Les Cahiers du GERAD ISSN: 0711–2440

On Tractable Markov Random Field Models and MapReduce Algorithms: Towards a Practitioner's Toolbox

P. Manggala, L. Devroye,
R. Dimitrakopoulos

G–2013–64

September 2013

On Tractable Markov Random Field Models and MapReduce Algorithms: Towards a Practitioner's Toolbox

Putra Manggala
Luc Devroye
Roussos Dimitrakopoulos*

COSMO – Stochastic Mine Planning Laboratory
Department of Mining and Materials Engineering
McGill University
Montreal (Quebec) Canada, H3A 2A7

putra.manggala@mail.mcgill.ca
luc@cs.mcgill.ca
roussos.dimitrakopoulos@mcgill.ca

* *and GERAD*

September 2013

Les Cahiers du GERAD
G–2013–64

Abstract: In the era of big data, data sets have been growing very large, and interest for algorithms and computation framework that handle such large scales is increasing. The number of computing cores per chip has also increased. Instead of developing ingenious ways of speeding up convergences and obtaining better approximations for a smaller subclass of the problems, it is interesting to simply "throw more cores" at exact algorithms and get a speed-up which scales accordingly. This allows practitioners who are not specialists to use the algorithms more efficiently. The MapReduce framework is one of the most widely used paradigms for parallel computation, and we show how to cast the exact deterministic algorithms for our customized spatial model in this framework. Abstract. Higher-order Markov random fields, i.e., Markov random fields with higher-order interactions, have been shown to be able to represent large-scale properties of typical spatial scenes via MCMC simulation. We present a Markov random field parametrization such that the specification of clique configurations that include higher-order interactions is included in a methodical fashion. This ought to be useful for a practitioner with respect to specifying the type of images desired. The general Markov random field model commonly requires approximation techniques due to its intractable normalization constant, and inference and simulation techniques for models with high dependence via MCMC tend to be time-consuming and may involve algorithms with sensitive convergence criteria. These algorithms are not suitable for practitioners or spatial modellers who are not well-versed in MCMC, and it is therefore more desirable for them to work with models that are able to reproduce spatial structures, while still being easier to wield. We construct variants of the Markov random field that seek to achieve the same goal, but whose inference and simulation is tractable without MCMC, and cast the latter in the MapReduce framework. We also consider the MapReduce formulation of MCMC simulation algorithms for Markov random fields in order to leverage the power of parallel computing.

Key Words: MCMC, Markov random fields, MapReduce, Multidimensional Markov Chain Model.

1 The Need for Higher-Order Spatial Models and the MapReduce Framework

It is reasonable for a geostatistician, or anyone who is to mathematically model the spatial data, to think that an observation that is made at a certain spatial location is generally not independent from the observation at a slightly different location. For statistical tasks, it is thus imperative to rigorously model either the extent of the correlation or the exact nature of the association is. For one dimensional data, a large body of theory has been developed, usually under the umbrella of time-series analysis (Durbin et al., 2001). The autocorrelation function largely quantifies the dependency between observations in this case. In a space of larger dimension, however, we must consider various directions, which is achieved by way of the variogram function (Goovaerts, 1997). To an extent this function characterizes the spatial process generating the observations. Practically, the variogram measures pairwise geological distances. This is a form of modelling, however it is not very extensive: for example, it does not give us a probability density function. Moreover, pairwise dependencies are quite crude and cannot capture complex spatial phenomena. Variogram models and parameters cannot accurately capture either the discrete complex sinusoidal-like patterns of channels and facies, or, in general, discrete spatial objects and textures.

Dubbed the variogram-free statistics, multiple point geostatistics came into play, Strebelle (2002), and was able to generate different types of discrete images by essentially applying a function, which takes as input a set of an priori defined spatial patterns, termed the training images and outputs images which to a degree consist of these inputted patterns. Modelling using high-order spatial cumulants has been shown to be effective in characterizing complex patterns (Mustapha and Dimitrakopoulos, 2010, 2010), as it allows for large possible combinations of random variables indexed by a set of spatial sites, and thus captures complex spatial patterns. The main approach discussed in this paper aims to reach the same goal.

Markov random fields with higher-order interactions have been shown to be able to represent large-scale properties of spatial scenes (Tjelmeland and Besag, 1998). However, their inference and simulation are highly time-consuming and unwieldable by non-experts, i.e., individuals with domain knowledge in the spatial data they work with, but less in statistical computing. The algorithmically driven approach la the multiple point geostatistics has been popular here. The SNESIM algorithm (Strebelle, 2002) makes use of pattern frequencies in the training image which are tightly coupled with the domain knowledge of the spatial field, and thus is very usable in practice. However, a criticism of this approach is that it is unclear how the method treats patterns that are not present in the training image, but should have a nonzero probability of occuring in the data (Stien and Kolbjornsen, 2011). Markov random field based models are consistent statistical models that, in a sense, abstractly summarize the observed data into parametrized probability distributions and thus can consider all the possible images associated with the models. Note that any brute force of manually enumerating the right discrete images for any task is impossible. A similar type of calculation appears in some important statistical tasks involving the Markov random field. Consider a small binary two dimensional image with 500×500 sites. Here we have $2^{500\times500}$ possible images that could occur on this set of sites; an astronomical quantity that is larger than the number of atoms in the universe. Even in the present Digital Age (Bollacker, 2010), this is intractable.

The size of geostatistical datasets can be large, and specialized techniques have been adopted for their analyses (Banerjee et al. 2008; Cornford et al., 2005). For the algorithms in this paper, we adopt a very widely used parallel programming framework, for which a robust open-source implementation is available. Originated by a paper from Google (Dean and Ghemawat, 2008), the MapReduce framework has emerged as one of the most widely used parallel computing platforms for processing data on large scales. This framework is heavily used in companies that are brimming with data, such as Google, Amazon, and Facebook, and has been adopted more recently due to its open source implementation, Hadoop. The framework allows for easy parallelization, and it allows one to interleave sequential and parallel computations within the same pipeline.

A tractable model can be obtained by adopting the divide and conquer paradigm. This probabilistic model, defined by imposing a directed model on an ordered set of tractable Markov random fields, can be used to approximate higher-order Markov random fields, by way of using similar parametrization and dependency graph. An example appeared in Nerhus (2009), Qian and Titterington (1992, and was termed

the Multidimensional Markov Chain model. The forward-backward algorithms (Baum et al., 1970) has been adopted and extended by many can be used to tractably perform parameter estimation and simulation in this model.

We present a generalization of this tractable model, which seeks to parallel the ability of the higher-order Markov random field to reproduce the large-scale properties of a typical spatial scene, and cast the simulation algorithms in the MapReduce framework. Moreover, we cast the MCMC simulation algorithms usable for the higher-order Markov random fields in the MapReduce framework, and lastly, present some Markov random fields simulation results.

2 Markov Random Fields

Discrete images, i.e., a collection of pixels arranged in a grid where each pixel can assume a value from a finite set of values, can be envisaged as an element of finite product spaces. Consider a graph $G = (V, E)$ where the set of vertices V corresponds to the set of spatial sites S, where vertex i corresponds to the site s_i. For every vertex $i \in V$, let Ω_i be a finite space of states $z(s_i)$, and the product $\Omega = \prod_{i \in V} \Omega_i$ is the space of finite configurations $z = (z(s_i))_{i \in V}$. In this paper, we consider the setting where $\Omega_i = \Omega_j$ for all $i, j \in V$. For $U \subset V$, let $\Omega_U = \prod_{i \in U} \Omega_i$ denote the space of configurations $z(s_U) = (z(s_i))_{i \in U}$. A strictly positive probability mass function $\mathbf{P}$ on Ω, where $\mathbf{P}(z) > 0$ for every $z \in \Omega$ is called a random field.

The random field $\mathbf{P}$ is a Markov random field with respect to G if for all $z \in \Omega$, for all $i \in V$,

$$\mathbf{P}(z(s_i) \mid \{z(s_j) : j \neq i\}) = \mathbf{P}(z(s_i) \mid \{z(s_j) : j \in N_i\}) \tag{1}$$

where $N_i \equiv \{j : (i, j) \in E\}$ is the set of neighbors of i. The point is that we want to define G such that the neighborhoods are relatively small while still representative of the true dependence to minimize computation. Instead of describing the random field using its global characteristics, such as the mean or the variogram function, we have described it using its conditional distributions, which are vertex-centric. This type of description is useful when the observed spatial phenomenon is given in terms of some sort of local conditional behavior. In order to achieve this, we must consider the relationship between these conditional distributions and $\mathbf{P}$. In general, conditional distributions described in 1 are not compatible in this way. We describe a general family of conditional distributions called the Gibbs specifications that are compatible. These specifications are characterized by the potentials, which would be the main driving force in our spatial modelling. A celebrated experimental result from statistical physics, which pertains to our model, can be summarized as follows. A piece of iron is exposed to a magnetic field that is gradually increased from zero to a maximum, and then back to zero. For low enough temperatures, the iron maintains residual magnetization (ferromagnetism). Within this spectrum of temperature, there exists a critical temperature. The Ising model (Ising 1925), can be used to describe this phenomenon. The particles of the piece of iron are modeled to assume a location in R^3, and each particle can be in either of two states, representing the magnetic physical 'spin-up' and 'spin-down'. The Gibbs distribution can then be used as a probability mass function of the configuration of these particles. The probability space for each particle can then be coded as $\{-1, 1\}$, and the cardinality of this set is commonly associated with the Ising model. The Gibbs distribution associated to probability space with higher cardinalities is commonly associated with the Potts mode (Baxter, 1973). The special features of the binary probability space allow analysis not yet replicable for the Potts model. For an expository of these models, one can refer to Grimmett (2010).

For discrete data, we can write the probability measure as:

$$\mathbf{P}(z) = \exp(Q(z)) / \sum_{y \in \Omega} \exp(Q(y)),$$

where the sum exists and is finite as probability measures sum to 1. In statistical mechanics, $-Q(\cdot)$ is commonly termed the potential function (Winkler, 2003), the negative of this, the negpotential function (Cressie and Wikle, 2011), and the normalizing term $\sum_{y \in \Omega} \exp(Q(y))$ is termed either the partition function, or the normalizing constant.

2.1 MCMC algorithms

Given $\mathbf{P}$, a Markov chain with state space Ω is constructed with stationary distribution $\mathbf{P}$, and a Markov chain Monte Carlo simulation (MCMC) is done by running such a Markov chain. We define a strictly positive Markov kernel K for which $\mathbf{P}$ is the invariant distribution. For every $U \subset V$, a Markov kernel on Ω is defined by:

$$K_U(z, z') = \begin{cases} Z_U^{-1} \exp\left(Q\left(z'\left(s_U\right) z\left(s_{V\setminus U}\right)\right)\right) & \text{if } z'\left(s_{V\setminus U}\right) = z\left(s_{V\setminus U}\right) \\ 0 & \text{otherwise.} \end{cases}$$

$$Z_U = \sum_{y(s_U)} \exp\left(Q\left(y\left(s_U\right) z\left(s_{V\setminus U}\right)\right)\right).$$

The set of Markov kernels $\{K_U : U \subset V\}$ is called the local characteristics of $\mathbf{P}$. We show that the Gibbs field $\mathbf{P}$ and its local characteristics fulfill the detailed balance equation, which is described below, showing that $\mathbf{P}$ is the stationary distribution of the local characteristics. For all $z, z' \in \Omega$ and $U \subset V$,

$$\mathbf{P}(z) K_U(z, z') = \mathbf{P}(z') K_U(z', z).$$

Now both sides vanish iff $z'\left(s_{V\setminus U}\right) \neq z\left(s_{V\setminus U}\right)$. Expanding, we get the equality:

$$\exp(Q(z)) \frac{\exp\left(Q\left(z'\left(s_U\right) z\left(s_{V\setminus U}\right)\right)\right)}{\sum_{y(s_U)} \exp\left(Q\left(y\left(s_U\right) z\left(s_{V\setminus U}\right)\right)\right)},$$

$$= \exp(Q(z')) \frac{\exp\left(Q\left(z\left(s_U\right) z'\left(s_{V\setminus U}\right)\right)\right)}{\sum_{y(s_U)} \exp\left(Q\left(y\left(s_U\right) z'\left(s_{V\setminus U}\right)\right)\right)},$$

which implies detailed balance. Summing both sides over z, we obtain the invariance of $\mathbf{P}$ with respect to its local characteristics. The Gibbs sampler (Geman and Geman, 1984), is used to simulate samples from the joint distribution $\mathbf{P}$. The local characteristics are used to sample from each variable. The sequential sweep Gibbs sampling variant visits the vertices in a certain ordering and performs updating at every visit. Let the ordering be $\sigma = (1, \ldots, n)$ where $|V| = n$, and a Markov kernel can be defined as $K(z, z') = K_{\{1\}} \cdots K_{\{n\}}(z, z')$. Thus running the chain with multiple sweeps produces a sample from $\nu K \cdots K$ where ν is the initial distribution on Ω. Now the Gibbs fields are invariant with respect to the local characteristics, and hence for the composition K of local characteristics, theoretically the sample from the Markov chain is one from a distribution close to $\mathbf{P}$. Indeed, for every $z \in \Omega$,

$$\lim_{n\to\infty} \nu K^n(z) = \mathbf{P}(z),$$

uniformly in all initial distributions ν. There is no restriction in choosing σ. One can also choose σ at random (Winkler, 2003).

