Fifth Colloquium on Mathematics and Computer Science DMTCS proc. **AI**, 2008, 283–294

A variant of the Recoil Growth algorithm to generate multi-polymer systems

Florian Simatos[1]

[1] *INRIA Paris-Rocquencourt, Domaine de Voluceau, 78153 Le Chesnay, France*

The Recoil Growth algorithm, proposed in 1999 by Consta *et al.*, is one of the most efficient algorithm available in the literature to sample from a multi-polymer system. Such problems are closely related to the generation of self-avoiding paths. In this paper, we study a variant of the original Recoil Growth algorithm, where we constrain the generation of a new polymer to take place on a specific class of graphs. This makes it possible to make a fine trade-off between computational cost and success rate. We moreover give a simple proof for a lower bound on the irreducibility of this new algorithm, which applies to the original algorithm as well.

Keywords: Metropolis, multi-polymer systems

1 Introduction

Designing an algorithm that efficiently samples from a multi-polymer system according to a given probability distribution is the focus of much research activity in chemical physics [CVH+99, Sie90]. The state of a multi-polymer system being a collection of self-avoiding paths that do not overlap each other, this problem is closely related to the classical problem in computer science of generating self-avoiding paths. The state space of such systems is huge, especially in high dimension or if the paths are long, which makes these problems hard. The main issue consists of defining an algorithm that both keeps the computational cost low and converges rapidly to the sampling distribution. For multi-polymer systems, various approaches have been suggested to tackle this problem, among which is the Recoil Growth (RG) algorithm, meant to be one of the most efficient algorithm currently available in the literature. For a precise description of this algorithm, the reader is referred to [CWFA99]. In the present paper, we define and analyze a variant of the RG algorithm, to which we refer as the RG* algorithm. The main ideas of both the RG and the RG* algorithms are to be found in two important classes of algorithms, which we briefly describe below: the Metropolis algorithm [MRR+53] and the auxiliary variable method [BG93, Hig98].

The Metropolis algorithm is a very generic algorithm that approximately samples according to a probability distribution π defined on some finite state space $\mathcal{X}$. It takes as other input a Markov Chain P, and constructs a reversible Markov Chain with π as stationary distribution. In the long-run, we can therefore sample from a distribution that is arbitrarily close to π. The implicit idea is that it is hard to sample from π, whereas it is easy to generate P, for instance by local modifications of a state. We do not aim at giving a precise description of the Metropolis algorithm, but the main idea is that the Markov Chain P acts as an exogenous process that, at each step, proposes a candidate for the next step. This candidate is then accepted or rejected according to some rejection procedure, which usually amounts to toss a coin with a suitable probability. In general, the stationary distribution of P is not π, and so the rejection procedure compensates this bias: by suitably rejecting or accepting the candidate, the new Markov Chain can be shown to be reversible with π as stationary distribution. Reversibility is a very important feature, because it makes it straightforward to show that π is indeed the stationary distribution. Otherwise, since the objects under consideration are fairly complex, proving stationarity is usually very challenging.

The auxiliary variable method is a conceptually easy extension of the Metropolis algorithm: instead of sampling from $\mathcal{X}$ according to π, it is sometimes easier to sample from a pair $\mathcal{X} \times \mathcal{Y}$ according to some distribution $\widetilde{\pi}(\cdot, \cdot)$ which has π as marginal. In such cases, the idea is to apply the Metropolis algorithm to $\widetilde{\pi}$, and then to recover π by summation. The new variable $y \in \mathcal{Y}$ is referred to as the *auxiliary variable*.

In this paper, we combine these two ideas to define the RG* algorithm. We prove that it achieves the correct sampling distribution, and give some results about its irreducibility. Since the RG and the RG* algorithms have many similarities, the results on the irreducibility hold for both algorithms. No such result was previously known for the RG algorithm.

2 Main ideas and notations

The state space of the RG* algorithm is the set of all possible configurations of non-intersecting paths of length L, or *polymers*, on a finite d-dimensional torus $\mathcal{G}$, to which we refer as the *underlying torus*. For simplicity, we assume that $\mathcal{G}$ is cubic, i.e., its set of edges is $(\mathbb{Z}/a\mathbb{Z})^d$ for some parameter $a \in \mathbb{N}^*$. There are N polymers in the system, and we note $Q = 2d$ the number of neighbors of each vertex (in $\mathcal{G}$). The density $\rho = NL/a^d$ of the system is the number of nodes occupied by the N polymers divided by the total number of nodes. Finally, q denotes the sampling distribution, i.e., we aim at defining a Markov Chain on the state space of the RG* algorithm for which q is the stationary distribution. Note that as soon as $NL \leq a^d$, there exists a configuration of the N polymers on $\mathcal{G}$. Figure 1 shows an example of a two-dimensional multi-polymer system with 100 polymers of length 25.

