Self-Avoiding Random Dynamics on Integer Complex Systems

FIRAS HAMZE, D-Wave Systems
ZIYU WANG, University of British Columbia
NANDO DE FREITAS, University of British Columbia

This paper introduces a new specialized algorithm for equilibrium Monte Carlo sampling of binary-valued systems, which allows for large moves in the state space. This is achieved by constructing self-avoiding walks (SAWs) in the state space. As a consequence, many bits are flipped in a single MCMC step. We name the algorithm SARDONICS, an acronym for Self-Avoiding Random Dynamics on Integer Complex Systems. The algorithm has several free parameters, but we show that Bayesian optimization can be used to automatically tune them. SARDONICS performs remarkably well in a broad number of sampling tasks: toroidal ferromagnetic and frustrated Ising models, 3D Ising models, restricted Boltzmann machines and chimera graphs arising in the design of quantum computers.

1. INTRODUCTION

Ising models, also known as Boltzmann machines, are ubiquitous models in physics, machine learning and spatial statistics [Ackley et al. 1985; Besag 1974; Newman and Barkema 1999; Kindermann and Snell 1980; Hopfield 1984]. They have recently lead to a revolution in unsupervised learning known as deep learning, see for example [Hinton and Salakhutdinov 2006; Memisevic and Hinton 2009; Ranzato et al. 2010; Lee et al. 2009; Marlin et al. 2010]. There is also a remarkably large number of other statistical inference problems, where one can apply Rao-Blackwellization [Hamze and de Freitas 2004; Robert and Casella 2004; Liu 2001] to integrate out all continuous variables and end up with a discrete distribution. Examples include topic modeling and Dirichlet processes [Blei et al. 2003], Bayesian variable selection [Tham et al. 2002], mixture models [Liu et al. 2003] and multiple instance learning [Kück and de Freitas 2005]. Thus, if we had effective ways of sampling from discrete distributions, we would solve a great many statistical problems. Moreover, since inference in Ising models can be reduced to max-SAT and counting-SAT problems [Barahona 1982; Welsh 1993; Bian

et al. 2011], efficient Monte Carlo inference algorithms for Ising models would be applicable to a vast domain of computationally challenging problems, including constraint satisfaction and molecular simulation.

Many samplers have been introduced to make large moves in continuous state spaces. Notable examples are the Hamiltonian and Riemann Monte Carlo algorithms [Duane et al. 1987; Neal 2010; Girolami and Calderhead 2011]. However, there has been little comparable effort when dealing with general discrete state spaces. One the most popular algorithms in this domain is the Swendsen-Wang algorithm [Swendsen and Wang 1987]. This algorithm, as shown here, works well for sparse planar lattices, but not for densely connected graphical models. For the latter problems, acceptance rates to make large moves can be very small. For example, as pointed out in [Munoz et al. 2003] Ising models with Metropolis dynamics can require 10^{15} trials to leave a metastable state at low temperatures, and such a simulation would take 10^{10} minutes. For some discrete models, it is however possible to compute the rejection probability of the next move [Munoz et al. 2003; Hamze and de Freitas 2007]. This leads to more efficient algorithms that always accept the next move. The problem with this is that at the next iteration the most favorable move is often to go back to the previous state. That is, these samplers may often get trapped in cycles.

To overcome this problem, this paper presents a specialized algorithm for equilibrium Monte Carlo sampling of binary-valued systems, which allows for large moves in the state space. This is achieved by constructing self-avoiding walks (SAWs) in the state space. As a consequence, many bits are flipped in a single MCMC step.

We proposed a variant of this strategy for constrained binary distributions in [Hamze and de Freitas 2010]. The method presented here applies to unconstrained systems. It has many advancements, but more free parameters than our previous version, thus making the sampler hard to tune. For this reason, we adopt a Bayesian optimization strategy [Močkus 1982; Brochu et al. 2009; Mahendran et al. 2011] to automatically tune these free parameters, thereby allowing for the construction of parameter policies that trade-off exploration and exploitation effectively.

Monte Carlo algorithms for generating SAWs for polymer simulation originated with [Rosenbluth and Rosenbluth 1955]. More recently, biased SAW processes were adopted as proposal distributions for Metropolis algorithms in [Siepmann and Frenkel 1992], where the method was named *configurational bias Monte Carlo*. The crucial distinction between these seminal works and ours is that those authors were concerned with simulation of physical systems that inherently possess the self-avoidance property, namely molecules in some state space. More specifically, the physics of such systems dictated that no component of a molecule may occupy the same spatial location as another; the algorithms they devised took this constraint into account. In contrast, we are treating a different problem, that of sampling from a binary state-space, where a priori, no such requirement exists. Our construction involves the idea of imposing self-avoidance on sequences of generated states as a process to instantiate a rapidly-mixing Markov Chain on the binary state-space. It is therefore more related to the class of optimization algorithms known as Tabu Search [Glover 1989] than to classical polymer SAW simulation methods. To our knowledge, though, a correct equilibrium Monte Carlo algorithm using the Tabu-like idea of self-avoidance in state-space trajectories has not yet been proposed.

We should also point out that sequential Monte Carlo (SMC) methods, such as Hot Coupling and annealed importance sampling, have been proposed to sample from Boltzmann machines [Hamze and de Freitas 2005; Salakhutdinov and Murray 2008]. SMC has also been discussed in the context of rare event simulation and counting SAT problems [Cérou et al. 2012]. Since SMC samplers often use an MCMC kernel as proposal distribution, the MCMC sampler proposed here could enhance those techniques.

The same observation applies when considering other population and meta-MCMC strategies such as parallel tempering [Geyer 1991; Earl and Deem 2005], multicanonical Monte Carlo [Berg and Neuhaus 1991; Gubernatis and Hatano 2000], Wang-Landau sampling [Wang and Landau 2001] and the generalized splitting method [Botev and Kroese 2012].

2. PRELIMINARIES

Consider a binary-valued system defined on the state space $\mathcal{S} \triangleq \{0,1\}^M$, i.e. consisting of M variables each of which can be 0 or 1. The probability of a state $\mathbf{x} = [x_1, \ldots, x_M]$ is given by the Boltzmann distribution:

$$\pi(\mathbf{x}) = \frac{1}{Z(\beta)} e^{-\beta E(\mathbf{x})} \tag{1}$$

where β is an *inverse temperature*. An instance of such a system is the ubiquitous *Ising model* of statistical physics, also called a *Boltzmann machine* by the machine learning community. Our aim in this paper is the generation of states distributed according to a Boltzmann distribution specified by a particular energy function $E(\mathbf{x})$.

A standard procedure is to apply one of the *local* Markov Chain Monte Carlo (MCMC) methodologies such as the Metropolis algorithm or the Gibbs (heat bath) sampler. As is well-known, these algorithms can suffer from issues of poor equilibration ("mixing") and trapping in local minima at low temperatures. More sophisticated methods such as Parallel Tempering [Geyer 1991] and the flat-histogram algorithms (e.g. multicanonical [Berg and Neuhaus 1991] or Wang-Landau [Wang and Landau 2001] sampling) can often dramatically mitigate this problem, but they usually still rely on local MCMC at some stage. The ideas presented in this paper relate to MCMC sampling using larger changes of state than those of local algorithms. They can be applied on their own or in conjunction with the previously-mentioned advanced methods. In this paper, we focus on the possible advantage of our algorithms over local methods.

Given a particular state $\mathbf{x} \in \mathcal{S}$, we denote by $\mathcal{S}_n(\mathbf{x})$ the set of all states at Hamming distance n from $\mathbf{x}$. For example if $M = 3$ and $\mathbf{x} = [1,1,1]$, then $\mathcal{S}_0(\mathbf{x}) = \{[1,1,1]\}$, $\mathcal{S}_1(\mathbf{x}) = \{[0,1,1],[1,0,1],[1,1,0]\}$, etc. Clearly $|\mathcal{S}_n(\mathbf{x})| = \binom{M}{n}$.

We define the set of *bits* in two states $\mathbf{x}, \mathbf{y}$ that agree with each other: let $\mathcal{A}(\mathbf{x}, \mathbf{y}) = \{i | x_i = y_i\}$. For instance if $\mathbf{x} = [0,1,0,1]$ and $\mathbf{y} = [0,0,0,1]$, then $\mathcal{A}(\mathbf{x}, \mathbf{y}) = \{1,3,4\}$. Clearly, $\mathcal{A}(\mathbf{x}, \mathbf{y}) = \mathcal{A}(\mathbf{y}, \mathbf{x})$ and $\mathcal{A}(\mathbf{x}, \mathbf{x}) = \{1,2,\ldots,M\}$.

Another useful definition is that of the *flip operator*, which simply inverts bit i in a state, $F(\mathbf{x}, i) \triangleq (x_1, \ldots, \bar{x}_i, \ldots, x_M)$, and its extension that acts on a sequence of indices, i.e. $F(\mathbf{x}, i_1, i_2, \ldots, i_k) \triangleq F(F(\ldots, F(\mathbf{x}, i_1), i_2), \ldots, i_k)$.

For illustration, we describe a simple Metropolis algorithm in this framework. Consider a state $\mathbf{x} \in \mathcal{S}$; a new state $\mathbf{x}' \in \mathcal{S}_1(\mathbf{x})$ is generated by flipping a random bit of $\mathbf{x}$, i.e. $\mathbf{x}' = F(\mathbf{x}, i)$ for i chosen uniformly from $\{1, \ldots, M\}$, and accepting the new state with probability:

$$\alpha = \min(1, e^{-\beta(E(\mathbf{x}') - E(\mathbf{x}))}) \tag{2}$$

If all the *single-flip neighbors* of $\mathbf{x}$, i.e. the states resulting from $\mathbf{x}$ via a single bit perturbation, are of higher energy than $E(\mathbf{x})$, the acceptance rate will be small at low temperatures.

3. SARDONICS

In contrast to the traditional single-flip MCMC algorithms, the elementary unit of our algorithm is a type of move that allows for large changes of state *and* tends to propose them such that they are energetically favourable. We begin by describing the move

operator and its incorporation into the Monte Carlo algorithm we call SARDONICS, an acronym for *Self-Avoiding Random Dynamics on Integer Complex Systems*. This algorithm will then be shown to satisfy the theoretical requirements that ensure correct asymptotic sampling. The move procedure aims to force exploration away from the current state. In that sense it has a similar spirit to the *Tabu Search* [Glover 1989] optimization heuristic, but the aim of SARDONICS is equilibrium sampling, a problem more general than (and in many situations at least as challenging as) minimization.

Suppose that we have a state $\mathbf{x}_0$ on $\mathcal{S}$. It is possible to envision taking a special type of biased *self-avoiding walk* (SAW) of length k *in the state space*, in other words a sequence of states such that no state recurs in the sequence. In this type of SAW, states on sets $\{\mathcal{S}_1(\mathbf{x}_0), \ldots, \mathcal{S}_k(\mathbf{x}_0)\}$ are visited consecutively. To generate this sequence of states $(\mathbf{u}_1, \ldots, \mathbf{u}_k)$, at step i of the procedure a bit is chosen to be flipped from those that have not yet flipped relative to $\mathbf{x}_0$. Specifically, an element σ_i is selected from $\mathcal{A}(\mathbf{x}_0, \mathbf{u}_{i-1})$, with $\mathbf{u}_0 \triangleq \mathbf{x}_0$, to yield state $\mathbf{u}_i = F(\mathbf{u}_{i-1}, \sigma_i)$, or equivalently, $\mathbf{u}_i = F(\mathbf{x}_0, \sigma_1, \ldots, \sigma_i)$. From $\mathbf{u}_{i-1} \in \mathcal{S}_{i-1}(\mathbf{x}_0)$, the set of states on $\mathcal{S}_i$ that can result from flipping a bit in $\mathcal{A}(\mathbf{x}_0, \mathbf{u}_{i-1})$ are called the *single-flip neighbours* of $\mathbf{u}_{i-1}$ on $\mathcal{S}_i(\mathbf{x}_0)$. The set of bits that can flip with nonzero probability are called the *allowable moves* at each step. A diagrammatic depiction of the SAW is shown in Figure 1.

