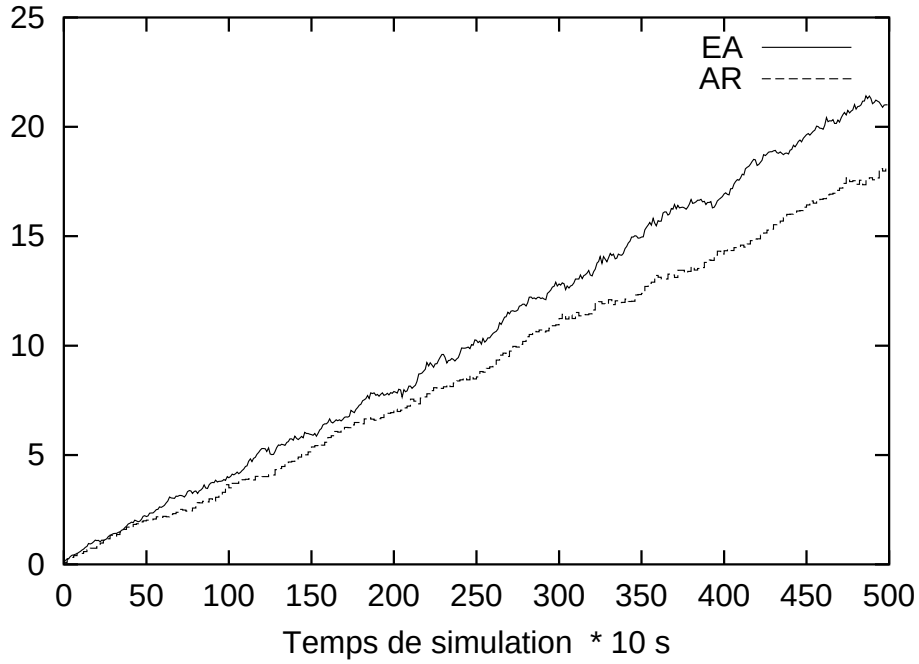


FIG. 4.3 – Série A : Courbe de déplacement carré moyen moyen du centre de masse en unités réduites (1 unité = σ_{LJ}^2)

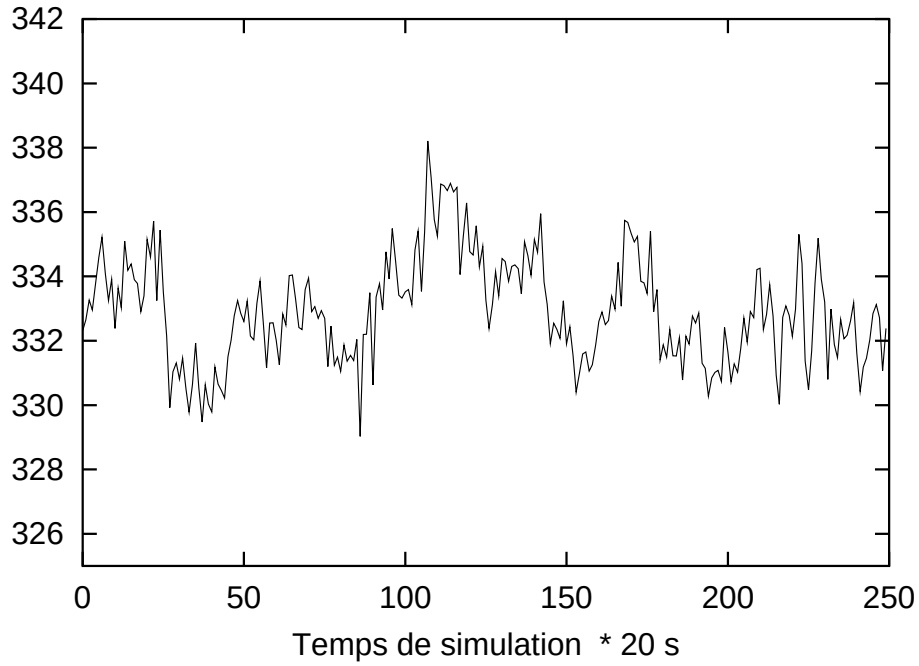


FIG. 4.4 – Série A : volume moyen des boîtes de simulation en unités réduites (1 unité $= \sigma_{LJ}^3$).

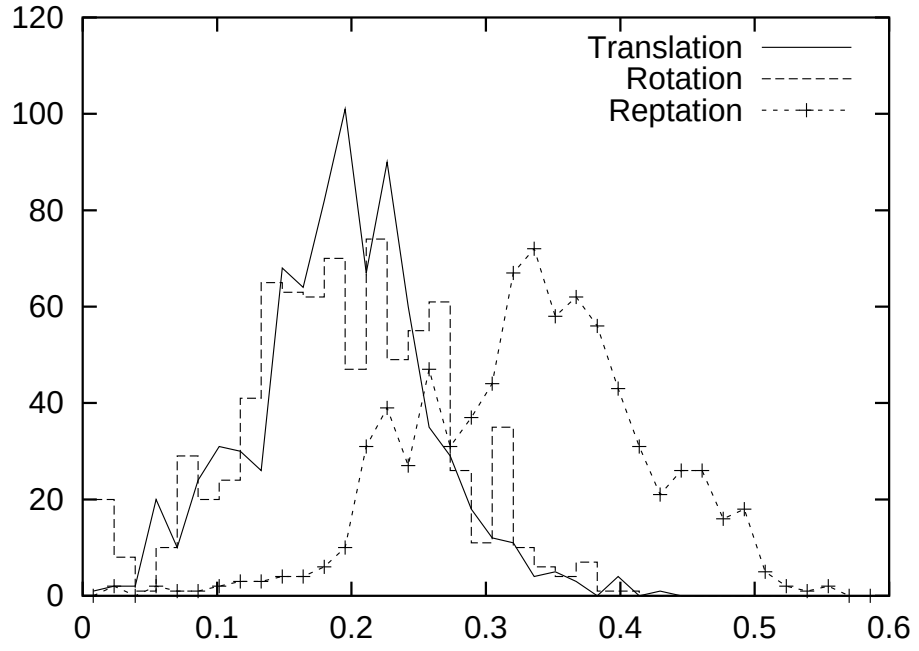


FIG. 4.5 – Série A : Histogrammes de FCE des mouvements Translation, Rotation et Reptation.

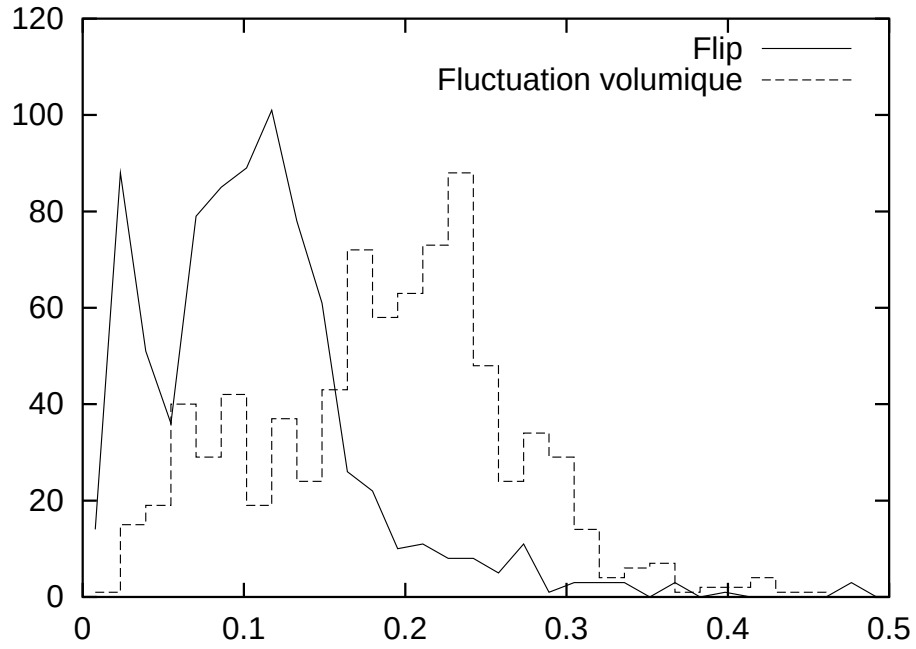


FIG. 4.6 – Série A : Histogrammes de FCE des mouvements Flip et Fluctuation Volumique.

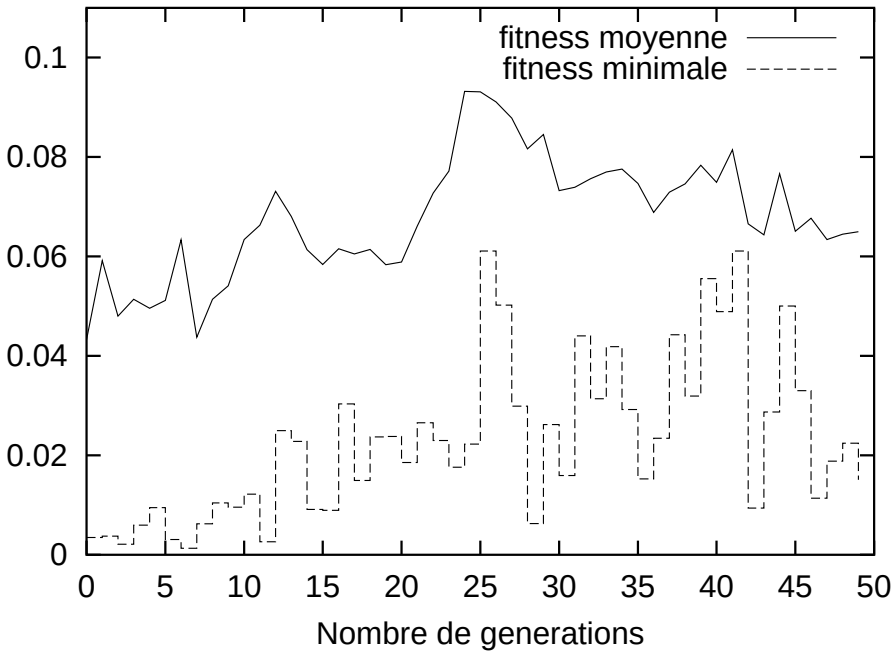


FIG. 4.7 – Série A : Courbes des valeurs moyenne et minimales de fitness au cours des générations.

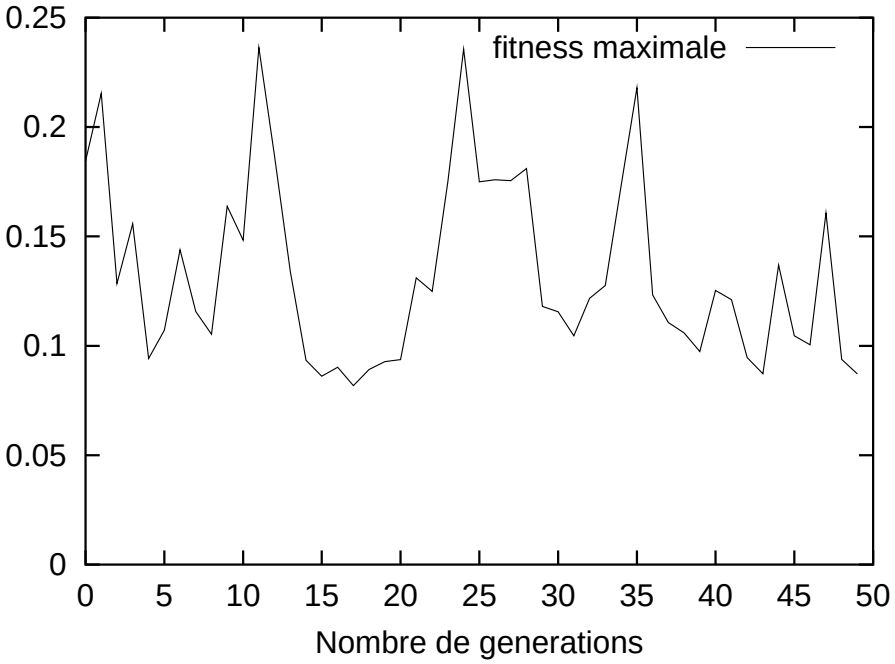


FIG. 4.8 – Série A : Courbe des valeurs maximales de fitness au cours des générations.

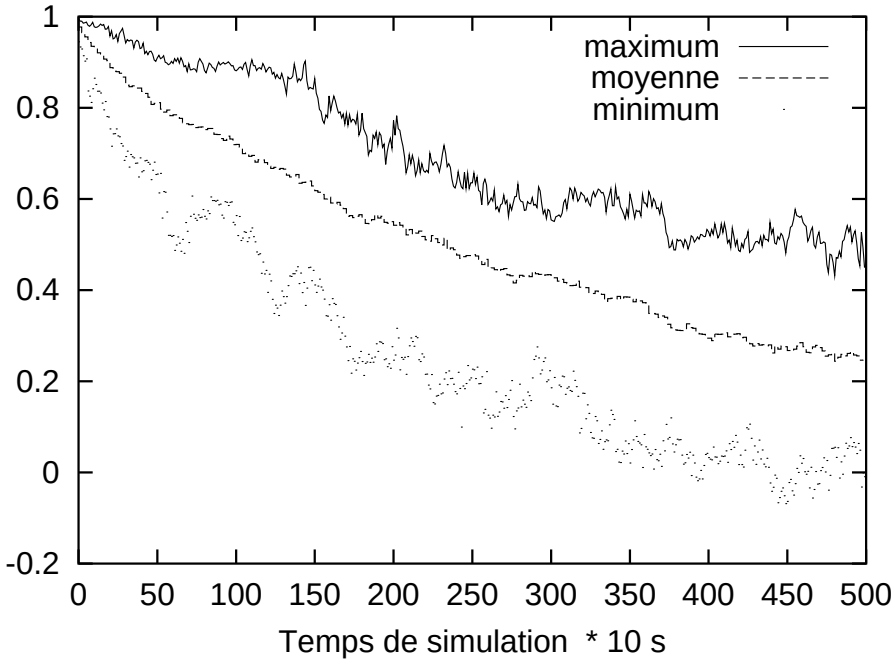


FIG. 4.9 – Série A : Courbes du maximum, minimum et moyenne d'autocorrélation instantanée des vecteurs de chaîne pour la simulation AR.

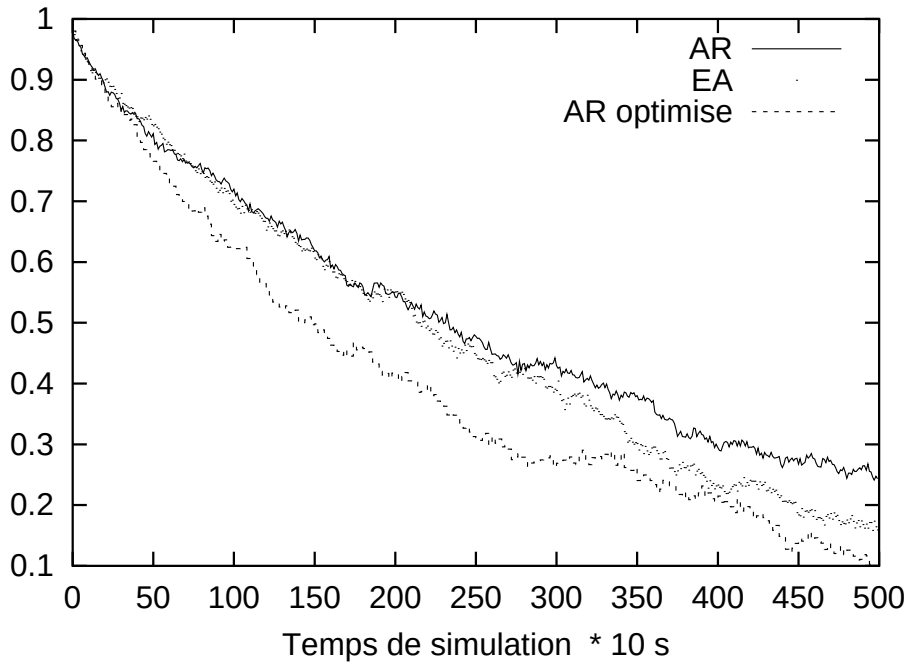


FIG. 4.10 – Série A : Autocorrélation instantanée des vecteurs de chaînes. Ajout de l'AR "optimisé".

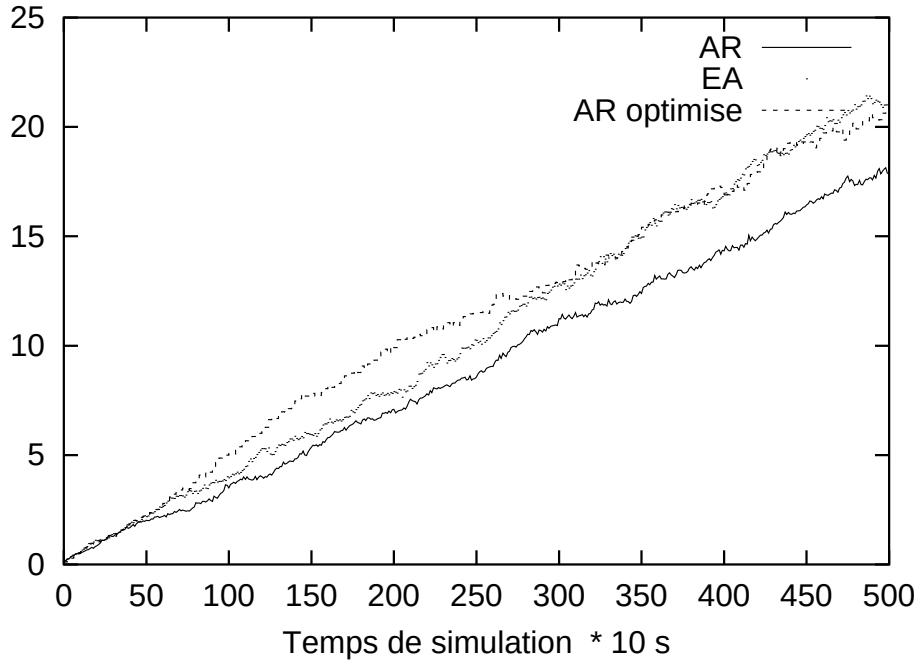


FIG. 4.11 – Série A : Déplacement carré moyen des centres de masse. Ajout de l'AR "optimisé".

4.2.3 Série B : ensemble $nNPT$, longueur de chaîne 64

Paramètres

Les paramètres sont identiques à la série A à l'exception des longueurs de chaîne et du temps de simulation :

- $n = 640$
- $N = 10$. Les chaînes comptent alors 64 monomères
- $P = 1atm$
- $T = 450K$
- $n_c = 50000$ cycles de simulations (1 cycle = 1seconde), soit 10 fois plus que pour la série A.

Les paramètres d'AE sont les suivants :

- population de taille 16, soit 16 systèmes simulés en parallèle.
- remplacement complet de la population.
- découpage de la simulation en $n_g = 50$ générations, soit 100 cycles pour une génération.
- mutation gaussienne avec $p_m = 0.1$
- croisement uniforme avec $p_c = 0.5$
- rayon de voisinage pour la *fitness pondérée*. $\sigma_{max} = 0.05$

Sachant qu'il y a 5 mouvements, toutes les FCE de l'AR sont égales à $\left(\frac{1}{5}\right)$ ce qui correspond aux FE suivantes :

Translation	Rotation	Reptation	Flip	Fluctuation volumique
10 / 1931	640 / 1931	640 / 1931	640 / 1931	1 / 1931

Résultats

On constate que la taille des chaînes ayant été doublée la vitesse de mélange du système est beaucoup plus lente. Cependant, on constate encore une fois que la simulation AE permet d'obtenir des performances un peu meilleures (figures 4.12 et 4.13). Le volume spécifique correspondant est $v = 1.29cm^3/g$, en diminution par rapport à la série A, ce qui s'explique par l'augmentation de la taille des chaînes et en accord avec les résultats présentés en [50].

Le meilleur individu ayant un poids supérieur à 20 est le suivant :

Mouvements	Translation	Rotation	Reptation	Flip	Fluctuation volumique
FCE	18.3 %	16.9 %	38.1 %	8.9 %	17.8 %
FE	0.444 %	26.3 %	59.3 %	13.9 %	0.0433 %

On constate alors encore une fois qu'un "AR optimisé" utilisant ces paramètres se comporte encore mieux (figures 4.20 et 4.21).

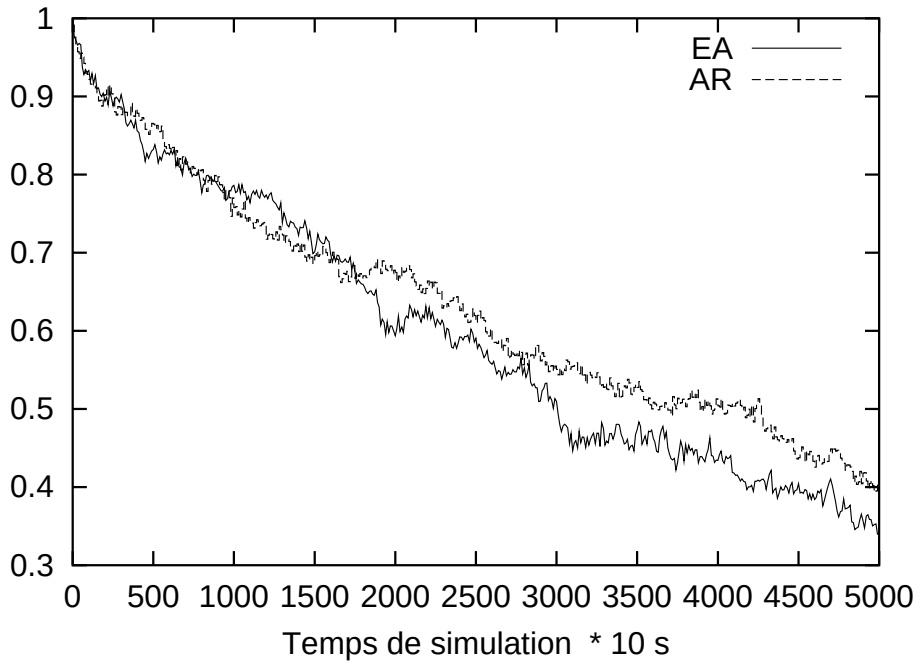


FIG. 4.12 – Série B : Courbe d'autocorrélation instantanée de vecteurs de chaînes.

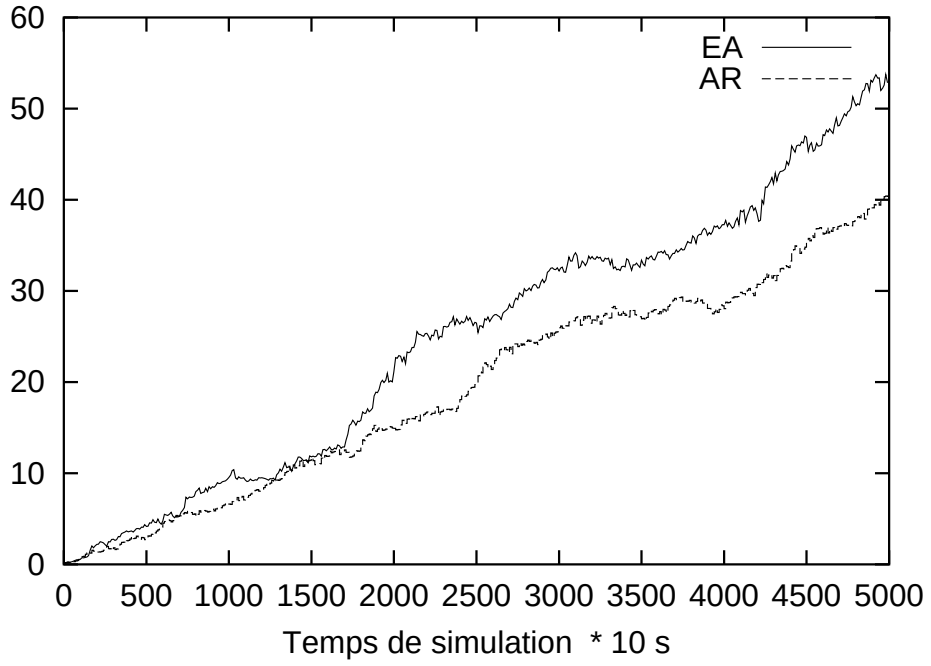


FIG. 4.13 – Série B : Courbe de déplacement carré moyen moyen du centre de masse en unités réduites (1 unité = σ_{LJ}^2).

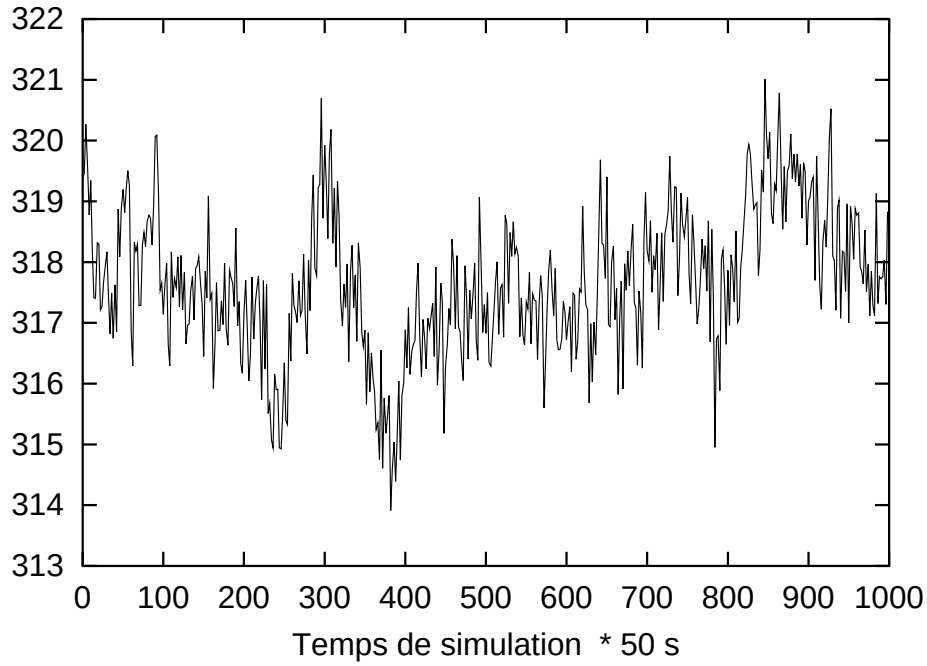


FIG. 4.14 – Série B : volume moyen des boîtes de simulation en unités réduites (1 unité $= \sigma_{LJ}^3$).

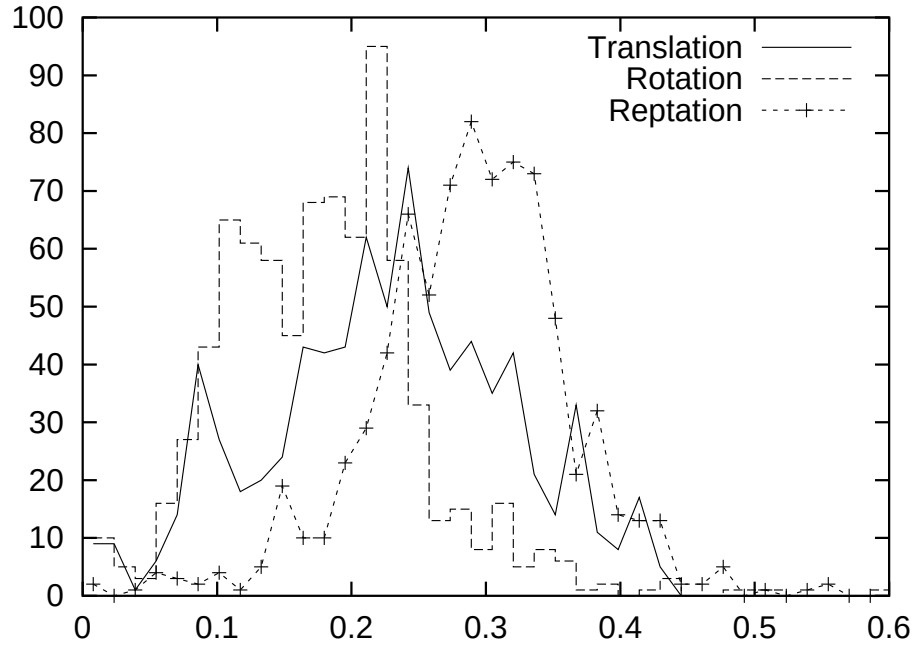


FIG. 4.15 – Série B : Histogrammes de FCE des mouvements Translation, Rotation et Reptation.

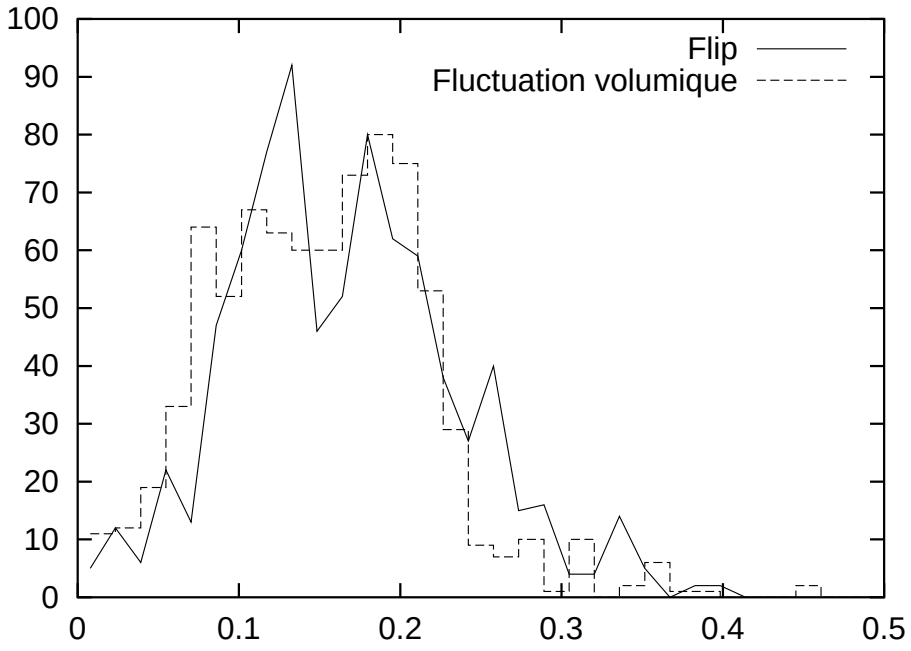


FIG. 4.16 – Série B : Histogrammes de FCE des mouvements Flip et Fluctuation Volumique.

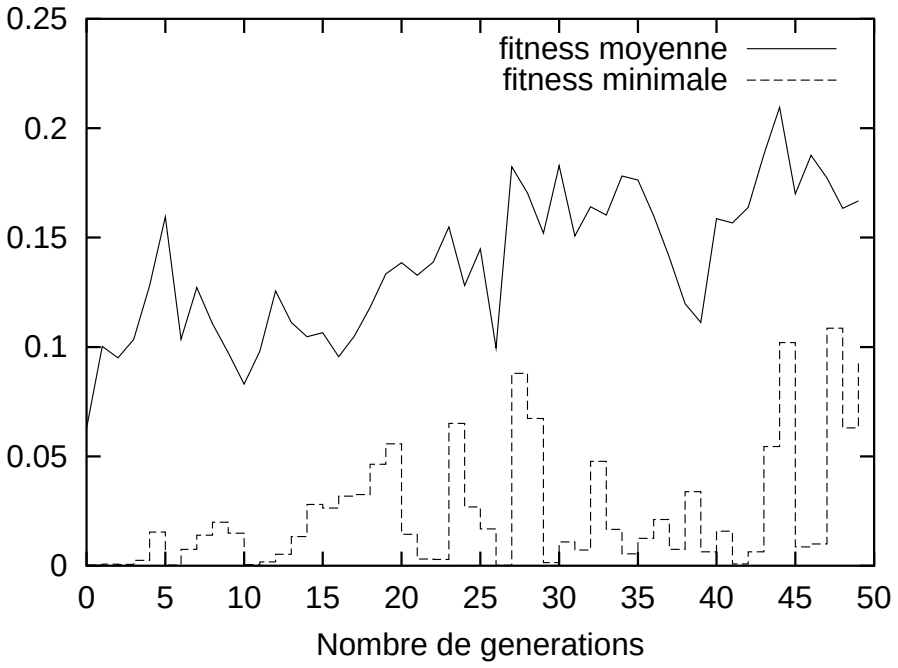


FIG. 4.17 – Série B : Courbes des valeurs moyenne et minimales de fitness au cours des générations.

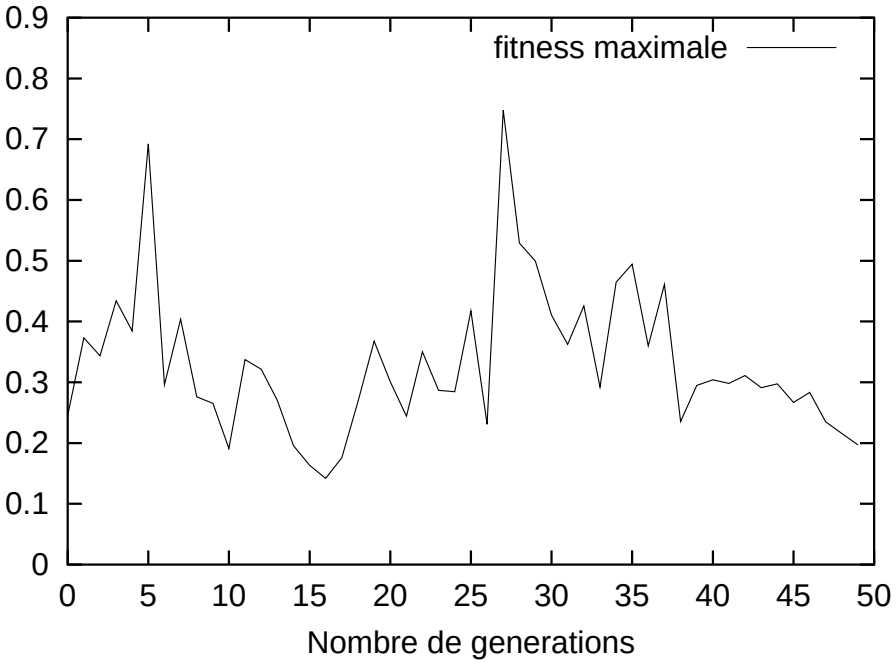


FIG. 4.18 – Série B : Courbe des valeurs maximales de fitness au cours des générations.

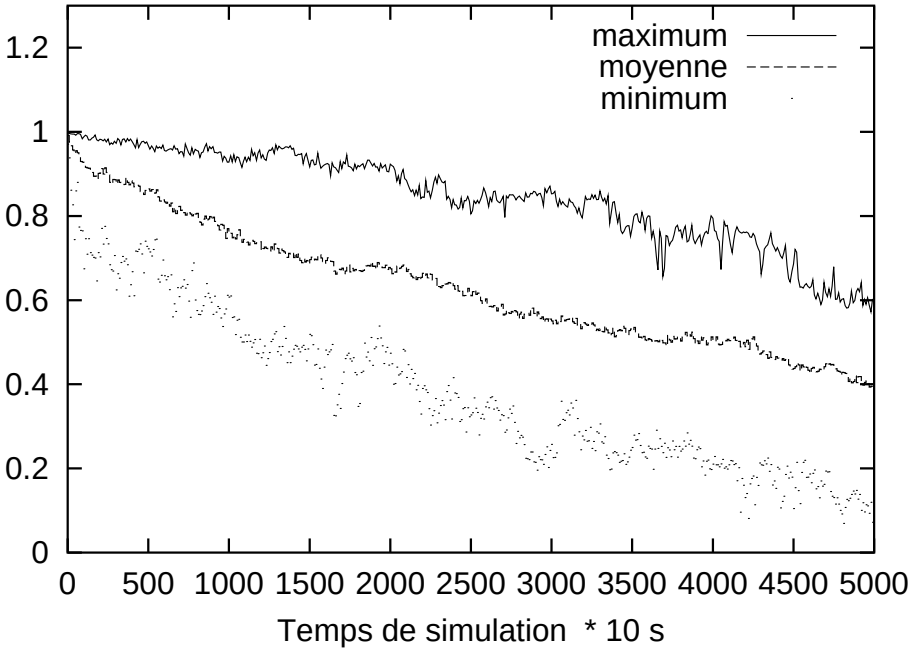


FIG. 4.19 – Série B : Courbes du maximum, minimum et moyenne d'autocorrélation instantanée des vecteurs de chaîne pour la simulation AR.

