MOLECULAR PHYSICS, 20 NOVEMBER 2003, VOL. 101, NO. 22, 3293–3308

Taylor & Francis
Taylor & Francis Group

Improving the sampling efficiency of Monte Carlo molecular simulations: an evolutionary approach

BENOIT LEBLANC[1], BERTRAND BRAUNSCHWEIG[1], HERVÉ TOULHOAT[1]* and EVELYNE LUTTON[2]

[1]Institut Français du Pétrole (IFP), 1 et 4, avenue de Bois-Préau BP 311, 92852 Rueil-Malmaison Cédex, France
[2]Institut National de Recherche en Informatique et Automatique (INRIA), Project FRATALES, BP 105 78153 Le Chesnay Cédex, France

(*Received 15 May 2003; revised version accepted 24 September 2003*)

We present a new approach in order to improve the convergence of Monte Carlo (MC) simulations of molecular systems belonging to complex energetic landscapes: the problem is redefined in terms of the dynamic allocation of MC move frequencies depending on their past efficiency, measured with respect to a relevant sampling criterion. We introduce various empirical criteria with the aim of accounting for the proper convergence in phase space sampling. The dynamic allocation is performed over parallel simulations by means of a new evolutionary algorithm involving 'immortal' individuals. The method is bench marked with respect to conventional procedures on a model for melt linear polyethylene. We record significant improvement in sampling efficiencies, thus in computational load, while the optimal sets of move frequencies are liable to allow interesting physical insights into the particular systems simulated. This last aspect should provide a new tool for designing more efficient new MC moves.

1. Introduction

In the field of Monte Carlo (MC) simulations of complex molecular systems, such as polymers in dense amorphous phase, much effort is currently being undertaken for the design of more efficient new MC schemes [1–3]. These researches are motivated by the fact that with the increase in size and complexity of molecules, the potential energy surface of such systems is characterized by numerous local minima separated by very high barriers, hence this energy surface is difficult to sample either along the trajectories obtained from direct molecular dynamics or through conventional Markovian Monte Carlo simulations. In this paper we will also address this sampling efficiency problem, but our approach will be to express it directly in the framework of optimization techniques.

Having defined our benchmark model for testing the efficiency of MC simulation schemes of complex molecular systems, we review in §2 possible numerical criteria for measuring this efficiency over the whole simulation.

Actually algorithms involving such criteria are controlled by numerous parameters, among which some are empirically set, such as the relative frequencies of the several MC moves generally used in conjunction.

We further present a new evolutionary algorithm (EA) designed to dynamically optimize these parameters with respect to an appropriate criterion. As we will see, such an algorithm is well suited for optimization problems where the fitness function (or cost function) has no derivative or is even subject to evaluation noise.

In §3, numerical experiments are presented which were designed to evaluate the interest of our approach for improving the simulation performance: the discussion puts forward the global reduction in computational load brought about by evolutionary dynamical optimization, as well as the outcome in terms of physical insight on the significance of individual MC moves in a given complex molecular system.

2. Methods

2.1. *Monte Carlo simulations*

2.1.1. *Choice of our reference molecular model, simulation ensemble, and forcefield*

We chose to work with the well-studied linear polyethylene model described in [4], with the same constants:

- Unified atom model considering each CH_2 or CH_3 group as a single active site.

*Author for correspondence. e-mail: Herve.Toulhoat@ifp.fr
†Also at INRIA, now at Materials Design 44, av F.A. Bartholdi, 72000 Le Mans, France.

Molecular Physics ISSN 0026–8976 print/ISSN 1362–3028 online © 2003 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/0026897031000163230

- Fixed monomer-to-monomer link length of 1.54 Å, corresponding to C–C bond length.
- Lennard-Jones pair interaction potential. The characteristic radius is $\sigma_{LJ} = 3.94$ Å, and the potential well depth $\epsilon_{LJ} = 0.098$ kcal mol^{-1}. For two sites i, j where the distance between them is r_{ij}, we have then:

$$\vartheta_{LJ}(r_{ij}) = 4\epsilon_{LJ} \times \left[\left(\frac{\sigma_{LJ}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{LJ}}{r_{ij}} \right)^{6} \right]. \quad (1)$$

In practice, in order to compute the total interaction potential at a particular site, we only consider neighbouring sites that are located within a cutoff radius σ_{cut} (here $\sigma_{cut} = 8.67$ Å). A term taking into account long-range interactions, depending on the density and on this radius, is finally added (see [5] pages 31–35).

- Van der Ploeg and Berendsen bending potential, function of a bond angle θ (given by three following monomers of the same chain) allowed to fluctuate around the mean value $\theta_0 = 112^\circ$. Given $k_\theta = 57\,950$ K rad^{-1}:

$$\frac{\vartheta_{VPB}(\theta)}{k_B} = \frac{1}{2} k_\theta (\theta - \theta_0)^2. \quad (2)$$

- Ryckaert and Bellman torsional potential, function of dihedral angle ϕ (given by four following monomers of the same chain). Given $c_0 = 1116$ K, $c_1 = 1462$ K, $c_2 = -1578$, $c_3 = -368$ K, $c_4 = 3156$ K, $c_5 = -3788$ K:

$$\frac{\vartheta_{tor}}{k_B} = \sum_{k=0}^{5} c_k \cos^k(\phi). \quad (3)$$

- Cubic simulation model box with periodic boundary conditions.

Simulations are performed in the following *nNPT* ensemble:

- $n = 640$ monomers.
- $N = 20$ or $N = 10$ chains, resulting in chains of 32 or 64 monomers.
- Pressure $P = 1$ atm.
- Temperature $T = 450$ K.

2.1.2. *Choice of a set of Monte Carlo moves*

We consider Monte Carlo moves commonly used in molecular simulations and which are all applicable in the *nNPT* ensemble. First of all, we draw attention to the fact that all MC moves do not require the same computation time because the number of displaced monomers varies from one to another. As the evaluation of the Lennard-Jones potential within the sphere of cutoff radius requires more computation time, we can reasonably assume that the total computation time of any move is proportional to the number of displaced monomers. We will further name this number *degree*, or *deg*.

(1) *Translation*, $deg = n/N$ (figure 1, upper left): the whole molecule is translated along a random vector. The translation distance is randomly chosen in the interval $[0, d_{max}]$. The parameter d_{max} is sometimes dynamically adjusted in order to reach a prescribed acceptance rate. This translation move can generate expensive calculations of the Lennard-Jones potential if there are too many interaction sites and is therefore costly for long chains.
(2) *Rotation*, $deg = 1$ (figure 1, upper right): a terminal monomer is rotated within a sphere centred on its preceding site; energy variation must be calculated for the three potentials; this simple move implies only one monomer.
(3) *Reptation*, $deg = 1$ (figure 1, middle left): a terminal monomer is removed, added at the other end of the chain, and rotated. The computation requirements are the same as for rotation.

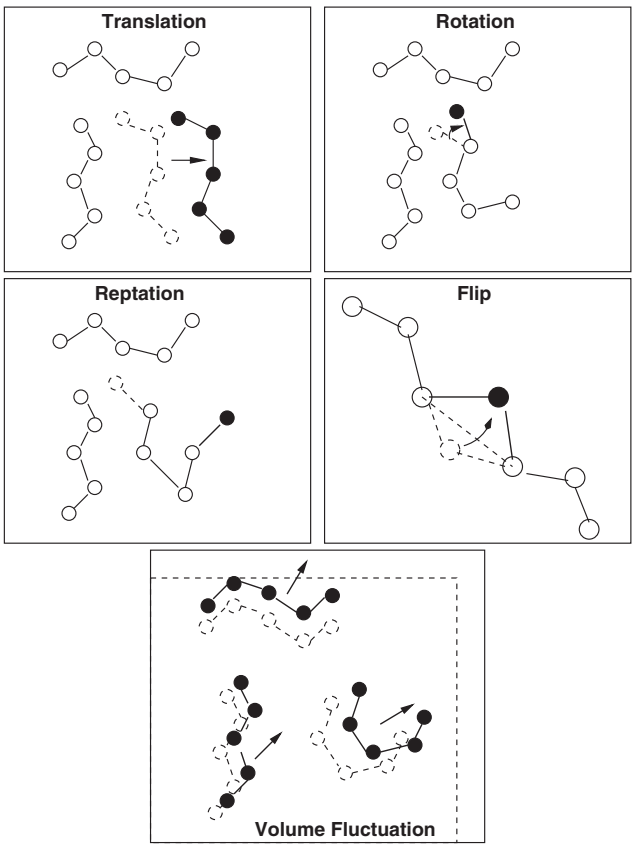

Figure 1. MC moves considered in this work.