The elements $\{\sigma_i\}$ are sampled in an *energy-biased* manner as follows:

$$q(\sigma_i = l | \sigma_{i-1}, \ldots, \sigma_1, \mathbf{x}_0) = \begin{cases} \dfrac{e^{-\gamma E(F(\mathbf{u}_{i-1}, l))}}{\sum_{j \in \mathcal{A}(\mathbf{x}_0, \mathbf{u}_{i-1})} e^{-\gamma E(F(\mathbf{u}_{i-1}, j))}} & \begin{array}{l} l \in \mathcal{A}(\mathbf{x}_0, \mathbf{u}_{i-1}) \text{ and} \\ \mathbf{u}_{i-1} = F(\mathbf{x}_0, \sigma_1, \ldots, \sigma_{i-1}) \end{array} \\ \\ 0 & \text{otherwise} \end{cases}$$

(3)

The larger the value that simulation parameter γ is assigned, the more likely the proposal q is to sample the lower-energy neighbors of $\mathbf{u}_{i-1}$. Conversely if it is zero, a neighbor on $\mathcal{S}_i(\mathbf{x}_0)$ is selected completely at random. In principle, the value of γ will be seen to be arbitrary; indeed it can even be different at each step of the SAW. We will have more to say about the choice of γ, as well as the related issue of the SAW lengths, in Section 4.2.

The motivation behind using such an energy-biased scheme is that when proposing large changes of configuration, it may generate final states that are "typical" of the system's target distribution. To make a big state-space step, one may imagine uniformly perturbing a large number of bits, but this is likely to yield states of high energy, and an MCMC algorithm will be extremely unlikely to accept the move at low temperatures.

At this point it can be seen why the term "self-avoiding" aptly describes the processes. If we imagine the state-space to be a high-dimensional lattice, with the individual states lying at the vertices and edges linking states that are single-flip neighbors, a self-avoiding walk on this graph is a sequence of states that induce a connected, acyclic subgraph of the lattice. In the move procedure we have described, a state can never occur twice at any stage within it and so the process is obviously self-avoiding.

Note however that the construction imposes a stronger condition on the state sequence; once a transition occurs from state $\mathbf{x}$ to state $F(\mathbf{x}, i)$, not only may state $\mathbf{x}$ not appear again, but neither may *any* state $\mathbf{y}$ with $y_i = x_i$. It seems natural to ask why not to use a less constrained SAW process, namely one that avoids returns to *individual* states and likely more familiar to those experienced in molecular and polymer simulations, without eliminating an entire dimension at each step. In our experience, trying to construct such a SAW as a proposal requires excessive computational memory and time to yield good state-space traversal. A local minimum "basin" of a combina-
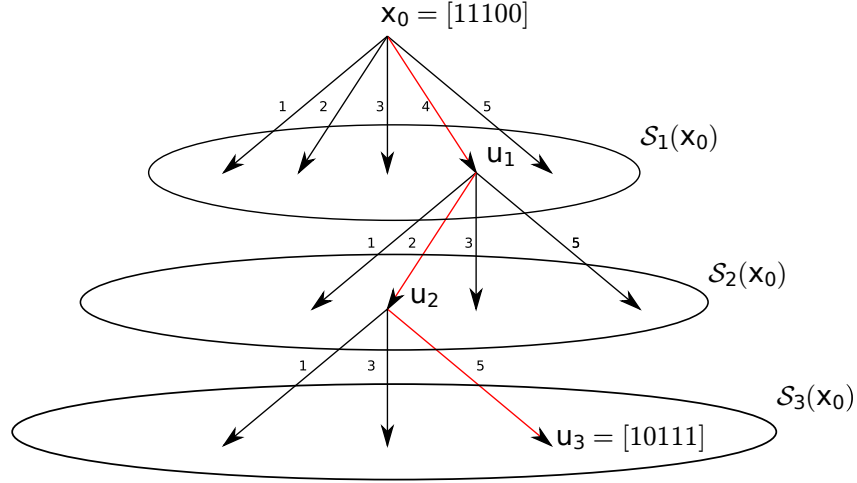
Fig. 1: A visual illustration of the move process for a SAW of length 3. The arrows represent the allowable moves from a state at that step; the red arrow shows the actual move taken in this example. With the system at state $\mathbf{x}_0$, the SAW begins. Bit 4 of $\mathbf{x}_0$ has been sampled for flipping according to Equation 3 to yield state $\mathbf{u}_1 = [11110]$; the process is repeated until state $\mathbf{u}_3$ on $\mathcal{S}_3(\mathbf{x}_0)$ is reached. The sequence of states taken by the SAW is $\boldsymbol{\sigma} = [4, 2, 5]$.

toric landscape can potentially contain a massive number of states, and a process that moves by explicitly avoiding particular states may be doomed to wander within and visit a substantial portion of the basin prior to escaping.

Let $\mathbf{x}_1$ be the *final* state reached by the SAW, i.e. $\mathbf{u}_k$ for a length k walk. By multiplying the SAW flipping probabilities, we can straightforwardly obtain the probability of moving from state $\mathbf{x}_0$ to $\mathbf{x}_1$ *along the SAW $\boldsymbol{\sigma}$*, which we call $q(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0)$:

$$q(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0) \triangleq \delta_{\mathbf{x}_1}[F(\mathbf{x}_0, \boldsymbol{\sigma})] \prod_{i=1}^{k} q(\sigma_i|\mathbf{u}_{i-1}) \tag{4}$$

The delta function simply enforces the fact that the final *state* $\mathbf{x}_1$ must result from the sequence of flips in $\boldsymbol{\sigma}$ from $\mathbf{x}_0$. The set of $\{\boldsymbol{\sigma}\}$ such that $q(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0) > 0$ are termed the *allowable SAWs* between $\mathbf{x}_0$ and $\mathbf{x}_1$.

Ideally, to implement a Metropolis-Hastings (MH) algorithm using the SAW proposal, we would like to evaluate the *marginal* probability of proposing $\mathbf{x}_1$ from $\mathbf{x}_0$, which we call $q(\mathbf{x}_1|\mathbf{x}_0)$, so that the move would be accepted with the usual MH ratio:

$$\alpha_m(\mathbf{x}_0, \mathbf{x}_1) \triangleq \min\left(1, \frac{\pi(\mathbf{x}_1)q(\mathbf{x}_0|\mathbf{x}_1)}{\pi(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)}\right) \tag{5}$$

Unfortunately, for all but small values of the walk lengths k, marginalization of the proposal is intractable due to the potentially massive number of allowable SAWs between the two states.

To assist in illustrating our solution to this, we recall that a sufficient condition for a Markov transition kernel K to have target π as its stationary distribution is *detailed balance:*

$$\pi(\mathbf{x}_0)K(\mathbf{x}_1|\mathbf{x}_0) = \pi(\mathbf{x}_1)K(\mathbf{x}_0|\mathbf{x}_1) \tag{6}$$

One special case obtains if we used the marginalized proposal $q(\mathbf{x}_1|\mathbf{x}_0)$ followed by the MH accept rule,

$$K_m(\mathbf{x}_1|\mathbf{x}_0) \triangleq q(\mathbf{x}_1|\mathbf{x}_0)\alpha_m(\mathbf{x}_0, \mathbf{x}_1) \qquad (7)$$

As we cannot compute $q(\mathbf{x}_1|\mathbf{x}_0)$, we shall use a kernel $K(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0)$ defined on the *joint space* of SAWs and states, and show that with some care, detailed balance (6) can still *hold marginally*. It will be clear, though that this does *not* mean that the resultant marginal kernel $K(\mathbf{x}_1|\mathbf{x}_0)$ is the same as that in (7) obtained using MH acceptance on the marginal *proposal* .

Define the *sequence reversal operator* $R(\boldsymbol{\sigma})$ to simply return a sequence consisting of the elements of $\boldsymbol{\sigma}$ in reverse order; for example $R([2, 3, 1, 4]) = [4, 1, 3, 2]$. One can straightforwardly observe that each allowable SAW $\boldsymbol{\sigma}$ from $\mathbf{x}_0$ to $\mathbf{x}_1$ can be uniquely mapped to the allowable SAW $R(\boldsymbol{\sigma})$ from $\mathbf{x}_1$ to $\mathbf{x}_0$. For example in Figure 1, the SAW $R(\boldsymbol{\sigma}) = [5, 2, 4]$ can be seen to be allowable from $\mathbf{x}_1$ to $\mathbf{x}_0$. Next, we have the following somewhat more involved concept, a variant of which we introduced in [Hamze and de Freitas 2010]:

Definition 3.1.   Consider a Markov kernel $K(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0)$ whose support set coincides with that of (4). We say that *pathwise detailed balance* holds if

$$\pi(\mathbf{x}_0)K(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0) = \pi(\mathbf{x}_1)K(\mathbf{x}_0, R(\boldsymbol{\sigma})|\mathbf{x}_1)$$

for all $\boldsymbol{\sigma}, \mathbf{x}_0, \mathbf{x}_1$.

It turns out that pathwise detailed balance is a *stronger condition* than marginal detailed balance. In other words,

PROPOSITION 3.2.   *If the property in Definition 3.1 holds for a transition kernel K of the type described there, then* $\pi(\mathbf{x}_0)K(\mathbf{x}_1|\mathbf{x}_0) = \pi(\mathbf{x}_1)K(\mathbf{x}_0|\mathbf{x}_1)$

PROOF.   Suppose, for given $\mathbf{x}_0, \mathbf{x}_1$, we summed both sides of the equation enforcing pathwise detailed balance over all allowable SAWs $\{\boldsymbol{\sigma}'\}$ from $\mathbf{x}_0$ to $\mathbf{x}_1$, i.e.

$$\sum_{\boldsymbol{\sigma}'} \pi(\mathbf{x}_0)K(\mathbf{x}_1, \boldsymbol{\sigma}'|\mathbf{x}_0) = \sum_{\boldsymbol{\sigma}'} \pi(\mathbf{x}_1)K(\mathbf{x}_0, R(\boldsymbol{\sigma}')|\mathbf{x}_1)$$

The left-hand summation marginalizes the kernel over allowable SAWs and hence results in $\pi(\mathbf{x}_0)K(\mathbf{x}_1|\mathbf{x}_0)$. The observation above that each allowable SAW from $\mathbf{x}_0$ to $\mathbf{x}_1$ can be reversed to yield an allowable one from $\mathbf{x}_1$ to $\mathbf{x}_0$ implies that the right-hand side is simply a re-ordered summation over all allowable SAWs from $\mathbf{x}_1$ to $\mathbf{x}_0$, and can thus be written as $\pi(\mathbf{x}_1)K(\mathbf{x}_0|\mathbf{x}_1)$.   □

We are now ready to state the final form of the algorithm (see Algorithm 1), which can be seen to instantiate a Markov chain satisfying pathwise detailed balance. After proposing $(\mathbf{x}_1, \boldsymbol{\sigma})$ using the SAW process, we accept the move with the ratio:

$$\alpha(\mathbf{x}_0, \mathbf{x}_1, \boldsymbol{\sigma}) \triangleq \min\left(1, \frac{\pi(\mathbf{x}_1)q(\mathbf{x}_0, R(\boldsymbol{\sigma})|\mathbf{x}_1)}{\pi(\mathbf{x}_0)q(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0)}\right) \qquad (8)$$

The computational complexity of evaluating this accept ratio is of the same order as that required to sample the proposed SAWs/state; the only additional operations required are those needed to evaluate the reverse proposal appearing in the numerator, which are completely analogous to those involved in calculating the forward proposal.