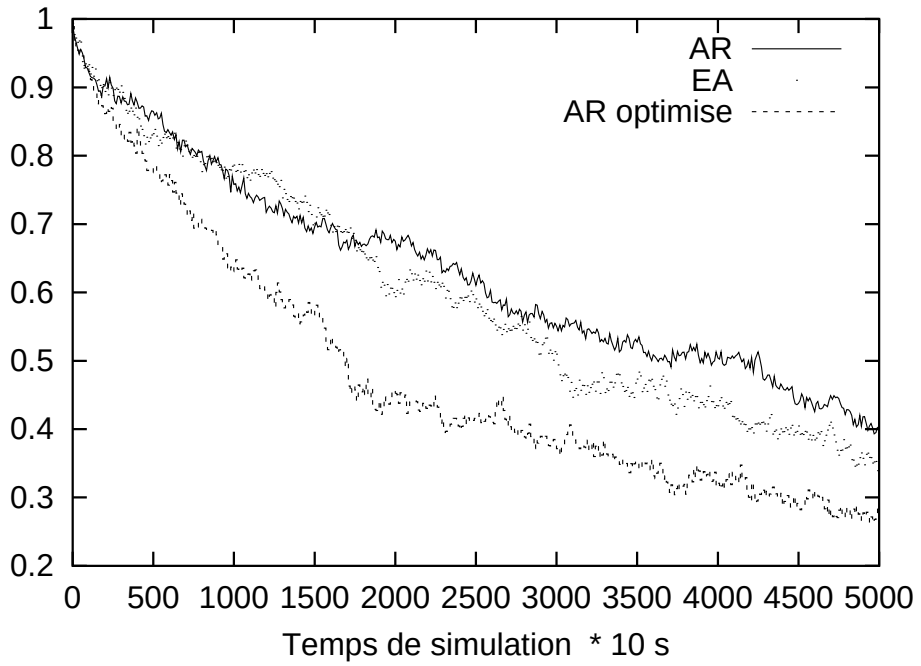


FIG. 4.20 – Série B : Autocorrélation instantanée des vecteurs de chaînes. Ajout de l'AR "optimisé"

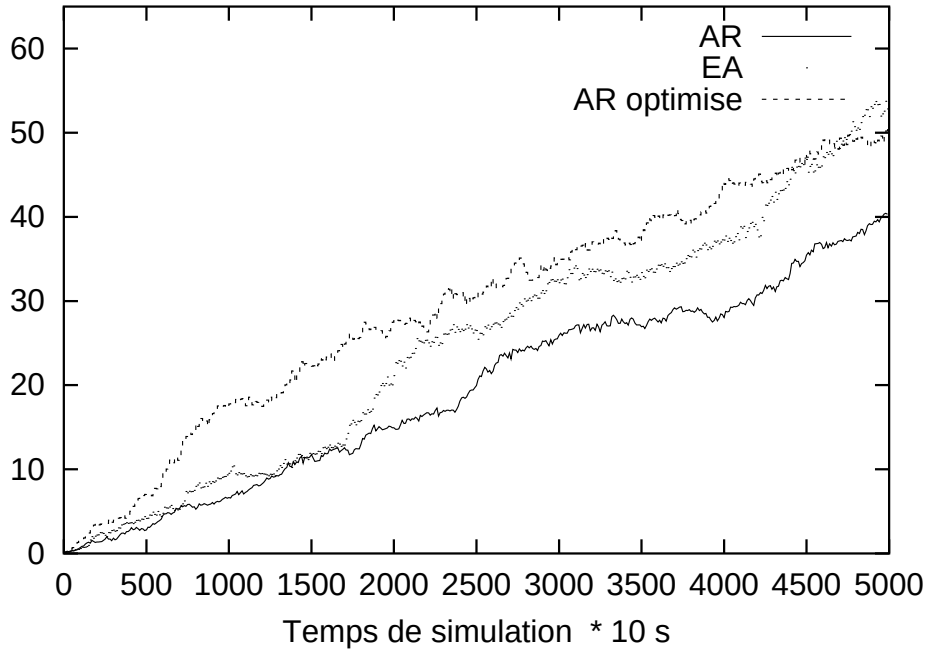


FIG. 4.21 – Série B : Déplacement carré moyen des centres de masse. Ajout de l'AR "optimisé".

4.2.4 Série C : ensemble $nN\mu^*PT$, longueur de chaîne 64

Paramètres

Nous présentons ici une série de simulations numériques réalisées dans l'ensemble statistique $nN\mu^*PT$ (voir section 2.3.2) :

- $n = 640$: nombre de monomères.
- $N = 10$: nombre de chaînes. Soit une taille moyenne de chaîne $\bar{X} = 64$, $\Delta = 0.4$ (voir plus bas).
- μ^* : spectre de potentiels chimiques réduits.
- $P = 1atm$: pression.
- $T = 450K$: température.

Dans cette série de simulations, le Pontage IC remplace le mouvement de Translation. Encore une fois les paramètres de l'EA sont :

- population de taille 16, soit 16 systèmes simulés en parallèle.
- remplacement complet de la population.
- découpage de la simulation en $n_g = 50$ générations, soit 100 cycles pour une génération.
- mutation gaussienne avec $p_m = 0.1$
- croisement uniforme avec $p_c = 0.5$
- rayon de voisinage pour la *fitness pondérée*. $\sigma_{max} = 0.05$

Pour la simulation AR, chaque mouvement a une FCE égale à $\left(\frac{1}{5}\right)$ ce qui correspond aux FE suivantes :

Rotation	Reptation	Flip	Fluctuation volumique	Pontage IC
640 / 2134	640 / 2134	640 / 2134	1 / 2134	213 / 2134

Résultats

On constate en premier lieu sur la figure 4.22 que la distribution des longueurs de chaînes tend bien vers l'uniformité quoique relativement lentement, ce qui s'explique par le taux d'acceptation très faible du mouvement de Pontage IC (de l'ordre 0.1%). Les points indiquent la distribution au bout de 5000 secondes de simulation et la courbe au bout de 50000 secondes.

Les courbes de performance (figures 4.23 et 4.24) montrent alors que le Pontage IC est bien un mouvement très performant au regard des critères adoptés. Par contre, de manière assez surprenante, la simulation AE a des performances ni pires ni meilleures que la simulation AR, alors que la courbe de fitness moyenne (figure 4.28) montre pourtant une progression assez nette. Cela révèle donc que l'amélioration des critères de performance mesurés sur une courte période (la durée d'une génération) n'est plus corrélée à l'amélioration de ces mêmes critères sur l'ensemble de la simulation.

Une explication de ce comportement pourrait être que ces critères ne sont pas les plus adaptés pour cet ensemble. En effet le mouvement de Pontage IC a pour effet de changer brutalement la localisation des centres de masses et les orientations des deux chaînes impliquées alors que seulement 3 monomères sont véritablement déplacés.

Remarque : Le code du Pontage IC a été adapté à partir d'un code écrit en FORTRAN (lequel a été utilisé pour les simulations présentées dans [50]) fourni par l'équipe du professeur Theodorou, de l'Université de Patras. Sa traduction n'a pas posé de problème en soi, mais il semble qu'à l'application, on observe une augmentation de l'énergie du système qui est proportionnelle au taux de ce mouvement (accompagné d'une augmentation du volume, visible sur la

figure 4.25, en comparaison de la figure 4.14), suggérant qu'il ne s'agit pas 'un effet de changement d'ensemble de simulation (passage de $nNPT$ à $nN\mu^*PT$) mais bel et bien d'un problème de biais qui est peut-être due à notre implantation dans notre code C, pour une raison qui nous demeure inconnue à ce jour. On peut toutefois retenir de cette simulation les conclusions concernant les fréquences des mouvements utilisés, tout en restant conscient que les échantillons ne sont pas valides statistiquement.

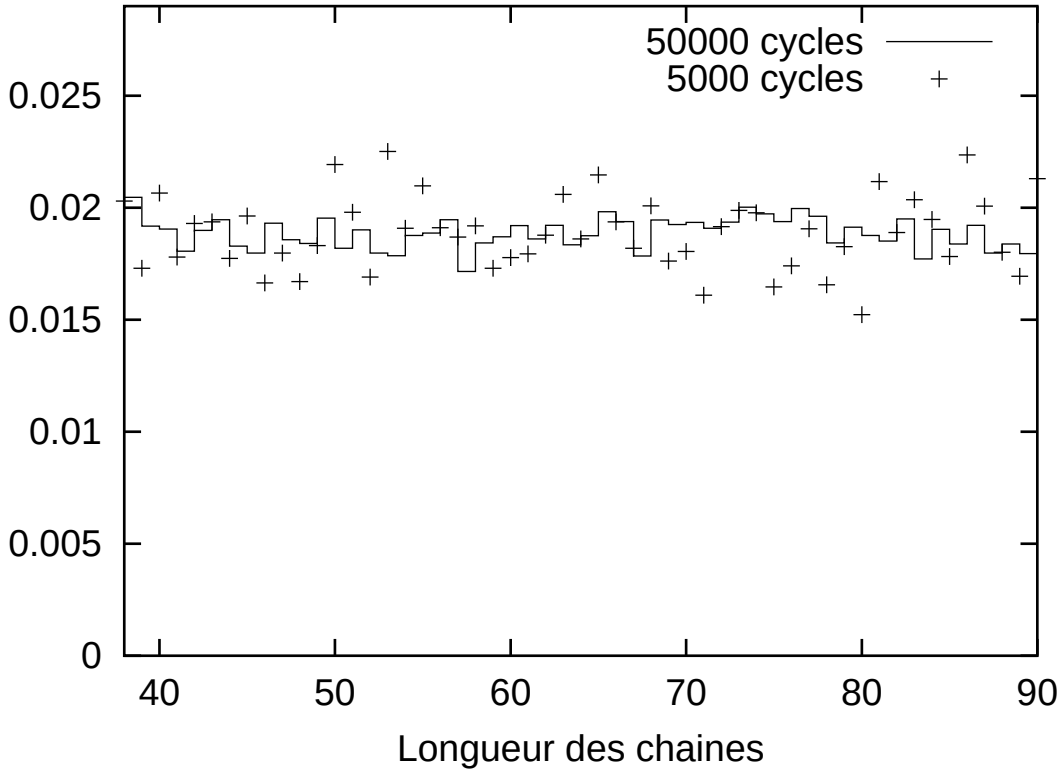


FIG. 4.22 – Série C : distribution estimée des longueurs de chaîne au bout de 5000 et 50000 secondes de simulation.

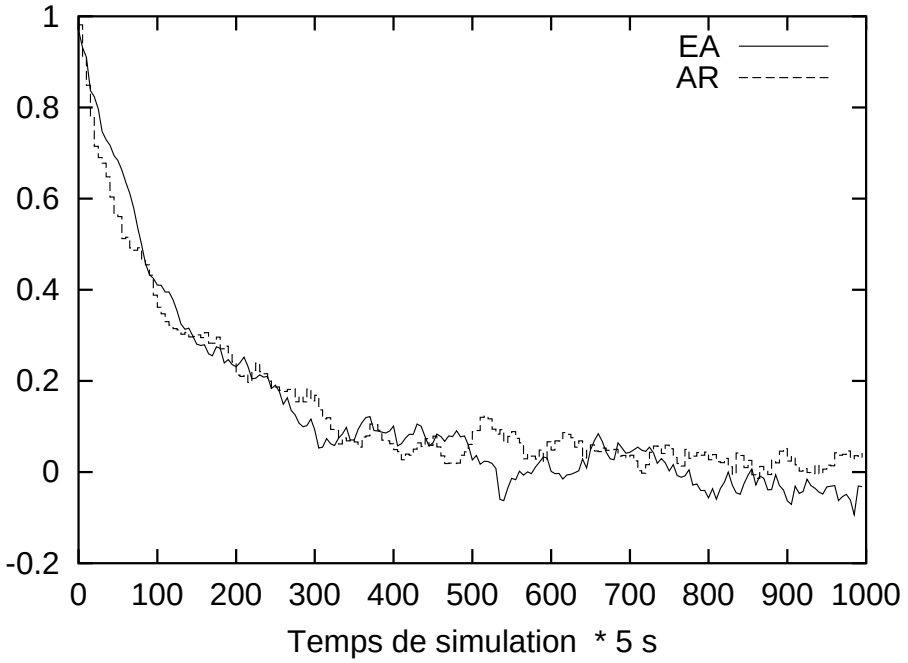


FIG. 4.23 – Série C : Courbe d'autocorrélation instantanée de vecteurs de chaînes.

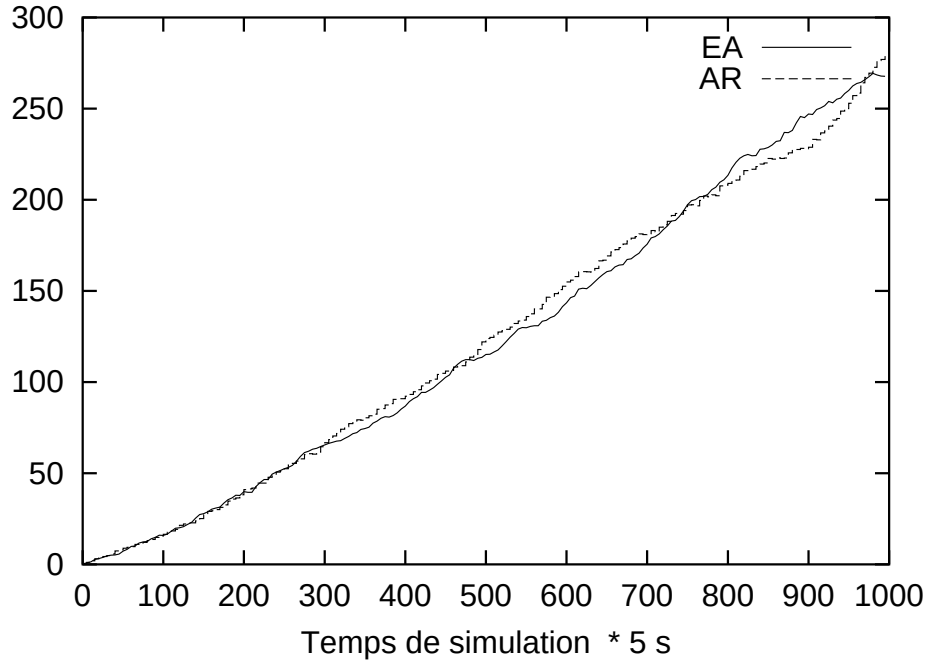


FIG. 4.24 – Série C : Courbe de déplacement carré moyen moyen du centre de masse en unités réduites (1 unité = σ_{LJ}^2).

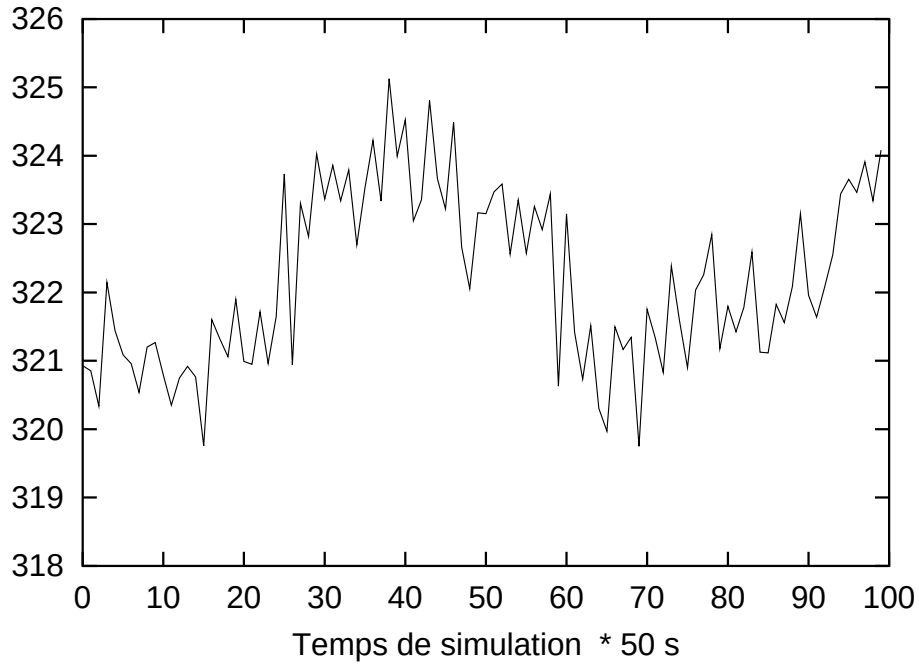


FIG. 4.25 – Série C : volume moyen des boîtes de simulation en unités réduites (1 unité $= \sigma_{LJ}^3$).

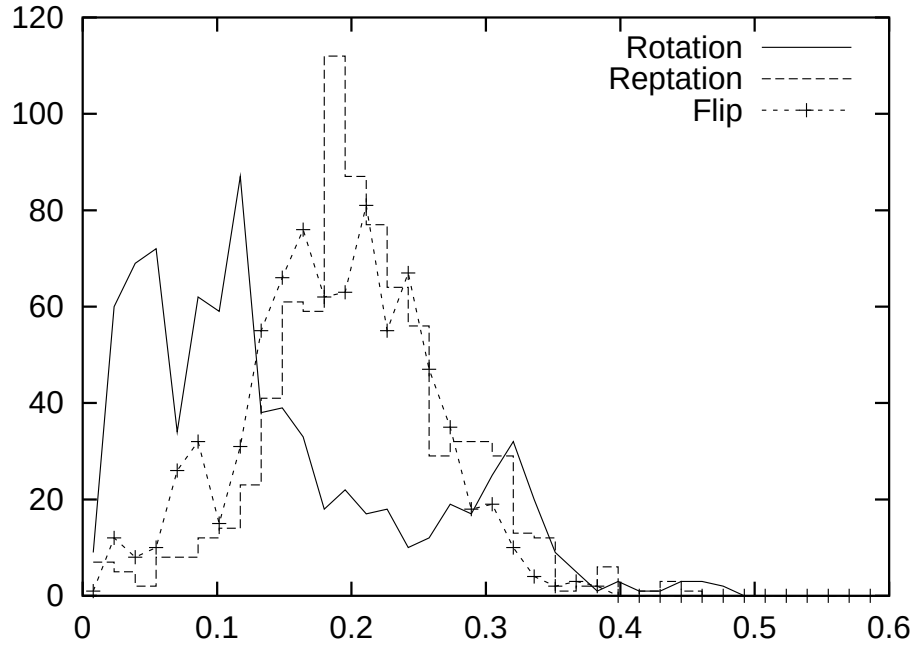


FIG. 4.26 – Série C : Histogrammes de FCE des mouvements Rotation, Reptation et Flip.

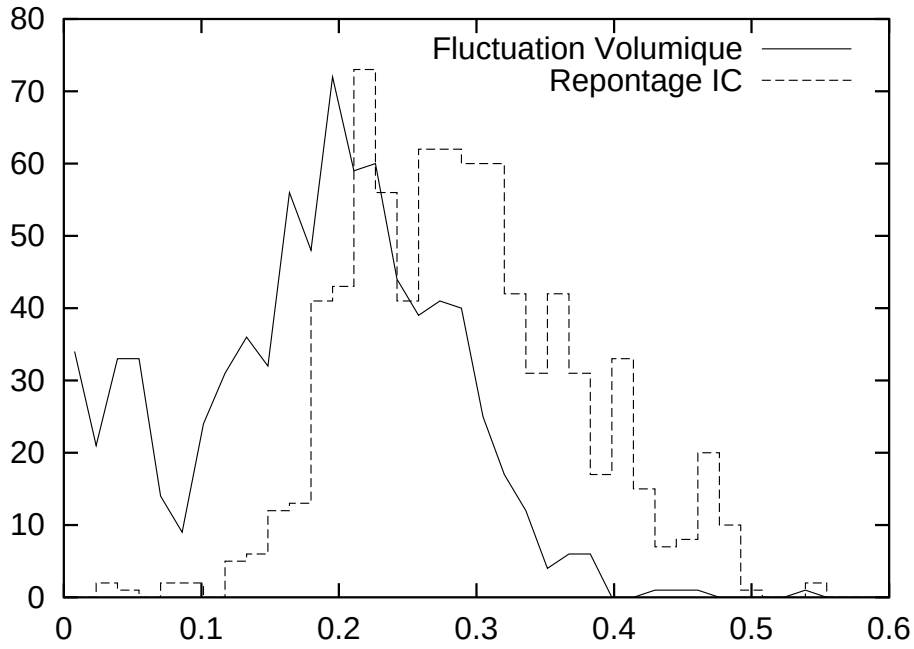


FIG. 4.27 – Série C : Histogrammes de FCE des mouvements Fluctuation Volumique et Pontage IC.

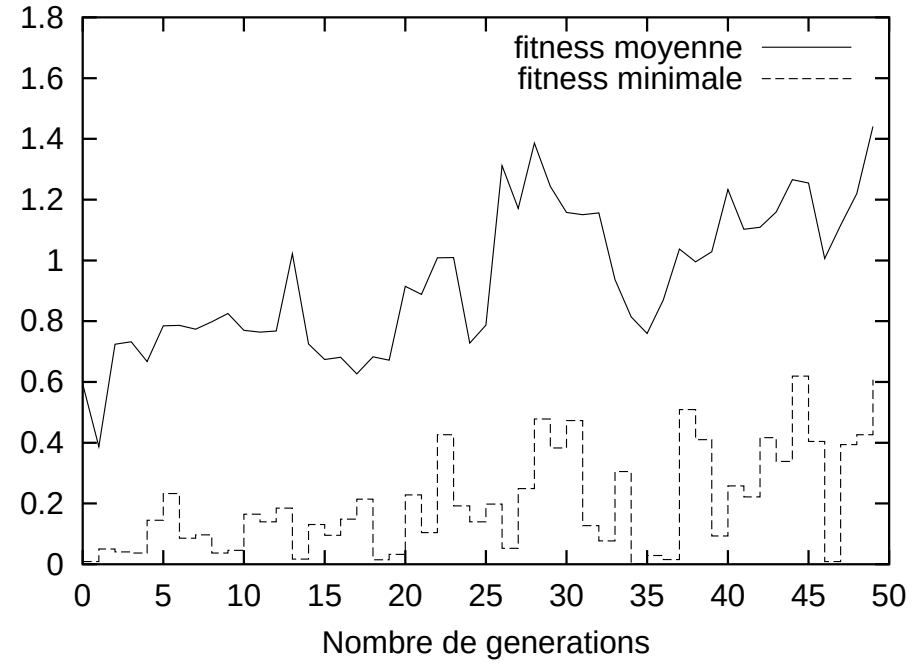


FIG. 4.28 – Série C : Courbes de la moyenne et du minimum des valeurs de fitness au cours des générations.

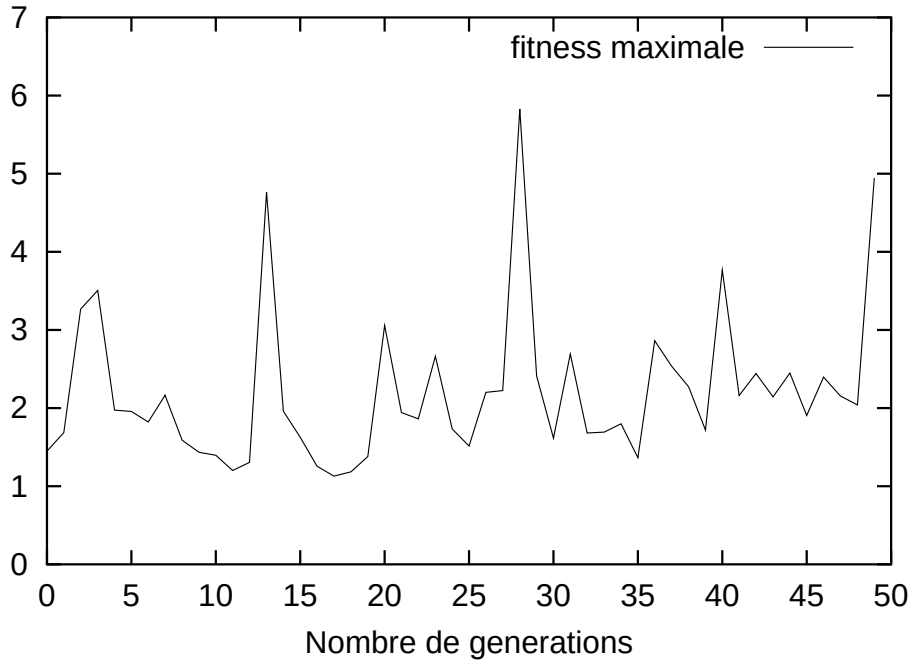


FIG. 4.29 – Série C : Courbe du maximum des valeurs de fitness au cours des générations.

4.2.5 Série D : utilité des fréquences à charge égale

Paramètres

Nous présentons ici une série de simulations réalisées précédemment aux séries A et B, mettant en jeu une comparaison entre l'utilisation de la fonction fitness instantanée et de la fonction de fitness pondérée.

L'ensemble statistique considéré est à $nNVT$ constants :

- $n = 1200$ monomères
- $N = 20$ chaînes. Les chaînes sont comptent alors 60 monomères
- $V = 4.45 \times 10^4$ Å^3.
- $T = 394.4K$
- $n_c = 5000$ cycles de simulations (1 cycle = 1seconde). Réalisés sur un bi-processeur Pentium II 350 MHz.

Remarquons d'abord que ces paramètres ne correspondent pas à un système stable physiquement car le volume caractéristique est alors $v = 1.59cm^3/g$, ce qui est largement supérieur à ce qu'il devrait être à pression nulle. En comparaison, une simulation ([50]) de polyéthylène avec des chaînes de taille 24, à $T = 450K$ et $P = 0$, donne un volume caractéristique $v = 1.42cm^3/g$, sachant que v diminue lorsque la taille moyenne des chaînes augmente et que la température décroit. Le but de cette simulation est simplement d'évaluer l'incidence de la pondération de la fonction de fitness sur un système simple où seulement les mouvements de Reptation, Rotation et Flip sont utilisés. Aucun de ces mouvements ne faisant intervenir de variation de volume, il est alors possible de les utiliser dans cet ensemble, où la densité de probabilité de trouver le système

dans une configuration r donnée est proportionnelle à :

$$\rho(r) = exp[-\beta U(r)] \tag{4.10}$$

La probabilité d'acceptation est alors identique à celle d'une simulation dans l'ensemble $nNPT$:

$$acc\left((r_o),(r_n)\right) = min\left\{1, exp\left(-\beta\left[U(r_n) - U(r_o)\right]\right)\right\} \tag{4.11}$$

Ajoutons que ces 3 mouvements n'impliquant le déplacement que d'un seul monomère, le FCE sont simplement égales aux FE.

Nous présentons donc 3 exécutions :

- AR avec des FE égales à $\left(\frac{1}{2}\right)$. Courbes référencées par "AR".
- AE sans utilisation de la pondération de fitness. Courbes référencées par "fitness instantanee".
- AE avec utilisation de la pondération de fitness. Courbes référencées par "fitness ponderee".

les paramètres d'EA sont ici :

- population de taille 16, soit 16 systèmes simulés en parallèle.
- remplacement complet de la population.
- découpage de la simulation en $n_g = 100$ générations, soit 50 cycles pour une génération.
- mutation gaussienne avec $p_m = 0.1$
- croisement uniforme avec $p_c = 0.5$
- rayon de voisinage pour la *fitness pondérée*. $\sigma_{max} = 0.05$

Résultats

On remarque sur sur les figures 4.30 et 4.31 que les deux exécutions AE sont plus performantes que l'exécution AR, et que parmi les exécutions AE, la pondération de fitness est plus efficace. Il est d'ailleurs intéressant de remarquer que les histogrammes (figures 4.33 et 4.34) des fréquences pour la fitness pondérée sont plus "larges", ce qui indique certes une moins grande stabilité, mais surtout une meilleure couverture de l'espace paramétrique S_g. Enfin on remarque que la pondération de la fonction a un effet direct sur la courbe de fitness moyenne de la population (figure 4.32).

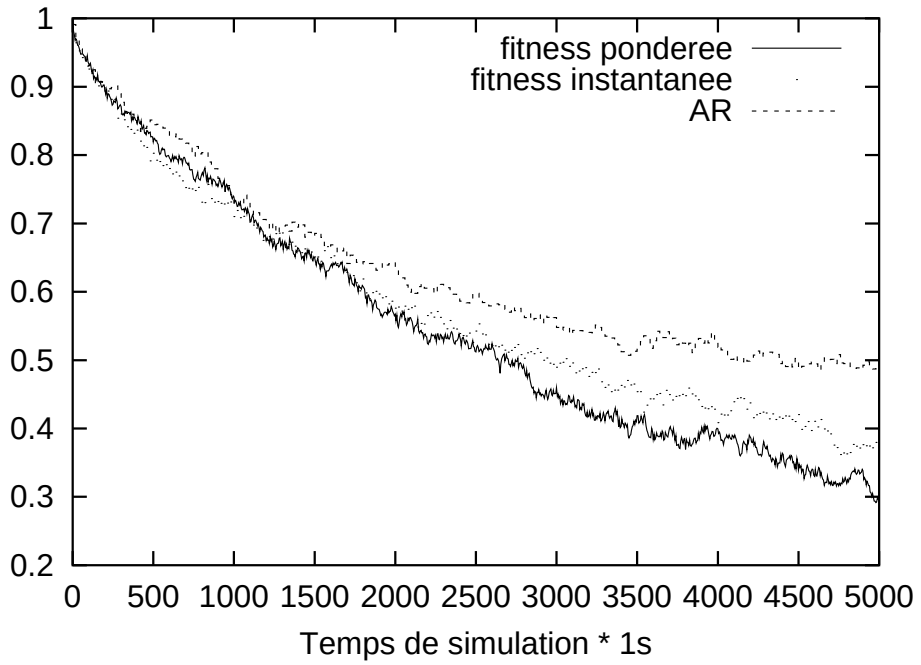


FIG. 4.30 – Série D : Courbe d'autocorrélation instantanée de vecteurs de chaînes.

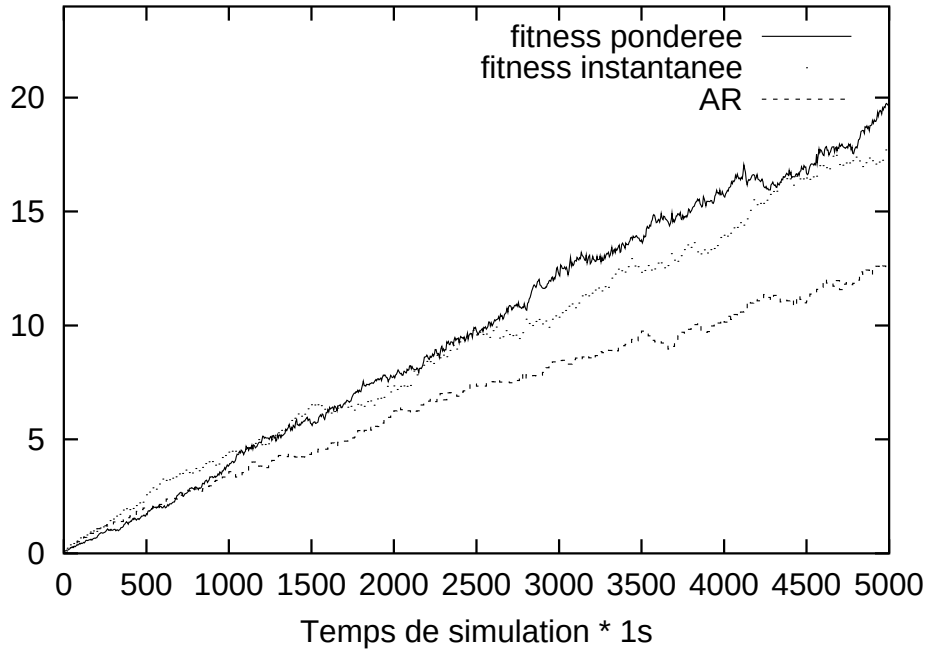


FIG. 4.31 – Série D : Courbe de déplacement carré moyen moyen du centre de masse en unités réduites (1 unité = σ_{LJ}^2).

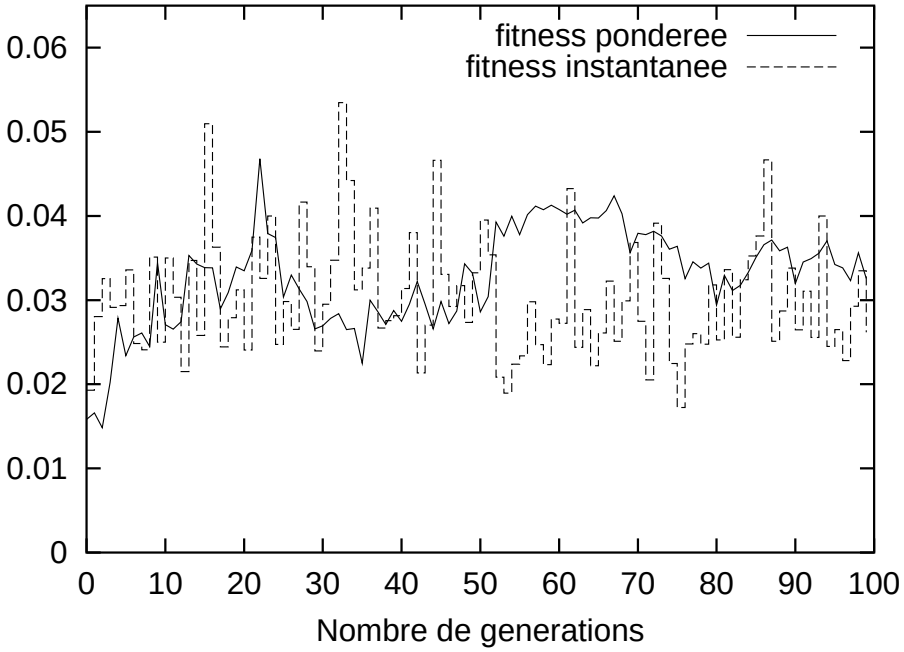


FIG. 4.32 – Série D : Courbe de fitness moyenne.

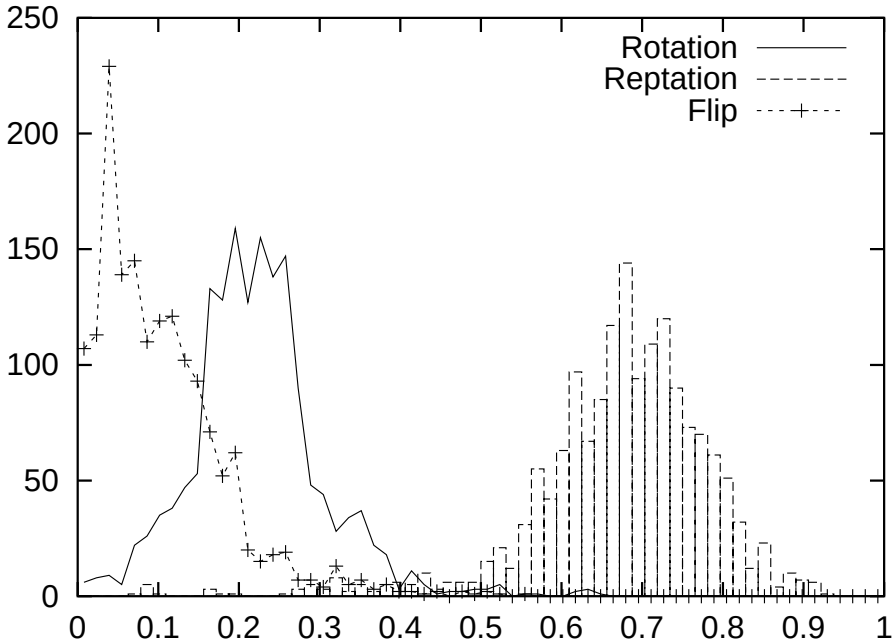


FIG. 4.33 – Série D : Histogrammes des fréquences de l'exécution AE sans pondération.

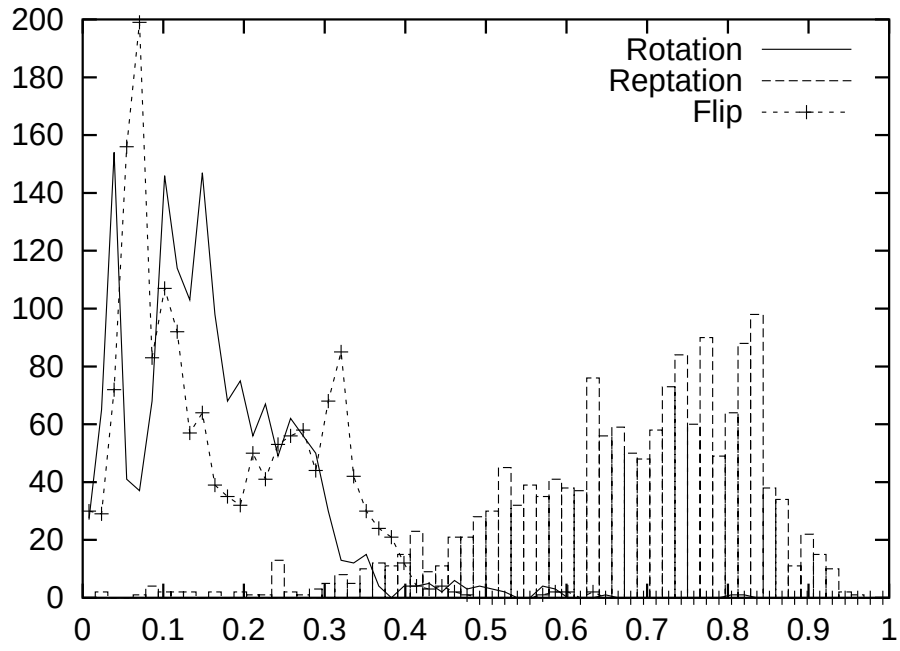


FIG. 4.34 – Série D : Histogrammes des fréquences de l'exécution AE avec pondération.

4.2.6 Série E : utilité de la pondération de fitness

Paramètres

Afin d'illustrer l'utilité du codage de FCE plutôt que de FE, nous présentons une série d'exécutions réalisées dans les mêmes conditions que la série D, mais en ajoutant le mouvement de translation, nécessitant alors le déplacement simultané des 60 monomères d'une chaîne. Ce qui implique un temps de calcul largement supérieur à ceux des autres mouvements. Pour les exécutions AR les FCE sont réglées à $\left(\frac{1}{4}\right)$ chacunes, ce qui correspond à des FE de $\left(\frac{60}{181}\right)$ pour la Rotation, le Reptation et le Flip et $\left(\frac{1}{181}\right)$ pour la Translation. Ce sont ces mêmes simulations qui sont présentées dans l'article soumis à CEC [44].

Résultats

On remarque sur les figures 4.35 et 4.36 que seul l'exécution AE sans pondération de fitness présente un avantage sur l'exécution AR. En effet si l'on regarde de plus près les histogrammes de FE (figures 4.38 et 4.39), on remarque que la pondération de fitness semble "ralentir l'élimination" du mouvement de Translation. En effet la fréquence de ce dernier n'étant pas codée à charge égale, même lorsqu'il lui est attribué une fréquence de l'ordre de 5%, cela correspond à une FCE de 75% soit un réglage très inefficace. Dès lors qu'il existe dans $\bar{\Pi}$ des individus pour lesquelles la Translation a une fréquence de 5%, tout nouvel individu pour lequel cette fréquence serait de 0.5% aurait certes une valeur de fitness instantanée bien meilleure, mais une valeur pondérée moins bonne du fait de la présence du "mauvais individu" à 5%. On comprend alors mieux l'intérêt du codage des FCE lorsqu'il y a une telle disparité de temps de calcul entre les mouvements utilisés.