The Metropolis-Hastings algorithm (Metropolis et al., 1953) is as follows: let Q be the negpotential function of interest and z is the current configuration. The update step is defined as:

1. (**Proposal**) A new configuration z' is proposed by sampling from a probability distribution $q(z, \cdot)$ on Ω.
2. **(Acceptance)** The new configuration z' is accepted with probability $\alpha \equiv \alpha(z, z') = \min\left\{1, \frac{\exp(Q(z'))q(z',z)}{\exp(Q(z))q(z,z')}\right\}$.

One possible $q(z, \cdot)$ is as follows. Fix a vertex i with value $v = z(s_i)$ and choose a new value uniformly at random from $\Omega_i \backslash z(s_i)$, say v^* and and then accepts $z(s_i) = v^*$ with probability α. This process allows us to simplify α , as if z, z' differ by more than one site, then $q(z, z') = z(z', z) = 0$, and if they differ by exactly one site i, then $q(z, z') = q(z', z) = \frac{1}{|\Omega_i|-1}$. This implies $\alpha = \min\{1, \exp(Q(z') - Q(z))\}$. If $Q(z') \geq Q(z)$, then z' is accepted as the new configuration, otherwise z' is accepted with a probability decreasing with

the increment of $Q(z) - Q(z')$. Note that the algorithm does not require us to normalize any probability distribution, i.e., we do not need the knowledge of the local characteristics. This may be desirable if the state space is large.

As an example, in the case of the Ising models, the acceptance probability is:

$$\min\left\{1, \exp\left(\theta\left(\#\left(z\left(s_i\right) = v^*\right) - \left(\#z\left(s_i\right) = v\right)\right)\right)\right\},$$

where $\#\left(z\left(s_i\right) = v\right)$ is the number of agreeing neighbors of i when $z\left(s_i\right)$ is assigned to be v.

For the Ising models with $\theta > 0$, the intuition is that if $\#\left(z\left(s_i\right) = v^*\right) \geq \#\left(z\left(s_i\right) = v\right)$, then the algorithm is proposing a change to a smoother image with more number of agreeing neighbors and will be accepted with probability 1. Otherwise, the algorithm is proposing a change to a coarser image with fewer agreeing neighbors and will be accepted with probability less than 1.

The explicit expression of the transition matrix of the Metropolis Markov chain is as follows:

$$K\left(z, z'\right) = \begin{cases} q\left(z, z'\right) \exp\left(\left(Q\left(z'\right) - Q\left(z\right)\right)^+\right) & \text{if } z \neq z' \\ 1 - \sum\limits_{y \in \Omega \setminus \{z\}} K\left(z, y\right) & \text{if } z = z'. \end{cases}$$

Convergence results are similar to the Gibbs sampling. Let q be symmetric, then $\mathbf{P}$ and K satisfies the detailed balance condition. It suffices to show this for $z \neq z'$. The identity is as follows:

$$\exp\left(Q\left(z\right)\right) \exp\left(\left(Q\left(z'\right) - Q\left(z\right)\right)^+\right) = \exp\left(Q\left(z'\right)\right) \exp\left(\left(Q\left(z\right) - Q\left(z'\right)\right)^+\right).$$

If $Q\left(z\right) \geq Q\left(z'\right)$, the left-hand side equals:

$$\begin{aligned} &\exp\left(Q\left(z'\right)\right) \exp\left(Q\left(z\right) - Q\left(z'\right)\right) = \exp\left(Q\left(z\right)\right) \\ &= \exp\left(Q\left(z\right)\right) \exp\left(\left(Q\left(z'\right) - Q\left(z\right)\right)^+\right). \end{aligned}$$

More advanced algorithms are available for simulation of Markov random fields. The Swendsen-Wang (SW) algorithm (Wang and Swendsen, 1990) constructs a Markov chain which updates clusters of sites. The SW algorithm has been shown to be very fast for simple Ising or Potts model, including the cases with phase transition, however, the complex dependency of the higher-order Markov random field has limited the computational gains of the SW algorithm (Morris, 1999). We describe the algorithm for a simple Potts model with probability mass function:

$$\mathbf{P}\left(z\right) \propto \exp\left(\theta \sum_{(i,j) \in E} 1_{[z(s_i) = z(s_j)]}\right).$$

A random field, Y, is defined conditionally on Z on the set of edges E such that the $y\left(s_i, s_j\right)$'s for all $(i, j) \in E$ are independent, i.e.:

$$\mathbf{P}\left(y\left(s_i, s_j\right) \mid z\right) \propto \exp\left(-\theta 1_{[z(s_i) = z(s_j)]}\right) 1_{\left[0 \leq y(s_i, s_j) \leq \exp\left(\theta 1_{[z(s_i) = z(s_j)]}\right)\right]},$$

where $y\left(s_i, s_j\right)$ can be considered as a continuous "bond" variable between the sites s_i and s_j. We then have the conditional distribution:

$$\mathbf{P}\left(z \mid y\right) \propto \prod_{(i,j) \in E} 1_{\left[0 \leq y(s_i, s_j) \leq \exp\left(\theta 1_{[z(s_i) = z(s_j)]}\right)\right]},$$

with joint distribution $\mathbf{P}\left(z, y\right) = \mathbf{P}\left(y \mid z\right) \mathbf{P}\left(z\right)$ and the desired marginal distribution $\mathbf{P}\left(z\right)$. The algorithm is run by alternatingly updating Y and Z. We consider the intuition of the auxiliary random field. For

$y(s_i, s_j) > 1$, we must have $\exp\left(\theta 1_{[z(s_i)=z(s_j)]}\right) > 1$, or equivalently, $z(s_i) = z(s_j)$, i.e., $y(s_i, s_j) > 1$ constrains $z(s_i)$ and $z(s_j)$ to have the same value. Conversely, if $z(s_i)$ and $z(s_j)$ are in the same state, we have that:

$$\mathbf{P}(y(s_i, s_j) > 1 \mid z(s_i) = z(s_j)) = 1 - \exp(-\theta).$$

Since it is only important whether $y(s_i, s_j)$ is greater or less than one, we may think of $y(s_i, s_j)$ as binary bond variables. Thus the bond variable is present between two sites in the same state with probability $1 - \exp(-\theta)$. Sampling from Y thus involves placing bonds between neighboring sites of the same state with probability $1 - \exp(-\theta)$ and omitting bonds between neighboring sites of differing states.

Upon placing the bonds, the conditional distribution $\mathbf{P}(z \mid y)$ says that all configurations where bonded sites are of the same state are equally probable. Thus, updating z entails forming clusters of connected sites and assign to all the sites of a cluster the same value, chosen uniformly from the allowed states.

Coupling from the past (CFTP) allows for exact simulation, done by ingeniously setting up the Gibbs chains (Haggstrom and Nelander, 1999), such that the output is an exact sample from $\mathbf{P}$. Consider running $|\Omega|$ copies of the Markov chain formed by the Gibbs sampler from every possible initial state. Let the same updating function and sequence of random numbers be used for all the chains. Two chains are said to have achieved coalescence if the two chains at one step reach the same state. From the point of coalescence, the two chains will have the same sample path, as the same updating function and random numbers are used. Let the number of iterations M be fixed and the update function of the Markov chains be denoted by $\phi : \Omega \times U \to \Omega$, where the chains are first run from time $-M$ to 0. The output from a chain with initial state $z^{(0)} \in \Omega$ is determined by the mapping $F^0_{-M}\left(z^{(0)}, u^{(0)}, \ldots, u^{(-M+1)}\right)$, where F^0_t is defined recursively by,

$$\begin{aligned} F^0_0(z) &= z \\ F^0_{-1}\left(z, u^{(0)}\right) &= \phi\left(z, u^{(0)}\right) \\ F^0_t\left(z, u^{(0)}, u^{(-1)}, \ldots, u^{(t+2)}, u^{(t+1)}\right) &= F^0_{t+1}\left(\phi\left(z, u^{(t+1)}\right), u^{(0)}, u^{(-1)}, \ldots, u^{(t+2)}\right). \end{aligned}$$

The point is that if $F^0_{t'}$ is a constant mapping for some $t' \in [-M, 0)$, for all $z, z' \in \Omega$, i.e.,

$$F^0_{t'}\left(z, u^{(0)}, u^{(-1)}, \ldots, u^{(t'+1)}\right) = F^0_{t'}\left(z', u^{(0)}, u^{(-1)}, \ldots, u^{(t'+1)}\right),$$

then holds for all $t \leq t'$. In this case we would have the identity $F^0_{-M} = F^0_{t'}$. Setting M equal to infinity, the output values of $F^0_{-\infty}$ is distributed exactly according to the wanted stationary distribution $\mathbf{P}$. This outputted value equals the value of F^0_t for all t's where F^0_t is a constant map, and thus it will be sufficient to dynamically determine the value of t where F^0_t is a constant map. Coming back to the case where $|\Omega|$ Markov chains is run, a constant map F^0_t is equivalent with the coalescence of all chains with different initial states. However, the value of the mapping F^0_t is not necessarily the same as the state of the chains at the time of coalescence, i.e., the chains may have achieved coalescence before time 0. This explains why we need to perform "backwards" simulation to obtain unbiased samples. If we were doing forward simulation from time 0, the time t at which the map F^t_0 became constant would coincide with the time of coalescence. Moreover, the value of F^t_0 is not necessarily the same as the value of F^∞_0 , where the latter is the unbiased sample from $\mathbf{P}$. Practically we would attempt to achieve coalescence iteratively by running the chains at from starting times $-M, -2M, \ldots$ until coalescence is achieved using the same random numbers.

Some Markov chains exhibit monotonicity which depend on a predefined partial ordering of Ω. The state space Ω is said to be partially ordered, if the configurations in Ω is ordered such that $z^\circ z'$ when $z(s_i)^\circ z'(s_i)$ for all $i \in V$. If K is the transition kernel of a Markov chain on a partially ordered state space, then K is monotone if K preserves the partial order, i.e., $K(x, \cdot)^\circ K(x', \cdot)$ for $x^\circ x'$. For the Ising model where the state space is $\Omega = \{1, -1\}^V$, we can make use of the monotonicty of the transition kernel of the Gibbs sampling in order to run two instead of $|\Omega|$ Markov chains. Consider single-site local characteristics that are used for

Gibbs sampling. For the Ising model, we have:

$$\begin{aligned}\mathbf{P}\left(z\left(s_i\right)=1 \mid \left\{z\left(s_j\right): j \neq i\right\}\right) &= \frac{\exp\left(\sum\limits_{j:(i,j)\in E} \theta 1_{[z(s_j)=1]}\right)}{\exp\left(\sum\limits_{j:(i,j)\in E} \theta 1_{[z(s_j)=1]}\right)+\exp\left(\sum\limits_{j:(i,j)\in E} \theta 1_{[z(s_j)=-1]}\right)} \\ &= \frac{1}{1+\exp\left(\sum\limits_{j:(i,j)\in E} -\theta z\left(s_j\right)\right)} \\ &= \frac{\exp\left(\sum\limits_{j:(i,j)\in E} \theta \frac{(z(s_j)+1)}{2}\right)}{\exp\left(\sum\limits_{j:(i,j)\in E} \theta \frac{(z(s_j)+1)}{2}\right)+\exp\left(\sum\limits_{j:(i,j)\in E} \theta \frac{(-z(s_j)+1)}{2}\right)}\end{aligned}$$

Thus $\mathbf{P}\left(z\left(s_i\right) \mid \left\{z\left(s_j\right): j \neq i\right\}\right)$ is an increasing function of $\left\{z\left(s_j\right): j \neq i\right\}$ in the partial order. In the CFTP context, we can run two Markov chains from the initial states 0^V and 1^V, and if these two chains achieve coalescence, then so will all the others.