Let us take a closer look at the marginal transition kernel $K(\mathbf{x}_1|\mathbf{x}_0)$. We can factor the joint *proposal* into:

$$q(\mathbf{x}_1, \boldsymbol{\sigma}|\mathbf{x}_0) = q(\mathbf{x}_1|\mathbf{x}_0)q(\boldsymbol{\sigma}|\mathbf{x}_0, \mathbf{x}_1)$$

ALGORITHM 1: SARDONICS

1: Let $\mathbf{x}_0$ be the initial state.
2: **for** $i = 1, 2, \ldots, n$ **do**
3:    Set $\mathbf{u}_0 = \mathbf{x}_i$
4:    **for** $j = 1, 2, \ldots, k$ **do**
5:       Sample σ_j according to equation (3)
6:       Flip bit indexed by σ_j: $\mathbf{u}_j = F(\mathbf{u}_{j-1}, \sigma_j)$
7:    **end for**
8:    Compute $\alpha(\mathbf{x}_i, \mathbf{u}_k, \boldsymbol{\sigma}) \triangleq \min\left(1, \frac{\pi(\mathbf{u}_k)q(\mathbf{x}_i, R(\boldsymbol{\sigma})|\mathbf{u}_k)}{\pi(\mathbf{x}_i)q(\mathbf{u}_k, \boldsymbol{\sigma}|\mathbf{x}_i)}\right)$
9:    Draw $u \sim U(0, 1)$
10:   **if** $u \le \alpha(\mathbf{x}_i, \mathbf{u}_k, \boldsymbol{\sigma})$ **then**
11:      $\mathbf{x}_i = \mathbf{u}_k$
12:   **else**
13:      $\mathbf{x}_i = \mathbf{x}_{i-1}$
14:   **end if**
15: **end for**

Of course, if we are assuming that $q(\mathbf{x}_1|\mathbf{x}_0)$ is intractable to evaluate, then the conditional $q(\boldsymbol{\sigma}|\mathbf{x}_0, \mathbf{x}_1)$ must be so as well, but it is useful to consider. If we now summed both sides of the *joint* probability of moving from $\mathbf{x}_0$ to $\mathbf{x}_1$ over allowable paths, we would observe:

$$\sum_{\boldsymbol{\sigma}'} \pi(\mathbf{x}_0)K(\mathbf{x}_1, \boldsymbol{\sigma}'|\mathbf{x}_0) = \pi(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0) \sum_{\boldsymbol{\sigma}'} q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1)\alpha(\mathbf{x}_0, \mathbf{x}_1, \boldsymbol{\sigma}')$$

The summation on the right-hand side is thus the *conditional expectation* of the accept rate given that we are attempting to move from $\mathbf{x}_0$ to $\mathbf{x}_1$; we call it

$$\alpha(\mathbf{x}_0, \mathbf{x}_1) \triangleq \sum_{\boldsymbol{\sigma}'} q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1)\alpha(\mathbf{x}_0, \mathbf{x}_1, \boldsymbol{\sigma}') \tag{9}$$

and it defines an *effective acceptance rate* between $\mathbf{x}_0$ and $\mathbf{x}_1$ under the sampling regime described since

$$K(\mathbf{x}_1|\mathbf{x}_0) = q(\mathbf{x}_1|\mathbf{x}_0)\alpha(\mathbf{x}_0, \mathbf{x}_1).$$

It is not difficult to show that $\alpha(\mathbf{x}_0, \mathbf{x}_1) \neq \alpha_m(\mathbf{x}_0, \mathbf{x}_1)$, i.e. the marginal accept rate for the joint proposal is not the same as the one that results from using the marginalized proposal. In fact we can make a stronger statement:

PROPOSITION 3.3. *For every pair of states* $(\mathbf{x}_0, \mathbf{x}_1)$, $\alpha(\mathbf{x}_0, \mathbf{x}_1) \le \alpha_m(\mathbf{x}_0, \mathbf{x}_1)$

PROOF. For conciseness, let $C \triangleq \frac{1}{\alpha_m(\mathbf{x}_0, \mathbf{x}_1)} = \frac{\pi(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)}{\pi(\mathbf{x}_1)q(\mathbf{x}_0|\mathbf{x}_1)}$. Define the sets $\mathcal{A} \triangleq \{\boldsymbol{\sigma}|\frac{q(R(\boldsymbol{\sigma})|\mathbf{x}_1, \mathbf{x}_0)}{q(\boldsymbol{\sigma}|\mathbf{x}_0, \mathbf{x}_1)} \ge C\}$ and $\bar{\mathcal{A}} \triangleq \{\boldsymbol{\sigma}|\frac{q(R(\boldsymbol{\sigma})|\mathbf{x}_1, \mathbf{x}_0)}{q(\boldsymbol{\sigma}|\mathbf{x}_0, \mathbf{x}_1)} < C\}$. Then

$$\begin{aligned}
\alpha(\mathbf{x}_0, \mathbf{x}_1) &= \sum_{\boldsymbol{\sigma}'} q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1) \min\left(1, \frac{\pi(\mathbf{x}_1)q(\mathbf{x}_0|\mathbf{x}_1)q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0)}{\pi(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1)}\right) \\
&= \sum_{\boldsymbol{\sigma}' \in \mathcal{A}} q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1) + \frac{\pi(\mathbf{x}_1)q(\mathbf{x}_0|\mathbf{x}_1)}{\pi(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)} \sum_{\boldsymbol{\sigma}' \in \bar{\mathcal{A}}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0) \\
&= \sum_{\boldsymbol{\sigma}' \in \mathcal{A}} q(\boldsymbol{\sigma}'|\mathbf{x}_0, \mathbf{x}_1) + \frac{1}{C} \sum_{\boldsymbol{\sigma}' \in \bar{\mathcal{A}}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0)
\end{aligned}$$

But by definition, for $\boldsymbol{\sigma} \in \mathcal{A}$, $q(\boldsymbol{\sigma}|\mathbf{x}_0, \mathbf{x}_1) \leq \frac{1}{C} q(R(\boldsymbol{\sigma})|\mathbf{x}_1, \mathbf{x}_0)$. Therefore,

$$
\begin{aligned}
\alpha(\mathbf{x}_0, \mathbf{x}_1) &\leq \frac{1}{C} \sum_{\boldsymbol{\sigma}' \in \mathcal{A}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0) + \frac{1}{C} \sum_{\boldsymbol{\sigma}' \in \bar{\mathcal{A}}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0) \\
&= \frac{1}{C} \Big(\sum_{\boldsymbol{\sigma}' \in \mathcal{A}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0) + \sum_{\boldsymbol{\sigma}' \in \bar{\mathcal{A}}} q(R(\boldsymbol{\sigma}')|\mathbf{x}_1, \mathbf{x}_0) \Big) \\
&= \frac{1}{C} = \alpha_m(\mathbf{x}_0, \mathbf{x}_1)
\end{aligned}
$$

□

There is another technical consideration to address. The reader may have remarked that while detailed balance does indeed hold for our algorithm, if the SAW length is constant at $k > 1$, then the resulting Markov chain is no longer *irreducible*. In other words, not every $\mathbf{x} \in \mathcal{S}$ is reachable with nonzero probability regardless of the initial state. For example if $k = 2$ and the initial state $\mathbf{x}_0 = [0, 0, 0, 0, 0, 0]$, then the state $\mathbf{x} = [0, 0, 0, 0, 0, 1]$ can never be visited. Fortunately, this is a rather easy issue to overcome; one possible strategy is to randomly choose the SAW length prior to each step from a set that ensures that the whole state space can eventually be visited. A trivial example of such a set is any collection of integers that include unity, i.e. such that single-flip moves are allowed. Another is a set that includes consecutive integers, i.e. $\{k_0, k_0 + 1\}$ for any $k_0 < M$. This latter choice could allow states separated by a single bit to occur in two steps; in the example above, if the set of lengths was $\{3, 4\}$ then we could have $[0, 0, 0, 0, 0, 0] \rightarrow [0, 0, 1, 1, 1, 1] \rightarrow [0, 0, 0, 0, 0, 1]$. While this shows how to enforce theoretical correctness of the algorithm, in upcoming sections we will discuss the issue of practical choice of the lengths in the set.

The experimental section discusses the practical matter of efficiently sampling the SAW for systems with sparse connectivity, such as the 2D and 3D Ising models.

3.1. Iterated SARDONICS

The SARDONICS algorithm presented in Section 3 is sufficient to work effectively on many types of system, for example Ising models with ferromagnetic interactions. This section, however, will detail a more advanced strategy for proposing states that uses the state-space SAW of Section 3 as its basic move. The overall idea is to extend the trial process so that the search for a state to propose can continue from the state resulting from a *concatenated series* of SAWs from $\mathbf{x}_0$. The reader familiar with combinatorial optimization heuristics will note the philosophical resemblance to the class of algorithms termed *iterated local search* [Hoos and Stutzle 2004], but will again bear in mind that in the present work, we are interested in equilibrium sampling as opposed to optimization.

We begin by noting that restriction to a *single* SAW is unnecessary. We can readily consider in principle an arbitrary concatenation of SAWs $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_N)$. A straightforward extension of the SAW proposal is then to select some number of iterations N (which need not be the same from one move attempt to the next,) to generate $\mathbf{x}_1$ from $\mathbf{x}_0$ by sampling from the concatenated proposal, defined to be

$$
g(\mathbf{x}_1, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_N | \mathbf{x}_0) \triangleq q(\mathbf{y}^1, \sigma_1 | \mathbf{x}_0) q(\mathbf{y}^2, \sigma_2 | \mathbf{y}^1) \ldots \tag{10}
$$
$$
q(\mathbf{y}^N, \boldsymbol{\sigma}_N | \mathbf{y}^{N-1}) \delta_{\mathbf{x}_1}(\mathbf{y}^N)
$$

and to accept the move with probability

$$
\alpha(\mathbf{x}_0, \mathbf{x}_1, \boldsymbol{\sigma}_1 \ldots, \boldsymbol{\sigma}_N) = \min \left(1, \frac{\pi(\mathbf{x}_1) g(\mathbf{x}_0, R(\boldsymbol{\sigma}_N), \ldots R(\boldsymbol{\sigma}_1) | \mathbf{x}_1)}{\pi(\mathbf{x}_0) g(\mathbf{x}_1, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_N | \mathbf{x}_0)} \right) \tag{11}
$$

ALGORITHM 2: Iterated SARDONICS

1: Let $\mathbf{x}_0$ be the initial state.
2: **for** $i = 1, 2, \ldots, n$ **do**
3:     Set $\mathbf{u}_0 = \mathbf{x}_i$
4:     **for** $p = 1, 2, \ldots, N$ **do**
5:       **for** $j = 1, 2, \ldots, k$ **do**
6:          Sample $\sigma_{p,j}$ according to $q(\sigma_{p,j} = l | \sigma_{p,j-1}, \ldots, \sigma_{p,1}, \mathbf{x}_0)$ defined by equation 3
7:          Flip bit indexed by $\sigma_{p,j}$: $\mathbf{u}_j = F(\mathbf{u}_{j-1}, \sigma_{p,j})$
8:       **end for**
9:       Set $\mathbf{y}^p = \mathbf{u}_k$ and $\mathbf{u}_0 = \mathbf{y}^p$
10:    **end for**
11:    Compute $\alpha(\mathbf{x}_i, \mathbf{y}^N, \boldsymbol{\sigma}_1 \ldots, \boldsymbol{\sigma}_N) = \min\left(1, \frac{\pi(\mathbf{y}^N)g(\mathbf{x}_i, R(\boldsymbol{\sigma}_N), \ldots R(\boldsymbol{\sigma}_1)|\mathbf{y}^N)}{\pi(\mathbf{x}_i)g(\mathbf{y}^N, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_N|\mathbf{x}_i)}\right)$
12:    Draw $u \sim U(0,1)$
13:    **if** $u \leq \alpha(\mathbf{x}_i, \mathbf{y}^N, \boldsymbol{\sigma})$ **then**
14:       $\mathbf{x}_i = \mathbf{y}^N$
15:    **else**
16:       $\mathbf{x}_i = \mathbf{x}_{i-1}$
17:    **end if**
18: **end for**

In (10), the superscripted $\{\mathbf{y}^i\}$ refer to the "intermediate" states on $\mathcal{S}$ generated by the flip sequences, and the functions q are the SAW proposals discussed in Section 3. The proposed state $\mathbf{x}_1$ is identical to the final intermediate state $\mathbf{y}^N$; to avoid obscuring the notation with too many delta functions we take it as implicit that the the proposal evaluates to zero for intermediate states that do not follow from the flip sequences $\{\boldsymbol{\sigma}_i\}$.