One step of both the RG and the RG* algorithms can be summarized as follows: starting from an old state, we remove a polymer chosen uniformly at random. Then, an attempt is made to grow a new polymer given the $N-1$ remaining polymers: as in the Metropolis algorithm, upon success of the growth procedure, this new polymer serves as a candidate for the next step. The probability of acceptance p of this new polymer, compared to the old state, is computed, and a coin is tossed: with probability p, the next state in the algorithm uses the new polymer, and with probability $1-p$, we use the old one. We will see that the cost of computing the correct probability of acceptance p is high, but the main difficulty lies in the generation of the new polymer. The variant RG* proposed here differs precisely from the original RG algorithm in the way the new polymer is generated. The algorithm that generates the candidate polymer is called the *growth procedure*.

During the growth procedure, the new polymer is grown step by step, i.e., vertices are added to, and removed from, the end of the current polymer one by one. Given $N-1$ polymers, the set of all possible partial polymers (i.e., of length strictly smaller than L) that do not intersect these polymers is huge, again, especially if the dimension d or the length L of the polymers is high. Thus the main problem is to define an

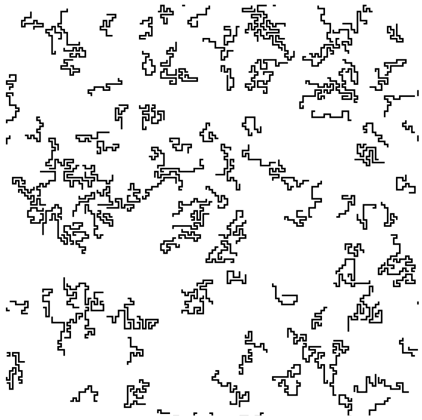

Fig. 1: Example of a state of a multi-polymer system on the two-dimensional torus $\mathcal{G} = \left(\mathbb{Z}/135\mathbb{Z}\right)^2$ with $N = 100$ polymers of length $L = 25$ each.

algorithm that grows a new polymer with both a reasonable computational cost and a reasonable success rate (the algorithm may indeed fail to grow a new polymer). The two main concepts of the RG* algorithm, the *feeler* and the *underlying graph*, characterized respectively by the two parameters $\ell \leq L$ and $k \leq Q$, make it possible to make a trade-off between the computational cost and the success rate. The idea is to restrict the possibilities for the growth of the current partial polymer.

2.1 Growth procedure and feeler

The concept of feeler makes it possible to avoid spending time in trying to grow a polymer in a dense area, in which the computational cost would be high in order to yield a good success rate. We have said that during the growth procedure, the partial polymer is grown, or shortened, one vertex at a time: we say that we *recoil* if the current polymer is shortened by removing the end vertex. This happens when no *allowed* neighbor of the current end vertex is a possible direction to grow the polymer: each allowed neighbor is either occupied by a polymer (be it the current polymer or another one), or it has already been tried previously. As we will see in the next section, the set of allowed neighbors is determined by the *underlying graph*: it is a subset of the set of Q neighbors of the current end vertex in $\mathcal{G}$.

The *feeler's length* ℓ is the number of steps the algorithm is allowed to look ahead: if the current partial polymer has length $i < L$, then it will never be possible to recoil back to a partial polymer of length less than $i - \ell$. In other words, if at some point, we need to recoil ℓ steps back, then the growth is a failure. Intuitively, such a scenario happens when the polymer is grown in a region with already many polymers. For $\ell = L$, we allow the current polymer to recoil all the way back to the original vertex, thus allowing the current polymer to visit all the set of partial polymers starting from this vertex. Whereas for $\ell = 0$, as soon as no allowed neighbor of the end vertex of the current polymer is a possible growth direction, the growth is unsuccessful, and the old polymer is put back in place. In the former case, the success rate is the best possible, and the computational cost may be immensely expensive; the opposite holds in the latter case.

2.2 Underlying graph

In the previous global description of the growth procedure, in which the concept of feeler plays a key role, we see that this procedure may be run on any graph: the idea is just that at each step, we look at a set of allowed neighbors of the current end vertex of the partial polymer. Among these allowed vertices, we may try to grow the partial polymer towards a vertex that is free (not occupied by any polymer), and that has not been tried previously (in order to avoid the algorithm to loop). The *underlying graph* is precisely the graph that defines this adjacency structure, and tells which neighbors are allowed during the growth procedure. It is a random directed subgraph of the underlying torus $\mathcal{G}$, such that each vertex has exactly k out-edges. The number k is a parameter of the algorithm. For $k = Q$, any underlying graph spans the whole underlying torus; for $k = 1$, an underlying graph is a directed path with a loop.

The underlying graph can be seen as an auxiliary variable, as previously described. Indeed, instead of growing a polymer directly on the underlying torus $\mathcal{G}$, we first generate an underlying graph at random, and then we try to grow a new polymer on this underlying graph. Thus the transition between two states $S_{\rm old}$ and $S_{\rm new}$ does not only depend on the two polymers $C_{\rm old}$ and $C_{\rm new}$ by which $S_{\rm old}$ and $S_{\rm new}$ differ: there is an additional dimension given by the underlying graph. The transition is made between two pairs $(S_{\rm old}, G_{\rm old})$ and $(S_{\rm new}, G_{\rm new})$, where $G_{\rm old}$ and $G_{\rm new}$ are underlying graphs. The underlying graph $G_{\rm new}$ is a necessary ingredient of the growth procedure, whereas $G_{\rm old}$ is generated in order to make the situation symmetric between the old and the new state. In particular, the probability of acceptance which corresponds to the Metropolis algorithm will be of the form $P_{\rm acc}((S_{\rm old}, G_{\rm old}), (S_{\rm new}, G_{\rm new}))$.