(4) *Flip*, $deg = 1$ (figure 1, middle right): a monomer inside a chain is rotated along the axis of its two neighbouring sites. The site is moved, two bond angles change, four torsion angles change. The rotation angle is randomly chosen in the interval $[0, \phi_{\max}]$. The parameter $\phi_{\max}$ is sometimes dynamically adjusted in order to reach a prescribed acceptance rate.

(5) *Volume Fluctuation, VF*, $deg = n$ (figure 1, bottom): under constant pressure conditions, this Monte Carlo move is required in order to let the system's density fluctuate around the corresponding mean density. The simulation box volume is increased or reduced with the following rule:

$$\ln(V_n) = \ln(V_o) + \Delta \ln V, \tag{4}$$

where $\Delta \ln V$ is randomly drawn in the interval $[-\Delta_{\max}, \Delta_{\max}]$. The acceptance rule in this case is:

$$\begin{aligned} &\mathrm{acc}((r_o^N, V_o), (r_n^N, V_n)) \\ &\quad = \min\Big\{1, \exp\Big(- \beta \times \Big[U(r_n^N, V_n) - U(r_o^N, V_o) \\ &\qquad + P(V_n - V_o) + (N+1)\beta^{-1} \ln(V_n/V_o)\Big]\Big)\Big\} \end{aligned} \tag{5}$$

where β stands for $1/(k_B T)$. Owing to the constant bond length between monomers when the volume changes, the chains are displaced according to their centre of mass. All pair interaction energies of the system have to be recomputed for each move.

2.1.3. *Criteria for sampling efficiency in Monte Carlo simulations of complex molecular systems*

The sampling efficiency problem in Monte Carlo simulations can be viewed as the statistical error due to lack of uncorrelated samples. Indeed if $\langle A \rangle$ is the quantity of interest, it is usually estimated with:

$$\langle A_{\mathrm{run}} \rangle = \frac{1}{n_s} \sum_{i=0}^{n_s - 1} A_i, \tag{6}$$

where n_s denotes the number of samples of the observable A taken during the simulation. In the ideal case where (A_i) are completely uncorrelated, we get:

$$\sigma^2(\langle A_{\mathrm{run}} \rangle) = \frac{\sigma^2(A)}{n_e}. \tag{7}$$

If the configurations on which the (A_i) are measured are partly correlated, the variance will be higher. The question of measuring this decrease in correlation will be our concern in this section.

2.1.3.1. *Analytical criterion.*

In the field of Markov chain Monte Carlo methods (see [6]), this issue has been identified as estimating the *mixing time* $\tau(\delta)$, that is the number of steps before the Markov chain is 'close' enough to the stationary distribution π. It can be formalized for the case of a discrete state space Ω as:

$$\begin{aligned} \tau(\delta) &= \max\{x \in \Omega : \tau_x(\delta)\} \quad \text{with} \\ \tau_x(\delta) &= \min\{t : \delta_x(t') \leq \delta, \forall t' \geq t\} \end{aligned} \tag{8}$$

where:

$$\delta_x(t) = \frac{1}{2} \sum_{y \in \Omega} |P^t(x, y) - \pi(y)|. \tag{9}$$

In this case $\delta_x(t)$ is a distance between the $P^t(x, .)$ distribution (distribution of the conditional law $Pr(X_t | X_0 = x)$) and the π distribution. From this definition and sufficient information about the Markov chain structure, it is sometimes possible to get an upper bound for this mixing time. Such a bound could be used in a sampling algorithm to get the number of steps between two uncorrelated states (or configurations). But to our knowledge, in the case of a complex Markov chain Monte Carlo sampling algorithm such as molecular Monte Carlo simulation, this kind of calculation has not yet been performed.

2.1.3.2. *Chain end-to-end vector autocorrelation.*

In the literature of molecular simulation of chain molecules, when the discussion comes to the efficiency of the simulation, we often find references to the chain end-to-end vector autocorrelation. Actually, we consider for each of the N chains the vector joining each end of the chain. Then these vectors are normalized, and often called 'chain orientation vectors'. In the case of dense amorphous polymers, where chains are tightly entangled, orientation vectors will vary slowly. Therefore it is reasonable to think that if, after a sufficient number of simulation steps, orientation vectors become uncorrelated with regard to their initial orientations, the global system configuration will be uncorrelated with regard to the initial configuration.

For each sampled configuration of the system at simulation step t, we can compute the collection $\{v_t(i)\}_{i=0,\ldots,N}$ of chain orientation vectors. After n_s sampling steps, we obtain $(n_s + 1)$ sampled configurations including the initial one, and it is possible to compute the three following quantities:

- *Instant autocorrelation*: this can be computed at any time, as it depends only on the initial and on

the current configuration:

$$C_v^i(n_s) = \frac{1}{N}\sum_{i=0}^{N} v_{n_s}(i).v_0(i). \tag{10}$$

- *Mean autocorrelation*: the true autocorrelation estimator. It corresponds to the average of autocorrelation of samples separated by $n_s/2$ sampling steps:

$$C_v^m(n_s) = \frac{2}{n_s}\sum_{t=[(n_s/2)+1]}^{n_s}\sum_{i=1}^{N} v_t(i).v_{t-(1/n_s)}(i). \tag{11}$$

- *Cumulated autocorrelation*: defined as the average at every sampling step of the instant autocorrelation:

$$C_v^c(n_s) = \frac{1}{n_s}\sum_{t=1}^{n_s} C_v^i(t). \tag{12}$$

Whatever the definition, autocorrelation will decrease toward 0 as the simulation goes on. But it is also clear that if the instant autocorrelation will decrease faster, it will be more oscillating because it includes less information. The definition of the cumulated autocorrelation is motivated by its iterative computation, and the need to keep only the initial orientation vectors. In comparison to the mean autocorrelation it requires less memory and less computation:

$$C_v^c(n_s+1) = \frac{nC_v^c(n_s) + C_v^i(n_s+1)}{n_s+1}. \tag{13}$$

The question now is what is the best definition of the autocorrelation to consider in order to define an efficiency criterion? Let us assume that we want to maximize this kind of criterion:

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

where $C(t)$ is one of the three previous definitions. Given the stochastic nature of the Monte Carlo algorithm, the outcome of this criterion measure for the same system simulated under the same conditions during the same time will be a random variable. Then it seems desirable to choose the autocorrelation definition that will lead to the smallest standard-deviation/average ratio.

For this purpose we present a simulation of polyethylene under the *nNPT* ensemble, under the conditions that are described in § 3. In this case 32 systems, each with a different equilibrated initial configuration, are simulated independently. Criteria are measured on increasing periods of simulation (counted in explicit seconds of simulation), ($\tau_i = i \times 200\,\mathrm{s}$, $i \in \{1, \ldots, 10\}$). The measures are repeated twice for each τ_i, giving 64 measures for each.

Figure 2 shows the graph for each criterion (c_v^i denoted ci, c_v^c denoted cc and c_v^m denoted cm), and also the standard deviation/average ratio. As expected the c_v^i grows faster than the others. We notice that if the standard deviation/average ratio decreases initially while τ_i increases, it stabilizes around 1/3 for each criterion. The first conclusion is that the remaining variance comes from the stochastic nature of the simulation (and not from an insufficiently long τ_i), and the second is that among our three criteria none can be considered 'the best' with respect to this ratio.

Finally if we consider the problem of memory complexity, c_v^i and c_v^c are the most interesting. Furthermore, just for visual comfort (see figure 3), we may prefer c_v^c, as it is smoother than c_v^i allowing easier visual comparisons.