We refer to this as the *iterated SARDONICS* algorithm (See algorithm 2). Its potential merit over taking a single SAW is that it may generate more distant states from $\mathbf{x}_0$ that are favorable. Unfortunately, a priori there is no guarantee that the final state $\mathbf{x}_1$ will be more worthy of acceptance than the intermediate states visited in the process; it is computationally wasteful to often propose long sequences of flips that end up rejected, especially when potentially desirable states may have been passed over. It is thus very important to choose the right value of N. For this reason, we will introduce Bayesian optimization techniques to adapt N and other parameters automatically in section 4.

3.2. Mixture of SAWs

A further addition we made to the basic algorithm was the generalization of the proposal to a mixture of SAW processes. Each segment of the iterated procedure introduced in Section 3.1 could in principle operate at a different level of the biasing parameter γ. A possible strategy one can envision is to occasionally take a *pair* of SAWs with the first at a small value of γ and the next at a large value. The first step encourages exploration away from the current state, and the second a search for a new low-energy state. More specifically, for the two SAWs $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$, we can have a proposal of the form:

$$q(\mathbf{x}_1, \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 | \mathbf{x}_0) = q(\mathbf{y}^1, \boldsymbol{\sigma}_1 | \mathbf{x}_0, \gamma_L) q(\mathbf{x}_1, \boldsymbol{\sigma}_2 | \mathbf{y}^1, \gamma_H)$$

where γ_H and γ_L are high and low inverse temperature biases respectively. Unfortunately, such a method on its own will likely result in a high rejection rate; the numer-

ator of the MH ratio enforcing detailed balance will be:

$$q(\mathbf{x}_0, R(\boldsymbol{\sigma}_2), R(\boldsymbol{\sigma}_1)|\mathbf{x}_1) = q(\mathbf{y}^1, R(\boldsymbol{\sigma}_2)|\mathbf{x}_1, \gamma_L)q(\mathbf{x}_0, R(\boldsymbol{\sigma}_1)|\mathbf{y}^1, \gamma_H)$$

The probability of taking the reverse sequences will likely be very small compared to those of the forward sequences. In particular, the likelihood of descending at low-temperature biasing parameter along the sequence $R(\boldsymbol{\sigma}_1)$, where $\boldsymbol{\sigma}_1$ was generated with the high-temperature parameter, will be low.

Our simple approach to dealing with this is to define the proposal to be a *mixture* of three types of SAW processes. The mixture weights, P_{LL}, P_{HL}, P_{LH}, with $P_{LL} + P_{LH} + P_{HL} = 1$, define, respectively, the frequencies of choosing a proposal unit consisting of a *pair* of SAWs sampled with (γ_L, γ_L), (γ_H, γ_L), and (γ_L, γ_H). The first proposal type encourages local exploration; both SAWs are biased towards low-energy states. The second one, as discussed, is desirable as it may help the sampler escape from local minima. The last one may seem somewhat strange; since it ends with sampling at γ_H, it will tend to generate states with high energy which will consequently be rejected. The purpose of this proposal, however, is to assist in the acceptance of the HL exploratory moves due to its presence in the mixture proposal. Thus, P_{LH} is a parameter that must be carefully tuned. If it is too large, it will generate too many moves that will end up rejected due to their high energy; if too small, its potential to help the HL moves be accepted will be diminished. The mixture parameters are thus ideal candidates to explore the effectiveness of adaptive strategies to tune MCMC.

4. ADAPTING SARDORNICS WITH BAYESIAN OPTIMIZATION

SARDONICS has several free parameters: upper and lower bounds on the SAW length (k_u and k_l respectively), $\gamma_H, \gamma_L, P_{LL}, P_{HL}, P_{LH}$ as explained in the previous section, and finally the number of concatenated SAWs N. We group these free parameters under the symbol $\boldsymbol{\theta} = \{k_u, k_l, \gamma_H, \gamma_L, P_{LL}, P_{HL}, P_{LH}, N\}$. Each $\boldsymbol{\theta}$ defines a stochastic policy, where the SAW length k is chosen at random in the set $[k_l, k_u]$ and where the SAW processes are chosen according to the mixture probabilities P_{LL}, P_{HL}, P_{LH}. Tuning all these parameters by hand is an onerous task. Fortunately, this challenge can be surmounted using adaptation. Stochastic approximation methods, at first sight, might appear to be good candidates for carrying out this adaptation. They have become increasingly popular in the subfield of adaptive MCMC [Haario et al. 2001; Andrieu and Robert 2001; Roberts and Rosenthal 2009; Vihola 2010]. There are a few reasons, however, that force us to consider alternatives to stochastic approximation.

In our discrete domain, there are no obvious optimal acceptance rates that could be used to construct the objective function for adaptation. Instead, we choose to optimize $\boldsymbol{\theta}$ so as to minimize the area under the auto-correlation function (of the energies in this paper) up to a specific lag. This objective was previously adopted in [Andrieu and Robert 2001; Mahendran et al. 2011]. One might argue that this is a reasonable objective given that researchers and practitioners often use it to diagnose the convergence of MCMC algorithms. However, the computation of gradient estimates for this objective is very involved and far from trivial [Andrieu and Robert 2001]. This motivates the introduction of a gradient-free optimization scheme known as Bayesian optimization [Močkus 1982; Brochu et al. 2009]. Bayesian optimization also has the advantage that it trades-off exploration and exploitation of the objective function. In contrast, gradient methods are designed to exploit locally and may, as a result, get trapped in unsatisfactory local optima.

The proposed adaptive strategy consists of two phases: adaptation and sampling. In the adaptation phase Bayesian optimization is used to construct a randomized policy. In the sampling phase, a mixture of MCMC kernels selected according to the learned randomized policy is used to explore the target distribution. Experts in adap-

---

ALGORITHM 3: Adaptation phase of SARDONICS

1: **for** $i = 1, 2, \ldots, I$ **do**
2:     Run SARDONICS for L steps with parameters $\boldsymbol{\theta}_i$.
3:     Use the drawn samples to obtain a noisy evaluation of the objective function:
       $z_i = h(\boldsymbol{\theta}_i) + \epsilon$.
4:     Augment the observation set $\mathcal{D}_{1:i} = \{\mathcal{D}_{1:i-1}, (\boldsymbol{\theta}_i, z_i)\}$.
5:     Update the GP's sufficient statistics.
6:     Find $\boldsymbol{\theta}_{i+1}$ by optimizing an acquisition function: $\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta}} u(\boldsymbol{\theta}|\mathcal{D}_{1:i})$.
7: **end for**

---

tive MCMC would have realized that there is no theoretical need for this two-phase procedure. Indeed, if the samplers are uniformly ergodic, which is the case in our discrete setting, and adaptation vanishes asymptotically, then ergodicity can still be established [Roberts and Rosenthal 2009; Andrieu and Robert 2001]. However, in our setting the complexity of the adaptation scheme increases with time. Specifically, Bayesian optimization, as we shall soon outline in detail, requires fitting a Gaussian process to I points, where I is the number of iterations of the adaptation procedure. In the worst case, this computation is $O(I^3)$. There are techniques based on conjugate gradients, fast multipole methods and low rank approximations to speed up this computation [Freitas et al. 2005; Halko et al. 2011]. However, none of these overcome the issue of increasing storage and computational needs. So, for pragmatic reasons, we restrict the number of adaptation steps. We will discuss the consequence of this choice in the experiments and come back to this issue in the concluding remarks.

The two phases of our adaptive strategy are discussed in more detail subsequently.

4.1. Adaptation Phase

Our objective function for adaptive MCMC is the area under the auto-correlation function up to a specific lag. This objective is intractable, but noisy observations of its value can be obtained by running the Markov chain for a few steps with a specific choice of parameters $\boldsymbol{\theta}_i$. Bayesian optimization can be used to propose a new candidate $\boldsymbol{\theta}_{i+1}$ by approximating the unknown function using the entire history of noisy observations and a prior over this function. The prior used in this paper is a Gaussian process.

The noisy observations are used to obtain the predictive distribution of the Gaussian process. An expected utility function derived in terms of the sufficient statistics of the predictive distribution is optimized to select the next parameter value $\boldsymbol{\theta}_{i+1}$. The overall procedure is shown in Algorithm 3. We refer readers to [Brochu et al. 2009] and [Lizotte 2008] for in-depth reviews of Bayesian optimization.

The unknown objective function $h(\cdot)$ is assumed to be distributed according to a Gaussian process with mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot)$:

$$h(\cdot) \sim GP(m(\cdot), k(\cdot, \cdot)).$$

We adopt a zero mean function $m(\cdot) = 0$ and an anisotropic Gaussian covariance that is essentially the popular automatic relevance determination (ARD) kernel [Rasmussen and Williams 2006]:

$$k(\boldsymbol{\theta}_j, \boldsymbol{\theta}_k) = \exp\left(-\frac{1}{2}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_k)^T \text{diag}(\psi)^{-2}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_k)\right)$$

where $\psi \in \mathbb{R}^d$ is a vector of hyper-parameters. The Gaussian process is a surrogate model for the true objective, which involves intractable expectations with respect to the invariant distribution and the MCMC transition kernels.

We assume that the noise in the measurements is Gaussian: $z_i = h(\boldsymbol{\theta}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\eta^2)$. It is possible to adopt other noise models [Diggle et al. 1998]. Our Gaussian process emulator has hyper-parameters ψ and σ_η. These hyper-parameters are typically computed by maximizing the likelihood [Rasmussen and Williams 2006]. In Bayesian optimization, we can use Latin hypercube designs to select an initial set of parameters and then proceed to maximize the likelihood of the hyper-parameters iteratively [Ye 1998; Santner et al. 2003]. This is the approach followed in our experiments. However, a good alternative is to use either classical or Bayesian quadrature to integrate out the hyper-parameters [Osborne et al. 2010; Rue et al. 2009].