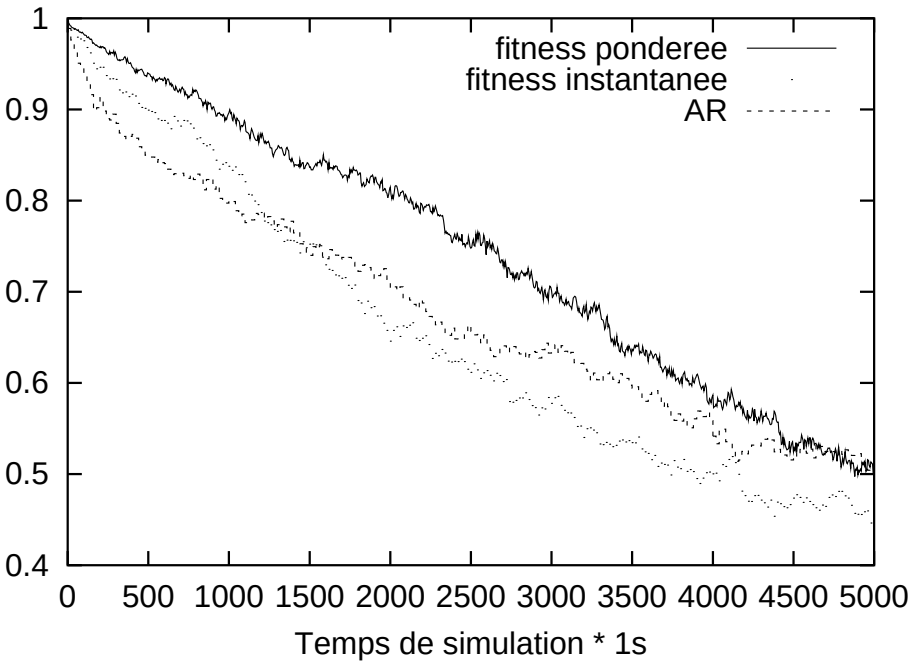


FIG. 4.35 – Série E : Courbe d'autocorrélation instantanée de vecteurs de chaînes.

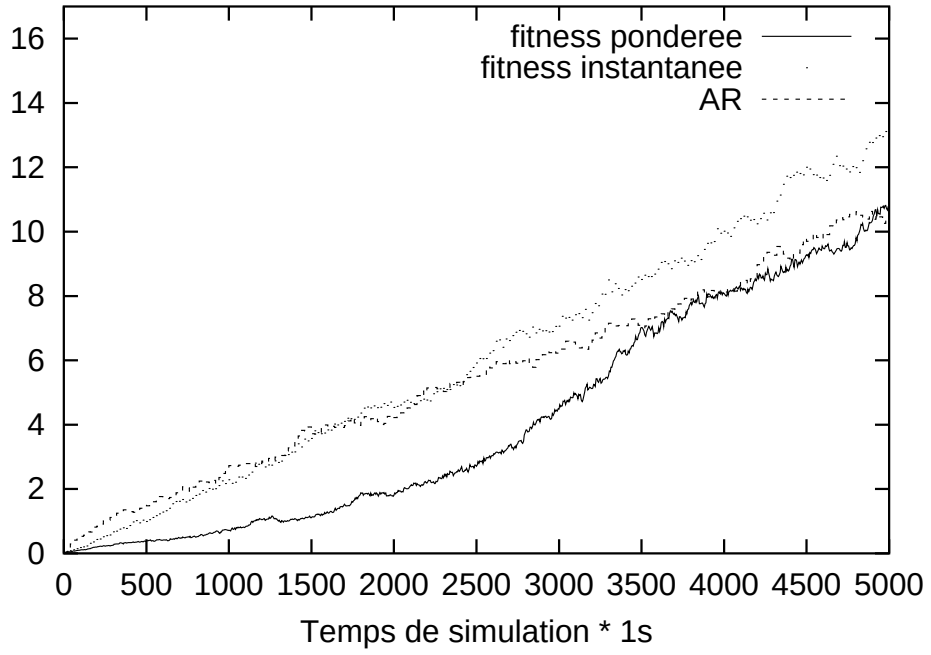


FIG. 4.36 – Série E : Courbe de déplacement carré moyen moyen du centre de masse en unités réduites (1 unité = σ_{LJ}^2).

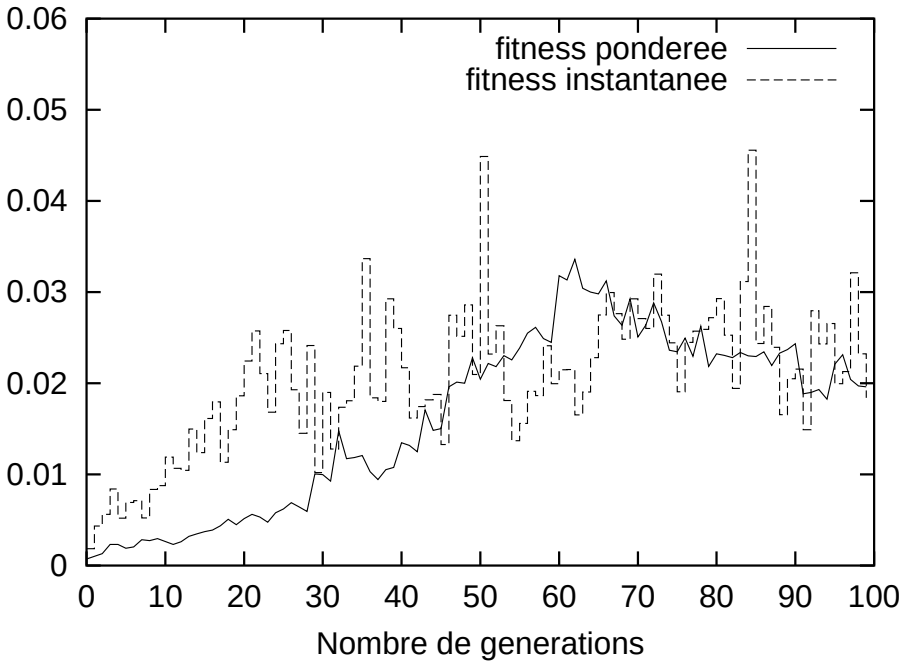


FIG. 4.37 – Série E : Courbe de fitness moyenne.

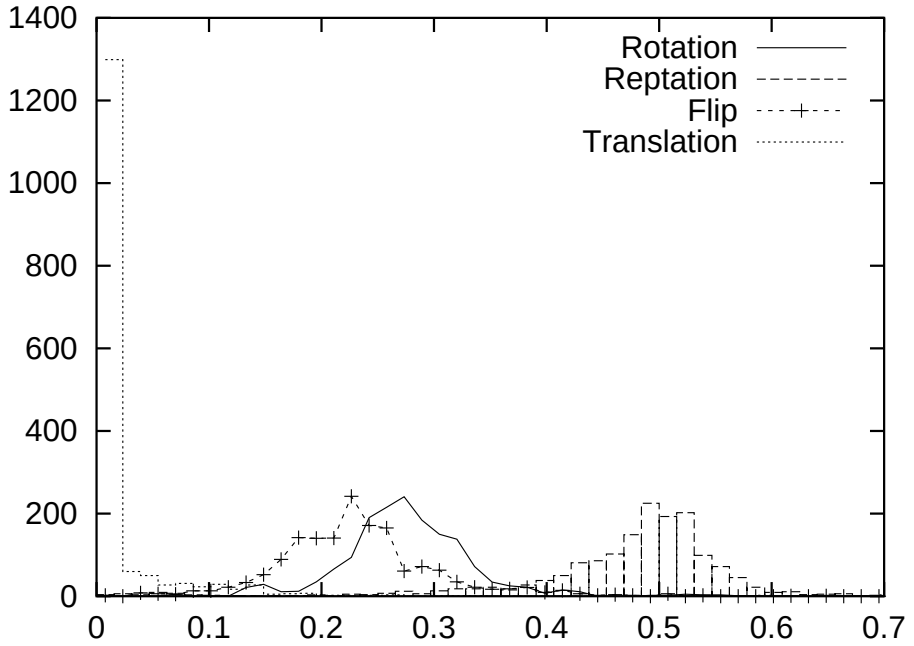


FIG. 4.38 – Série E : Histogrammes de FE de l'exécution AE sans pondération de fitness.

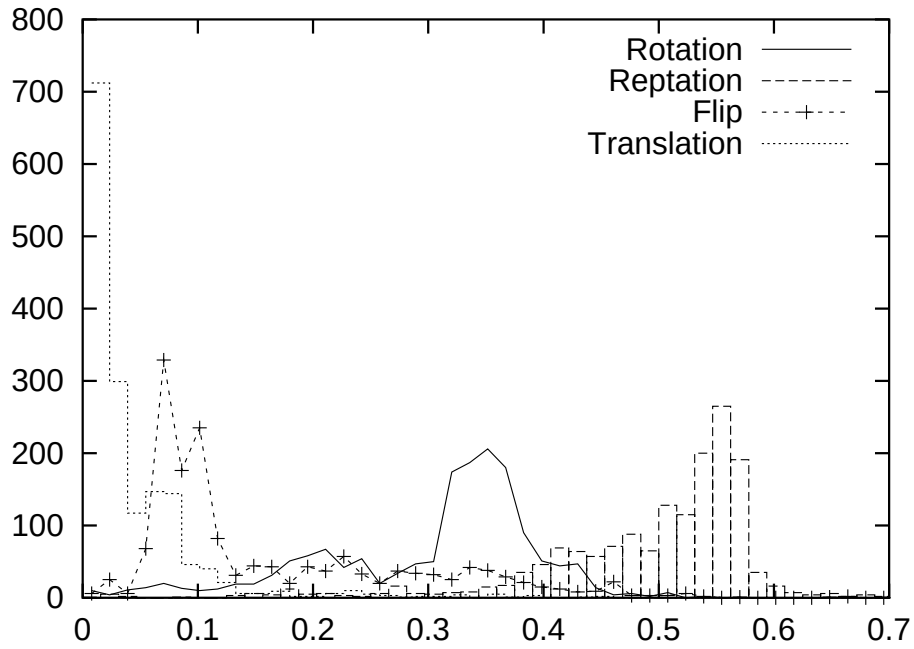


FIG. 4.39 – Série E : Histogrammes des FE de l'exécution AE avec pondération de fitness.

4.3 Discussion.

Dans ce chapitre, nous avons détaillé un algorithme d'évolution artificielle pour optimiser l'efficacité d'une simulation moléculaire de Monte Carlo appliquée aux polymères amorphes en état dense. Pour cela, nous avons commencé par identifier les degrés de liberté de cet algorithme qui conditionnent son efficacité : les fréquences relatives de différents mouvements possibles. Ensuite nous avons défini un critère numérique à optimiser faisant intervenir l'autocorrélation des orientations des chaînes et le déplacement de leurs centres de masse. Enfin nous avons précisé les conditions sous lesquelles ces mesures de performances peuvent être prises au cours de la simulation. Les différentes séries de simulations numériques ont alors montré que si on se réfère à une simulation effectuée en allouant à chaque mouvement une fréquence relative égale, l'algorithme évolutionnaire, en optimisant ces fréquences, permet alors d'améliorer ces critères de performance sur la durée totale de la simulation. De plus nous avons montré que tous les types de mouvements n'impliquant pas le même nombre de degrés de liberté (et donc n'ayant pas tous le même temps de calcul), il est plus judicieux de considérer les fréquences relatives à charge de calcul égale plutôt qu'à nombre de tentatives égal. Enfin, du fait de la nature aléatoire du processus pendant lesquels ils sont mesurés, les critères de performances doivent être considérés comme des variables aléatoires conditionnées par les paramètres du systèmes et particulièrement par les fréquences des mouvements. C'est pourquoi il nous a fallu considérer que la fonction de fitness est bruitée, ce qui a amené à définir une méthode de réduction de ce bruit fondée sur l'utilisation de l'histoire de toutes les évaluations. Ces différents ajustement étant faits, les simulations numériques montrent l'efficacité de la démarche.

Chapitre 5

Histoire et Immortalité en Evolution Artificielle.

Tirant parti des spécificités du problème décrit dans le chapitre 4, nous proposons un algorithme évolutionnaire en section 5.1 moins conventionnel mais plus adapté aux cas pathologiques où la fonction de fitness est très coûteuse et bruitée. En section 5.2, nous évaluons sa validité et sa robustesse sur quelques fonctions test présentant ces caractéristiques.

5.1 Individus immortels.

5.1.1 Motivations.

Dans le chapitre précédent, nous avons défini une approche évolutionnaire pour un problème d'optimisation où l'évaluation des performances est non seulement coûteuse en temps de calcul mais également fortement bruitée. Ayant bien précisé ces particularités d'application (section 4.1.4), nous avons alors proposé une méthode de réduction du bruit de la fonction de fitness en utilisant l'histoire des populations (section 4.1.5). Allant un peu plus loin dans l'usage des populations passées, Corne et al [12] et Zitzler et al [67] ont proposé d'utiliser l'histoire des évaluations dans le cadre de l'optimisation multi-critères. Leur démarche est alors de conserver tous les individus non-dominés (front de Pareto, voir section 2.6.4) dans une "archive", et d'appliquer la sélection sur l'ensemble de cette archive afin de disposer d'une plus grande diversité. Nous proposons maintenant, de manière analogue à cette technique, d'étendre l'utilisation de notre historique d'évaluations H, et d'autoriser l'algorithme à y sélectionner directement des individus pour la reproduction. En d'autres termes un individu, une fois créé, peut à tout moment se reproduire : l'algorithme fait alors évoluer une population d'individus immortels. Ainsi l'ensemble de l'information produite n'est plus simplement utilisée pour obtenir des évaluations plus précises, mais sert aussi à augmenter la diversité génétique lors de la phase de sélection. Une telle stratégie implique alors la gestion d'un ensemble important d'informations ainsi que la mise en oeuvre d'algorithmes pour leur manipulation. Cela implique alors une charge de calcul plus importante que la gestion d'une population de taille fixe. Mais il faut considérer que dans notre cas le nombre d'évaluations de fitness reste limité et que la durée de ces évaluations est tellement importante que la surcharge de calcul de l'EA reste négligeable en regard de la charge globale.

5.1.2 EAII : Evolution Artificielle d'Individus Immortels.

Ayant défini la notion d'individus immortels et le cadre dans lequel il peut s'appliquer, nous détaillons maintenant un algorithme que nous appellerons algorithme d'Evolution Artificielle d'Individus Immortels (**EAII**). Le principe est d'utiliser une population croissante, soit l'histoire H des évaluations. Nous proposons alors un modèle d'algorithme suivant une architecture client/serveur adapté à une simulation distribuée : un serveur génétique fournit des individus à des clients qui sont chargés de les évaluer et de retourner les résultats. Le fonctionnement (illustré par la figure 5.1) en est le suivant :

- Une liste d'enfants est initialement créée de manière aléatoire.
- Dès qu'un client le demande, le serveur fournit un enfant de sa liste. Le client calcule sa valeur de fitness et la retourne au serveur qui ajoute l'individu évalué à H.
- Lorsque la liste des enfants est vide, le serveur en crée de nouveaux en sélectionnant des parents dans H et en appliquant des opérateurs génétiques.
- Lorsque le serveur sélectionne les individus dans H, on impose qu'un nombre minimum d'individus y soient déjà présents (nous l'appellerons alors min_{par}). A défaut, les nouveaux enfants sont encore créés de manières aléatoire.
- L'algorithme est arrêté lorsque qu'un nombre maximum d'évaluations est atteint (paramètre dépendant du problème).

On peut alors appliquer une sélection par tournoi, en choisissant un nombre n_t prédéfinis d'individus de H avec un tirage aléatoire sans biais, puis retenir celui ayant la meilleure valeur de fitness pondérée (f_H).

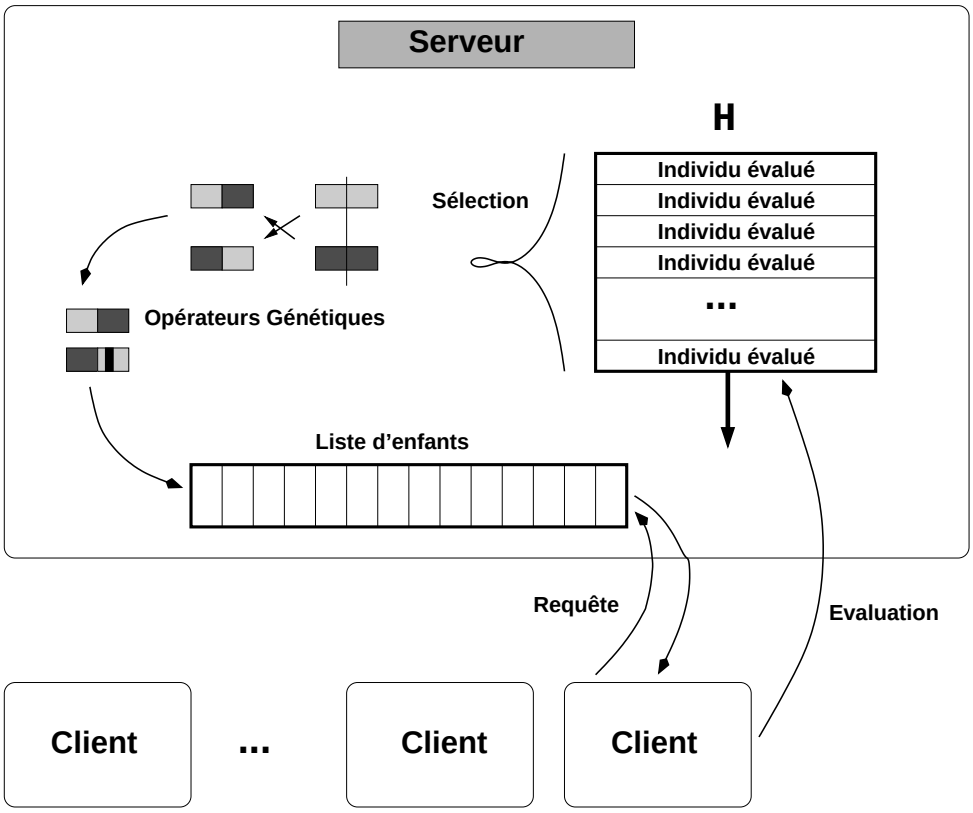


FIG. 5.1 – Représentation schématique de l'EAII.

Remarquons que cette stratégie se prête bien à un modèle de calcul distribué car il n'implique pas de synchroniser les évaluations des individus pour effectuer la sélection de parents. Cet aspect autorise un nombre de clients quelconques, disposant éventuellement de capacités de calcul hétérogènes. Pour notre application à des simulations moléculaires distribuées le nombre de clients ne peut toutefois excéder le nombre de systèmes simulés[1], par contre il peut être inférieur.

Pour résumer, on peut décrire le fonctionnement du serveur par le pseudo-code suivant :

- création aléatoire de n_e individus dans la liste d'enfants L_e
- H = liste vide
- tant que (nombre d'évaluations faites $<$ nombre d'évaluations à faire)
  - si requête client
    - si L_e vide
      - si $n_H \geq min_{par}$
        - création de n_e individus par sélection (avec f_H), croisement et mutation.
      - sinon
        - création aléatoire de n_e individus.
      - ajout des n_e nouveaux individus dans L_e.
    - extraire un individu μ de L_e, pour évaluation.
  - si retour client : couple (individu, valeur de fitness) $(\mu, f(\mu, \omega))$
    - ajout de $(\mu, f(\mu, \omega))$ dans H
- fin

Partage.

Nous avons vu en section 2.6.3 que des méthodes de partage des ressources avaient été introduites en EA pour à la fois éviter la perte de diversité génétique et la découverte de plusieurs optima pour des problèmes multimodaux. Nous en proposons maintenant une forme fondée sur la pondération de la fonction de fitness : à chaque fois que l'on calcule $f_H(\mu)$, on calcule aussi la quantité suivante :

$$W_H(\mu) = \sum_{\nu \in H \cap B_{\sigma_\infty}(\mu)} (w(\mu, \nu) \times n_\nu) \tag{5.1}$$

$W_H(\mu)$ représentant alors la somme des "poids" des voisins de μ peut être vu comme un décompte "pondéré" du nombre de voisins, les plus proches ayant plus de poids. On peut alors utiliser cette quantité pour effectuer un partage de la fonction de fitness de la manière suivante :

$$\forall \mu \in S_\mu, \quad g_H(\mu) = f_H(\mu) \times \left(1 + \frac{1}{W_H(\mu)}\right) \tag{5.2}$$

On a bien :

$$\forall \mu \in H, \quad W_H(\mu) \geq 1 \tag{5.3}$$

L'effet produit est alors qu'un individu isolé (sans voisin proche d'une distance inférieure à σ_∞) aura un poids $W_H(\mu) = 1$ impliquant $g_H(\mu) = 2 * f_H(\mu)$, augmentant donc ses chances d'être sélectionné. A l'inverse un individu dont le nombre de voisins proches sera élevé ne bénéficiera pas de cet effet car dans ce cas on aura $g_H(\mu) \simeq f_H(\mu)$.

[1] Tous les mouvements de Monte Carlo n'offrant pas tous de possibilités de parallélisation, il n'est pas possible de distribuer la simulation d'un système sur plusieurs clients.

Sélection par seuillage

Nous proposons maintenant une autre méthode de sélection pour l'EAII, en partant du principe que l'on souhaite à la fois obtenir des individus qui aient de bonnes valeurs de fitness et qui soient bien répartis sur les différents optima locaux. Le principe en est le suivant :

- Un seuil de fitness est calculé en considérant la valeur de fitness du meilleur individu ($f_H^{max} = max_{x \in H}\{f_H(x)\}$). Ce seuil est alors simplement le produit de f_H^{max} par un coefficient de seuil c_s ($c_s \in [0,1]$) :
$$f_H^s = f_H^{max} * c_s \tag{5.4}$$
- Des individus sont tirés aléatoirement uniformément dans H jusqu'à obtenir un individu y tel que $f_H(y) > f_H^s$ qui devient alors l'individu sélectionné.
- Si au bout d'un nombre n_t de tirages donné, on ne trouve pas d'individus satisfaisant, le meilleur d'entre eux est choisi. Dans ce cas, on retrouve le cas d'un sélection par tournoi avec n_t participants.

5.2 Simulation numériques : fonctions de test

Nous commencerons donc par tester cet algorithme pour optimiser des fonctions de test auxquelles un bruit gaussien est ajouté.

5.2.1 Choix algorithmiques et paramètres de simulations

Nous testerons les algorithmes évolutionnaires (AE) suivants :

- Un Algorithme Génétique (noté **AG**) sans utilisation de H.
- Un AG utilisant la pondération de fitness (noté **AGP**).
- Un EAII avec la sélection par tournoi simple, noté **EAII + tournoi**
- Un EAII avec le partage de fitness et la sélection par tournoi simple **EAII + Partage**
- Un EAII avec la sélection par seuillage, noté **EAII + seuil**

Les espaces de recherche sont de la forme :

$$S = \prod_{i=1}^{n} [a_i, b_i] \ , \ \forall i \in \{1, ...n\} \quad (a_i, b_i) \in \mathbb{R}^2 \ , \ \mathrm{a_i} < \mathrm{b_i} \tag{5.5}$$

Un codage réel est utilisé et la distance euclidienne retenue comme métrique. La méthode de sélection pour les AG est la sélection universelle stochastique (**SUS**, voir la section 2.5.3) avec remplacement complet de la population. La **mutation gaussienne** est utilisée avec un paramètre de variance fixé pour chaque composante $\sigma_i = 0.1 \times (b_i - a_i)$.

Concernant le croisement nous testerons 3 configurations :

- Croisement arithmétique, défini en section 2.5.5.
- Croisement arithmétique avec restriction de voisinage : seuls les individus proches, c'est-à-dire d'une distance inférieure à un certain seuil, sont autorisés à se croiser (nous l'appellerons ultérieurement croisement restreint). Pratiquement, si la dimension de l'espace de recherche est n, ce seuil sera fixé à $(0.1 \times \sqrt{n})$. L'intérêt de cette technique est de limiter la portée du croisement, lui conférant une rôle de recherche locale, si toutefois celui-ci ne présente pas la possibilité de créer des enfants plus distants de leur deux parents que la distance entre leurs deux parents. Avec le croisement arithmétique et la distance euclidienne, on vérifie bien que les enfants reste proche de leur parents.

- Pas de croisement.

Pour chaque fonction de test, les paramètres de simulation seront les suivants :

- Chaque simulation est répétée **25 fois**, les résultats présentés en sont alors les moyennes.
- Un nombre maximum de **3200** évaluations est fixé. Pour les AG cela revient à effectuer **100 générations** avec des populations de taille **32**.
- Les valeurs de fitness non bruitées sont gardées "hors-ligne", c'est-à-dire qu'elles servent uniquement à mesurer l'écart entre les valeurs de fitness effectivement attribuée lors de l'EA (donc bruitées) et les véritables valeurs correspondantes.
- Dans un but de comparaison, les mesures prises sur les populations des AG, seront également prises dans le cas de l'EAII en regroupant les individus par groupe de 32 au cours de l'algorithme.
- Paramètres de l'EAII :
  - Sélection par tournoi simple avec une taille $n_t = 4$
  - Sélection par seuillage avec : $c_s = 0.8$, $n_t = 10$.
  - Nombre minimal de parents requis avant la sélection $min_{par} = 32$.
- Probabilités des opérateurs génétiques, plusieurs valeurs sont testées :
  - **probabilités de croisement** $p_c = \{0.2, 0.5, 0.8\}$
  - **probabilités de mutation** $p_m = \{0, 0.01, 0.1\}$
  - En absence de croisement $p_m = \{0.02, 0.05, 0.1, 0.2\}$

Nous ne présenterons pas toutes les courbes correspondant à toutes les combinaisons de probabilités de croisement de mutations possibles, mais seulement celles qui auront donné les meilleurs résultats pour chaque type d'algorithme.

5.2.2 F_1 : une fonction multimodale simple

Définition

Considérons la fonction suivante :

$$\begin{array}{rcl} F_1 : [0,1]^3 : & \rightarrow & \mathbb{R}^+ \\ (x_0, x_1, x_2) & \rightarrow & \left(\sum_{i=0}^{2} t(x_i)\right) \end{array} \tag{5.6}$$

Un bruit gaussien est ajouté pour obtenir la fonction de fitness perçue par l'AE :

$$f_1 = F_1 \times (1 + N_{(0,0.5)}) \tag{5.7}$$

où $N_{(0,0.5)}$ est un tirage aléatoire gaussien renouvelé à chaque évaluation.

En définissant t (figure 5.2, gauche) comme :

$$\begin{array}{rcl} [0,1] : & \rightarrow & [0,1] \\ x & \rightarrow & \begin{cases} (4 * x) & \text{si} \quad x \in [0, 0.25[\\ (2 - 4 * x) & \text{si} \quad x \in [0.25, 0.5[\\ (4 * x - 2) & \text{si} \quad x \in [0.5, 0.75[\\ (4 - 4 * x) & \text{si} \quad x \in [0.75, 1] \end{cases} \end{array} \tag{5.8}$$

Comme t présente deux maxima (en $\frac{1}{4}$ et $\frac{3}{4}$) de même amplitude, F_1 en présente alors 8 de la même amplitude. Afin de mesurer l'aptitude des algorithmes à localiser plusieurs optima, nous donnerons le décompte des individus proches de ces optima. Précisément, pour chaque optima

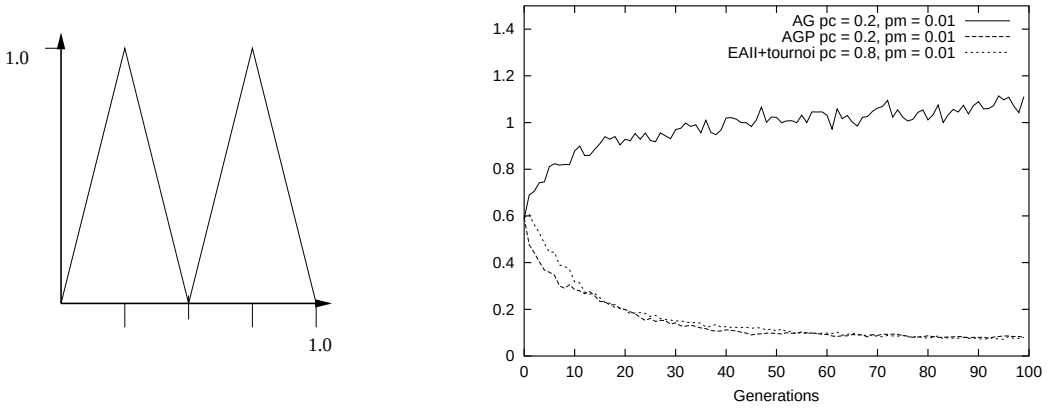


FIG. 5.2 – Gauche : fonction t. Droite : bruit moyen constaté lors des évaluations de f_1 et f_H (sur 25 runs).

$(o_{i,i\in\{0,\ldots,7\}})$, on compte le nombre d'individus vérifiant $d_\infty(o_i, x) < 0.1$, obtenant une liste de mesures de voisinage des optima $(no_{i,i\in\{0,\ldots,7\}})$. En triant cette liste en ordre décroissant, il est alors possible de faire une moyenne (sur les 25 runs) conservant l'information de répartition. Enfin, F_1 étant clairement une fonction régulière, on choisit $\sigma_\infty = 0.05$.

Résultats

La figure 5.2 (droite) montre le bruit moyen des fonctions de fitness, soit pour l'AG :

$$\sum_{x\in population} |f_1(x) - F_1(x)|$$

et pour les autres (utilisant la pondération) :

$$\sum_{x\in population} |f_H(x) - F_1(x)|$$

Rappelons que pour l'EAII, le terme population désigne simplement le regroupement arbitraire des nouveaux individus dans un but de comparaison (il n'a aucune conséquence sur la dynamique de l'algorithme). On voit alors qu'en l'absence de pondération, le bruit d'évaluation augmente au cours de générations à cause de la nature multiplicative du bruit. En revanche l'utilisation de la pondération diminue significativement le bruit moyen.

Les figures 5.3, 5.5 et 5.7 montrent les performances en termes de Moyenne de Fitness Débruitée (ultérieurement dénommée **MFD**), correspondant aux valeurs moyennes de F_1 sur les individus de la population. La capacité à localiser plusieurs optima est illustrée dans les figures 5.4, 5.6 et 5.8 avec les décomptes de voisinages ordonnés $(no_{i,i\in\{0,\ldots,7\}})$.

On remarque qu'avec une faible probabilité de croisement arithmétique ($p_c = 0.2$), l'AG obtient les meilleures performances de MFD, tout en ayant tendance à se concentrer sur un seul maximum. Les effets de la seule pondération apportent peu de changement. L'EAII obtient une moins bonne MFD avec le croisement arithmétique, mais lorsqu'il est utilisé avec le croisement

restreint et la sélection par seuillage il est rapidement à la fois efficace en termes de MFD et découvre tous les optima (même si tous ne sont pas exploités). Il est d'ailleurs intéressant de remarquer que pour ce comportement le partage, sensé favoriser la diversité, est moins efficace.

En conclusion, on peut voir que sur cette fonction un AG obtient rapidement de bonnes performances en termes de fitness moyenne de la population, et que l'EAII avec l'utilisation du croisement restreint et de la sélection se comporte aussi bien mais découvre plus d'optima.

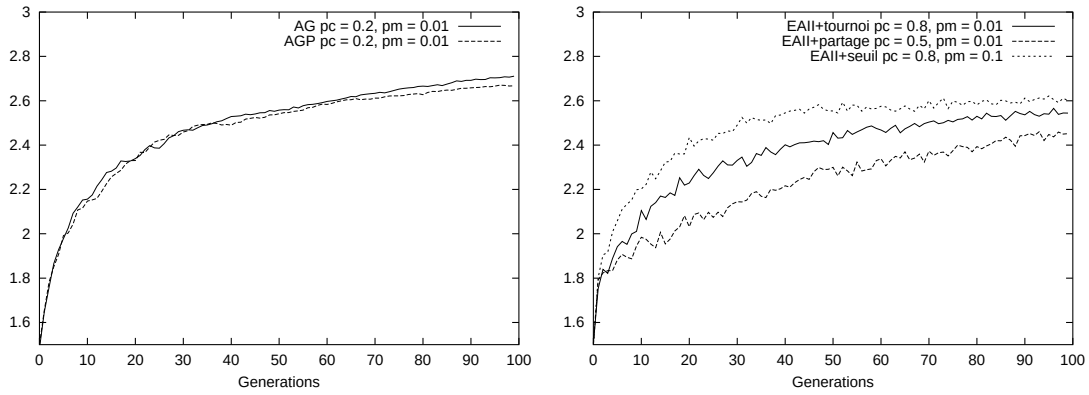


FIG. 5.3 – F_1 : Croisement arithmétique : moyenne sur la population de la fonction de fitness débruitée.

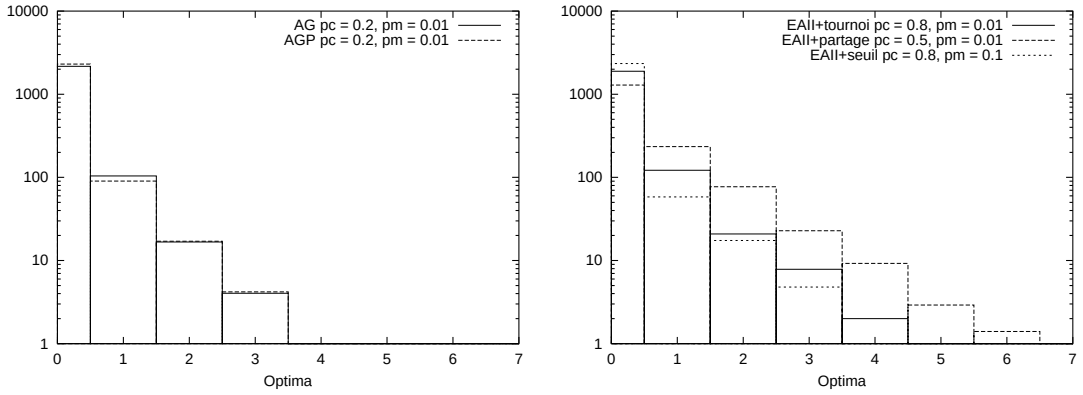


FIG. 5.4 – F_1 : Croisement arithmétique : Décompte des voisinages d'optima.

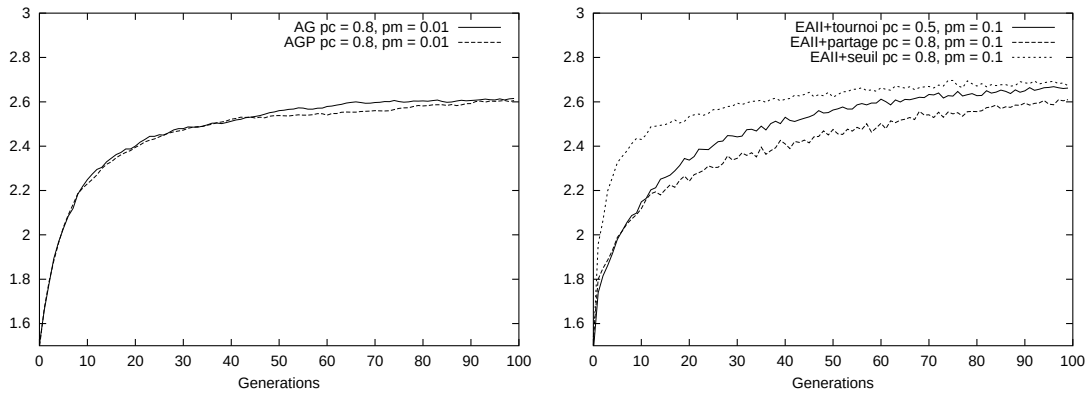


FIG. 5.5 – F_1 : Croisement restreint : moyenne sur la population de la fonction de fitness débruitée.

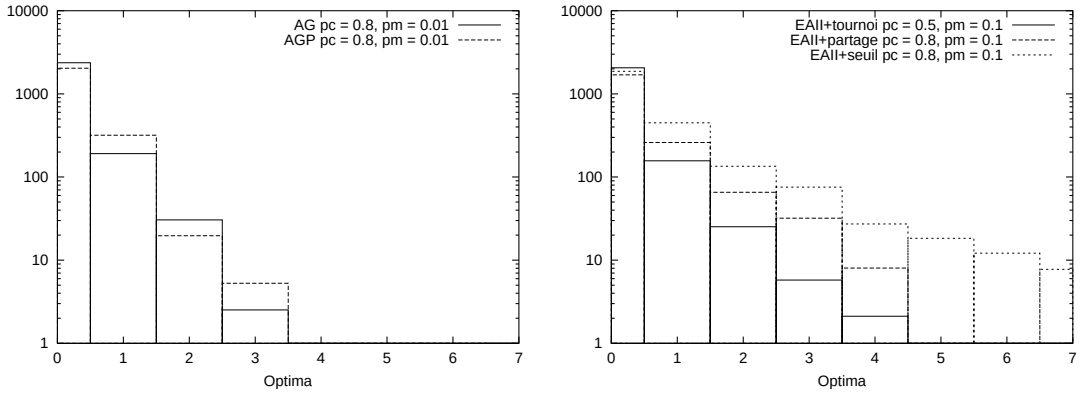


FIG. 5.6 – F_1 : Croisement restreint : Décompte des voisinages d'optima.

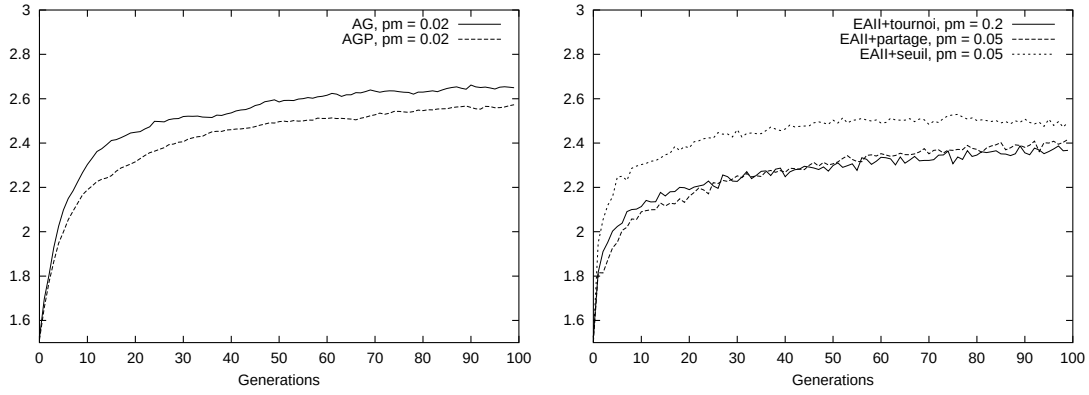


FIG. 5.7 – F_1 : Sans croisement : moyenne sur la population de la fonction de fitness débruitée.