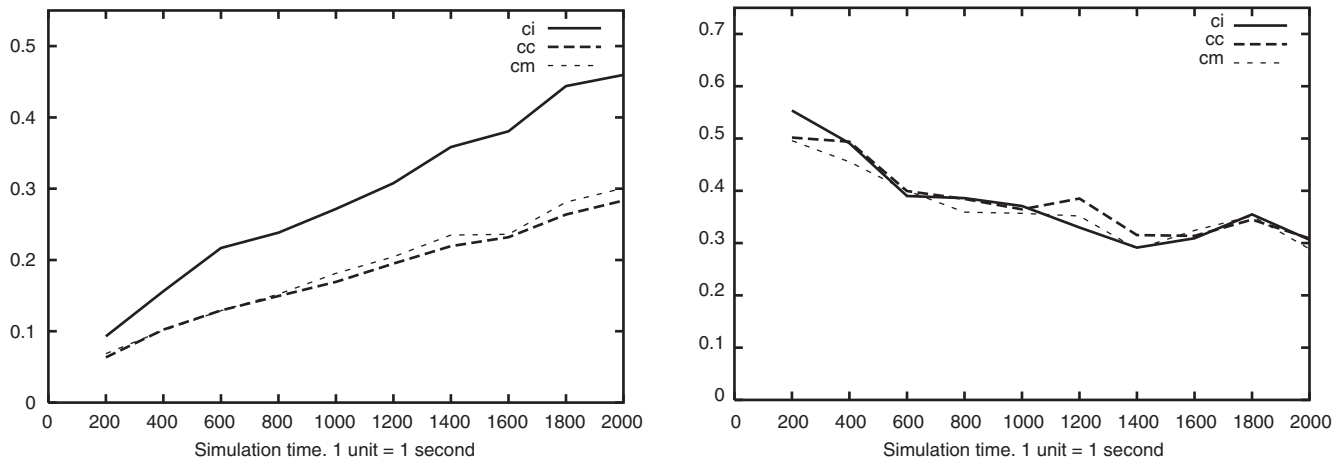

Figure 2. Left: average of measures of orientation vector autocorrelation criteria (c_v^i denoted ci, c_v^c denoted cc and c_v^m denoted cm) as a function of simulation time. Right: standard deviation/average ratio.

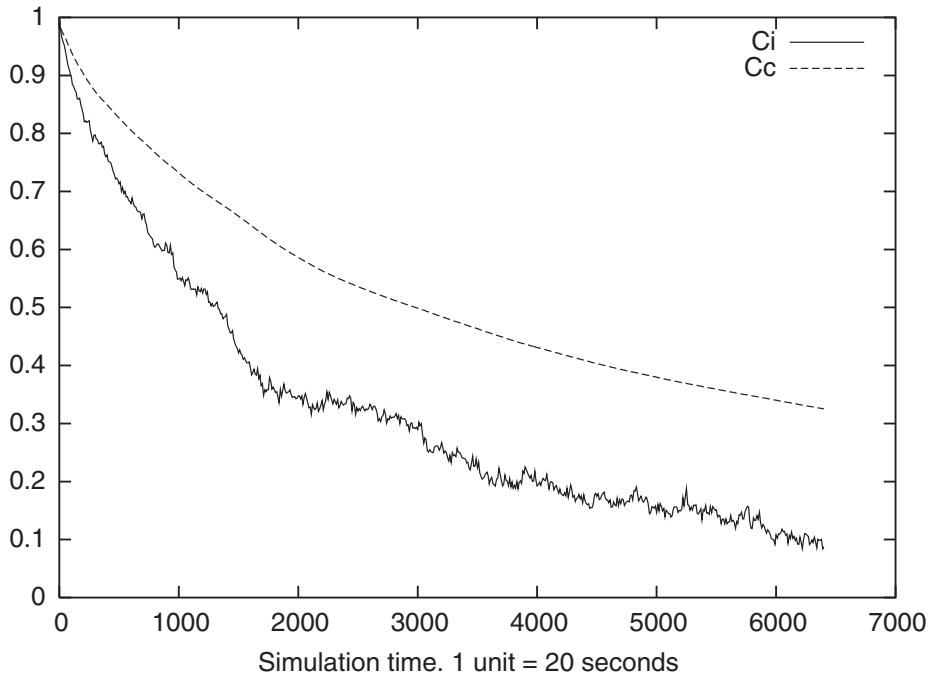

Figure 3. Comparison between instant autocorrelation and cumulated autocorrelation for the same simulation (C_v^i denoted Ci, C_v^c denoted Cc).

2.1.3.3. *Displacement of molecules*. Another quantity generally used when monitoring efficiency of molecular simulation is the displacement of molecules with respect to their initial positions. In the case of large molecules such as polymers, the positions of the centres of mass of chains are generally tracked. Considering that in dense states, chains are tightly entangled, their centres of mass have a slow motion with regard to their own size, and therefore a significant displacement is a good guarantee that configurations are sufficiently uncorrelated.

We propose the two following criteria based on centre of mass chain displacement:

- The *chain's centre of mass mean square displacement*:

$$d^2(t) = \frac{1}{N}\sum_{i=1}^{N}(rc_t(i) - rc_0(i))^2, \tag{14}$$

  with $rc_t()$ denoting the coordinates of the centre of mass of a chain at simulation time t.

- The *chain's centre of mass cumulated mean square displacement*:

$$d_c^2(n_s) = \frac{1}{n_s}\sum_{t=1}^{n_s} d^2(i). \tag{15}$$

A 'cumulated displacement' has no relevance with regard to the physical properties of the system, but once again it allows us to take all the information along the trajectory of the system into account, in a simple manner.

As these properties are increasing along the simulation, they can be used directly as criteria for a maximization. During the simulation presented in § 3, we also measured these criteria, displayed in figure 4. The standard deviation/average ratio here is in favour of the cumulated criterion, but once again it appears that it decreases and then stabilizes when the simulation time increases.

2.1.3.4. *Lacunarity*. With the previous criteria we have seen how to use the geometry of chains in order to measure the mixing speed of the simulated system. But one may object that, if that kind of measure is perfectly suited to molecular dynamics, it should be less suited to some Monte Carlo schemes. For example, in the case of the displacement of molecules, care should be taken on how particle coordinates are updated in order not to create too 'artificial' displacements, particularly in handling the periodic boundary conditions of the simulation box. These reasons motivated us to find a criterion that may be applied without the need for the labelling of molecules, or other kinds of arbitrary information. Therefore we choose to look at the 'local variations' of density of the system: for most systems, molecules are not uniformly distributed in the simulation box. Holes and bulks are generally observed, resulting in local variations of density in subparts of

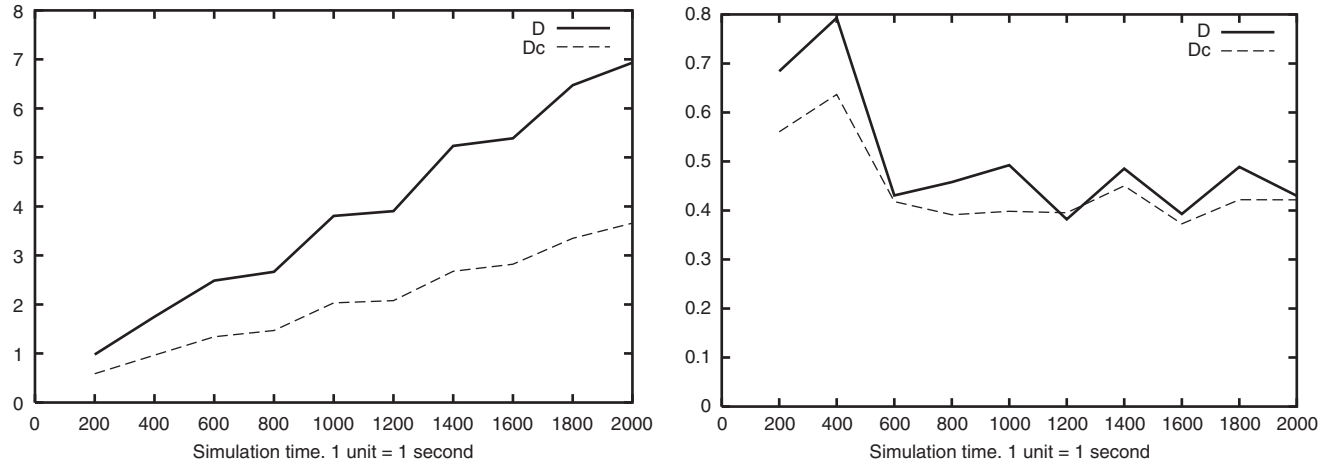

Figure 4. Left: average of criteria d^2 (denoted D) and d_c^2 (denoted Dc), measured in reduced units (1 unit $= \sigma_{\mathrm{LJ}}^2$) against simulation time. Right: standard deviation/average ratio.

the simulation box. These variations will help us in defining a new criterion.