By growing the new candidate polymer on an underlying graph instead of $\mathcal{G}$, we forbid $Q-k$ directions at each step of the growth of the polymer. This reduces the success rate of the growth, since we could forbid directions that would have lead to a complete polymer. However, this reduces the computational cost as well, since it amounts to try to grow a polymer in dimension $k/2$ instead of d. In particular, as announced earlier, this makes it possible to make a trade-off between the computational complexity and the success rate of the growth procedure. We discuss in Section 4.3 an extension in which it is possible to tune the parameter k more finely by allowing it to take any real value between 1 and Q.

3 The RG* algorithm to generate multi-polymer systems

3.1 Overview

In Section 2, the main ingredients of the RG* algorithm are presented. The only point not mentioned is the way the acceptance probability $P_{\rm acc}((S_{\rm o}, G_{\rm o}), (S_{\rm n}, G_{\rm n}))$ is computed, where from now on, we more simply refer to the old and new states through the subscripts $\rm o$ and $\rm n$, respectively. We need a few definitions before entering more details:

Definition 1 *Let C be a polymer and G an underlying graph. The probability that the growth procedure run on G returns C is noted $P_{\rm g}(C|G)$. We say that C and G are* compatible *if $P_{\rm g}(C|G) > 0$, in which case we define the* weight $W(C|G)$ *as the inverse of this probability: $W(C|G) = 1/P_{\rm g}(C|G)$.*

We show in Section 4.1 that the correct choice for the probability of acceptance is given by the formula

$$P_{\rm acc}\big((S_{\rm o}, G_{\rm o}), (S_{\rm n}, G_{\rm n})\big) = \min\left(1, \frac{q(S_{\rm n})W(C_{\rm n}|G_{\rm n})}{q(S_{\rm o})W(C_{\rm o}|G_{\rm o})}\right). \tag{1}$$

As the next pseudo-code overview of one step of the RG* algorithm shows, $P_{\rm g}(C_{\rm o}|G_{\rm o})$ and $P_{\rm g}(C_{\rm n}|G_{\rm n})$ are always positive, so that the weights are well defined in (1):

```
1. Choose a polymer $C_o$ w.p. $1/N$ and remove it from the system
2. Generate at random an underlying graph $G_n$ w.p. $P_u(G_n)$
3. Run the growth procedure on $G_n$
   (a) If the procedure did not successfully grow a new polymer:
       i. Put back $C_o$
      ii. Go to 1.
   (b) Else, we have grown a new polymer $C_n$ w.p. $P_g(C_n|G_n)$:
       i. Generate at random an underlying graph $G_o$ compatible with
          $C_o$ w.p. $P_c(G_o|C_o)$
      ii. Compute the weights $W(C_n|G_n)$ and $W(C_o|G_o)$
     iii. Set $p = P_{acc}((S_o,G_o),(S_n,G_n))$, flip a $p$-coin:
          A. If heads, add $C_n$ to the polymer system
          B. If tails, put back $C_o$
      iv. Go to 1.
```

So $W(C_\mathrm{n}|G_\mathrm{n})$ is well defined since C_n is the result of the growth procedure run on G_n, whereas $W(C_\mathrm{o}|G_\mathrm{o})$ is well defined by construction of G_o. It is now straightforward to write down the transition probability $P(S_\mathrm{o} \to S_\mathrm{n})$ where S_o and S_n differ by only one polymer C_o in S_o and C_n in S_n:

$$P(S_\mathrm{o} \to S_\mathrm{n}) = \frac{1}{N} \sum_{G_\mathrm{n},\, G_\mathrm{o}} P_\mathrm{u}(G_\mathrm{n}) P_\mathrm{g}(C_\mathrm{n}|G_\mathrm{n}) P_\mathrm{c}(G_\mathrm{o}|C_\mathrm{o}) P_\mathrm{acc}\big((S_\mathrm{o},G_\mathrm{o}),(S_\mathrm{n},G_\mathrm{n})\big). \tag{2}$$

A complete description of the RG* algorithm, that fills in the gap of this global overview, is as follows: (i) describe the class of underlying graphs and provide algorithms to generate them; (ii) explain in details the growth procedure, and finally (iii) explain the computation of the weight of a polymer on an underlying graph. The next sections deal with these points successively.

3.2 Definition and generation of underlying graphs

Definition 2 *An underlying graph $G \subset \mathcal{G}$ is a directed subgraph of $\mathcal{G}$ rooted at a vertex r such that:*

1. *for any vertex $v \in G\backslash\{r\}$, there exists a directed path from r to v*
2. *each vertex has exactly k out-edges.*

Note that several vertices may satisfy the same property as the root, namely accessibility to all other vertices. However, the root plays a special role, as it will be the starting vertex for the growth of the candidate polymer. In the following proposition, which gives an alternative definition for the compatibility between a polymer and an underlying graph, we see a polymer as a directed path. We orientate the polymer by choosing one of the two end vertices, which then serves as a source, i.e., the polymer is seen as a directed path starting at this vertex.