Let $\mathbf{z}_{1:i} \sim \mathcal{N}(0, \mathbf{K})$ be the i noisy observations of the objective function obtained from previous iterations. (Note that the Markov chain is run for L steps for each discrete iteration i. The extra index to indicate this fact has been made implicit to improve readability.) $\mathbf{z}_{1:i}$ and h_{i+1} are jointly multivariate Gaussian:

$$\begin{bmatrix} \mathbf{z}_{1:i} \\ h_{i+1} \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_\eta^2 I & \mathbf{k}^T \\ \mathbf{k} & k(\boldsymbol{\theta}, \boldsymbol{\theta}) \end{bmatrix}\right),$$

where

$$\mathbf{K} = \begin{bmatrix} k(\boldsymbol{\theta}_1, \boldsymbol{\theta}_1) & \ldots & k(\boldsymbol{\theta}_1, \boldsymbol{\theta}_i) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_1) & \ldots & k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) \end{bmatrix}$$

and $\mathbf{k} = [k(\boldsymbol{\theta}, \boldsymbol{\theta}_1) \ \ldots \ k(\boldsymbol{\theta}, \boldsymbol{\theta}_i)]^T$. All the above assumptions about the form of the prior distribution and observation model are standard and less restrictive than they might appear at first sight. The central assumption is that the objective function is smooth. For objective functions with discontinuities, we need more sophisticated surrogate functions for the cost. We refer readers to [Gramacy et al. 2004] and [Brochu et al. 2009] for examples.

The predictive distribution for any value $\boldsymbol{\theta}$ follows from the Sherman-Morrison-Woodbury formula, where $\mathcal{D}_{1:i} = (\boldsymbol{\theta}_{1:i}, \mathbf{z}_{1:i})$:

$$p(h_{i+1}|\mathcal{D}_{1:i}, \boldsymbol{\theta}) = \mathcal{N}(\mu_i(\boldsymbol{\theta}), \sigma_i^2(\boldsymbol{\theta}))$$
$$\mu_i(\boldsymbol{\theta}) = \mathbf{k}^T(\mathbf{K} + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{z}_{1:i}$$
$$\sigma_i^2(\boldsymbol{\theta}) = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \mathbf{k}^T(\mathbf{K} + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{k}$$

The next query point $\boldsymbol{\theta}_{i+1}$ is chosen to maximize an acquisition function, $u(\boldsymbol{\theta}|\mathcal{D}_{1:i})$, that trades-off exploration (where $\sigma_i^2(\boldsymbol{\theta})$ is large) and exploitation (where $\mu_i(\boldsymbol{\theta})$ is high). We adopt the expected improvement over the best candidate as this acquisition function [Močkus 1982; Schonlau et al. 1998; Brochu et al. 2009]:

$$\text{EI}(\boldsymbol{\theta}|\mathcal{D}_{1:i}) = \mathbb{E}\left(\max\left[0, h(\boldsymbol{\theta}) - h(\boldsymbol{\theta}_i^+)\right]|\mathcal{D}_{1:i}\right),$$

where $\boldsymbol{\theta}_i^+$ is the best known action so far. This is a standard acquisition function for which asymptotic rates of convergence have been proved [Bull 2011]. However, we point out that there are a few other reasonable alternatives, such as Thompson sampling [May et al. 2011] and upper confidence bounds (UCB) on regret [Srinivas et al. 2010]. A comparison among these options as well as portfolio strategies to combine them appeared recently in [Hoffman et al. 2011]. There are several good ways of optimizing the acquisition function, including the method of DIvided RECTangles (DIRECT) of [Finkel 2003] and many versions of the projected Newton methods of [Bertsekas 1982]. We found DIRECT to provide a very efficient solution in our domain. Note that optimizing the acquisition function is much easier than optimizing the original objective function. This is because the acquisition functions can be easily evaluated

k_l	k_u	γ_L	γ_H	P_{LL}	P_{HL}	P_{LH}	N
$\{1,\ldots,70\}$	$\{2,\ldots,120\}$	$[0.89, 1.05]$	$[0.9, 1.15]$	$[0,1]$	$[0,1]$	$[0,1]$	$\{1,\ldots,5\}$

Table I: The ranges from which Bayesian Optimization chooses parameters for SAR-DONICS. The selection mechanism ensures that $k_u \geq k_l$ and $\gamma_H \geq \gamma_L$.

and differentiated. Finally, we can apply appropriate normalization to ensure that the parameters remain within their permissible ranges, as discussed in the second paragraph of Section 5.

4.2. Sampling Phase

The Bayesian optimization phase results in a Gaussian process on the I noisy observations of the performance criterion $\mathbf{z}_{1:I}$, taken at the corresponding locations in parameter space $\boldsymbol{\theta}_{1:I}$. This Gaussian process is used to construct a discrete stochastic policy $p(\boldsymbol{\theta}|\mathbf{z}_{1:I})$ over the parameter space $\boldsymbol{\Theta}$. The Markov chain is run with parameter settings randomly drawn from this policy at each step.

One can synthesize the policy $p(\boldsymbol{\theta}|\mathbf{z}_{1:I})$ in several ways. The simplest is to use the mean of the GP to construct a distribution proportional to $\exp(\mu(\boldsymbol{\theta}))$. This is the so-called Boltzmann policy. We can sample M parameter candidates $\boldsymbol{\theta}_i$ according to this distribution. Our final sampler then consists of a mixture of M transition kernels, where each kernel is parameterized by one of the $\boldsymbol{\theta}_i$, $i = 1, \ldots, M$. The distribution of the samples generated in the sampling phase will approach the target distribution $\pi(\cdot)$ as the number of iterations tends to ∞ provided the kernels in this finite mixture are ergodic.

In high dimensions, one reasonable approach would be to use a multi-start optimizer to find maxima of the unnormalized Boltzmann policy and then perform local exploration of the modes with a simple Metropolis algorithm. This is a slight more sophisticated version of what is often referred to as the epsilon greedy policy.

The strategies discussed thus far do not take into account the uncertainty of the GP. A solution is to draw M functions according to the GP and then find he optimizer $\boldsymbol{\theta}_i$ of each of these functions. This is the strategy followed in [May et al. 2011] for the case of contextual bandits. Although this strategy works well for low dimensions, it is not clear how it can be easily scaled.

5. EXPERIMENTS

Our experiments compare different versions of SARDONICS to the popular Gibbs, block-Gibbs and Swendsen-Wang samplers. Several types of binary-valued systems, all belonging to the general class of undirected graphical model called the *Ising model*, are used. The energy of a binary state s, where $s_i \in \{-1, 1\}$ is given by:

$$E(\mathbf{s}) = -\sum_{(i,j)} J_{ij} s_i s_j - \sum_i h_i s_i$$

(One can trivially map $x_i \in \{0, 1\}$ to $s_i \in \{-1, 1\}$ and vice-versa.) The interaction weights J_{ij} between variables i and j are zero if they are topologically disconnected; positive (also called "ferromagnetic") if they tend to have the same value; and negative ("anti-ferromagnetic") if they tend to have opposite values. The presence of interactions of mixed sign can significantly complicate Monte Carlo simulation due to the proliferation of local minima in the energy landscape. Interaction weights of different sign produce unsatisfiable constraints and cause the system to become "frustrated".

Models	k_l	k_u	γ_L	γ_H	P_{LL}	P_{HL}	P_{LH}	N
2D Ferromagnetic model	55.8	91.2	1.002	1.029	0.385	0.371	0.243	1.6
Frustrated 2D grid Ising	63.3	91.1	1.040	1.082	0.208	0.674	0.119	3.4
Frustrated 3D cube Ising	4.4	30.1	1.039	1.134	0.616	0.231	0.153	2.9
Chimera lattice	3.8	19.8	1.043	1.139	0.397	0.324	0.279	3.9
RBM (Pure SARDONICS)	17.9	49.7	1.029	1.029	-	-	-	1.0

Table II: Typical parameters that the adaptation procedure discovers. These values are computed by taking an average over the mixture of M transition kernels generated by the adaptation process.

We apply SARDONICS with the *same* parameter ranges to a varied set of models, including 2D ferromagnets, 2D and 3D frustrated Ising models, restricted Boltzmann machines and chimera lattices. These parameter ranges are shown in Table I. We normalized the values of P_{LL}, P_{HL}, and P_{LH} to ensure they sum up to one. We enforce the constraint $k_u \geq k_l$ by choosing k_l and an additive value a_k such that $k_u = k_l + a_k$. The same procedure is applied to γ_H and γ_L. Table II shows some of the parameters computed by the Bayesian optimization scheme.

The experimental protocol for all models was the same: For 10 independent trials, run the competing samplers for a certain number of iterations, storing the sequence of energies visited. Using each trial's energy sequence, compute the *auto-correlation function* (ACF). Comparison of the algorithms consisted of analyzing the energy ACF averaged over the trials. Without going into detail, a more rapidly decreasing ACF is indicative of a faster-mixing Markov chain; see for example [Robert and Casella 2004]. For all the models, each sampler is run for 10^5 steps. For SARDONICS, we use the first 2×10^4 iterations to adapt its hyper-parameters. For fairness of comparison, we discard the first 2×10^4 samples from each sampler and compute the ACF on the remaining 8×10^4 samples.

5.1. Ferromagnetic 2D Ising Model

The first set of experiments considers the behavior of Gibbs, SARDONICS and Swendsen-Wang on a ferromagnetic Ising model on a planar, regular grid of size 60×60. The model has connections between the nodes on one boundary to the nodes on the other boundary for each dimension. As a result of these periodic boundaries, the model is a square toroidal grid. Hence, each node has exactly four neighbors. In the first experiment, the interaction weights, J_{ij}, are all 1 and the biases, h_i, are all 0. We test the three algorithms on this model at three different temperatures: 1, 2.27 and 5. The value $\beta = 1/2.27$ corresponds to the so-called *critical temperature*, where many interesting phenomena arise [Newman and Barkema 1999] but where simulation also becomes quite difficult.

As shown in Figure 2, at the critical temperature, Swendsen-Wang does considerably better than its competitors. It is precisely for these types of lattice that Swendsen-Wang was designed to mix efficiently, through effective clustering. This part of the result is therefore not surprising. However, we must consider the performance of SARDONICS carefully. Although it does much better than Gibbs, as expected, it underperforms in comparison to Swendsen-Wang at the critical temperature.