Fractal geometry offers a tool named *lacunarity* [7, 8], often used in image analysis, to measure the mass dispersion of an object (with respect to homogeneity). If we consider an object of mass M, volume V and a subpart of volume v (for example a cube of side ϵ), we want to compare the observed mass m_o of this subpart to the expected mass m_e defined as $(v/V \times M)$. Lacunarity measures the variations of m_o with respect to m_e:

$$L = \left\langle \left(\frac{m_e}{m_o} - 1 \right)^2 \right\rangle. \tag{16}$$

We can directly adapt this measure to the spatial distribution of monomers, using a subdivision of the cubic simulation box in R^3 subcubes (R is a scale parameter). Then we look at the number of monomers per subcube (if the mass of each monomer is the same) and name them $m_R(i), i \in \{1, \ldots R^3\}$. The expected number is $m_e(R) = n/R^3$ if n denotes the number of monomers of the system, so that one gets:

$$L(R) = \frac{1}{R^3} \sum_{i=1}^{R^3} \left(\frac{m_R(i)}{m_e(R)} - 1 \right)^2. \tag{17}$$

$L(R)$ gives the lacunarity of a configuration of the system, given the scale parameter R.

During the simulation of the system at equilibrium, it is possible to compute the lacunarity at different scales of the sampled configurations, and then estimate the equilibrium lacunarity. Furthermore, we can *cumulate* monomer spatial distributions of each configuration, and then observe the evolution of lacunarity as configurations are added. If we assume that an efficiently simulated system should rapidly move holes and bulks, it should result in a rapidly decreasing lacunarity of these cumulated distributions as shown in figure 5.

For a given scale R, we propose the following efficiency criterion:

$$c_L^R(n_s) = \max\left(0, \frac{\hat{L}(R)}{L_c(R, n_s)} - 1\right), \tag{18}$$

with $L_c(R, n_s)$ denoting the lacunarity of the n_s cumulated distributions of monomers and $\hat{L}(R)$ denoting the average of lacunarity measures of each distribution (generally higher than $L_c(R, n_s)$):

$$\hat{L}(R) = \frac{1}{n_s} \sum_{i=1}^{n_s} L(R, i). \tag{19}$$

2.2. *Dynamic allocation of Monte Carlo move frequencies through evolutionary algorithms*

2.2.1. *Evolutionary algorithms*

Evolutionary algorithms (EAs) are population-based stochastic optimizers [9], inspired by the Darwinian principles of evolution of species. They can be briefly described as follows: given a search space, the goal is to find one (or more) point of this space that optimizes a criterion:

(1) Generate a set of points (called *individuals*) of the search space: a *population*.
(2) Compute the criterion (positive real-valued function) for each individual: their *fitness* score.
(3) Select individuals from the population, with random trial biased according to their fitness

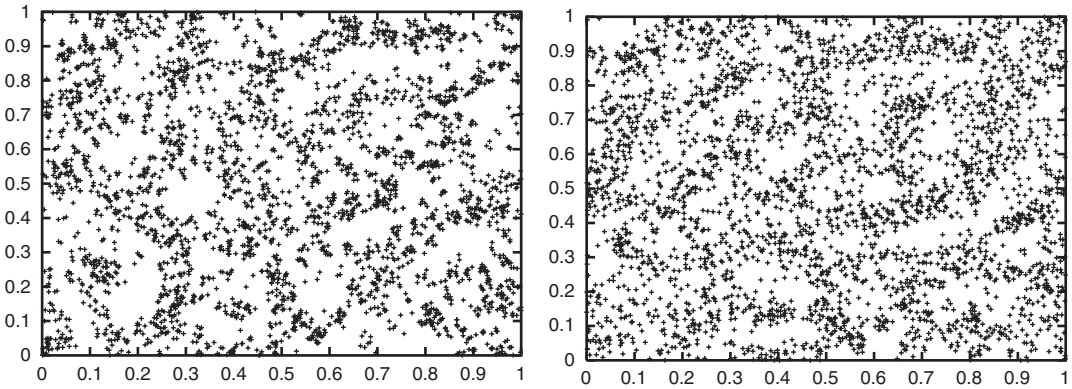

Figure 5. Five sampled configurations of 640 monomers each are added, and cast in the X–Y plane. Configurations come from the same simulation, but with a longer sampling step on the right. The result is that configurations are more correlated on the left, as it can be visually checked: lacunarity is higher on the left.

score: best individuals are more likely to be selected.

(4) Selected individuals (called *parents*) are allowed to reproduce, i.e. *genetic operators* are applied:

   — with a probability p_c each pair of parents is crossed (else duplicated)
   — with a probability p_m resulting *offspring* undergo *mutation* (generally a small random perturbation of the individual).

   These genetic operators are specific to the type of search space.

(5) Offspring are used to build a new generation and the algorithm loops to step 2 until an end criterion is reached (limited number of evaluations for example).

The artificial evolution encompasses, among others, the fields of genetic algorithms [10, 11], evolution strategies [12] and genetic programming [13]. The main differences come from different encoding schemes (binary strings, real vectors, logical trees, etc.) and reproduction strategies, but owing to wide use in various fields of application, these techniques have more or less merged in the EAs acronym. One quality of EAs is the fact that only the value of the fitness function at the points represented by the individuals is required. There is no need for derivatives or continuity, and EAs may also be applied in the presence of noise. For these reasons, EAs are good candidates for irregular, complex problems such as the present one.

2.2.2. *Dynamic evolutionary optimization of parallel molecular simulations*

In § 2.1.3 we proposed possible criteria that could be used in order to turn the sampling efficiency problem into a maximization problem. In order to apply an EA to our problem, it remains to identify parameters of the molecular simulation algorithm that could be tuned in order to optimize one of the criteria.

For that purpose we propose an algorithm based on parallel simulations of the same system. In traditional Monte Carlo approaches, practitioners empirically adjust the parameters of a simulation, for example in the case of several allowed Monte Carlo movements, the relative frequencies of those movements along the simulation. These frequencies have no consequence on the limit distribution of valid configurations, since they only impact the way the search space is sampled during the simulation. However, choosing good sets of such frequencies for a specific problem can significantly improve performances. More formally stated, our problem is the following.

In the case of a Monte Carlo molecular simulation, given a molecular model, specific physico-chemical conditions and a set of adequate MC moves $(M_1, \ldots, M_m)$, *find a frequency distribution* $(\mu_1, \ldots, \mu_m)$ *for those moves that maximize an adequate efficiency criterion*.

In terms of optimization the search space is the space of possible frequency distributions S_μ:

$$S_\mu = \left\{ \mu \in \prod_{i=1}^{m}]0, 1[; \sum_{i=1}^{m} \mu_i = 1 \right\}. \tag{20}$$

We also define a reference algorithm (RA) that is used for comparison: it consists of n_{ps} simulations of polymer systems with different initial states in the same physico-chemical conditions (i.e. different points from the same phase space), each simulation using an equiprobable distribution of allowed movements.

We can describe the algorithm as follows:

- n_{ps} systems are simulated with identical physico-chemical parameters;

- an individual (representing a specific frequency set) is assigned to each system; the initial population is randomly generated;
- the simulation time of one system is divided into n_c cycles;
- the n_c cycles are further divided in n_g generations (see figure 6) during which individuals of the population are evaluated, along with one of the fitness criteria presented in § 2.1.3.

A cycle consists of several elementary MC moves (trials to generate new conformations), totalling a pre-defined CPU time. This way, fitness scores represent the true efficiency of a frequency set over a specified period of simulation time. This makes our algorithm dependent on the hardware, operating system and software used, but supportive of cheap and efficiently mixing moves.

It appears first that the volume fluctuation move is needed but it is very heavy in terms of computation time. Furthermore it has no direct effect on the system mixing. For these reasons we will set once the frequency of this move, and optimize the relative frequencies of the four other moves. In addition, as translation also has a high degree, instead of considering the effective frequencies (EF, probability that a move is chosen at each simulation step), an individual will encode what we call *Equal Charge Frequencies* (ECFs), that will be defined for a move M such that:

$$EF(M) = \frac{ECF(M)}{deg(M)}. \tag{21}$$

ECF is more representative of the share of the total computation load spent in the computation of a trial move. Therefore the search space S_μ will be the space of ECF instead of EF.

For the present case, in the reference algorithm (RA) all MC moves are given an equal ECF of (1/5). This implies that the volume fluctuation will always have an ECF of 20% in the case of the evolutionary runs. The corresponding EFs are given in table 1.