Proposition 1 *A polymer C and an underlying graph G are compatible if and only if the root r of G is one of the two extremities of C and if $C \subset G$, where the choice of r induces the orientation on C.*

This proposition is obvious in light of the description of the growth procedure given in Section 3.3, because the growth procedure always starts growing a polymer from the root of the underlying graph, and because an edge of C is necessarily an out-edge of G.

We now turn our attention to the generation of underlying graphs. First we describe how to generate an underlying graph without constraint, which corresponds to step `2` in the above pseudo-code description of the RG* algorithm. The procedure is then easily adapted to generate an underlying graph constrained to be compatible with a given polymer (what is needed at step `3-b-i`).

To generate an underlying graph, we proceed iteratively, taking care that every vertex added to the graph has exactly k out-edges. This algorithm is basically a greedy algorithm that at each step assigns k out-edges to some randomly chosen vertex.

We have two sets V and W: V is the set of vertices that will be part of the final underlying graph that we are trying to generate, whereas W is a waiting room for the vertices that will eventually end up in V, but to which no out-edge has been assigned yet. Initially, we choose the root r uniformly at random in the set of vertices that are not occupied by any of the $N-1$ polymers in the system. Thus $W=\{r\}$ and $V=\emptyset$ because we have not assigned out-edges to the root yet. The first step is to assign k out-edges to r: r has Q neighbors in $\mathcal{G}$, and we choose k distinct out-edges out of these Q possible uniformly at random. Call $v_1,\dots,v_k$ the corresponding vertices chosen. Then we have assigned out-edges to r but not to the k new vertices, so $W=\{v_1,\dots,v_k\}$ and $V=\{r\}$. Then we take any vertex in W, say v_1, and we choose k out of its Q neighbors. Since r and v_1 are neighbors in $\mathcal{G}$, it may happen that r is among these chosen neighbors, so we have to be careful: we add to W only the vertices chosen that are not already in V or in W, and we move v_1 from W to V. We keep on assigning out-edges to vertices in W this way until $W=\emptyset$. When W is empty, each vertex in V has exactly k out-edges, and there exists a directed path from r to any vertex: we have indeed generated an underlying graph.

Since the underlying torus $\mathcal{G}$ is finite, it is clear that this procedure terminates in finite time. Moreover, it is easy to get convinced that this procedure does not depend on the order in which vertices are picked up in W: once a vertex is in W, its set of out-edges does not depend on when it is chosen. This algorithm readily yields the following proposition, since the generation of an underlying graph is unambiguous:

Proposition 2 *The probability $P_{\mathrm{u}}(G)$ to generate the underlying graph G is given by*

$$P_{\mathrm{u}}(G)=\frac{\alpha^{|G|}}{\gamma} \tag{3}$$

where $\alpha=\binom{Q}{k}^{-1}$, $|G|$ is the number of vertices in G, and $\gamma=a^d-(N-1)L$ is the number of free vertices when $N-1$ polymers are in the system.

When we add the constraint of generating an underlying graph compatible with a given polymer C, we only have to add two minor modifications to the previous algorithm thanks to Proposition 1. First, the root of the underlying graph is chosen at random between the two end vertices of C. Moreover, each time we assign out-edges to a vertex that belongs to $C=(v_1,\dots,v_L)$, we must take care that for each $1\le i\le L-1$, we do have the edge $v_i\to v_{i+1}$ in the set of out-edges. Hence for these $L-1$ vertices, one out-edge is imposed, and we have to choose the $k-1$ others among $Q-1$ possible. Hence we can derive the following property similarly as for general underlying graphs:

Proposition 3 *Given a polymer C of length L, the probability $P_{\mathrm{c}}(G|C)$ to generate the underlying graph*

G compatible with C is given by

$$P_{\mathrm{c}}(G|C) = \frac{\alpha^{|G|-L+1}\beta^{L-1}}{2} \tag{4}$$

where $\beta = \binom{Q-1}{k-1}^{-1}$ and α and $|G|$ are as in Proposition 2.

