Abstract. Sampling from complex distributions is an important but challenging topic in scientific and statistical computation. We synthesize three ideas, tempering, resampling, and Markov moving, and propose a general framework of resampling Markov chain Monte Carlo (MCMC). This framework not only accommodates various existing algorithms, including resample-move, importance resampling MCMC, and equi-energy sampling, but also leads to a generalized resample-move algorithm. We provide some basic analysis of these algorithms within the general framework, and present three simulation studies to compare these algorithms together with parallel tempering in the difficult situation where new modes emerge in the tails of previous tempering distributions. Our analysis and empirical results suggest that generalized resample-move tends to perform the best among all the algorithms studied when the Markov kernels lead to fast mixing or even locally so toward restricted distributions, whereas parallel tempering tends to perform the best when the Markov kernels lead to slow mixing, without even converging fast to restricted distributions. Moreover, importance resampling MCMC and equi-energy sampling perform similarly to each other, often worse than independence Metropolis resampling MCMC. Therefore, different algorithms seem to have advantages in different settings.

Key words and phrases. Equi-energy sampling; Importance resampling; Markov chain Monte Carlo; Parallel tempering; Potts model; Resample-move; Sequential Monte Carlo.
1 Introduction

Sampling from complex distributions is an important but challenging topic in scientific and statistical computation (e.g., Liu 2001; Robert & Casella 2001). There have been several classes of Monte Carlo techniques introduced. The class of Markov chain Monte Carlo (MCMC) techniques include Metropolis-Hasting sampling, Gibbs sampling, and various extensions, for example, parallel tempering (Geyer 1991). The class of sequential Monte Carlo (SMC) techniques include sequential importance resampling or particle filtering (Gordon et al. 1993; Kong et al. 1994), resample-move (Gilks & Berzuini 2001), and various extensions (Liu & Chen 1998; Del Moral et al. 2006). Recent developments of Monte Carlo techniques include equi-energy sampling (Kou et al. 2006), importance resampling MCMC (Atchadé 2009), and self-interacting MCMC (Andrieu et al. 2011).

The foregoing review of Monte Carlo techniques is far from complete, but intended to highlight several basic ideas shared by these techniques. The first idea, tempering, is broadly to consider a sequence of distributions $P_1, \ldots, P_m$ with unnormalized densities $q_1(x), \ldots, q_m(x)$ on the same sample space. The classical example is a sequence of Boltzmann distributions with $q_j(x) = \exp\{-u(x)/T_j\}$, where $u(x)$ is a potential function and $T_j$ is a temperature. There are at least two distinct situations. For one situation, all the $m$ distributions are of interest, as in the example of Potts models in Section 3.3. For another situation, only $P_m$ is of interest and the rest are constructed to facilitate sampling from $P_m$, as in the example of Gaussian mixtures in Section 3.2. In the latter case, it is typically expected that $P_1, \ldots, P_m$ are increasingly more difficult to simulate. But the levels of difficulty in sampling from distributions are usually complicated to categorize. Moreover, this condition is not necessarily required for the validity of algorithms discussed later, although violation of this condition can have considerable impacts on the performances of algorithms.

The second idea, resampling, is to randomly select elements with replacement from a sample for $P_{j-1}$ as a way to generate a sample for $P_j$. Importance resampling (Rubin 1987) is central to SMC techniques, where selection probabilities are proportional to $q_j(x)/q_{j-1}(x)$ and the selected elements are then automatically accepted except in some extensions of SMC, for example, rejection control by Liu et al. (1998). Alterna-
tively, as discussed in Section 2.3.2, independence Metropolis resampling is essentially used in equi-energy sampling (Kou et al. 2006) and a modified version by Atchadé (2010), where selection probabilities are uniform but the selected elements are either accepted or rejected according to the rejection rule in independence Metropolis sampling (e.g., Liu 2001, Section 5.4.2).

The third idea, Markov moving, is, given an initial observation, to generate new observations for $P_j$ using a Markov kernel that leaves $P_j$ invariant. This exercise may lead directly to MCMC. But the use of Markov moving in the resample-move algorithm (Gilks & Berzuini 2001) is different, where one Markov move is performed on each element in an intermediate sample, which is obtained for $P_j$ by importance resampling. The effect of Markov moving is twofold. First, the new sample remains to be approximately distributed according to $P_j$ by the invariance of $P_j$ under the Markov kernel. Second, an observation may be selected into the intermediate sample for several times, but the copies of this observation can then be moved to different points. Therefore, more diversity is introduced into the new sample.

In this article, we synthesize these ideas and propose a general framework of resampling MCMC algorithms. This framework accommodates various existing algorithms, including resample-move, importance resampling MCMC, and equi-energy sampling, although not parallel tempering. We also propose a generalized resample-move algorithm by performing and retaining multiple Markov moves given each draw from importance resampling, while reducing the number of such draws. We provide some basic analysis of these algorithms within the general framework, which sheds new light on their connections and comparisons. Finally, we present three simulation studies to compare various algorithms including parallel tempering.

## 2 Resampling MCMC

### 2.1 A general framework

Let $P_1, \ldots, P_m$ be a sequence of distributions and $\Psi_1, \ldots, \Psi_m$ be the corresponding Markov transition kernels such that $P_j$ is invariant under $\Psi_j$ for $j = 1, \ldots, m$. Pick an initial value $x_{j0}$ for sampling from $P_j$ for $j = 1, \ldots, m$. We propose the following framework of algorithms, called resampling MCMC, to generate samples $S_1, \ldots, S_m$
Table 1: Intermediate observations from $S_{j-1}$ to $S_j$

| $S_{j-1}$ | $R_j$ | $S_j$ |
|-----------|-------|-------|
| $x_{j-1,1}$ | $z_{j1}$ | $y_{j1}$ | $(y_{j1;1}, \ldots, y_{j1;\ell_{j1}})$ |
| \vdots | \vdots | \vdots | \vdots |
| $x_{j-1,i}$ | $z_{ji}$ | $y_{ji}$ | $(y_{ji;1}, \ldots, y_{ji;\ell_{ji}})$ |
| \vdots | \vdots | \vdots | \vdots |
| $x_{j-1,n_{j-1}}$ | $z_{jk_j}$ | $y_{jk_j}$ | $(y_{jk_j;1}, \ldots, y_{jk_j;\ell_{jk_j}})$ |

of sizes $n_1, \ldots, n_m$ for $P_1, \ldots, P_m$ respectively.

Resampling MCMC:

1. Generate $S_1 = \{x_{11}, \ldots, x_{1n_1}\}$ using $\Psi_1$ with the initial value $x_{10}$ for $P_1$.
2. For $j = 2, 3, \ldots, m$,
   2a. Generate a positive integer $k_j (\leq n_j)$ and then $k_j$ positive integers $\{\ell_{ji} : i = 1, \ldots, k_j\}$ such that $\sum_{i=1}^{k_j} \ell_{ji} = n_j$.
   2b. Draw $k_j$ observations $\{z_{j1}, \ldots, z_{jk_j}\}$ with replacement from the sample $S_{j-1} = \{x_{j-1,1}, \ldots, x_{j-1,n_{j-1}}\}$.
   2c. For $i = 1, \ldots, k_j$,
      accept $y_{ji} = z_{ji}$ or reject $z_{ji}$ and let $y_{ji} = x_{j0}$ if $i = 1$ or $y_{ji-1,\ell_{ji-1}}$ if $i \geq 2$ by a rejection rule, and then generate a chain of $\ell_{ji}$ observations $\{y_{ji;1}, \ldots, y_{ji;\ell_{ji}}\}$ using the Markov kernel $\Psi_j$ with the initial value $y_{ji}$.
   2d. Define a sample $S_j = \{x_{j1}, \ldots, x_{jn_j}\}$ for $P_j$ as the collection of $\{y_{ji;1}, \ldots, y_{ji;\ell_{ji}}\}$ over $i = 1, \ldots, k_j$.

Table 1 illustrates the relationship between different samples. The general algorithm involves two main operations, resampling (including acceptance-rejection) and Markov moving. As discussed in Introduction, resampling is used to build an intermediate sample $R_j = \{y_{j1}, \ldots, y_{jk_j}\}$ from $S_{j-1}$ and then Markov moving is used to rejuvenate the intermediate sample $R_j$ and obtain a final sample $S_j$.

It is helpful to see heuristically why resampling MCMC can be justified before delving into specific algorithms. The basic idea is an induction argument, which can be found in the formal proofs of asymptotic results on various specific algorithms, for example, Chopin (2004) on the resample-move algorithm, Kou et al. (2006) on equi-energy sampling, and Atchadé (2010) on importance resampling and independence Metropolis resampling MCMC. First, $P_1$ is usually constructed such that $S_1$ can
be directly obtained either by independent sampling or by MCMC with fast mixing. Suppose that $S_{j-1}$ is a valid sample from $P_{j-1}$ for a certain $j \geq 2$. The resampling operation is designed, possibly with acceptance-rejection, such that the intermediate sample $R_j$ becomes a valid sample from $P_j$. Then, by Markov moving that leaves $P_j$ invariant, the final sample $S_j$ remains valid for $P_j$.

Even given the sequence of distributions $P_1, \ldots, P_m$ and Markov kernels $\Psi_1, \ldots, \Psi_m$, there are a number of factors left to be specified in the general framework: how $S_1$ is generated in step (1), how $k_j$ and $\ell_{ji}$ are generated in step (2a), what resampling scheme is used in step (2b), what rejection rule is used in step (2c). Different choices of these factors lead to even different classes of algorithms, including but not restricted to various existing algorithms mentioned in Introduction.

At one extreme, generating $S_1$ by MCMC, taking $k_j = 1$ and $\ell_{ji} = n_j$, and always rejecting $z_{ji}$ and letting $y_{ji} = x_{j0}$ leads to standard MCMC algorithms for sampling from $P_1, \ldots, P_m$ separately. No resampling is done. At the other extreme, sequential importance resampling (Gordon et al. 1993; Kong et al. 1994) corresponds to generating $S_1$ by independent sampling (made possible by the choice of $P_1$), taking $k_j = n_j$ and $\ell_{ji} = 1$, using importance resampling in step (2b), and always accepting $y_{ji} = z_{ji}$ but skipping Markov moving so that $y_{ji:1} = z_{ji}$ in step (2c). The resample-move algorithm (Gilks & Berzuini 2001) is obtained in the same way as sequential importance resampling, except that Markov moving is indeed performed in step (2c). Therefore, the general framework of resampling MCMC accommodates both standard MCMC and SMC algorithms as special cases.