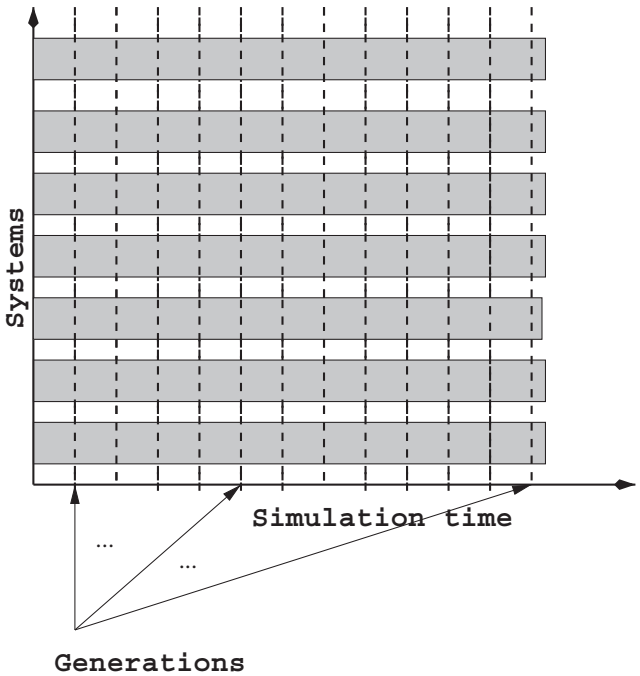

Figure 6. Schematic representation of our EA-based parallel MC simulation: the shaded bars stand for simulated systems. Each system is given MC move frequencies corresponding to an individual of the population, performance is measured on n_c/n_g MC cycles and returned as the fitness score. At each new generation, new individuals are created and each simulation continues with corresponding (hopefully better) frequencies. In comparison, for the reference algorithm (RA), a unique set of frequencies is used for all the systems during all the simulation time.

Table 1. Set of effective frequencies chosen for the reference algorithm (RA).

N	Translation	Rotation	Reptation	Flip	Volume fluctuation
20	20/1941	640/1941	640/1941	640/1941	1/1941
10	10/1931	640/1931	640/1931	640/1931	1/1931

2.2.3. *A new evolutionary algorithm involving immortal individuals*

Some specific characteristics of our algorithm concerning the evolutionary part will now be outlined. It appears first that the evaluation of an individual is the result of a long simulation time (compared to a simple optimization problem). Furthermore, for the same frequency set applied for the same system in the same conditions for the same duration, two independent simulations will lead to different performances, due to the stochastic nature of the Monte Carlo algorithm. For this reason we need to consider that the fitness function is subject to noise. In order to face these two aspects, that is a costly fitness function (in terms of evaluation time) subject to noise, we have proposed in [14] a new EA, called an *immortal evolutionary algorithm* (IEA). It has been tested on reference fitness functions, and has shown good performance in rapidly finding good individuals despite the presence of noise: in order to reduce the effect of noise, similarities between individuals have been used (many instances of a single individual frequently coexist inside a population). Going further in that direction, the whole information produced along the evolution may also be considered: it often happens that an individual is a copy — or a slightly modified copy — of a 'dead' ancestor. In our

problem, if two frequency sets are very similar, we can reasonably assume that performances will be similar on average. As we will see below, keeping track of all evaluations performed along the evolution provides another way to reduce the noise of the fitness function. For this purpose we define an H (history of evaluations) set in the following way:

- Keep information of all evaluated individuals, the H set:

$$H = \left\{ \left(\nu_i, n_{\nu_i}, \tilde{f}(\nu_i) \right), i \in \{1, \ldots, n_H\} \right\},$$

  with ν_i being an individual, n_{ν_i} its number of evaluations and $\tilde{f}(\nu_i)$ the average of these evaluations.
- Consider the max distance on S_μ:

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

- Consider the *euclidian* distance on S_μ:

$$d_2(\mu, \nu) = \left(\sum_{i=1}^{m} (\mu_i - \nu_i)^2 \right)^{1/2}.$$

- Consider neighbourhoods based on max distance:

$$\forall \mu \in S_\mu, \sigma_\infty \in \mathbb{R}_+^*, B_{\sigma_\infty}(\mu) = \left\{ \nu \in S_\mu ; d_\infty(\mu, \nu) \leq \sigma_\infty \right\}.$$

- Define the following *similarity measure* on H:

$$w(\mu, \nu) = \left(1 - \frac{d_2(\mu, \nu)}{m^{1/2} \sigma_\infty} \right).$$

- For each point of H, assign the following weighted fitness score:

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

Actually the fitness score of an individual (considered for the selection) is a weighted average of the scores of all similar individuals.

2.2.3.1. *Immortal Individuals*. The H set can now be used in the following way: each time an individual x has been evaluated, its 'raw' (not yet weighted) fitness score is used to update H. The weighted fitness score can be returned with the computation of $f_H(x)$. Moreover this H set may be used to modify the classical birth and death cycle of a classical EA. More precisely the individuals to be reproduced can be directly selected in this genetic database. This can be seen as a *growing population of immortal individuals*.

Any individual of H may thus have offspring at any time. Thereby the information of the whole evolution is not only used to produce more accurate fitness evaluations but it offers a simple way to maintain diversity. We should also emphasize the asynchronous aspect of this algorithm, that is we do not have to wait for an arbitrary sized population to be fully evaluated in order to perform selection, but at any time we are able to choose from all already evaluated individuals. It is adapted to distributed implementations, for example with a client–server model: a genetic server feeds clients that perform the fitness evaluations. The server can manage the database with the following principles (see also figure 7):

- A pool of random offspring is initially created.
- For any client request, the server supplies an offspring from its pool until this pool is empty.
- As soon as a client has finished the evaluation of its current individual, it is returned to the server that adds the information to H.
- When the offspring pool is empty the server creates new individuals to fill it again. This creation is made by selecting parents from H and applying genetic operators.
- In order to have a minimum initial diversity, we impose that when the server creates new individuals a minimum number of individuals has to be present in H before selection can be applied. If this condition is not fulfilled, offspring are generated randomly until H is sufficiently large.

2.2.3.2. *Selection and genetic operators*. There are many different kinds of selection schemes and genetic operators, applicable to different encoding schemes. We are not going to discuss all of them here, but we will simply specify a combination that appeared to be effective in the test runs presented in [14], and that we decided to use for our present problem.

We proposed to use a specific selection method for the IEA, called *threshold selection*:

- A fitness threshold is computed from the best fitness value of H, ($f_H^{\max} = \max_{x \in H}\{f_H(x)\}$). This threshold is simply the product of $f_H^{\max}$ by a threshold coefficient c_s ($c_s \in [0, 1]$):

$$f_H^s = f_H^{\max} c_s. \tag{22}$$

- Individuals are randomly drawn (without any bias) from H, until an individual y comes out such that $f_H(y) > f_H^s$, which becomes the 'selected individual', allowed to reproduce.
- If no individual satisfies this condition after n_t trials, the best of them is selected.

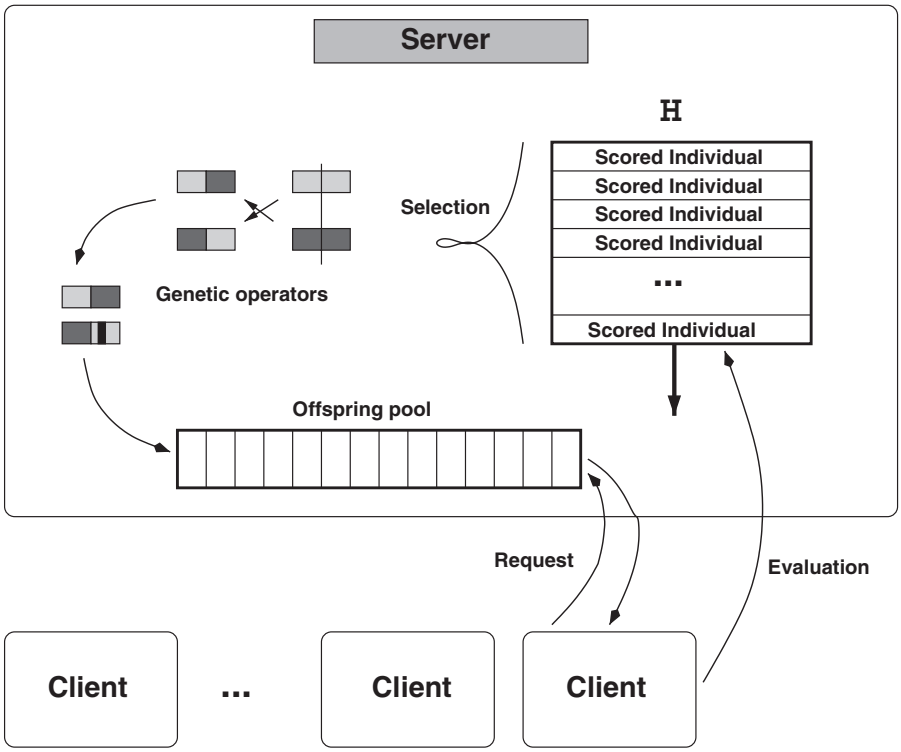

Figure 7. Schematic representation of the IEA algorithm.

