Bernoulli **26**(3), 2020, 2082–2104
https://doi.org/10.3150/19-BEJ1184

Perfect sampling for Gibbs point processes using partial rejection sampling

SARAT B. MOKA[*] and DIRK P. KROESE[†]

School of Mathematics and Physics, The University of Queensland, Brisbane.
E-mail: [*]*s.babumoka@uq.edu.au;* [†]*kroese@maths.uq.edu.au*

We present a perfect sampling algorithm for Gibbs point processes, based on the partial rejection sampling of Guo, Jerrum and Liu (In *STOC'17 – Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (2017) 342–355 ACM). Our particular focus is on pairwise interaction processes, penetrable spheres mixture models and area-interaction processes, with a finite interaction range. For an interaction range $2r$ of the target process, the proposed algorithm can generate a perfect sample with $O(\log(1/r))$ expected running time complexity, provided that the intensity of the points is not too high and $\Theta(1/r^d)$ parallel processor units are available.

Keywords: area-interaction process; hard-core process; pairwise interaction process; partial-rejection sampling; penetrable spheres mixture model; perfect sampling; Strauss process

1. Introduction

Various phenomena in physics, chemistry and biology are modelled by Gibbs point processes. A Gibbs point process – or simply, Gibbs process – is a spatial point process whose distribution is absolutely continuous with respect to that of a Poisson point process (PPP). Pairwise interaction point (PIP) processes and penetrable spheres mixture (PSM) models are two widely studied examples of Gibbs processes; see, for example, Møller and Waagepetersen [23], Huber [16], Kendall and Møller [19], Baddeley and Nair [1], Baddeley and Turner [2]. The PIP family includes hard-core processes and Strauss processes.

Perfect sampling for Gibbs processes is an active area of research. A sampling algorithm for a given distribution is called *perfect* if it generates an exact sample from this distribution within a finite time. We refer to Kendall [18], Fill [10], Kendall and Møller [19], Garcia [12], Ferrari, Fernández and Garcia [9], Huber [15], Moka, Juneja and Mandjes [20], Guo and Jerrum [13] for some of the existing perfect sampling algorithms for Gibbs processes. The methods in Moka, Juneja and Mandjes [20] and Guo and Jerrum [13] generate perfect samples of hard-core processes. The other methods in the references above are applicable to more general Gibbs processes, including PIP processes and PSM models. Among these methods, the dominated coupling from the past (dCFTP) methods by Kendall [18], Kendall and Møller [19] and Huber [15] are shown to be efficient when the density of the points is small; see, for example, Huber [16]. As we show in this paper, for an interaction range $2r$ of the target Gibbs process and dimension d of the points, the expected running time complexity of any dCFTP method is at least of order $\frac{1}{r^d}\log(\frac{1}{r})$ even when the density of the reference PPP is very small. However, dCFTP algorithms are sequential, and thus they do not take advantage of parallel computing. In this paper, we propose

1350-7265 © 2020 ISI/BS

a method for generating perfect samples of PIP processes and PSM models using the *partial rejection sampling* (PRS) method of Guo, Jerrum and Liu [14] and show how one can obtain, using parallel computing, an expected running time complexity of $O(\log(1/r))$, provided that the density of the reference PPP is not too high.

The PRS method provides a general methodology for generating perfect samples from a product distribution, conditioned on none of a number of *bad events* occurring. Such problems are in general **NP**-hard; see, for example, Bezáková et al. [5] and Guo, Jerrum and Liu [14]. However, for certain types of parametric product distributions, the PRS algorithm is efficient and terminates within $O(\log n)$ iterations on average, where n is the number of bad events. Unlike the naive rejection method where samples from the product distribution are generated independently until there is no bad event, in each iteration of the PRS method, depending on the bad events seen, some of the randomness from the iteration is *retained* for the next iteration. This idea of retaining a partial randomness is rooted in the method called *Randomness Recycler*, which was first appeared in Fill and Huber [11] and later refined in Huber [16].

An additional feature of the PRS algorithm is that it is distributive, in the sense that it allows parallel computation within each iteration. As a consequence, the PRS algorithm can be implemented with $O(\log n)$ expected running time complexity using n processors in parallel. By exploiting the distributive property of the PRS, we use the PRS algorithm for generating perfect samples of Gibbs processes on a Euclidean subset S. Our algorithm is useful for methods where it is efficient to generate *quickly* but few perfect samples of a Gibbs process. For example, in each iteration of Bayesian inference using the Metropolis–Hastings algorithm proposed by Møller et al. [22], only one perfect sample of a Gibbs process is required for a vector of proposed parameters; also see Berthelsen and Møller [4]. A brief description of our contributions is as follows:

- We partition S into a finite number of cells and define a product measure by ignoring the *cross* interactions between the cells. Further by defining appropriate bad events that depend on the cross interactions, we express the distribution of the target Gibbs process as the product distribution conditioned on none of the bad events occurring. This construction allows generation of perfect samples using PRS.
- To analyze the running time complexity of the proposed algorithm, we take $S = [0, 1]^d$ and the intensity of the reference PPP as $\kappa = \frac{\kappa_0}{\mathsf{v}_d r^d}$ for some constant κ_0, where $2r$ is the interaction range of the Gibbs process and v_d is the volume of a d-dimensional sphere of unit radius. We consider the regime where κ_0 is fixed and r goes to zero, and prove that if the volume of each cell is of order r^d, there exists a constant $\bar{\kappa} > 0$ such that for all $\kappa_0 \leq \bar{\kappa}$, the expected running time complexity of the algorithm is $O(\log(1/r))$ as a function of r, provided that the number of parallel processor units available is $\Theta(1/r^d)$.
- To illustrate the application of the proposed algorithm, we consider a d-dimensional hypercube grid partitioning of $S = [0, 1]^d$ and conduct extensive simulations to estimate the expected number of iterations of the algorithm for different values of κ_0 and the interaction range $2r$.

To the best of our knowledge, there are very few perfect sampling algorithms for Gibbs processes that exploit parallel computation to obtain $O(\log(1/r))$ running time complexity; see, for example, Guo and Jerrum [13] and Huber et al. [17]. Specifically, the method of Guo and Jerrum [13] is a continuous version of the PRS algorithm and it has the same order of expected running time complexity as our algorithm, but restricted to hard-core processes. Our simulation results also provide a comparison between the expected number of iterations of the proposed method and the method of Guo and Jerrum [13] for a hard-core process.

The remaining paper is organized as follows: In Section 2, we introduce some notation. Section 3 provides definitions of the spatial point processes of interest. In Section 4, the PRS method is presented and we illustrate its application with an example. In Section 5, we propose our new perfect sampling method for Gibbs processes using PRS, and in Section 6 we analyze the running time complexities of the proposed method, naive rejection sampling, and dCFTP methods. Simulation results for a Strauss process and a PSM model are presented in Section 7. The paper is concluded in Section 8.

2. Notation

First, some notation. $\mathbb{R}_+$ is the set of nonnegative real numbers and $\mathbb{Z}_+$ is the set of non-negative integers. $\mathbb{R}^d$ denotes the d-dimensional Euclidean space with the corresponding Euclidean norm $\|\cdot\|$. The distance between any two sets $C, D \subseteq \mathbb{R}^d$ is defined by

$$\mathsf{Dist}(C, D) = \inf\big\{\|x-y\| : x \in C \text{ and } y \in D\big\},$$

with $\mathsf{Dist}(\varnothing, C) = \infty$, where $\varnothing$ denotes the empty set. We use e to denote exp(1). For any $x \in \mathbb{R}_+$, $\lfloor x \rfloor$ is the largest $n \in \mathbb{Z}_+$ such that $n \le x$. For any two probability measures μ_1 and μ_2 that are defined on the same measurable space, we write $\mu_1 \ll \mu_2$ to denote that μ_1 is absolutely continuous with respect μ_2. We write $X \sim \mu_1$ to indicate that the distribution of a random object X is μ_1. The distributions of a Bernoulli random variable with success probability p, a uniform random variable over (0, 1) and a Poisson random variable with mean λ are denoted, respectively, by $\mathsf{Bern}(p)$, $\mathsf{Unif}(0, 1)$ and $\mathsf{Poi}(\lambda)$. For any event A, $\mathbb{I}(A)$ is equal to 1 if the event holds, otherwise it is equal to 0. The underlying probability space is denoted by $(\Omega, \mathcal{F}, \mathbb{P})$.

3. Spatial point processes

Consider a finite measure ν on a Euclidean subset $S \subseteq \mathbb{R}^d$ that is absolutely continuous with respect to the Lebesgue measure. Let $\mathscr{G}$ be the set of all finite sets on S, defined by

$$\mathscr{G} := \big\{\mathbf{x} = \{x_1, x_2, \dots, x_n\} : n \in \mathbb{Z}_+ \text{ and } x_i \in S, \forall i \le n\big\},$$

where $n = 0$ corresponds to the empty set. We assume that the elements of $\mathscr{G}$ are *simple*, that is, they do not have multipoints. For any $\mathbf{x} \in \mathscr{G}$, $|\mathbf{x}_A|$ denotes the cardinality of $\mathbf{x}_A := \mathbf{x} \cap A$. A *point process* is a random element $\mathbf{X} : \Omega \to \mathscr{G}$.

Poisson point process (PPP): A point process $\mathbf{X}$ is called *Poisson* on S with intensity measure ν if it satisfies the following two properties:

(i) $|\mathbf{X}_A| \sim \mathsf{Poi}(\nu(A))$ for any measurable $A \subseteq S$ and

(ii) $|\mathbf{X}_{A_1}|, \dots, |\mathbf{X}_{A_n}|$ are independent if $A_1, \dots, A_n$ are measurable disjoint subsets of S.

A PPP is called *κ-homogeneous* if the intensity $\nu(\mathrm{d}x) = \kappa\, \mathrm{d}x$ for some constant $\kappa > 0$.

In several scenarios, it is important to associate an independent mark with each point in a PPP to characterize the shape or type of the object at that point. A *marked* PPP on S is a PPP such that each point has a (random) mark independent of all other points. The mark associated with a point can *depend* on the point. For example, a mark at a point denotes the radius of a circle centered at that point. A typical realization of a marked PPP with n points is of the form $\mathbf{x} = \{(z_1, m_1), (z_2, m_2), \dots, (z_n, m_n)\}$, where $\{z_1, z_2, \dots, z_n\} \in \mathcal{G}$ and m_i is the mark associated with z_i for $i = 1, \dots, n$. For such a marked configuration, we define $\mathbf{x}_A = \{(z_i, m_i) \in \mathbf{x} : z_i \in A\}$ for any $A \subseteq S$. If the mark space is M, then it is easy to see that the marked PPP is a PPP on $S \times M$.