The latter may benefit from the MapReduce framework simulation of Gibbs sampling, since Gibbs sampling is used as a subroutine. Moreover, its convergence-free runs fall into the type of algorithm desired for our non-MCMC-expert-friendly pipeline.

3 Higher-Order Markov Random Field

In this section, the parametrization of higher-order Markov Random fields based on the discussion in Tjelmeland and Besag (1998), is first discussed and shown to be able to represent macroscopic spatial scenes. After alluding to the difficulty of its simulation, we present a novel generalization of an alternate tractable model that matches the capabilities of the higher-order Markov Random fields, and the forward-backward algorithm for this model.

Higher-order Markov random field, i.e., one where the cliques are larger than two, are considered in this section. This has been shown to be favorable for constructing large-scale structures in a methodical fashion. Recall that in general the joint probability mass function can be written as:

$$\mathbf{P}\left(z\right)=\frac{\exp Q\left(z\right)}{\sum\limits_{y\in\Omega}\exp Q\left(y\right)}.$$

A clique c is a set of vertices that are fully connected and recall that z_c is the projection of z on c. Such a projection defines a clique configuration type which we will denote as $\bar{c}$. Note that another clique c' with its projection $z_{c'}$ can have the same clique configuration type $\bar{c}$. Let $\mathcal{C}$ be the set of cliques and let $\bar{\mathcal{C}} \equiv \bar{\mathcal{C}}\left(\mathcal{C},\Omega\right)$ be the set of clique types. The latter is also a function of Ω as it requires the notion of a realization $z \in \Omega$ to consider a clique configuration type $\bar{c}$. The negpotential function can then be written as:

$$Q\left(z\right)=\sum_{c\in\mathcal{C}} Q_c\left(z_c\right),$$

where $Q_c\left(z_c\right)$ is the potential function. To describe our parametrization, it is handy to write the potential function as follows:

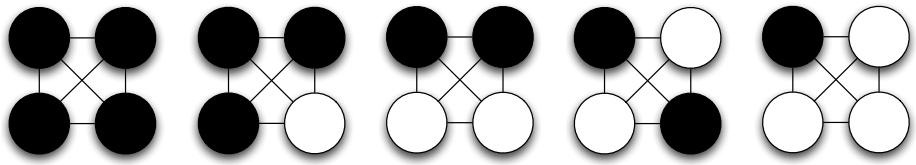
$$Q_c\left(z_c\right)=\sum_{\bar{c}\in\bar{\mathcal{C}}} Q_{\bar{c}}\left(z_c\right),$$

where:

$$Q_{\bar{c}}(z_c) = \begin{cases} \theta_{\bar{c}} & \text{if } z_c \text{ has the clique configuration type } \bar{c} \\ 0 & \text{otherwise.} \end{cases}$$

The parameter of the model is then the set of reals $\left\{\theta_{\bar{c}} : \bar{c} \in \bar{\mathcal{C}}\right\}$, where the larger the parameter value, the higher the probability of the corresponding clique configuration type occuring. As an example, we may consider a 2×2 binary clique configuration type, where a subset of all the possible clique configuration types is shown in 0. Here we have enumerated all the possible clique configuration types that contain a black vertex, up to rotation. This particular set is useful, for instance, for describing black objects on a white backdrop. In order to describe higher-order interactions that are more complex than what a simpler model, such as the Ising model, would, we can assign a nonzero real to a subset of the set of five parameters $\{\theta_{\bar{c}_1}, \theta_{\bar{c}_2}, \theta_{\bar{c}_3}, \theta_{\bar{c}_4}, \theta_{\bar{c}_5}\}$. In this case, $\theta_{\bar{c}_1}$ corresponds to 2-by-2 full black object, $\theta_{\bar{c}_2}$ to a corner turn of two black edges for all four possible corner turns, $\theta_{\bar{c}_3}$ to horizontal and vertical edges, $\theta_{\bar{c}_4}$ to all diagonal edges and $\theta_{\bar{c}_5}$ to singleton black vertices. Note that all these features, i.e., full object, corner turn, edges and singletons are relative to its foreground clique and thus is more refined than the pairwise Ising model. Intuitively, consider a 2 by 2 set of sites and let the goal be to describe an edge, i.e., two black vertices and two white vertices. Using higher-order interactions, we may may give a value to $\theta_{\bar{c}_3}$ and/or $\theta_{\bar{c}_4}$ higher than the rest, while with the pairwise Ising model, we may only control two sites at once and assigning a high value to the black edges and low value to the white edges in the pairwise model may render the whole 2 by 2 set of sites black.

Table 1: The 5 possible configurations containing black pixel in a 4-clique for a binary field (up to rotations).



For a example of a larger scale, we can consider cliques larger than 4. Values are then methodically assigned to the potentials associated with the clique configurations such that certain configurations are favored. Here we give an explicit example in order to simulate compact objects using cliques of size 9. It has been shown that an equivalent formulation on a hexagonal array is able to capture the convexity or concavity of compact shapes against a background. In this case, the reduced set of clique configuration types, up to translation and rotation of black against white, is enumerated in 0. We can then partition this set based on the spatial feature the elements represent. The clique configuration type corresponding to a 3-by-3 full black object is (a), and this is assigned a high value. The set of configurations with convex corner comprises of (d),(k),(o),(q) and this is assigned a relatively high value. The set of lines comprises of (g),(h),(n),(p). and are assigned a low value. The set of concave corner comprises of (c),(e),(f), and this set is given a low value in order to capture more convexity in the simulated images. Other configurations, including (i), (j), (l),(m),(r), are not useful for representing convex objects and are given low values. To calibrate the density of the objects, the configuration of all foreground pixels (b) is to be tuned.

We show some realizations from a similar model in a hexagonal array in 1. One artifact of these images is that the object borders have the same alignment with the axis, due to the parametrization. The typical difficulty is that the construction of various angles of the object border would require the concave corners such as (e) and (f) to be used, while still ensuring that the objects are convex. In order to allow for a larger variety of angles, and still maintain a large degree of convexity, a simulation from a model whose neighborhood is of a higher order can be used. This has been shown to rid unwanted artifacts, such as the axial alignments. As an example, we can consider the set of clique type configurations 3, where we have different angles of the object borders. Realizations from a model with a fourth-order neigborhood is shown in 2, and the appearance of the artifacts has been reduced, albeit with a more complicated parametrization.

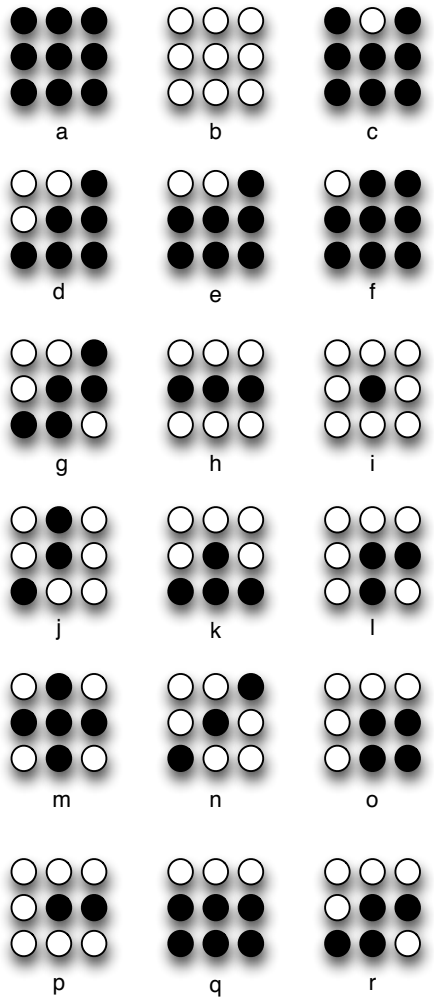


Figure 1: Clique configurations on 3×3 spatial sites useful for constructing larger structures (up to rotations). This appears in the second-order neighborhood simulation.

For a different flavor of model, we can consider a labyrinths of black lines against white, where the parametrization is as follows: the set of configurations corresponding to lines comprises of (h) and (n), those corresponding to turns comprises of (l), (j), (g), and (r), those corresponding to crosses comprises of (m), and (k), and lastly, a dead end can be represented by (p). This model is especially attractive for constructing channels in geostatistics, which can be done by increasing the width of of the lines. For instance, the set of configurations corresponding to lines would instead comprise of (q). It is thus clear that a larger clique type is necessary for this. For a complete parametrization, refer to Tjelmeland and Besag (1998). Ad hoc models can also be constructed, based on the problem at hand. For example, a two dimensional model for shale barriers and a three dimensional mosaic model with a similar parametrization is constructed in Tjelmeland (1997).

Conditional simulation is done by fixing some of the values a priori to the simulation. This conditional Markov random field notion is heavily studied in machine learning under the term Conditional random field (Lafferty et al., 2001). In all our simulation algorithms, including the MCMC and forward-backward, conditional simulation amounts to having the update function not change the fixed values (Tjelmeland and Besag, 1998).

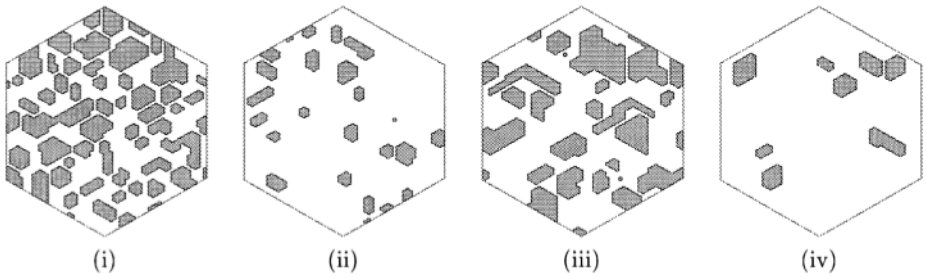


Figure 2: Example of a higher-order model simulation with a second-order neighborhood. Source: [49].

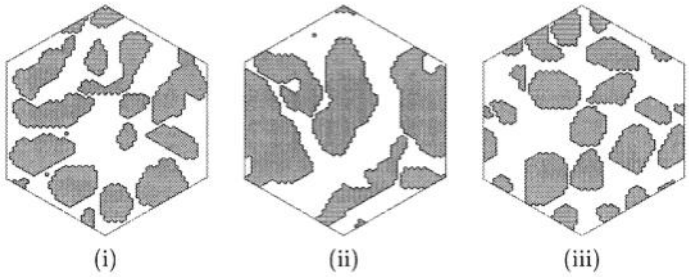


Figure 3: Example of a higher-order model simulation with a fourth-order neighborhood. Source: [49].

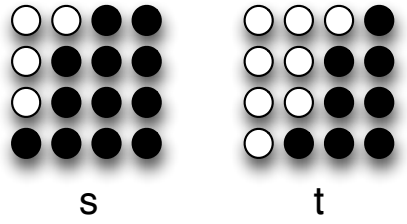


Figure 4: Two fourth-order clique type configurations.