As individuals have a vector of real numbers for chromosomes, we applied classical genetic operators for this encoding. After the selection phase, parent individuals are grouped by pairs and crossover is applied with a probability p_c (if crossover is not applied, the offspring are simply copies of their parents) and mutation with a probability p_m (the test is done for each offspring separately):

- Arithmetic crossover, which produces two offspring (x', y') from two parents (x, y):

$$\begin{cases} x' = \gamma x + (1-\gamma)y, \\ y' = (1-\gamma)x + \gamma y. \end{cases} \tag{23}$$

  Where γ is a scalar randomly drawn in the [0, 1] interval. This rule can be applied component by component with an independent γ trial each time.
- Gaussian mutation:

$$x' = x + N(0, \sigma_{\text{mut}}) \tag{24}$$

  where the σ_{mut} variance parameter depends on the feasible region of the search space.

In our case, applying the genetic operators will not result in two normalized offspring vectors. The normalization is performed afterwards in order to always get valid frequency vectors.

3. Numerical experiments: comparison of simulation efficiencies for dynamically optimized versus *a priori* set move frequencies and influence of the sampling criterion

Having discussed possible criteria for the sampling efficiency and designed a specific EA for our problem, we now present numerical experiments in the *nNPT* ensemble.

3.1. *Common simulation parameters*

The simulations presented here have been performed with our own software written in C language. It has been designed with a client-server architecture: the server handles all the IEA routines and gathers all information about the system. Clients connect to the server (TCP/IP 'sockets' are directly used) in order to get an instance (a given configuration) along with a frequency set. The simulations have been performed on a PC cluster of the ID-IMAG laboratory† (Grenoble, France), where each

†http://www-id.imag.fr/Grappes/

Table 2. Fitness functions.

Criterion	Nature
$c_1(\mu,\omega) = (1 - C_v^n(\mu,\omega))$	Cumulated end-to-end chain vector autocorrelation
$c_2(\mu,\omega) = d_c^2(\mu,\omega)$	Cumulated chain centre of mass mean square displacement
$c_3(\mu,\omega) = \left(c_L^3(\mu,\omega) + c_L^4(\mu,\omega)\right)$	Sum of lacunarity criteria for scales $R = 3, 4$
$c_4(\mu,\omega) = \left(c_L^6(\mu,\omega) + c_L^7(\mu,\omega) + c_L^8(\mu,\omega)\right)$	Sum of lacunarity criteria for scales $R = 6, 7, 8$

node is a Pentium III 733 MHz running under the LINUX/Mandrake system. Each client was allocated a node, and the simulation time was directly measured with the client process time.

In the following sections we present a set of simulations performed under the same conditions, except that each one uses a different fitness function; see table 2.

The ω is added as a reminder that the fitness function may be considered to be a random variable, depending on μ.

The remaining parameters are:

- 32 systems are simulated in parallel.
- The simulation time for one system is 80 000 seconds (client process execution time).
- The simulation time is divided into 80 evaluation periods or generations, totalling 2560 evaluations.
- Threshold selection is applied with $c_s = 0.8$, $n_t = 10$.
- Genetic operator probabilities are $p_c = 0.8$, $p_m = 0.05$.
- The neighbourhood radius is $\sigma_\infty = 0.05$.
- The simulation time is divided into two phases. The first 1280 evaluation periods are dedicated to exploration: the ECFs are generated by the IEA algorithm exactly as described previously. The 1280 remaining periods are dedicated to exploitation: the individual of H having the best f_H score is used, but as the criterion is still being evaluated, this 'best individual' may change. For example, if an individual has been declared the best according to a few lucky evaluations, further evaluations will lower its f_H score, and let another one appear as the best.

Simulations will be presented in four sections (one per criterion), and as there are two chain lengths (32 and 64 corresponding to $N = 20$ and $N = 10$), runs will be named A32, A64, ..., D32, D64. The following list sums up the characteristics; throughout all these sections the following results will be inspected:

- Average fitness curves: as there are 32 systems running for 80 evaluation periods, we will display the average fitness average of every period. Both unweighted and weighted scores will be displayed.
- ECF histograms: at the end of the simulations, each individual of H represents an ECF set. For each MC move we look at the histogram of these values.
- The curves of the average (over the 32 systems) of the cumulated end-to-end vector autocorrelations measured over all the simulation time. This global performance measure will be further called CVA.
- The curves of the average (over the 32 systems) of the chain centre of mass mean square displacements measured over all the simulation time. This global performance measure will be further called CMD.
- The curves of the average (over the 32 systems) of the cumulated configuration lacunarities measured over all the simulation time, at scale $R = 3$ and $R = 7$. These global performance measures will be further called **CCL**.

Table 3. Runs lookup table.

Runs	Criterion	Chain length	Section
A32	c_1	32	3.2
A64	c_1	64	3.2
B32	c_2	32	3.3
B64	c_2	64	3.3
C32	c_3	32	3.4
C64	c_3	64	3.4
D32	c_4	32	3.4
D64	c_4	64	3.4

Each time performance will be compared to RA runs RA32 and RA64 (see table 3), depending on chain lengths.

3.2. *Chain end-to-end vector autocorrelation criterion*

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

Fitness curves are displayed in figure 8, and as expected performances are better for the short chains.

In both cases the IEA brings an improvement even if in the case of the long chains (A64) the curves are more oscillating, even when considering the weighted fitness scores. We clearly notice on the A32 fitness curves, the exploration/exploitation transition at generation 40, where the fitness jumps because only the best individual is used. The fitness decreases a little afterwards due to the 'correction' of the fitness score of the best individual. Regarding the CVA (figure 9) for the A64 run, the improvement of the criterion c_1 (measured on a short period) induces an improvement on the whole simulation. However, the decorrelation speed in the A32 run prevents making a clear comparison on the criterion, because the instant autocorrelation C_v^i rapidly oscillates around 0 in both cases, so we will now ignore this criterion for short chains. Now if we look at the CMD performances (figure 11), the global performance is in favour of the IEA runs in both cases. It is important to note here that when optimizing the CVA (c_1 criterion), both CVA and CMD are improved. This gives an experimental confirmation of the intuitive idea that these two criteria are correlated.

Looking at the ECF of moves (figure 10), we see that the reptation move is preferred in both cases. In addition, we can check that for each move a broad band of ECF values has been explored. The effect of the exploitation phase is an improved emergence of the histograms peaks. It is interesting to note on the A64 histograms, that a plurality of peaks reveals that the 'best individual' title changed during the exploitation phase. This behaviour can be explained by a greater variance of the A64 fitness scores.

For information we provide in table 4 the best individual of both runs. We declare the 'best individual' of a run as the one having the best f_H score at the end of the run, but with a sufficiently important weight w (see § 2.2.3). For the A32 run, we have $w = 1229$, $f_H = 0.26$ and for A64 $w = 478$, $f_H = 0.052$ (because an IEA run in these cases contains exactly 3200 evaluations, w is bounded in the [0; 3200] interval).

Finally, we notice that an improvement in terms of CVA and CMD is not necessarily linked to an improvement in CCL. Final values of these performance criteria are grouped in table 5.