It is common approach in statistical physics to wrap S on a torus (i.e., S has periodic boundary) when large interacting particle systems are considered. In that case, throughout the paper, the Euclidean distance is replaced by geodesic distance.

Gibbs point process: Suppose that ρ is the distribution of a (marked) PPP. A point process with distribution $\mu \ll \rho$ is called a *Gibbs point process* (or simply, Gibbs process) if the associated Radon–Nikodym derivative is of the form

$$\frac{\mathrm{d}\mu}{\mathrm{d}\rho}(\mathbf{x}) = \frac{\exp(-\mathcal{U}(\mathbf{x}))}{Z}, \tag{3.1}$$

for every possible realization $\mathbf{x}$ under ρ, where $\mathcal{U}$ is a nonnegative real-valued potential function that is *nondegenerate* (i.e., $\mathcal{U}(\{x\}) < \infty$), and *hereditary* (i.e., $\mathcal{U}(\mathbf{x}) \le \mathcal{U}(\mathbf{x}')$ for all $\mathbf{x} \subseteq \mathbf{x}'$). The normalizing constant Z is equal to $\mathbb{E}_\rho[\exp(-\mathcal{U}(\mathbf{X}))]$.

Pairwise interaction point (PIP) processes: A pairwise interaction point (PIP) process is a Gibbs point process for which the potential function is of the form

$$\mathcal{U}(\mathbf{x}) = \sum_{\{x,y\} \subseteq \mathbf{x}} f(x, y), \quad \mathbf{x} \in \mathcal{G}, \tag{3.2}$$

where $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+ \cup \{\infty\}$ is called the pairwise interaction function; see, for example, Chiu et al. [6]. We say that a PIP has *finite* range interaction if there exists $a < \infty$ such that $f(x, y) = 0$ for all $x, y \in S$ for which $\|x - y\| \ge a$; that is, the interaction between any two points is zero if they are separated by a distance of at least a. The smallest such t is called the *interaction range* of the PIP. Some important PIP processes are considered below.

Hard-core process: A hard-core process with hard-core distance $2r > 0$ (i.e., the hard-core radius is r) has

$$f(x, y) = \begin{cases} \infty, & \text{if } \|x - y\| < 2r, \\ 0, & \text{otherwise.} \end{cases}$$

In a hard-core process, no two points are within a distance of $2r$. Note that the interaction range here is $2r$. One generalization of the hard-core process is *hard-sphere* model with random radii, where the centers of spheres with independent and identically distributed (i.i.d.) radii constitute a PPP on S conditioned on the event that no two spheres overlap.

Strauss process: Another well-studied PIP process is the *Strauss* process with parameters $\gamma \in [0, 1]$ and $r > 0$. Here, the interaction function is defined by

$$f(x, y) = \begin{cases} -\log \gamma, & \text{if } \|x - y\| < 2r, \\ 0, & \text{otherwise.} \end{cases}$$

The interaction range of this PIP process is r. This process becomes a hard-core process if $\gamma = 0$ with the convention that $0^0 = 1$.

Strauss-hard core process: This PIP process is a hybrid of the Strauss and hard-core processes, and has interaction function

$$f(x, y) = \begin{cases} \infty, & \text{if } \|x - y\| < r_1, \\ -\log \gamma, & \text{if } r_1 \le \|x - y\| < r_2, \\ 0, & \text{otherwise,} \end{cases}$$

for some $\gamma \in [0, 1]$ and $0 < r_1 < r_2$. Here, r_1 is called hard-core distance. Clearly, the interaction range for this process is r_2.

Penetrable spheres mixture (PSM) model: This model was introduced by Widom and Rowlinson [24] to study liquid-vapor phase transitions. Let ρ is the distribution of κ-homogeneous marked PPP, where each point is independently marked as type-1 with probability $\kappa_1/(\kappa_1 + \kappa_2)$, otherwise, as type-2 (with probability $\kappa_2/(\kappa_1 + \kappa_2)$), for some constants $\kappa_1, \kappa_2 \ge 0$. A realization of a PSM model can be viewed as a realization of $\mathbf{X} \sim \rho$ conditioned on the event that no two points from different types are within a distance $2r$ from each other. The corresponding potential function is given by (3.2) with

$$f(x, y) = \begin{cases} \infty, & \text{if } \|x - y\| < 2r \text{ and } x, y \text{ have different marks,} \\ 0, & \text{otherwise.} \end{cases}$$

Area-interaction process: This process was first studied by Baddeley and van Lieshout [3] (see, also, Kendall and Møller [19], Ferrari, Fernández and Garcia [9] and Møller [21]). For any $A \subseteq \mathbb{R}^d$, let $\mathsf{Vol}(A)$ be the volume of A and $\mathsf{Ball}(x, a)$ be the d-dimensional sphere centered at x with radius a. The distribution of an area-interaction process on S is absolutely continuous with respect to that of a λ-homogeneous PPP for some $\lambda > 0$, with the potential function given by

$$\mathcal{U}(\mathbf{x}) = \beta \mathsf{Vol}\left(\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, 2r) \right), \quad \mathbf{x} \in \mathcal{G}, \tag{3.3}$$

where the constant $\beta > 0$ is called *inverse temperature*; see Figure 1(a). The definition of area-interaction process given in Baddeley and van Lieshout [3] is more general, as it allows $\beta < 0$. However, in this paper, we focus only on the case $\beta > 0$.

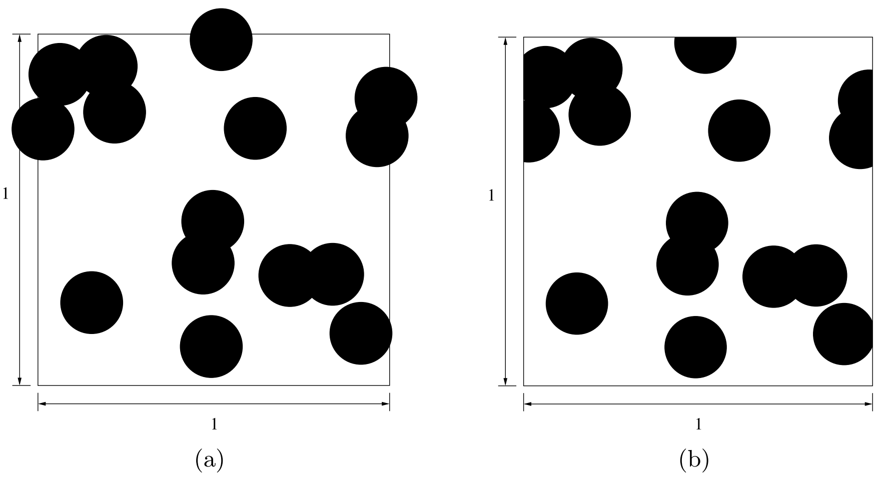

Figure 1. A realization $\mathbf{x} = \{x_1, \dots, x_{15}\}$ of an area-interaction process on a unit square $S = [0, 1]^2$, where each x_i is the center of a circle with radius $2r$. The dark regions in Panel (a) and Panel (b) are $\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, r)$ and $S \cap (\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, 2r))$, respectively.

There is an interesting connection between area-interaction processes and PSM models (Baddeley and van Lieshout [3]). To see this, instead of (3.3), if we suppose that the potential function is

$$\mathcal{U}(\mathbf{x}) = \beta \mathsf{Vol}\left(S \cap \left(\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, 2r) \right) \right), \quad \mathbf{x} \in \mathcal{G}, \tag{3.4}$$

then the distribution of this modified area-interaction process is identical to the distribution of type-1 points of the PSM model with $\kappa = \lambda + \beta$, $\kappa_1 = \lambda$ and $\kappa_2 = \beta$; see Figure 1(b). This is because from the property (i) in the definition of PPPs, for any $\mathbf{x} \in \mathcal{G}$, the probability that none of the points of a realization of a β-homogeneous PPP falls within the set $S \cap (\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, 2r))$ is equal to $\exp(-\beta \mathsf{Vol}(S \cap (\bigcup_{x \in \mathbf{x}} \mathsf{Ball}(x, 2r))))$. A Further fact is that if S is periodic, both (3.3) and (3.4) are the same. Hence, under the periodic assumption, an area-interaction process can be viewed as a realization of one type of points of a PSM model, and vice versa. This is the reason why area-interaction processes are sometimes referred as PSM models.

In the definition of PSM models, type-1 and type-2 points are independent PPPs on S with intensities κ_1 and κ_2, respectively. Instead, if we assume that the type-2 points constitute a κ_2-homogeneous PPP on a bigger set $S(r)$ such that

$$\bigcup_{x \in S} \mathsf{Ball}(x, 2r) \subseteq S(r)$$

(when $S = [0, 1]^d$, $S(r)$ can be $[-2r, 1 + 2r]^d$) then, with the choice of $\kappa_1 = \lambda$ and $\kappa_2 = \beta$, the distribution of type-1 points of this modified PSM model is identical to the distribution of the area-interaction process.

4. Partial rejection sampling

In this section, we briefly discuss the partial rejection sampling (PRS) method proposed in Guo, Jerrum and Liu [14], Section 4. This method generates perfect samples from a product distribution, conditioned on none of a number of *bad* events happening.

To be precise, let $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ be a set of easy to simulate independent random objects taking values on $\mathcal{Y}$. Suppose that $\mu^{\otimes}$ is the distribution of $\mathbf{Y}$. Clearly, $\mu^{\otimes}$ is a product distribution. Without loss of generality, we assume that $\mathcal{Y}$ is the support of $\mu^{\otimes}$. Let $\{B_v \in \mathcal{F} : v \in V\}$ be a set of bad events indexed by elements of a finite set V. Each bad event B_v depends on a subset of $\mathbf{Y}$. Let $\mathcal{I}(v) \subseteq \{1, 2, \ldots, n\}$ be the largest set such that B_v is dependent on Y_i for all $i \in \mathcal{I}(v)$; that is, $\mathcal{I}(v)$ is the smallest set such that the set of variables $\{Y_i : i \in \mathcal{I}(v)\}$ imply whether the event B_v occurs or not; see Figure 2 for an illustration. By definition, B_v does not depend on $\{Y_i : i \in \{1, \ldots, n\} \setminus \mathcal{I}(v)\}$. The goal of PRS is to generate perfect samples from $\mu^{\otimes}$, conditioned on the event that none of the bad events $\{B_v : v \in V\}$ occur.