3.3 The growth procedure

During the growth procedure, the partial polymer $\overline{C} = (v_1, \ldots, v_i)$ of length $i < L$, being grown, can always be decomposed into two disjoint parts, that evolve in time: the *fixed part* and the *feeler*. Before defining this decomposition, it is important to have in mind the rule that it stands for:

Rule: *During the growth procedure, we are allowed to recoil from a vertex only if it is part of the feeler.*

This fits the intuition given previously on the feeler, which was presented as a way for the algorithm to look a few steps ahead: the feeler is the retractable part of the current polymer. This rule implies that if at some point we need to recoil from a vertex that belongs to the fixed part, the attempt is a failure. Since the feeler is the complementary part of the fixed part, one only needs to define the fixed part:

Definition 3 *A node v_j is in the fixed part if $j = 1$ or if during the history of the growth of $\overline{C}$, there was a partial polymer of the form $(v_1, \ldots, v_j, v_{j+1}, \ldots, v_{j+\ell})$.*

Equivalently, the fixed part is the set of vertices v such that either v is the starting vertex of the polymer, or at some point in the growth history, a partial polymer has been grown through v and has been ℓ steps further. The previous definition yields the following property:

Proposition 4 *There exists an increasing index Δ such that at any time during the growth procedure, the fixed part is of the form $(v_1, \ldots, v_\Delta)$ and the feeler of the form $(v_{\Delta+1}, \ldots, v_i)$. Moreover, the feeler is of length at most ℓ, and if $L_{\max}$ is the length of the longest polymer ever grown at any time during the growth procedure, then $\Delta = \max(1, L_{\max} - \ell)$.*

Proof: Consider a vertex v_j in the fixed part, and the corresponding partial polymer $\overline{C} = (v_1, \ldots, v_j, v_{j+1}, \ldots, v_{j+\ell})$. Since the growth procedure extends or shortens a partial polymer only one vertex at a time, this implies that necessarily, at some point, the shorter polymer $(v_1, \ldots, v_{j-1}, v_j, \ldots, v_{j+\ell-1})$ was grown: this exactly means that $j - 1$ is in the fixed part as well, i.e., the fixed part is indeed of the form $(v_1, \ldots, v_\Delta)$.

Moreover, by definition of the fixed part, a vertex stays in the fixed part for the remainder of the growth procedure if it belongs to it at some point. Thus the fixed part is increasing, and so is the index Δ.

Concerning the feeler's length, imagine for a moment that the feeler is of length $\ell+1$, so that the partial polymer $\overline{C}$ can be written $(v_1, \ldots, v_\Delta, v_{\Delta+1}, v_{\Delta+2}, \ldots, v_{\Delta+\ell+1})$. But then, by definition of the fixed part, the vertex $\Delta + 1$ should belong to the fixed part, which contradicts the definition of Δ.

We can now draw a picture of the way the fixed part and the feeler evolve. If $\overline{C} = (v_1, \ldots, v_i)$ grows towards v_{i+1}, the feeler is increased except when it was already of length ℓ. If the feeler is of length ℓ when $\overline{C}$ grows, then the feeler stays of length ℓ, and the fixed part grows. It is then straightforward to derive the formula $\Delta = \max(1, L_{\max} - \ell)$. □

We now have all the necessary ingredients to give a precise description of the growth procedure.

Given an underlying graph G, the growth procedure keeps record of the current partial polymer $\overline{C}$, the length $L_{\max}$ of the longest partial polymer ever grown so far, and L sets D_i for $i = 1, \dots, L$. For a partial polymer of length at least i, D_i represents the set of vertices that the ith vertex v_i of $\overline{C}$ has already tried as directions of growth. In particular, at any time, D_i is a subset of the set of (out-)neighbors of v_i in G. If at some point, v_i is the end vertex of the current polymer and every neighbor of v_i in G is in D_i (equivalently, $|D_i| = k$), this means that we have to recoil from v_i, because all the possible growth directions have already been tried. Recoiling is allowed only when v_i is in the feeler, which, by Proposition 4, is verified by checking the simple condition $i > \max(1, L_{\max} - \ell)$.

A pseudo-code description of the growth procedure run on the underlying graph G is given next:

---

1. Initialization:
   (a) $\overline{C} = (v_1)$, where v_1 is the root of G
   (b) $D_1 = \emptyset$
   (c) $L_{\max} = 1$
2. At each step, with $\overline{C} = (v_1, \dots, v_i)$:
   (a) If $i = L$, stop, $\overline{C}$ is a full polymer
   (b) Else if $|D_i| = k$, recoil:
      i. If $i > \max(1, L_{\max} - \ell)$, set $\overline{C} = (v_1, \dots, v_{i-1})$ and go to 2
      ii. Otherwise, stop, the generation has failed
   (c) Else, keep picking uniformly at random a vertex v neighbor of v_i in G and not in D_i, add it to D_i, and stop when either v is unoccupied or $|D_i| = k$:
      i. If v is unoccupied, set $\overline{C} = (v_1, \dots, v_i, v)$, $D_{i+1} = \emptyset$, update $L_{\max}$ (if necessary) and go to 2
      ii. Else, $|D_i| = k$, and recoil as specified in 2.(b)

3.4 Weight computation

In order to complete in details the overview of the RG* algorithm given in Section 3.1, all is left to do is explain how the probability of acceptance $P_{\text{acc}}((S_\text{o}, G_\text{o}), (S_\text{n}, G_\text{n}))$ is computed. Equivalently, because of (1), this amounts to explain how to compute the weight $W(C|G)$ (or its inverse, $P_\text{g}(C|G)$) of a polymer C compatible with an underlying graph G.