4 Generalized Factorizable Model and Forward-Backward Algorithm

In order to pave our way to the probabilistic model, we first discuss a general framework of probabilistic model which allows us to define the forward-backward algorithm, pertinent to various statistical computation tasks which we will also discuss.

Given the negpotential function $Q(z)$, let $q(z) = \exp Q(z)$, i.e., the unnormalized probability distribution. Suppose that the sites of z can be ordered such that $q(z)$ can be factorized in the following way:

$$q(z) = \prod_{i=1}^{n} q_i \left(z \left(s_{L_i} \right) \right),$$

where the set L_i is such that $L_i \subseteq \{i, \ldots, n\}$ with $i \in L_i$ and $L_i \backslash \{i\} \subseteq L_{i+1}$. Let $r = \max_{i \in n} |L_i|$, the probability mass function is then termed a generalized factorizable lag-r model (Reeves and Pettitt, 2004; and Friel and Rue, 2007). Thus independent $z(s_i)$ for all $i \in V$ corresponds to a lag-0 model.

The forward-backward algorithm (Baum et al., 1970), allows us to compute Z at the forward step and compute the likelihood and sample exactly from $\mathbf{P}$ at the backward step. We discuss the algorithm in the

setting of a generalized factorizable model. The normalizing constant Z can be computed as follows:

$$\begin{aligned} Z &= \sum_{z} q\left(z\right) \\ &= \sum_{x_n} \left(q_n\left(z\left(s_{L_n}\right)\right) \cdots \sum_{x_i} \left(q_i\left(z\left(s_{L_i}\right)\right) \cdots \sum_{x_1} q_1\left(z\left(s_{L_1}\right)\right) \right) \right) \end{aligned}$$

Recursively, letting $z\left(s_{i:j}\right) \equiv \left(z\left(s_i\right), \ldots, z\left(s_j\right)\right)$, we may define the following w-functions:

$$\begin{aligned} w_1\left(z\left(s_{L_1 \setminus \{1\}}\right)\right) &= \sum_{z(s_1)} q_1\left(z\left(s_{L_1}\right)\right), \\ w_i\left(z\left(s_{L_i \setminus \{i\}}\right)\right) &= \sum_{z(s_i)} \left(q_i\left(z\left(s_{L_i}\right)\right) w_i\left(z\left(s_{L_{i-1} \setminus \{i-1\}}\right)\right)\right) \quad (i = 2, \ldots k) \\ Z &= \sum_{z(s_{k+1:n})} w_k\left(z\left(s_{L_k \setminus \{k\}}\right)\right), \end{aligned}$$

where k is such that $\{1, \ldots, n\}$ is covered by $\{L_i : i = 1, \ldots, k\}$. Let $\mathrm{X} \equiv \Omega_i$ for all $i \in V$. Then this calculation requires $\sum_{i=1}^{n} |\mathrm{X}|^{L_i - 1}$ evaluations of the q-functions. This phase of the algorithm, which yields Z, can be considered as the forward part of the algorithm.

At the end of the forward part, we immediately obtain the following probabilities:

$$\begin{aligned} \mathbf{P}\left(z\left(s_{k+1:n}\right)\right) &= \frac{w_k\left(z\left(s_{L_k \setminus \{k\}}\right)\right)}{Z} \\ \mathbf{P}\left(z\left(s_i\right) \mid z\left(s_{L_i \setminus \{i\}}\right)\right) &= \frac{q_i\left(z\left(s_{L_i}\right)\right) w_{i-1}\left(z\left(s_{L_{i-1} \setminus \{i-1\}}\right)\right)}{w_i\left(z\left(s_{L_i \setminus \{i\}}\right)\right)}, \quad i = k, k-1, \ldots, 2 \\ \mathbf{P}\left(z\left(s_1\right) \mid z\left(s_{L_1 \setminus \{1\}}\right)\right) &= \frac{q_1\left(z\left(s_{L_1}\right)\right)}{w_1\left(z\left(s_{L_1 \setminus \{1\}}\right)\right)} \end{aligned}$$

from which we can obtain the likelihood of the parameter given the realization z:

$$\mathbf{P}\left(z\right) = \mathbf{P}\left(z\left(s_n\right)\right) \prod_{i=1}^{n-1} \mathbf{P}\left(z\left(s_i\right) \mid z\left(s_{i+1:n}\right)\right).$$

The backward part of the algorithm can be used to sample from $\mathbf{P}$; first sample from $\mathbf{P}\left(z\left(s_{k+1:n}\right)\right)$, then $\mathbf{P}\left(z\left(s_i\right) \mid z\left(s_{L_i \setminus \{i\}}\right)\right)$ for $i = k, \ldots, 1$. This sample requires drawing n times from distributions which have already been computed in the forward part. Define a stripe of thickness m and length n to be a graph with vertices representing a set of spatial sites laid out on a grid of size $m \times n$. For example, the graphs in 3 correspond to a stripe of thickness 1 and length n and a stripe of thickness 1 and length 6, respectively. This notion of stripes thus represents how V is laid out spatially in two dimensions.

Viewing Markov random fields on these stripes as a generalized factorizable model, we can systematically consider the lag-r of such models on stripes of thickness 1 as the spatial distance between two vertices connected by an edge. In 3, the largest of such distances is 1 and 4, respectively, corresponding to the lag-r and thus to the computational cost of the forward step. For these graphs, the sizes of L_i's are uniform, and we may take $L_i = \{i, \ldots, i + r\}$ for $i = 1, \ldots, k$. The negpotential function is as follows:

$$\begin{aligned} Q\left(z\right) = \exp\Bigg(&\sum_{i=1}^{n} V_i\left(z\left(s_i\right)\right) + \sum_{i<j,(i,j)\in E} V_{i,j}\left(z\left(s_i\right), z\left(s_j\right)\right) + \\ &\sum_{i<j<k,(i,k)\in E} V_{i,j,k}\left(z\left(s_i\right), z\left(s_j\right), z\left(s_k\right)\right) + \sum_{i<\ldots<k,(i,\ldots,k)\in\mathcal{C}} V_{i,\ldots,k}\left(z\left(s_i\right), \ldots, z\left(s_k\right)\right) \Bigg) \end{aligned}$$

where we have enumerated all the possible cliques. A more parsimonious representation would be to only consider the larger cliques that contain the configurations governed by the smaller cliques. One approximation in the manner that assigns some of the parameters in the V functions to zero is shown in Austad (2012), and is a very promising avenue of work. In this case, the algorithm is as follows, write:

$$q_i\left(z\left(s_{L_i}\right)\right) = \exp\left(V_i\left(z\left(s_i\right)\right) + \sum_{i<j,(i,j)\in E} V_{i,j}\left(z\left(s_i\right), z\left(s_j\right)\right) + \ldots + V_{i,\ldots,i+r}\left(z\left(s_i\right), \ldots, z\left(s_{i+r}\right)\right)\right)$$

for $i = 1, \ldots, k$. This knowledge is sufficient to start the forward recursion for computing the w- functions, and thus Z.

In the example where $r = 4$, we can solely consider the cliques of size 5 with the following negpotential function:

$$Q\left(z\right) = \exp\left(V_5\left(z\left(s_{1:5}\right)\right) + V_5\left(z\left(s_{2:6}\right)\right)\right),$$

where the computation of the w-functions in the forward recursion would be cheaper.

Stripes of thickness 1 are useful for describing the algorithm as there is an implicit one-dimensional ordering in them. Thus may be useful for one dimensional data such as that from time-series analysis. However, these stripes may not be sufficient for representing a spatial scene. Examples of stripes of thickness larger than 1 are shown in 4 and 5. In the former, also described in Friel and Rue (2007), thickness is equal to 3. By the ordering given in the image, we have:

$$q_1\left(z\left(s_{L_1}\right)\right) = \exp\left(\sum_{i=1}^{4} V_i\left(z\left(s_i\right)\right) + V_{1,2}\left(z\left(s_1\right), z\left(s_2\right)\right) + V_{2,3}\left(z\left(s_2\right), z\left(s_3\right)\right) + V_{1,4}\left(z\left(s_1\right), z\left(s_4\right)\right)\right),$$

corresponding to all of the leftmost column and v_4. For $i = 2, n-3$,

$$q_i\left(z\left(s_{L_i}\right)\right) = \exp\left(V_{i+3}\left(z\left(s_{i+3}\right)\right) + V_{i+2,i+3}\left(z\left(s_{i+2}\right), z\left(s_{i+3}\right)\right) + V_{i,i+3}\left(z\left(s_i\right), z\left(s_{i+3}\right)\right)\right),$$

corresponding to the rest of the vertices except for the top row vertices. For these, we write:

$$q_i\left(z\left(s_{L_i}\right)\right) = \exp\left(V_{i+3}\left(z\left(s_{i+3}\right)\right) + V_{i,i+3}\left(z\left(s_i\right), z\left(s_{i+3}\right)\right)\right).$$

To engineer this approach it is desirable to describe an automated grouping algorithm for the cliques into the set $\{L_i : i \in V\}$. It is clear that the lag is directly related to the size of largest clique of the graph. Given a lexicographical indexing of the vertices on the regular grid, we traverse the graph by the index and at i, adding the vertices larger than i greedily into L_i's while the vertices are connected to i, doing so until a valid k is achieved. The stripes with thickness larger than 1 can be generalized as having a different neighborhood. In example 5, thickness is equal to m and lag is $m + 1$.

The parameters that determine the computational cost of the forward step are thus the thickness and complexity of the neighborhood. A more complex neighborhood, which thus allows for more complex clique configurations, would require larger computational cost.

We can perform other statistical tasks using a similar paradigm. To compute the modal configuration of $\mathbf{P}\left(z\right)$, we define the following forward recursion with w^f-functions:

$$\begin{aligned} w_2^f\left(z\left(s_{L_1\setminus\{1\}}\right)\right) &= \max_{z(s_1)} q_1\left(z\left(s_{L_1}\right)\right), \\ w_i^f\left(z\left(s_{L_{i-1}\setminus\{i-1\}}\right)\right) &= \max_{z(s_{i-1})}\left(w_{i-1}^f\left(z\left(s_{L_{i-2}\setminus\{i-2\}}\right)\right) q_{i-1}\left(z\left(s_{L_{i-1}}\right)\right)\right) \quad (i = 3, \ldots k+1) \end{aligned}$$

These functions can be used to give the probability of the modal configuration, denoted by $\hat{z}$. The probability can be written as:

$$\mathbf{P}\left(\hat{z}\right) = \frac{\max\limits_{z\left(s_{L_k\setminus\{k\}}\right)} w_{k+1}^f\left(z\left(s_{L_k\setminus\{k\}}\right)\right)}{Z}.$$

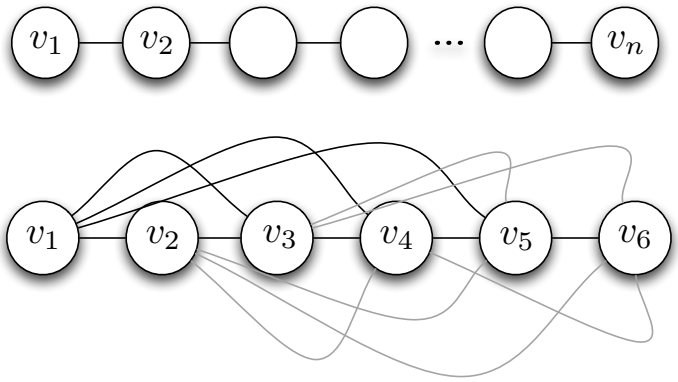


Figure 5: Stripes of thickness 1.

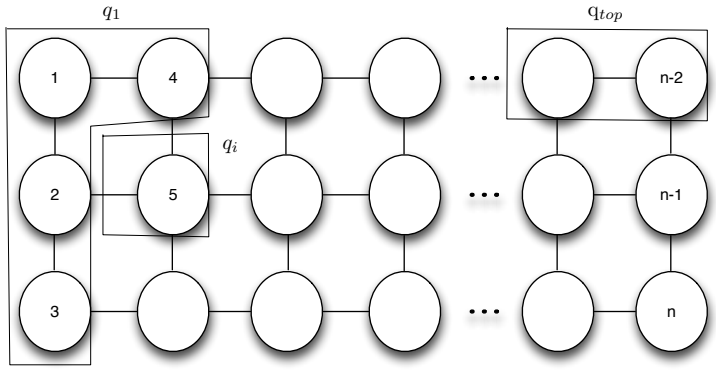


Figure 6: Stripe of thickness 3 and lag 3.