We can generate the desired samples using the naive rejection sampling algorithm: repeatedly generate a sample from $\mu^{\otimes}$ until none of the bad events occur. The last sample has the desired distribution. In each iteration of this naive method, a fresh copy of the entire set $\mathbf{Y}$ is generated. Whereas, as we see below, in each iteration of the PRS method, only a subset of $\mathbf{Y}$ is resampled based on which bad events occurred in the previous iteration. This helps to significantly reduce the running time complexity compared with naive rejection sampling.

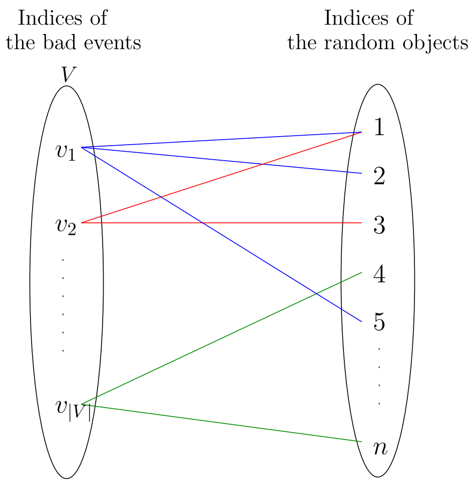

Figure 2. An illustration of the relationship between the bad events $\{B_v \in \mathcal{F} : v \in V\}$ and the random objects $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$. The mapping shows that $\mathcal{I}(v_1) = \{1, 2, 5\}$, $\mathcal{I}(v_2) = \{1, 3\}$ and $\mathcal{I}(v_{|V|}) = \{4, n\}$. That is, the bad event B_{v_1} depends only on the random objects Y_1, Y_2, Y_5, the bad event B_{v_2} depends only on Y_1, Y_3, and the bad event $B_{|V|}$ depends only on Y_4, Y_n. Consequently, $\{v_1, v_2\} \in E$, $\{v_1, v_{|V|}\} \notin E$ and $\{v_2, v_{|V|}\} \notin E$. Furthermore, if $W = \{v_1, v_2\}$ then $\mathcal{I}(W) = \{1, 2, 3, 5\}$.

For any $u, v \in V$, we write $u \leftrightarrow v$ if $\mathcal{I}(u) \cap \mathcal{I}(v) \neq \varnothing$. Define the *dependency graph* $G = (V, E)$ to be the graph, with the vertex set V and edge set E given by

$$E = \{\{u, v\} : u, v \in V, u \neq v \text{ and } u \leftrightarrow v\}.$$

That is, there is an edge between two nodes in V if the bad events associated with the nodes depend on at least one common random object. In Example 4.1 and Section 5, G is a *line* graph (also known as *edge-to-vertex dual* graph) as the node set V consists of edges on a lattice. For $\omega \in \Omega$, let

$$\mathsf{Bad}\big(\mathbf{Y}(\omega)\big) = \{v \in V : \omega \in B_v\}.$$

For any subset $W \subseteq V$, let ∂W be the boundary of the set W defined by

$$\partial W = \{v \in V : v \notin W \text{ and } \exists u \in W \text{ such that } u \leftrightarrow v\}.$$

Also define $\mathcal{I}(W) = \bigcup_{u \in W} \mathcal{I}(u)$, and for any assignment $\mathbf{y} = \{y_i : i = 1, \dots, n\} \in \mathcal{Y}$ of $\mathbf{Y}$, let $\mathbf{y}|_W := \{y_i : i \in \mathcal{I}(W)\}$ denote the partial assignment of $\mathbf{y}$ restricted to $\mathcal{I}(W)$. For instance, in Figure 2, we have $\mathcal{I}(\{v_1, v_2\}) = \{1, 2, 3, 5\}$, and hence $\mathbf{Y}|_{\{v_1, v_2\}} = \{Y_1, Y_2, Y_3, Y_5\}$. For any two assignments $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$ of $\mathbf{Y}$, if $\mathbf{y}|_W = \mathbf{y}'|_W$ then $\mathbf{y}'$ is called an *extension* of $\mathbf{y}|_W$, and vice versa. Furthermore, an event B is said to be *disjoint* from $\mathbf{y}|_W$ if either $\mathcal{I}(v) \cap \mathcal{I}(W) = \varnothing$ or B cannot occur for any extension of $\mathbf{y}|_W$.

Algorithm 1 generates a perfect sample $\mathbf{Y} \sim \mu^{\otimes}$, conditioned on the event that none of the bad events B_v occurs. In each iteration of Algorithm 1, the inner while-loop constructs the *resampling* set $\mathsf{Res} \subseteq V$. It starts with $\mathsf{Res} = \mathsf{Bad}(\mathbf{Y})$ where the initial assignment of random objects is $\mathbf{Y}$, and recursively adds to Res the set of all the boundary vertices that are not disjoint from Res, until there are no more boundary vertices to add. The final Res is the resampling set, and all the random objects with indices in $\bigcup_{u \in \mathsf{Res}} \mathcal{I}(u)$ are resampled. This construction is deterministic, in the sense that the final resampling set is a deterministic function of $\mathbf{Y}$.

Algorithm 1: Partial rejection sampling algorithm

```
Simulate $Y_1, Y_2, \dots, Y_n$ independently
$\mathbf{Y} \leftarrow \{Y_1, Y_2, \dots, Y_n\}$
**while** $\mathsf{Bad}(\mathbf{Y}) \neq \varnothing$ **do**
    $\mathsf{Res} \leftarrow \mathsf{Bad}(\mathbf{Y})$ and $N \leftarrow \varnothing$
    **while** $\partial\mathsf{Res} \setminus N \neq \varnothing$ **do**
        Let $D = \{v \in \partial\mathsf{Res} \setminus N : B_v$ is disjoint from $\mathbf{Y}|_{\mathsf{Res}}\}$
        $N \leftarrow N \cup D$
        $\mathsf{Res} \leftarrow \mathsf{Res} \cup (\partial\mathsf{Res} \setminus N)$
    **end**
    Resample only the objects in $\{Y_i : i \in \mathcal{I}(\mathsf{Res})\}$
**end**
Output $\mathbf{Y}$
```

We refer to Guo, Jerrum and Liu [14] for a proof of correctness of the algorithm. We must note that in Guo, Jerrum and Liu [14], each Y_i is a real-valued random variable, whereas in this paper, we allow a more general setting by treating each Y_i as a random object. However, the correctness proof still holds true for this general case as well.

Example 4.1 (Hard-core model on a lattice). To illustrate the PRS algorithm, consider the following hard-core model defined on a square lattice. Each node i of the lattice is associated with an independent Bernoulli random variable $Y_i \sim \mathsf{Bern}(\frac{\lambda}{1+\lambda})$ for some $\lambda > 0$. The node i is said to be occupied if $Y_i = 1$. Associated with each edge $\{i, j\}$, there is a bad event $B_{i,j}$ which holds if both the endpoints i and j are occupied; see Figure 3. If we let $\mu^{\otimes}$ be the distribution of Y_i's, then using the PRS algorithm we can generate a sample $\mathbf{Y} \sim \mu^{\otimes}$ conditioned on none of the bad events occurring.

The corresponding dependency graph $G = (V, E)$ consists of V, the set of all the edges in the lattice, and

$$E = \bigl\{\{v, u\} : v \neq u, v \text{ and } u \text{ are connected by a common node}\bigr\}.$$

Clearly, if $v = \{i, j\} \in V$, we have $\mathcal{I}(v) = \{i, j\}$. As shown in Section 6.2 of Guo, Jerrum and Liu [14], for this hard-core model, it is easy to find the resampling set for any given set of bad events. Suppose at an iteration of the algorithm, if Bad is the set of bad edges of the lattice, then the resampling set Res is the union of Bad and $\partial\mathsf{Bad}$, where one endpoint of each edge in $\partial\mathsf{Bad}$ is *not* occupied and the other endpoint is shared with one of the edges in Bad; see Figure 3.

As mentioned earlier, in general, generating a sample from $\mu^{\otimes}$ conditioned on none of the bad events happening is **NP**-hard. However, under some additional conditions, Algorithm 1 can be efficient in the sense that the expected number of iterations of the algorithm can be $O(\log|V|)$; see Guo, Jerrum and Liu [14]. Lemma 4.1 deals with one such case. Refer to Guo, Jerrum and Liu [14], Section 5, for a proof. Let $A_{u,v}$ be the event that the partial assignment on $\mathcal{I}(v) \cap \mathcal{I}(u)$

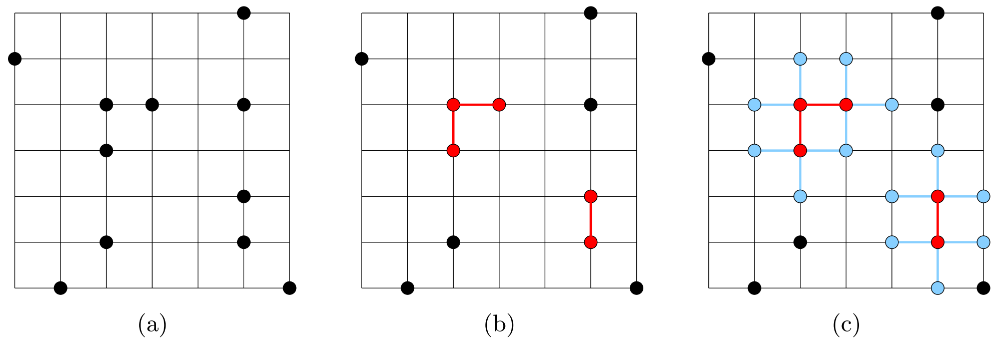

Figure 3. Hard-core model on a square lattice. Panel (a): A typical realization of the hard-core model, where the occupied nodes are denoted by dark circles. Panel (b): The red coloured edges denote the bad events. Panel (c): The red and blue colored edges together are the final resampling set; all the red and blue nodes will be resampled.

can be extended to make B_v occur; that is,

$$A_{u,v} = \{\omega \in \Omega : \exists \omega' \in B_v \text{ such that } \mathbf{Y}(\omega)|_{\{u,v\}} = \mathbf{Y}(\omega')|_{\{u,v\}}\}.$$

In particular, if $\{u, v\} \in E$ then $u \leftrightarrow v$ implies that $\mathcal{I}(u) \cap \mathcal{I}(v) \neq \varnothing$, and hence $A_{u,v}$ is the set of ω for which B_v is not disjoint from $\mathbf{Y}(\omega)|_{\{u,v\}}$.