2.2 Generalized resample-move

The foregoing discussion of the resample-move algorithm suggests an interesting extension in the general framework. Let $\ell_j$ be a divisor of $n_j$. Instead of drawing a sample of size $n_j$ from $S_{j-1}$ and performing one Markov move to obtain $S_j$, it is possible to draw a subsample of size $k_j = n_j/\ell_j$ from $S_{j-1}$ and then perform $\ell_j$ successive Markov moves and retain all the $\ell_j$ observations in $S_j$. The resulting algorithm, called generalized resample-move, is as follows. Only the initial value $x_{10}$ for $P_1$ is used and the other values $x_{20}, \ldots, x_{m0}$ are ignored.
Generalized resample-move:

(1) Generate $S_1 = \{x_{11}, \ldots, x_{1n_1}\}$ using $\Psi_1$ with the initial value $x_{10}$ for $P_1$.

(2) For $j = 2, 3, \ldots, m$,

(2a) Take $\ell_j$ as a divisor of $n_j$.

(2b) Draw $k_j = n_j/\ell_j$ observations $R_j = \{y_{j1}, \ldots, y_{jk_j}\}$ with replacement by importance resampling from the sample $S_{j-1}$.

(2c) For $i = 1, \ldots, k_j$, generate a chain of $\ell_j$ observations $\{y_{jic}, \ldots, y_{jic\ell_j}\}$ using the kernel $\Psi_j$ with the initial value $y_{ji}$.

(2d) Define a sample $S_j = \{x_{j1}, \ldots, x_{jn_j}\}$ for $P_j$ as the collection of $\{y_{jic}, \ldots, y_{jic\ell_j}\}$ over $i = 1, \ldots, k_j$.

The performance of the algorithm depends on the choices of $\ell_j$, the distances between distributions $P_{j-1}$ and $P_j$, and mixing of Markov chains with kernels $\Psi_j$. To make clear the effects of these factors, we provide a general asymptotic result for two successive distributions $P_{j-1}$ and $P_j$, which are represented below by $P_1$ and $P_2$. First, we give a definition, formalizing the condition that a sample can consist of dependent observations such as a Markov chain or a bootstrap sample or in other complicated manner, but some types of laws of large numbers and central limit theorems are valid for sample averages of functions of the observations.

**Definition 1.** Let $S_1 = \{x_1, \ldots, x_{n_1}\}$ be a sample from $P_1$. Say that $S_1$ is an asymptotically regular sample if for any function $\varphi(\cdot)$,

$$E(\bar{\varphi}_1) = E_1(\varphi) + O\left(\frac{1}{n_1}\right),$$

$$\text{var}(\bar{\varphi}_1) = \frac{1}{n_1}v_1(\varphi) + o\left(\frac{1}{n_1}\right),$$

where $\bar{\varphi}_1 = n_1^{-1}\sum_{i=1}^{n_1} \varphi(x_i)$, $E_1(\varphi)$ is the expectation of $\varphi(x)$ for $x$ drawn from $P_1$, and $n_1^{-1}v_1(\varphi)$ is the $O(n_1^{-1})$ term in the asymptotic variance of $\bar{\varphi}_1$. If $S_1$ is an independent and identically distributed sample, then $v_1(\varphi) = V_1(\varphi)$, the variance of $\varphi(x)$ for $x$ drawn from $P_1$. In general, it is expected that $v_1(\varphi) \geq V_1(\varphi)$.

**Proposition 1.** Let $S_1 = \{x_1, \ldots, x_{n_1}\}$ be an asymptotically regular sample from $P_1$, and $R_2 = \{y_1, \ldots, y_k\}$ be a sample of size $k = n_2/\ell$ for $P_2$ obtained by importance resampling from $S_1$. Let $r(\cdot) = p_2(\cdot)/p_1(\cdot)$, where $p_1(\cdot)$ and $p_2(\cdot)$ are the normalized densities for $P_1$ and $P_2$. For $i = 1, \ldots, k$, let $\{y_{ic1}, \ldots, y_{ic\ell}\}$ be a chain of $\ell$ observations
obtained using the kernel $\Psi_2$ with the initial value $y_i$. Then for any function $g(\cdot)$ under suitable conditions, as $n_1 \to \infty$, $n_2 \to \infty$, and either $\ell$ or $k$ being bounded,

$$E(\bar{g}_2) = E_2(g) + O\left(\frac{1}{n_1} + \frac{1}{n_2}\right),$$

$$\text{var}(\bar{g}_2) = \frac{1}{n_1} v_1 \left[ r \{ h_\ell - E_2(g) \} \right] + \frac{1}{k} \text{var}_{g_0 \sim P_2} \left\{ \frac{1}{\ell} \sum_{j=1}^{\ell} g(y_{0,j}) \right\} + o\left(\frac{1}{n_1} + \frac{1}{n_2}\right),$$

(1)

where $\bar{g}_2 = n_2^{-1} \sum_{i=1}^{k} \sum_{j=1}^{\ell} g(y_{i,j})$, $E_2(g)$ is the expectation of $g(x)$ for $x$ drawn from $P_2$, $\{y_{0,1}, \ldots, y_{0,\ell}\}$ are a chain of observations obtained using the kernel $\Psi_2$ with an initial value $y_0$, $h_\ell(\cdot) = E\{\ell^{-1} \sum_{j=1}^{\ell} g(y_{0,j})|y_0 = \cdot\}$, and $\text{var}_{g_0 \sim P_2}(\cdot)$ denotes the variance with $y_0$ drawn from $P_2$.

The remainder terms in $E(\bar{g}_2)$ and $\text{var}(\bar{g}_2)$ are provided to accommodate both the case of $\ell$ bounded and $k \to \infty$ and the case of $\ell \to \infty$ and $k$ bounded, which are separately treated in the proof in Appendix I. The foregoing result does not provide specific regularity conditions needed. But the variance formula (1) evidently agrees with the formal results in Gilks & Berzuini (2001) and Chopin (2004) in the special case of $n_1 = n_2 = k$ and $\ell = 1$. Our result can also be extended by induction to $m (> 2)$ distributions as in Gilks & Berzuini (2001) and Chopin (2004). A rigorous treatment of these issues is beyond the scope of the current paper.

There are clear, interesting interpretations of the leading terms in (1). The first term in (1) depends on three factors: the distance from $P_1$ to $P_2$, encoded by the density ratio $r(\cdot)$, how flat the conditional expectation $h_\ell(\cdot)$ is about the constant $E_2(g)$ due to Markov transitions, and the dependency of observations in $S_1$. In fact, the first two factors directly affect $V_1[r\{h_\ell - E_2(g)\}]$, the variance of $r\{h_\ell - E_2(g)\}$ under $P_1$, whereas the third factor affects how large $v_1[r\{h_\ell - E_2(g)\}]$ is relatively to $V_1[r\{h_\ell - E_2(g)\}]$. The second term in (1) reflects the variance of the sample average $\ell^{-1} \sum_{j=1}^{\ell} g(y_{0,j})$ from the Markov chain. In the following, we discuss how $\text{var}(\bar{g}_2)$ is affected by the choice of $\ell$, for fixed $(P_1, S_1)$ and $(P_2, \Psi_2)$.

For the special case of $\ell = 1$, corresponding to the resample-move algorithm (Gilks & Berzuini 2001), the proof of Proposition 1 gives

$$\text{var} \left\{ n_2^{-1} \sum_{i=1}^{n_2} g(y_{1:i}) \right\} = \frac{1}{n_1} v_1 \left[ r \{ h_1 - E_2(g) \} \right] + \frac{1}{n_2} V_2(g) + o\left(\frac{1}{n_1}\right),$$

6
where \( V_2(g) \) is the variance of \( g(x) \) for \( x \) drawn from \( P_2 \) and \( h_1(\cdot) = E\{g(y_{0,1})|y_0 = \cdot \} \),
the conditional expectation after one Markov transition. For \( \ell = 0 \) or sequential importance resampling without Markov moving (Gordon et al. 1993; Kong et al. 1994), Lemma 1 in Appendix I gives

\[
\text{var}\left\{ \frac{1}{n_2} \sum_{i=1}^{n_2} g(y_i) \right\} = \frac{1}{n_1} v_1 [r\{g - E_2(g)\}] + \frac{1}{n_2} V_2(g) + o\left(\frac{1}{n_1}\right). \tag{2}
\]

The two asymptotic variances differ only in their first terms. The term \( v_1[r\{h_1 - E_2(g)\}] \) tends to be smaller than \( v_1[r\{g - E_2(g)\}] \), because the conditional expectation \( h_1(\cdot) \) is usually flatter than \( g(\cdot) \) about \( E_2(g) \), as can be seen by the inequality \( V_2(h_1) \leq V_2(g) \). If, hypothetically, the kernel \( \Psi_2 \) is perfect such that \( y_{0,1} \) is generated directly from \( P_2 \), regardless of \( y_0 \), then \( h_1(\cdot) = E_2(g) \) and \( v_1[r\{h_1 - E_2(g)\}] = 0 \). But, when the kernel \( \Psi_2 \) leads to moderate to slow mixing, \( h_1(\cdot) \) may not differ much from \( g(\cdot) \) and \( v_1[r\{h_1 - E_2(g)\}] \) may not be much smaller than \( v_1[r\{g - E_2(g)\}] \).