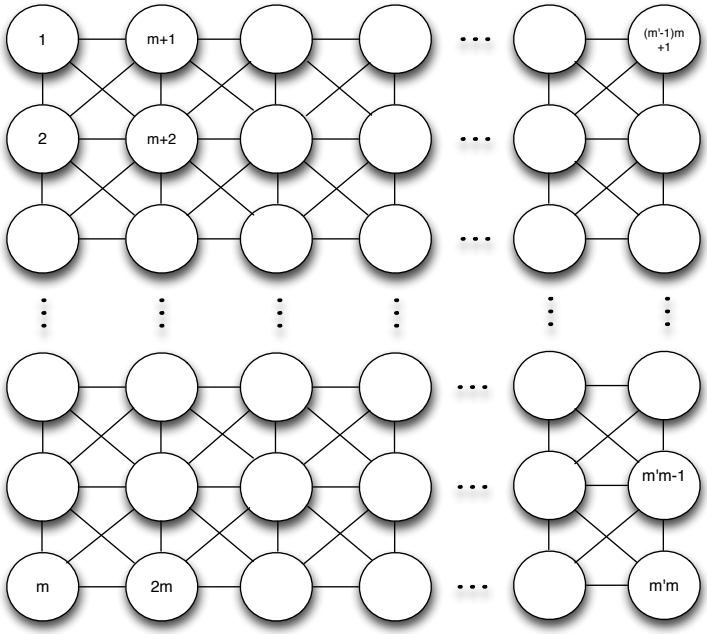


Figure 7: Stripe of thickness m and lag $m+1$, $m < m'$.

Similarly, computing the modal configuration uses the backwards step:

$$\begin{aligned}\hat{z}\left(s_{L_k\setminus\{k\}}\right) &= \arg\max_{z\left(s_{L_k\setminus\{k\}}\right)} w_{k+1}\left(z\left(s_{L_k\setminus\{k\}}\right)\right),\\ \hat{z}\left(s_i\right) &= \arg\max_{z(s_i)} w_i\left(z\left(s_i\right),\hat{z}\left(s_{L_{i-1}\setminus\{i-1\}}\right)\right) \quad (i=k-1,\ldots,2)\\ \hat{z}\left(s_1\right) &= \arg\max_{z(s_1)} q_1\left(z\left(s_1\right),\hat{z}\left(s_{L_1\setminus\{1\}}\right)\right).\end{aligned}$$

Computationally, a unique global maximum might not exist and when this happens, there exists a w^f function with multiple outputs, in that case all the solutions must be cached in the forward step, and during the backward step, the algorithm can sample uniformly at random from the set of solutions.

Full marginal distributions can also be computed; to compute any joint distribution $\mathbf{P}\left(z\left(s_{L_i\setminus\{i\}}\right)\right)$ using the stored w-functions, write:

$$\begin{aligned}\mathbf{P}\left(z\left(s_{i:n}\right)\right) &= Z^{-1}\sum_{z\left(s_{1:i-1}\right)} q\left(z\right)\\ &= Z^{-1}w_{i-1}\left(z\left(s_{L_{i-1}\setminus\{i-1\}}\right)\right)\prod_{j=i}^{k} q_j\left(z\left(s_{L_j}\right)\right).\end{aligned}$$

Moreover, we can compute the following conditional probabilites immediately:

$$\mathbf{P}\left(z\left(s_{1:i-1}\right)\mid z\left(s_{i:i+r}\right)\right) = \frac{\mathbf{P}\left(z\right)}{\mathbf{P}\left(z\left(s_{i:n}\right)\right)}.$$

In order to compute the marginals $\mathbf{P}\left(z\left(s_{1:i}\right)\right)$ and conditional probabilities $\mathbf{P}\left(z\left(s_{1:i-1}\right)\mid z\left(s_{i:i+r}\right)\right)$, we can run the forward step of the algorithm in a decreasing index.

The computation of the Maximum likelihood estimator (MLE) involves the computation of the normalizing constant, and thus the forward step of the algorithm can be used. Pseudolikelihood was proposed in Besag (1974), in order to avoid computing the normalizing constant. In a nutshell, the expression of the joint-likelihood is replaced by the expression of pseudolikelihood, and then maximization is performed to obtain the maximum pseudolikelihood estimator (MPLE). The pseudolikelihood of the parameter β is:

$$L\left(\beta;z\right) = \prod_{i=1}^{n} P\left(z\left(s_i\right)\mid z\left(s_{N_i}\right)\right).$$

To obtain the probabilities in the product, the unnormalized conditional distribution can be obtained immediately from the joint probability mass function:

$$\mathbf{P}\left(z\left(s_i\right)\mid z\left(s_{N_i}\right)\right) \propto \exp\left(\sum_{C:i\in C} V_C\left(x_C\right)\right),$$

and the computation of normalizing constant is required to normalize the RHS. This computation is still tractable as we are only considering |X| computations. MPLE, although consistent (Gidas, 1987), has been shown to be unsatisfactory when used to estimate the parameters of a model with strong interactions, as per the higher-order Markov random field. A remedy is to use MCMC (Geyer, 1994), but we consider a natural approach that lies in between the computation of MLE and MPLE.

In the generalized pseudo-likelihood (Huang and Ogata, 2002) approach, we consider blocks of vertices instead of singletons. The generalized pseudolikelihood of a parameter β is:

$$L\left(\beta;z\right) = \prod_{i=1}^{n}\mathbf{P}\left(z\left(s_{G_i}\right)\mid z\left(s_{N_{G_i}}\right)\right),$$

where G_i is a set of vertices containing i. Thus if $G_i = \{i\}$, we get Besag's pseudolikelihood and if $G_i = V$, we obtain the likelihood. The computation of normalizing constant depends on how G_i is defined, and thus can become computationally heavy as G_i becomes more complex. The forward-backward algorithm can be used here.

Similarly, the algorithm can be used for normalizing the full conditionals in Gibbs sampling such if the blocks used are large enough. This algorithm can be used as a subroutine for performing the statistical tasks of the model that we are going to describe next.

5 Multidimensional Markov Chain Model

The Multidimensional Markov Chain (MDMC) is discussed in Qian and Titterington (1991), as a model for textures. is a Markov chain where each element of the chain is also a Markov chain; indeed, we can use any random field from above as an element of the chain and define the Markov chain property on these nested models.

We define MDMC by considering a set of stripes $GS = \{GS_i : i = 1, \ldots, l\}$ between which we impose dependence. Consider a finite set of spatial sets on a grid of size $m \times n$. We then cover the grid with a set of stripes of size l, where the thickness of GS_i is m_i and number of vertices $m_i \times n$, such that $\sum_{i=1}^{l} m_i = m$. As an example, consider 6. In this discussion we index z by the set of coordinates $\{(j,k) : j \in m, k \in n\}$.

Probabilistic dependencies between the stripes are then defined directionally. For simplicity, we consider the vertices to be on a plane, i.e., a two-dimensional model. Three or four dimensional models, which are possible in spatial analysis, can be generalized from the following by considering more directions in the definition of the inter-dependencies of the G's.

Recall a $t-th$ order Markov chain, defined as:

$$\mathbf{P}\left(x_k \mid x_1, x_2, \ldots, x_{k-1}\right) = \mathbf{P}\left(x_k \mid x_{k-t}, x_{k-t+1}, \ldots, x_{k-1}\right),$$

Using this, we can define a sequential MDMC model $\mathbf{P}_{GS}$ for GS:

$$\mathbf{P}_{GS}\left(z\right) = \mathbf{P}\left(z_{GS_1}\right)\mathbf{P}\left(z_{GS_2} \mid z_{GS_1}\right)\cdots\mathbf{P}\left(z_{GS_t} \mid z_{GS_1}, \ldots, z_{GS_{t-1}}\right) \times \prod_{h=t+1}^{l} \mathbf{P}\left(z_{GS_h} \mid z_{GS_{h-t}}, \ldots, z_{GS_{h-1}}\right),$$

where each probability mass function is an MRF defined conditionally on a subset of the stripes. In Qian and Titterington, 1991, stripes of thickness 1 and pairwise dependencies across stripes are defined. To wit, this model with $t = 1$ and lag-1 model in each stripe can be be written as:

$$\mathbf{P}\left(z_{GS_1}\right) = Z_1^{-1}\exp\left(\sum_{i=1}^{n} V_{1,i}\left(z\left(s_{1,i}\right)\right) + \sum_{i=1}^{n-1} V_{(1,i),(1,i+1)}\left(z\left(s_{(1,i)}\right), z\left(s_{(1,i+1)}\right)\right)\right),$$

and for $h = 2, \ldots, l$,

$$\begin{aligned}\mathbf{P}\left(z_{GS_h} | z_{GS_1}, \ldots, z_{GS_{h-1}}\right) &= \mathbf{P}\left(z_{GS_h} | z_{GS_{h-r}}, \ldots, z_{GS_{h-1}}\right) \\ &= Z_h^{-1}\exp\left(\sum_{i=1}^{n} \bar{V}_{h,i}\left(z\left(s_{h,i}\right)\right) + \sum_{i=1}^{n-1} V_{(h,i),(h,i+1)}\left(z\left(s_{h,i}\right), z\left(s_{h,i+1}\right)\right)\right),\end{aligned}$$

where:

$$\bar{V}_{h,i}\left(z\left(s_{h,i}\right)\right) = \begin{cases} V_{h,i}\left(z\left(s_{h,i}\right)\right) + V_{h,i}^{(0)}\left(z\left(s_{h-1,i}\right), z\left(s_{h,i}\right)\right) + \\ V_{h,i}^{(1)}\left(z\left(s_{h-1,i+1}\right), z\left(s_{h,i}\right)\right) & i = 1 \\ V_{h,i}\left(z\left(s_{h,i}\right)\right) + V_{h,i}^{(-1)}\left(z\left(s_{h-1,i-1}\right), z\left(s_{h,i}\right)\right) + \\ V_{h,i}^{(0)}\left(z\left(s_{h-1,i}\right), z\left(s_{h,i}\right)\right) + V_{h,i}^{(1)}\left(z\left(s_{h-1,i+1}\right), z\left(s_{h,i}\right)\right) & 2 \le i \le n-1 \\ V_{h,i}\left(z\left(s_{h,i}\right)\right) + V_{h,i}^{(-1)}\left(z\left(s_{h-1,i-1}\right) z\left(s_{h,i}\right)\right) + \\ V_{h,i}^{(0)}\left(z\left(s_{h-1,i}\right), z\left(s_{h,i}\right)\right) & j = n. \end{cases}$$

The overall dependency graph then induces the neighborhood 7, where thethick edges denote the dependency induced by the MRFs in the stripes, and the thin edges denote the dependency induced by the probabilistic model on the ordered set of MRFs. This neighborhood has an equivalent dependency graph induced by an MRF, although the sole MRF is more computationally heavy.

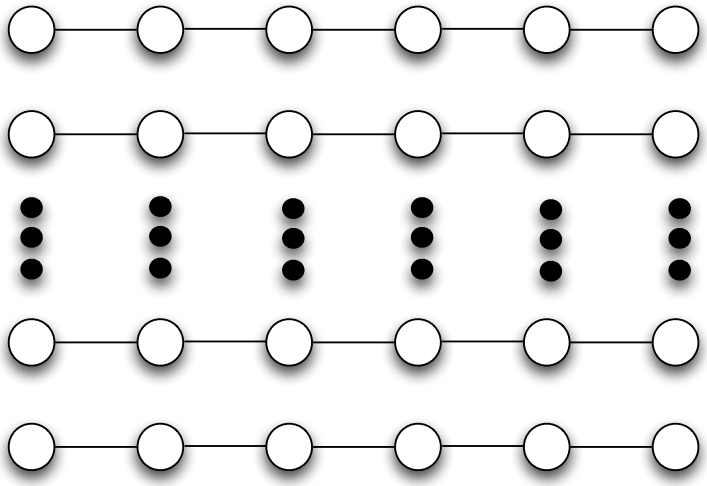


Figure 8: A set of with stripes of thickness 1 with lag 1 covering a grid.