Define $p = \max_{v \in V} \mathbb{P}_{\mu^{\otimes}}(B_v)$ and $q = \max_{\{u,v\} \in E} \mathbb{P}_{\mu^{\otimes}}(A_{u.v})$. Let Bad_t and Res_t be, respectively, the set of bad vertices and the resampling set at iteration $t \geq 0$ of Algorithm 1. Further, let Δ be the maximum degree of the dependency graph G.

Lemma 4.1 (Lemma 5.4 of Guo, Jerrum and Liu [14]). *For any* $\Delta \geq 2$, *if* $6\mathrm{e}p\Delta^2 \leq 1$ *and* $3\mathrm{e}q\Delta \leq 1$, *then for all* $t \geq 0$,

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \mid |\mathsf{Res}_0, \dots, \mathsf{Res}_t] \leq (1 - p)|\mathsf{Res}_t|.$$

Note that $\mathsf{Bad}_t \subseteq \mathsf{Res}_t$ for all $t \geq 0$. From Lemma 4.1 and the fact that $|\mathsf{Bad}_0| = |\mathsf{Res}_0| = |V|$ (since the algorithm starts with a fresh copy of all the random elements),

$$\mathbb{E}[|\mathsf{Bad}_t|] \leq \mathbb{E}[|\mathsf{Res}_t|] \leq (1 - p)^t |V|, \tag{4.1}$$

for all $t \geq 0$, under the hypothesis of the lemma. These observations are useful for the running time complexity analysis in Section 6.

5. Perfect sampling for Gibbs point processes

In this section, we propose a methodology to use the PRS algorithm for generating perfect samples of the Gibbs processes defined in Section 3. For this, we first partition the underlying space S. Then, using this partition, we identify certain bad events such that the target distribution can be expressed as a product distribution conditioned on none of these bad events occurring. For the case where $S = [0, 1]^d$, we consider a hypercube grid partitioning and specialize the PRS algorithm.

Recall the definition of Gibbs process with distribution μ, given in Section 3. Assume that the corresponding potential function $\mathcal{U}$ has a finite interaction range $2r$ and

$$\mathcal{U}(\mathbf{x}) = \sum_{\{x,y\} \subseteq \mathbf{x}} f(x, y), \quad \mathbf{x} \in \mathcal{G}, \tag{5.1}$$

for a function $f : S \times S \to \mathbb{R}_+ \cup \{\infty\}$ such that $f(x, y) = f(y, x)$. Recall that $\mu \ll \rho$, with ρ being the distribution of a (marked) PPP. Clearly, both PIP process or PSM model (defined in Section 3) can be seen as special cases of the above description.

Suppose $\{C_1, C_2, \dots, C_n\}$ is a partition of S (i.e., the C_i's are mutually disjoint and $\bigcup_{i=1}^n C_i = S$). Let $V = \{v = \{i, j\} : \mathsf{Dist}(C_i, C_j) < 2r \text{ and } i \neq j\}$ be the set of unordered pairs $\{i, j\}$ such that points that fall in C_i can interact with points in C_j and vice versa. As a consequence of (5.1), for any $\mathbf{x} \in \mathscr{G}$,

$$\mathcal{U}(\mathbf{x}) = \sum_{i=1}^{n} \mathcal{U}(\mathbf{x}_{C_i}) + \sum_{\{i,j\} \in V} \sum_{\substack{x \in \mathbf{x}_{C_i} \\ y \in \mathbf{x}_{C_j}}} f(x, y).$$

Hence,

$$\begin{aligned}\mathbb{E}_\rho\big[\mathrm{e}^{-\mathcal{U}(\mathbf{X})}\big] &= \mathbb{E}_\rho\left[\prod_{i=1}^{n} \mathrm{e}^{-\mathcal{U}(\mathbf{X}_{C_i})} \prod_{\{i,j\} \in V} \exp\left(-\sum_{x \in \mathbf{X}_{C_i}, y \in \mathbf{X}_{C_j}} f(x, y)\right)\right] \\ &= \mathbb{E}_\rho\left[\prod_{i=1}^{n} \mathrm{e}^{-\mathcal{U}(\mathbf{X}_{C_i})} \prod_{\{i,j\} \in V} \mathbb{I}\left\{U_{i,j} \le \exp\left(-\sum_{x \in \mathbf{X}_{C_i}, y \in \mathbf{X}_{C_j}} f(x, y)\right)\right\}\right],\end{aligned}$$

where $\{U_{i,j} : \{i, j\} \in V\}$ is a set of i.i.d. $\mathsf{Unif}(0, 1)$ random variables, independent of everything else.

For each i, let ρ_i be the distribution of the reference (marked) PPP restricted to the cell C_i, that is, if $\mathbf{X} \sim \rho$ then ρ_i is the distribution of $\mathbf{X}_{C_i}$, and $\mathbf{X}_{C_i}$ and $\mathbf{X}_{C_j}$ are independent when $i \neq j$ (see the property (ii) in the definition of PPPs). Now let μ_i be the distribution of a Gibbs process on C_i such that $\mu_i \ll \rho_i$ with the interaction range $2r$ and the potential function $\mathcal{U}(\mathbf{x}) = \sum_{\{x,y\} \in \mathbf{x}} f(x, y)$ for all finite subsets $\mathbf{x} \subseteq C_i$. This means that μ_i is the distribution of the target Gibbs process restricted to C_i. Furthermore, define bad events

$$B_{i,j} = \left\{\omega \in \Omega : U_{i,j}(\omega) > \exp\left(-\sum_{x \in \mathbf{X}_{C_i}(\omega), y \in \mathbf{X}_{C_j}(\omega)} f(x, y)\right)\right\}, \quad \{i, j\} \in V.$$

Let $\rho = \rho_1 \times \cdots \times \rho_n$ and $\mu^\otimes := \mu_1 \times \mu_2 \times \cdots \times \mu_n$. From the definition of μ, $\mu \ll \mu^\otimes$ and

$$\frac{\mathrm{d}\mu}{\mathrm{d}\mu^\otimes}(\mathbf{x}) = \frac{1}{\widetilde{Z}} \prod_{\{i,j\} \in V} \exp\left(-\sum_{\{i,j\} \in V} \sum_{x \in \mathbf{x}_{C_i}, y \in \mathbf{x}_{C_j}} f(x, y)\right), \quad \mathbf{x} \in \mathscr{G}.$$

Equivalently, μ is equal to the distribution $\mu^\otimes$ conditioned on none of the bad events $B_{i,j}$ happening. Here, the normalizing constant is

$$\widetilde{Z} = \mathbb{P}_{\mu^\otimes}\left(\bigcap_{\{i,j\} \in V} B_{i,j}^c\right) = \mathbb{E}_{\mu^\otimes}\left[\prod_{\{i,j\} \in V} \exp\left(-\sum_{x \in \mathbf{X}_{C_i}, y \in \mathbf{X}_{C_j}} f(x, y)\right)\right].$$

Since $\mu^\otimes$ is a product measure, if it is possible to generate samples from μ_i's, we can use the PRS, Algorithm 1, to generate samples from μ. The corresponding dependency graph is $G = (V, E)$, where, from the definition, $E = \{\{u, v\} : u, v \in V \text{ and } u \leftrightarrow v\}$ with $B_{\{i,j\}} = B_{i,j}$.

To complete the argument, we need to spell out how to identify the resampling subset $\mathsf{Res}(\mathbf{Y}(\omega))$ of V for every $\omega \in \Omega$, where

$$\mathbf{Y} := \{\mathbf{X}_{C_i}, i = 1, \dots, n, \text{ and } U_{i,j}, \{i, j\} \in V\}.$$

This depends on knowing the condition for $B_{i,j}$ being disjoint from $\mathbf{Y}(\omega)|_W$ for all $W \subseteq V$ and all $\{i, j\} \in \partial W$. Lemma 5.1 establishes this condition.

To simplify the notion, we can take $\mathcal{I}(\{i, j\}) = \{i, j\}$ for all $\{i, j\} \in V$. That means, at any iteration of PRS, if $\{i, j\} \in \mathsf{Res}$ then $\mathbf{X}_{C_i}$, $\mathbf{X}_{C_j}$ and $U_{i,j}$ will be resampled independently from their respective distributions.

Lemma 5.1. *Let $W \subseteq V$ and $\{j, k\} \in \partial W$ with $j \in \mathcal{I}(W)$. Then for any $\omega \in \Omega$, $B_{j,k}$ is disjoint from the partial assignment $\mathbf{Y}(\omega)|_W$ if and only if $\mathsf{Dist}(\mathbf{X}_{C_j}(\omega), C_k) \geq 2r$.*

Proof. Observe that $k \notin \mathcal{I}(W)$, because $j \in \mathcal{I}(W)$ and $\{j, k\} \in \partial W$. Hence $\mathcal{I}(\{j, k\}) \cap \mathcal{I}(W) = \{j\}$. This implies that if $\mathsf{Dist}(\mathbf{X}_{C_j}(\omega), C_k) \geq 2r$ then $B_{j,k}$ cannot occur for any extension of $\{\mathbf{X}_{C_j}(\omega)\}$, because no matter what is the configuration on C_k, it never interacts with the points of $\mathbf{X}_{C_j}(\omega)$. On the other hand, if $\mathsf{Dist}(\mathbf{X}_{C_j}(\omega), C_k) < 2r$, we can find a $\omega' \in \Omega$ such that $\mathbf{X}_{C_j}(\omega) = \mathbf{X}_{C_j}(\omega')$ and $\omega' \in B_{j,k}$ (i.e., the points of $\mathbf{X}_{C_k}(\omega')$ interact with points of $\mathbf{X}_{C_j}(\omega)$ to result in occurrence of $B_{j,k}$). □

5.1. Hypercube grid partitioning

Consider the PIP processes and PSM models defined in Section 3. Suppose that $S = [0, 1]^d$ is equipped with a hypercube grid of cell edge length $2r$. The area-interaction case is discussed at the end of this section.

So, the cells are d-dimensional hypercubes with volume $(2r)^d$, except some of the boundary cells that can be rectangular in shape with each edge is of length at most $2r$. Let $n(r)$ be the total number of cells. Note that when $2r = 1/K$ for some integer K, every cell is a cube. We say that cells C_i and C_j are adjacent if $\mathsf{Dist}(C_i, C_j) = 0$. From this construction, it is evident that $\{i, j\} \in V$ if and only if C_i and C_j are adjacent. Furthermore, there is no cross interaction between point configurations on two non-adjacent cells, that is,

$$\mathcal{U}(\mathbf{X}_{C_i}(\omega) \cup \mathbf{X}_{C_j}(\omega)) = \mathcal{U}(\mathbf{X}_{C_i}(\omega)) + \mathcal{U}(\mathbf{X}_{C_j}(\omega)),$$

for all $\omega \in \Omega$ if C_i and C_j are not adjacent to each other; see Figure 4.