To get an insight of what the right answer is, we begin by inspecting the case $\ell = 0$: we write $C = (v_1, \dots, v_L)$, and consider any $1 \leq i \leq L - 1$. In the process of growing C, we have necessarily grown at some point the partial polymer $\overline{C} = (v_1, \dots, v_i)$, and from v_i, we have had k choices. But since $\ell = 0$, any choice different than v_{i+1} will not lead to C. Indeed, if we choose $v' \neq v_{i+1}$ as growth direction for $\overline{C}$, since $\ell = 0$, we will never recoil below v', and we will not grow C. In opposition, if we choose v_{i+1} for each $1 \leq i \leq L - 1$, we will grow C: this happens with probability $(1/k)^{L-1}$, which is the answer for $\ell = 0$.

In the case $\ell > 0$, we have still necessarily grown at some point the partial polymer $(v_1, \dots, v_i)$, and from v_i, there were still k potential choices. But in this case, if we choose a vertex $v \neq v_{i+1}$, we may still eventually grow C, because we may recoil back to v_i. In this case, and when $k > 1$, we then have a second opportunity to pick the right vertex v_{i+1}.

Since $\ell > 0$, when we grow from v_i to v, v is in the feeler part, and we will eventually recoil back to v_i if and only if v never belongs to the fixed part. Since by definition, v belongs to the fixed part if and only if at some point, we grow the chain ℓ steps further, we get that we will recoil back to v_i if and only if the partial polymer $(v_1, \ldots, v_i, v)$ cannot be extended ℓ steps further on G. Note that this equivalence is true only for vertices that may potentially belong to the fixed part, i.e., for vertices v_i with $i \leq L - \ell$.

For $i \geq L - \ell + 1$, we recoil back to v_i if and only if the partial polymer $(v_1, \ldots, v_i, v)$ cannot be completed to a full polymer. This yields the following natural definition:

Definition 4 *For* $1 \leq i \leq L - 1$, *a neighbor* v *of* v_i *is called* admissible *if either* $i < L - \ell$ *and the partial polymer* $(v_1, \ldots, v_i, v)$ *can be extended* ℓ *steps further, or if* $i \geq L - \ell$ *and* $(v_1, \ldots, v_i, v)$ *can be completed to a full polymer. The* elementary weight w_i *of* v_i *is the number of admissible neighbors of* v_i.

Note that when C and G are compatible, v_{i+1} is always admissible as neighbor of v_i, because we know that $(v_1, \ldots, v_L) = C$ is possible. And now we can compute the weight $W(C|G)$:

Proposition 5 *Given the underlying graph G compatible with a polymer C, we have*

$$W(C|G) = \prod_{i=1}^{L-1} w_i. \tag{5}$$

Proof: Let $i \in \{1, \ldots, L - 1\}$. In the process of growing C, the partial polymer was at some point $(v_1, \ldots, v_i)$. By definition, if we choose as next vertex v, then we will eventually recoil back to v_i if and only if v is not admissible. Hence the probability from v_i to make the right choice is one over the number of admissible neighbors, i.e., $1/w_i$. □

So in order to compute the weight $W(C|G)$, we need to compute the elementary weights w_i. For this, we have no choice but to use an exhaustive algorithm that tries all the possibilities to grow a self-avoiding path of length ℓ (that does not intersect the beginning of the polymer, nor the $N - 1$ other polymers), because we need to know wether such a path exists. So if ℓ is large, computing the weight can be computationally expensive. However, this is only to be done when we have successfully grown a polymer, in which case paying an extra cost is not too restrictive.

The description of the RG* algorithm is now complete, and we turn our attention to proving some of its properties.

4 Properties of the RG* algorithm

4.1 Reversibility

As in the Metropolis algorithm, showing that q is the stationary distribution is made by showing that it satisfies the detailed balance equations.

Proposition 6 *If the probability of acceptance* $P_{\mathrm{acc}}((S_{\mathrm{o}}, G_{\mathrm{o}}), (S_{\mathrm{n}}, G_{\mathrm{n}}))$ *is chosen according to Equation* (1)*, then the RG* algorithm is reversible with q as reversible distribution.*

Proof: Let S_{o} and S_{n} be two states that differ only by one polymer C_{o} in S_{o} and C_{n} in S_{n}: reversibility amounts to show the equality $q(S_{\mathrm{o}})P(S_{\mathrm{o}} \to S_{\mathrm{n}}) = q(S_{\mathrm{n}})P(S_{\mathrm{n}} \to S_{\mathrm{o}})$. Starting from Equation (2), we actually show a stronger condition: namely, we write $NP(S_\mathrm{o} \to S_\mathrm{n}) = \sum_{G_\mathrm{o},G_\mathrm{n}} P(S_\mathrm{o}, G_\mathrm{o}, S_\mathrm{n}, G_\mathrm{n})$ where

$$P(S_\mathrm{o}, G_\mathrm{o}, S_\mathrm{n}, G_\mathrm{n}) = P_\mathrm{u}(G_\mathrm{n})P_\mathrm{g}(C_\mathrm{n}|G_\mathrm{n})P_\mathrm{c}(G_\mathrm{o}|C_\mathrm{o})P_\mathrm{acc}\big((S_\mathrm{o}, G_\mathrm{o}), (S_\mathrm{n}, G_\mathrm{n})\big).$$