For the generalized resample-move algorithm, there is a tradeoff in the effects of increasing \( \ell \) on \( \text{var}(\tilde{g}_2) \) for fixed \( n_2 = k\ell \). On one hand, \( h_\ell(\cdot) \) is based on \( \ell \) Markov transitions and hence can be much flatter than \( g(\cdot) \) about \( E_2(g) \), so that the first term of \( \text{var}(\tilde{g}_2) \) in (1) for can be much smaller for a large \( \ell \) than that for \( \ell = 1 \). On the other hand, the second term in (1) typically increases as \( \ell \) increases by the standard formula for the normalized variance,

\[
\ell \text{var}_{y_{0}\sim P_2} \left\{ \frac{1}{\ell} \sum_{j=1}^{\ell} g(y_{0,j}) \right\} = V_2(g) + \sum_{j=1}^{\ell-1} 2 \left( 1 - \frac{j}{\ell} \right) \text{cov}_{y_{0}\sim P_2} \{g(y_{0,1}), g(y_{0,j+1})\}. \tag{3}
\]

In fact, for a larger \( \ell \), not only there are more auto-covariance terms in (3) but also the coefficient, \( 1 - j/\ell \), increases for the auto-covariance at each lag \( j \). Therefore, as \( \ell \) increases with \( n_2 = k\ell \) fixed, \( \text{var}(\tilde{g}_2) \) depends on two competing factors: how much \( h_\ell(\cdot) \) is flattened and hence \( v_1[r\{h_\ell - E_2(g)\}] \) is reduced by Markov transitions and how much \( \ell \text{var}_{y_{0}\sim P_2} \{\ell^{-1} \sum_{j=1}^{\ell} g(y_{0,j})\} \) is inflated due to auto-covariances. At the two extremes \( \ell = 0 \) versus \( \ell = n_2 \) (no versus full Markov moving), the comparison of asymptotic variances reduces essentially to the difference between the first term in (2) and the second term in (3) divided by \( \ell = n_2 \).

Based on the foregoing discussion, we examine qualitatively how a good choice of \( \ell \) depends on three factors: the distance from \( P_1 \) to \( P_2 \), mixing of the Markov chain
for $P_2$, and dependency within the sample $S_1$ (which, by the sequential structure, depends on the distances between successive distributions prior to $P_1$ and mixing of the Markov chains for $P_1$ and earlier distributions). There are at least two distinct cases, depending on mixing of the Markov chains:

- If the Markov chain is fast mixing or even locally so (to be explained below) for $P_2$, then a choice of moderate $\ell$, say 10 or 100, might lead to satisfactory results. This is because, after a moderate number of Markov transitions, the conditional expectation $h_\ell(\cdot)$ can already be reduced to a nearly flat (or piecewise flat) function, but the sum of auto-covariances in (3) can keep increasing in $\ell$. Choosing a larger $\ell$ would not be worthwhile.

- If the Markov chain is slow mixing for $P_2$, then a choice of large $\ell$, say 1000 or more, might be warranted. This is because, even after a moderate number of Markov transitions, $h_\ell(\cdot)$ can be only slightly flattened, and the accumulation of auto-covariances in (3) can be far from its limit as $\ell \to \infty$. Choosing a larger $\ell$ would lead to a better performance.

These two cases are, in fact, observed respectively in the two numerical examples in Section 3. See Appendix IV for further discussion.

There are several ideas that need to be explained for the first case above. We say that a Markov chain with transition kernel $\Psi$ locally geometrically converges to $P$ (or, informally, is locally fast mixing) if the sample space can be partitioned into $J$ regions $B_1, \ldots, B_J$ such that the total variation distance

$$\left\| \sum_{j=1}^{J} P(B_j) \Psi^n(x_j, \cdot) - P(\cdot) \right\|$$

decreases at an exponential rate as $n \to \infty$, uniformly over $x_j \in B_j$ ($j = 1, \ldots, J$), where $\Psi^n(x, \cdot)$ is the $n$-step transition kernel. That is, if the chain is initialized, with stationary probability $P(B_j)$, from any $x_j \in B_j$, then the chain geometrically converges to $P$. This concept reduces to the usual geometric convergence for $J = 1$, but can be much weaker for $J \geq 2$. For example, a random-walk Metropolis chain with Gaussian proposals for sampling a well-separated Gaussian mixture is easily trapped in the region where the initial value is located, but it is locally fast mixing with the sample space partitioned according to the mixture components. There are unusual
features for locally fast mixing Markov chains. First, as mentioned above, \( h_\ell(y_0) \) defined in Proposition 1 converges to only a piecewise constant function, because the limiting value depends on the region where \( y_0 \) is located. Second, the stationary auto-correlation, \( \text{cor}_{y_0 \sim P}(y_0:1, y_{0:j}) \), does not, in general, converge to 0 as \( j \to \infty \), even though the sample auto-correlation converges to 0 for a chain initialized with any fixed \( y_0 \). Then \( \ell \var{y_0 \sim P_2} \{\ell^{-1} \sum_{j=1}^{\ell} g(y_{0:j})\} \) in (3) diverges as \( \ell \) increases, in contrast with the usual case for geometrically convergent chains. We provide numerical illustration in Appendix IV, and leave further analysis to future work.

Finally, the generalized resample-move algorithm for two distributions (\( m = 2 \)) is reminiscent of the recommended strategy for posterior simulation in Bayesian statistics by Gelman et al. (2003). For that strategy, \( P_2 \) is the posterior distribution and \( P_1 \) is a mode-based, possibly mixture approximation to \( P_2 \). First, a sample \( S_1 \) is drawn from \( P_1 \). Then a few (say \( k \)) observations are obtained by importance resampling from \( S_1 \), and used as starting points to run parallel Markov chains, each of length \( \ell \), for \( P_2 \). Gelman et al. (2003) seemed to emphasize that importance resampling is useful to provide over-dispersed starting points for MCMC, and focus on the case where \( k \) is much smaller than \( \ell \). By comparison, we allow the case where \( \ell \) is smaller than or comparable to \( k \), and aim to exploit importance resampling as a technique to build an intermediate sample for \( P_2 \) given a good sample for \( P_1 \).

### 2.3 Other algorithms

The main feature of the generalized resample-move algorithm in the framework of resampling MCMC is that both \( k_j \) and \( \ell_{ji} \) are pre-specified constants. We show that choosing \( k_j \) and \( \ell_{ji} \) randomly leads to other existing algorithms, including importance resampling MCMC (Atchadé 2009), equi-energy sampling (Kou et al. 2006), and modified equi-energy sampling by Atchadé (2010). Moreover, we examine the relationship between these algorithms and parallel tempering (Geyer 1991), which, however, does not belong to resampling MCMC.

For each \( j \geq 2 \), consider a Bernoulli process to choose \( k_j \) and \( \ell_{ji} \). First, generate a sequence of independent Bernoulli variables \( (\Delta_{j1}, \ldots, \Delta_{jn_j}) \) with a fixed success probability \( 0 < \alpha_j < 1 \). The total number of successes is \( K_j = \sum_{i=1}^{n_j} \Delta_{ji} \). Then \( k_j \) and
\( \ell_{ji} \) are determined as follows. The two cases, depending on \( \Delta_{j1} \), are mainly created to accommodate certain details at the beginning of various existing algorithms, even though such details are asymptotically negligible.

- If \( \Delta_{j1} = 1 \), then let \( k_j = K_j \) and, for \( i \geq 1 \), \( \ell_{ji} \) be the number of trials from the \( i \)th success until and excluding the \((i + 1)\)th success or until and including the \( n_j \)th trial. For example, \( n_j = 9, k_j = 3, (\ell_{j1}, \ell_{j2}, \ell_{j3}) = (3, 4, 2) \) for the Bernoulli sequence 100100010.
- If \( \Delta_{j1} = 0 \), then let \( k_j = 1 + K_j, \ell_{j1} \) be the number of trials before the first success, and, for \( i \geq 2 \), \( \ell_{ji} \) be the number of trials from the \((i - 1)\)th success until and excluding the \( i \)th success or until and including the \( n_j \)th trial. For example, \( n_j = 9, k_j = 3, (\ell_{j1}, \ell_{j2}, \ell_{j3}) = (2, 4, 3) \) for the Bernoulli sequence 001000100.

Therefore, \( k_j \) is essentially a binomial variable with mean \( n_j \alpha_j \), and \( \ell_{ji} \) is a geometric variable with mean \( \alpha_j^{-1} \) except for \( i = 1 \) or \( k_j \). If \( \alpha_j \) is specified as \( \ell_j^{-1} \), then the expected size of resampling and the expected number of Markov moves are the same as those in the generalized resample-move algorithm.

### 2.3.1 Importance resampling MCMC

To obtain importance resampling MCMC (Atchadé 2009) in the framework of resampling MCMC, consider a Bernouilli process to choose \( k_j \) and \( \ell_{ji} \) in step (2a), importance resampling in step (2b), and the following rejection rule in step (2c):

- If \( \Delta_{j1} = 1 \), then accept \( y_{ji} = z_{ji} \) for all \( i = 1, \ldots, k_j \).
- If \( \Delta_{j1} = 0 \), then reject \( z_{j1} \) and let \( y_{j1} = x_{j0} \), and accept \( y_{ji} = z_{ji} \) for \( i = 2, \ldots, k_j \).

All observations \( z_{ji} \) are accepted except \( z_{j1} \) obtained when \( \Delta_{j1} = 0 \) (not when \( \Delta_{j1} = 1 \)). By concatenation of the individual chains \((y_{ji:1}, \ldots, y_{ji:\ell_{ji}})\), the resulting algorithm can be equivalently described as follows such that observations are generated in a single chain for each distribution as in MCMC.

**Static importance resampling MCMC:**

- Generate \( S_1 = \{x_{11}, \ldots, x_{1n_1}\} \) using \( \Psi_1 \) with the initial value \( x_{10} \) for \( P_1 \).
- For \( j = 2, 3, \ldots, m \),
  - For \( i = 1, \ldots, n_j \),
* Generate a Bernoulli variable $\Delta_{ji}$ with success probability $\alpha_j$. If $\Delta_{ji} = 1$, draw $\xi_{ji}$ from $S_{j-1} = \{x_{j-1,1}, \ldots, x_{j-1,n_{j-1}}\}$ by importance resampling. If $\Delta_{ji} = 0$, let $\xi_{ji} = x_{j,i-1}$.

* Generate $x_{ji}$ using the kernel $\Psi_j$ with the initial value $\xi_{ji}$.
  - Define a sample for $P_j$ as $S_j = \{x_{j1}, \ldots, x_{jn_j}\}$.

It is interesting to see how the foregoing algorithm agrees with the general framework. For each success $\Delta_{ji}$, a draw $\xi_{ji}$ is obtained by importance resampling from $S_{j-1}$. Then a chain of observations are generated using the kernel $\Psi_j$ with the initial value $\xi_{ji}$, until the next success. The number of draws from $S_{j-1}$ is the number of successes, and the length of the chain generated given a draw from $S_{j-1}$ is an inter-success time or a truncated one at the end of trials.