To examine SARDONICS with a finer lens, we consider the role of each of its components in Figure 3. The plot illustrates the performance of the vanilla SARDONICS algorithm (Pure SARDONICS), SARDONICS enhanced with the iterative scheme of Section 3.1, SARDONICS with the mixture of kernels, and SARDONICS with both the
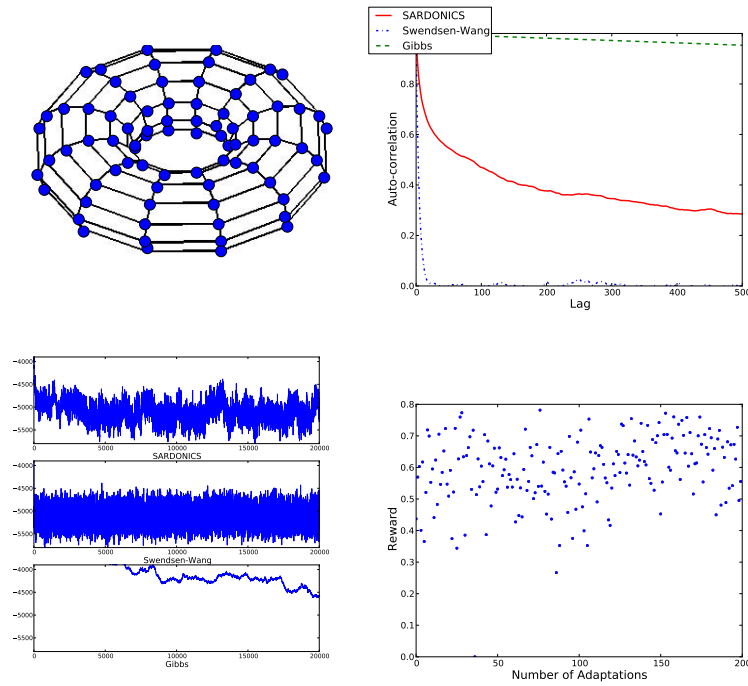
Fig. 2: The 2D Ferromagnetic model with periodic (toroidal) boundaries [top left], the auto-correlations of the samplers for $T = 2.27$ (critical temperature) [top right], traces of every 5 out of the 10^5 samples of the energy [bottom left] and rewards obtained by the Bayesian optimization algorithm during the adaptation phase [bottom right].
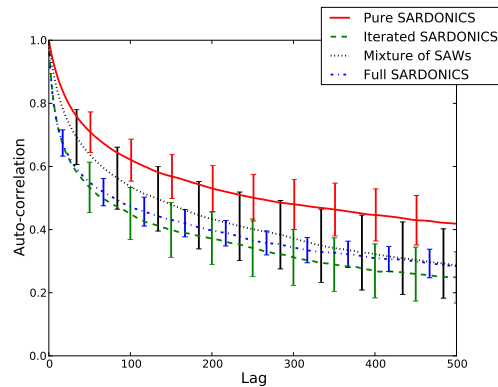

Fig. 3: Different versions of SARDONICS on the Ferromagnetic 2D grid Ising model .

iterative scheme and the mixture approach (Full SARDONICS). It is clear that the enhancements improve Pure SARDONICS, but that none of the approaches outperform Swendsen-Wang. We therefore focus our investigation on the role played by the temperature of the target distribution.
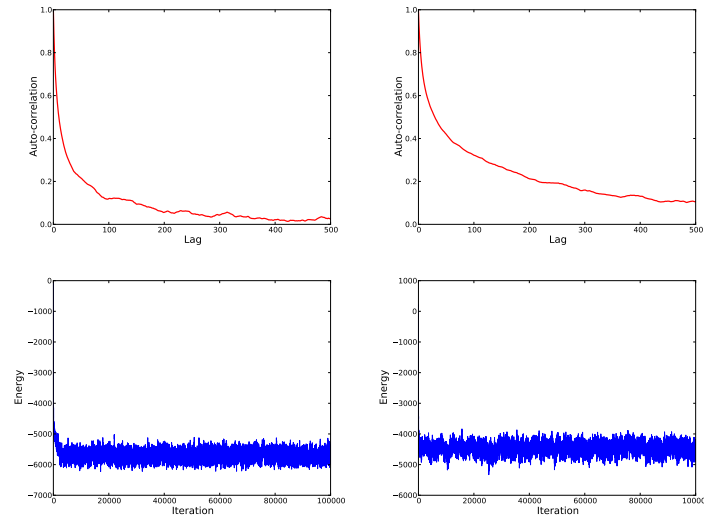
Fig. 4: Auto-correlations of the SARDONICS sampler, on the Ferromagnetic 2D Ising model, for $T = 2.17$ [top left] and $T = 2.37$ [top right]. Traces of the 10^5 samples of the energy at $T = 2.17$ [bottom left] and $T = 2.37$ [bottom right].
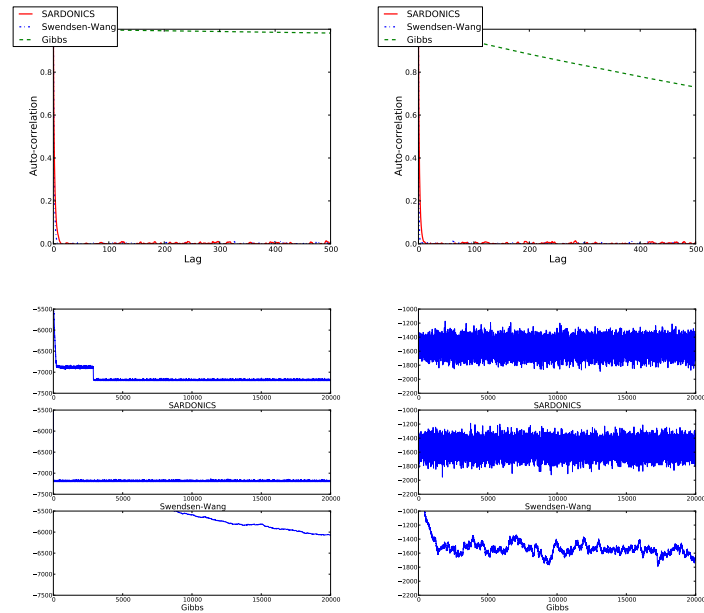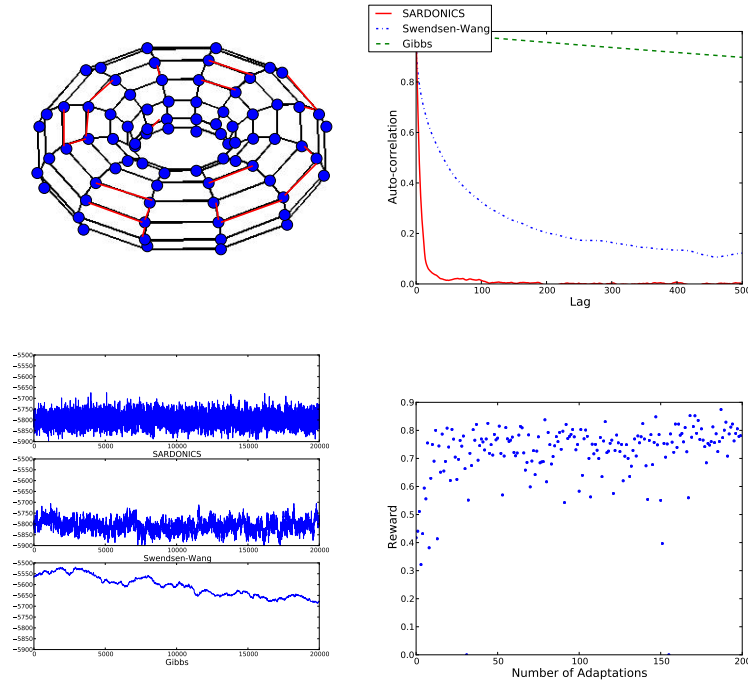

Fig. 5: Auto-correlations of the samplers, on the Ferromagnetic 2D Ising model, for $T = 1$ [top left] and $T = 5$ [top right]. Traces of every 5 out of the 10^5 samples of the energy at $T = 1$ [bottom left] and $T = 5$ [bottom right].

Fig. 6: Frustrated 2D Ising model with periodic boundaries [top left], auto-correlations of the three samplers [top right], traces of the last 20000 of 100000 samples of the energy [bottom left] and rewards obtained by the Bayesian optimization algorithm during the adaptation phase [bottom right].

Figure 4 examines the performance of SARDONICS around the critical temperature. The auto-correlation plots seem to indicate that at temperatures $T = 2.17$ and $T = 2.37$, SARDONICS mixes more rapidly than at the critical temperature $T = 2.27$. This is true for the higher temperature, but for the lower temperature the sampler explores a narrower range of energies.

To expand on this argument, we consider the performance of the sampler at temperatures $T = 1$ and $T = 5$. At temperature 5, when the distribution is flattened, the performance of SARDONICS is comparable to that of Swendsen-Wang as depicted in Figure 5. The same figure also shows the results for $T = 1$, where the target distribution has many narrow peaks with low probability valleys in between. Here, we see that the autocorrelations vanish quickly, but upon examining the traces of energies, we realize that what is actually happening is that the samplers, both SARDONICS and Swendsen-Wang, are trapped at a few of the peaks of the target distribution. At this near-deterministic setting, Swendsen-Wang and SARDONICS are better strategies for finding low-energy solutions. As such, they should be preferred to Gibbs within population Monte Carlo strategies aimed at exploring several of the peaks of the distribution.

In conclusion, for the ferromagnetic 2D Ising model, the adaptive SARDONICS method is competitive with Swendsen-Wang, except at the critical temperature.

5.2. Frustrated 2D and 3D Ising Models
In addition to studying the effect of temperature changes on the performance of the algorithms, we also investigate their sensitivity with respect to the addition of unsatis-
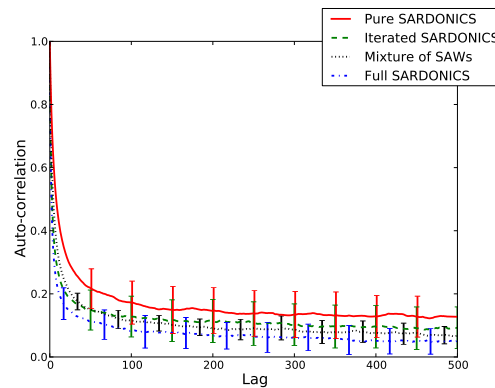
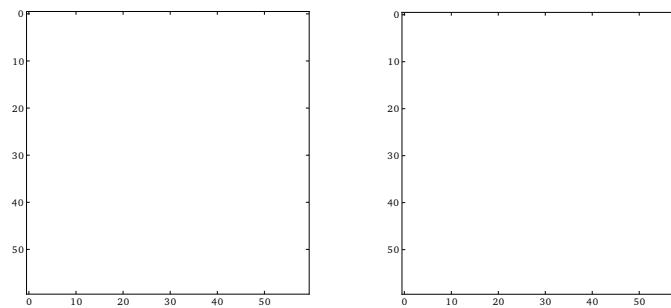Fig. 7: Different versions of SARDONICS on the Frustrated 3D cube Ising model.

Fig. 8: [Left] Bits flipped by SARDONICS in one step on the Frustrated 2D Ising model. Green represents unchanged bits, red represents bits changed from -1 to 1, and finally blue represents bits changed from 1 to -1. [Right] Bits flipped by the Swendsen-Wang algorithm in one step.

fiable constraints. To accomplish this, we set the interaction weights J_{ij} and the biases h_i uniformly at random on the set $\{-1, 1\}$. We set the temperature to $T = 1.0$. We refer to this model as the frustrated 2D grid Ising model. As shown by the auto-correlations and energy traces plotted in Figure 6, SARDONICS does considerably better than its rivals. It is interesting to note that Swendsen-Wang does much worse on this model as the unsatisfiable constraints hinder effective clustering. The figure also shows the reward obtained by the Bayesian optimization scheme as a function of the number of adaptations. The adaptation algorithm traded-off exploration and exploitation effectively in this case.

We analyze the contribution of each of the components of SARDONICS in Figure 7. Once again it is clear that each of the individual components leads to an increase in performance. Finally, Figure 8 shows the bits flipped in a typical step of SARDONICS and Swendsen-Wang for this frustrated model. This figure illustrates clearly that SARDONICS is able to make large moves in state space, with many variables being flipped simultaneously.