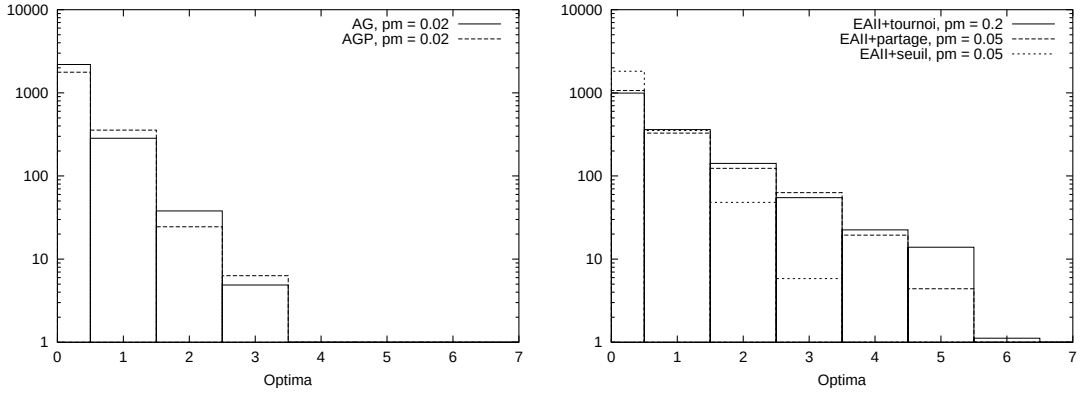


FIG. 5.8 – F_1 : Sans croisement : Décompte des voisinages d'optima.

5.2.3 F_2 : une version épistatique de F_1.

Définition

Considérons la fonction suivante :

$$\begin{array}{rcl} F_2 : [0,1]^3 : & \rightarrow & \mathbb{R}^+ \\ (x_0, x_1, x_2) & \rightarrow & t_2(x_0, x_1) + t_2(x_1, x_2) + t_2(x_2, x_0) \end{array} \tag{5.9}$$

avec t_2 défini à l'aide de la fonction t défini en section 5.2.2 :

$$t_2(x,y) = \begin{cases} t(x) & \text{si} \quad (x-0.5) \times (y-0.5) > 0 \\ 0.5 \times t(x) & \text{si} \quad (x-0.5) \times (y-0.5) < 0 \end{cases} \tag{5.10}$$

La fonction de fitness bruitée est alors définie comme :

$$f_2 = F_2 \times (1 + N_{(0,0.5)}) \tag{5.11}$$

où $N_{(0,0.5)}$ est un tirage aléatoire gaussien renouvelé à chaque évaluation.
Comme F_2 est aussi une fonction localement régulière, on fixe $\sigma_\infty = 0.05$. En comparaison de F_1, F_2 présente aussi 8 maxima locaux. La différence vient du caractère "épistatique" de t_2, autrement dit de la forte interdépendance entre les variables. En fait il y a deux maxima principaux aux points $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ et $(\frac{3}{4}, \frac{3}{4}, \frac{3}{4})$ avec une valeur de 3, et 6 maxima locaux avec une valeur de 2 pour toutes les autres combinaisons de $\frac{1}{4}$ et $\frac{3}{4}$. On voit, de plus, que les deux maxima globaux sont éloignés et représentent ainsi deux pôles d'attraction opposés. On observera alors si les algorithmes parviennent à trouver au moins un des maxima globaux, ou éventuellement les deux simultanément. La liste des décomptes de voisinages d'optima sera alors ordonnée de manière différente : $(no_{i,i\in\{0,1\}})$ représenteront alors la liste ordonnée de maxima principaux et $(no_{i,i\in\{2,..,7\}})$ la liste ordonnée des maxima secondaires.

Résultats

Les figures 5.9 à 5.14 montrent encore une fois qu'un AG simple est capable de localiser un optimum global et de l'exploiter mais ne peut aisément localiser le deuxième, il aura même plus tendance à visiter un optimum secondaire (cet effet étant plus marqué avec l'utilisation du croisement restreint). En revanche on constate que l'EAII, spécialement avec l'application de la

restriction de voisinage et la sélection par seuillage, exploite plus efficacement les deux optima principaux, l'optimum principal le moins exploité étant tout de même plus exploité que tous les optima secondaires. Ainsi sur ce problème, cette configuration offre de bonnes performances à la fois en capacités d'exploration et en capacité d'exploitation. Enfin, là encore, on voit que l'absence de croisement se révèle utile en termes de diversité, mais la progression de la FDM est la plus lente de toutes les configurations.

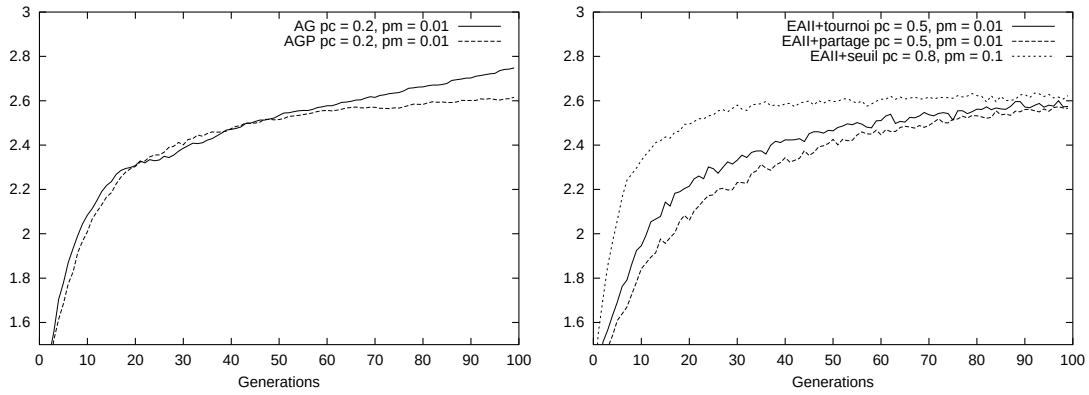


FIG. 5.9 – F_2 : Croisement arithmétique : moyenne sur la population de la fonction de fitness débruitée.

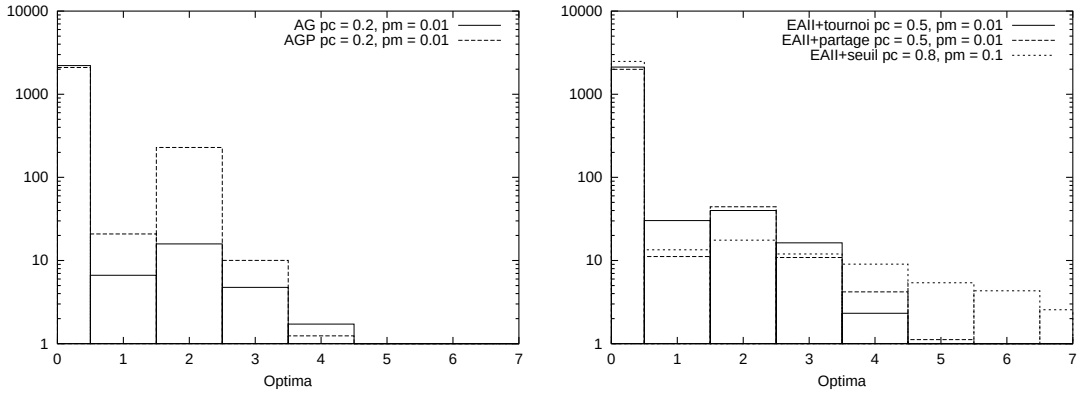


FIG. 5.10 – F_2 : Croisement arithmétique : Décompte des voisinages d'optima.

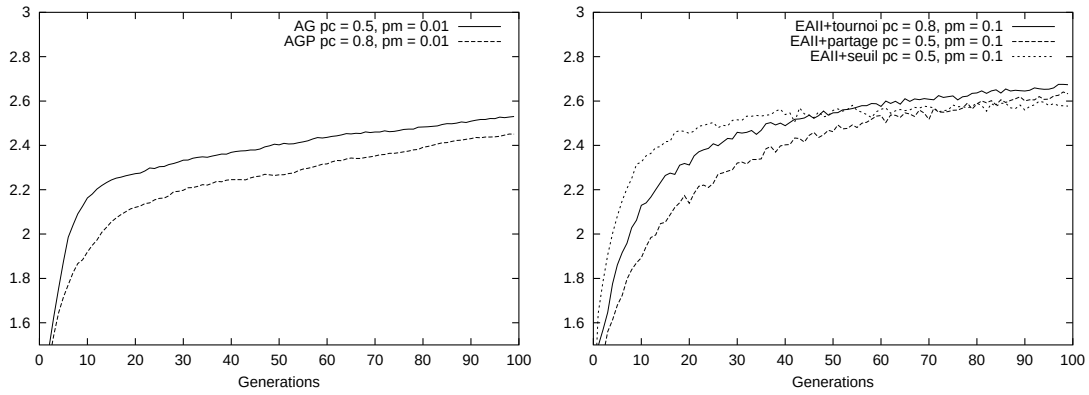


FIG. 5.11 – F_2 : Croisement restreint : moyenne sur la population de la fonction de fitness débruitée.

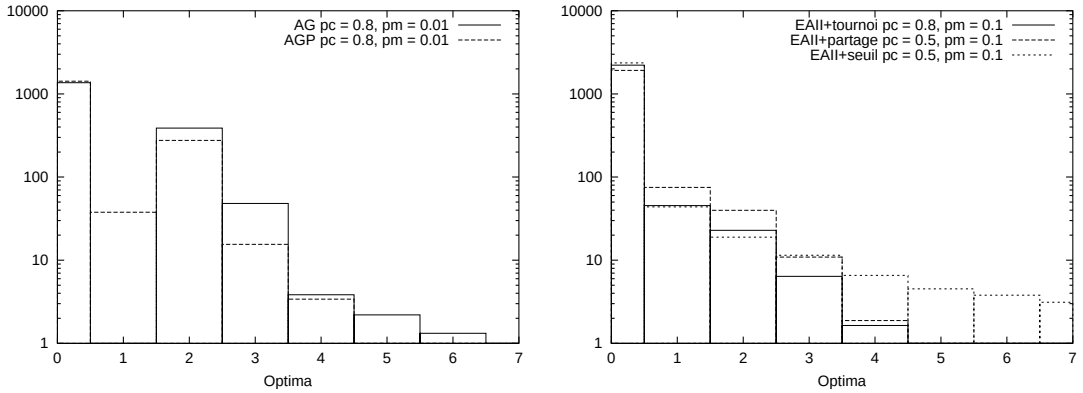


FIG. 5.12 – F_2 : Croisement restreint : Décompte des voisinages d'optima.

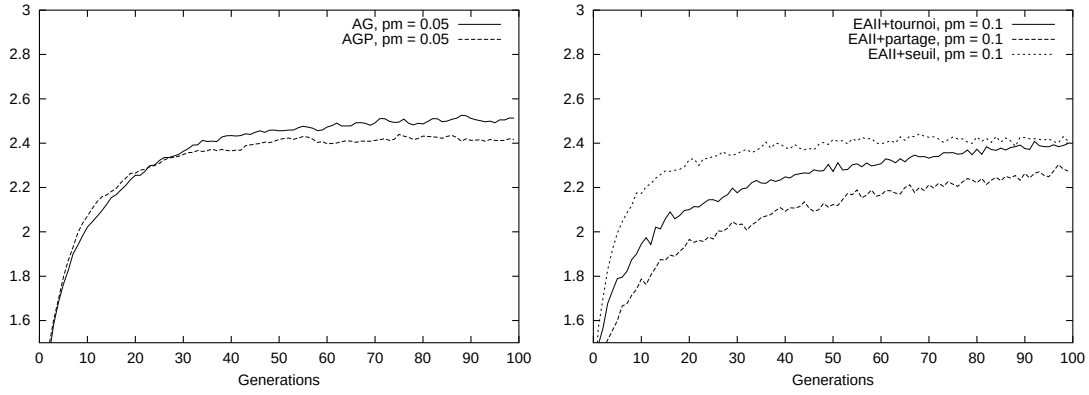


FIG. 5.13 – F_2 : Sans croisement : moyenne sur la population de la fonction de fitness débruitée.

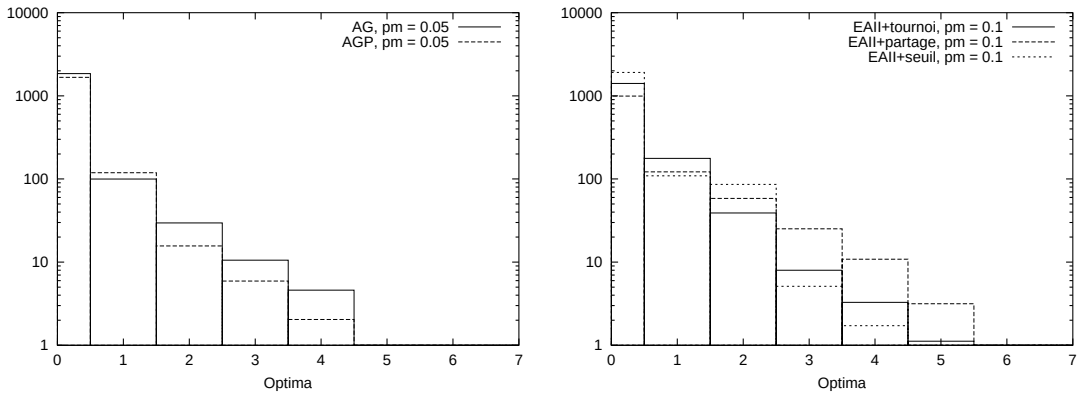


FIG. 5.14 – F_2 : Sans croisement : Décompte des voisinages d'optima.

5.2.4 F_3 : une fonction localement irrégulière.

Définition

Il existe des familles de fonctions réelles pour lesquelles la régularité höldérienne peut être explicitement contrôlée, entre autres les fonctions de Weierstrass-Mandelbrot (voir l'annexe A) :

$$W_{b,s}(x) = \sum_{i=1}^{\infty} b^{i(s-2)} cos(b^i x) \quad \text{avec } b > 1 \text{ et } 1 < s < 2 \tag{5.12}$$

Ces fonctions ne sont nulle part différentiables et possèdent une infinité d'optima locaux. On appelle s la dimension fractale et on a alors directement le coefficient de Hölder par $h = (2 - s)$.

Nous définissons alors F_3 à partir de la fonction wm :

$$\begin{aligned} g(x) &= \frac{2 \times cos(2\pi x)}{1 + 3 \times \sqrt{|x|}} + 2 + W_{2,1.3}(2\pi x) \\ wm(x) &= \frac{g(x)}{g(0)} \end{aligned}$$

La figure 5.15 montre le graphe de cette fonction ainsi que celui de sa "porteuse" :

$$\frac{1}{g(0)} \times \left(\frac{2 \times cos(2\pi x)}{1 + 3 \times \sqrt{|x|}} + 2 \right)$$

La fonction de fitness non bruitée F_3 est alors :

$$\begin{array}{rccc} F_3 : [-0.5, 1.5]^3 : & \rightarrow & \mathbb{R}^+ \\ (x_0, x_1, x_2) & \rightarrow & \sum_{i=0}^{3} wm(x) \end{array} \tag{5.13}$$

Encore une fois la fonction de fitness bruitée est définie comme :

$$f_3 = F_3 \times (1 + N_{(0,0.5)}) \tag{5.14}$$

où $N_{(0,0.5)}$ est un tirage aléatoire gaussien renouvelé à chaque évaluation.

Cette fois la fonction ne comporte qu'un seul optimum global en 0. Même si wm comporte une infinité d'optima locaux, on voit qu'il existe un deuxième bassin d'attraction autour du maximum local ($wm(1) \simeq 0.73$). Nous regarderons donc les décomptes des voisinages des huit optima locaux suivants :

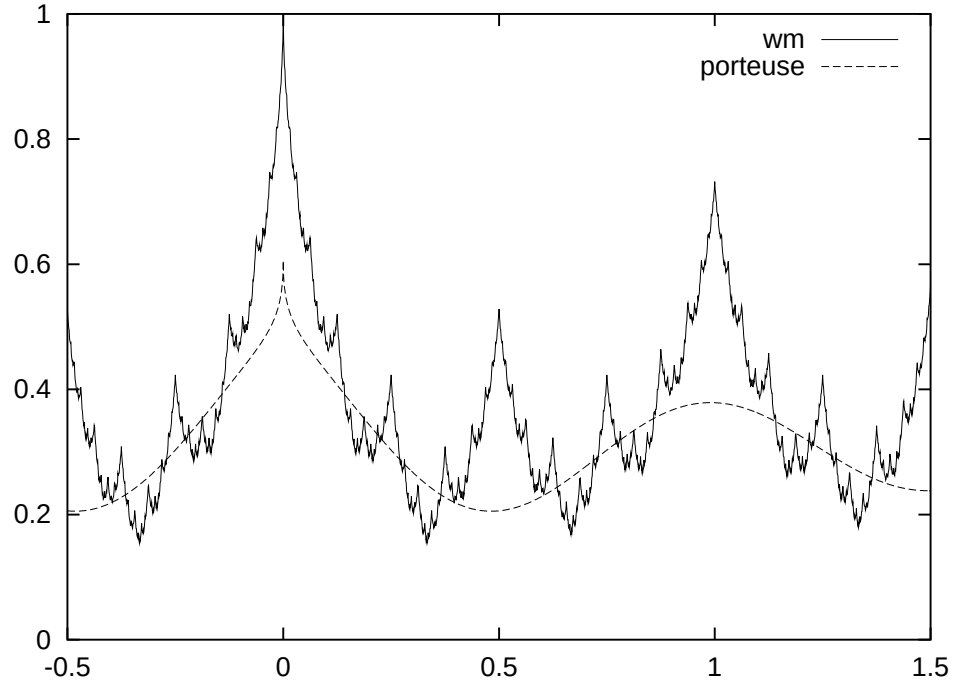


FIG. 5.15 – Graphes de $wm(x)$ et de sa "porteuse"

- $x = (0,0,0)$: optimum global $F_3(0,0,0) = 3$, décompte n_0.
- $x \in \{(0,0,1),(0,1,0),(1,0,0)\}$: optima de second rang ($F_3(.) \simeq 2.73$), décomptes (ordonnés) $n_{1,2,3}$.
- $x \in \{(0,0,1),(0,1,0),(1,0,0)\}$: optima de troisième rang ($F_3(.) \simeq 2.46$), décomptes (ordonnés) $n_{4,5,6}$.
- $x = (1,1,1)$: optimum de quatrième rang ($F_3(.) \simeq 2.19$) décompte n_7.

Pour rendre compte de l'effet de lissage de la pondération de fitness (voir la section section 4.1.5), nous montrons en figure 5.16 le lissage en une dimension avec des noyaux de rayon $\sigma_\infty = 0.05, 0.025$. On voit que même avec un rayon petit, l'erreur par rapport à la fonction non lissée reste significative et plus particulièrement aux alentours des optima. C'est pourquoi nous utiliserons ici une valeur $\sigma_\infty = 0.025$.

Résultats

Les figures 5.17 et 5.19 montrent les performances de FDM avec les deux types de croisement et les figures 5.18 et 5.20 les décomptes de voisinage des optima. Sur ce problème encore la restriction de voisinage permet d'obtenir de meilleures performances. Mais la constatation essentielle issue de l'observation de ces courbes est que dans ce cas c'est l'algorithme génétique qui obtient la meilleure performance malgré le bruit de la fonction de fitness. La différence avec l'AGP met en évidence que dans ce cas précis d'irrégularité locale, la pondération de la fonction de fitness n'est pas une aide efficace. Dans ce problème où l'exploitation du seul optimum global donne les bonnes performances, l'EAII est moins efficace du fait du maintien dans la population d'individus localisés dans le voisinage des optima secondaires. Malgré tout, on constate encore une fois que l'utilisation de la sélection par seuillage offre toujours un bon compromis entre efficacité et

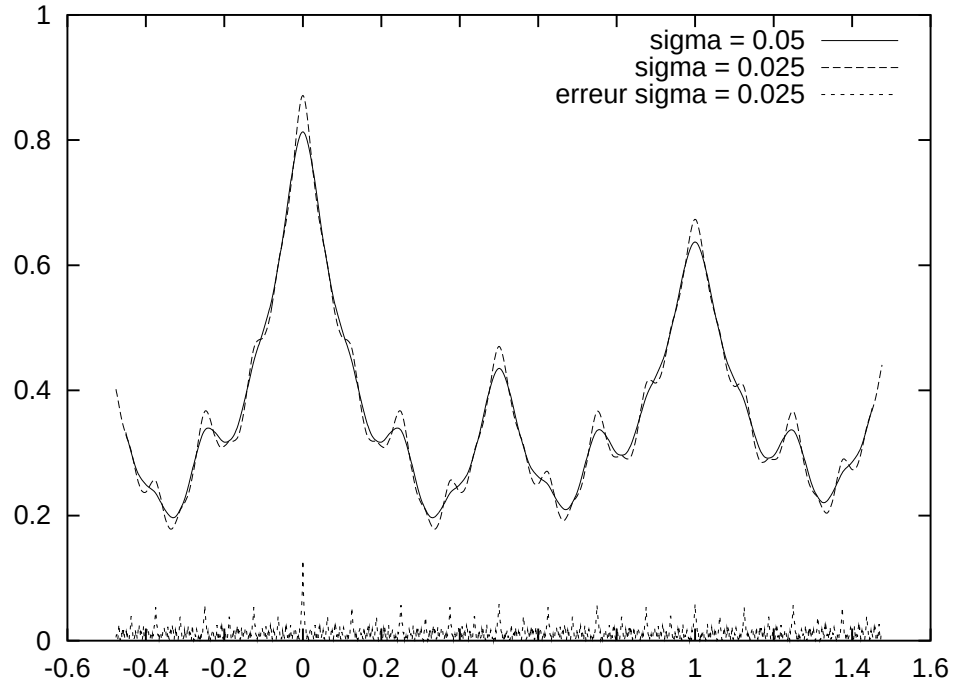


FIG. 5.16 – Graphes de $wm(x)$ lissée avec des noyaux de rayons $\sigma_\infty = 0.05, 0.025$ et de l'erreur avec $\sigma_\infty = 0.025$

diversité.

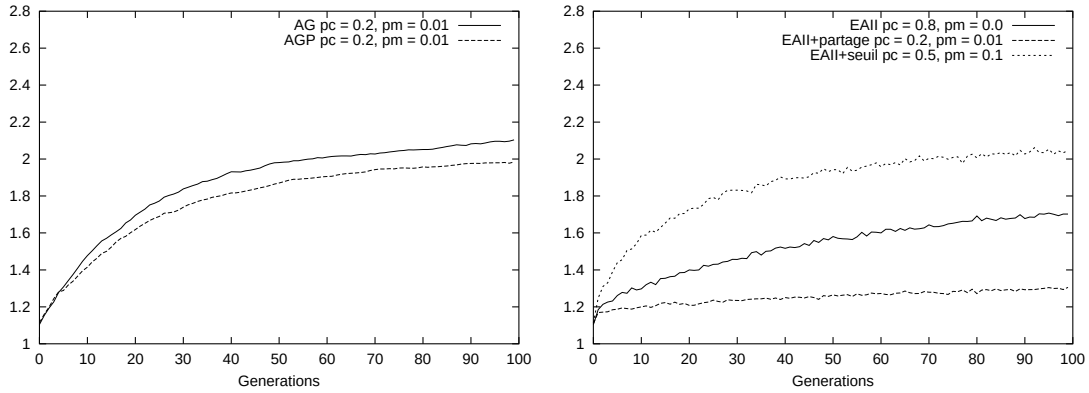


FIG. 5.17 – F_3 : Croisement arithmétique : moyenne sur la population de la fonction de fitness débruitée.

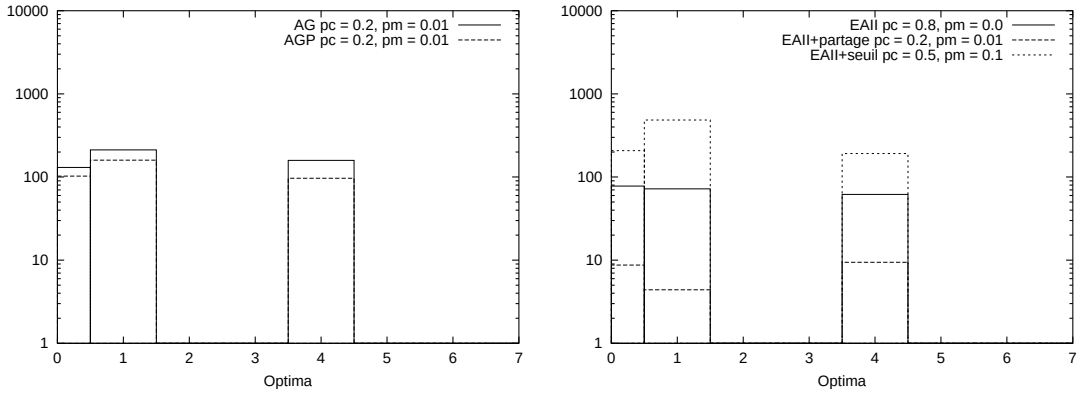


FIG. 5.18 – F_3 : Croisement arithmétique : Décompte des voisinages d'optima..

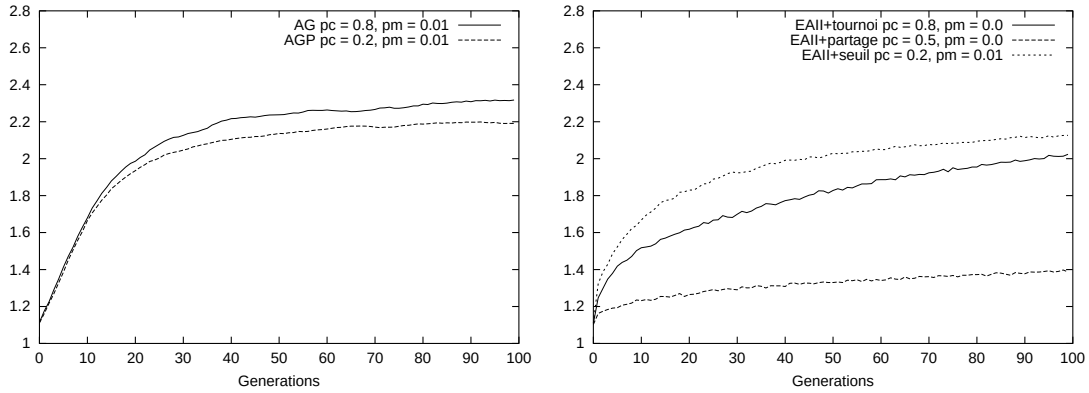


FIG. 5.19 – F_3 : Croisement restreint : moyenne sur la population de la fonction de fitness débruitée.

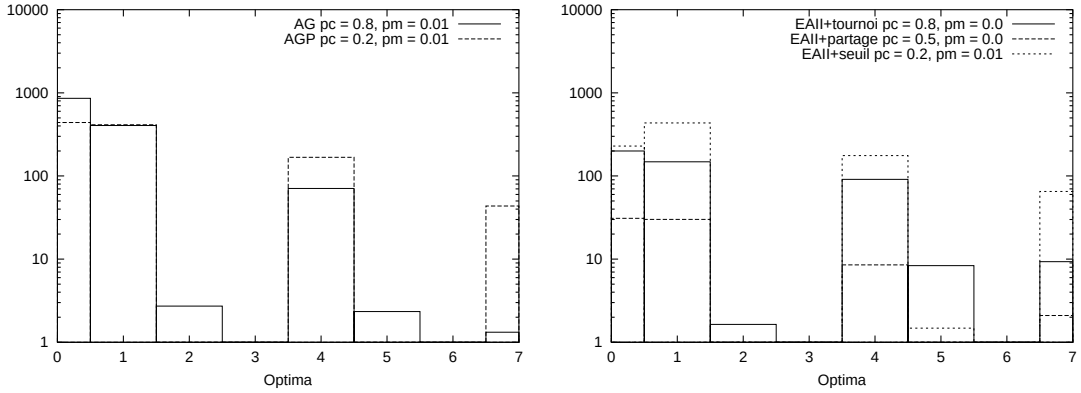


FIG. 5.20 – F_3 : Croisement restreint : Décompte des voisinages d'optima..

5.3 Discussion

Les simulations présentées dans cette section nous ont permis de tester la robustesse de différents algorithmes évolutionnaires dans des conditions bien particulières : un nombre restreint d'évaluations sujettes à un bruit important. Les fonctions de test présentées ne permettent pas d'émettre des conclusions généralisables, néanmoins elles présentent des environnements variés : une fonction multimodale simple (F_1), une fonction épistatique (F_2), et une fonction localement irrégulière (F_3). Nous avons alors cherché à évaluer ces algorithmes avec des critères correspondant à notre problème décrit au chapitre 4 : trouver rapidement des individus performants et si possible localiser tous les optima locaux intéressants. Notons qu'en cela ces objectifs ne sont pas les plus répandus en optimisation, au strict sens du terme, où l'on s'attache plutôt à localiser avec la plus grande précision un optimum global. Au vu des ces résultats, il apparaît alors que des combinaisons testées, la plus robuste est l'EAII avec la sélection par seuillage et l'application du croisement arithmétique avec restriction de voisinage.

Chapitre 6

EAII appliquée à l'optimisation des fréquences des mouvements.

Dans le chapitre 4 nous avons proposé un premier algorithme évolutionnaire permettant d'optimiser les fréquences de Monte Carlo. Puis au regard des conditions d'application nous avons proposé un algorithme évolutionnaire, décrit dans le chapitre 5, moins canonique mais nous paraissant plus adapté à nos besoins. D'autre part nous avons aussi défini dans le chapitre 3 plusieurs critères numériques dans le but de quantifier explicitement l'efficacité de nos simulations. Il nous reste maintenant à faire la synthèse de tout cela et à appliquer l'EAII (Evolution Artificielle d'Individus Immortels) à l'optimisation des fréquences tout en explorant les possibilités offertes par nos différents critères. Nous donnons alors en section 6.1 les détails de cette adaptation, et dans les section suivantes nous présentons des simulations utilisant séparément chacun de ces critères pour l'optimisation des performances.

6.1 Adaptation de l'EAII aux différents critères de mélange.

Ayant défini un nouvel algorithme évolutionnaire pour des fonctions de fitness bruitées et coûteuses en temps de calcul, nous souhaitons maintenant l'appliquer à notre problème de simulation moléculaire ayant justement motivé la définition de cet algorithme.

6.1.1 Modifications de l'algorithme.

Reprenant l'algorithme défini en section 4.1.3, nous sommes maintenant en mesure d'apporter certaines modifications. La première est donc l'utilisation de l'EAII au lieu d'un algorithme évolutionnaire générationnel utilisant uniquement l'histoire H des évaluations afin de réduire le bruit de la fonction de fitness.

Fonction de fitness.

Nous avons vu au chapitre 3 que l'on peut mesurer la vitesse de mélange du système à travers plusieurs indicateurs. Dans le chapitre 4, nous avons utilisé les critères "instantanés", c'est-à-dire comparant uniquement l'état courant à l'état initial. Nous étudierons maintenant l'utilité de chacun des critères que nous avons répertoriés (en choisissant lorsqu'il y a lieu les critères dit cumulés) la fonction de fitness sera alors de la forme :

$$f(\mu, \omega) = c(\mu, \omega) \tag{6.1}$$

où $c(\mu, \omega)$ est l'un des critères. Chacun de ceux-ci étant mesuré sur un temps de simulation identique nous abandonnerons la dépendance au nombre d'échantillons, mais ajouterons celles due aux fréquences des mouvements utilisées (les individus de l'EA) et à la réalisation particulière :

Critère	Nature
$c_1(\mu, \omega) = (1 - C_v^n(\mu, \omega))$	Autocorrélation cumulée des vecteurs d'orientation des chaînes
$c_2(\mu, \omega) = d_c^2(\mu, \omega)$	Déplacement carré des centres de masse des chaînes.
$c_3(\mu, \omega) = d_m^2(\mu, \omega)$	Déplacement carré des monomères.
$c_4(\mu, \omega) = (c_L^3(\mu, \omega) + c_L^4(\mu, \omega))$	Somme des critères de lacunarité aux échelles $R = 3, 4$.
$c_5(\mu, \omega) = (c_L^6(\mu, \omega) + c_L^7(\mu, \omega) + c_L^8(\mu, \omega))$	Somme des critères de lacunarité aux échelles $R = 6, 7, 8$.

Phases exploration/ exploitation

La durée totale de simulation sera divisée en deux parties : une phases dite d'exploration où les fréquences des mouvements correspondent aux individus créés par l'EAII, et une phase dite d'exploitation où les fréquences utilisées correspondront au meilleur individu présent dans l'histoire H de l'évolution. Cependant les performances seront toujours mesurées, et donc la valeur de fitness pondérée f_H de cet individu sera susceptible de varier encore. Ainsi un individu jugé très bon sur la base de quelques évaluations chanceuses aura sa valeur corrigée rapidement laissant un autre meilleur individu apparaître : si l'on ne crée plus d'individus durant la phase

d'exploitation on a toujours la possibilité de réviser l'attribution du titre de "meilleur individu" parmi ceux présents dans H.

Fréquence constante des fluctuations de volume

Ce mouvement est nécessaire à l'équilibration du volume (voir la section 2.3.3) lorsqu'une pression constante est imposée. Il n'est donc pas souhaitable qu'il puisse disparaître s'il l'évolution le juge inefficace au vu des critères que nous nous sommes donnés. En fait il remplit seul une fonction indispensable et le fait de lui affecter une fréquence fixe permet d'économiser un paramètre dans l'optimisation. **Les individus coderont donc les FCE (Fréquences à Charge Egale) des mouvement hors Fluctuation de Volume (FV).**

6.1.2 Conditions de simulation.

Paramètres généraux.

Les simulations présentées dans les prochaines sections ont toutes pour objet la simulation de polyéthylène linéaire [1] dans l'ensemble $nNPT$. Les paramètres physico-chimiques seront identiques dans toutes les séries de simulations, l'objet d'étude étant alors le critère d'efficacité utilisé comme fonction de fitness. Toutefois nous considèrerons deux longueurs de monomères :

- $n = 640$ monomères $CH2$ ou $CH3$.
- $N = 20$ chaînes ou $N = 10$ chaînes. Les chaînes comptent alors 32 ou 64 monomères.
- $P = 1atm$
- $T = 450K$
- $n_c = 80000$ cycles de simulations (1 cycle = 1 seconde de simulation sur un Pentium III 733 Mhz du cluster de l'INRIA Grenoble).

Les mouvements utilisés (voir la section 2.3.3) sont :

- La Translation ($deg = \frac{n}{N}$)
- La Rotation ($deg = 1$)
- La Reptation ($deg = 1$)
- Le Flip ($deg = 1$)
- La Fluctuation de Volume ($deg = n$), FCE fixée à 0.2.

Encore une fois, sachant qu'il y a 5 mouvements, toutes les FCE de l'Algorithme de Référence (désigné par **AR**) sont égales à 0.2, ce qui correspond aux FE suivantes :

N	Translation	Rotation	Reptation	Flip	Fluctuation volumique
20	20 / 1941	640 / 1941	640 / 1941	640 / 1941	1 / 1941
10	10 / 1931	640 / 1931	640 / 1931	640 / 1931	1 / 1931

Evolution Artificielle.

Au vu des résultats présentés au chapitre 5, l'EAII utilisé dans les conditions suivantes nous semble être une solution robuste :

- Sélection par seuillage (section 5.1.2), avec $c_s = 0.8,\ n_t = 10$.
- Croisement arithmétique avec restriction de voisinage : $p_c = 0.8$.
- Mutation gaussienne : $p_m = 0.05$.

[1] Nous renvoyons à la section 2.3.1 pour la description du modèle moléculaire et à la section 4.2.1 pour la description des conditions générales des simulations numériques.

- Rayon de voisinage $\sigma_\infty = 0.05$.
- Nombre de cycles de simulation : $n_c = 80000$
- Nombre total d'évaluations : 2560, soit 32 systèmes simulés avec 80 subdivisions du temps de simulation. Le temps d'évaluation est donc de 2000 secondes.
- Taux d'exploration de 0.5 : 1280 individus créés pour l'exploration, 1280 pour l'exploitation.

6.1.3 Présentation des résultats

Les sections suivantes présentent les résultats de ces simulations, chacun des critères faisant l'objet d'une section propre. Nous y analyserons les simulations pour les deux longueurs de chaîne, et les simulations y seront alors référencées par une lettre suivie de la longueur des chaînes (F32, F64,...).

Les courbes suivantes y seront systématiquement présentées :

- Les courbes de fitness moyenne pondérée et non pondérée : les valeurs sont les moyennes des évaluations des critères faites sur la population des 32 individus (fréquences) alloués aux 32 systèmes simulés en parallèle, au fur et à mesure de la simulation. Ici encore le terme population ne désigne qu'un regroupement des nouveaux individus de taille égale au nombre de systèmes, qui n'intervient pas dans la dynamique de l'EAII.
- Les histogrammes des FCE : histogrammes de toutes les valeurs prises par les FCE fournies par l'EAII pendant toute l'évolution. Les pics de ces histogrammes fournissent donc les valeurs préférées par l'EAII.
- Les courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes mesurées sur toute la durée de la simulation, que nous désignerons par l'abréviation **ACO**. Rappelons que les critères servant à donner la fitness des individus sont mesurés sur une subdivision de cette durée et non sur sa globalité.
- Les courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes (que nous désignerons par l'abréviation **DCM**) et des monomères (désigné par **DM**), mesurées sur toute la durée de la simulation
- Les courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées (désignée par **LCC**) sur toute la durée de la simulation, généralement aux échelles $R = 3$ et $R = 7$.

6.2 Autocorrélation des vecteurs d'orientation : F32 et F64

$$f(\mu, \omega) = c_1(\mu, \omega) = (1 - C_v^n(\mu, \omega))$$

On voit sur la figure 6.1 les courbes de fitness et comme on s'y attend le critère de décorrélation est plus important pour les systèmes à chaînes courtes. Toutefois dans les deux cas le critère est amélioré, même si du fait du nombre moins important des longues chaînes, il est plus oscillant pour la série F64, que l'on considère ou non la pondération. On remarque qu'au milieu de la simulation, lors du passage à la phase d'exploitation (seul l'individu ayant la meilleure valeur de fitness pondérée est utilisé), il y a un saut important de la moyenne de fitness pondérée pour la série F32. On remarque en outre que celle-ci décroit un peu par la suite et se stabilise. Cela s'explique par un nombre restreint d'évaluations pour ce bon individu, parmi lesquelles une ou plusieurs évaluations chanceuses (dont la valeur est largement au dessus de l'espérance), et les évaluations supplémentaires venant, cette surestimation est corrigée.