The original algorithm of importance resampling MCMC in Atchadé (2009) is the same as the foregoing algorithm except for an important difference. The reservoir for drawing $\xi_{ji}$ by importance resampling is not fixed as the entire sample $S_{j-1}$, but restricted to the history of observations $\{x_{j-1,1}, \ldots, x_{j-1,i}\}$ up to time $i$. For this reason, the foregoing algorithm or Atchadé’s (2009) algorithm is called, respectively, static or dynamic importance resampling MCMC. Assume that the sample size $n_j$ is equal to $n$ for each $j = 1, \ldots, m$. By an exchange of the order of iterations in $j$ and $i$, dynamic importance resampling MCMC is equivalent to running $m$ parallel chains as in the original description of Atchadé (2009).

**Dynamic importance resampling MCMC (Atchadé 2009):**

- For $i = 1, \ldots, n$,
  - Generate $x_{1i}$ using the kernel $\Psi_1$ with the initial value $x_{1,i-1}$.
  - For $j = 2, 3, \ldots, m$,
    * Generate a Bernoulli variable $\Delta_{ji}$ with success probability $\alpha_j$. If $\Delta_{ji} = 1$, draw $\xi_{ji}$ from $\{x_{j-1,1}, \ldots, x_{j-1,i}\}$ by importance resampling. If $\Delta_{ji} = 0$, let $\xi_{ji} = x_{j,i-1}$.
    * Generate $x_{ji}$ using the kernel $\Psi_j$ with the initial value $\xi_{ji}$.

- For $j = 1, \ldots, m$, define a sample for $P_j$ as $S_j = \{x_{j1}, \ldots, x_{jn_j}\}$.

There is another minor difference ignored between dynamic importance resampling MCMC described above and the original algorithm in Atchadé (2009). For the latter
algorithm, it is typical that $x_{ji}$ is taken directly as a weighted draw from the previous chain up to time $i$ if $\Delta_{ji} = 1$ and generated using the kernel $\Psi_j$ with the initial value $x_{j,i-1}$ if $\Delta_{ji} = 0$, so that the total number of Markov moves is random with mean $(1 - \alpha_j) n$ for the $j$th chain. For either static or dynamic importance resampling MCMC presented here, the total number of Markov moves is fixed at $n$ for each sample. This property makes it convenient to compare various algorithms with the same number of Markov moves used.

### 2.3.2 Independence Metropolis resampling MCMC and equi-energy sampling

Generalized resample-move and importance resampling MCMC have the common feature that importance resampling is used in step (2b) and all selected observations $z_{ji}$ are accepted in step (2c), except for an artificial rejection at the beginning of the latter algorithm, in the framework of resampling MCMC. An alternative approach is independence Metropolis resampling, where simple random sampling from $S_{j-1}$ is used and the selected observations $z_{ji}$ are either rejected or accepted according to the rejection rule in independence Metropolis sampling (e.g., Liu 2001, Section 5.4.2).

The corresponding steps (2b) and (2c) in Section 2.1 are as follows.

1. **(2b)** Draw $k_j$ observations $\{z_{j1}, \ldots, z_{jk_j}\}$ with replacement by simple random sampling from $S_{j-1} = \{x_{j-1,1}, \ldots, x_{j-1,n_{j-1}}\}$.

2. **(2c)** For $i = 1, \ldots, k_j$, accept $y_{ji} = z_{ji}$ with probability
   $$
   \min \left\{ 1, \frac{q_j(z_{ji})q_{j-1}(y_{j(i-1)}\ell_{j,i-1})}{q_{j-1}(z_{ji})q_j(y_{j(i-1)}\ell_{j,i-1})} \right\}
   $$
   and, with the remaining probability, reject $z_{ji}$ and let $y_{ji} = x_{j0}$ if $i = 1$ or $y_{j,i-1}\ell_{j,i-1}$ if $i \geq 2$, and then generate a chain of $\ell_{ji}$ observations $\{y_{ji1}, \ldots, y_{ji\ell_{ji}}\}$ using the kernel $\Psi_j$ with the initial value $y_{ji}$.

It is important to notice that each acceptance or rejection of $z_{ji}$ involves a reference observation, which is specifically set to $y_{j,i-1}\ell_{j,i-1}$, the last observation in the chain generated with the initial value $y_{j,i-1}$.

With a Bernoulli process to choose $k_j$ and $\ell_{ji}$, the resulting algorithm can be equivalently described in the form of running a single chain for each distribution, similarly to static importance resampling MCMC in Section 2.3.1.
Static independence Metropolis resampling MCMC:

- Generate \( S_1 = \{x_{11}, \ldots, x_{1n_1}\} \) using \( \Psi_1 \) with the initial value \( x_{10} \) for \( P_1 \).

- For \( j = 2, 3, \ldots, m \),
  - For \( i = 1, \ldots, n_j \),
    * Generate a Bernoulli variable \( \Delta_{ji} \) with success probability \( \alpha_j \). If \( \Delta_{ji} = 1 \), then draw \( \zeta_{ji} \) from \( S_{j-1} = \{x_{j-1,1}, \ldots, x_{j-1,n_{j-1}}\} \) by simple random sampling, accept \( \xi_{ji} = \zeta_{ji} \) with probability
    \[
    \min \left\{ 1, \frac{q_j(\zeta_{ji})q_{j-1}(x_{j,i-1})}{q_{j-1}(\zeta_{ji})q_j(x_{j,i-1})} \right\},
    \]  
    and, with the remaining probability, reject \( \zeta_{ji} \) and let \( \xi_{ji} = x_{j,i-1} \). If \( \Delta_{ji} = 0 \), then let \( \xi_{ji} = x_{j,i-1} \).
    * Generate \( x_{ji} \) using the kernel \( \Psi_j \) with the initial value \( \xi_{ji} \).
  - Define a sample for \( P_j \) as \( S_j = \{x_{j1}, \ldots, x_{jn_j}\} \).

Changing the reservoir from \( S_{j-1} \) to the history \( \{x_{j-1,1}, \ldots, x_{j-1,i}\} \) up to time \( i \) for drawing \( \zeta_{ji} \) in the foregoing algorithm leads to dynamic independence Metropolis resampling MCMC, which, similarly to dynamic importance resampling MCMC, can be equivalently described as running \( m \) parallel chains.

Dynamic independence Metropolis resampling MCMC or modified equi-energy sampling (Atchadé 2010):

- For \( i = 1, \ldots, n \),
  - Generate \( x_{1i} \) using the kernel \( \Psi_1 \) with the initial value \( x_{1,i-1} \).
  - For \( j = 2, 3, \ldots, m \),
    * Generate a Bernoulli variable \( \Delta_{ji} \) with success probability \( \alpha_j \). If \( \Delta_{ji} = 1 \), then draw \( \zeta_{ji} \) from \( \{x_{j-1,1}, \ldots, x_{j-1,i}\} \) by simple random sampling and accept \( \xi_{ji} = \zeta_{ji} \) with probability (4) or reject \( \zeta_{ji} \) and let \( \xi_{ji} = x_{j,i-1} \). If \( \Delta_{ji} = 0 \), then let \( \xi_{ji} = x_{j,i-1} \).
    * Generate \( x_{ji} \) using the kernel \( \Psi_j \) with the initial value \( \xi_{ji} \).
- For \( j = 1, \ldots, m \), define a sample for \( P_j \) as \( S_j = \{x_{j1}, \ldots, x_{jn_j}\} \).

This algorithm is identical to modified equi-energy sampling in Atchadé (2010), except for a minor difference (as in the end of Section 2.3.1). For the latter algorithm,
$x_{ji}$ is generated using the kernel $\Psi_j$ with the initial value $\xi_{ji}$ if $\Delta_{ji} = 0$, but taken directly as $\xi_{ji}$ if $\Delta_{ji} = 1$, so that the total number of Markov moves is random with mean $(1 - \alpha_j)n$ for the $j$th chain. In contrast, the total number of Markov moves is fixed for each sample for static or dynamic independence Metropolis resampling MCMC presented here. Another related algorithm is sequentially interacting MCMC in Brockwell et al. (2010) designed for sampling from distributions on nested spaces. When restricted to a common state space, the algorithm of Brockwell et al. corresponds to dynamic independence Metropolis resampling MCMC, with $\alpha_j = 1$ (i.e., always drawing $\zeta_{ji}$) and no Markov moving (hence $x_{ji} = \xi_{ji}$).

Equi-energy sampling (Kou et al. 2006) is the same as dynamic independence Metropolis sampling described above except for the construction of the reservoir for drawing $\zeta_{ji}$. Consider the case where $q_j(x) = \exp\{-u(x)/T_j\}$ with $T_1 > T_2 > \cdots > T_H$ and partition the sample space into energy rings $D_h = \{x : U_h \leq u(x) < U_{h+1}\}$ for $h = 1, \ldots, H$, where $U_1 < U_2 < \cdots < U_H < U_{H+1} = \infty$ are fixed energy levels and $U_1$ is below the minimum energy. Two versions of equi-energy sampling are shown as follows, with the dynamic version corresponding more closely to the original algorithm of Kou et al. (2006) (see Remark 6 in Section 2.4).

- **Static equi-energy sampling**: Same as static independence Metropolis resampling MCMC except that the resampling step is to draw $\zeta_{ji}$ from $\hat{D}_{j-1,h} = \{x_{j-1,\nu} \in D_h : \nu = 1, \ldots, n_j\} = D_h \cap S_{j-1}$, with the ring index $h$ such that $x_{j,i-1} \in D_h$.

- **Dynamic equi-energy sampling** (Kou et al. 2006): Same as dynamic independence Metropolis resampling MCMC except that the resampling step is to draw $\zeta_{ji}$ from $\hat{D}_{j-1,i,h} = \{x_{j-1,\nu} \in D_h : \nu = 1, \ldots, i\} = D_h \cap \{x_{j-1,1}, \ldots, x_{j-1,i}\}$, with the ring index $h$ such that $x_{j,i-1} \in D_h$.

Therefore, $\zeta_{ji}$ is drawn from the higher-temperature ring $\hat{D}_{j-1,h}$ constructed from the entire sample $S_{j-1}$ or from $\hat{D}_{j-1,i,h}$ constructed from the history $\{x_{j-1,1}, \ldots, x_{j-1,i}\}$, with the ring index determined by $x_{j,i-1}$.