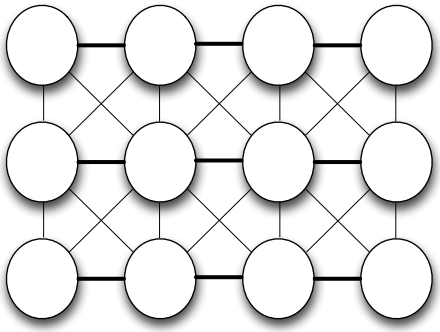


Figure 9: Induced neighborhood of a 1st order MDMC with stripes of thickness 1 with lag 1.

We can thus map an MDMC model to an MRF model. Note that by reversing the index of the stripes the same dependency graph is obtained, but $\mathbf{P}_{GS}$ is different. The dependencies of the MRFs on the stripes can be represented by the directed model 8. Here the MRF, defined by $\mathbf{P}(GS_1)$, is the sole top parent.

Sampling of this model can be done sequentially by first sampling the stripe GS_1 from $\mathbf{P}(z_{GS_1})$, and then GS_i for $i = 2, \dots n$ from $\mathbf{P}\left(z_{GS_i} \mid z_{GS_{i-r}}, \dots, z_{GS_{i-1}}\right)$.

We can define a different type of dependency on GS, without loss of generality for l odd:

$$\mathbf{P}_{GS}(z) = \prod_{t=1, t\ odd}^{l} \mathbf{P}\left(z\left(s_{GS_t}\right)\right) \prod_{t=1, t\ even}^{l} \mathbf{P}\left(z\left(s_{GS_t}\right) \mid z\left(s_{GS_{t-1}}\right), z\left(s_{GS_{t+1}}\right)\right).$$

The directed model for the MRFs in the stripes is depicted in 9. We can partition the stripes into two levels: the odd indices, which do not depend on anything, and the even indices, which depend on their surrounding odd indices of distance 1. We say that such a model has more than one top parent, corresponding to the odd indexed stripes.

This model is amenable to parallel computation. It is thus desirable to deduce the right MDMC model for the task at hand, as computation is cheaper compared to that of MRF. The higher-order MRF model has

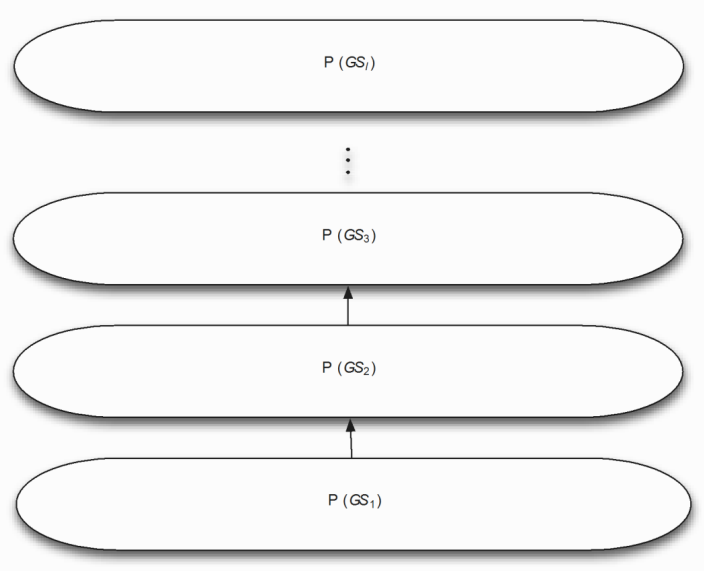


Figure 10: Directed model with one top parent.

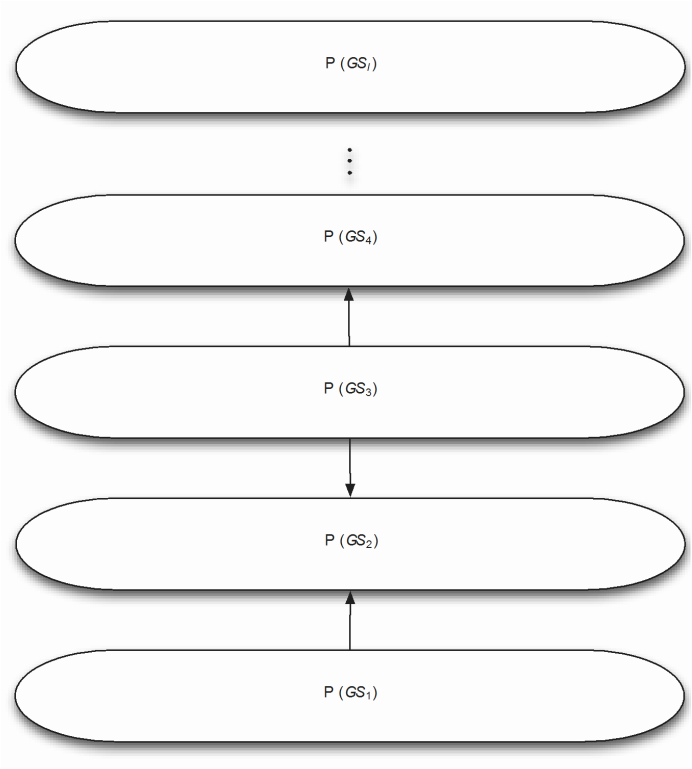


Figure 11: Directed model with more than one top parent.

been shown to be potent in simulating typical spatial scenes. We can then define an MDMC model whose dependency graph is equivalent that one from the higher-order MRF model.

Consider the higher-order MRF model defined on 3×3 cliques with neighborhood 10; we define an MDMC model whose dependency graph is equivalent to that of the higher-order MRF. Without loss of generality, let the set of stripes $GS = \{GS_i : i = 1, \ldots, l\}$, where each stripe has thickness 2. In this case, some of the dependencies must then be represented between the stripes, i.e., 11. We can then carry over the

parametrization of the higher-order MRF to the cliques of this model since there is a bijection between them. The realizations are very similar, but there is some directional effect that disappears as the thickness of the stripes is increased.

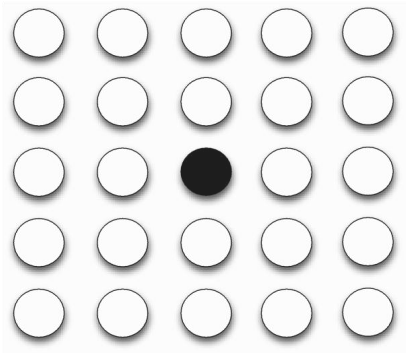


Figure 12: Neighborhood of the model with 3×3 clique.

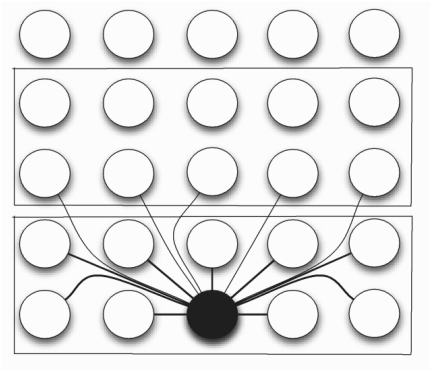


Figure 13: MDMC Model with higher-order dependency graph.

6 Engineering the Pipeline.

What we refer to as the pipeline is the sequence of algorithms which takes as input either an image or a set of object features and outputs the simulated probabilistic models of the input. The number of clique configurations for manual inspection is unwieldy, yet there is a need to sift through them in order to identify the size and type of clique configurations that are suitable for the task at hand. A domain specific language (DSL) in Mernik et al. (2005), is thus desired; as a query example, the practitioner inputs the object characteristics and outputs not only the configurations that directly match the object, but also the combinations of configurations that would achieve the same object. This would avoid assigning value of the same order to conflicting configurations.

Due to the inherent mapping of the higher-order MRF parametrization, automated computation of the set of MDMC models with the same dependency graph as the MRF is readily available. Forward-backward algorithm can be used to simulate from the set of models, and the practitioner can evaluate these simulation results readily in order to decide the thickness and orientation of the stripes in the MDMC model.

Originated by Dean and Ghemawat (2008), The MapReduce framework has emerged as one of the most widely used parallel computing platforms for processing data on large scales. This framework is heavily used in companies that are brimming with data, such as Google, Amazon, and Facebook, and has been adopted by more recently due to its open source implementation, Hadoop. The framework allows for easy parallelization, and it allows one to interleave sequential and parallel computations within the same pipeline.

We present the programming paradigm and cast our algorithms in it. The elementary unit of the paradigm is a $\langle key, value \rangle$ pair, where each key and each value are of any type, even composite ones. The input to any MapReduce algorithm is a set of $\langle key, value \rangle$ pairs. One MapReduce algorithm consists of the map stage, the shuffle stage and the reduce stage, which we describe now.

During the map stage, the mapper M takes as input a single $\langle key, value \rangle$ pair and produces any number of new $\langle key, value \rangle$ pairs. The point is that the mapper must be stateless, i.e., each pair of the data is processed only once by a mapper. This allows for easy parallelization as different inputs for the mapper can be processed by different machines.

In the shuffle stage, the system sends all of the values that are associated with an individual key to the same machine that will perform the reducer stage, seamlessly to the programmer. The reducer R takes all of the values associated with a single key k and performs sequential computations on these values. At this

stage, parallelization happens as different reducers, which have access to values from different keys, may run in different machines.

A major appeal of MapReduce is its ease of use. The framework protects the programmer from the low-level details of parallel programming, such as data replication, fault tolerance, machine scheduling, etc. The shuffling and aggregation of the pairs are handled by the underlying system; the programmer only needs to specify the map and reduce functions and the number of mappers and reducers whose sole jobs are to run their respective functions. It is not a priori clear which problems cannot be efficiently solved by this framework. For a work on the theoretical aspects of MapReduce, see Karloff et al (2010). We begin with an example of computing the maximum and (weighted) average grade in a grid. Consider a grid of grades ordered as follows: $z = (z(s_1), \ldots, z(s_n))$, where s_i is a coordinate and $z(s_i)$ is the grade of the coordinate for all $i \in [n]$. Sequentially, the maximum can be computed in $\Theta(n)$ by going through the vector in this ordering, keeping an incumbent maximum value. With MapReduce, suppose that we have l mappers, $M_1, \ldots, M_L$, and one reducer, R. We give as input to the mappers the pair $\langle s_i, z(s_i)\rangle$ for all $i \in [n]$, and, without loss of generality, let the set of all pairs be partitioned equally to the mappers, i.e., each mapper receives $\frac{n}{L}$ pairs. Each mapper then outputs the pair $\langle c, s_l^{max}\rangle$ where c is a chosen constant which is merely a vehicle to drive all the pairs outputted by the mappers to one reducer, and s_l^{max} is the maximum grade amongst all the grades inputted to the mapper M_l. The reducer then performs a sequential scan over s_l^{max} for all $l \in [L]$ to obtain $\max_{l\in L} s_l^*$. This computation takes $\Theta\left(L + \frac{n}{L}\right)$. Ideally we want to choose L such that most computations occur in the mapper, and the sequential computation in the reducer is minimized.