Then the "local" equality

$$q(S_\mathrm{o})P(S_\mathrm{o}, G_\mathrm{o}, S_\mathrm{n}, G_\mathrm{n}) = q(S_\mathrm{n})P(S_\mathrm{n}, G_\mathrm{n}, S_\mathrm{o}, G_\mathrm{o})$$

holds, that implies that q is the reversible probability distribution. To check this last equality, one just has to plug in (1), (3) and (4), and to use the definition of the weight $W(C|G)$ as the inverse of $P_\mathrm{g}(C|G)$. □

4.2 Irreducibility

Knowing that q satisfies the detailed balance equations is not enough for the algorithm to work properly: one needs results on the irreducibility of the Markov Chain as well. This question is a hard problem, and we aim at giving some hints in this section. Note that the results and the discussion provided in this section apply to the original RG algorithm as well (for which they are new), because the irreducibility of the algorithm does not depend on the precise growth procedure, as long as this procedure works by replacing one polymer at a time. The proofs of some results are not given here, but can be found in the extended version of this paper [Sim07]. Recall that $\rho = NL/a^d$ is the density of the system.

Proposition 7 *For any d, N, L and $a \geq L$, the chain is irreducible if $\rho \leq \lfloor a/L \rfloor / a$.*

Proof: The idea is to show that from any configuration S, we can reach a particular configuration $\widetilde{S}$ where all the polymers are straight, and lie in some specific boxes.

Recall that the set of vertices of the underlying torus $\mathcal{G}$ is equal to $(\mathbb{Z}/a\mathbb{Z})^d$. For any $x \in (\mathbb{Z}/a\mathbb{Z})^{d-1}$, we consider the subset $F_x = \{(y_i) \in \mathcal{G} : (y_1, \dots, y_{d-1}) = x, y_d \in a\mathbb{Z}\}$ where the $d-1$ first coordinates are equal to x. The set F_x is in bijection with $\mathbb{Z}/a\mathbb{Z}$. Since $a \geq L$, we write $a = nL + r$ with $n = \lfloor a/L \rfloor \geq 1$ and $0 \leq r \leq L-1$. For each x, we can then divide F_x into n boxes, the first box being in bijection with $\{0, \dots, L-1\}$, the second with $\{L, \dots, 2L-1\}$, etc... The total number of boxes B is equal to $a^{d-1}n$, therefore we get, using the definition of ρ and the hypotheses:

$$\frac{B}{NL} = \frac{n}{\rho a} \geq 1.$$

But NL is the number of occupied vertices, so the previous inequality means that we have more boxes than occupied vertices. One of two things may then happen.

If each box is occupied by exactly at most one vertex, then each box is intersected by exactly one polymer. So we can consider each polymer in turn, and put each one in some box that it occupies. Each polymer will then be straight and lie in some box.

If some boxes are occupied by more than one vertex, then necessarily, some boxes are empty. Then we can consider the polymers in turn and assign them to empty boxes while we have empty boxes. We stop either when we have no more empty boxes or when every polymer has been assigned to a box. In the latter case, we are done. And it is easy to see that the former case never happens, because by putting a polymer in a box, we create empty boxes.

In any case, we can go to a state where all the polymers are in boxes. Such states are connected, and the chain is therefore irreducible. □

With regards to simulation, the longer the polymers, the better. In [CWFA99], the authors investigate cases when $L = 100$. So this result insures irreducibility at very low densities, around 0.01, compared to the values we would be interested in, like 0.6. Moreover, one question still open at this point is to give a lower bound independent of the parameters of the system. For instance, it seems a reasonable bet that the algorithm is always irreducible for $\rho = 10^{-10}$.

The best result one could hope for would be to give a criterion that tells whether $\mathrm{RG}^*_d(N, L, a, \rho)$, which denotes the RG^* algorithm with parameters d, N, L, a and ρ, is irreducible or not. The next proposition (see [Sim07] for the proof) shows that the simplest criterion, which would be the existence of a cutoff density ρ^* such that the algorithm is irreducible if and only if $\rho < \rho^*$, is not true:

Proposition 8 *For any $\rho < 1$ and any other parameters d, N, L and a, $RG^*_d(N, 2, a, \rho)$ is irreducible. However, $RG^*_2(N, 5, a, 80/81)$ is not irreducible.*

These two simple examples tend to show that the critical parameter is actually the length of the polymers, and so one could wonder whether for fixed length, and for fixed dimension d, the density is not enough to characterize the irreducibility. This idea, in addition with the intuition that the longer the polymers, the harder it is for the system to be irreducible, is the object of the following conjecture:

Conjecture 1 *If $RG^*_d(N, L, a, \rho)$ is irreducible, then $RG^*_d(N', L, a', \rho')$ is irreducible for any N' and a' such that $\rho' < \rho$. Thus the quantity $\rho_d(L) = \sup_{N,a}\{\rho : \mathrm{RG}^*_d(N, L, a, \rho)$ is irreducible$\}$ is well defined, and is decreasing in L.*

Finally, we state a more natural, and probably easier, conjecture: if the system is irreducible, then removing polymers, shortening the polymers or growing the box should leave the system irreducible.