### 2.3.3 Parallel tempering

Parallel tempering (Geyer 1991) involves running $m$ parallel chains, similarly to dynamic versions of resampling MCMC. However, there are fundamental differences.
For resampling MCMC, an observation from a higher-temperature chain can be fed into a lower temperature chain by resampling and acceptance-rejection, but there is no mechanism for an observation from a lower-temperature chain to be taken into a higher-temperature chain. For parallel tempering, an observation from a higher-temperature chain and another from a lower-temperature chain can literally be exchanged under some criterion. An advantage of parallel tempering is that the m chains jointly constitute a Markov chain in the product sample space. None of the resampling MCMC algorithms discussed have this property.

To connect resampling MCMC and parallel tempering (rather than introducing a competitive algorithm), consider the following procedure, obtained from dynamic independence Metropolis resampling MCMC by changing the reservoir for drawing $\zeta_{ji}$ to the singleton $\{x_{j-1,i-1}\}$ so that $\zeta_{ji} = x_{j-1,i-1}$.

**Independence Metropolis tempering:**

- For $i = 1, \ldots, n$,
  - Generate $x_{1i}$ using the kernel $\Psi_1$ with the initial value $x_{1,i-1}$.
  - For $j = 2, 3, \ldots, m$,
    * Generate a Bernoulli variable $\Delta_{ji}$ with success probability $\alpha_j$. If $\Delta_{ji} = 1$, then let $\xi_{ji} = x_{j-1,i-1}$ with probability
      $$\min \left\{ 1, \frac{q_j(x_{j-1,i-1})q_{j-1}(x_{j,i-1})}{q_{j-1}(x_{j-1,i-1})q_j(x_{j,i-1})} \right\},$$
      (5)
    and, with the remaining probability, let $\xi_{ji} = x_{j,i-1}$. If $\Delta_{ji} = 0$, then let $\xi_{ji} = x_{j,i-1}$.
    * Generate $x_{ji}$ using the kernel $\Psi_j$ with the initial value $\xi_{ji}$.

- For $j = 1, \ldots, m$, define a sample for $P_j$ as $S_j = \{x_{j1}, \ldots, x_{jn}\}$.

The foregoing algorithm appears similar to parallel tempering described below, with the same formula (5) for acceptance probability. However, in the case of acceptance, $x_{j-1,i-1}$ is substituted for, not exchanged with, $x_{j,i-1}$. For the modification from dynamic independence Metropolis resampling MCMC, it is also possible to take $\zeta_{ji} = x_{j-1,i}$, which would be as good as taking $\zeta_{ji} = x_{j-1,i-1}$, but the acceptance probability would be (5) with $x_{j-1,i-1}$ replaced by $x_{j-1,i}$.

**Parallel tempering (Geyer 1991):**
For $i = 1, \ldots, n$,
- For $j = 2, 3, \ldots, m$,
  * Generate a Bernoulli variable $\Delta_{ji}$ with success probability $\alpha_j$. If $\Delta_{ji} = 1$, then exchange $x_{j-1,i-1}$ and $x_{j,i-1}$ with probability (5) and, with the remaining probability, take no action. If $\Delta_{ji} = 0$, then take no action.
- For $j = 1, 2, 3, \ldots, m$,
  * Generate $x_{ji}$ using the kernel $\Psi_j$ with the initial value $x_{j,i-1}$.

For $j = 1, \ldots, m$, define a sample for $P_j$ as $S_j = \{x_{j1}, \ldots, x_{jn}\}$.

Our implementation of parallel tempering somewhat differs from those in the literature (e.g., Geyer 1991; Liu 2001, Section 10.4). At each time $i$, a proposal of exchange is made with some probability between every two adjacent chains. There can be exchanges between multiple pairs of adjacent chains, in the order of decreasing temperatures. Moreover, at each time $i$, there is always one Markov move for each chain. The total number of Markov moves is fixed at $n$ for each sample, similarly as in the resampling MCMC algorithms discussed earlier.

2.4 Comparisons

Given the multitude of algorithms discussed, it is important to understand how their performances are compared to each other. A complete theoretical study of this problem is beyond the scope of this article. In Section 3, we evaluate various algorithms empirically in three simulation studies. In this section, we provide some basic analysis of the relationship between these algorithms.

Remark 1. Among the resampling MCMC algorithms discussed in Section 2.3, static importance resampling MCMC is the most related to generalized resample-move. The two algorithms differ only in their choices of $k_j$ and $\ell_{ji}$. For each $j$, generalized resample-move involves drawing a sample of fixed size $k_j$ from $S_{j-1}$ with importance weights and, given the $i$th draw, generating a chain of fixed length $\ell_{ji} = \ell_j$. By comparison, static importance resampling involves drawing a sample of random size $k_j$ from $S_{j-1}$ with importance weights and, given the $i$th draw, generating a chain of random length $\ell_{ji}$, where $k_j$ and $\{\ell_{ji} : i = 1, \ldots, k_j\}$ are determined from a Bernoulli process. The Bernoulli process is independent of other parts of the algorithm, and
hence seems to only introduce additional randomness (or noise) into the algorithm. By this relationship, we expect that static importance resampling MCMC might not be as efficient as generalized resample-move.

Remark 2. Static independence Metropolis resampling MCMC appears to differ from static importance resampling MCMC only in their resampling schemes (including the acceptance or rejection step), as indicated by their names. However, a careful examination of the details reveals that there are more subtle differences between the two algorithms than might be inferred from their names.

Formally, a single iteration of independence Metropolis resampling is as follows. Let \( S_1 \) be a sample for \( P_1 \) and \( y_{i-1} \) be the previous (or initial) observation for \( P_2 \). Then draw 1 observation \( z_i \) from \( S_1 \) with uniform probabilities, and accept \( y_i = z_i \) with probability \( \min[1, q_2(z_i)q_1(y_{i-1})/\{q_1(z_i)q_2(y_{i-1})\}] \) and, with the remaining probability, reject \( z_i \) and let \( y_i = y_{i-1} \). A seemingly small but important issue is how the previous observation \( y_{i-1} \) is specified. For static independence Metropolis resampling MCMC in Section 2.3.2, the previous observation used in the acceptance or rejection of a draw \( z_{ji} \) from \( S_j \) is set to \( y_{ji} = y_{ji-1} \), the last observation generated by Markov moving from \( y_{ji-1} \), instead of \( y_{ji-1} \) itself. (The index \( j \) can be taken as 2 and then removed to understand the notations.) In fact, changing the previous observation \( y_{ji-1} \) to \( y_{ji-1} \) leads to an entirely different algorithm, which we call lazy independence Metropolis resampling MCMC.

Lazy independence Metropolis resampling MCMC can be organized such that the sample \( \{y_{j1}, \ldots, y_{jk_j}\} \) is completely obtained from \( S_{j-1} \) with the initial value \( x_{j0} \) by independence Metropolis resampling and then, for \( i = 1, \ldots, k_j \), a chain of \( \ell_{ji} \) observations are generated from \( y_{ji} \) by Markov moving. Therefore, this algorithm is the same as static importance resampling MCMC, except that, in the latter algorithm, \( \{y_{j1}, \ldots, y_{jk_j}\} \) is drawn from \( S_{j-1} \) by importance resampling without using the initial value \( x_{j0} \). However, by this relationship, we expect that lazy independence Metropolis sampling MCMC might not be as efficient as importance resampling MCMC for the following reason. It is known that importance sampling is more efficient than independence Metropolis sampling (Tan 2006). Suppose that \( \{x_1, \ldots, x_n\} \) is an independent and identically distributed sample from \( P_1 \) and \( \{y_1, \ldots, y_n\} \) is a sequence.
of observations obtained for $P_2$ by independence Metropolis sampling. Under mild regularity conditions, Tan (2006) showed that for the expectation of a function $\varphi(x)$ under $P_2$, the ratio estimator $\sum_{i=1}^n \varphi(x_i)w(x_i)/\sum_{i=1}^n w(x_i)$ has no greater asymptotic variance than $n^{-1}\sum_{i=1}^n \varphi(y_i)$, where $w(x) = q_2(x)/q_1(x)$. Resampling is more complicated than sampling, especially when resampling is done from a dependent sample. Nevertheless, the relative efficiency of importance resampling over independence Metropolis resampling is likely to be carried over to the comparison between the corresponding resampling algorithms.

For static independence Metropolis resampling MCMC, $y_{ji}$ is determined by acceptance or rejection, depending on the last observation $y_{j;i-1;\ell_{j;i-1}}$ generated by Markov moving from $y_{j;i-1}$. Therefore, independence Metropolis resampling and Markov moving are interwoven in an interactive manner. This structure sets static independence Metropolis sampling MCMC apart from both lazy independence Metropolis and importance resampling MCMC. The foregoing argument on the comparison between the latter two algorithms is no longer applicable to that between static independence Metropolis and importance resampling MCMC.

**Remark 3.** The interweaving structure can give independence Metropolis resampling MCMC an advantage over importance resampling MCMC and even generalized resample-move with the choice of $\alpha_j$ or $\ell_j$ matched, when both the distance from $P_{j-1}$ to $P_j$ is substantial and the Markov kernel $\Psi_j$ leads to slow mixing.

We provide some heuristic reasons for this claim. For independence Metropolis resampling MCMC, whenever a draw $z_{ji}$ from $S_{j-1}$ is rejected, $y_{ji}$ is set to $y_{j;i-1;\ell_{j;i-1}}$. Then the concatenation of $\{y_{j;i-1:1}, \ldots, y_{j;i-1:\ell_{j;i-1}}\}$ and $\{y_{j;i:1}, \ldots, y_{j;i:\ell_j}\}$ forms a longer chain of successive observations under the kernel $\Psi_j$. If two or more consecutive draws from $S_{j-1}$ are rejected, then the corresponding chains can be concatenated into an even longer chain. But as a consequence of a substantial distance from $P_{j-1}$ to $P_j$, a large proportion of the draws $\{z_{ji} : i = 1, \ldots, k_j\}$ are likely to be rejected. Then the individual chains $\{(y_{j;i:1}, \ldots, y_{j;i:\ell_{j;i}}) : i = 1, \ldots, k_j\}$ in $S_j$ can be grouped into a smaller number of longer chains, each generated with the initial value a draw $z_{ji}$ from $S_{j-1}$ that is accepted as $y_{ji}$. Such longer chains tend to better represent the distribution $P_j$ than a larger number of shorter chains, because $S_{j-1}$, a sample for $P_{j-1}$,
might not sufficiently cover $P_j$ and the Markov chain is slow mixing under $\Psi_j$. This is similar to the situation for generalized resample-move in Section 2.2: if the Markov chain is slow mixing, reducing the size $k_j$ of resampling from $S_{j-1}$ while increasing the number $\ell_j$ of Markov moves might improve the performance. The analogue is not exact: the number of accepted draws and the lengths of Markov moves are dynamically determined for independence Metropolis resampling MCMC. Nevertheless, the concatenation of shorter chains into longer ones seems to be advantageous in the case of a large distance from $P_{j-1}$ to $P_j$ and slow mixing under $\Psi_j$.