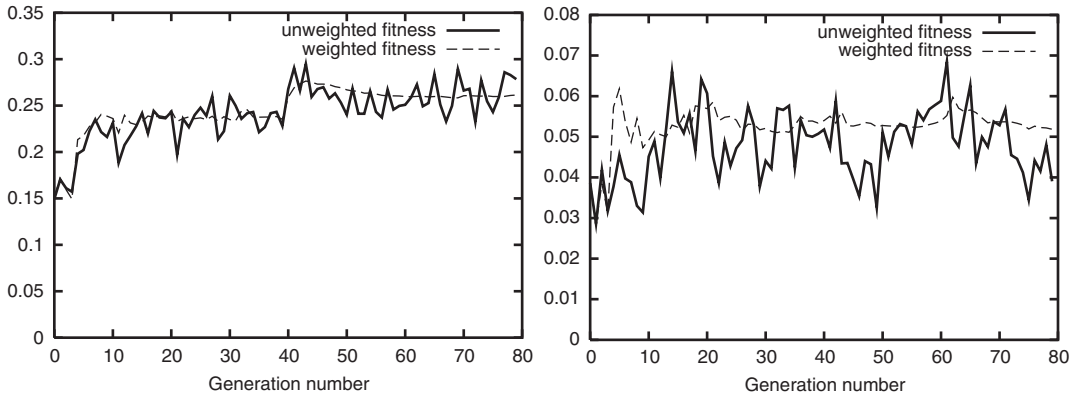

Figure 8. Runs A32 (left) and A64 (right): average weighted and unweighted fitness scores versus generations.

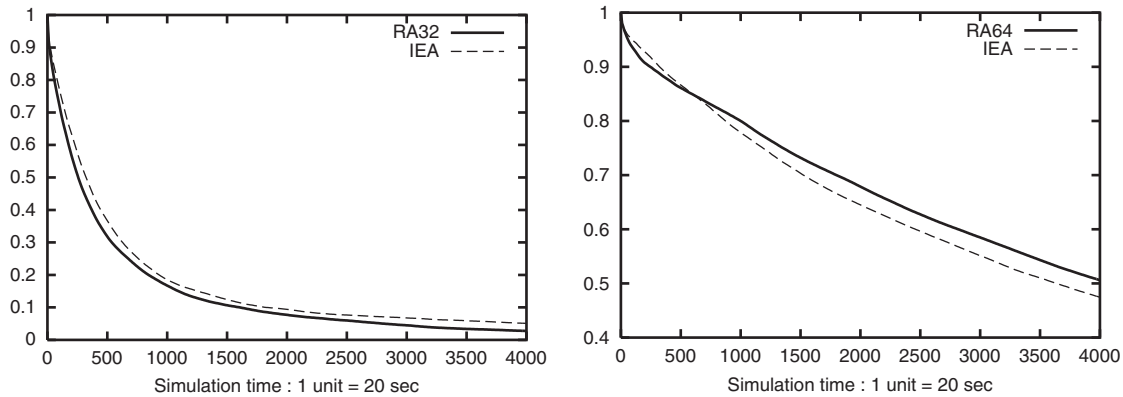

Figure 9. Runs A32 (left) and A64 (right): average (over the 32 systems) of the cumulated end-to-end vector autocorrelation measured over all the simulation time.

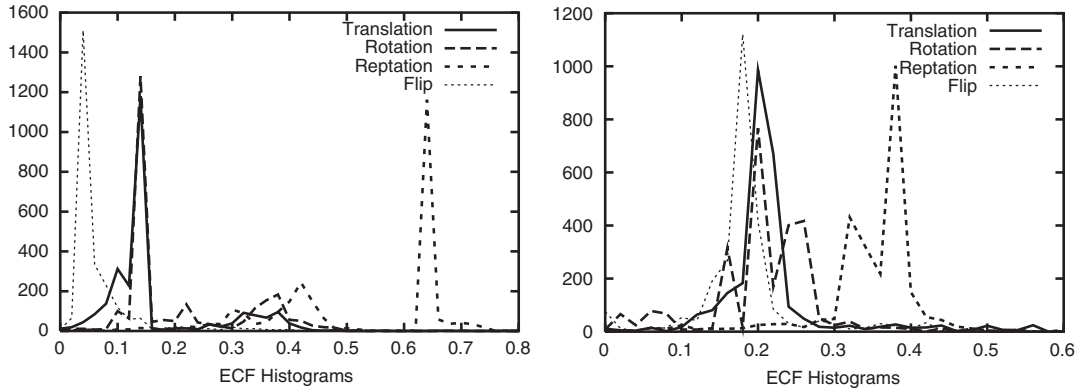

Figure 10. Runs A32 (left) and A64 (right): MC move ECF histograms.

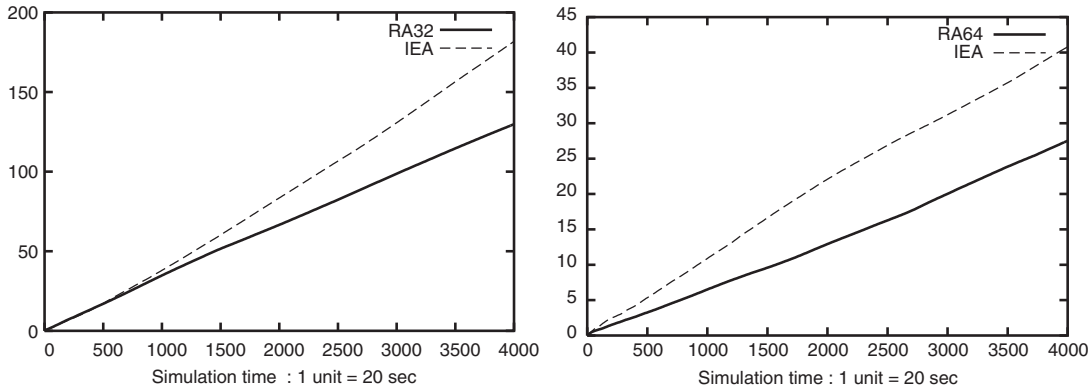

Figure 11. Runs A32 (left) and A64 (right): average (over the 32 systems) of the chain centre of mass mean square displacement measured over all the simulation time. Ordinates are in reduced units (1 unit $= \sigma_{LJ}^2$).

Table 4. Best frequencies obtained for the chain end-to-end vector autocorrelation criterion. Uniform ECF for RA (20% for each move).

	Run	Translation	Rotation	Reptation	Flip	VF
ECF (VF excluded)	A32	14.9%	14.3%	65.9%	4.9%	—
ECF	A32	11.9%	11.5%	52.7%	3.9%	20%
EF	A32	0.54%	16.8%	76.9%	5.7%	0.0456%
ECF (VF excluded)	A64	22.2%	24.8%	33.3%	19.7%	—
ECF	A64	17.8%	19.8%	26.6%	15.8%	20%
EF	A64	0.444%	31.7%	42.5%	25.3%	0.05%

3.3. *Centre of mass rms displacement criterion*

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

Remark: In order to limit the number of figures, we will now provide the table of criteria final values.

We see in table 6 that CVA and CMD performances are simultaneously improved here too, with an advantage for B64 compared to A64. The ECFs show that the reptation is still favoured with regard to this criterion, but still with a non-negligible contribution of other MC moves and in the case of B64 there are 'hesitations' between rotation and reptation. It reveals a fact outlined in [4]: if we want the reptation to have a significant effect, it is necessary to include MC moves that change end monomer environments. Once a reptation succeeds, it leaves a hole at the former position of the displaced monomer. If no other move is used in conjunction, it is highly probable that a 'reverse' reptation will succeed

Table 5. Criteria comparison table for runs A32 and A64. Bold values denote the best between RA and IEA.

Run	CVA	CMD (in σ_{LJ}^2)	CCL ($R = 3$)	CCL ($R = 7$)
RA 32	not relevant	129.8	3.515×10^{-4}	9.177×10^{-3}
A32 (IEA)	not relevant	**181.8**	$\mathbf{2.79} \times 10^{-4}$	$\mathbf{8.49} \times 10^{-3}$
RA 64	0.506	27.51	$\mathbf{1.737} \times 10^{-3}$	$\mathbf{4.115} \times 10^{-2}$
A64 (IEA)	**0.474**	**40.8**	1.973×10^{-3}	4.313×10^{-2}

Table 6. Criteria comparison table for runs B32 and B64. Bold values denote the best between RA and IEA.

Run	CVA	CMD (in σ_{LJ}^2)	CCL ($R = 3$)	CCL ($R = 7$)
RA 32	not relevant	129.8	3.515×10^{-4}	$\mathbf{9.177} \times 10^{-3}$
B32 (IEA)	not relevant	**180.8**	$\mathbf{3.476} \times 10^{-4}$	9.482×10^{-3}
RA 64	0.506	27.51	$\mathbf{1.737} \times 10^{-3}$	$\mathbf{4.115} \times 10^{-2}$
B64 (IEA)	**0.428**	**48.68**	2.441×10^{-3}	5.811×10^{-2}

Table 7. Criteria comparison table for runs C32 and C64. Bold values denote the best between RA and IEA.