Recall from Lemma 5.1 that the implementation of the PRS algorithm for Gibbs processes depends on deciding whether $\mathsf{Dist}(\mathbf{X}_{C_i}, C_j) < 2r$, or not, for a given $\{i, j\} \in V$ and a point configuration $\mathbf{X}_{C_i}$ on the i^{th} cell. The interesting aspect of this hypercube grid partitioning is that for any $\{i, j\} \in V$, $\mathsf{Dist}(\mathbf{X}_{C_i}, C_j) < 2r$ if and only if the point configuration $\mathbf{X}_{C_i} = \varnothing$. To verify this claim, notice that the 'if' part trivially follows from the definition of Dist, and the 'only if' part follows from the observation that each cell has edges of length at most $2r$ and when $\mathbf{X}_{C_i} \neq \varnothing$, we can select a point configuration on C_j and a value for $U_{i,j}$ so that the bad event $B_{i,j}$ occur, making it not disjoint from $\mathbf{X}_{C_i}$. As a consequence, observe that for any realization

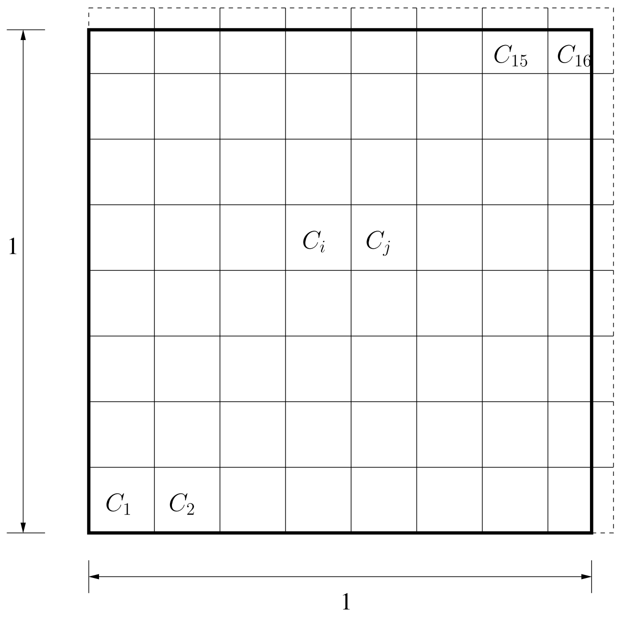

Figure 4. For a Gibbs process with interaction range $2r = 0.13$, the partition of the unit cube $S = [0, 1]^2$ using a square grid of size 8×8 with the cell edge length 0.13. Since $7 \times 0.13 < 1 < 8 \times 0.13$, some of the boundary cells are smaller in size than the other cells. Clearly, for any i and j, $\{i, j\} \in V$ if and only if C_i and C_j are adjacent (i.e., $\mathsf{Dist}(C_i, C_j) = 0$).

of $\mathbf{Y}$ in an iteration of the PRS algorithm, Res is the minimal subset of V such that $\mathsf{Bad} \subseteq \mathsf{Res}$ and

$$\mathbf{X}_{C_i} = \varnothing, \quad \text{for all } \{i, j\} \in \partial\mathsf{Res} \text{ with } i \in \mathcal{I}(\mathsf{Res}).$$

Under this setup, we now restate the PRS algorithm for Gibbs processes; see Algorithm 2. We remind the reader that for each i, samples $\mathbf{X}_{C_i}$ from μ_i are generated using any existing method such as dCFTP.

To generate perfect samples of an area-interaction process on $S = [0, 1]^d$ with inverse temperature β and the intensity of the reference PPP is λ, we use Algorithm 2 to generate samples of the modified PSM model on $S(r) = [-2r, 1 + 2r]^d$ where the reference PPP consists of type-1 and type-2 points, with type-1 points being λ-homogeneous PPP on S and type-2 points being β-homogeneous PPP on $S(r)$. Type-1 points in the output of the algorithm is a sample of the target area-interaction process. Refer to Section 3 for the connection between area-interaction processes and PSM models. In Algorithm 2, instead of S, we equip $S(r)$ with a hypercube grid partitioning. On each cell C_i, μ_i is the distribution of a modified PSM model where the reference PPP consists of type-1 and type-2 points, with type-1 points being λ-homogeneous PPP on $C_i \cap S$ and type-2 points being β-homogeneous PPP on $C_i \setminus S$.

Algorithm 2: PRS using hypercube grid partitioning

```
Simulate $\mathbf{X}_{C_i} \sim \mu_i, i = 1, \dots, n(r)$, and $U_{i,j} \sim \mathsf{Unif}(0,1)$, $\{i,j\} \in V$, independently
$\mathbf{Y} \leftarrow \{\mathbf{X}_{C_i}, i = 1, \dots, n(r),$ and $U_{i,j}, \{i,j\} \in V\}$
while $\mathsf{Bad}(\mathbf{Y}) \neq \varnothing$ do
    $\mathsf{Res} \leftarrow \mathsf{Bad}(\mathbf{Y})$
    repeat
        Let $W = \{\{i,j\} \in \partial\mathsf{Res} : i \in \mathcal{I}(\mathsf{Res})$ and $\mathbf{X}_{C_i} \neq \varnothing\}$
        $\mathsf{Res} \leftarrow \mathsf{Res} \cup W$
    until $W = \varnothing$;
    Resample $\mathbf{X}_{C_i} \sim \mu_i, i \in \mathcal{I}(\mathsf{Res})$, and $U_{i,j} \sim \mathsf{Unif}(0,1)$, $\{i,j\} \in \mathsf{Res}$, independently
end
Output $\mathbf{Y}$
```

6. Running time analysis

In this section, we assume that $S = [0,1]^d$ and the target Gibbs distribution $\mu \ll \rho$, with potential function given by (5.1) and interaction range $2r \leq 1$. We further assume that ρ is the distribution of a κ-homogeneous (marked) PPP on S with $\kappa = \kappa_0/(\mathsf{v}_d r^d)$ for some $\kappa_0 > 0$. We analyze the running time complexity of the partitioning based PRS (described in Section 5) as $r \to 0$. We further compare this method with two well-known existing methods by establishing trivial lower bounds on the running time complexities of the existing methods.

Suppose that $\{C_1, C_2, \dots, C_{n(r)}\}$ is a hypercube grid partitioning of S (as shown in Figure 4) such that samples from μ_i can be simulated using any of the existing perfect sampling algorithms, such as the dCFTP (see Section 5 for the definition of μ_i). For each $i = 1, 2, \dots, n(r)$, let N_i be the number of cells C_j, $j \neq i$, such that $\mathsf{Dist}(C_i, C_j) < 2r$; these N_i cells are referred as *neighboring cells* of the cell C_i. Theorem 6.1 below establishes that if the volume of each cell is chosen to be of order r and the N_i's are uniformly bounded for all r, then there exist a constant $\bar{\kappa}$ such that for all $\kappa_0 \leq \bar{\kappa}$, the expected number of iterations the PRS algorithm takes to generate a perfect sample is $O(\log(1/r))$. Observe that for the hypercube grid partition of Section 5.1, N_i is bounded by a constant uniformly and $\frac{\mathsf{Vol}(C_i)}{r^d} \leq 2^d$, for all r and i. Hence the conditions in Theorem 6.1 hold.

As mentioned in Guo, Jerrum and Liu [14], an interesting feature of the PRS algorithm is that it is *distributive*. In particular, if we assume that each cell i is associated with a processor that can generate samples from μ_i and can communicate with all the N_i neighboring cells within a constant time, then with the help of another processor as a top-node that can communicate with all the $n(r)$ cell processors, the expected running time complexity of each iteration of the algorithm can be reduced to $O(1)$; see the proof of Theorem 6.1. In that case, the expected running time complexity of the PRS algorithm is simply of the order of the expected number of iterations, which is $O(\log(1/r))$. See, for example, Feng and Yin [8], Feng, Sun and Yin [7] for recent works on distributed sampling.

Theorem 6.1. *Suppose that there exists constants $a, b > 0$ such that for all $i = 1, \dots, n(r)$, $N_i \le a$ and*

$$\frac{\mathsf{Vol}(C_i)}{r^d} \le b \quad \textit{for all } r. \tag{6.1}$$

Then there exists $\bar{\kappa} > 0$ such that for all $\kappa_0 \le \bar{\kappa}$, we have:

(i) *the expected number of iterations of the PRS algorithm using hypercube grid is $O(\log(1/r))$,*
(ii) *the expected running time complexity of the algorithm is $O(\frac{1}{r^d}\log(1/r))$, and*
(iii) *if the implementation of the algorithm is distributive then the expected running time complexity is $O(\log(1/r))$.*

Proof. The expected number of points generated under ρ within cell i is $\kappa\mathsf{Vol}(C_i)$, which can be upper bounded by $(\kappa_0 b)/\mathsf{v}_d$, under the assumption (6.1). Since the bound is independent of r, for any given κ_0, the running time complexity of generating a sample from μ_i is $O(1)$, for all i, using any standard perfect sampling algorithm; see Chapter 7 of Huber [16]. We first show that both p and q go to zero as κ_0 goes to zero, and then we prove $(i) - (iii)$ as a consequence of (4.1).