Remark 4. Equi-energy sampling differs from independence Metropolis resampling MCMC only in the use of energy rings to construct resampling reservoir. Atchadé (2010) studied the dynamic version of the latter algorithm as modified equi-energy sampling, and claimed that using energy rings “does not add any significant feature to the algorithm from the theoretical standpoint.” However, we argue that, by the use of energy rings, equi-energy sampling is more similar to importance resampling MCMC than to independence Metropolis resampling MCMC.

For static equi-energy sampling, recall from Section 2.3.2 that $\zeta_{ji}$ is drawn from the energy ring of $x_{ji-1}$, estimated based on $S_{j-1}$. Then the energy $u(\zeta_{ji})$ is similar to $u(x_{ji-1})$, so that the probability (4) of accepting $\zeta_{ji}$ is large or close to 1, depending on how finely the sample space is partitioned into energy rings. Furthermore, $\zeta_{ji}$ given $x_{ji-1}$ is uniformly distributed over the estimated energy ring of $x_{ji-1}$. But what is approximately the distribution of $\zeta_{ji}$ integrated over $x_{ji-1}$? If, hypothetically, $x_{ji-1}$ were considered a simple random draw from $S_j$ and if $S_{j-1}$ and $S_j$ were fixed, then, by direct calculation, $\zeta_{ji}$ would be a random draw from $S_{j-1}$ with probabilities proportional to $\hat{q}_j(x_{j-1,\nu})/\hat{q}_{j-1}(x_{j-1,\nu})$ for $x_{j-1,\nu} \in S_{j-1}$, where $\hat{q}_{j-1}(x_{j-1,\nu})$ or $\hat{q}_j(x_{j-1,\nu})$ is the relative frequency of the energy ring of $x_{j-1,\nu}$ based on $S_{j-1}$ or $S_j$ respectively. [In fact, the probability of $x_{ji-1}$ falling into the energy ring of $x_{j-1,\nu}$ is proportional to $\hat{q}_j(x_{j-1,\nu})$ and the probability of selecting $x_{j-1,\nu}$ within its energy ring based on $S_{j-1}$ is proportional to $\hat{q}_{j-1}^{-1}(x_{j-1,\nu})$.] In other words, $\zeta_{ji}$ would be drawn from $S_{j-1}$ by importance resampling, except that the ratio of unnormalized densities $q_j(x_{j-1,\nu})/q_{j-1}(x_{j-1,\nu})$ is estimated by that of relative frequencies $\hat{q}_j(x_{j-1,\nu})/\hat{q}_{j-1}(x_{j-1,\nu})$. Although not rigorous, the foregoing analysis strongly suggests that equi-energy sampling is similar
to importance resampling MCMC in Section 2.3.1, where $\zeta_j$ is drawn from $S_{j-1}$ by importance resampling and then always accepted as $\xi_j$.

Remark 5. A dynamic resampling MCMC algorithm is less efficient than the static counterpart, because the resampling reservoir is only the history of observations in the previous chain. On the other hand, a dynamic algorithm can be implemented as running $m$ parallel chains. Such implementation is time-saving on parallel computing systems, although the total computational load is unchanged. As a compromise, a semi-dynamic scheme can be used as in Kou et al. (2006). For $j \geq 2$, the chain for $P_j$ is not started until the chain for $P_{j-1}$ has been run for a certain number of iterations (which we call waiting period), in addition to any burn-in period. An important issue is then how to choose the length of the waiting period. For space limitation, we do not investigate the semi-dynamic option further.

Remark 6. Parallel tempering differs fundamentally from resampling MCMC algorithms, as discussed in Section 2.3.3. Nevertheless, there is a similar mechanism to that in independence Metropolis resampling MCMC, such that shorter sequences of Markov moves are concatenated into longer ones. First, the $j$th chain $S_j$ can be segmented into subchains at times $i$ where an exchange is proposed between the $j$th chain and an adjacent chain. The number of subchains and theirs lengths are controlled by the Bernoulli processes $\{\Delta_{j-1,i}\}$ and $\{\Delta_{ji}\}$, independently of other parts of the algorithm. But whenever a proposed exchange is rejected, the initial value for the subsequent subchain is exactly the last observation in the previous chain, so that the two subchains can be smoothly connected. Then the subchains in $S_j$ can be grouped into a smaller number of longer chains. By this structure, parallel tempering might perform better than importance resampling MCMC and generalized resample-move in the case of a substantial distance from $P_{j-1}$ to $P_j$ and slow mixing under the kernel $\Psi_j$, similarly as discussed in Remark 3.

Remark 7. A number of algorithms discussed in the previous sections are also studied, sometimes developed independently, in statistical physics. Parallel tempering is known as replica exchange MCMC (Hukushima & Nemoto 1996). Independence Metropolis tempering and static independence Metropolis resampling MCMC are essentially equivalent to two implementations of the J-walking method (Frantz
et al. 1990). The former involves running two walkers (or chains) in tandem at two temperatures and the latter involves generating and storing a chain at a higher temperature beforehand. The resample-move algorithm seems to be introduced under the name of population annealing in statistical physics (Machta 2010).

3 Simulation studies

3.1 Preliminary

We compare 11 algorithms, grouped and labeled as follows:

(1–3) sequential importance resampling, resample-move, and generalized resample-move,

(4–6) static dynamic importance resampling MCMC, static independence Metropolis resampling MCMC, and static equi-energy sampling,

(7) parallel tempering,

(8–11) dynamic importance resampling MCMC, dynamic independence Metropolis resampling MCMC, dynamic equi-energy sampling, and independence Metropolis tempering.

For all the algorithms except parallel tempering, a sample for \( P_1 \) is obtained by directly running a Markov chain using the kernel \( \Psi_1 \) with an initial value \( x_{10} \). The other initial values \( x_{j0} \) are set to \( x_{10} \) for \( j \geq 2 \).

The sample sizes are set to facilitate a fair comparison between algorithms with different burn-in schedules. For our examples, \( m = 5 \) is used. Let \( n \) and \( b \) be roughly the sample and burn-in sizes. For dynamic algorithms labeled 8–11, a sample for \( P_j \) is obtained with the first \( b \) iterations of the chain discarded as burn-in. A chain for \( P_{j+1} \) is also started after the first \( b \) iterations done in the chain for \( P_j \). For \( (P_1, P_2, \ldots, P_5) \), the numbers of iterations are set respectively to \( (n + 5b, n + 4b, n + 3b, n + 2b, n + b) \), so that the sample sizes are \( (n + 4b, n + 3b, n + 2b, n + b, n) \). For static algorithms labeled 1–6, a sample for \( P_1 \) is still obtained after the first \( b \) iterations discarded, but the remaining samples are defined without burn-in. For the same numbers of iterations as above, the sample sizes are then \( (n + 4b, n + 4b, n + 3b, n + 2b, n + b) \) for \( (P_1, P_2, \ldots, P_5) \). Finally, for parallel tempering, the number of parallel iterations is set to \( n + 3b \), and the sample size is \( n + 2b \) with the first \( b \) iterations discarded as
burn-in for all the distributions. Therefore, the combined number of iterations over the 5 distributions is $5n + 15b$, the same for all the 11 algorithms.

We present three simulation studies, each with interesting features that make the sampling problem difficult. The first two studies involve sampling from Gaussian mixtures with well-separated modes, using a sequence of tempering distributions. Of particular interest is that the difficulty of such problems can substantially increase when the Gaussian components have different variances and weights. In fact, the sequence of tempering distributions in the second study exhibits a phenomenon somewhat similar to phase transition in statistical physics. The third study considers the Potts model near phase transition, where the energy histogram is bimodal. Such a phase transition is known to be a major obstacle to Monte Carlo sampling.

3.2 Bimodal multivariate Gaussian mixtures

3.2.1 Setup and difficulty

We present in Appendix II an example on multimodal bivariate Gaussian mixtures, taken and extended from Liang & Wong (2001) and Kou et al. (2006). Here we consider sampling from a bimodal multivariate Gaussian mixture:

\[
p(x) = \sum_{i=1}^{2} \frac{\lambda_i}{(2\pi)^{d/2}\sigma_i^d} \exp\left\{-\frac{1}{2\sigma_i^2}(x - \mu_i)^T(x - \mu_i)\right\}
\]

where $d = 6$, $\mu_1 = (-1, \ldots, -1)$, $\mu_2 = (1, \ldots, 1)$, $(\sigma_1, \sigma_2) = (0.2, 0.1)$, and $(\lambda_1, \lambda_2) = (1/3, 2/3)$. The two Gaussian components are well separated, and the component with mean $\mu_1$ has a larger variance but a smaller weight than the other component. As explained below, sampling from such a Gaussian mixture with heteroscedastic components can be much more difficult than with homoscedastic components, for methods using the geometric type of tempering distributions.