Run	CVA	CMD (in σ_{LJ}^2)	CCL ($R = 3$)	CCL ($R = 4$)
RA 32	not relevant	129.8	3.515×10^{-4}	1.02×10^{-3}
B32 (IEA)	not relevant	**162.8**	$\mathbf{2.783} \times 10^{-4}$	$\mathbf{8.84} \times 10^{-4}$
RA 64	**0.506**	27.51	$\mathbf{1.737} \times 10^{-3}$	5.174×10^{-3}
B64 (IEA)	0.543	**31.06**	1.755×10^{-3}	$\mathbf{4.222} \times 10^{-3}$

Table 8. Criteria comparison table for runs D32 and D64. Bold values enote the best between RA and IEA.

Run	CVA	CMD (in σ_{LJ}^2)	CCL ($R = 6$)	CCL ($R = 7$)	CCL ($R = 8$)
RA 32	not relevant	129.8	3.393×10^{-3}	9.177×10^{-3}	1.64×10^{-2}
B32 (IEA)	not relevant	**164.4**	$\mathbf{3.356} \times 10^{-3}$	$\mathbf{7.87} \times 10^{-3}$	$\mathbf{1.46} \times 10^{-2}$
RA 64	**0.506**	**27.51**	2.044×10^{-2}	4.116×10^{-2}	7.186×10^{-2}
B64 (IEA)	0.551	25.2	$\mathbf{1.256} \times 10^{-2}$	$\mathbf{2.45} \times 10^{-2}$	$\mathbf{3.97} \times 10^{-2}$

with the help of this unchanged hole: the two successful reptations will have a negligible effect on the system mixing. Once again results show that the CCL criteria are not correlated to CVA and CMD.

3.4. *Lacunarity criterion*

$$f(\mu, \omega) = c_3(\mu, \omega) = \left(c_L^3(\mu, \omega) + c_L^4(\mu, \omega)\right).$$

The simulation box is divided into 16 or 27 cells, for which the c_3 criterion will measure the decrease in density variance. Looking at table 7 we see that the global performance is improved regarding the CCL criteria in comparison to the RA. But the CVA and CMD are less improved than the previous runs, and furthermore the B64 is less efficient than the RA regarding the CVA, and that can be explained by the predominance given to the translation move, with an ECF varying roughly between 30% and 50%.

$$f(\mu, \omega) = c_4(\mu, \omega) = \left(c_L^6(\mu, \omega) + c_L^7(\mu, \omega) + c_L^8(\mu, \omega)\right).$$

At this scale the IEA seems to better improve the CCL criteria (see table 8). As for the previous run, and more clearly in this case, the translation is the most favoured, with the highest ECF (roughly ranging from 40% to 70%). But this time the CVA and CMD performances for the long chains are inverted. From the runs C and D, it appears that this move is efficient for small frequent displacements, but has a weak effect on the global conformation of long chains. Concerning short chains, it seems that the translation succeeds in still bringing some improvement concerning the CMD criterion.

3.5. *Discussion of results*

Comparing our different numerical simulations, we can now sum up and make several remarks. First of all,

it appears that chain end-to-end vector autocorrelation c_1 and centre of mass displacement c_2 criteria are closely linked (and this is not surprising): if one of them is improved the other is also improved. But when the simulations are optimized with the lacunarity criterion, the result is an improvement for this criterion but also a degradation of other criteria. Even if these results suggest a kind of opposition between these criteria, they do not provide a definitive answer. For example it would be worth developing a multi-objective optimization [15] scheme using both a 'chain criterion' (c_1 or c_2) and a lacunarity criterion.

Another interesting aspect is the difference in the most favoured MC moves depending on the criterion. Even if the reptation appears to be more favoured when a chain criterion is used, surprisingly the translation appears to be efficient for the lacunarity criterion (i.e. it produces faster local fluctuations of monomer positions). But perhaps the more interesting aspect is that, even if there are clearly preferred moves depending on the criterion, none of our candidate moves is discarded during the simulation. This fact outlines the importance of combining complementary moves even if some of them could be individually considered as less efficient.

Finally we also insist on the comparison of the systems with short (32 monomers) or long (64 monomers) chains. In each of our simulations a cell contains the same number of monomers, but the effect of the chain length is substantial. Of course a system with longer chain mixes slower, but our simulation showed that there is also a difference in the resulting best frequencies set given by the evolutionary algorithm. For example in the case of chain criteria, reptation is given more importance with shorter chain systems. This means that depending on the type of system and its physico-chemical conditions, good frequencies for MC moves may vary significantly. And this fact is also supportive of our method, because it lets the evolutionary algorithm automatically find an adapted combination of moves.

4. Conclusions

In this article, we turned the problem of sampling efficiency of molecular simulation into an optimization problem. We started with the identification of numerical criteria and the available free parameters (the relative frequencies of MC moves). We designed a new evolutionary algorithm, well suited to solving this particular class of optimization problem. We verified the improvement of the efficiency of our polyethylene simulations brought about by this IEA algorithm. Furthermore, this improvement was tested with respect to different efficiency criteria.

Our numerical experiments, implying four simple moves (in addition to the volume fluctuation), have shown that this improvement does not rely only on a particular move, but also on a good combination of all available moves. In this case, it appears that reptation is the more efficient at moving and changing orientations of chains, and translation is the more efficient at producing rapid local density variations, corresponding to our lacunarity criterion.

Furthermore we observed that 'chains' criteria and the lacunarity criteria were not correlated, especially in the case of long chains. This suggests that these criteria carry different information about how the system is mixing, and it would be worth applying multi-objective techniques in order to improve simultaneously these complementary criteria.

Even if the numerical experiments presented here are limited to a specific molecular model in specific conditions, our algorithm can be easily extended. In fact, it could be used to test various MC moves in a common framework. In its generality it can be applied to any Monte Carlo scheme where a set of free parameters (having no consequences on the limit distribution) can be used to improve the efficiency, through the use of an appropriate numerical criterion. For that reason we will report in a forthcoming paper on the extension of our approach to parallel tempering.

References

[1] Siepmann, J. J., and Frenkel, D., 1992, *Molec. Phys.*, **75,** 59.
[2] Pant, P. V. K., and Theodorou, D. N., 1995, *Macromolecules*, **28,** 7224.
[3] Consta, S., Vlugt, T. J. H., Wichers Hoeth, J., Smit, B., and Frenkel, D., 1999, *Molec. Phys.*, **97,** 1243.
[4] Mavrantzas, V. G., Boone, T. D., Zervopoulou, E., and Theodorou, D. N., 1999, *Macromolecules*, **32,** 5072.
[5] Frenkel, D., and Smit, B., 1996, *Understanding Molecular Simulation: From Algorithms to Applications* (London: Academic Press).
[6] Jerrum, M., and Sinclair, A., 1996, The Markov Chain Monte Carlo method: an approach to approximate counting and integration, *Approximation Algorithms for NP-hard Problems*, edited by D. Hochbaum (Boston: PWS), pp. 482–520.
[7] Levy-Vehel, J., 1990, About lacunarity, some links between fractal and integral geometry, and application to texture segmentation. Technical Report RR-1188, Institut National de Recherche en Informatique et Automatique.
[8] Mandelbrot, B. B., 1982, *The Fractal Geometry of Nature* (San Francisco, CA: Freeman).
[9] Schoenauer, M., and Michalewicz, Z., 1997, *Control Cybernetics*, **26,** 307.
[10] Holland, J., 1975, *Adaptation in Natural and Artificial Systems* (Ann Arbor: University of Michigan Press).

[11] Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading, MA: Addison-Wesley).
[12] Rechenberg, I., 1973, *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution* (Stuttgart: Frommann-Holzboog).
[13] Koza, J. R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Evolution* (Cambridge, MA: MIT Press).
[14] Leblanc, B., Lutton, E., Braunschweig, B., and Toulhoat, H., 2001,. History and immortality in evolutionary computation, *Proceedings of EA'01: The 5th International Conference on Artificial Evolution*, Lecture Notes in Computer Science, Le Creusot, France, October 2001 (Berlin: Springer).
[15] Fonseca, C. M., and Fleming, P. J., 1995, *Evolutionary Computation*, **3,** 1.