The next batch of experiments compares the algorithms on an Ising model where the variables are topologically structured as a $9 \times 9 \times 9$ three-dimensional cube, J_{ij} are
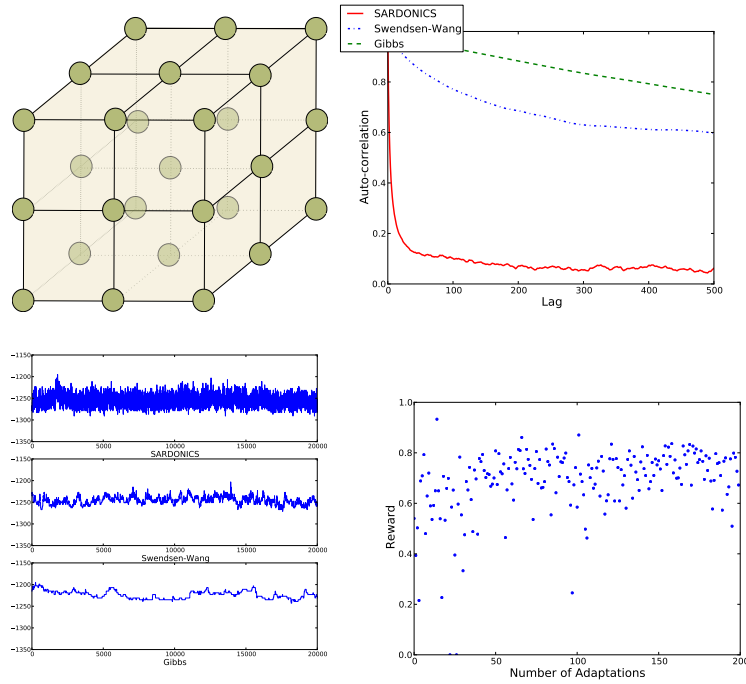
Fig. 9: Frustrated 3D cube Ising model with periodic boundaries (for visualization simplicity, the boundary edges are not shown) [top left], auto-correlations of the three samplers [top right], traces of the last 20000 of 100000 samples of the energy [bottom left] and rewards obtained by the Bayesian optimization algorithm during the adaptation phase [bottom right].

uniformly sampled from the set $\{-1, 1\}$, and the h_i are zero. β was set to 1.0, corresponding to a lower temperature than the value of 0.9, at which it is known [Marinari et al. 1997] that, roughly speaking, regions of the state space become very difficult to visit from one another via traditional Monte Carlo simulation. Figure 9 shows that for this more densely connected model, the performance of Swendsen-Wang deteriorates substantially. However, SARDONICS still mixes reasonably well.

5.3. Restricted Boltzmann Machine

While the three-dimensional-cube spin-glass is a much harder problem than the 2D ferromagnet, it represents a worst case scenario. One would hope that problems arising in practice will have structure in the potentials that would ease the problem of inference. For this reason, the third experimental set consisted of runs on a binary restricted Boltzmann machine [Smolensky 1986] with parameters trained from binary, natural image patches via stochastic maximum likelihood [Swersky et al. 2010; Marlin et al. 2010]. RBMs are bipartite undirected probabilistic graphical models. The variables on one side are often referred to as "visible units", while the others are called "hidden units". Each visible unit is connected to all hidden units. However there are no connections among the hidden units and among the visible units. Therefore, given the visible units, the hidden units are conditionally independent and vice-versa. Our model consisted of 784 visible and 500 hidden units. The model is illustrated in Figure 10.
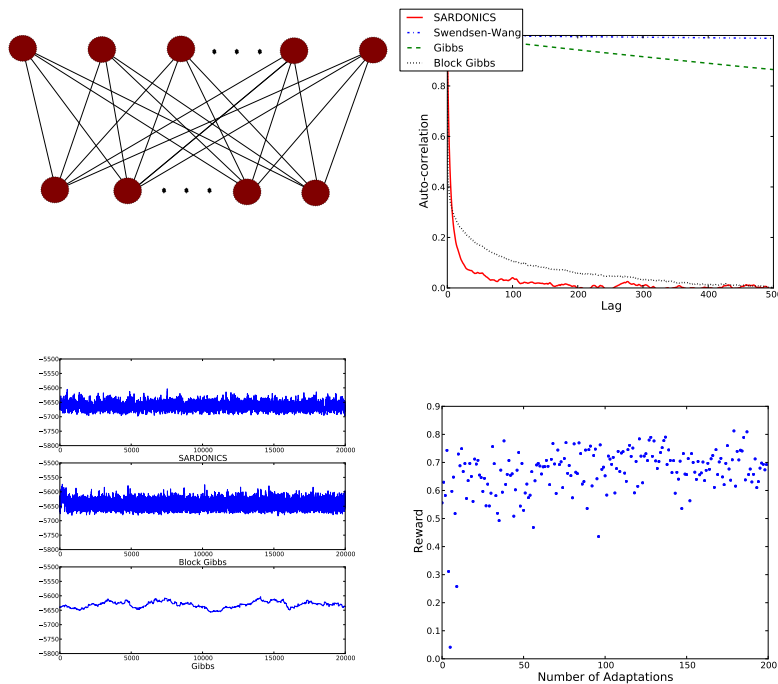
Fig. 10: RBM [top left], auto-correlations of the three samplers [top right], traces of the last 20000 of 100000 samples of the energy [bottom left] and rewards obtained by the Bayesian optimization algorithm during the adaptation phase [bottom right].
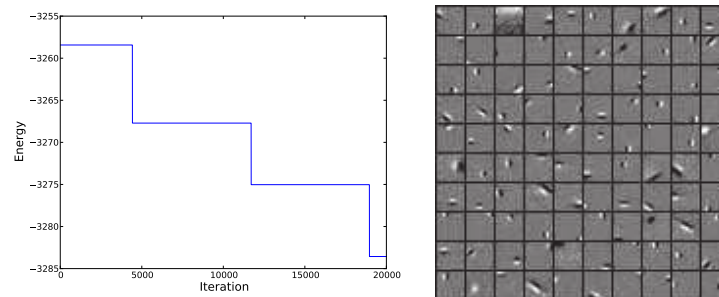

Fig. 11: [Left] Trace of the last 20000 of 100000 samples of the energy for the Swendsen-Wang sampler. As we can see, the sampler performs poorly on the RBM model. [Right] RBM parameters. Each image corresponds to the parameters connecting a specific hidden unit to the entire set of visible units.

The pre-learned interaction parameters capture local regularities of natural images [Hyvarinen et al. 2009]. Some of these parameters are depicted as images in Figure 11. The parameter β was set to one. The total number of variables and edges in the graph were thus 1284 and 392000 respectively.

Figures 10 and 11 show the results. Again SARDONICS mixes significantly better than Swendsen-Wang and the naive Gibbs sampler. Swendsen-Wang performs poorly
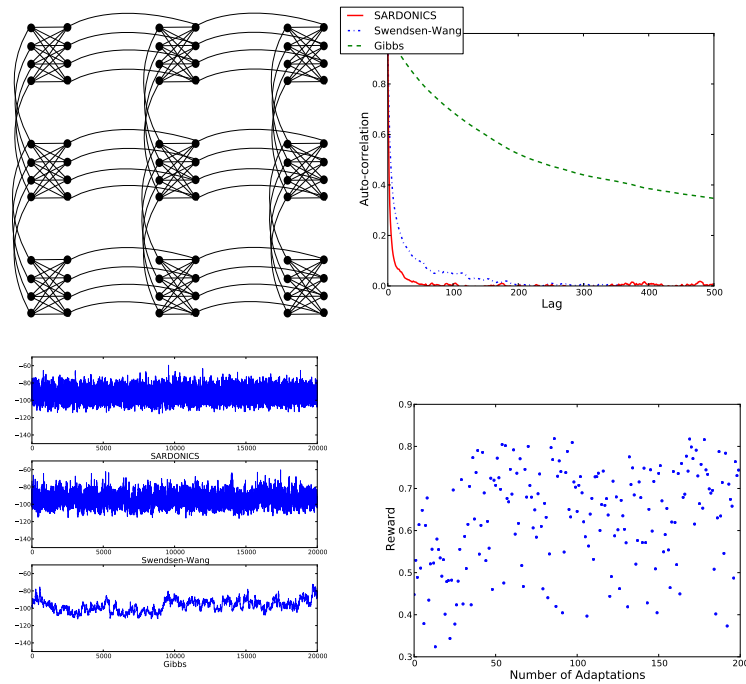
Fig. 12: Chimera lattice [top left], auto-correlations of the three samplers [top right], traces of the last 20000 of 100000 samples of the energy [bottom left] and rewards obtained by the Bayesian optimization algorithm during the adaptation phase [bottom right].

on this model. As shown in Figure 11, it mixes slowly and fails to converge after 10^5 iterations.

For this bipartite model, it is possible to carry out block-Gibbs sampling (the standard method of choice). Encouragingly, SARDONICS compares well against this popular block strategy. This is important, because computational neuro-scientists would like to add lateral connections among the hidden units, in which case block-Gibbs would no longer apply unless the connections form a tree structure. SARDONICS thus promises to empower computational neuro-scientists to address more sophisticated models of perception. The catch is that at present SARDONICS takes considerably more time than block-Gibbs sampling for these models. We discuss this issue in greater length in the following section.

5.4. Chimera Lattice

Finally, we consider a frustrated 128-bit chimera lattice that arises in the construction of quantum computers [Bian et al. 2011]. As depicted in Figure 12, SARDONICS once again outperforms its competitors.

5.5. Computational Considerations

The bulk of the computational time of the SARDONICS algorithm is spent in generating states with the SAW proposal. At each step of the process, a component from a discrete probability vector, corresponding to the variable to flip, must be sampled. Naively, the time needed to do so scales linearly with the length l of the vector. In

	Samplers				
Models	Pure SARDONICS	Full SARDONICS	Swendsen-Wang	Gibbs	Block Gibbs
Ferromagnetic Ising Model	8 minutes	20 minutes	50 minutes	tens of seconds	N/A
Frustrated 2D Ising Model	7 minutes	20 minutes	50 minutes	tens of seconds	N/A
Frustrated 3D Ising Model	3 minutes	10 minutes	few minutes	tens of seconds	N/A
RBM	3 hours	10 hours	few hours	20 minutes	few minutes
Chimera	2 minutes	10 minutes	tens of seconds	tens of seconds	N/A

Table III: Rough computation time for each sampler on different models. All samplers are run on the same computer with 8 CPUs. The Swendsen-Wang sampler is coded in Matlab with the computationally intensive part written in C. The SARDONICS sampler is coded in Python also with its computationally intensive part coded in C. The Gibbs and block Gibbs samplers are coded in Python. The Swendsen-Wang and block Gibbs samplers take advantage of parallelism via parallel numerical linear algebra operations. The SARDONICS sampler and the Gibbs sampler, however, run on a single CPU. The adaptation time of SARDONICS is also included.

graphical models of sparse connectivity, however, it is possible achieve a dramatic computational speedup by storing the vector in a *binary heap*. Sampling from a heap is of $O(\log l)$, but for *sparsely connected* models, updating the heap in response to a flip, which entails replacing the energy changes that would result if the flipped variable's *neighboring* variables were themselves to flip, is also of logarithmic complexity. In contrast, for a densely connected model, the heap update would be of $O(M \log l)$ where M is the maximum degree of the graph, while recomputing the probability vector in the naive method is $O(M)$. The simple method is thus cheaper for dense models. In our experiments, we implemented SARDONICS with the binary heap despite the fact that the RBM is a densely connected model.

For densely connected models, one could easily parallelize the computation of generating states with the SAW proposal. Suppose we have n parallel processes. Each process P holds one section U_p of the unnormalized probability vector U. To sample from the discrete vector in parallel, each parallel process can sample one variable v_p to flip according to U_p. Each process also calculates the sum of its section of the unnormalized probability vector s_p. Then the variable to flip is sampled from the set $\{v_p : 1 \le p \le n\}$ proportional to the discrete probability vector $[s_1, s_2, ..., s_n]$. To update U in response to a flip, we could use the naive method mentioned above. If we divide the U evenly among processes and assume equal processor speed, sampling from and updating the unnormalized probability vector would take $O(\frac{M+l}{n} + n)$ operations. When n is small compared to $M + l$, we could achieve near linear speed up.