The main difficulty is that the weight of a larger-variance component is exponentially amplified, relatively to a smaller-variance component in the geometric sequence of tempering distributions for a Gaussian mixture. For the bimodal mixture $p(x)$, we show in Appendix I that for $T \leq 10$, the tempering distribution with the un-normalized density $\{p(x)\}^{1/T}$ is approximately a Gaussian mixture with two components, with means $\mu_1$ and $\mu_2$, variances $\sigma_1^2TI_d$ and $\sigma_2^2TI_d$, and weights proportional
Figure 1: Smoothed histograms of the $d$th component $x(d)$ at temperatures $T_1$ (dotted), $T_2$ (dotdash), $T_3$ (dash), $T_4$ (longdash), and $T_5$ (solid) and the true density at $T_5$ ($\times$), depending on proposal standard deviation $\tau_j$.

$$
\begin{align*}
\tau_j &= 0.1 \sqrt{T_j} \\
\tau_j &= 0.2 \sqrt{T_j}
\end{align*}
$$

Figure 1 shows the smoothed histograms of the $d$th component of $x$ at the five temperatures, based on single runs of the generalized resample-move algorithm using the random-walk Metropolis kernel $\Psi_j$ with proposal standard deviation $0.1 \sqrt{T_j}$ or $0.2 \sqrt{T_j}$ at temperature $T_j$. See Figures A8–A9 for a clearer view of the same histograms in separate plots, instead of a single plot.
Consider the histograms under the former choice of $\Psi_j$. The two histograms for $P_1$ and $P_2$ are essentially unimodal at $\mu_1$, those for $P_3$ and $P_4$ are bimodal, with a major mode at $\mu_1$ and a minor one at $\mu_2$, and that for the target $P_5$ is bimodal, although the weights of the two modes differ somewhat from the truth. These results, in agreement with the foregoing analysis of tempering, make clear how challenging it is to sample from such tempering distributions. As the temperature goes down to 1, the mode at $\mu_2$ emerges from being a negligible one, to a minor one, and eventually to a major one, all in the tail area of the other mode at $\mu_1$. The challenge is borne out in the results under the second choice of the kernel $\Psi_j$. The bulk of $P_3$ or $P_4$ appears to be adequately sampled, but the minor mode at $\mu_2$ is not well sampled in the tail of $P_3$ or $P_4$. These small discrepancies are then transferred into substantial ones for the sample of $P_5$, where the relative weights of the two modes are reversed.

In the following subsections and Appendix III, we present three simulation experiments to compare various algorithms, depending on different choices of Markov kernels, subsampling strategies, and tempering distributions.

### 3.2.2 Experiment on Markov kernels

We compare the 7 algorithms labeled 1–7, excluding dynamic algorithms, under two choices of the random-walk Metropolis kernel $\Psi_j$ with proposal standard deviation $0.1\sqrt{T_j}$ or $0.2\sqrt{T_j}$. The parameter $\ell_j$ is set to 10 and $\alpha_j$ to 10%. The initial value $x_{10} = \cdots = x_{50}$ is generated as a Gaussian vector with mean 0 and variance $I_6$. The sample sizes are set with $n = 40,000$ and $b = 4,000$. For equi-energy sampling, $u(x)$ is defined as $-\log\{(2\pi)^d/2p(x)\}$ and $(U_1, U_2, \ldots, U_5)$ are set to $(-9.25, -8.12, -5.12, 2.88, 24.12)$ such that $(U_{h+1} - U_h)/T_{m+1-h} \approx 1.13$ for $h = 1, \ldots, 5$ with $U_6 = 80.62$, in accord with the suggestion of Kou et al. (2006).

Figure 2 shows the squared biases and MSEs for the estimates of $E(x_{(d)})$ and $\text{var}(x_{(d)})$ under $P_5$ based on 200 repeated simulations, where $x_{(d)}$ is the $d$th component of $x$. The performances of the algorithms depend on whether the random-walk Metropolis kernel $\Psi_j$ is chosen with proposal standard deviation $0.1\sqrt{T_j}$ or $0.2\sqrt{T_j}$. For all the 6 algorithms using Markov moving, the results under the first choice of $\Psi_j$ with proposal standard deviation $0.1\sqrt{T_j}$ are substantially more accurate than
Figure 2: Squared bias (×) and MSE (○) for estimates of \( E(x_{(d)}) \) in first and third plots and \( \text{var}(x_{(d)}) \) in second and fourth plots under the target \( P_5 \), depending on proposal standard deviation \( \tau_j \). The dotted lines are placed at 1, 2, 3, and 4 times the MSE based on generalized resample-move.

under the second choice. A possible explanation is that using the proposal standard deviation \( 0.1 \sqrt{T_j} \) leads to better mixing near the mode at \( \mu_2 \), by the result that the scale of an optimal jumping rule is \( 2.4/\sqrt{d} = 0.98 \) times the scale of the target distribution (Gelman et al. 1996). The accuracy of sampling near this mode is crucial to obtaining a good sample for the target \( P_5 \), as discussed earlier.

For the first choice of \( \Psi_j \), generalized resample-move and independence Metropolis resampling MCMC have the smallest MSEs. For the second choice of \( \Psi_j \), independence Metropolis resampling MCMC and parallel tempering have the smallest MSEs. By Remark 3 in Section 2.4, there is, on average, a longer stretch of successive Markov moves starting from each accepted draw by resampling or swapping in independence Metropolis resampling MCMC or parallel tempering than in generalized resample-move. This difference may contribute to better performances of the former two algorithms under the second choice of the kernel \( \Psi_j \). As discussed above, Markov moving does not mix efficiently near the mode at \( \mu_2 \) for this choice of \( \Psi_j \), but the accuracy of sampling near \( \mu_2 \) is crucial.

The MSEs of importance resampling MCMC and equi-energy sampling are similar to each other, consistently with Remark 4 in Section 2.4. The performances of these two algorithms are comparable to or worse than that of generalized resample-move under the two choices of the kernel \( \Psi_j \).
3.2.3 Experiment on subsampling strategies

For the proposal standard deviation $0.2\sqrt{T_j}$, we study the following ways of improving accuracy by increasing the number of iterations and then perform subsampling:

- Make no change except redefining the kernel $\Psi_j$ as the transition from an initial value to the 10th iteration by random-walk Metropolis sampling.
- Increase the number of overall iterations in $i$ by 10 times, with the same $\Psi_j$ and $\ell_j = 10$ or $\alpha_j = 10\%$ as before, and then perform subsampling of the draws at the rate of 1 in 10, successively, for $(P_1, \ldots, P_m)$.
- Increase the number of overall iterations in $i$ by 10 times, with the same $\Psi_j$ as before but $\ell_j = 100$ or $\alpha_j = 1\%$, and then perform subsampling of the draws at the rate of 1 in 10, successively, for $(P_1, \ldots, P_m)$.

For convenience, the three strategies are referred to as ITER, SUB1, and SUB2. For generalized resample-move, the strategy ITER is equivalent to SUB2, both corresponding to increasing $\ell_j$ by 10 times to 100 with the same $k_j$ and then subsampling, whereas SUB1 involves increasing $k_j$ by 10 times with the same $\ell_j = 10$ and then subsampling. For algorithms labeled 4–7, the strategy ITER differs in a subtle manner from SUB2, although the expected numbers of operations of resampling or swapping are the same for both strategies.

In Appendix III, Figure A3 shows the squared biases and MSEs based on 200 repeated simulations. A summary of the results is as follows. Among the three strategies, SUB1 is most effective for parallel tempering, whereas SUB2 is most effective for resampling MCMC algorithms (including ITER for generalized resample-move). Moreover, if the 7 algorithms are compared with their best strategies of subsampling, independence Metropolis resampling MCMC and parallel tempering remain the most competitive, as in the simple case before the increase of iterations.

3.3 Potts model

3.3.1 Setup and difficulty

The Potts model is important in statistical physics and has been used in various applications such as imaging analysis and spatial statistics. Consider a 10-state Potts model on a $20 \times 20$ lattice with periodic boundary conditions in the absence of a magnetic field. Each observation $x$ corresponds to a collection of $K = 20^2$ spins.
Figure 3: Smoothed histograms of $u(x)/K$ at the temperatures $(T_1, T_2, \ldots, T_5)$ labeled as 1, 2, \ldots, 5 under the Potts model. Two vertical lines are placed at −1.7 and −0.85.

$(s_1, \ldots, s_K)$ on the lattice, where $s_i$ takes $q = 10$ possible values. At a temperature $T$, the density function of the Potts distribution is

$$Z^{-1} e^{-u(x)/T},$$

where $u(x) = -\sum_{i,j} 1\{s_i = s_j\}$, with $i \sim j$ indicating that sites $i$ and $j$ are nearest neighbors, and $Z = \sum_x \exp\{-u(x)/T\}$ is the normalizing constant. Statistically, the Potts distribution belongs to an exponential family, with canonical statistic $-u(x)$ and natural parameter $\beta = T^{-1}$. Let $U = E\{u(x)\}$ and $C = \text{var}\{u(x)\}$ under the Potts distribution. For notational simplicity, the dependency of $Z$, $U$, and $C$ on $T$ or $\beta$ is suppressed. Then $U = -(d/d\beta) \log Z$ and $C = (d^2/d\beta^2) \log Z$ by theory of exponential family. In statistical physics, $Z$ is called the partition function, $U$ is internal energy, and $C/T^2$ is specific heat (Newman & Barkema 1999).

A special case of the Potts model with two states ($q = 2$) is equivalent to the Ising model, where $u(x) = -\sum_{i,j} s_i s_j$ and each $s_i$ is either $-1$ or $1$. Like the Ising model, the Potts model on an infinite lattice exhibits a phase transition at the inverse temperature $\beta_c = T_c^{-1} = \log(1+\sqrt{q})$, about $1.426$ for $q = 10$. But the critical behavior is richer and more general than that of the Ising model (Wu 1982). As shown later by Figure 3, the histograms of $u(x)$, known as the energy histograms, are bimodal near the critical temperature $T_c$. In contrast, the energy histograms are unimodal, centered at different locations for different temperatures under the Ising model (e.g.,
Newman & Barkema 1999, Figure 8.3).

For illustration, we take \( m = 5 \) inverse temperatures \((T^{-1}_1, \ldots, T^{-1}_5) = (1.4, 1.4065, 1.413, 1.4195, 1.426)\), which are evenly spaced between 1.4 and 1.426. Figure 3 shows the smooth histograms of \( u(x)/K \) at the 5 temperatures, based on single runs of generalized resample-move and parallel tempering. See Figures A10–A11 for a clearer view of the same histograms in separate plots, instead of a single plot. The Markov kernel \( \psi_j \) is defined as a random sweep using the single-spin-flip Metropolis algorithm (Newman & Barkema 1999, Section 4.5.1). Each sweep consists of \( K \) iterations, 1 per spin, where each iteration involves randomly picking a spin \( s_i \), choosing a new value from the \( q - 1 \) remaining values, and then accepting or rejecting the new value by the Metropolis rule. The two sets of energy histograms obtained by generalized resample-move and parallel tempering appear similar to each other, but a careful examination shows that there are differences in the weights of the two modes.