Conjecture 2 *Suppose that $RG^*_d(N, L, a, \rho)$ is irreducible. Then $RG^*_d(N', L', a', \rho')$ is irreducible for any $N' \leq N, L' \leq L$ and $a \geq a'$, where the value of $\rho' \leq \rho$ is determined by the other parameters.*

A natural idea to try to prove this last conjecture is the following: suppose for instance that one is interested in showing that $\mathrm{RG}^*_d(N-1, L, a, \rho')$ is irreducible, given that $\mathrm{RG}^*_d(N, L, a, \rho)$ is. Then one could add a fake polymer, make any move in $\mathrm{RG}^*_d(N, L, a, \rho)$, and remove the fake polymer. However, it is not always possible to add a polymer to $\mathrm{RG}^*_d(N-1, L, a, \rho')$, even if $\mathrm{RG}^*_d(N, L, a, \rho)$ is irreducible (for instance, consider in $\mathrm{RG}^*_2(3, 2, 3, \rho')$ the state with the three unoccupied vertices on a diagonal). Hence, Conjecture 2 would be true if from any state in $\mathrm{RG}^*_d(N-1, L, a, \rho)$, we could reach a state where we can add a fake polymer. Similar ideas hold for L and a.

4.3 Polymers of different lengths and underlying graphs with random out-degrees

First, all what we have done still holds when the polymers have different lengths, under the reasonable condition that we always try to replace a polymer by another polymer with the same length. Then it is easy to check that all the above formulas still hold, where each time L is to be replaced by the length L_C of the polymer C that we are trying to replace, and the constant γ is to be replaced by a number γ_{L_C}.

When the polymers have different length, a new problem arises which we have not mentioned so far: the polymers are then distinguishable, what influences the irreducibility of the algorithm. A trivial example is when $L = 1$ and $\rho = 1$, i.e., the full torus is covered by its N vertices as N polymers: if the polymers are indistinguishable, the system is irreducible, because there is only one state. But if they are distinguishable, then the system is not irreducible, because there are $N!$ isolated states.

A second extension consists in allowing the out-degree of a vertex in an underlying graph to be random, instead of deterministically equal to k (in which case item 2 in Definition 2 has to removed). The generation of underlying graphs, compatible or not, is very similar as in the case where the out-degrees are deterministic. There is only one additional step: instead of assigning k or $k-1$ uniformly at random, we first draw a number $1 \leq \kappa \leq Q$ with probability p_κ, and then we pick κ or $\kappa - 1$ vertices uniformly at random. The probability distribution $(p_i, 1 \leq i \leq Q)$ is therefore an additional parameter of the algorithm.

The interest of this extension is to allow any mean random degree $< k >= \sum \kappa p_\kappa$, so that the optimization of the algorithm can be more efficient. If such an extension is used, the probability of acceptance needs to be changed as follows (see [Sim07] for the proof):

Proposition 9 *For any polymer C and any compatible underlying graph G, let $W^0(C|G) = \prod_{i=1}^{L-1} d(v_i)$, where $C = (v_1, \ldots, v_L)$ and $d(v)$ is the out-degree of $v \in G$. Then if one sets*

$$P_{\mathrm{acc}}\big((S_\mathrm{o}, G_\mathrm{o}), (S_\mathrm{n}, G_\mathrm{n})\big) = \min\left(1, \frac{q(S_\mathrm{n}) W(C_\mathrm{n}|G_\mathrm{n}) W^0(C_\mathrm{n}|G_\mathrm{n})^{-1}}{q(S_\mathrm{o}) W(C_\mathrm{o}|G_\mathrm{o}) W^0(C_\mathrm{o}|G_\mathrm{o})^{-1}}\right), \tag{6}$$

then q is the stationary distribution of the RG* algorithm with random out-degrees.

Acknowledgment

The author would like to thank Persi Diaconis and Philippe Robert for their persistent support and interest in this work, as well as Frédéric Meunier for very fruitful discussions.

References

[BG93] Julian Besag and Peter J. Green. Spatial Statistics and Bayesian Computation. *Journal of the Royal Statistical Society*, 55(1):25–37, 1993.

[CVH+99] S. Consta, T.J.H. Vlugt, J.W. Hoeth, B. Smit, and D. Frenkel. Recoil growth algorithm for chain molecules with continuous interaction. *Molecular Physics*, 97(12):1243–1254, 1999.

[CWFA99] 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):3320–3228, February 1999.

[Hig98] David M. Higdon. Auxiliary Variable Methods for Markov Chain Monte Carlo with Applications. *Journal of the American Statistical Association*, 93(442):585–595, June 1998.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[Sie90] J.I. Siepmann. A method for the direct calculation of chemical potentials for dense chain systems. *Molecular Physics*, 70:1145–1158, 1990.

[Sim07] Florian Simatos. A variant of the Recoil Growth algorithm to generate multi-polymer systems. Available at http://aps.arxiv.org/abs/0708.1116, 2007.