Concernant la performance ACO (figure 6.4), pour la série F64 l'amélioration du critère c_1 sur une courte durée se révèle corrélée avec une amélioration du critère sur la globalité de la simulation. Toutefois la rapidité de décorrélation pour la série F32 empêche que le critère ACO pour l'EAII soit meilleur que pour l'AR. Cependant en regardant de plus près la figure 6.5, on se rend compte de la rapidité de décroissance de l'autocorrélation instantanée, qui oscille rapidement autour de 0. Sachant qu'au début de la simulation EAII, la phase exploratoire (lors de laquelle sont testées des fréquence peu efficaces) donne un léger retard à la décroissance de l'autocorrélation instantanée et que celui-ci se retrouve dans le décalage observé des courbes ACO. Par contre, on voit bien avec le critère DCM (figure 6.3), que pour les deux séries l'EAII finit par trouver une dynamique plus rapide. Il est important de remarquer qu'en optimisant uniquement le critère c_1, on améliore les performances à la fois en termes d'ACO et de DCM. Cela confirme expérimentalement l'intuition qui consiste à dire que les chaînes changent d'autant plus vite d'orientations qu'elles se déplacent vite.

En regardant les histogrammes des mouvements (figure 6.2), on retrouve le fait que la reptation est le mouvement privilégié dans les deux cas. De plus on remarque que pour chacun des mouvements, une large plage de fréquences a été explorée. L'effet de la phase d'exploitation se retrouve dans les pics des histogrammes. Insistons encore sur le fait que les combinaisons correspondantes ne peuvent être considérées que comme bonnes (meilleures que la moyenne) et non comme optimales au sens strict. On remarque d'ailleurs pour la série F64, que plusieurs pics apparaissent, résultant du fait que lors de la phase d'exploitation, la correction d'estimation d'un individu initialement surestimé fait apparaître un autre "bon individu" comme meilleur, expliquant l'oscillation de la fitness pondérée même lors de la phase d'exploitation.

Pour information nous donnons les meilleurs individus. Par meilleur individu nous entendons l'individu ayant la meilleure valeur de fitness pondérée à la fin de l'évolution, tout en ayant un poids d'estimation w suffisamment important. Dans ce cas on a $w = 1229$ et $f_H = 0.26$ pour le meilleur individu de la simulation F32, et $w = 478$ et $f_H = 0.052$ pour la simulation F64 (sachant que 3200 évaluations sont faites, w est compris dans l'intervalle $[0, 3200]$). Il est intéressant de remarquer que dans ce dernier cas, ce meilleur individu n'a pas été le plus "exploité", car il correspond à des pics secondaires des histogrammes pour les mouvements de la reptation et de rotation.

	Série	Translation	Rotation	Reptation	Flip	FV
FCE hors FV	F32	14.9 %	14.3 %	65.9 %	4.9 %	-
FCE	F32	11.9 %	11.5 %	52.7 %	3.9 %	20 %
FE	F32	0.54 %	16.8 %	76.9 %	5.7 %	0.0456 %
FCE hors FV	F64	22.2 %	24.8 %	33.3 %	19.7 %	-
FCE	F64	17.8 %	19.8 %	26.6 %	15.8 %	20 %
FE	F64	0.444 %	31.7 %	42.5 %	25.3 %	0.05 %

Enfin remarquons que l'amélioration des performances en termes d'ACO et de DCM ne s'accompagne pas forcément d'une amélioration en termes de lacunarité (figures 6.6 et 6.7).

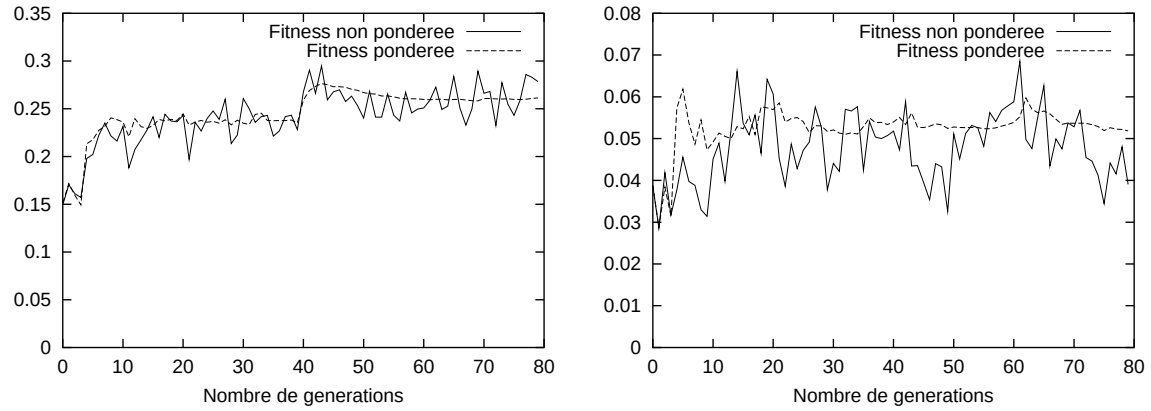


FIG. 6.1 – Séries F32 (gauche) et F64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

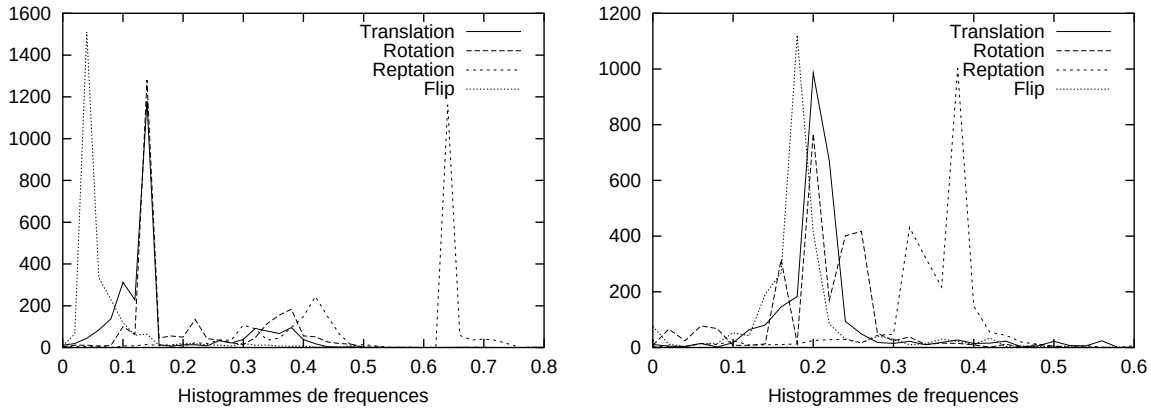


FIG. 6.2 – Séries F32 (gauche) et F64 (droite) : histogrammes des FCE des mouvements.

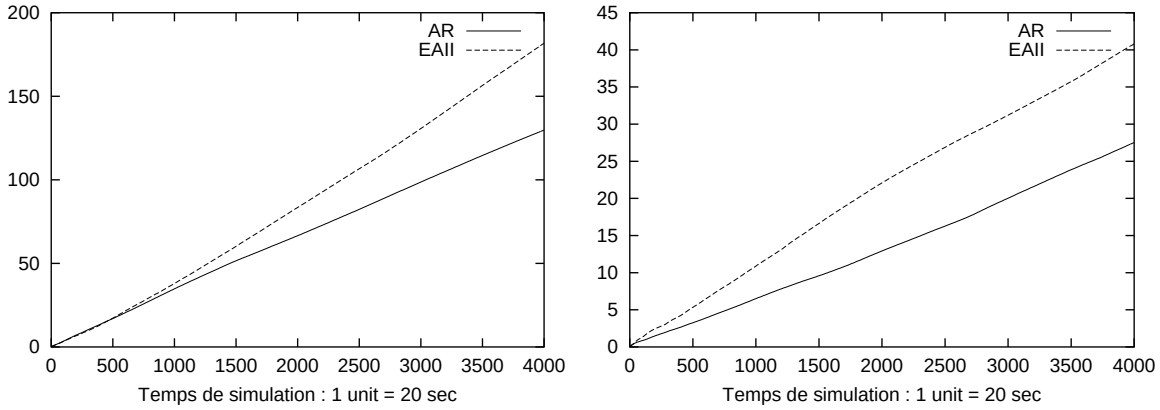


FIG. 6.3 – Séries F32 (gauche) et F64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité = σ_{LJ}^2).

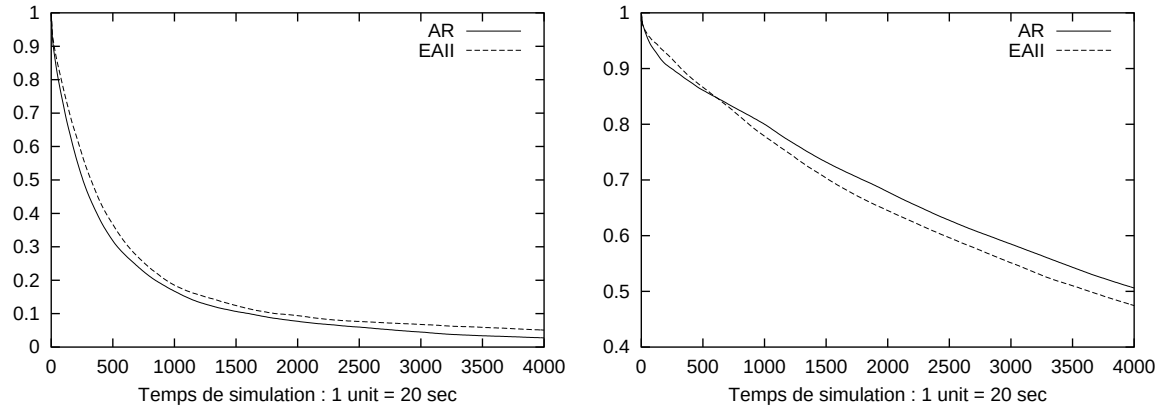


FIG. 6.4 – Séries F32 (gauche) et F64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes.

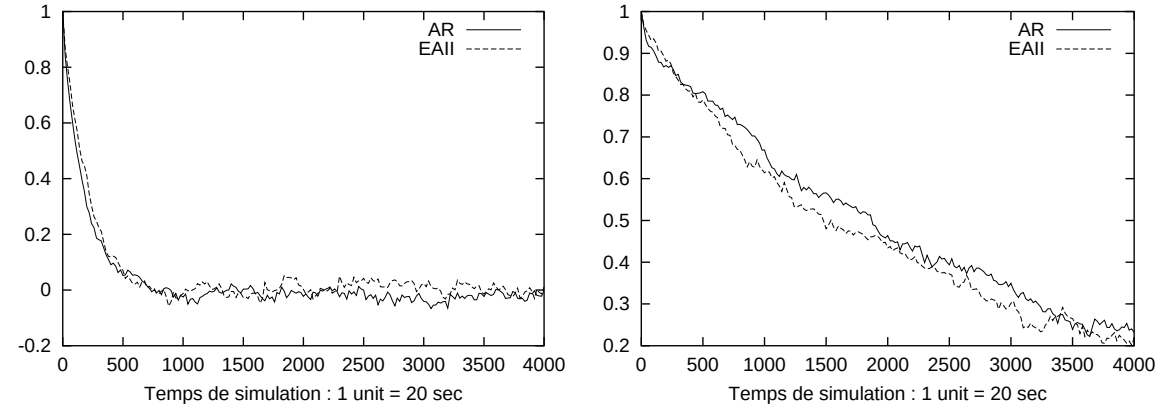


FIG. 6.5 – Séries F32 (gauche) et F64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation instantanée des vecteurs d'orientation des chaînes.

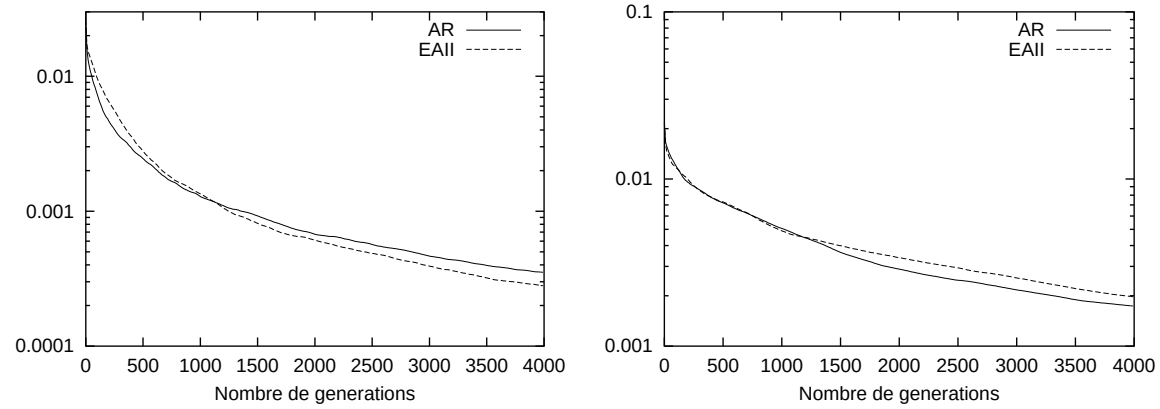


FIG. 6.6 – Séries F32 (gauche) et F64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 3$.

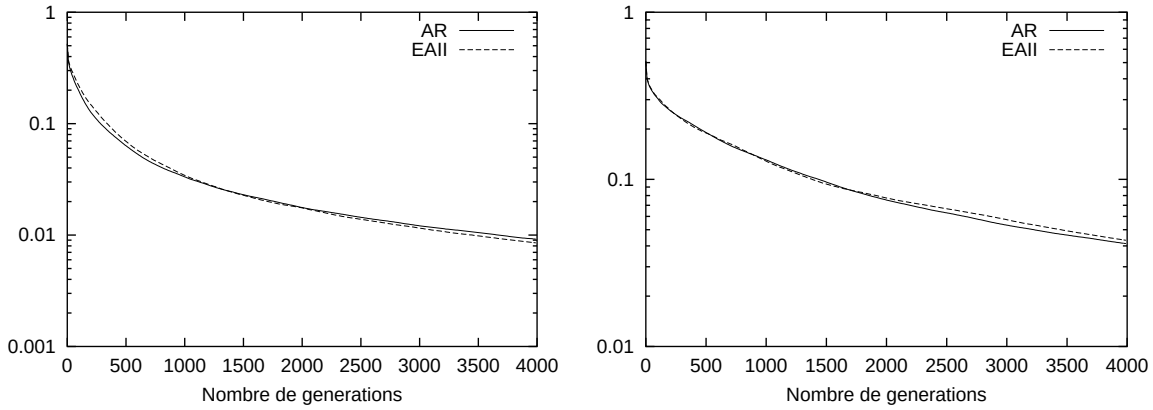


FIG. 6.7 – Séries F32 (gauche) et F64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 3$.

6.3 Déplacement des centres de masse : G32 et G64

$$f(\mu, \omega) = c_2(\mu, \omega) = d_c^2(\mu, \omega)$$

Encore une fois on note des oscillations plus importantes des courbes des fitness (figure 6.8) pour la simulation G64, fait que l'on peut encore expliquer par le nombre moins important de chaînes. On remarque aussi la corrélation entre les performances ACO et DCM (figures 6.9 et 6.11), qui sont améliorées conjointement en utilisant le critère c_2, avec un léger avantage de G64 sur F64. D'autre part on retrouve dans les histogrammes des FCE des chaînes de taille 32, la nette préférence allant à la reptation. Concernant les chaînes plus longues, cette tendance est moins marquée, mais on remarque surtout une "hésitation" entre les fréquences de la reptation et de la rotation, fait déjà observé sur les histogrammes de F64, mais peut-être moins clairement. On retrouve alors un fait souligné dans [50] : pour que le reptation soit efficace pour des longues chaînes, il est nécessaire qu'elle soit accompagnée de mouvements qui changent l'environnement des monomères de bout de chaîne. En effet, lorsqu'une reptation s'effectue, cela laisse un espace vide à l'ancien emplacement du monomère déplacé à l'autre bout de la chaîne. Si aucun autre mouvement n'est utilisé conjointement il est très probable qu'une reptation "inverse" se produise, profitant de cet espace libre : les deux reptations auront un effet à peu près nul sur l'orientation et le déplacement de la chaîne. Par contre si une rotation du monomère de bout de chaîne se produit, cela peut fermer cette possibilité de retour. De même, un mouvement de flip peut amener un monomère interne d'une autre chaîne à occuper cet espace libre réduisant la possibilité de retour. Ces autres mouvements peuvent donc être considérés comme des mouvements auxiliaires augmentant l'efficacité de la reptation.

Enfin on a confirmation avec les figures 6.12 et 6.13 que l'amélioration des performances ACO et DCM n'est pas corrélée avec une amélioration des performances LCC.

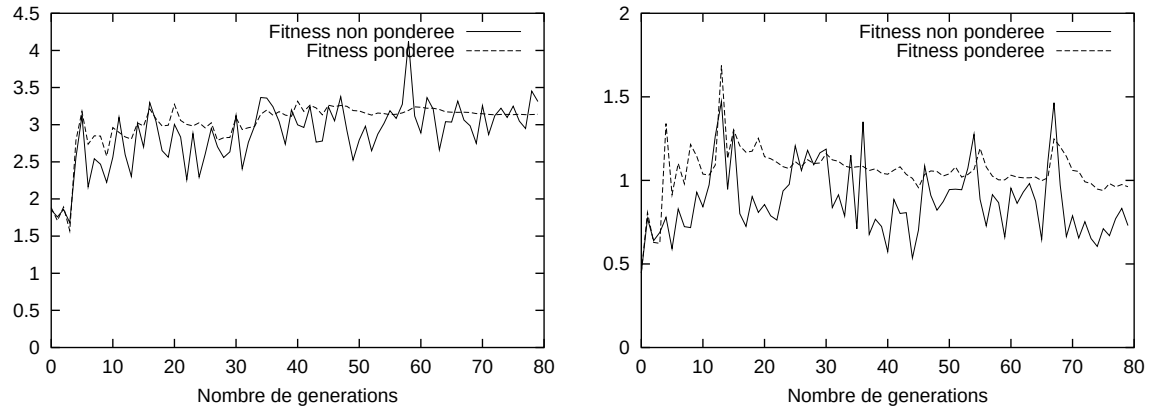


FIG. 6.8 – Séries G32 (gauche) et G64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

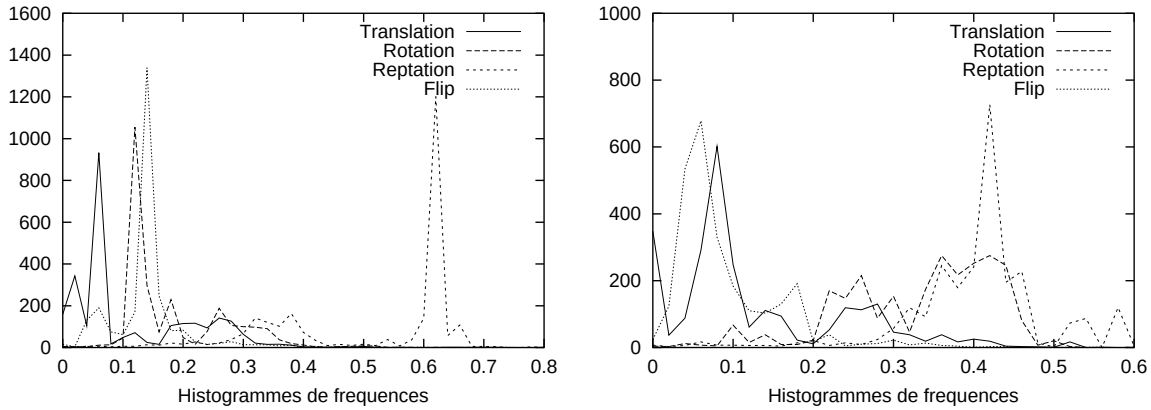


FIG. 6.9 – Séries G32 (gauche) et G64 (droite) : histogrammes des FCE des mouvements.

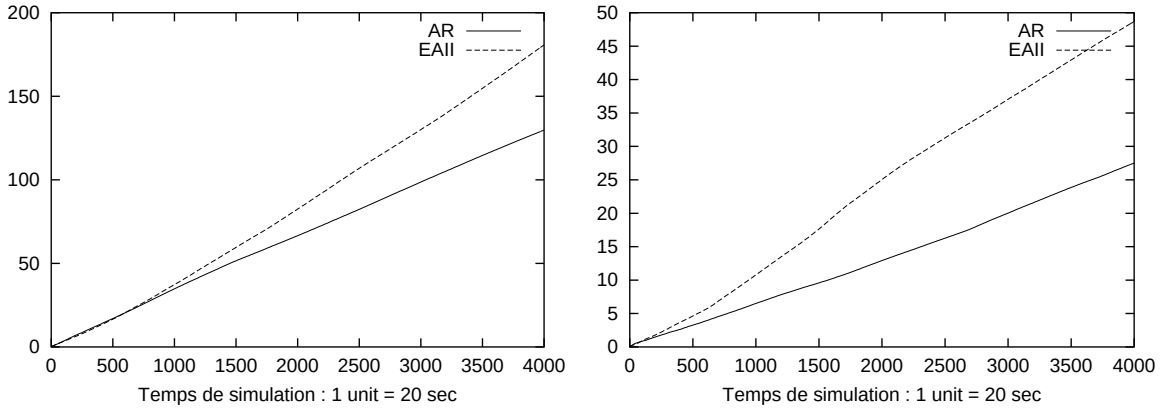


FIG. 6.10 – Séries G32 (gauche) et G64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité = σ_{LJ}^2).

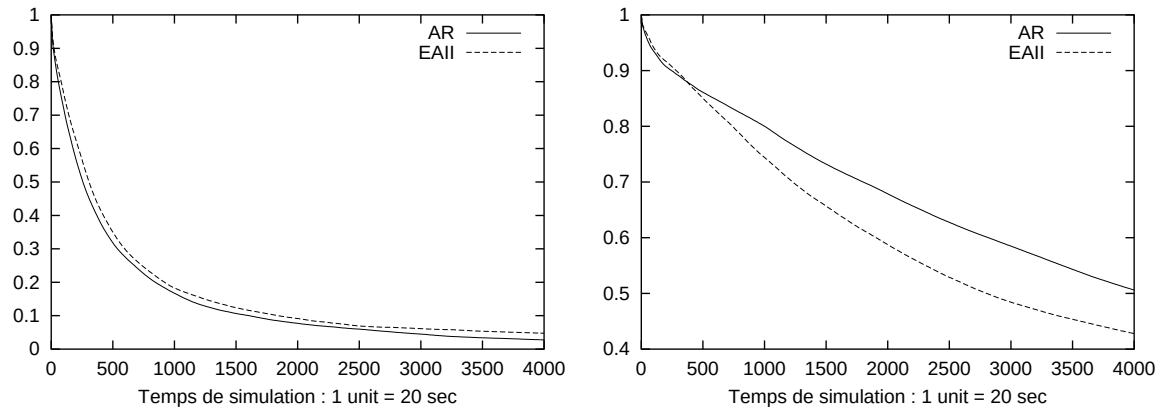


FIG. 6.11 – Séries G32 (gauche) et G64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes.

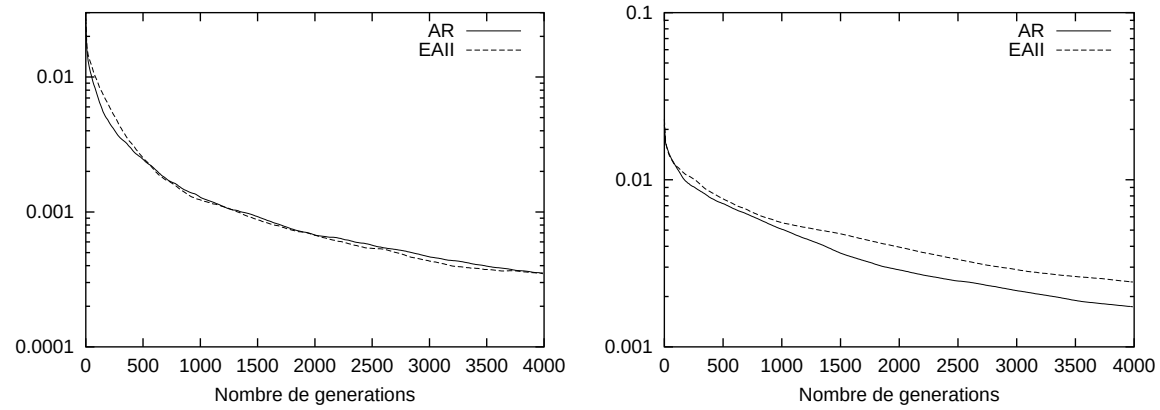


FIG. 6.12 – Séries G32 (gauche) et G64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 3$.

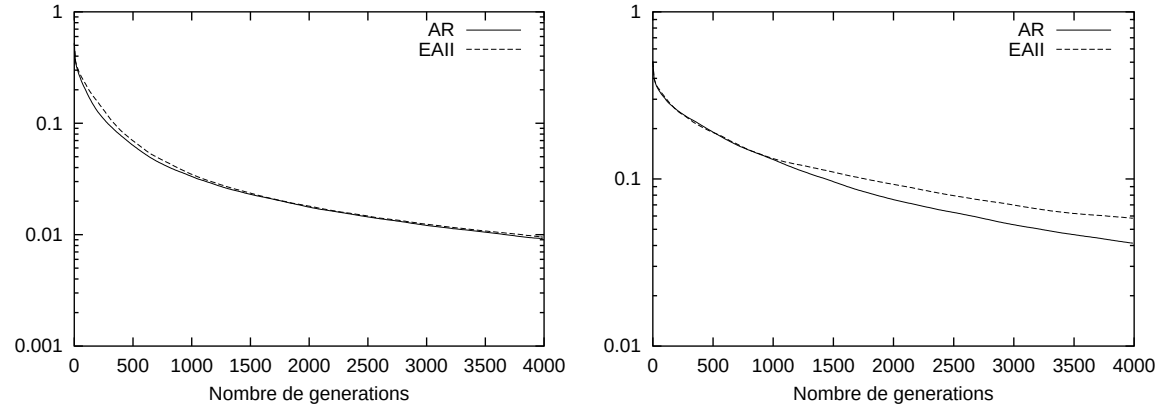


FIG. 6.13 – Séries G32 (gauche) et G64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 7$.

6.4 Déplacement des monomères : H32 et H64

$$f(\mu,\omega) = c_3(\mu,\omega) = d_m^2(\mu,\omega))$$

Le déplacement du centre de masse d'une chaîne est lié au déplacement de ses monomères dans la mesure où, par définition, un déplacement du centre de de masse d'une chaîne ne peut s'effectuer qu'avec un déplacements d'au moins un de ses monomères. Par contre il est aussi évident que ces déplacement ne seront pas toujours proportionnels. On comprend que de petites fluctuations des monomères puissent entraîner un déplacement moyen de ceux-ci beaucoup plus important que celui du centre de masse. Nous commencerons donc par regarder la figure 6.18, montrant conjointement les déplacements carrés moyens des centres des masses et des monomères pour les simulations H32 et H64. On peut alors observer qu'une fois que le centre de masse d'une chaîne s'est éloigné d'une certaine distance (d'autant plus grande que la chaîne est grande), ces déplacements deviennent proportionnels à un décalage près, i.e. ils ont une pente commune.

Cette "vélocité" initiale plus importante pour les monomères se traduit naturellement par un critère c_3 meilleur que c_2, ce que l'on constate sur les courbes de fitness (figure 6.14). Si l'EAII apporte une amélioration du critère, elle est quantitativement moins importante que pour c_2 précisément à cause de cette différence. Pour le reste, on retrouve des histogrammes similaires aux précédents, bien qu'on puisse dire au vu des courbes de DCM et d'ACO que les performances sont un peu moins bonnes.

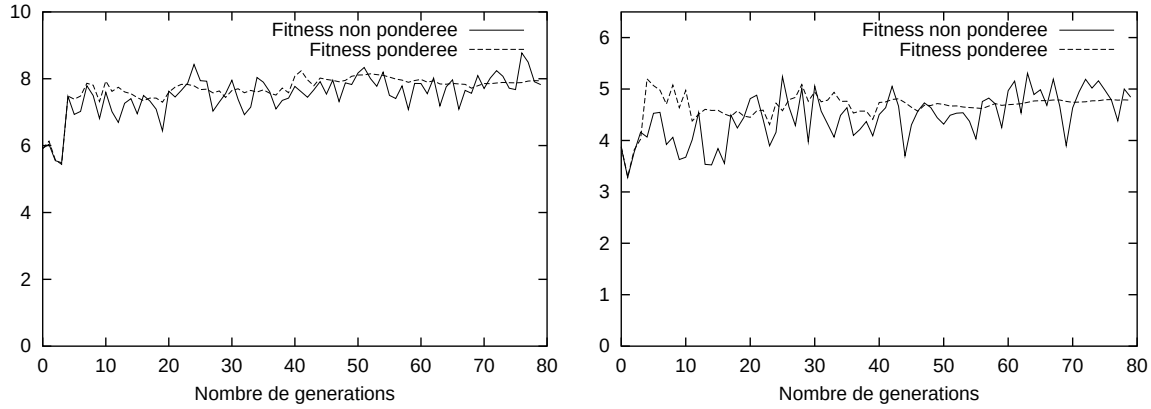


FIG. 6.14 – Séries H32 (gauche) et H64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

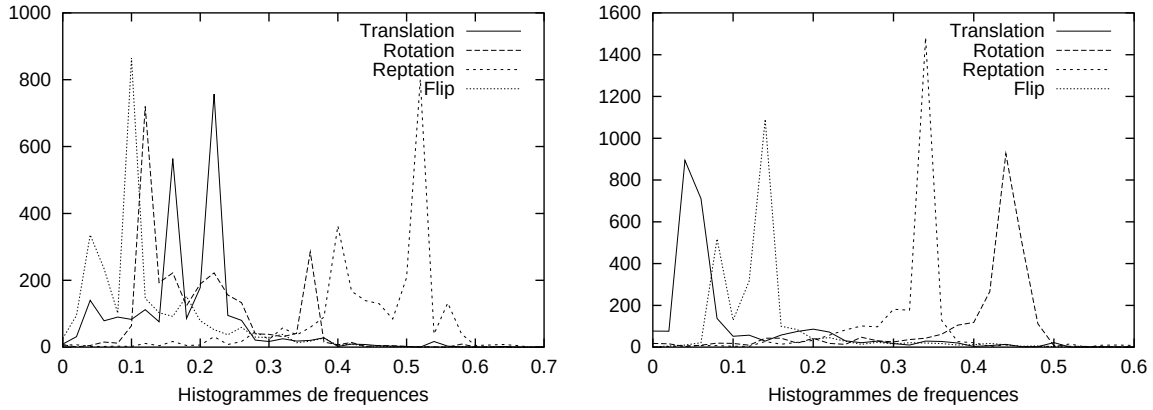


FIG. 6.15 – Séries H32 (gauche) et H64 (droite) : histogrammes des FCE des mouvements.

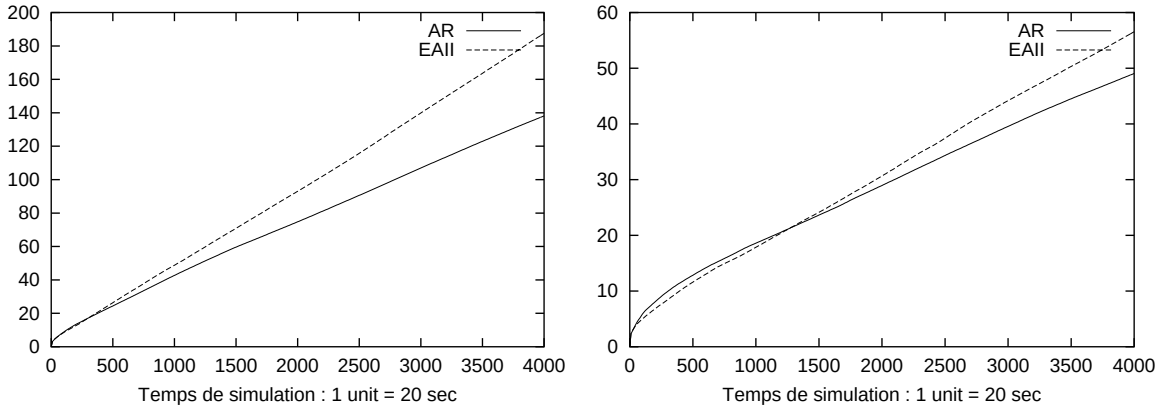


FIG. 6.16 – Séries H32 (gauche) et H64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des monomères. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

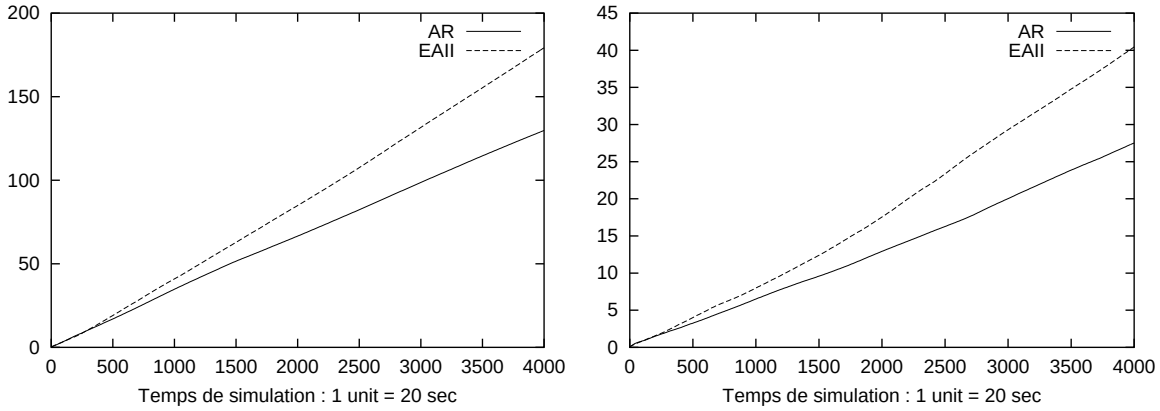


FIG. 6.17 – Séries H32 (gauche) et H64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

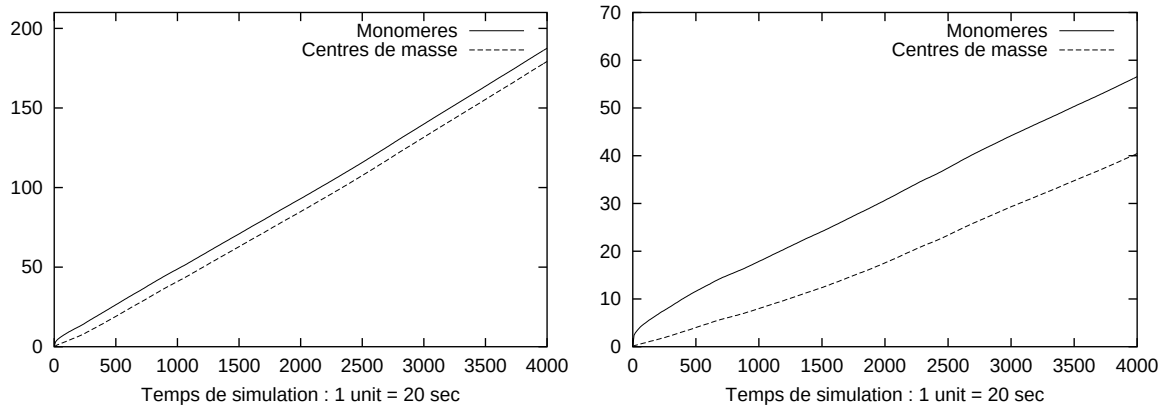


FIG. 6.18 – Séries H32 (gauche) et H64 (droite) : comparaison des courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse **et** des monomères. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

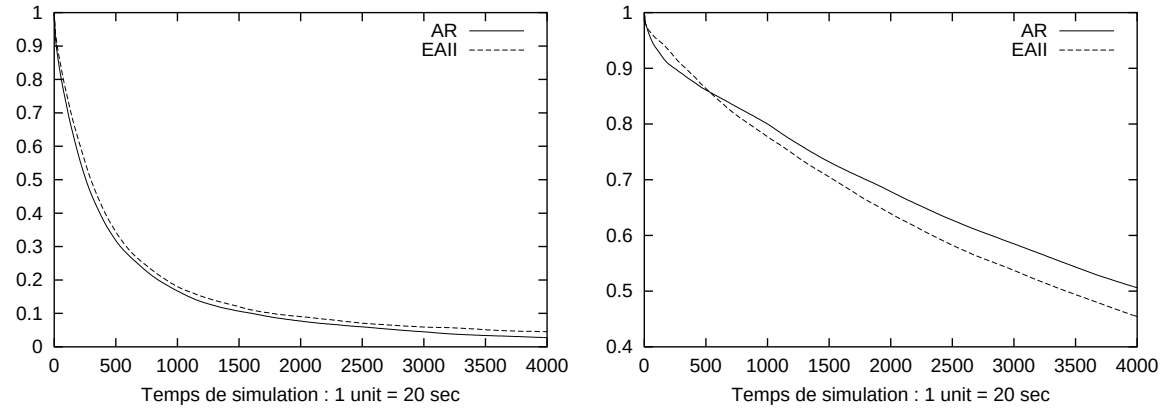


FIG. 6.19 – Séries H32 (gauche) et H64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes.

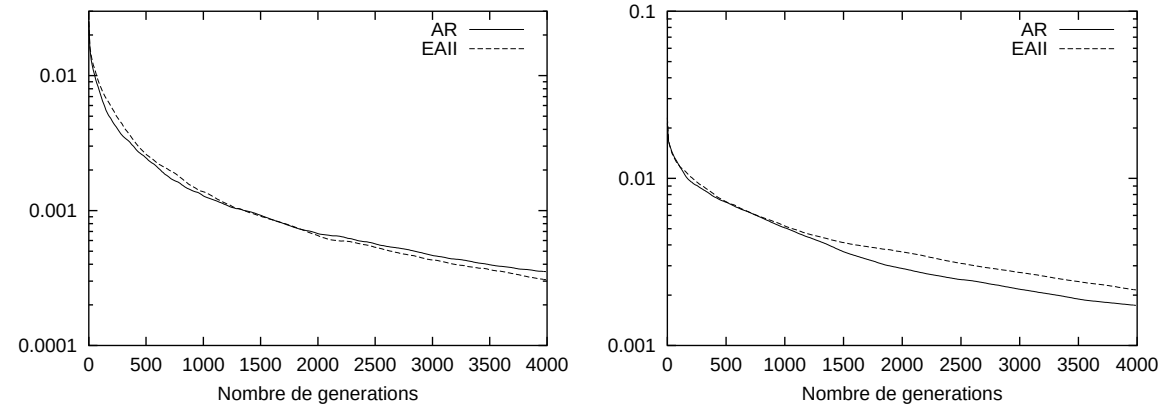


FIG. 6.20 – Séries H32 (gauche) et H64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 3$.