Since $N_i \le a$ for all r and i, the total number of nodes $|V|$ of the dependency graph is of order $n(r)$ and the maximum degree Δ is uniformly bounded for all r. The lower bound in (6.1) implies that $n(r)$ is of order $1/r^d$. For any $\{i, j\} \in V$, occurrence of the event $B_{i,j}$ implies that both the cells i and j have nonempty configurations. Since $\mu_i \ll \rho_i$, from the definition of Gibbs process, the probability of cell i has an empty configuration is

$$\mathbb{P}_{\mu^\otimes}(\mathbf{X}_{C_i} = \varnothing) = \exp\bigl(-\kappa\mathsf{Vol}(C_i)\bigr)/\widetilde{Z}_i,$$

where $\widetilde{Z}_i = \mathbb{E}_\rho[\exp(-\mathcal{U}(\mathbf{X}_{C_i}))] = \mathbb{E}_{\rho_i}[\exp(-\mathcal{U}(\mathbf{X}))]$. Observe that for any $\omega \in \Omega$, if either $\mathbf{X}_{C_i} = \varnothing$ or $\mathbf{X}_{C_j} = \varnothing$ then $\omega \in B_{i,j}^c$. Hence,

$$\begin{aligned}
p &\le \max_{\{i,j\}\in V} \mathbb{P}_{\mu^\otimes}(\mathbf{X}_{C_i} \neq \varnothing \text{ and } \mathbf{X}_{C_j} \neq \varnothing) \\
&= \max_{\{i,j\}\in V} \bigl[\mathbb{P}_{\mu^\otimes}(\mathbf{X}_{C_i} \neq \varnothing)\mathbb{P}_{\mu^\otimes}(\mathbf{X}_{C_j} \neq \varnothing)\bigr] \\
&= \max_{\{i,j\}\in V} \bigl[\bigl(1 - \exp\bigl(-\kappa\mathsf{Vol}(C_i)\bigr)/\widetilde{Z}_i\bigr)\bigl(1 - \exp\bigl(-\kappa\mathsf{Vol}(C_j)\bigr)/\widetilde{Z}_j\bigr)\bigr] \\
&\le \max_{\{i,j\}\in V} \bigl[\bigl(1 - \exp\bigl(-\kappa\mathsf{Vol}(C_i)\bigr)\bigr)\bigl(1 - \exp\bigl(-\kappa\mathsf{Vol}(C_j)\bigr)\bigr)\bigr],
\end{aligned} \tag{6.2}$$

where the last inequality holds because $\widetilde{Z}_i \le 1$ for all i. Using (6.1), we write that

$$p \le \left(1 - \exp\left(-\frac{\kappa_0 b}{\mathsf{v}_d}\right)\right)^2.$$

Therefore, p goes to zero as κ_0 goes to zero.

Recall from the definition that the event $A_{u,v}$ holds true if the partial assignment on $\mathcal{I}(u)\cap\mathcal{I}(v)$ can be extended to make B_u true. Also recall that if $\{u,v\}\in E$, there exists i, j and k such that $u=\{i,k\}$ and $v=\{j,k\}$, and thus, $\mathcal{I}(u)\cap\mathcal{I}(v)=\{k\}$. This implies that the event $A_{v,u}$ can not occur if the common cell k is empty. Thus,

$$\begin{aligned}\mathbb{P}_{\mu^{\otimes}}(A_{v,u}) &\le \bigl(1-\exp\bigl(-\kappa \mathsf{Vol}(C_k)\bigr)/\widetilde{Z}_k\bigr)\\ &\le \bigl(1-\exp\bigl(-\kappa \mathsf{Vol}(C_k)\bigr)\bigr)\\ &\le 1-\exp\biggl(-\frac{\kappa_0 b}{\mathsf{v}_d}\biggr).\end{aligned}\tag{6.3}$$

As a consequence $q\le 1-\exp(-\frac{\kappa_0 b}{\mathsf{v}_d})$, which goes to zero as κ_0 goes to zero.

Since the maximum degree Δ of the dependency graph does not change with the value of κ_0, there exists a constant $\bar{\kappa}$ such that $6\mathrm{e}p\Delta^2\le 1$ and $3\mathrm{e}q\Delta\le 1$ for all $\kappa_0\le\bar{\kappa}$. From (4.1), within an order of $\log|V|$ iterations the expected number of bad events is less than 1. Since $|V|$ is $O(1/r^d)$, the expected number of iterations of the PRS algorithm is $O(\log(1/r))$. This proves (i).

Furthermore, since the number of random objects resampled at iteration t of the algorithm is $|\mathsf{Res}_t|$, the expected running time of the algorithm is of order $\sum_{t=0}^{\infty}|\mathsf{Res}_t|$, which is less than $|V|/p$, by (4.1). Hence (ii) is established.

To prove (iii), recall that for parallel computation, each cell C_i is associated with a processor unit that can communicate with all the N_i neighboring cells and there is another processor acting as a top-node to communicate in parallel with all $n(r)$ processors associated with the cells. At the beginning of the algorithm, with the help of the top-node, all the cell processors can run in parallel to generate a fresh copy of initial state $\{\mathbf{X}_{C_i}, i=1,\dots,n(r),$ and $U_{i,j},\{i,j\}\in V\}$, where $\mathbf{X}_{C_i}\sim\mu_i$, $i=1,\dots,n(r)$, and $U_{i,j}\sim\mathsf{Unif}(0,1)$, $\{i,j\}\in V$. Then the bad events can be identified by making the cell processors interact in parallel with their respective neighbors. Hence the expected running time complexity of the initialization of the algorithm is $O(1)$. In each iteration of the algorithm the top-node can run a breadth-first search algorithm starting from bad events to identify the resampling cells $\mathcal{I}(\mathsf{Res})$. The top-node executes the breadth-first search algorithm in a sequence of steps where each step has $O(1)$ expected running time complexity. In the first step, in parallel all the cells correspond to the bad events and their neighboring cells are labelled as resampling cells. In every step after that, labeling of all the unlabeled neighboring cells of the resampling cells can be done in parallel. This labeling continues until no new cells are labeled as resampling.

Now we show that the expected number of steps in the breadth-first search is maximum for the first iteration of the algorithm. To see this, let $\mathbf{X}^{(t)}_{C_i}$ be the point configuration on the cell C_i, $i=1,\dots,n(r)\}$ at the beginning of the t^{th} iteration and RC_{t-1} be the set of cells labelled as resampling cells at the end of the $(t-1)^{th}$ iteration. Further, let $\overline{\mathsf{RC}}_{t-1}=\{1,\dots,n(r)\}\setminus\mathsf{RC}_{t-1}$. Then it is easy to see that $\{\mathbf{X}^{(t)}_{C_i}: i\in\mathsf{RC}_{t-1}\}$ is a fresh sample from the product measure $\prod_{i\in\mathsf{RC}_{t-1}}\mu_i$, and on the other hand, $\{\mathbf{X}^{(t)}_{C_i}: i\in\overline{\mathsf{RC}}_{t-1}\}$ is a realization of the desired Gibbs process restricted to the cells $C_i: i\in\{1,\dots,n(r)\}\setminus\mathsf{RC}_{t-1}$. Hence the distribution of $\{\mathbf{X}^{(t)}_{C_i}: i\in\overline{\mathsf{RC}}_{t-1}\}$ is absolutely continuous with respect to the product measure $\prod_{i\in\overline{\mathsf{RC}}_{t-1}}\mu_i$. Therefore, from the definition of Gibbs process, we can argue that there exists a point process $\{\widetilde{\mathbf{X}}_{C_i}: i=1,\dots,n(r)\}\sim\mu^{\otimes}$ such that $\mathbf{X}_{C_i}^{(t)} \subseteq \widetilde{\mathbf{X}}_{C_i}$ for each $i = 1, \dots, n(r)$. Using this observation, as a consequence of the fact that the initial configuration $\{\mathbf{X}_{C_i}^{(0)} : i = 1, \dots, n(r)\}$ is distributed as $\mu^{\otimes}$, the expected number of steps in the breadth-first search is higher for the first iteration.

Now we show that the expected number of steps in the breadth-first search for the first iteration is $O(1)$. Note that $\mathbf{X}_{C_1}^{(0)}, \dots, \mathbf{X}_{C_{n(r)}}^{(0)}$ are i.i.d. and the probability of $\mathbf{X}_{C_i}^{(0)} \neq \varnothing$ is upper bounded by $\widetilde{q} := 1 - \exp(-\kappa_0 b/\mathrm{v}_d)$, as shown earlier in this proof. Note that for each cell, the number of neighboring cells is at most a. At any step of the breadth-first search, for each resampling cell, on average at most $\widetilde{q}a$ neighboring cells are added to the resampling set. Because of the parallel computation, if we show that for each resampling cell added in the first step, if the expected number of cells that will be eventually labeled as resampling cells is bounded by a constant independent of r, then we can conclude that the expected running time of each iteration is $O(1)$. This is true when $\widetilde{q}a < 1$, because for each resampling cell at the first step of breadth-first search, the expected number of cells that will be eventually labeled as resampling cells is bounded by

$$a\big[1 + \widetilde{q}a\big[1 + \widetilde{q}a[1 + \cdots]\big]\big].$$

We complete the proof by stating that there exists $\bar{\kappa} > 0$ such that $\widetilde{q} < 1/a$ for all $\kappa_0 \le \bar{\kappa}$. $\square$

6.1. Comparison with well-known existing methods

In this subsection, we consider two well-known methods, namely, the naive rejection sampling and the dCFTP methods, and establish lower bounds on their expected running time complexities.

Naive rejection method: For a Gibbs process of the form (3.1), the naive rejection sampling method repeatedly simulates a sample $\mathbf{X}$ from ρ until it is accepted. The last sample has the target distribution μ. The acceptance probability at each iteration is $Z = \mathbb{E}_\rho[\exp(-\mathcal{U}(\mathbf{X}))]$, and thus the expected number of iterations is $1/Z$. Since the expected number of points generated in each iteration is κ (because ρ is κ-homogeneous), the expected running time of the naive algorithm is proportional to κ/Z. Below, we use a standard argument to show that κ/Z increases faster than an exponential function as r decreases to 0.

Consider the hypercube grid partitioning of Section 5.1. Note that each cell is at most as big as a cube with the edge length $2r$ and the number of cells $n(r)$ is at least $\lceil 1/2r \rceil$. Therefore, by ignoring the cross correlations between the cells and using the fact that there are at least $(n(r) - 2)^d$ hypercube cells, we obtain

$$\begin{aligned} Z &\le \prod_{i=1}^{(n(r)-2)^d} \mathbb{E}_\rho\big[\exp\big(-\mathcal{U}(\mathbf{X}_{C_i})\big)\big] \\ &\le \big(\mathbb{E}_\rho\big[\exp\big(-\mathcal{U}(\mathbf{X}_C)\big)\big]\big)^{(\frac{1-4r}{2r})^d} =: \varepsilon^{(\frac{1-4r}{2r})^d}, \end{aligned}$$

where $C = [0, 2r]^d$, $\varepsilon = \mathbb{E}_\rho[\exp(-\mathcal{U}(\mathbf{X}_C))]$, and the inequalities hold because $n(r) \geq 1/2r$ and $\varepsilon \leq 1$.

Since for any fixed $\kappa_0 > 0$, the value of ε is strictly less than 1 and does not depend on r (because ε is the same if $C = [0, 1]^d$, ρ is the distribution of $(2^d\kappa_0/\mathsf{v}_d)$-homogeneous PPP on C, and the interaction range of the potential function $\mathcal{U}$ is 1). By using the value of κ and the upper bound on Z, we obtain a lower bound on the expected running time complexity κ/Z, given by

$$\frac{\kappa}{Z} \geq \left(\frac{\kappa_0}{\mathsf{v}_d r^d}\right)\left(\frac{1}{\varepsilon}\right)^{(\frac{1-4r}{2r})^d},$$

which increases faster than an exponential function, as r goes to 0.