In the case that the mappers do not get the same number of pairs, i.e., when $\frac{n}{L}$ is not an integer, the same set up can be used. This, however, is not the case with computing the average grade. Sequentially, the average grade can be computed by adding all the grades and dividing it by n, taking $\mathrm{O}(n)$. In MapReduce, however, we may feed the mapper the set of pairs $\langle s_i, z(s_i)\rangle$ for all $i \in [n]$ and each mapper then outputs the pair $\langle c, (|z_l|, s_l^{avg})\rangle$ where $|z_l|$ is the cardinality of the set of pairs fed into mapper M_l and s_l^{avg} is the average grade of grades inputted. The reducer then performs a sequential scan over the set of pairs $\{(|z_l|, s_l^{avg})\}_{l\in[L]}$ and computes $\sum_{l\in L} |z_l| s_l^{avg} / \sum_{l\in L} |z_l|$. This computation takes $\mathrm{O}\left(L + \frac{n}{L}\right)$. Similarly, for weighted average, we can view the weights of the sites as an extra value to the set of pairs fed into the mappers and thus the input to the mapper is the pair $\langle s_i, (z(s_i), weight(s_i))\rangle$. The MapReduce framework aids the forward-backward algorithm in a generic way; the CPU-bound computation part of the algorithm is the summation over all possible configurations of $\arg\max_i |L_i|$, which contains an exponential factor. Such a computation is embarassingly parallel and thus we can simply assign the evaluation of the functions to different mappers. Thus, for a stage of the forward step which takes $\mathrm{O}(|\mathrm{X}|^r)$, we can use $|\mathrm{X}|^c$ mappers, which reduces the power in computational cost. Although still exponential, this allows the practitioner to either increase the thickness of the stripes, or to apply a larger neighborhood in the MDMC model. We can apply the Map-Reduce framework in the Markov random field settings. Given the higher-order Markov random field, we can perform Gibbs sampling in parallel by sampling the vertices that are disjoint. This can be scheduled by graph coloring (Winkler, 2003). The chromatic number of the graph is the number of rounds we must perform in order to perform a total sweep of the vertices. In order to use the Map-Reduce framework, we have to partition the set of vertices into a set of stable sets S, readily available from the graph coloring: let the vertices of the same color form a stable set. Let the chromatic number be equal to X_c. We then need to perform X_c MapReduce executions, each of which is given a stable set. The input to the mapper is the pair $\langle s_i, z(s_i)\rangle$ for $i \in \mathrm{S}_j$ for a stable set $\mathrm{S}_j \in \mathrm{S}$. The mappers then perform an update on these vertices based on the Gibbs update and output $\langle s_i, \hat{z}(s_i)\rangle$, where $\hat{z}(s_i)$ is the value at s_i, after the Gibbs update. The reducer then simply updates the image z sequentially to obtain $\hat{z}$. After running this for all $j \in \{1, \ldots, \mathrm{X}_c\}$, we have completed one full sequential sweep of the graph. This MapReduce formulation maintains the ergodicity of the Gibbs chain. When the dependencies are high, single site updates do not mix rapidly, and thus there is a need to introduce blocking updates (Barbu and Zhu, 2005) for Gibbs sampling. The evaluation of the local conditionals can be done via forward-backward algorithm, and blocks of vertices that are conditionally independent given another set of vertices can then again be updated in parallel.

The MDMC model where there exists more than one top parent is also very amenable to parallel computation with the MapReduce framework. The nodes in the same level can be processed in parallel and, similar to the Gibbs sampling setting, forms a stable set.

7 Simulation Results.

Let $\omega(G)$ be the clique number of the graph, defined as the number of vertices contained in the largest clique. We construct some examples on a regular grid to represent a spatial setting. The graph in example 12 is such that each non-boundary vertex has four neighbors on the North, East, South and West and the clique number is 2; this is a structure which has been proven to be very interesting despite its simplicity. Example 13 verifies eight neighbors on the North, East, South, West, North-East, North-West, South-East and South-West, and yields a large clique, K_4, a complete graph on four vertices, which is spatially a 2×2 grid. Such a graph allows the inclusion of potentials that are larger than pairwise, i.e., higher-order interactions. Example 14 shows that we may impose directionality in the underlying graph by way of removing dependencies from a more complex model. Indeed, one can think of the latter as a the model that verifies eight neighbors, but with some of the interactions fixed to zero.

We use parallel Gibbs sampling to simulate these 12 and 14 in a 250×250 grid with a progression of parameters.

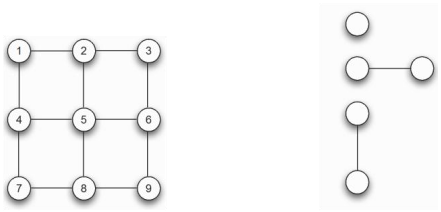


Figure 14: Graph G with $|E| = 10$ and the cliques up to translation, with $\omega(G) = 2$.

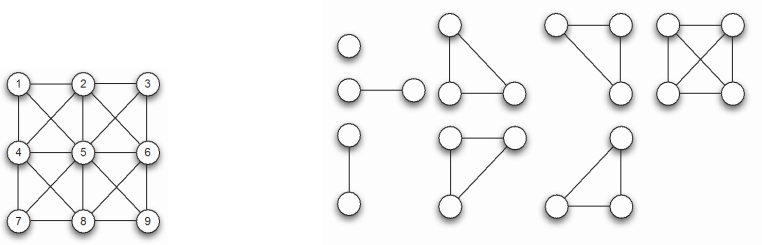


Figure 15: Graph H with $|E| = 18$ and the cliques up to translation, with $\omega(H) = 4$.

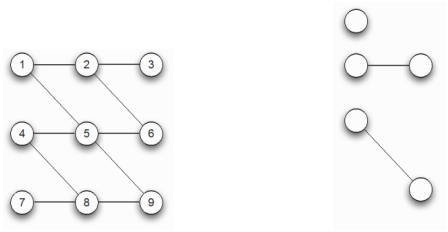


Figure 16: Graph I with $|E| = 10$ and the cliques up to translation, with $\omega(I) = 2$.

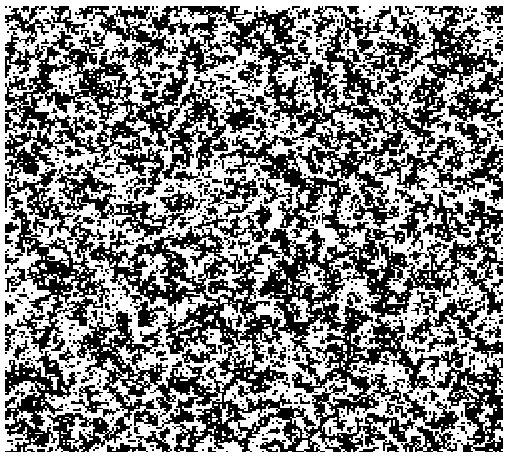


Figure 17: Simulation of G, with parameter value 0.6.

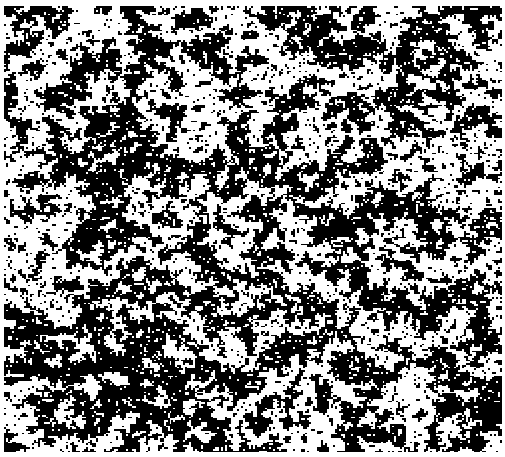


Figure 18: Simulation of G, with parameter value 0.8.

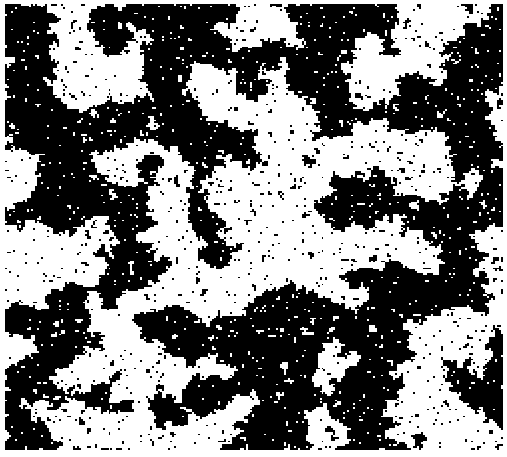


Figure 19: Simulation of G, with parameter value 1.

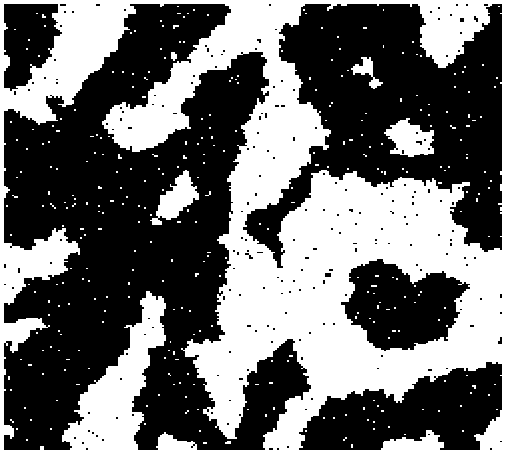


Figure 20: Simulation of G, with parameter value 1.2.

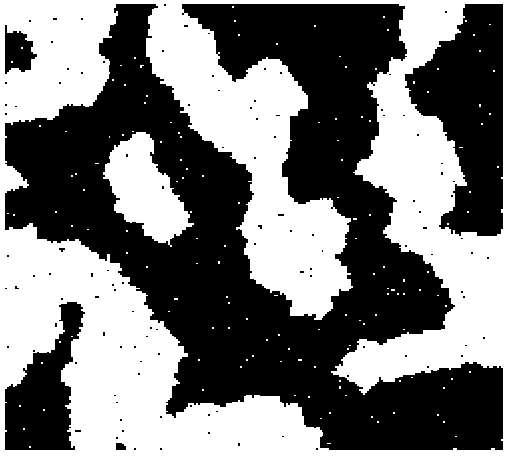


Figure 21: Simulation of G, with parameter value 1.4.

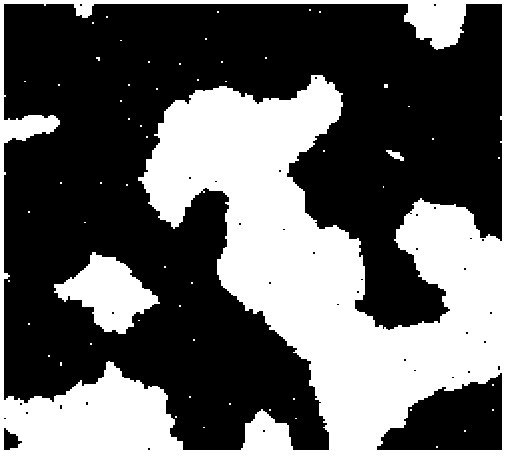


Figure 22: Simulation of G, with parameter value 1.6.

Figure 23: Simulation of G, with parameter value 1.8.

Figure 24: Simulation of G, with parameter value 2.

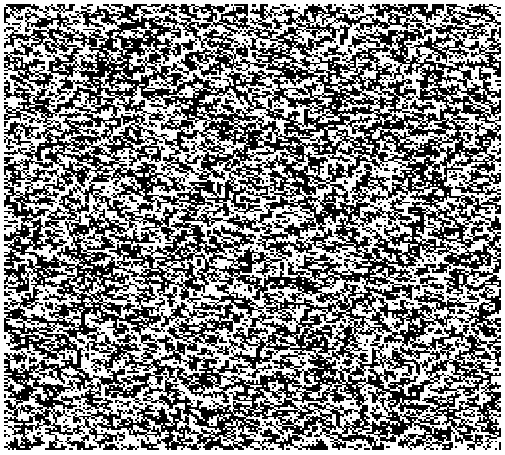


Figure 25: Simulation of I, with parameter value 0.6.

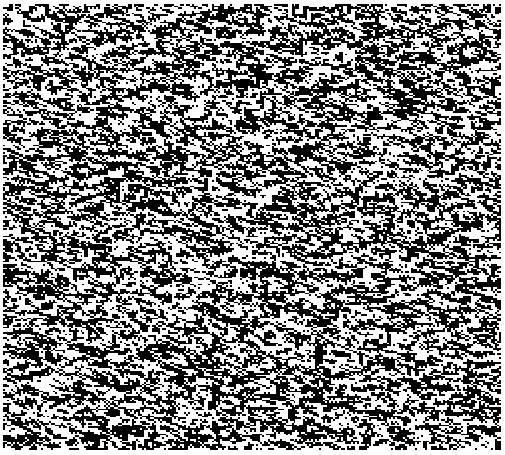


Figure 26: Simulation of I, with parameter value 0.8.