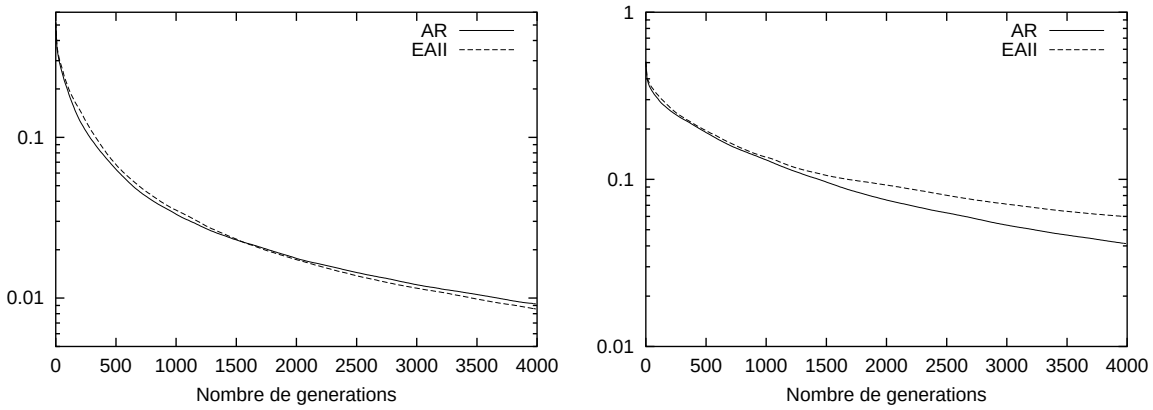


FIG. 6.21 – Séries H32 (gauche) et H64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 7$.

6.5 Lacunarité, $R = 3, 4$: I32 et I64.

$$f(\mu, \omega) = c_4(\mu, \omega) = \left(c_L^3(\mu, \omega) + c_L^4(\mu, \omega)\right)$$

A ces échelles, la boîte de simulation est divisée en 16 et 27 cellules, dont le critère c_4 va mesurer la vitesse d'uniformisation de la répartition de la matière. Tout d'abord sur de courtes périodes, il est évident au vu des courbes de fitness (figures 6.22), que l'EAII trouve des FCE qui améliorent ces critères. Si l'on regarde les figures 6.24 et 6.25, on constate aussi une amélioration sur toute la durée de la simulation. Par contre il y a une inversion des performances ACO et DCM (figures 6.26 et 6.27). Regardant les histogrammes des FCE, on voit que la translation est le mouvement privilégié, à l'opposé des séries précédentes. On peut alors en conclure que le mouvement de translation aidera à rapidement à uniformiser les variations de densité locale, mais qu'en augmentant la rapidité de cette uniformisation, on ralentit le déplacement des chaînes. On peut le comprendre intuitivement en considérant qu'une telle uniformisation peut être obtenue en impulsant de petits déplacements fréquents (effet typique des translations) plutôt qu'à des déplacements rares mais plus importants.

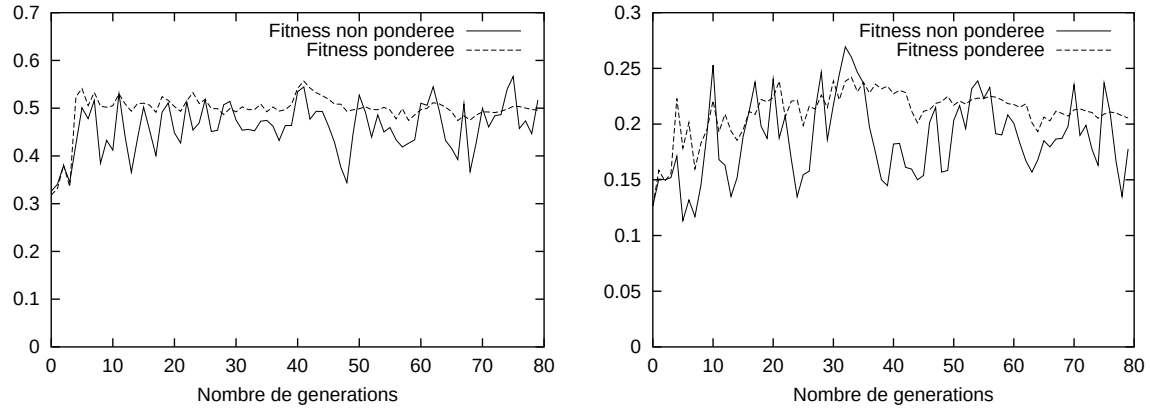


FIG. 6.22 – Séries I32 (gauche) et I64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

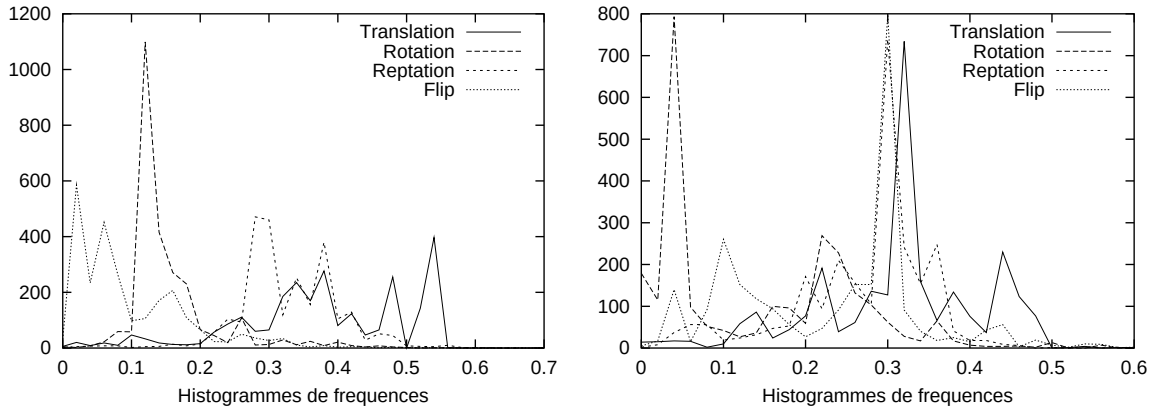


FIG. 6.23 – Séries I32 (gauche) et I64 (droite) : histogrammes des FCE des mouvements.

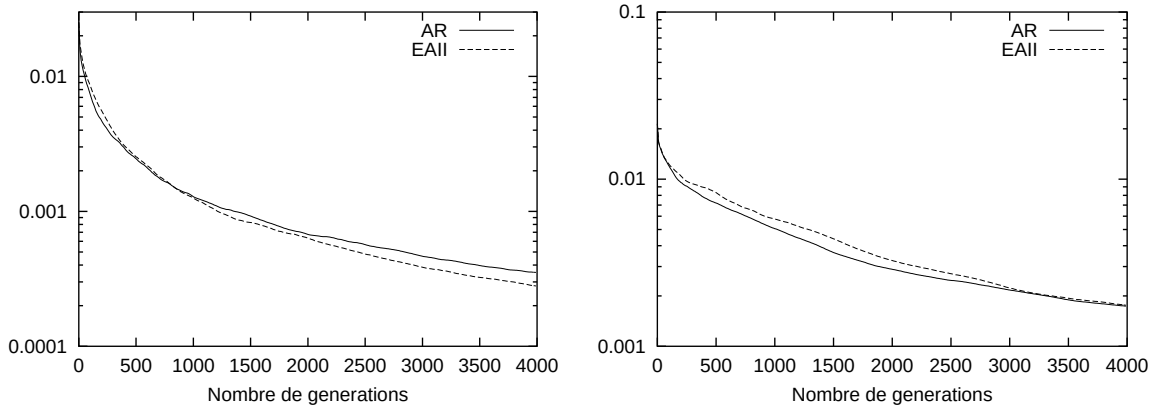


FIG. 6.24 – Séries I32 (gauche) et I64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 3$.

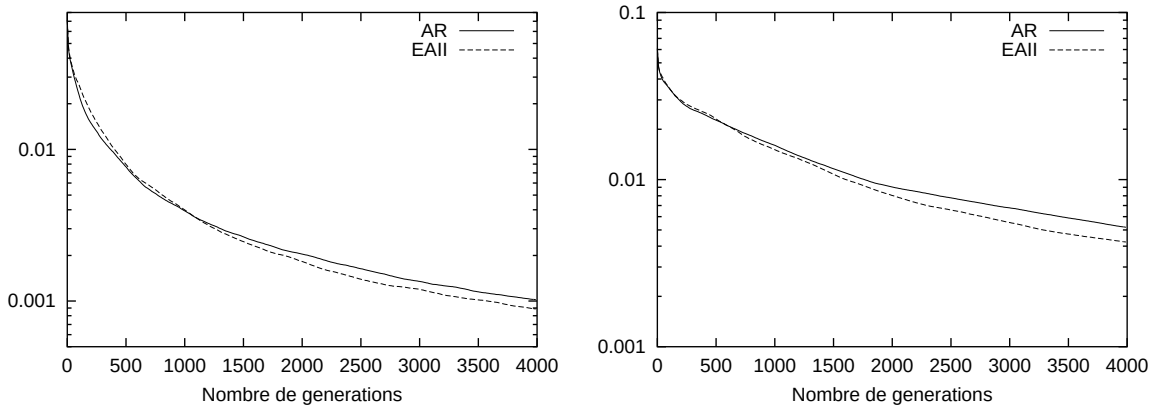


FIG. 6.25 – Séries I32 (gauche) et I64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 4$.

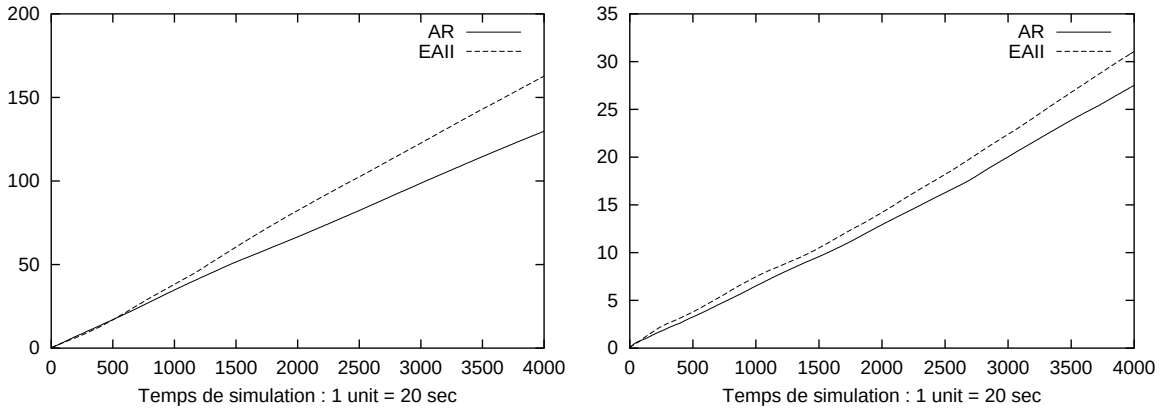


FIG. 6.26 – Séries I32 (gauche) et I64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma^2_{LJ}$).

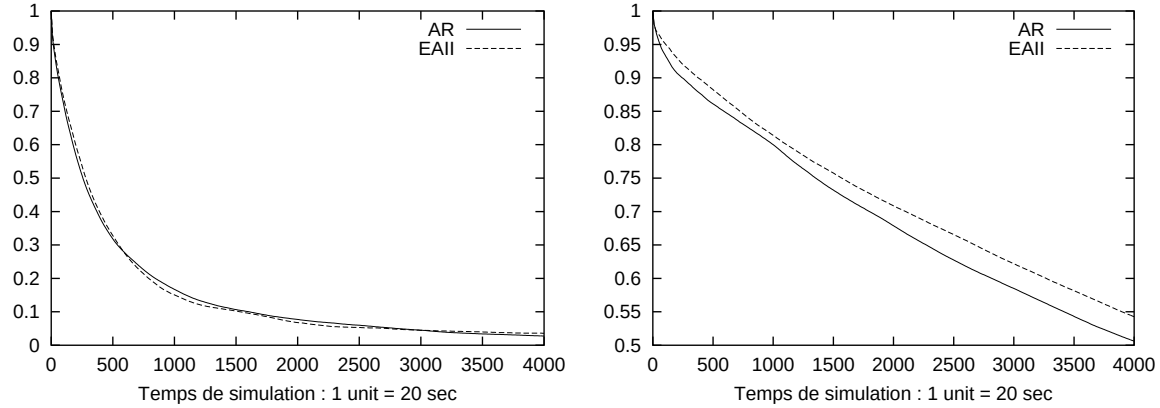


FIG. 6.27 – Séries I32 (gauche) et I64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes.

6.6 Lacunarité, $R = 6, 7, 8$: J32 et J64.

$$f(\mu,\omega) = c_5(\mu,\omega) = \left(c_L^6(\mu,\omega) + c_L^7(\mu,\omega) + c_L^8(\mu,\omega)\right)$$

Comme nous l'avons vu à la section 3.2.2, ces valeurs d'échelles correspondent alors aux échelles d'échantillonnage des trous (la largeur des cellules étant de l'ordre de la distance caractéristique de Lennard Jones). On remarque alors que si à ces échelles il est possible de faire décroître plus vite la lacunarité grâce à l'EAII (figures 6.30 à 6.32), on trouve comme pour les simulations précédentes que cela se fait au détriment des performances ACO et DCM (figures 6.33 et 6.34), quoique pour les chaînes courtes on note tout de même un petite amélioration du DCM. On constate encore que c'est le mouvement de translation qui est privilégié (figure 6.29), et au regard de la précédente remarque on peut dire que la translation peut avoir un effet non négligeable sur le DCM des petites chaînes, même si la reptation reste le mouvement le plus efficace en ce sens, parmi ceux considérés ici.

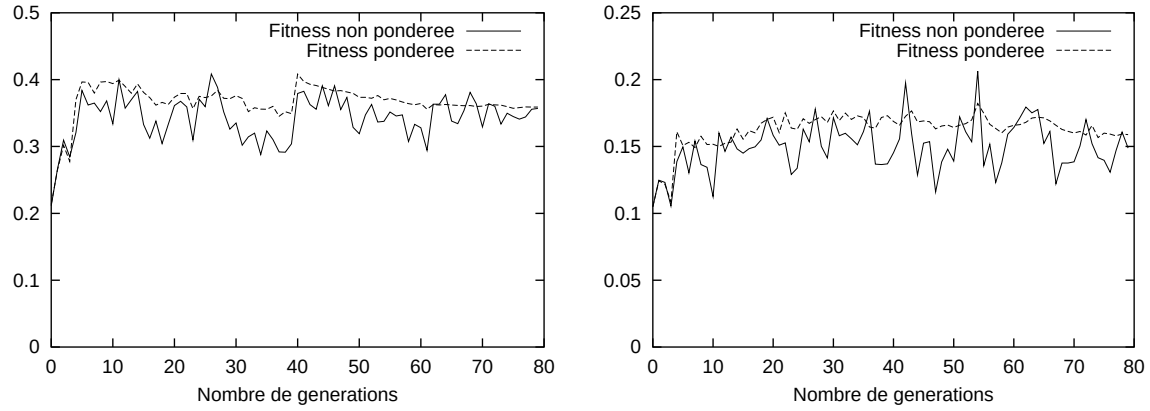


FIG. 6.28 – Séries J32 (gauche) et J64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

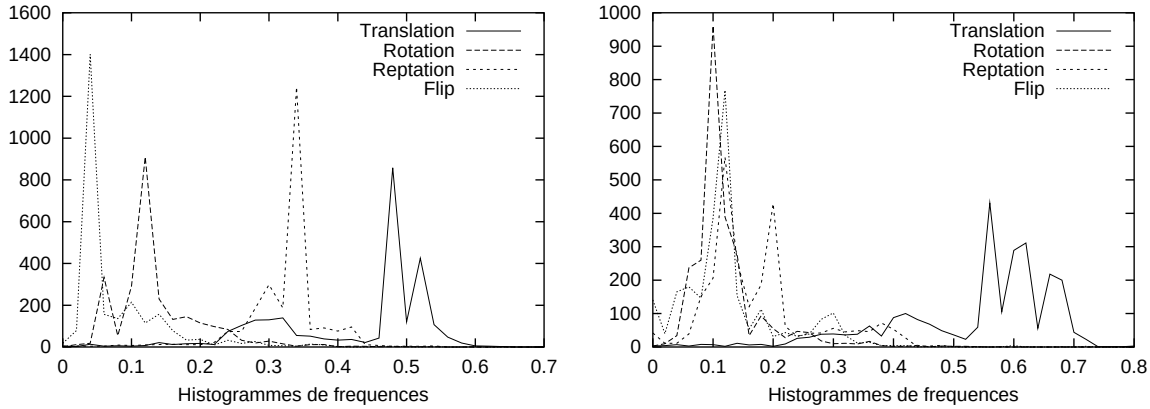


FIG. 6.29 – Séries J32 (gauche) et J64 (droite) : histogrammes des FCE des mouvements.

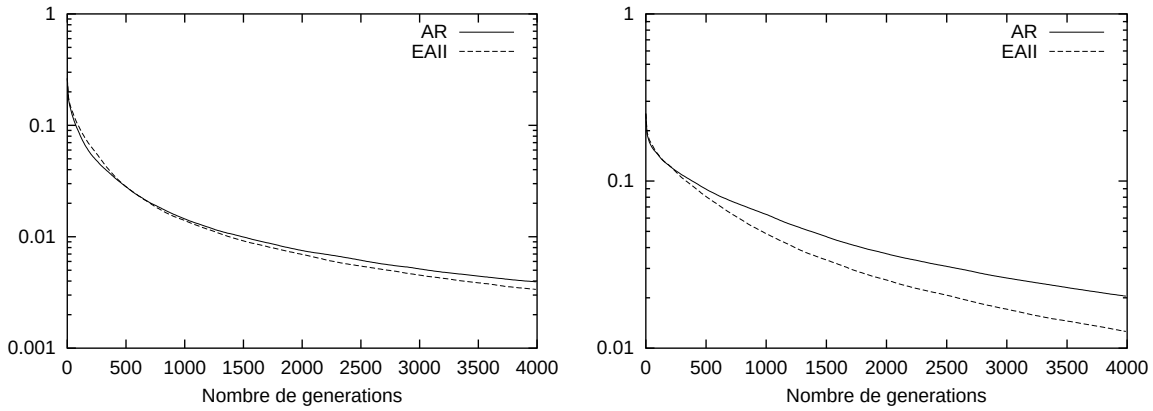


FIG. 6.30 – Séries J32 (gauche) et J64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 6$.

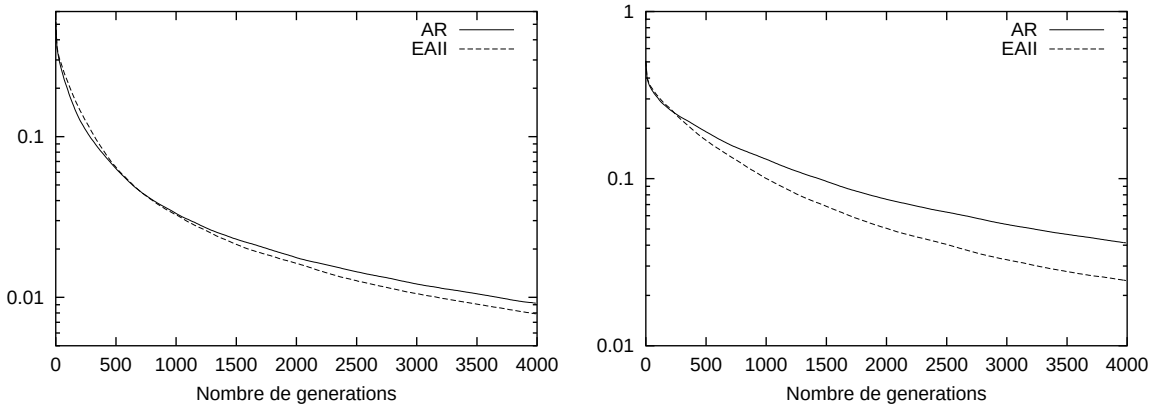


FIG. 6.31 – Séries J32 (gauche) et J64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 7$.

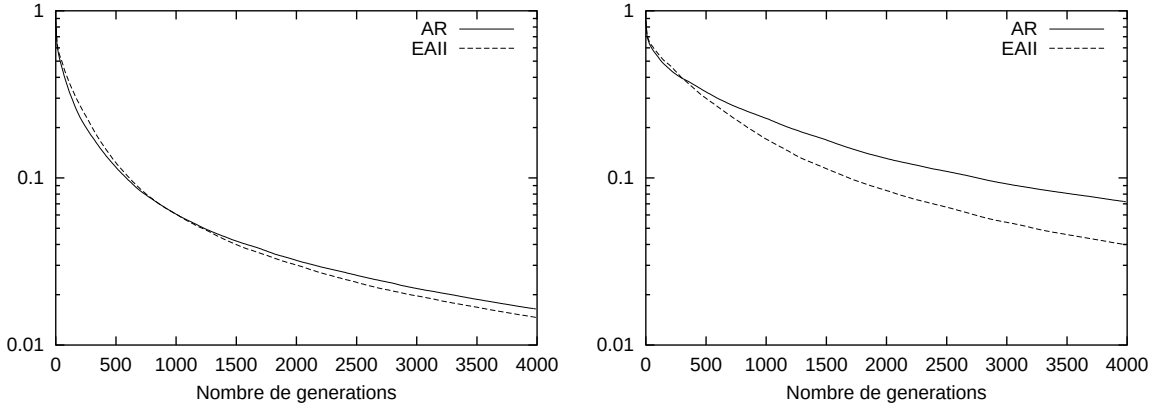


FIG. 6.32 – Séries J32 (gauche) et J64 (droite) : courbes des moyennes (sur les 32 systèmes) de lacunarité des configurations cumulées à l'échelle $R = 8$.

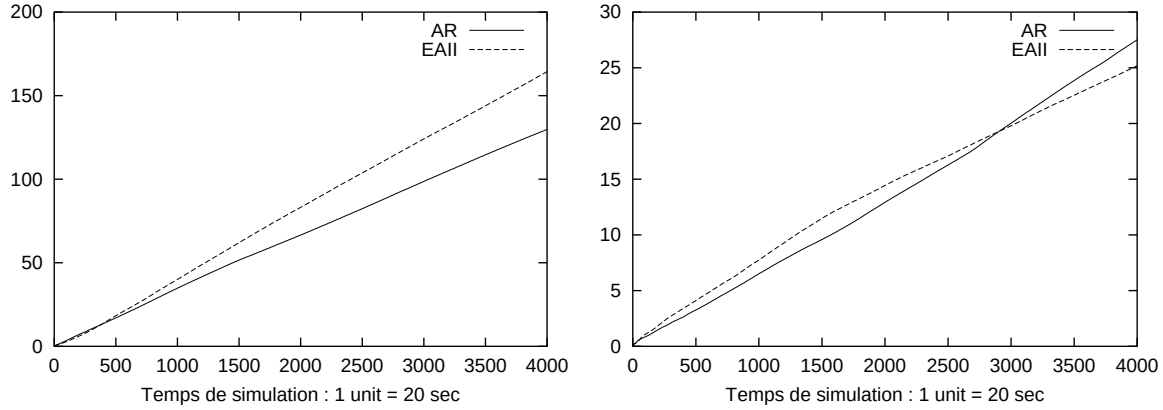


FIG. 6.33 – Séries J32 (gauche) et J64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma^2_{LJ}$).

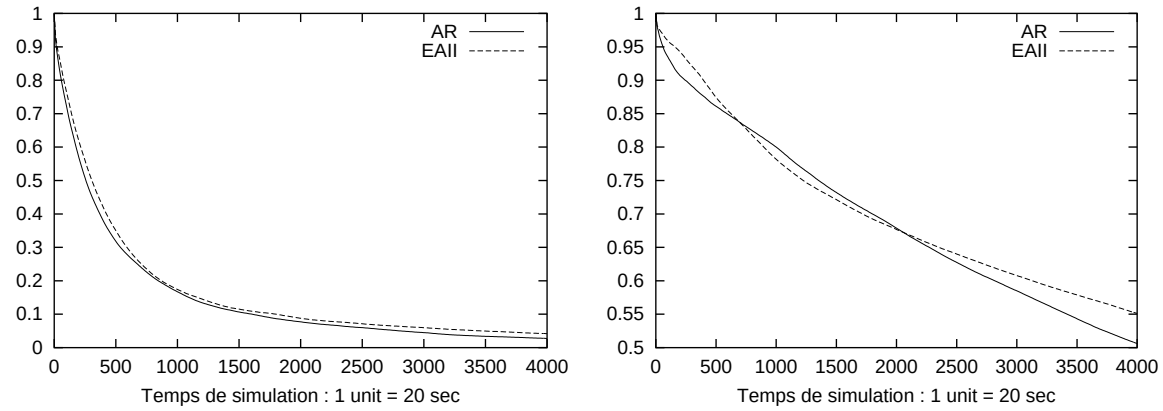


FIG. 6.34 – Séries J32 (gauche) et J64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation cumulée des vecteurs d'orientation des chaînes.

6.7 Reptation par croissance arborescente.

Ayant constaté lors des simulations précédentes l'importance du mouvement de Reptation pour l'efficacité des simulations, nous proposons maintenant une nouvelle version de la reptation, où plusieurs monomères sont simultanément déplacés. Pour ce faire nous adaptons la technique du *Recoil Growth* ([10], [11]), que nous traduirons par "croissance arborescente".

6.7.1 Le mouvement.

Motivations et principe.

Comme nous l'avons vu à la section 2.3.3, le mouvement de recroissance consiste à faire croître une chaîne aléatoirement, avec toutefois une procédure de recherche de positions plus "stables" énergétiquement, induisant un "biais configurationnel" [62]. Toutefois ce mouvement a un taux de succès d'autant plus faible que les chaînes sont longues et que le système est dense. Nous proposons ici d'utiliser le principe de recroissance pour généraliser la reptation à un nombre quelconque de monomères, c'est-à-dire d'enlever une partie de l'extrémité d'une chaîne et de la faire recroître à l'autre extrémité de la chaîne. L'intérêt de cette méthode réside dans le fait que la taux de succès ne dépend plus de la longueur totale de la chaîne mais seulement de la longueur de la "sous-chaîne".

Pour bien le comprendre, on peut approximer le taux de succès d'un mouvement de croissance d'une chaîne de taille L d'après une loi de la forme :

$$P(\text{succès}, L) = p_s^L \tag{6.2}$$

où p_s est une probabilité de succès "atomique", qui est dépendante de la méthode de construction et de la densité du système. Pour des chaînes longues (comprenant L monomères), la reptation classique représente un extrême ou un seul monomère est en jeu, avec un taux de succès de l'ordre de p_s. L'autre extrême est la recroissance complète de la chaîne, qui a certes un effet beaucoup plus important lorsqu'il réussit, mais possède une probabilité de succès qui décroit exponentiellement avec L. L'intérêt de la Reptation par Croissance Arborescente (RCA) est qu'elle permet de trouver un compromis entre la reptation d'un seul monomère et la recroissance d'une chaîne complète, car elle permet d'ajuster la longueur l de la sous-chaîne, contrôlant ainsi plus librement le taux de succès. Il reste maintenant à vérifier que la possibilité de déplacer plusieurs monomères avec un taux de succès plus faible apporte une amélioration concrète à l'efficacité de nos simulations, ce que nous nous proposons de faire dans notre cadre d'expérimentation.

Le fonctionnement.

Il convient d'abord d'expliquer plus en détail la méthode de croissance dite du *Recoil Growth* ([10], [11]), que nous traduirons par "croissance arborescente". Les techniques précédentes de croissance font croître la chaîne linéairement monomère par monomère, en générant plusieurs positions possibles pour le nouveau monomère et en en sélectionnant un par un tirage aléatoire biaisé par l'énergie. L'apport spécifique de la croissance arborescente est d'autoriser un "retour" lorsque la croissance de la chaîne est bloquée : à partir du monomère de bout de chaîne, il devient impossible (très improbable) de générer aléatoirement une position qui soit énergétiquement acceptable, c'est le problème du "cul-de-sac". La croissance arborescente est alors gouvernée par deux paramètres (outre la longueur l) :

– k_{ca} : le nombre de positions d'essai générées pour chaque nouvelle insertion.
– l_{ca} : le nombre de "retours" consécutifs possibles lorsque la croissance est bloquée.

Il faut alors disposer d'un critère effectif pour savoir quand effectuer un retour. Pour cela, on distingue parmi les positions d'essais celles qui sont "ouvertes" et "fermées" en s'appuyant leur facteur de Boltzmann, soit leur énergie. Une position d'essai qui aura une haute énergie (resp. basse) augmentera (resp. diminuera) la probabilité de rejet de la nouvelle configuration, on voudra donc la déclarer "fermée" (resp. "ouverte"). Pour décider si une position d'essai b, d'énergie $u(b)$ est ouverte ou fermée, on effectue alors un tirage selon la loi :

$$P(b \text{ ouverte}) = p^o(b) = min\{1, exp[-\beta u(b)]\} \tag{6.3}$$

Si à partir d'une position sur la chaîne en croissance, toutes les positions d'essais sont fermées on effectue un retour.

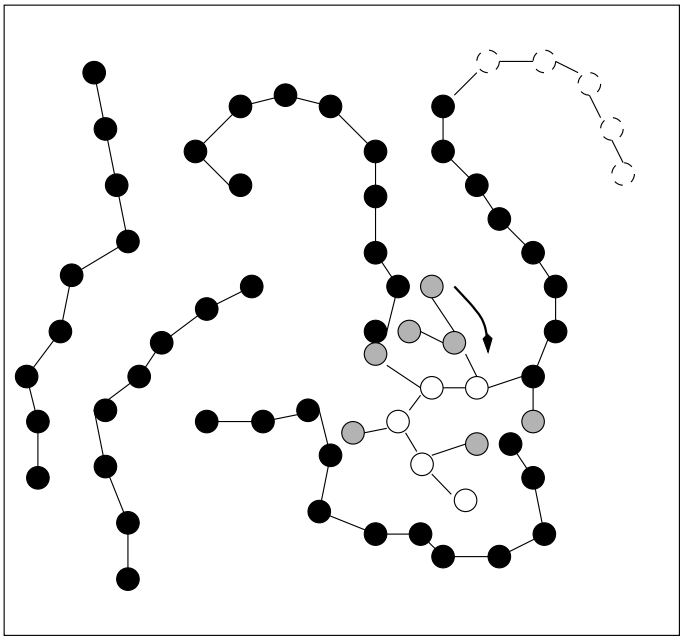


FIG. 6.35 – Reptation par croissance arborescente. La sous-chaîne enlevée est en pointillés, et les nouvelles positions sont représentées par les ronds blancs (positions ouvertes), les ronds gris désignant des positions d'essai fermées. La flèche symbolise un phase de retour en considérant $k_{ca} = 2$.

On peut synthétiser le mouvement de RCA (en se référant aussi à la figure 6.35) de la manière suivante :

1 On choisit aléatoirement un chaîne, et une de ses extrémités. On enlèves alors l monomères de cette extrémité et on commence la recroissance de cette sous-chaîne à l'autre extrémité.
2 Une position d'essai b_i est générée pour le monomère i (i indice sa position dans la sous-chaîne et non dans la chaîne complète). On calcule son énergie u_i et on décide (avec la loi 6.3) si elle est ouverte ou fermée. Si elle est fermée on génère alors une autre position d'essai jusqu'à un maximum de k_{ca} tentatives. Dès que l'on trouve une position ouverte on passe à la phase 3. Si aucune position ouverte n'est trouvée, on effectue une phase de

"retour". La chaîne se rétracte, et on essaie de générer une nouvelle position à partir du monomère $(i-2)$ (s'il est possible de revenir jusque là), pour le monomère $(i-1)$ si les k_{ca} possibilité pour $(i-1)$ ne sont pas toutes déjà utilisées. On peut ainsi effectuer l_{ca} retours consécutifs jusqu'à trouver une position ouverte. Si après ces l_{ca} retours consécutifs, aucune position ouverte n'est trouvée, le mouvement est abandonné.

3 On a trouvé une position ouverte pour le monomère i. Dès ce moment le monomère $i - l_{ca}$ est ajouté définitivement à la nouvelle configuration, le retour ne pouvant plus y ramener. On continue ainsi jusqu'à la croissance complète de la sous-chaîne.

Biais de construction

Une fois qu'une sous-chaîne est ainsi construite il reste à déterminer la probabilité d'accepter le mouvement. La procédure de construction induit évidemment un biais qu'il faut calculer explicitement. Dans ce but, notons $W(n)$ et $W(a)$ les biais respectifs de la nouvelle et de l'ancienne configuration. Pour calculer $W(n)$, on procède comme suit :

1 Supposons que l'on soit au monomère i. Durant la phase de construction, on a pu trouver une position ouverte, faisant partie de la nouvelle configuration. De plus on a pu trouver k_f positions fermées, qui ont été testées mais rejetées au bout l_{ca} étapes. Il reste encore $k_r = (k_{ca} - 1 - k_f)$ possibilités qui n'ont pas été utilisées. Pour chacune d'elles on essaie de faire croître une "branche" de longueur l_{ca} (ou moins si on atteint la profondeur maximale de l'arbre). Remarquons qu'il n'est pas nécessaire d'explorer toutes les possibilités, seul le fait de savoir s'il est possible de trouver au moins une branche importe. Dans ce cas cette direction est dite également ouverte. Pour chaque noeud de la nouvelle configuration, on a ainsi un décompte de positions "ouvertes" que nous noterons o_i. Chaque noeud contribue alors de la manière suivante au biais de construction :

$$wo_i(n) = \frac{o_i(n)}{p^o(b_i(n))} \tag{6.4}$$

où p^o est donné par l'équation 6.3.

2 Répéter le calcul pour $i \in \{1, ..., (l-1)\}$. Pour le dernier monomère considérer $o_l(n) = 1$.

3 Calcul du biais de la construction complète :

$$W(n) = \prod_{i=1}^{l} wo_i(n) = \prod_{i=1}^{l} \frac{o_i(n)}{p^o(b_i(n))} \tag{6.5}$$

Pour le calcul de l'ancienne configuration, on utilise presque la même procédure, la différence étant que l'on doit alors générer $(k_{ca} - 1)$ positions d'essai (en essayant de faire croître une branche de profondeur l_{ca}) pour chacun des monomères. De même on obtient :

$$W(a) = \prod_{i=1}^{l} wo_i(a) = \prod_{i=1}^{l} \frac{o_i(a)}{p^o(b_i(a))} \tag{6.6}$$

et la probabilité d'acceptation devient :

$$acc(a \to n) = min\left\{1, \frac{exp[-\beta U(n)]W(n)}{exp[-\beta U(a)]W(a)}\right\} \tag{6.7}$$

6.7.2 Simulation numériques : série RCA64.

Paramètres.

$$f(\mu,\omega) = c_2(\mu,\omega) = d_c^2(\mu,\omega)$$

Nous avons utilisé les mêmes paramètres de simulation que pour la série G64, en remplaçant le mouvement de translation par le mouvement RCA. Ici nous choisirons $k_{ca} = 2$, correspondant alors à la croissance d'un arbre binaire : ce choix permet de restreindre le nombre de positions à tester (chacune d'elle demandant l'évaluation du potentiel de Lennard-Jones) tout en gardant l'avantage de la recherche arborescente. Nous conditionnons aussi ce mouvement par une longueur maximale de sous-chaîne $l_{sc}^{max} = 4$ et $l_{ca} = 2$, le nombre maximal de retours possibles. Pratiquement à chaque mouvement une longueur l_{sc} de sous-chaine comprise entre l_{ca} et l_{sc}^{max} sera choisie aléatoirement sans biais. Des tests préliminaires nous ont donné des taux de réussite en fonction de l_{sc} l'ordre de :

Longueur l_{sc}	2	3	4
Taux de succès	1.5%	0.3%	0.05%

Ce qui nous indique que la densité de monomères empêche réellement de considérer des sous-chaînes trop longues. Ajoutons pour information que le taux de succès du mouvement de reptation se situe entre 6% et 7%.Toutefois ces taux de succès ne sont pas des indicateurs directs de l'efficacité, ils indiquent simplement qu'un mouvement réussit plus ou moins souvent, il convient toujours de mesurer l'efficacité globale comme nous l'avons fait jusqu'à présent. Le degré du mouvement RCA est plus délicat à déterminer, dans la mesure où il n'implique pas à chaque fois le même nombre d'évaluations de potentiels d'interaction de Lennard Jones. Dans le cas d'un rejet pour le placement du premier monomère de la sous-chaîne, on n'a à effectuer que deux évaluation, si le mouvement réussit complètement, il peut y en avoir plus de 20, pour les plus longues chaînes. Toutefois sachant que le taux de succès reste très faible, nous avons pris le parti de fixer le degré à 3.

Résultats.

On remarque encore sur la figure 6.36 (à gauche) que les valeurs de fitness restent fortement bruitées. Le mouvement de reptation simple reste encore le mouvement privilégié mais le mouvement de RCA n'est pas pour autant disqualifié. Le tableau suivant donne alors les fréquences du meilleur individu ($f_H = 1.024$, $w_H = 891.47$) qui correspond à la combinaison des pics des histogrammes (6.36 à droite).

	RCA	Rotation	Reptation	Flip	FV
FCE hors FV	13.7 %	11.8 %	73.2 %	1.3 %	-
FCE	10.9 %	9.4 %	58.5 %	1.2 %	20 %
FE	5 %	12.9 %	80.5 %	1.6 %	0.0456 %

Nous avons ensuite utilisé cet individu pour refaire une simulation complète uniquement avec ses fréquences que nous avons baptisé "OPT" dans les graphes de résultat. Nous voyons encore une fois que l'EAII fini par trouver un réglage plus efficace en termes d'ACO et de DCM, ce qui se voit plus nettement avec la simulation OPT. En revanche l'ordre des performances est nettement inversé avec le critères LCC à $R = 7$. Pour $R = 3$ cette inversion n'est vraie qu'entre

la simulation AR et la simulation OPT. Enfin si l'on considère avec la série G64, on remarque que le remplacement du mouvement de translation par le mouvement RCA apporte un nette amélioration en DCM et RCA ainsi qu'une nette détérioration en LCC.