Dominated CFTP: In order to establish a lower bound on the expected running time of a dominated CFTP method, we first briefly state the general description of the method (for a detailed description, we refer the reader to, e.g., Kendall and Møller [19]). Let $\mathbf{D} = \{\mathbf{D}(t) : t \in \mathbb{R}\}$ be the (free) birth-and-death process on $S = [0, 1]^d$ with birth rate κ, where each birth is a (marked) point uniformly and independently selected on S and alive for a random time exponentially distributed with mean one. It is not difficult to show that the steady-state distribution of $\mathbf{D}$ is ρ. Since the target Gibbs distribution $\mu \ll \rho$, using coupling, it is possible to construct a process $\mathbf{Z} = \{\mathbf{Z}(t) : t \in \mathbb{R}\}$ such that $\mathbf{Z}(t) \subseteq \mathbf{D}(t)$ for all $t \in \mathbb{R}$ and the steady-state distribution of $\mathbf{Z}$ is μ. Any dCFTP method consists of two steps: (i) constructing the dominating spatial birth-and-death process $\{\mathbf{D}(t) : -s \leq t \leq 0\}$ backward in time, for some $s > 0$, starting at time zero with $\mathbf{D}(0) \sim \rho$, and (ii) use *thinning* on the dominating process to obtain an upper bounding process $\{\mathbf{U}_s(t) : t \geq -s\}$ with $\mathbf{U}_s(-s) = \mathbf{D}(-s)$ and a lower bounding process $\{\mathbf{L}_s(t) : t \geq -s\}$ with $\mathbf{L}_s(-s) = \varnothing$, forward in time such that the condition

$$\mathbf{L}_s(t) \subseteq \mathbf{Z}(t) \subseteq \mathbf{U}_s(t) \subseteq \mathbf{D}(t)$$

is guarantee to hold for all $t \geq -s$. If $\mathbf{U}_s$ and $\mathbf{L}_s$ *coalescence* at time 0, that is, $\mathbf{U}_s(0) = \mathbf{L}_s(0)$, then $\mathbf{U}_s(0)$ is a perfect sample from the target distribution μ. If there is no coalescence, then in the next iteration, increase s and extend the dominating process further backward to time $-s$ and repeat the same procedure.

The criteria for thinning depends on the definition of the target distribution. However, the dominating process depends only on ρ. Let $\widetilde{T}$ be the *backward coalescence* time given by

$$\widetilde{T} := \min\{s \geq 0 : \mathbf{L}_s(0) = \mathbf{U}_s(0)\}.$$

The running time complexity of a dCFTP method is at least of order of the number of computations needed to construct the dominating process $\{\mathbf{D}(t) : -\widetilde{T} \leq t \leq 0\}$. Let

$$T_s = \{t \geq 0 : \text{none of the points in } \mathbf{D}(-s) \text{ are alive at } -s + t\}.$$

Since the dominating process $\mathbf{D}$ is time-reversible and $\mathbf{D}(0) \sim \rho$, we have $\mathbf{D}(-s) \sim \rho$ for all $s \geq 0$. Hence, the distribution of T_s does not depend on s.

Since $\mathbf{U}_s(-s) = \mathbf{D}(-s)$ and $\mathbf{L}_s(-s) = \varnothing$, if a point of $\mathbf{D}(-s)$ is alive at time 0, then $\mathbf{U}_s(0) \neq \mathbf{L}_s(0)$. Therefore, $T_s \geq s$ implies that $\widetilde{T} \geq s$, and thus s is at least T_s for the coalescence to happen. As a consequence, the expected running time complexity of any dCFTP algorithm is at least of order of the expected number of births in $\mathbf{D}$ that are generated during an interval of length $\mathbb{E}[T_s]$, which is $\kappa\mathbb{E}[T_s]$ because the births in the dominating process are Poisson with rate κ.

Further, recall that each birth is alive for a random time independently and exponentially distributed with mean one. Conditioned on $|\mathbf{D}(-s)| = m$, T_s is the maximum of m i.i.d. mean one exponential random variables. Since $|\mathbf{D}(-s)| \sim \mathsf{Poi}(\kappa)$ for all s, we have

$$\begin{aligned}
\mathbb{E}[T_s] &= \sum_{m=0}^{\infty} \mathrm{e}^{-\kappa}\frac{\kappa^m}{m!}\mathbb{E}\big[T_s \mid \big|\mathbf{D}(-s)\big| = m\big] \\
&\geq \sum_{m\geq\kappa/2} \mathrm{e}^{-\kappa}\frac{\kappa^m}{m!}\mathbb{E}\big[T_s \mid \big|\mathbf{D}(-s)\big| = m\big] \\
&= \sum_{m\geq\kappa/2}^{\infty} \mathrm{e}^{-\kappa}\frac{\kappa^m}{m!}H(m),
\end{aligned}$$

where $H(m) = \sum_{i=1}^{m}\frac{1}{i}$ is the m^{th} harmonic number. Using the fact that $H(m) \geq \log m$ for all $m \geq 1$,

$$\mathbb{E}[T_s] \geq \log(\kappa/2)\mathbb{P}\big(\big|\mathbf{D}(-s)\big| \geq \kappa/2\big).$$

From Chernoff bound on the tail probabilities of Poisson distribution, there exists a constant $a > 0$ such that $\mathbb{P}(|\mathbf{D}(-s)| \geq \kappa/2) = 1 - \exp(-a\kappa)$, for all values of κ. In conclusion, the expected running time complexity of any dCFTP algorithm is at least of order of $\kappa \log \kappa$, which is of order of $\frac{1}{r^d}\log(\frac{1}{r})$ for any κ_0.

7. Simulations

In this section, we take $S = [0, 1]^2$ and apply Algorithm 2 to generate perfect samples of PSM model, hard-core process and Strauss process. We ignore the case of area-interaction process as the implementation is similar to that of PSM model and expected to have same order of complexity.

We estimate the expected number of iterations of the algorithm for different values of the model parameters. As long as the samples on each cell are perfect, the reported results are the same for any choice of existing method to generate samples from μ_i's. The following simulation results are obtained using Python programming, and the corresponding codes are available at https://github.com/saratmoka/PRS_with_DCFTP

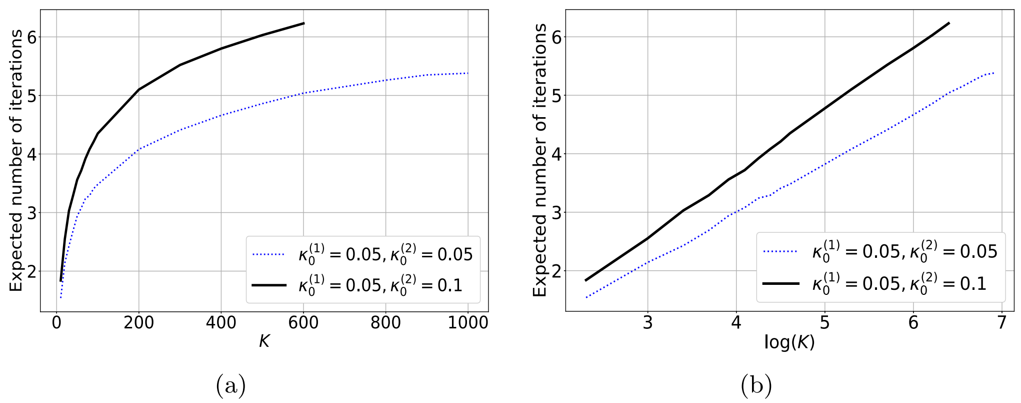

Figure 5. Expected number of iterations of Algorithm 2 for PSM model as functions of K (panel (a)) and $\log(K)$ (panel (b)).

PSM model: Consider the PSM model with the interaction range $2r = 1/K$ for some $K \geq 1$. We take the intensities of type-1 and type-2 points to be $\kappa_0^{(1)}/(\pi r^2)$ and $\kappa_0^{(2)}/(\pi r^2)$, respectively. Figure 5 plots the estimated expected number of iterations of the algorithm as a function of K. Perfect samples from μ_i on each cell i are generated using the dCFTP method by Kendall [18]. Observe that, the expected number of iterations of the algorithm seems to be $O(\log(1/r))$ as shown in Theorem 6.1 for small values of $\kappa_0 = \kappa_0^{(1)} + \kappa_0^{(2)}$; see Figure 5(b).

Hard-core and Strauss processes: Consider the Strauss process. Suppose that the intensity of the reference PPP is $\kappa = \kappa_0/(\mathrm{v}_d r^d)$. Recall that the Strauss process is a hard-core process if $\gamma = 0$. The panels (a), (b) and (c) in Figure 6 correspond to κ_0 is equal to 0.1, 0.2 and 0.25, respectively. Each panel has two curves correspond to $\gamma = 0$ (i.e., hard-core process) and $\gamma = 0.5$. Each curve denotes the estimated expected number of iterations of the algorithm as a function of K, when the interaction range $2r = 1/K$. Perfect samples from μ_i on each cell i are generated using the dCFTP method by Huber [15]. Again observe that, the expected number of iterations of the algorithm seems to be $O(\log(1/r))$; see Figures 6(d) and 6(e). These results suggest that $\bar{\kappa}$ in Theorem 6.1 can be at least 0.25.

Figure 6(f) corresponds a hard-core process with the interaction range $2r = 0.01$, and it compares the expected number of iterations of the new algorithm with that of the method proposed by Guo and Jerrum [13], for different values of κ_0. The complexity of Algorithm 2 is slightly higher than that of Guo and Jerrum [13]. However, as mentioned earlier, the algorithm of Guo and Jerrum [13] is restricted to hard-core processes, whereas the new algorithm can be applied to more general Gibbs processes.