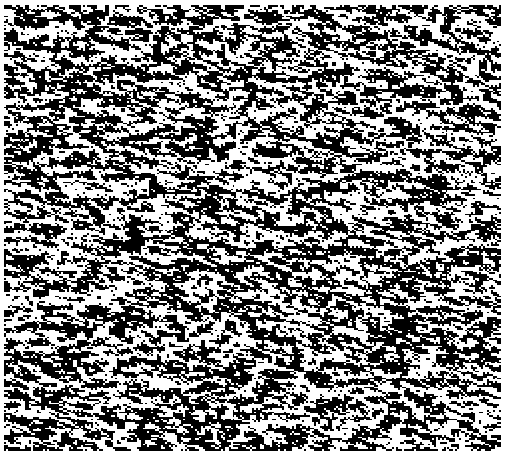


Figure 27: Simulation of I, with parameter value 1.

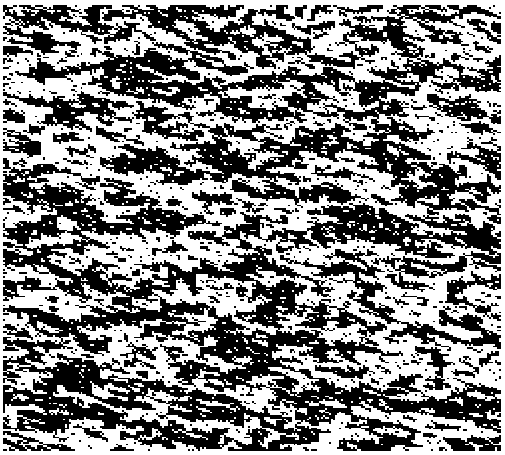


Figure 28: Simulation of I, with parameter value 1.2.

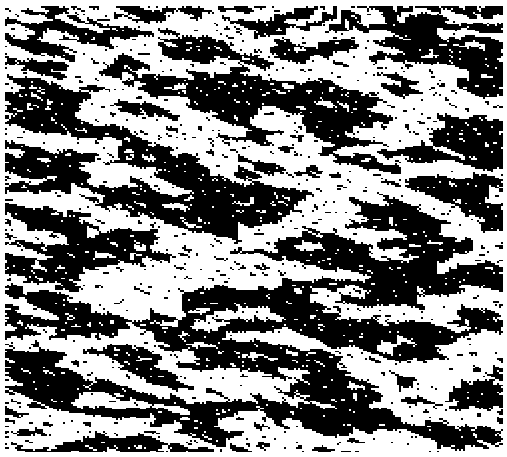


Figure 29: Simulation of I, with parameter value 1.4.

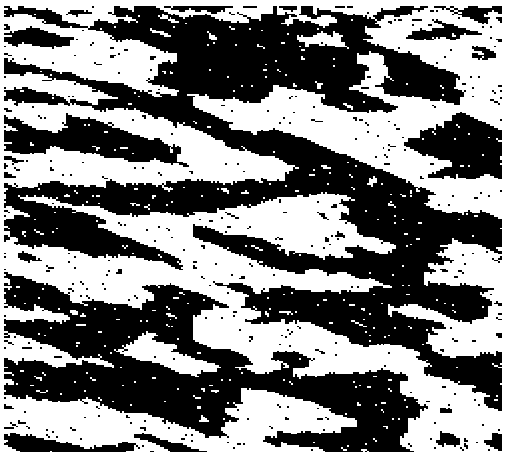


Figure 30: Simulation of I, with parameter value 1.6.

Figure 31: Simulation of I, with parameter value 1.8.

Figure 32: Simulation of I, with parameter value 2.

8 Conclusion

The forward-backward algorithm is the driving force behind the MDMC model. We propose the usage of stripes of various thickness in such a modelling attempt. Given a higher-order Markov random field, a set of MDMC models with similar dependency graphs can be constructed, and the MapReduce formulation allows us to consider thicker stripes and more complex MRF neighborhoods within the stripes.

The slow mixing of the MCMC for a higher-order Markov random field can be alleviated by blocked sampling, which may benefit from MapReduce in two ways. First, the forward-backward algorithm with MapReduce, which computes the normalizing constant, can compute larger blocks, and second, blocks that are conditionally independent from each other can be updated simultaneously.

We have attempted MDMC formulation, in which trees are used to cover the grid; this turned out to be unhelpful with respect to simulating macroscopic scenes. The stripes represent the higher-order dependencies more realistically than the trees.

The MDMC model, with the forward-backward algorithm, is easier for the non-experts to wield. Future work would entail evaluating the neighborhood choice and stripe thickness further and quantifying the comparisons to the higher-order Markov random field.

References

Austad, H.M., (2012). Approximations of Binary Markov Random Fields. Norwegian University of Science and Technology.

Banerjee, S., Gelfand, A.E., Finley, A.O. and Sang, H., (2008). Gaussian predictive process models for large spatial data sets. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70(4):825–848.

Barbu, A. and Zhu, S.C., (2005). Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(8):1239–1253.

Baum, L.E., Petrie, T., Soules, G. and Weiss, N., (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. The Annals of Mathematical Statistics, 41(1):164–171.

Baxter, R.J., (1973). Potts model at the critical temperature. Journal of Physics C: Solid State Physics, 6:L445.

Bennett, J.M., Catovsky, D., Daniel, M.T., Flandrin, G., Galton, D.A.G., Gralnick, H.R. and Sultan, C., (1976). Proposals for the Classification of the Acute Leukaemias French-American-British (FAB) Co-operative Group. British Journal of Haematology, 33(4):451–458.

Besag, J., (1974). Spatial interaction and the statistical analysis of lattice systems. Journal of the Royal Statistical Society. Series B (Methodological), 36(2):192–236.

Bollacker, K.D., (2010). Avoiding a digital dark age. American Scientist, 98(2):106–110.

Brook, D, (1964). On the distinction between the conditional probability and the joint probability approaches in the specification of nearest-neighbour systems. Biometrika, 51(3/4):481–483.

Cornford, D. and Csató, L. and Opper, M., (2005). Sequential, Bayesian geostatistics: a principled method for large data sets. Geographical Analysis, 37(2):183–199.

Cressie, N. and Wikle, C.K., (2011). Statistics for Spatio-temporal Data, volume 465. Wiley.

Dean, J. and Ghemawat, S., (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1):107–113.

Descombes, X., Mangin, J.F., Sigelle, M. and Pechersky, E., (1995). Fine structures preserving Markov model for image processing. Proceedings of the Scandinavian Conference on Image Analysis, pages 349–356.

Dimitrakopoulos, R., Mustapha, H. and Gloaguen, E., (2010). High-order statistics of spatial random fields: Exploring spatial cumulants for modeling complex non-Gaussian and non-linear phenomena. Mathematical Geosciences, 42(1):65–99.

Durbin, J., Koopman, S.J. and Atkinson, A.C., (2001). Time Series Analysis by State Space Methods, volume 15. Oxford University Press Oxford.

Friel, N. and Rue, H., (2007). Recursive computing and simulation-free inference for general factorizable models. Biometrika, 94(3):661–672.

Geman, S. and Geman, D., (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (6):721–741.

Geyer, C.J., (1994). Estimating normalizing constants and reweighting mixtures in Markov chain Monte Carlo. Technical Report 568, University of Minnesota.

Gidas, B.C., (1987). Center for Intelligent Control Systems (US). Parameter estimation for Gibbs distributions, I: Fully observed data.

Goovaerts, P., (1997). Geostatistics for Natural Resources Evaluation. Oxford University Press, USA.

Gray, A.J., Kay, J.W. and Titterington, D.M., (1994). An empirical study of the simulation of various models used for images. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 16(5):507–513.

Grimmett, G., (2010). Probability on Graphs: Random Processes on Graphs and Lattices, volume 1. Cambridge University Press.

Häggström, O., (2002). Finite Markov Chains and Algorithmic Applications, volume 52. Cambridge University Press.

Haggstrom, O. and Nelander, K., (1999). On exact simulation of Markov random fields using coupling from the past. Scandinavian Journal of Statistics, 26(3):395–411.

Hammersley, J.M. and Clifford, P., (1971). Markov fields on finite graphs and lattices.

Huang, F. and Ogata, Y., (2002). Generalized pseudo-likelihood estimates for Markov random fields on lattice. Annals of the Institute of Statistical Mathematics, 54(1):1–18.

Ising, E., (1925). Beitragzurtheorie des ferromagnetismus. ZeitschriftfrPhysik A Hadrons and Nuclei, 31(1):253–258.

Kaiser, M.S. and Cressie, N., (2000). The construction of multivariate distributions from Markov random fields. Journal of Multivariate Analysis, 73(2):199–220.

Karloff, H., Suri, S. and Vassilvitskii, S., (2010). A model of computation for MapReduce. Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pages 938–948. Society for Industrial and Applied Mathematics.

Kindermann, R. and Snell, J.L., (1980). Markov Random Fields and their Applications. American Mathematical Society.

Lafferty, J., McCallum, A. and Pereira, F.C.N., (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. International Conference on Machine Learning (ICML).

Lauritzen, S.L., (1996). Graphical Models, volume 17. Oxford University Press, USA.

Lee, J., Kaiser, M.S. and Cressie, N., (2001). Multiway dependence in exponential family conditional distributions. Journal of Multivariate Analysis, 79(2):171–190.

Levin, D.A., Peres, Y. and Wilmer, E.L., (2009). Markov Chains and Mixing Times. American Mathematical Society.

Mernik, M., Heering, J. and Sloane, A.M., (2005). When and how to develop domain-specific languages. ACM computing surveys (CSUR), 37(4):316–344.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E., (1953). Equation of state calculations by fast computing machines. The Journal of Chemical Physics, 21:1087.

Meyn, S.P., Tweedie, R.L. and Glynn, P.W. (2009). Markov Chains and Stochastic Stability, volume 2. Cambridge University Press.

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(2):451–458.

Morris, R., (1999). Auxiliary variables for Markov random fields with higher order interactions. Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 731–731. Springer.

Mustapha, H. and Dimitrakopoulos, R., (2010). A new approach for geological pattern recognition using high-order spatial cumulants. Computers & Geosciences, 36(3):313–334.

Mustapha, H. and Dimitrakopoulos, R., (2010). High-order stochastic simulation of complex spatially distributed natural phenomena. Mathematical Geosciences, 42(5):457–485.

Nerhus, S., (2009). Parametrization of multi-dimensional Markov chains for rock type modeling. Norwegian University of Science and Technology.

Qian, W. and Titterington, DM., (1991). Multidimensional Markov chain models for image textures. Journal of the Royal Statistical Society. Series B (Methodological), 53(3):661–674.

Reeves, R. and Pettitt, A.N., (2004). Efficient recursions for general factorisable models. Biometrika, 91(3):751–757.

Ripley, B.D., (1987). Stochastic Simulation, volume 183. Wiley Online Library.

Stien, M. and Kolbjørnsen, O., (2011). Facies modeling using a markov mesh model specification. Mathematical Geosciences, 43(6):611–624.

Strebelle, S., (2002). Conditional simulation of complex geological structures using multiple-point statistics. Mathematical Geology, 34(1):1–21.

Tjelmeland, H., (1997). Modeling of the spatial facies distribution by Markov random fields. Geostatistics Wollongong'96, Proceedings of the Fifth International Geostatistical Congress. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Tjelmeland, H. and Besag, J., (1998). Markov Random Fields with Higher-order Interactions. Scandinavian Journal of Statistics, 25(3):415–433.

Wang, J.S. and Swendsen, R.H., (1990). Cluster Montecarlo algorithms. Physica A: Statistical and Theoretical Physics, 167(3):565–579.

Winkler, G., (2003). Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction, volume 27. Springer Verlag.