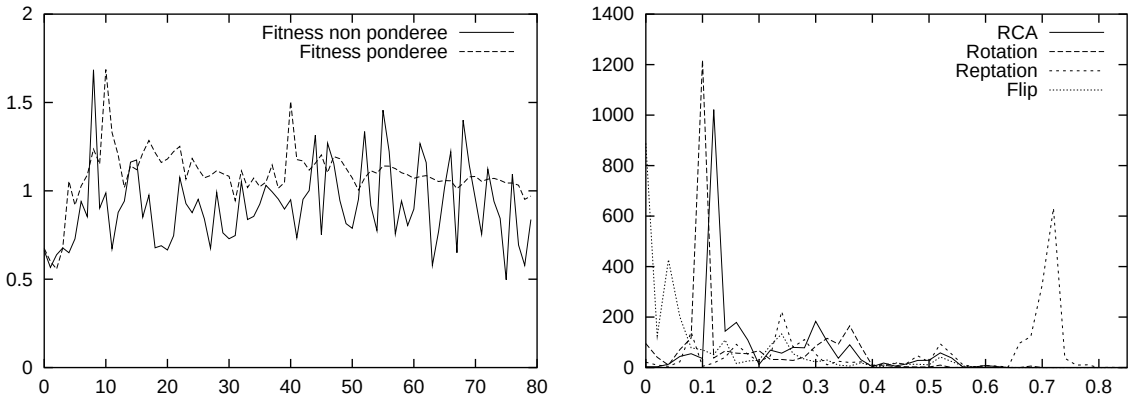


FIG. 6.36 – Série RCA64. A gauche : courbes de fitness pondéré et non pondérées. A droite : histogrammes de fréquences des mouvements

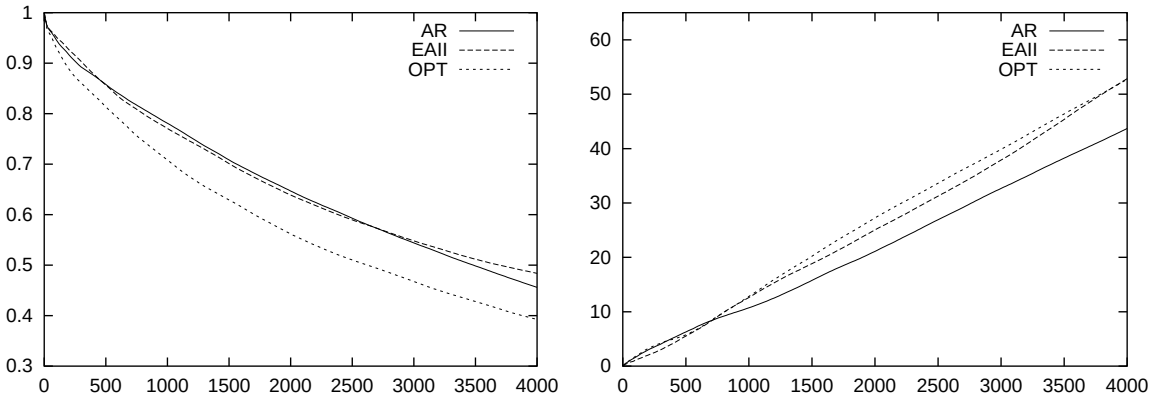


FIG. 6.37 – Série RCA64. Performances globales : ACO à gauche, DCM à droite.

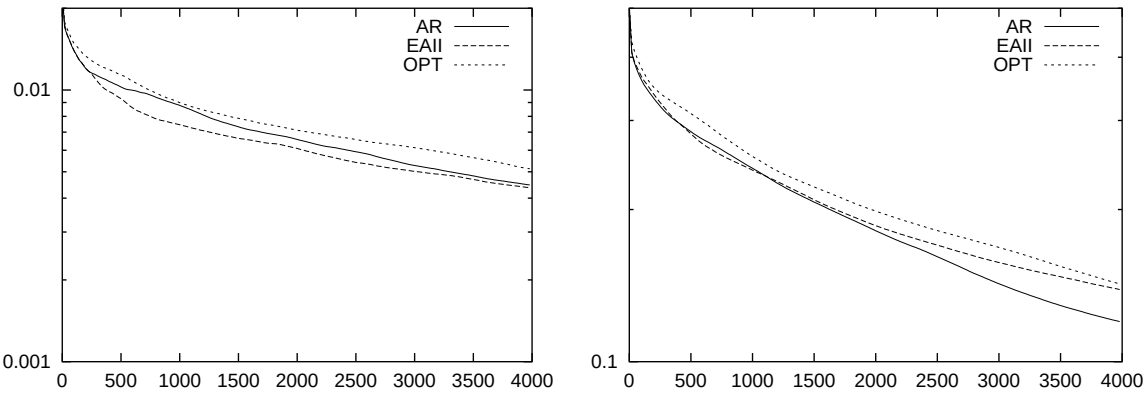


FIG. 6.38 – Série RCA64. Performances globales : LLC $R = 3$ à gauche, LCC $R = 7$ à droite.

6.8 Discussion

L'EAII proposée en section 5.1 a permis de montrer qu'il était possible de tirer parti efficacement de toute l'information produite lors de l'évolution dans les conditions précisées. Sur la base de cet algorithme, on peut appliquer nombre de méthodes d'évolution artificielle, mais les simulations décrites en section 5.2 nous ont permis d'en retenir une combinaison qui semble robuste. Ces améliorations utilisées conjointement à quelques autres du domaines de la simulation moléculaire, nous ont permis d'augmenter plus sensiblement la qualité de nos simulations de polymères amorphes en état dense. Par ailleurs, nos simulations mettant en jeu seulement 4 mouvements de Monte Carlo parmi les plus simples (hormis peut-être la fluctuation de volume), ont permis par la diversité des critères considérés d'étayer l'idée que l'efficacité de telles simulations dépend non seulement de l'efficacité propre de chacun des mouvements, mais surtout de leur bonne combinaison en fonction des conditions de simulations. Pour faire la synthèse de ces observations on peut dire que parmi ces mouvements la reptation est le mouvement "moteur" du mélange, mais qu'il nécessite le concours d'autres mouvements et que la translation peut toujours avoir une certaine utilité (bien qu'avec une fréquence effective toujours petite), même pour des molécules complexes. Le mouvement de Reptation par Croissance Arborecsente, semble être en ce sens une extension prometteuse de la reptation simple. Parmi les critères d'efficacité, nous avons pu observer expérimentalement la corrélation entre la vitesse de déplacement du centre de masse des chaînes et la vitesse de décorrélation des orientations de chaînes. La vitesse de déplacement des monomères apporte peu d'informations supplémentaires et ne constitue pas un critère plus intéressant que le DCM. En revanche les critères de lacunarité semblent être d'une certaine manière en opposition avec les autres, car ils sont plus indicateurs d'une agitation locale des monomères (résultant en de plus grandes fluctuations locales de densité) que d'un changement notable de la configuration des chaînes. Néanmoins cette information n'est sans doute pas à négliger et nous encourage dans l'idée qu'il est souhaitable de juger de l'efficacité selon plusieurs critères et d'appliquer alors des techniques d'optimisation multi-critère.

Chapitre 7

Optimisation de la thermalisation parallèle.

Ayant présenté au chapitre 2, la technique de la thermalisation parallèle (traduction de "parallel tempering"), vantée pour pouvoir très souvent améliorer l'efficacité des simulation de Monte Carlo pour peu qu'on choisisse des paramètres appropriés, nous souhaitons maintenant voir s'il est possible d'utiliser notre EAII pour trouver automatiquement de bonnes valeurs pour ces paramètres. Devant la complexité de cette méthode et le nombre important de paramètres qu'elle met en jeu, nous commencerons par déterminer ceux qu'il est possible d'optimiser. Puis nous présenterons des simulations dont les résultats bien qu'encourageants, confirment l'idée que la prise en compte de tous les aspects d'une telle technique par une méthode d'optimisation reste encore délicate.

7.1 Quels paramètres optimiser ?

Nous avons vu à la section 2.4 que la technique de la thermalisation parallèle implique, une fois les paramètres du système physico-chimique choisis, nécessite encore de spécifier des paramètres qui lui sont particuliers. Par ailleurs, ayant vu de quelle manière ils conditionnent l'efficacité de cette technique, nous discuterons de la possibilité d'optimiser certains d'entre eux par évolution artificielle et de la manière d'adapter nos critères d'efficacité dans ce cadre.

7.1.1 Les températures.

Le choix des températures pose sans doute le problème crucial de cette technique. Rappelons qu'étant donnée T_1, la température la plus basse qui est aussi celle pour laquelle on souhaite obtenir un échantillon, il reste à déterminer leur nombre et leurs valeurs de manière à ce qu'à la fois la plus haute température soit suffisamment haute pour bien mélanger le système, et que les températures intermédiaires permettent des échanges suffisamment fréquents. Ceci afin que chacun des sous-systèmes puissent "visiter" toutes les températures, pour bénéficier de la dynamique plus rapide des hautes températures, et d'apporter une plus grande diversité de configurations à la température la plus basse.

Commençons maintenant par supposer que le nombre des températures n_t est fixé. On peut alors souhaiter optimiser, avec un algorithme évolutionnaire par exemple, les valeurs des températures $(T_i)_{i\in\{2,\ldots,n_t\}}$. Si l'on reprend notre principe de plusieurs systèmes simulés en parallèle (sur lequel repose notre algorithme d'optimisation des fréquences de mouvements présenté aux chapitres 4 et 6), avec des mises à jour des paramètres fournis par l'AE (qui centralise les performances de toutes les simulations pour explorer l'espace paramétrique), les paramètres qui changeront pour un système au début de chaque nouvelle phase d'évaluation seront donc les valeurs de ces températures.

Or si cela n'est d'aucune incidence directe sur l'ensemble statistique Q_1 du sous-système à la température T_1, cela change l'ensemble de chacun des autres sous-systèmes et du système complet $Q = \prod_i^{n_t} Q_i$. Pour autant, on peut toujours se dire que n'étant réellement intéressé que par l'échantillonnage de Q_1, il importe peu que les autres configurations se retrouvent soudainement peu représentatives de l'ensemble. Or le mouvement d'échange des températures se plaçant dans l'ensemble Q, se trouve aussi soudainement être appliqué à une configuration (constitué des n_t sous-configurations) que nous devons considérer d'un point de vue physique en "déséquilibre". D'un point de vue statistique, cela veut dire que la série de configurations qui suivra ne sera pas valide, car ne faisant pas partie des états probables. Or en l'absence de preuve, on ne peut pas affirmer que les mouvements d'échange entre les sous-configurations à T_1 et les autres (non valides statistiquement) produiront toujours des sous-configurations valides pour Q_1.

Une possibilité cependant, reste d'effectuer une phase "d'équilibration" après un tel changement. Autrement dit, cela revient à ne pas retenir les configurations pour l'échantillonnage jusqu'à ce que tous les sous-systèmes soient équilibrés. Malheureusement ce temps de rééquilibration peut se révéler difficile à déterminer, et surtout trop important en regard des périodes de changement de températures.

Finalement, on voit que si ce type d'optimisation est souhaitable, sa réalisation pose encore d'importants problèmes de justifications théoriques et de mise en pratique.

7.1.2 Les fréquences des sous-systèmes.

Nous avons également vu que la thermalisation parallèle nécessitait de choisir aléatoirement à chaque pas de simulation un des sous-systèmes suivant une loi fixe, puis de lui appliquer un mouvement "interne" (tout mouvement n'étant pas un échange de température), sous peine de briser la règle de micro-réversibilité du mouvement de Monte Carlo, nécessaire pour garantir la validité de l'échantillonnage.

Toutefois on peut encore décider de modifier de temps en temps ce que nous appellerons les fréquences des sous-sytèmes $(\lambda_i)_{i\in\{1,\dots,n_t\}}$, pour peu que ces modifications soient très rares. Nous avons vu que généralement, il est conseillé d'affecter la plus grande fréquence à la température la plus basse, puis des fréquences décroissantes en fonction de la température (technique conseillée dans [18] et généralement suivie).

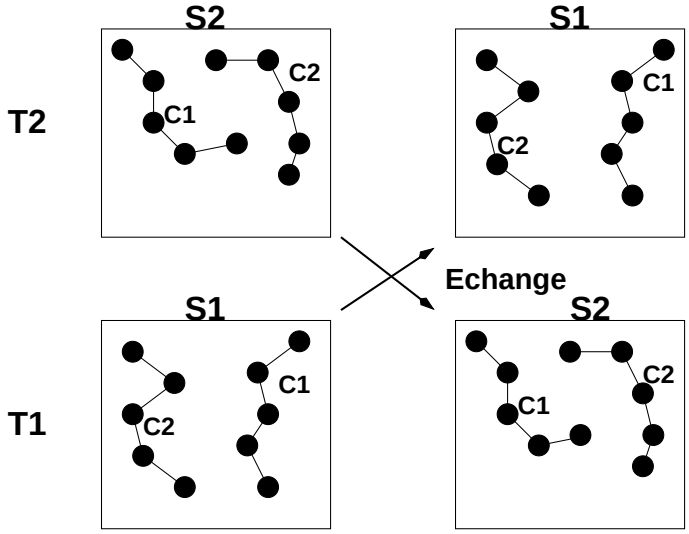


FIG. 7.1 – Mouvement d'échange entre deux sous-sytèmes : $(s_1, T_1), (s_2, T_2) \to (s_1, T_2)(s_2, T_1)$.

Nous proposons dans un premier temps, une fois les températures $(T_i)_{i\in\{1,\dots,n_t\}}$ fixées, d'adapter notre EAII pour optimiser, non plus le vecteur μ des fréquences des mouvements, mais le vecteur λ des fréquences des sous-sytèmes. L'algorithme décrit au chapitre 6 est alors transposable en considérant que les individus codent ces fréquences, mais en prenant garde d'adapter convenablement les critères d'efficacité. Nous avons vu au chapitre précédent que les critères d'autocorrélation des orientations des chaînes et de déplacement des centres de masse sont les plus intéressants au regard des simulations que nous avons faites. Dans le cas présent, il faut toutefois adapter ces critères : seules les configurations à la température T_1 nous intéressant, il est souhaitable de les mesurer uniquement pour cette température. Or si un mouvement d'échange (voir figure 7.1) intervient entre les sous-sytèmes s_1 et s_2 aux températures T_1 et T_2, les références des positions et des orientations des chaînes de s_1 avant l'échange n'auront plus de sens lorque s_2 sera à la température T_1. Il faut donc considérer, au début d'une phase d'évaluation, les références des chaînes pour chacun des sous-systèmes $(s_i)_{i\in\{1,\dots,n_t\}}$, et se servir de la référence appropriée lors du calcul du déplacement et des orientations des chaînes à la température T_1. Cela implique que les courbes de déplacement des centres de masse et d'autocorrélation des orientations changeront brusquement à chaque échange de température. En revanche l'informa-

tion résultante sera effectivement repésentative de l'ensemble du déplacement des chaînes de tous les systèmes, lorqu'ils repassent à la température T_1. De cette manière on a bien des mesures cohérentes du mélange de chacun des sous-systèmes.

7.2 Optimisation des fréquences des sous-sytèmes : série K

7.2.1 Paramètres de simulation.

Nous nous placerons encore une fois dans la cadre de la simulation de polyéthylène, suivant le modèle moléculaire décrit à la section 2.3.1 dans l'ensemble $Q = \prod_{i=1}^{n_t} nNPT_i$. Les paramètres physico-chimiques, à l'exception des températures supplémentaires, sont identiques à ceux des séries des chaînes de taille 32 présentées au chapitre 6. Nous considérerons $n_t = 5$ températures, suivant la loi $T_{(i+1)} = 1.1 \times T$:

- $n = 640$ monomères $CH2$ ou $CH3$.
- $N = 20$ chaînes. Les chaînes comptent alors 32 ou 64 monomères.
- $P = 1atm$
- $T_1 = 450\ K$, $T_2 = 495\ K$, $T_3 = 544\ K$, $T_4 = 599\ K$, $T_5 = 659\ K$
- $n_c = 1600000$ cycles de simulations (1 cycle = 1 seconde de simulation sur un Pentium III 733 Mhz du cluster de l'INRIA Grenoble), soit deux fois plus que pour les simulations du chapitre 6.

Les mouvements utilisés (voir la section 2.3.3) sont :

- La Translation ($deg = \frac{n}{N}$)
- La Rotation ($deg = 1$)
- La Reptation ($deg = 1$)
- Le Flip ($deg = 1$)
- L'échange. Bien que ce mouvement soit très rapide car il n'implique pas de perturbations des énergies d'interactions des sous-sytèmes concernés, nous lui attribuons un dégré de 1 pour rester cohérent avec le concept des fréquences à charges éguale
- La Fluctuation de Volume ($deg = n$).

Nous attribuons des FCE de 18% au 5 premiers mouvements et une FCE de 10% à la fluctuation de volume, ce qui se traduit par les FE suivantes :

Translation	Rotation	Reptation	Flip	Echange	Fluctuation volumique
36 / 4645	1152 / 4645	1152 / 4645	1152 / 4645	1152 / 4645	1 / 4645

Nous effectuerons donc deux simulations, une suivant un algorithme de référence (notée AR), pour laquelle nous fixons le vecteur λ comme suit :

$$\lambda_{i+1} = 0.5 * \lambda_i \quad \forall i > 1 \quad , \quad \lambda_{ref} = \left\{ \frac{16}{31}, \frac{8}{31}, \frac{4}{31}, \frac{2}{31}, \frac{1}{31} \right\} \tag{7.1}$$

L'autre se faisant grâce à l'EAII, où un individu codera un vecteur λ, et en utilisant comme fonction de fitness le critère d'autocorrélation cumulée des vecteurs d'orientation des chaînes (pour lesquelles on utilises les 5 références pour les 5 sous-systèmes) :

$$f(\lambda, \omega) = c_1(\lambda, \omega) = (1 - C_v^n(\lambda, \omega)) \tag{7.2}$$

Les paramètres de l'EAII sont toujours :

- Sélection par seuillage (section 5.1.2), avec $c_s = 0.8,\ n_t = 10$.
- Croisement arithmétique avec restriction de voisinage : $p_c = 0.8$.
- Mutation gaussienne : $p_m = 0.05$.
- Rayon de voisinage $\sigma_\infty = 0.05$.
- Nombre de cycles de simulation : $n_c = 80000$
- Nombre total d'évaluations : 2560, soit 32 systèmes simulés avec 80 subdivisions du temps de simulation. La temps d'évaluation est donc de 4000 secondes.
- Taux d'exploration de 0.5 : 1280 individus créés pour l'exploration, 1280 pour l'exploitation.

Enfin précisons qu'afin d'éviter que certaines des fréquences λ_i deviennent négligeables, rendant inopérant le principe de la thermalisation parallèle, nous les contraignons à rester supérieures à un seuil que nous fixons à 2%. L'espace de recherche est alors de la forme :

$$S_\lambda = \left\{ \lambda \in \prod_{i=1}^{n_t}]0.02, 1[\ ; \ \sum_{i=1}^{n_t} \lambda_i = 1 \right\} \tag{7.3}$$

7.2.2 Résultats

La figure 7.3 montre que l'EAII parvient encore à améliorer le critère c_1 adapté à la thermalisation parallèle. Si l'on regarde les courbes des déplacements cumulés des centres de masse des chaînes (DCM) (figure 7.4, gauche), on voit que l'EAII prend un retard plus important en début de simulation, mais qu'il finit par trouver un jeu de fréquences λ plus efficaces que l'AR : la pente sur la deuxième moitié de la courbe est nettement en faveur de l'EAII. Il est intéressant d'observer l'impact des échanges des systèmes sur la courbe des déplacements (non cumulés) des centres de masse des chaînes (figure 7.4, droite), où l'on voit l'effet des échanges : de brusques variations qui vont en s'amplifiant. Ces fluctuations restent moins importantes concernant l'AR, suggérant qu'avec la distribution de référence, les systèmes font des aller-retours très réguliers, ce qui ne semble pas être un avantage. Signalons aussi que les taux d'acceptation des mouvements d'échange varient dans les deux cas de 2% à 3%, indiquant un bon taux taux d'échange entre systèmes voisins.

Regardons maintenant les histogrammes de toutes les valeurs des λ_i créées par l'EAII sur la figures 7.5. Sans surprise, la valeur la plus élevée est attribuée au système le plus froid. Par contre si on regarde le meilleur individu trouvé (voir la figure 7.2 pour une comparaison avec λ_{ref}), il diffère tout de même sensiblement de la référence :

$$\lambda^* = \{0.578, 0.053, 0.062, 0.14, 0.167\} \quad , \quad w(\lambda^*) = 362 \tag{7.4}$$

On voit donc que les températures les plus chaudes sont plus favorisées, ce qui suggère que l'efficacité vient du fait qu'il faut non seulement privilégier en premier la température d'échantillonnage mais aussi consacrer un effort important aux températures les plus chaudes, car ce sont celles qui mélangent le plus efficacement les sous-systèmes.

Enfin nous définissons pour chacun des sous-systèmes $(s_i)_{i\in\{1,\dots,n_t\}}$ des *indices d'occupation* des températures : pour chaque sous-système, on définit un vecteur de compteurs de température $CT_{i,j}$, i désignant le sous-système, j désignant la température. Chaque fois qu'un échange entre deux sous-systèmes à des températures voisines réussit :

$$(s_i, T_k), (s_j, T_{k+1}) \rightarrow (s_j, T_k), (s_i, T_{k+1})$$

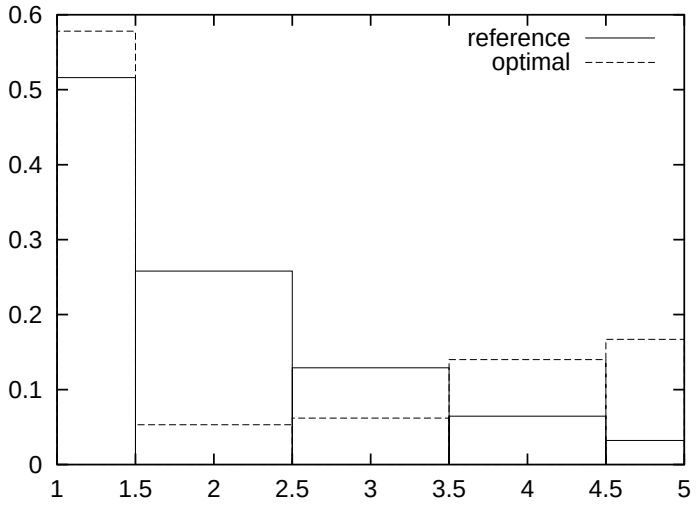


FIG. 7.2 – Séries K : λ^* versus λ_{ref}.

on incrémente les compteurs $CT_{j,k}$ et $CT_{i,k+1}$. On définit les indices d'occupation, calculés en fin de simulation :

$$\theta_{i,j} = \frac{CT_{i,j}}{\sum_{j=1}^{n_t} CT_{i,j}} \tag{7.5}$$

On a ainsi une idée, pour chaque sous-système, des proportions de ses changements des températures[1]. La figure 7.6 en montre alors les courbes pour chacun des sous-systèmes, qui sont indicés à partir de leur température en début de simulation : à $t = 0$ on a (s_1, T_1), ..., (s_{n_t}, T_{n_t}). On voit alors que tous les sous-systèmes sont passés par toutes les températures. De plus on voit que pour ceux qui sont initialement les plus froids ces indices sont plus importants aux températures froides. Par contre ils sont bien répartis pour ceux ayant commencé aux températures chaudes. On pourrait éventuellement trouver anormal ce déséquilibre, mais il faut se rappeler que ces indices sont normalisés et que les facteurs normalisateurs se sont pas tous égaux (précisément du fait d'une distribution non uniforme des λ_i).

[1] Qu'il ne faut pas confondre avec la proportion du temps, comptés en nombre de cycles, passé à aux différentes températures.

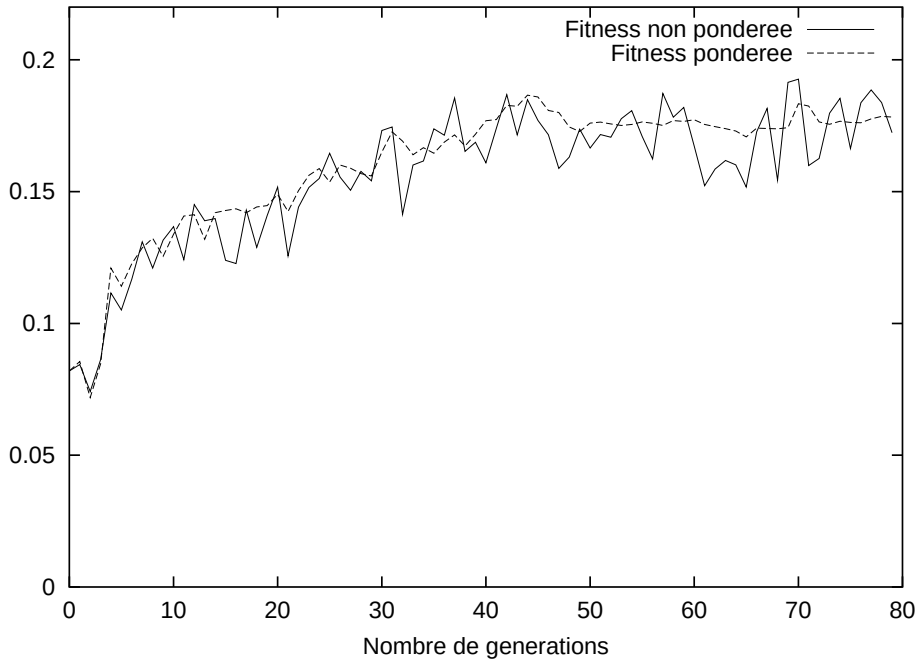


FIG. 7.3 – Séries K : courbes de fitness moyenne pondérée et non pondéré.

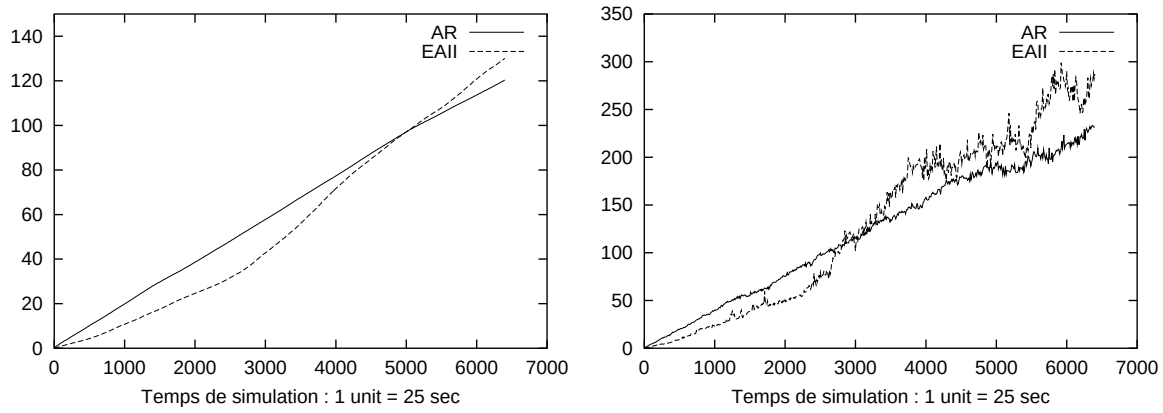


FIG. 7.4 – Séries K : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés (gauche) et non cumulés (droite) des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

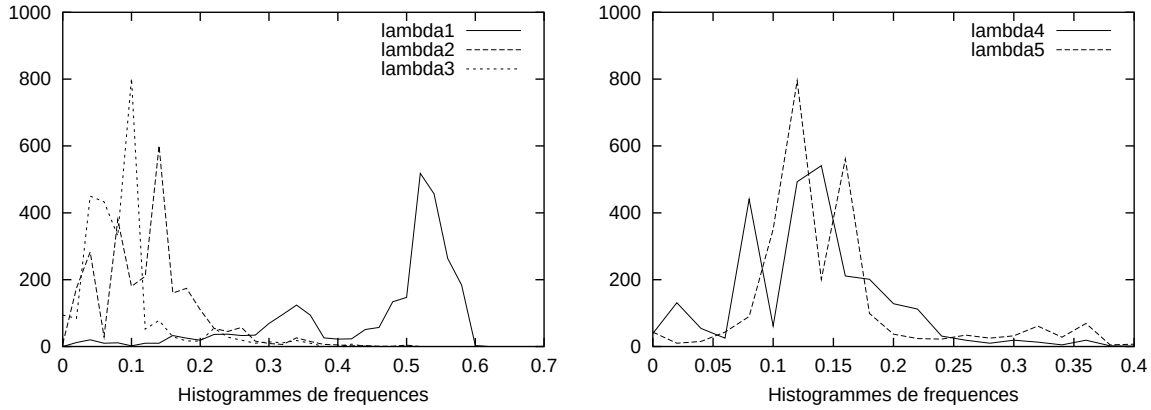


FIG. 7.5 – Séries K : histogrammes de fréquences

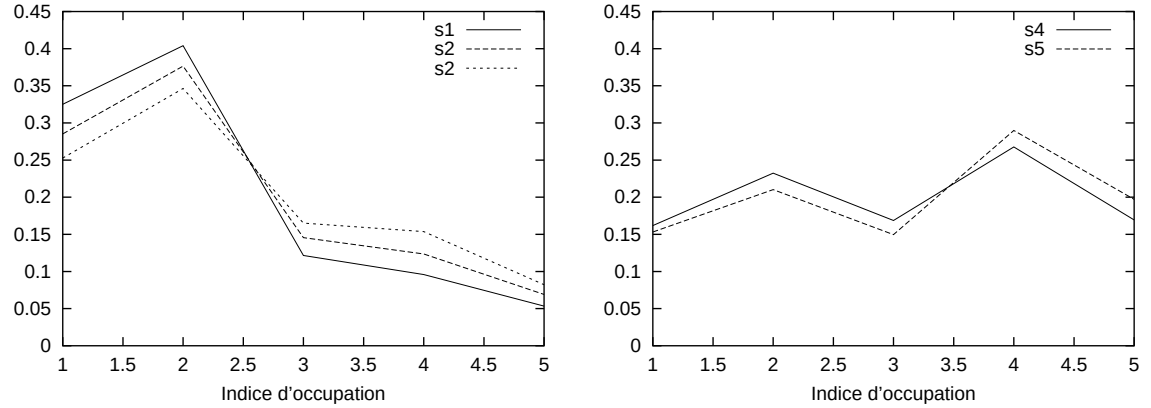


FIG. 7.6 – Séries K : indices d'occupation.

7.3 Optimisation des fréquences des mouvements : L32 et L64

Toujours dans le cadre de la thermalisation parallèle, nous souhaitons optimiser à nouveau les fréquences μ des mouvements, mais cette fois en utilisant les fréquences des sous-systèmes λ obtenus par l'optimisation présentée à la section précédente.

7.3.1 Paramètres de simulation.

Les paramètres physico-chimiques sont identiques à ceux de la section précédente, à ceci près que l'on considère, comme au chapitre 6, des chaînes de taille 32 (série L32) et des chaînes de taille 64 (série L64).

Concernant l'EAII, la différence par rapport à la série K, est évidement l'optimisation des fréquences μ de mouvements au lieu des fréquences λ des sous-systèmes. Pour ces dernières, en se basant sur les résultats de la série K, nous fixons maintenant :

$$\lambda = \{0.6, 0.05, 0.05, 0.15, 0.15\} \tag{7.6}$$

Nous conservons les mêmes paramètres pour l'algorithme de référence.

7.3.2 Résultats

En regardant les courbes de fitness (figure 7.7), on voit que l'EAII est encore capable de trouver de bons individus pour les chaînes courtes, mais que le cas des chaînes longues est plus problématique. L'écart observé entre les valeurs de fitness pondérées et non pondérées est d'ailleurs encore plus important que pour la série F64 par exemple. Si on regarde la courbes de performances ACO de la série L64 (figure 7.12), on voit qu'il y a tout de même une amélioration par rapport à l'AR, mais concernant les performances DCM (figure 7.10), les résultats sont plutôt comparables. En revanche les résultats de DCM pour la série L32, montrent une bien meilleure amélioration : le déplacement est quasiment deux fois plus important. En comparant au déplacement observé pour la série F32 (sans thermalisation parallèle, mais pour un temps de simulation deux fois plus court), on passe d'un DCM final de 180 à 220. Même si le temps de simulation est ici deux fois plus long, il ne faut pas oublier que la DCM dans le cas présent est représentative du déplacement des chaînes appartenant à 5 systèmes (même s'il n'en est pas la moyenne exacte), prouvant que l'effort de simulation supplémentaire imputé au parrallel temepring se révèle bénéfique en termes de nombre de configurations décorrélées.

Enfin en observant les indices d'occupation (figure 7.13 et 7.14, définis en section 7.2.2), on voit que si dans le cas des chaînes courtes tous les systèmes visitent bien toutes les températures, dans le cas des chaînes longues, les sous-systèmes se diffusent beaucoup moins facilement. On peut alors supposer que la dynamique des chaînes longues étant plus lente, il faudrait simuler plus longtemps ou encore choisir une échelle de températures plus proches pour remédier à cette lenteur de diffusion. On peut aussi supposer que les résultats de la série K ayant été obtenus en simulant des chaînes courtes, le vecteur de fréquences λ utilisé ne soit pas aussi efficace pour des chaînes plus longues.

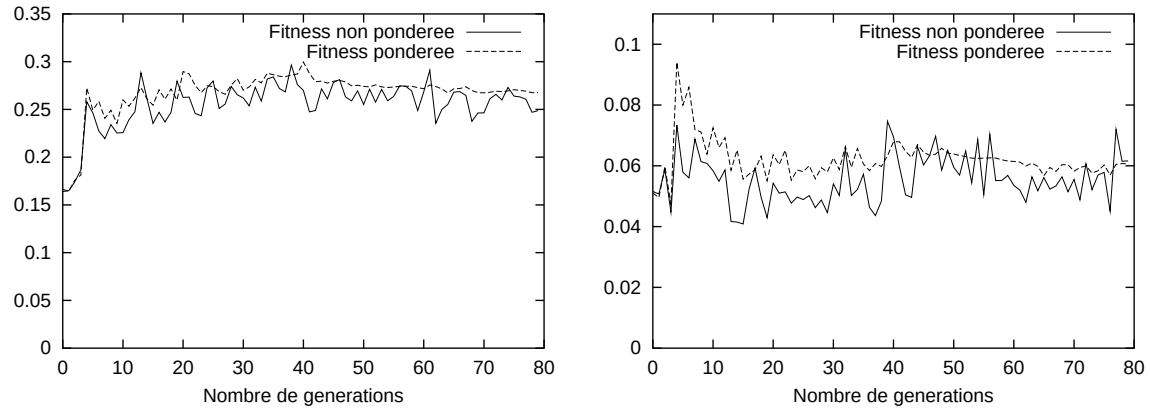


FIG. 7.7 – Séries L32 (gauche) et L64 (droite) : courbes de fitness moyenne pondérée et non pondéré.

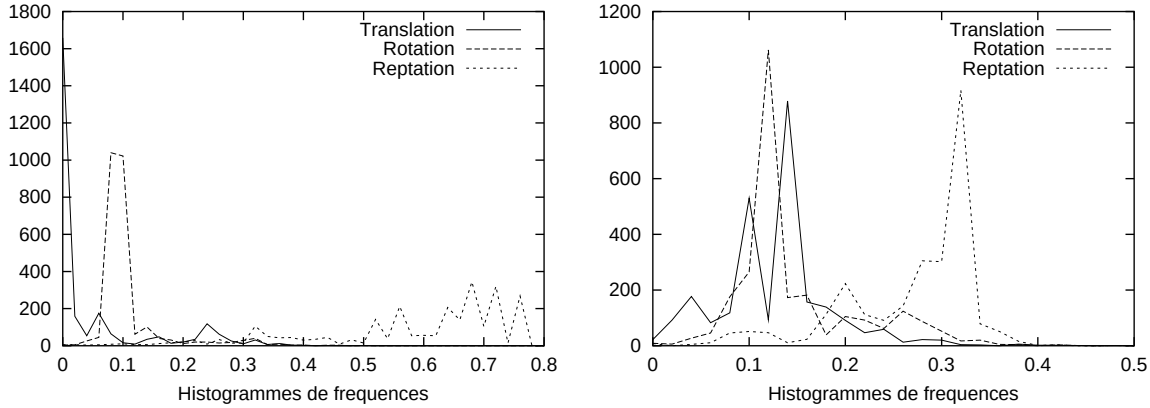


FIG. 7.8 – Séries L32 (gauche) et L64 (droite) : histogrammes des FCE des mouvements Translation, Rotation et Reptation.

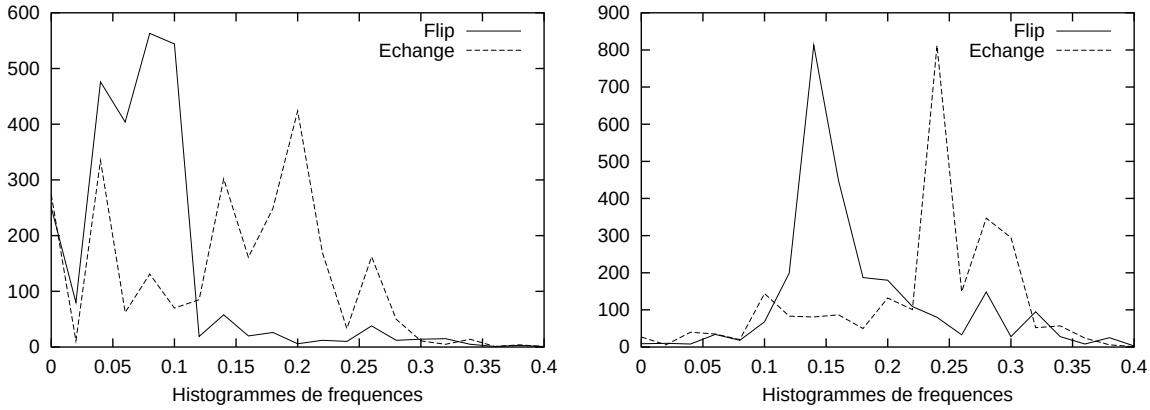


FIG. 7.9 – Séries L32 (gauche) et L64 (droite) : histogrammes des FCE des mouvements Flip et Echange.