We compare, in table III, the amount of time it takes to draw 10^5 samples using different samplers. In principle, it is unwise to compare the computational time of samplers when this comparison depends on specific implementation details and architecture. The table simply provides rough estimates. The astute reader might be thinking that we could run Gibbs for the same time as SARDONICS and expect similar performance. This is however not what happens. We tested this hypothesis and found Gibbs to still underperform. Moreover, in some cases, Gibbs can get trapped as a result of being a small move sampler.

The computational complexity of SARDONICS is affected by the degree of the graph, whereas the complexity of Swendsen-Wang increases with the number of edges in the

graph. So for large sparsely-connected graphs, we expect SARDONICS to have a competing edge over Swendsen-Wang, as already demonstrated to some extent in the experiments.

6. DISCUSSION

The results have shown that the proposed sampler mixes very well in a broad range of complex stochastic models. Although we are forced to choose the ranges of permissible parameters a priori, the experiments illustrate that using identical parameter ranges one can a attack a large an varied set of models in a fully automatic way.

However, we believe that we need to consider alternatives to nonparametric methods in Bayesian optimization. Parametric methods would enable us to carry out infinite adaptation.

Although the proposed technique applies to arbitrary binary systems, we only tested it on models with pair-wise interactions. It would interesting to investigate its performance in models with arbitrary potentials, such as the ones that arise in Bayesian variable selection and counting SAT problems. Another important avenue for future work is to develop a multi-core implementation of SARDONICS as outlined in the previous section. Presently, the SARDONICS software is available at `https://github.com/ziyuw/AdaptSAW`.

Finally, it should be possible to extend SARDONICS from binary systems to multinomial valued systems in an efficient way.

REFERENCES

ACKLEY, D. H., HINTON, G., AND SEJNOWSKI, T. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science 9*, 147–169.

ANDRIEU, C. AND ROBERT, C. 2001. Controlled MCMC for optimal sampling. Tech. Rep. 0125, Cahiers de Mathematiques du Ceremade, Universite Paris-Dauphine.

BARAHONA, F. 1982. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General 15,* 10, 3241.

BERG, B. A. AND NEUHAUS, T. 1991. Multicanonical algorithms for first order phase transitions. *Physics Letters B 267,* 2, 249 – 253.

BERTSEKAS, D. P. 1982. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization 20,* 2, 221–246.

BESAG, J. 1974. Spatial interaction and the statistical analysis of lattice systems. *J. Roy. Statist. Soc., Ser. B 36*, 192–236.

BIAN, Z., CHUDAK, F., MACREADY, W. G., AND ROSE, G. 2011. The Ising model: teaching an old problem new tricks. Tech. rep., D-Wave Systems.

BLEI, D. M., NG, A. Y., AND JORDAN, M. I. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res. 3*, 993–1022.

BOTEV, Z. I. AND KROESE, D. P. 2012. Efficient Monte Carlo simulation via the generalized splitting method. *Statistics and Computing 22,* 1, 1–16.

BROCHU, E., CORA, V. M., AND DE FREITAS, N. 2009. A tutorial on Bayesian optimization of expensive cost functions. Tech. Rep. TR-2009-023, University of British Columbia, Department of Computer Science.

BULL, A. D. 2011. Convergence rates of efficient global optimization algorithms. Tech. Rep. arXiv:1101.3501v2.

CÉROU, F., MORAL, P., FURON, T., AND GUYADER, A. 2012. Sequential Monte Carlo for rare event estimation. *Statistics and Computing 22,* 3, 795–808.

DIGGLE, P. J., TAWN, J. A., AND MOYEED, R. A. 1998. Model-based geostatistics. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 47,* 3, 299–350.

DUANE, S., KENNEDY, A. D., PENDLETON, B. J., AND ROWETH, D. 1987. Hybrid Monte Carlo. *Physics Letters B 195,* 2, 216–222.

EARL, D. J. AND DEEM, M. W. 2005. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.* 7, 3910–3916.

FINKEL, D. E. 2003. *DIRECT Optimization Algorithm User Guide*. Center for Research in Scientific Computation, North Carolina State University.

FREITAS, N. D., WANG, Y., MAHDAVIANI, M., AND LANG, D. 2005. Fast Krylov methods for N-body learning. In *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, 251–258.

GEYER, C. J. 1991. Markov Chain Monte Carlo maximum likelihood. In *Computer Science and Statistics: 23rd Symposium on the Interface*. 156–163.

GIROLAMI, M. AND CALDERHEAD, B. 2011. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73,* 2, 123–214.

GLOVER, F. 1989. Tabu search – Part I. *ORSA Journal on Computing 1*, 190–206.

GRAMACY, R. B., LEE, H. K. H., AND MACREADY, W. 2004. Parameter space exploration with gaussian process trees. In *Proceedings of the International Conference on Machine Learning*. Omnipress and ACM Digital Library, 353–360.

GUBERNATIS, J. AND HATANO, N. 2000. The multicanonical monte carlo method. *Computing in Science Engineering 2,* 2, 95–102.

HAARIO, H., SAKSMAN, E., AND TAMMINEN, J. 2001. An adaptive Metropolis algorithm. *Bernoulli 7,* 2, 223–242.

HALKO, N., MARTINSSON, P. G., , AND TROPP, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *Science 53,* 2, 217–288.

HAMZE, F. AND DE FREITAS, N. 2004. From fields to trees. In *Uncertainty in Artificial Intelligence*. 243–250.

HAMZE, F. AND DE FREITAS, N. 2005. Hot Coupling: a particle approach to inference and normalization on pairwise undirected graphs. *Advances in Neural Information Processing Systems 18*, 491–498.

HAMZE, F. AND DE FREITAS, N. 2007. Large-flip importance sampling. In *Uncertainty in Artificial Intelligence*. 167–174.

HAMZE, F. AND DE FREITAS, N. 2010. Intracluster moves for constrained discrete-space MCMC. In *Uncertainty in Artificial Intelligence*. 236–243.

HINTON, G. AND SALAKHUTDINOV, R. 2006. Reducing the dimensionality of data with neural networks. *Science 313,* 5786, 504–507.

HOFFMAN, M., BROCHU, E., AND DE FREITAS, N. 2011. Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence*. 327–336.

HOOS, H. H. AND STUTZLE, T. 2004. *Stochastic Local Search: Foundations and Applications*. Elsevier, Morgan Kaufmann.

HOPFIELD, J. J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences 81,* 10, 3088–3092.

HYVARINEN, A., HURRI, J., AND HOYER, P. 2009. *Natural Image Statistics*. Springer.

KINDERMANN, R. AND SNELL, J. L. 1980. *Markov Random Fields and their Applications*. Amer. Math. Soc.

KÜCK, H. AND DE FREITAS, N. 2005. Learning about individuals from group statistics. In *Uncertainty in Artificial Intelligence*. 332–339.

LEE, H., GROSSE, R., RANGANATH, R., AND NG, A. Y. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*. 609–616.

LIU, J. S. 2001. *Monte Carlo strategies in scientific computing*. Springer.

LIU, J. S., ZHANG, J. L., PALUMBO, M. J., AND LAWRENCE, C. E. 2003. Bayesian clustering with variable and transformation selections. *Bayesian Statistics 7*, 249–275.

LIZOTTE, D. 2008. Practical Bayesian optimization. Ph.D. thesis, University of Alberta, Edmonton, Alberta, Canada.

MAHENDRAN, N., WANG, Z., HAMZE, F., AND DE FREITAS, N. 2011. Bayesian optimization for adaptive MCMC. Tech. Rep. arXiv:1110.6497v1.

MARINARI, E., PARISI, G., AND RUIZ-LORENZO, J. 1997. Numerical simulations of spin glass systems. *Spin Glasses and Random Fields*, 59–98.

MARLIN, B., SWERSKY, K., CHEN, B., AND DE FREITAS, N. 2010. Inductive principles for restricted Boltzmann machine learning. In *Artificial Intelligence and Statistics*. 509–516.

MAY, B. C., KORDA, N., LEE, A., AND LESLIE, D. S. 2011. Optimistic Bayesian sampling in contextual bandit problems.

MEMISEVIC, R. AND HINTON, G. 2009. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation 22*, 1473–1492.

MOČKUS, J. 1982. The Bayesian approach to global optimization. In *System Modeling and Optimization*. Vol. 38. Springer Berlin / Heidelberg, 473–481.

MUNOZ, J. D., NOVOTNY, M. A., AND MITCHELL, S. J. 2003. Rejection-free Monte Carlo algorithms for models with continuous degrees of freedom. *Phys. Rev. E 67*.

NEAL, R. M. 2010. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo 54*, 113–162.

NEWMAN, M. AND BARKEMA, G. 1999. *Monte Carlo Methods in Statistical Physics*. Oxford University Press.

OSBORNE, M., GARNETT, R., AND ROBERTS, S. 2010. Active data selection for sensor networks with faults and changepoints. In *IEEE International Conference on Advanced Information Networking and Applications*.

RANZATO, M., MNIH, V., AND HINTON, G. 2010. How to generate realistic images using gated MRF's. In *Advances in Neural Information Processing Systems*. 2002–2010.

RASMUSSEN, C. E. AND WILLIAMS, C. K. I. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts.

ROBERT, C. P. AND CASELLA, G. 2004. *Monte Carlo Statistical Methods* 2nd Ed. Springer.

ROBERTS, G. O. AND ROSENTHAL, J. S. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics 18,* 2, 349–367.

ROSENBLUTH, M. N. AND ROSENBLUTH, A. W. 1955. Monte Carlo calculation of the average extension of molecular chains. *J. Chem. Phys. 23*, 356–359.

RUE, H., MARTINO, S., AND CHOPIN, N. 2009. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal Of The Royal Statistical Society Series B 71,* 2, 319–392.

SALAKHUTDINOV, R. AND MURRAY, I. 2008. On the quantitative analysis of Deep Belief Networks. In *International Conference on Machine Learning*. 872–879.

SANTNER, T. J., WILLIAMS, B., AND NOTZ, W. 2003. *The Design and Analysis of Computer Experiments*. Springer.

SCHONLAU, M., WELCH, W. J., AND JONES, D. R. 1998. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series 34*, 11–25.

SIEPMANN, J. I. AND FRENKEL, D. 1992. Configurational bias Monte Carlo: a new sampling scheme for flexible chains. *Molecular Physics: An International Journal at the Interface Between Chemistry and Physics 75,* 1, 59–70.

SMOLENSKY, P. 1986. Information processing in dynamical systems: Foundations of harmony theory. *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: foundations*, 194–281.

SRINIVAS, N., KRAUSE, A., KAKADE, S. M., AND SEEGER, M. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*.

SWENDSEN, R. H. AND WANG, J.-S. 1987. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett. 58,* 2, 86–88.

SWERSKY, K., CHEN, B., MARLIN, B., AND DE FREITAS, N. 2010. A tutorial on stochastic approximation algorithms for training restricted Boltzmann machines and deep belief nets. In *Information Theory and Applications Workshop*. 1 –10.

THAM, S. S., DOUCET, A., AND KOTAGIRI, R. 2002. Sparse Bayesian learning for regression and classification using Markov chain Monte Carlo. In *International Conference on Machine Learning*. 634–641.

VIHOLA, M. 2010. Grapham: Graphical models with adaptive random walk Metropolis algorithms. *Computational Statistics and Data Analysis 54,* 1, 49 – 54.

WANG, F. AND LANDAU, D. P. 2001. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett. 86*, 2050–2053.

WELSH, D. J. A. 1993. *Complexity: knots, colourings and counting*. Cambridge University Press.

YE, K. Q. 1998. Orthogonal column Latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association 93,* 444, 1430–1439.