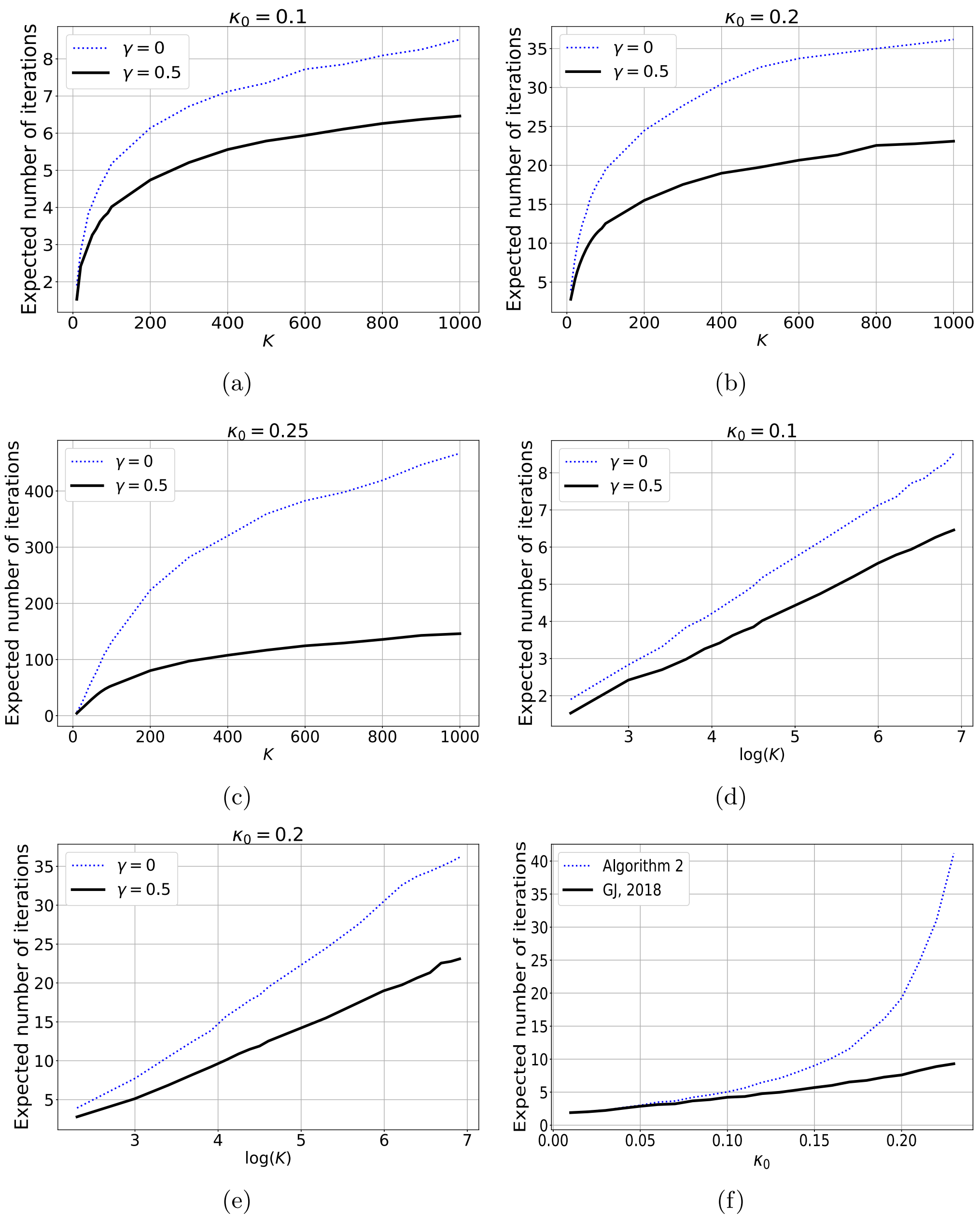

Figure 6. For panels (a)–(c) (respectively, (d)–(e)), the expected number of iterations of Algorithm 2 is plotted as a function of K (resp., as a function of $\log(K)$) for the Strauss process with the interaction range $2r = 1/K$. Panel (f) compares the expected number of iterations of Algorithm 2 with that of the algorithm of Guo and Jerrum [13] for a hard-core model with the interaction range $2r = 0.01$.

8. Conclusion

In this paper, we considered the problem of perfect sampling for Gibbs point processes with a finite interaction range $2r$, defined on $S \subseteq \mathbb{R}^d$. We proposed a new perfect sampling algorithm by combining the existing perfect sampling methods and the partial rejection sampling proposed by Guo, Jerrum and Liu [14]. For pairwise interaction processes, penetrable spheres mixture models and area-interaction processes that are absolutely continuous with respect to a κ-homogeneous Poisson point process on $S = [0, 1]^d$, we showed that if $\kappa = \kappa_0/(\mathrm{v}_d r^d)$, the proposed algorithm can be implemented with the expected running time complexity of $O(\log(1/r))$ as r goes to 0, for sufficiently small values of κ_0. We illustrated our findings using several simulation results. From these simulations, we notice that the value of κ_0 can be at least 0.25 for Strauss processes. However, at this stage, we do not have a theoretical justification to support this claim, and we would like to address this in future research.

Acknowledgments

This work has been supported by the Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS), under grant number CE140100049. We would like to thank Professor Michel Mandjes for bringing the partial rejection sampling to our attention.

References

[1] Baddeley, A. and Nair, G. (2012). Fast approximation of the intensity of Gibbs point processes. *Electron. J. Stat.* **6** 1155–1169. MR2988442 https://doi.org/10.1214/12-EJS707

[2] Baddeley, A. and Turner, R. (2005). An R package for analyzing spatial point patterns. *J. Stat. Softw.* **12** 1–42.

[3] Baddeley, A.J. and van Lieshout, M.N.M. (1995). Area-interaction point processes. *Ann. Inst. Statist. Math.* **47** 601–619. MR1370279 https://doi.org/10.1007/BF01856536

[4] Berthelsen, K.K. and Møller, J. (2006). Bayesian analysis of Markov point processes. In *Case Studies in Spatial Point Process Modeling*. *Lect. Notes Stat.* **185** 85–97. New York: Springer. MR2232124 https://doi.org/10.1007/0-387-31144-0_4

[5] Bezáková, I., Galanis, A., Goldberg, L.A., Guo, H. and Štefankovič, D. (2016). Approximation via correlation decay when strong spatial mixing fails. In 43*rd International Colloquium on Automata*, *Languages*, *and Programming*. *LIPIcs*. *Leibniz Int*. *Proc*. *Inform*. **55** Art. No. 45, 13. Wadern: Schloss Dagstuhl. Leibniz-Zent. Inform. MR3577106

[6] Chiu, S.N., Stoyan, D., Kendall, W.S. and Mecke, J. (2013). *Stochastic Geometry and Its Applications*, 3rd ed. *Wiley Series in Probability and Statistics*. Chichester: Wiley. MR3236788 https://doi.org/10.1002/9781118658222

[7] Feng, W., Sun, Y. and Yin, Y. (2017). What can be sampled locally? In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, *PODC*'17 121–130. New York, NY, USA.

[8] Feng, W. and Yin, Y. (2018). On local distributed sampling and counting. In *Proceedings of the* 2018 *ACM Symposium on Principles of Distributed Computing*, *PODC*'18 189–198. New York, NY, USA.

[9] Ferrari, P.A., Fernández, R. and Garcia, N.L. (2002). Perfect simulation for interacting point processes, loss networks and Ising models. *Stochastic Process*. *Appl*. **102** 63–88. MR1934155 https://doi.org/10.1016/S0304-4149(02)00180-1

[10] Fill, J.A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *Ann. Appl. Probab.* **8** 131–162. MR1620346 https://doi.org/10.1214/aoap/1027961037

[11] Fill, J.A. and Huber, M. (2000). The randomness recycler: A new technique for perfect sampling. In 41*st Annual Symposium on Foundations of Computer Science* (*Redondo Beach*, *CA*, 2000) 503–511. Los Alamitos, CA: IEEE Comput. Soc. Press. MR1931847 https://doi.org/10.1109/SFCS.2000.892138

[12] Garcia, N.L. (2000). Perfect simulation of spatial processes. *Resenhas* **4** 283–325. MR1797367

[13] Guo, H. and Jerrum, M. (2018). Perfect simulation of the hard disks model by partial rejection sampling. In 45*th International Colloquium on Automata*, *Languages*, *and Programming*. *LIPIcs*. *Leibniz Int*. *Proc*. *Inform*. **107** Art. No. 69, 10. Wadern: Schloss Dagstuhl. Leibniz-Zent. Inform. MR3830000

[14] Guo, H., Jerrum, M. and Liu, J. (2017). Uniform sampling through the Lovász Local Lemma. In *STOC'*17 *– Proceedings of the* 49*th Annual ACM SIGACT Symposium on Theory of Computing* 342–355. New York: ACM. MR3678193

[15] Huber, M. (2012). Spatial-birth-death swap chains. *Bernoulli* **18** 1031–1041. MR2948911 https://doi.org/10.3150/10-BEJ350

[16] Huber, M.L. (2016). *Perfect Simulation*. *Monographs on Statistics and Applied Probability* **148**. Boca Raton, FL: CRC Press. MR3443710

[17] Huber, M.L., Villella, E., Rozenfeld, D. and Xu, J. (2012). Bounds on the artificial phase transition for perfect simulation of hard core Gibbs processes. *Involve* **5** 247–255. MR3044611 https://doi.org/10.2140/involve.2012.5.247

[18] Kendall, W.S. (1998). Perfect simulation for the area-interaction point process. In *Probability Towards* 2000 (*New York*, 1995). *Lect*. *Notes Stat*. **128** 218–234. New York: Springer. MR1632588 https://doi.org/10.1007/978-1-4612-2224-8_13

[19] Kendall, W.S. and Møller, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Adv. in Appl. Probab.* **32** 844–865. MR1788098 https://doi.org/10.1239/aap/1013540247

[20] Moka, S.B., Juneja, S. and Mandjes, M.R.H. (2017). Perfect sampling for Gibbs processes with a focus on hard-sphere models. arXiv:1705.00142 [math.PR].

[21] Møller, J. (2001). A review of perfect simulation in stochastic geometry. In *Selected Proceedings of the Symposium on Inference for Stochastic Processes* (*Athens*, *GA*, 2000). *Institute of Mathematical Statistics Lecture Notes – Monograph Series* **37** 333–355. Beachwood, OH: IMS. MR2002519 https://doi.org/10.1214/lnms/1215090699

[22] Møller, J., Pettitt, A.N., Reeves, R. and Berthelsen, K.K. (2006). An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika* **93** 451–458. MR2278096 https://doi.org/10.1093/biomet/93.2.451

[23] Møller, J. and Waagepetersen, R.P. (2004). *Statistical Inference and Simulation for Spatial Point Processes*. *Monographs on Statistics and Applied Probability* **100**. Boca Raton, FL: CRC Press/CRC. MR2004226

[24] Widom, B. and Rowlinson, J.S. (1970). New model for the study of liquid-vapor phase transitions. *J*. *Chem*. *Phys*. **52** 1670–1684.

Received March 2019 and revised August 2019