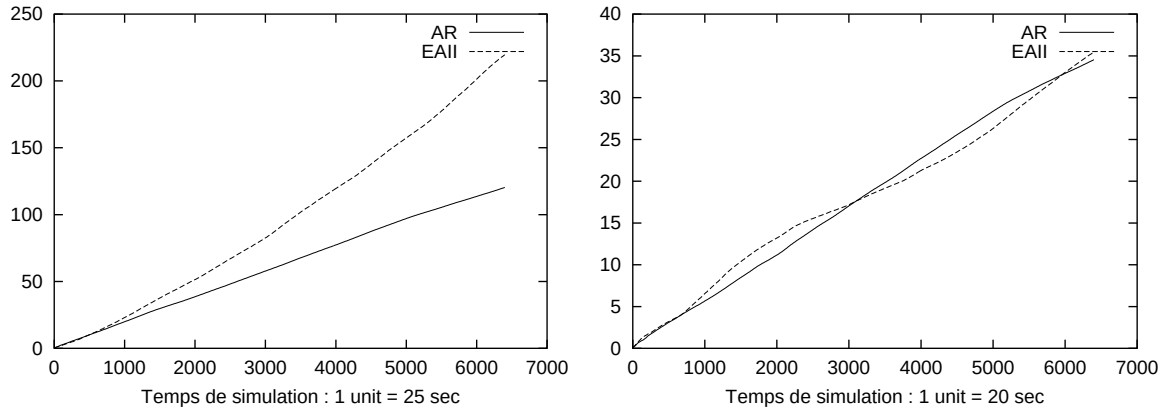


FIG. 7.10 – Séries L32 (gauche) et L64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés cumulés des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

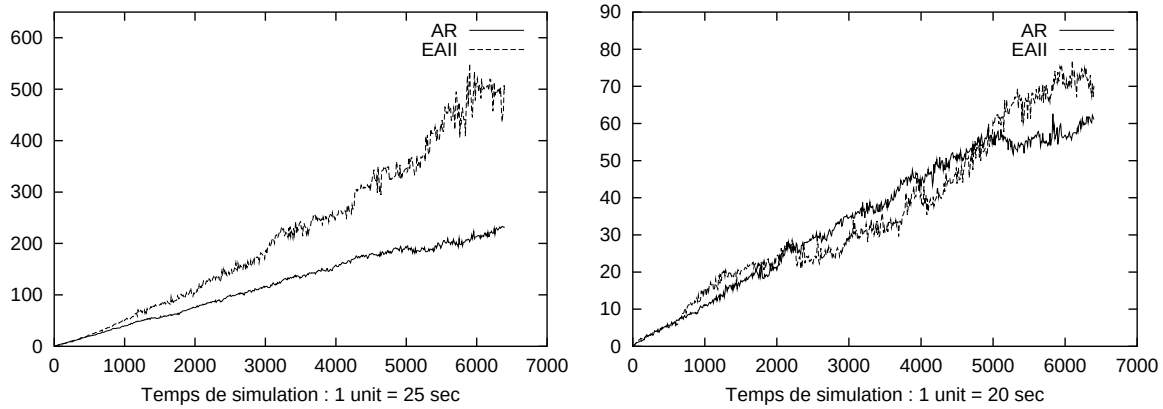


FIG. 7.11 – Séries L32 (gauche) et L64 (droite) : courbes des moyennes (sur les 32 systèmes) des déplacements carrés **(non cumulés)** des centres de masse des chaînes. Les ordonnées sont en unités réduites (1 unité $= \sigma_{LJ}^2$).

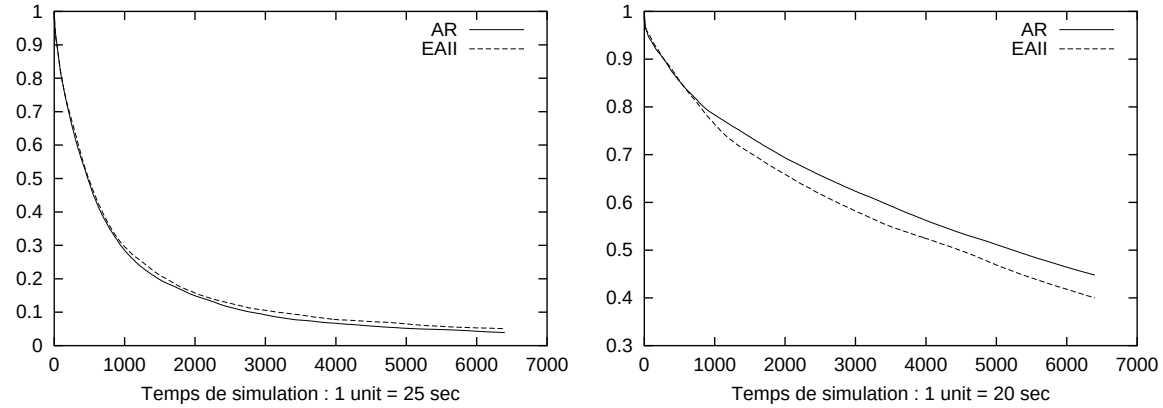


FIG. 7.12 – Séries L32 (gauche) et L64 (droite) : courbes des moyennes (sur les 32 systèmes) d'autocorrélation instantanée des vecteurs d'orientation des chaînes.

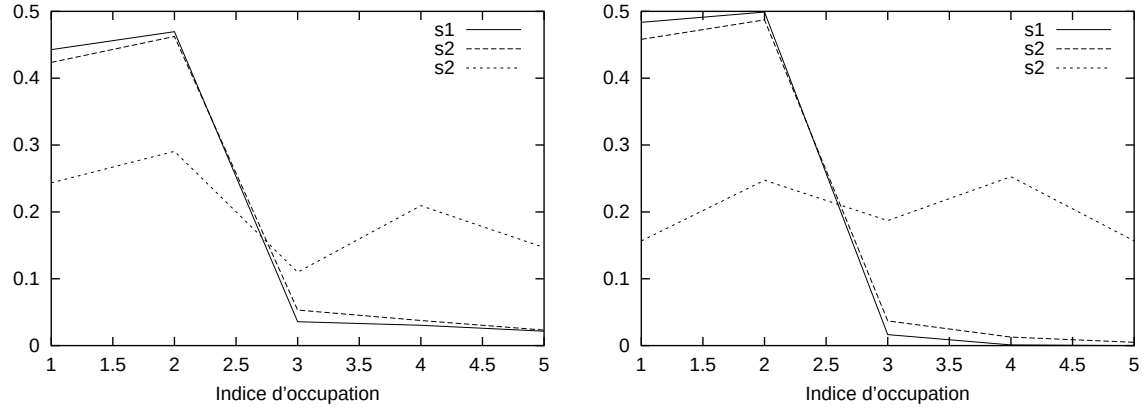


FIG. 7.13 – Séries L32 (gauche) et L64 (droite) : Indices d'occupation des sous-systèmes s_1, s_2, s_3.

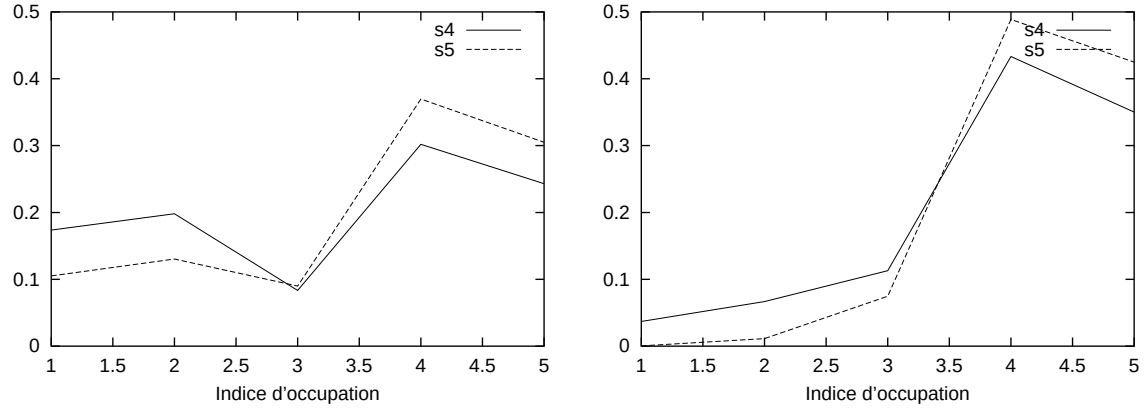


FIG. 7.14 – Séries L32 (gauche) et L64 (droite) : Indices d'occupation des sous-systèmes s_4, s_5.

7.4 Discussion.

Nous venons de voir avec les séries de simulations K et L, qu'il est possible d'appliquer avec succès l'EAII pour améliorer l'efficacité statistique de simulations moléculaires, tout en utilisant la thermalisation parallèle. Nous avons pu de cette manière optimiser successivement les fréquences de sous-systèmes et des mouvements de Monte Carlo eux-mêmes. Le résulat principal à tirer de ces simulations est alors la distribution des fréquences des sous-systèmes dont la répartition optimale, nuançant la recommandation habituelle pour cette technique, n'est pas strictement décroissante en fonction de la température, même si la plus forte probabilité est attribuée au sous-système le plus froid.

En revanche s'il nous a été possible d'obtenir des résultats satisfaisants, le nombre de paramètres devenant beaucoup plus important, il devient très difficile de les optimiser tous en même temps avec un nombre restreint d'évaluations pour l'EAII. Pour en effectuer plus, sans réduire la durée de simulation d'une phase d'évaluation, il faut nécessairement effectuer des simulations beaucoup plus longues. On a vu d'autre part que la dynamique de diffusion des sous-systèmes à travers les températures pouvant se révéler relativement lente, on ne peut pas considérer des périodes d'évaluation trop courtes, si on veut pouvoir juger sur ces durées de l'efficacité de la thermalisation parallèle en elle-même.

Finalement nous dirons qu'il est souhaitable d'arriver à optimiser simultanément les températures utilisées, les fréquences des sous-systèmes et les fréquences des mouvements, car nous avons vu qu'il existe un interdépendance entre elles. Enfin il reste non seulement des obstacles théoriques (distributions des température), mais aussi pratiques pour parvenir à cette fin. Rien n'indique cependant qu'ils soient insurmontables.

Chapitre 8

Conclusions.

8.1 Rappel du problème.

Nous nous sommes confrontés dans cette thèse au problème de l'efficacité statistique des simulations moléculaires de Monte Carlo, qui se pose essentiellement pour des molécules complexes, au nombre desquelles les polymères amorphes en état dense. Dans ce cas l'efficacité désigne l'aptitude à fournir en un temps de calcul fini, un échantillon statistiquement valide afin qu'ils puisse servir à donner des estimations fiables des propriétés du système physico-chimique simulé. Nous avons vu que cela revenait à fournir une suite de configurations qui soient les plus décorrélées possible et que ce concept, intuitivement simple, n'est pas évident à mesurer explicitement. Nous avons donc déterminé des critères numériques représentatifs de cette efficacité, applicables à la simulation des polymères. Il nous a encore fallu identifier des paramètres "libres" de ces simulations, c'est-à-dire qui ne soient pas imposés par les conditions de simulations et dont le changement occasionnel en cours de simulation n'invalide pas celle-ci. Nous avons voulu transformer le choix de ces paramètres en atout plutôt qu'en embarras. Ayant alors des paramètres à optimiser et des critères d'optimisation nous avons pu appliquer les algorithmes d'EA, ayant rappelé par ailleurs que si leur application est rarement triviale, ils offrent souvent la possibilité de venir à bout de problèmes d'optimisation pour lesquels on ne dispose pas d'algorithme simple et efficace.

8.2 Principaux résultats.

Nous insisterons d'abord sur la manière dont nous avons combiné des simulations parallèles avec un "serveur" génétique (chapitres 4 et 6) afin d'effectuer un échantillonnage tout en optimisant simultanément les fréquences des mouvements de Monte Carlo. Ainsi, il n'y pas de temps perdu pour l'optimisation proprement dite, car elle se fait en cours de simulation même. C'est à ce titre que l'on peut qualifier notre algorithme d'outil de contrôle adaptatif, car il a pour but d'améliorer les performances d'un système en cours de simulation et non de trouver un optimum global a posteriori. Essayer d'atteindre ce but serait d'ailleurs sans doute illusoire étant donné la nature aléatoire des simulations, et donc la nature bruitée des évaluations de performances que l'on en fait. A partir des conditions particulières de l'application de notre AE, nous avons introduit au chapitre 5 une nouvelle technique évolutionnaire (l'Evolution Artificielle d'Individus Immortels : EAII) qui s'est révélée être robuste pour les problèmes à fonction de fitness bruitée

et coûteuse.

Revenant à la nature des critères de performances nous avons introduit l'utilisation de la lacunarité au chapitre 3, comme mesure de la vitesse d'uniformisation de densité locale. De manière intéressante ce nouveau critère apparaît décorrélé des autres critères (orientation et déplacement des chaînes), apportant une information supplémentaire pour juger de l'efficacité de ces simulations.

Enfin dans le chapitre 7, nous avons montré qu'il est aussi possible d'utiliser cet algorithme pour tirer profit de la thermalisation parallèle et même d'en améliorer l'usage. De ce point de vue nous avons pu montrer qu'une distribution strictement décroissante des fréquences des sous-systèmes, solution habituellement recommandée, n'est pas optimale.

8.3 Intérêt pour l'évolution artificielle.

La technique de l'EAII constitue un apport évident car elle s'inscrit comme une nouvelle méthode pour tenir compte du bruit d'évaluation en EA. Ajoutons que son adaptation à notre problème de simulation moléculaire nous a permis aussitôt de l'appliquer à un problème concret. Mais nous insisterons surtout sur le fait que notre application a montré une nouvelle fois que l'EA n'est pas seulement vouée à la stricte optimisation de fonction, où la seule localisation d'un optimum global est visée, mais se révèle aussi un excellent outil de contrôle, utilisé dans nombre de domaines (citons par exemple la vision robotique 3D par "mouches évolutionnaires" [6]).

8.4 Intérêt pour la simulation moléculaire.

Outre l'intérêt précédemment évoqué de notre critère de lacunarité et de notre optimisation de la thermalisation parallèle, nous avons apporté dans cette thèse un cadre unifié pour pouvoir juger de l'efficacité de différentes techniques en simulation moléculaire. Et cet avantage est d'autant plus appréciable qu'il nous a permis de tester non seulement l'efficacité de tel ou tel mouvement mais surtout de leur combinaison. Il permet par ailleurs de mettre à l'épreuve des hypothèses ou des usages établis tels que la distribution des fréquences des sous-systèmes en thermalisation parallèle.

8.5 Développements futurs envisageables.

Le développement le plus évident de ces travaux est l'extension de l'utilisation de l'algorithme proposé. En effet, nous avons pris comme banc de test la simulation moléculaire de polymères linéaires, mais il existe une diversité importante en nature des molécules simulables ainsi qu'en mouvements de Monte Carlo. Par exemple, les mouvements de reptation par croissance arborescente, de pontage inter-chaînes méritent d'être étudiés plus longuement au travers de cet outil. Insistons aussi sur l'optimisation des paramètres de la thermalisation parallèle pour laquelle il serait intéressant d'optimiser simultanément les fréquences des sous-systèmes ainsi que celle des mouvements de Monte Carlo. Toutefois multipliant ainsi la complexité de l'espace de recherche, il faut sans aucun doute envisager des simulations beaucoup plus longues que celles présentées ici. Enfin on peut aussi ajouter aux paramètres d'optimisation ceux de certains mouvements de Monte Carlo, habituellement réglés en fonction du taux de réussite du mouvement et non de son efficacité globale, comme le déplacement maximal lors d'une translation.

Un autre développement est l'utilisation de stratégies multicritères pour tenir compte à la fois du critère de lacunarité et des autres critères. Il existe déjà pour cela des possibilités en EA qu'il doit être possible d'ajoindre au fonctionnement de l'EAII.

8.6 Problèmes ouverts.

Nous finirons en revenant sur trois problèmes essentiels qui sont encore ouverts. Tout d'abord comme nous le faisions remarquer en section 2.7, les algorithmes de Monte Carlo et les algorithmes évolutionnaires sont d'une évidente parenté, étant de nature markovienne. Il reste encore à contrôler explicitement cette nature markovienne des algorithmes évolutionnaires, comme c'est le cas pour l'algorithme de Metropolis, pour envisager de les utiliser directement comme algorithme d'échantillonage. Enfin l'optimisation des températures (aussi bien leur nombre et que les valeurs qu'elle prennent) en thermalisation parallèle reste un problème ouvert, dans la mesure où le changement des valeurs des température en cours de simulation requiert au préalable une justification théorique, quelle que soit la manière dont on effectue ce changement. Enfin de la même manière que nous avons défini le critère de lacunarité, on peut aussi envisager de définir d'autres critères, par exemple en se fondant sur la maximisation de l'entropie du système.

Bibliographie

[1] M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[2] D.V. Arnold and H.G. Beyer. Efficiency and mutation strength adaptation of (μ/μ_i, λ)-es in a noisy environment. In M. Schoenauer, K. Deb, G. Rüdolph, X. Yao, E. Lutton, J. J. Merelo, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 39–48. Springer, 2000.

[3] T. Bäck. Evolution strategies : An alternative evolutionary algorithm. In J.M. Alliot, E. Lutton, E. Ronald, and D. Snyers, editors, *Artificial Evolution*, Lecture Notes in Computer Science, pages 3–21. Springer, 1995.

[4] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In Grefenstette J.J., editor, *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 101–111, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

[5] S. Beiersdörfer, J. Schmitt, M. Sauer, A. Shulz, S. Siebert, J. Hesser, R. Männer, and J. Wolfrum. Finding the conformation of organic molecules with genetic algorithms. Number 1141 in Lecture Notes in Computer Science, pages 972–981, Heidelberg, 1996. Springer.

[6] A.M. Boumaza and J. Louchet. Dynamic flies : Using real-time parisian evolution in robotics. In E.J.W. Boers and al., editors, *Applications of Evolutionary Computing : Proceedings of EvoWorkshops 2001, EvoCOP, EvoFlight, EvoIASP, EvoLean and EvoSTIM*, volume 2037 of *Lecture Notes in Computer Science*, pages 288–297. Springer, 2001.

[7] R. Cerf. *Une théorie asymptotique des algorithmes génétiques*. PhD thesis, Université de Montpelier II, 1994.

[8] R. Cerf. Asymptotic convergence of genetic algorithms. In *Artificial Evolution, AE95, Selected Papers*, volume 1063 of *Lecture Notes in Computer Science*, pages 37–54. Springer Verlag, 1995.

[9] D.E. Clark, editor. *Evolutionary Algorithms in Molecumar Design*, volume 8 of *Methods and Principles in Medicinal Chemistry*. Wiley VCH, 2000.

[10] S. Consta, T.J.H. Vlugt, J. Wichers Hoeth, B. Smit, and D. Frenkel. Recoil growth algorithm for chain molecules with continuous interactions. *Molecular Physics*, 97 :1243–1254, 1999.

[11] S. Consta, N.B. Wilding, D. Frenkel, and Z. Alexandrowicz. Recoil growth : An efficient simulation method for multi-polymer systems. *Journal of Chemical Physics*, 110(6) :3220–3228, 1999.

[12] D.W. Corne, J.D. Knowles, and M.J. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. In M. Schoenauer, K. Deb, G. Rüdolph, X. Yao, E. Lutton,

J. J. Merelo, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 839–848. Springer, 2000.

[13] Y Davidor. Epistasis variance : suitability of a representation to genetic algorithms. *Complex Systems*, 4 :369–383, 1990.

[14] Y. Davidor. Epistasis variance : A viewpoint on GA-hardness. In J.E. Rawlins, editor, *Foundations of Genetic Algorithms.*, pages 23–35. Morgan Kaufmann Publishers, 1991.

[15] J. Devillers, editor. *Genetic Algorithms in Molecular Modeling.* Academic Press, 1996.

[16] J.S. Dixon. Flexible docking of ligands to receptor sites using genetic algorithms. In C.G. Wermuth, editor, *Trends in QSAR and Molecular Modeling 92 : Proceedings of the 9th European Symposium on Structure-Activity Relationships*, QSAR and Molecular Modeling, pages 412–413, Leiden, The Netherlands, 1993. ESCOM Science Publishers.

[17] L.R. Dodd and D.N. Theodorou. *Atomistic Monte Carlo Simulation and Continuum Mean Field Theory of the Structure of Equation of State Properties of Alkanes and Polymer Melts*, volume 116 of *Advances in Polymer Science*, pages 249–283. Springer-Verlag, Berlin Heidelberg, 1994.

[18] M. Falcioni and M.W. Deem. A biased monte carlo scheme for zeolite structure solution. *Journal of Chemical Physics*, 110(3), 1999.

[19] D.B. Fogel. Asymptotic convergence properties of genetic algorithms and evolutionary programming : analysis and experiments. *Cybernetics and Systems*, 25 :389–407, 1994.

[20] C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1) :1–16, 1995.

[21] O. François. An evolutionary strategy for global optimization and its markov chain analysis. *IEEE Transactions on Evolutionary Computation*, 2 :77–90, 1998.

[22] M.I. Freidlin and A.D. Wentzell. *Random perturbations of dynamical systems.* Springer Verlag, New York, 1984.

[23] D. Frenkel and B. Smit. *Understanding Molecular Simulation : From Algorithms to Applications.* Academic Press, 1996.

[24] C.J. Geyer. Markov chain monte carlo maximum likelihood. In *Computing Science and Statistics : Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.

[25] C.J. Geyer and E.A. Thompson. Annealing markov chain monte carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431) :909–920, 1995.

[26] R.C. Glen and A.W.R. Payne. A genetic algorithm for the automated generation of molecules within constraints. *Journal of Computer-Aided Molecular Design*, 9 :181–202, 1995.

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

[28] D.E. Goldberg. Genetic algorithms and walsh functions : I. A gentle introduction. *Complex Systems*, 3(2) :129–152, April 1989.

[29] D.E. Goldberg. Genetic algorithms and walsh functions : II. Deception and its analysis. *Complex Systems*, 3(2) :153–171, April 1989.

[30] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J.J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[31] D.E. Goldberg and P. Segrest. Finite markov chain analysis of genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

[32] J.J. Grefenstette. Deception considered harmful. In L.D. Whitley, editor, *Foundations of genetic algorithms 2*, volume 2, pages 75–91. Morgan Kaufmann Publihsers, 1992.

[33] S. Hahn. *Handbook of Evolutionary Computation*, chapter Tuning Monte Carlo generator parameters to measured data by using genetic algorithms. IOP Publishing Ltd and Oxford Univeristy Press, 1997.

[34] U. Hammel and T. Bäck. Evolution strategies on noisy functions. how to improve convergence properties. In Y. Davidor, R. Männer, and Schwefel H.P., editors, *Paralel Problem Solving from Nature - PPSN III*. Springer, 1994.

[35] J. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press, 1975.

[36] M. Jerrum. *Probabilistic Methods for Algorithmic Discrete Mathematics*, chapter Mathematical Foundations of Markov Chain Monte Carlo Method. Springer-Verlag, 1998.

[37] M. Jerrum and A. Sinclair. *Approximation Algorithms for NP-hard Problems*, chapter The Markov Chain Monte Carlo method : an approach to approximate counting and integration. PWS, 1996.

[38] R. Judson. *Genetic Algorithms and their Use in Chemistry*, volume 10 of *Reviews in Computational Chemistry*. VCH Publishers Inc., New York, 1997.

[39] J.R. Koza. *Genetic Programming : On the programming of Computers by means of Natural Evolution*. MIT Press, Massachusetts, 1992.

[40] Y. Landrin-Schweitzer and E. Lutton. Perturbation theory for evolutionary algorithms : Toward an estimation of convergence speed. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, and H.-P. Merelo, J.J Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, volume 1917 of *Lecture Notes in Computer Science*, pages 109–118. Springer, 2000.

[41] B. Leblanc and E. Lutton. Bitwise regularity coefficients as a tool for deception analysis of a genetic algorithm. Technical Report RR-3274, INRIA, 1997.

[42] B. Leblanc and E. Lutton. Bitwise regularity and GA-hardness. In *Proceedings of the 5th IEEE International Conference on Evolutionary Computation.*, 1998.

[43] B. Leblanc, E. Lutton, B. Braunschweig, and H. Toulhoat. History and immortality in evolutionary computation. In *Proceedings of EA'01 : the 5th international conference on Artificial Evolution*, Lecture Notes in Computer Science, Le Creusot, France, October 2001. Springer Verlag.

[44] B. Leblanc, E. Lutton, B. Braunschweig, and H. Toulhoat. Improving molecular simulation : a meta optimisation of monte carlo parameters. In *Proceedings of CEC 2001, Congress on Evolutionary Computation*, Seoul, Korea, May 2001.

[45] J. Levy-Vehel. About lacunarity, some links between fractal and integral geometry, and aplication to texture segmentation. Technical Report RR-1188, INRIA, 1990.

[46] E. Lutton and J. Lévy Véhel. Some remarks on the optimization of holder functions with genetic algorithms. Research Report 2627, INRIA, July 1995.

[47] E. Lutton and J. Lévy Véhel. Hölder functions and deception of genetic algorithms. *IEEE transactions on Evolutionary computation*, 2(2) :56–72, July 1998.

[48] S.W. Mahfoud. *Niching Methods for Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1995.

[49] B.B. Mandelbrot. *The Fractal Geometry of Nature.* CA :Freeman, San Francisco, 1982.

[50] V.G. Mavrantzas, T.D. Boone, E. Zervopoulou, and D.N. Theodorou. End-brdiging monte carlo : A fast algorithm for atomostic simulation of condensed phases of long polymer chains. *Macromolecules*, 32 :5072–5096, 1999.

[51] N. Metroplis, A.W. Rosenbluth, M.N. Rosenbluth, A.N. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21 :1087–1092, 1953.

[52] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained paramater optimizations. *Evolutionary Computation*, 4(1) :1–32, 1996.

[53] B.L. Miller. *Noise, Sampling and Efficient Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1997.

[54] A.E. Nix and D.V. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5 :79–88, 1992.

[55] P.V.K. Pant and D.N. Theodorou. Variable connectivity methods for the atomistic monte carlo simulation of polydispers. *Macromolecules*, 28 :7224–7234, 1995.

[56] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer systeme nach prinzipien der biologischen evolution.* Frommann-Holzboog Verlag, Stuttgart, 1973.

[57] S. Rochet. *Convergence des algorithmes génétiques : modèles stochastiques et épistasie.* PhD thesis, Université de Provence, France, 1998.

[58] M.N. Rosenbluth and A.W. Rosenbluth. Monte carlo simulations of the average extension of molecular chains. *Journal of Chemical Physics*, 23 :356–359, 1955.

[59] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1) :96–101, 1994.

[60] Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history of search. In M. Schoenauer, K. Deb, G. Rüdolph, X. Yao, E. Lutton, J. J. Merelo, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 571–580. Springer, 2000.

[61] M. Schoenauer and Z. Michalewicz. Evolutionary computation : An introduction. *Control and Cybernetics*, 26(3) :307–338, 1997.

[62] J.J. Siepmann and D. Frenkel. Configurational-bias monte carlo : A new sampling scheme for flexible chains. *Molecular Physics*, 75 :59–70, 1992.

[63] L.H. Sperling. *Introduction to Physical Polymer Science.* Wiley-Interscience, 1992.

[64] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231 :75–81, 1993.

[65] T.J.H. Vlugt. Efficiency of parallel cbmc simulations. *Molecular Simulations*, 23 :63–78, 1999.

[66] M.G. WU and M.W. Deem. Efficient monte carlo methods for cyclic peptides. *Molecular Physics*, 97(4) :559–580, 1999.

[67] E. Zitzler and Thiele L. Multiobjective evolutionary algorithms : A comparative study case and the strength pareto approach. *IEEE Transactions on Evolutionary Computation.*, 2(4) :257–272, 1999.

Glossaire

Nous rapellons ici les significations des principales abrévations utilisées dans ce mémoire, ainsi que la section où ils sont définis.

- **ACO** : **A**utocorrélation **C**umulée des **O**rientations de vecteurs normalisés de chaîne. Section 6.1.3.
- **AE** : **A**lgorithme **E**volutionnaire. Section 2.5.
- **AG** : **A**lgorithme **G**énétique. Section 2.5.
- **AGP** : **A**lgorithme **G**énétique avec **P**ondération de fitness. Section 5.2.
- **AR** : **A**lgorithme de **R**éférence. Section 4.1.2.
- **DCM** : **D**éplacement carré cumulé des **C**entres de **M**asse. Section 6.1.3.
- **DM** : **D**éplacement carré cumulé des **M**onomères. Section 6.1.3.
- **EA** : **E**volution **A**rtificielle. Section 2.5.
- **EAII** : **E**volution **A**rtificielle d'**I**ndividus **I**mmortels. Chapitre 5.
- **FCE** : **F**réquence à **C**harge **E**gale. Section 4.2.1.
- **FE** : **F**réquence **E**ffective. Section 4.2.1.
- **FV** : **F**luctuation de **V**olume. Section 2.3.3.
- **MCM** : **M**onte **C**arlo **M**arkovien. Section 2.1.
- **Pontage IC** : Pontage **I**nter-**C**haînes. Section 4.2.1.
- **RCA** : Reptation par Croissance Arborescente. Section 6.7.
- **Reptationn IC** : Reptation **I**nter-**C**haînes. Section 4.2.1.

Annexe A

Régularité Höldérienne et fonctions de Weierstrass.

La régularité au sens de Hölder est un outil caractérisant la régularité locale d'une fonction :

Définition 1 (Fonctions Höldériennes) *Soient (X, d_X) et (Y, d_Y) deux espaces métriques. Une fonction $F : X \to Y$ est dite Höldérienne d'exposant $h > 0$, si pour tout couple $(x, y) \in X$ vérifiant $d_X(x, y) < 1$, on a :*

$$d_Y(F(x), F(y)) \leq k.d_X(x, y)^h \tag{A.1}$$

pour une constante $k > 0$.

Toute fonction Höldérienne d'exposant h est continue (mais non nécessairement dérivable) et est également Höldérienne d'exposant h' pour tout $h' \in]0, h[$. Intuitivement, on peut caractériser une fonction Höldérienne d'exposant h bas comme plus "irrégulière" qu'une fonction Höldérienne d'exposant h élevé [1].

Définition 2 (Fonctions de Weierstrass :) *Elles sont définies par :*

$$W_{b,s}(x) = \sum_{i=1}^{\infty} b^{i(s-2)} sin(b^i x) \quad avec\ b > 1\ et\ 1 < s < 2$$

Ces fonctions ne sont nulle part différentiables et possèdent une infinité d'optima locaux. On appelle s la dimension et on a alors directement le coefficient de Hölder par $h = (2 - s)$. Plus s est proche de 1, plus h est aussi proche de 1, et plus la fonction est régulière, ce qu'on peut constater sur figures A.1 et A.2.

[1] En prenant pour valeur h la plus grande vérifiant (A.1).

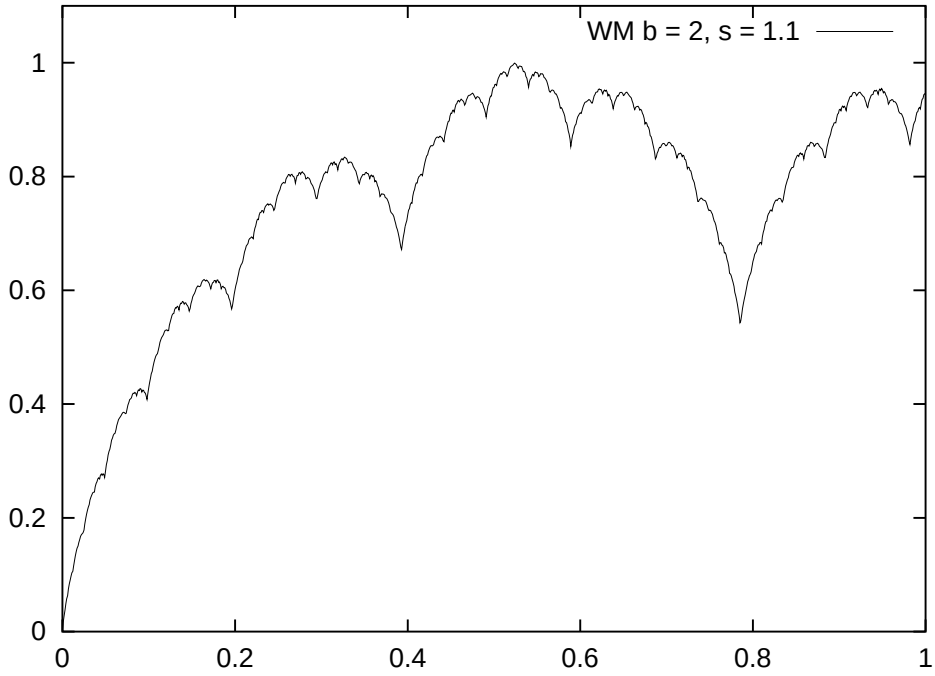


FIG. A.1 – Graphe de la fonction $W_{2,1.1}$.

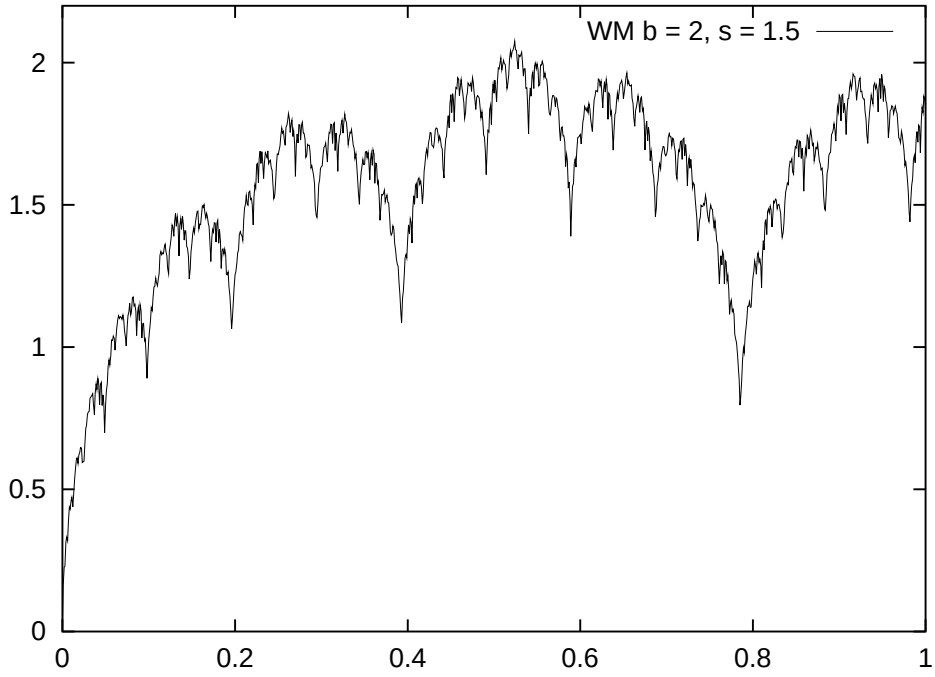


FIG. A.2 – Graphe de la fonction $W_{2,1.5}$.

Résumé :

Le domaine de la simulation moléculaire a pour but de simuler un ensemble de particules en interaction représentant un système physico-chimique. Les algorithmes de Monte Carlo par chaîne de Markov appliqués dans ce cas peuvent rencontrer des problèmes d'efficacité statistique analogues à celles rencontrés par la dynamique moléculaire lors de la simulation de molécules complexe, comme par exemple des polymères. Le but étant d'échantillonner l'ensemble des configurations possibles, en accord avec la distribution de Boltzmann-Gibbs, l'efficacité statistique de tels algorithmes réside dans la capacité à fournir plus rapidement des états décorrélés couvrant l'espace des congigurations, constituant ainsi un échantillonnage statistiquement valide. Nous nous sommes intéressés aux possibilités offertes par l'évolution artificielle (EA, classe d'algorithmes d'optimisation stochastique inspirés du principe biologique de l'évolution darwinienne) pour contribuer à améliorer cette efficacité. Ayant exploré l'utilisation de différentes mesures comme critère d'optimisation, nous avons identifié les fréquences relatives des différents mouvements de Monte Carlo, applicables conjointement lors d'une même simulation, comme degrés de liberté pouvant être optimisés. Nous avons combiné des simulations parallèles avec un "serveur" génétique afin d'effectuer un échantillonnage tout en optimisant simultanément les fréquences des mouvements de Monte Carlo. Nos simulations ont montré qu'il était possible d'obtenir des améliorations par rapport à des réglages de références, pour les critères considérés. Adaptant cet outil au cadre du Monte Carlo avec thermalisation parallèle (*parallel tempering*) nous avons pu améliorer certains de ses paramètres et indiqué des pistes pour améliorer encore le choix des températures additionnelles.

Abstract :

Molecular simulation aims at simulating particles in interaction, describing a physico-chemical system. When considering Markov Chain Monte Carlo sampling in this context, we often meet the same problem of statistical efficiency as with Molecular Dynamics for the simulation of complex molecules (polymers for example). The search for a correct sampling of the space of possible configurations with respect to the Boltzmann-Gibbs distribution is directly related to the statistical efficiency of such algorithms (i.e. the ability of rapidly providing uncorrelated states covering all the configuration space). We investigated how to improve this efficiency with the help of Artificial Evolution (AE). AE algorithms form a class of stochastic optimization algorithms inspired by Darwinian evolution. Efficiency measures that can be turned into efficiency criteria have been first searched before identifying parameters that could be optimized. Relative frequencies for each type of Monte Carlo moves, usually empirically chosen in reasonable ranges, were first considered. We combined parallel simulations with a "genetic server" in order to dynamically improve the quality of the sampling during the simulations progress. Our results shows that in comparison with some reference settings, it is possible to improves the quality of samples with respect to the chosen criterion. The same algorithm has been applied to improve the Parallel Tempering technique, in order to optimize in the same time the relative frequencies of Monte Carlo moves and the relative frequencies of swapping between sub-systems simulated at different temperatures. Finally, hints for further research in order to optimize the choice of additional temperatures are given.