At the higher temperature \( T_1 \) or \( T_2 \), the energy histogram per spin is essentially unimodal, centered about \(-0.85\), corresponding to the disordered phase. At the lower temperature \( T_3, T_4 \) or \( T_5 \), the energy histogram per spin becomes bimodal, with one mode still at \(-0.85\) and a new mode at \(-1.7\), corresponding to the ordered phase. As the temperature decreases, the mode located at about \(-1.7\) grows in its weight, from being a negligible one, to a minor one, and eventually to a major one, so that the spin system moves from the disordered phase to the ordered one. These changes in the modes are somewhat similar to those in Figure 1 for the bimodal Gaussian mixture, and hence sampling from the Potts distributions faces similar difficulties to those discussed in Section 3.2.1. On the other hand, there is a subtle difference between Figures 1 and 3. The former shows the histograms of a component \( x_{(d)} \) of \( x \), whereas the latter shows the histograms of \( u(x)/K \). For the Gaussian mixture in Section 3.2, the histograms of \( u(x) \) or \(-\log\{p(x)\}\) (not shown for space limitation) differ markedly from those of \( x_{(d)} \): they are positively skewed and unimodal except at temperature \( T_5 = 1 \), where the energy histogram is bimodal.

3.3.2 Simulation details

We compare the algorithms labeled 3–7 in Section 3.1, excluding sequential importance resampling, resample-move, and dynamic algorithms. Without sufficient
Markov moving, sequential importance resampling and resample-move perform poorly, relatively to other static algorithms. Instead, we include in the study two standard MCMC algorithms, applied separately to the Potts distributions at the selected temperatures. One is the single-spin-flip Metropolis algorithm mentioned above and the other is the Swendsen-Wang (1987) cluster algorithm, which is known to work more efficiently near the phase transition than the single-spin-flip Metropolis algorithm. For the algorithm labeled 3–7, the Markov kernel $\Psi_j$ corresponds to a random sweep using the single-spin-flip Metropolis algorithm. The Swendsen-Wang algorithm is used mainly to provide benchmark results. It is possible to combine resampling MCMC or parallel tempering with a cluster algorithm, with the Markov kernel $\Psi_j$ defined as the cluster update (e.g., Bittner & Janke 2011).

The initial value $x_{10} = \cdots = x_{50}$ is generated by randomly setting each spin. For equi-energy sampling, the cutoff energy levels $(U_1, U_2, \ldots, U_5)$ are set to $(-800, -606, -459, -348, -264)$ such that $U_{h+1}/U_h \approx 0.76$ for $h = 1, \ldots, 5$ with $U_6 = -200$. Such choices seem appropriate given that $(T_1, \ldots, T_m)$ are close to each other.

The sample sizes are set as described in Section 3.1 with $n = 400,000$ and $b = 40,000$. But, to obtain reasonably accurate results, we increase the number of overall iterations by 10 times and then perform subsampling for all the 7 algorithms under study. By the discussion at the end of Section 3.2.3, we employ different subsampling strategies for different algorithms as follows.

- For generalized resample-move labeled 3, redefine the kernel $\Psi_j$ by 10 random sweeps using the single-spin-flip Metropolis algorithm, without increasing the number of overall iterations in $i$. Take $\ell_j = 10, 100$ or 1000.
- For Metropolis sampling, Swendsen-Wang algorithm, resampling MCMC labeled 4–6 and parallel tempering labeled 7, use the single-sweep kernel $\Psi_j$, increase the number of overall iterations in $i$ by 10 times, and then perform subsampling at the rate of 1 in 10. For the latter 4 algorithms, take $\alpha_j = 10\%$, $1\%$ or $0.1\%$, corresponding to $\ell_j = 10, 100$, or 1000.

For convenience, the strategies are referred to as ITER and SUB. The strategy ITER can also be seen as subsampling. For $\ell_j = 100$, ITER is equivalent to increasing $\ell_j$ by 10 times to 1000 with the same $k_j$ and then subsampling.
3.3.3 Results with 5 tempering distributions

Consider simulations from the Potts model at the 5 temperatures specified earlier. Different algorithms appear to have their best performances, using different choices of \( \ell_j \) or \( \alpha_j \). Figure 4 shows the Monte Carlo means and standard deviations for the estimates of \(-U/K\), the internal energy per spin, and \(C/K\), the specific heat per spin multiplied by \(T^2\), for each algorithm with two best-performing choices of \( \ell_j \) or \( \alpha_j \) under the strategy ITER or SUB. The results are based on 100 repeated simulations. For resampling MCMC including generalized resample-move labeled 3-6, the most accurate estimates are obtained using \( \ell_j = 1000 \) or \( \alpha_j = 0.1\% \), whereas the estimates with the choice \( \ell_j = 100 \) or \( \alpha = 1\% \) have both greater variances and absolute biases. For parallel tempering, the most accurate estimates are obtained for \( \alpha_j = 10\% \), and the estimates are only slightly less accurate for \( \alpha_j = 1\% \).

Among the algorithms labeled -7, parallel tempering appears to yield the most accurate estimates, especially at temperatures \( T_1 \) (not shown), \( T_2 \), and \( T_3 \). The 4 resampling MCMC algorithms are ranked in their performances roughly as independence Metropolis resampling MCMC, generalized resample-move, equi-energy sampling, and importance resampling MCMC. The single-spin-flip Metropolis algorithm performs reasonably well, even better than the 4 resampling MCMC algorithms with the choice \( \ell_j = 100 \) or \( \alpha = 1\% \). But Metropolis sampling yields less accurate estimates, most noticeably at temperatures \( T_4 \) and \( T_5 \), than independence Metropolis resampling MCMC and generalized resample-move with \( \ell_j = 1000 \) or \( \alpha = 0.1\% \). A partial explanation is that the mode located at about \(-1.7\) in the energy histogram becomes non-negligible at temperature \( T_3 \) or lower. Then importance weights between the successive Potts distributions are sufficiently controlled, so that variance can be reduced by combining importance resampling and shorter than complete Markov moving. See the discussion on the choice of \( \ell \) in Section 2.2.

3.3.4 Results with more tempering distributions

Similarly to the Gaussian mixture example, we study the effect of using more finely spaced tempering distributions than the 5 specified ones, with a comparable total number of iterations (after subsampling described in Section 3.2.3):
Figure 4: Summary of estimates of $-U/K$ in top plots and $C/K$ in bottom plots for 7 algorithms including Metropolis ($m$) and Swendsen-Wang ($sw$) under the strategy ITER or SUB. For each vertical bar, the center indicates Monte Carlo mean ($\ell_j = 100$ or $\alpha_j = 1\%$, $\times: \ell_j = 1000$ or $\alpha_j = .1\%$) for resampling MCMC algorithms labeled 3–6 and $\alpha_j = 10\%$ for parallel tempering labeled 7), and the radius indicates Monte Carlo standard deviation. For comparison, a horizontal line is placed at the Monte Carlo mean of Swendsen-Wang. A triangle ($\triangle$) is placed at the upper side of each bar, with the distance from the center equal to the Monte Carlo standard deviation of Metropolis.

- 9 distributions at temperatures ($t_1, t_3, t_6, t_8, t_{10}, t_{12}, t_{14}, t_{17}, t_{19}$), with the number of iterations ($n + 9b, n + 8b, \ldots, n + b$) for static algorithms 3–6, where $n = 10b$ and $b = 20000$ is the nearest integer that is divisible by 1000 and no smaller than $40000 \times \sum_{j=1}^{5} (10 + j)/\sum_{j=1}^{9} (10 + j) \approx 19260$.

- 12 distributions at temperatures ($t_1, t_2, t_3, t_4, t_5, t_8, t_{10}, t_{12}, t_{14}, t_{16}, t_{18}, t_{19}$), with the number of iterations ($n + 12b, n + 11b, \ldots, n + b$) for static algorithms 3–6, where $n = 10b = 140000$, determined as in the first case.

- 18 distributions at temperatures ($t_1, \ldots, t_5, t_7, \ldots, t_{19}$), with the number of iterations ($n + 18b, n + 17b, \ldots, n + b$) for static algorithms 3–6, where $n = 10b = 80000$, determined as in the first case.
The total number of iterations for parallel tempering is set to match that of the other algorithms. The 19 inverse temperatures used are \((t_1^{-1}, t_2^{-1}, \ldots, t_{19}^{-1}) = (1.4, 1.40325, 1.4065, 1.408667, 1.40975, 1.410833, 1.411917, 1.413, 1.414084, 1.415167, 1.41625, 1.417333, 1.418416, 1.4195, 1.420583, 1.421667, 1.42275, 1.423833, 1.426)\). These temperatures are specified manually, but mimicking an adaptive approach for choosing distributions in SMC (e.g., Jasra et al. 2011). For example, the 9 temperatures are chosen by examining the relative variances of importance weights (described below) from the repeated simulations with 5 temperatures and placing additional temperatures between those associated with large relative variances.

For the generalized resample-move algorithm, the distance between two successive distributions, \(P_{j-1}\) and \(P_j\), can be assessed by the relative variance of the unnormalized importance weights:

\[
RV_j = \frac{1}{n_j-1} \sum_{i=1}^{n_j-1} \frac{(w_i^{(j-1)})^2 - \bar{w}^{(j-1)})^2}{\bar{w}^{(j-1)}},
\]

where \(w_i^{(j-1)} = q_j(x_{j-1,i})/q_{j-1}(x_{j-1,i})\) and \(\bar{w}^{(j-1)} = \frac{1}{n_j-1} \sum_{i=1}^{n_j-1} w_i^{(j-1)}\). The relative variance is directly related to the effective sample size commonly used (Liu & Chen 1998; Del Moral et al. 2006), \(ESS_j = \frac{(\sum_{i=1}^{n_j-1} (w_i^{(j-1)})^2)^2}{\sum_{i=1}^{n_j-1} (w_i^{(j-1)})^2} = n_j-1/(1 + RV_j)\). Moreover, there is an interesting interpretation for summing \(RV_j\) over \(j\). In fact, \(\bar{w}^{(j-1)}\) serves as an estimator of \(Z_j/Z_{j-1}\), where \(Z_{j-1}\) and \(Z_j\) are the normalizing constants of \(q_{j-1}(\cdot)\) and \(q_j(\cdot)\) respectively. By the delta method, if \(\{x_{j-1,1}, \ldots, x_{j-1,n_{j-1}}\}\) were independent from \(P_{j-1}\), then \(RV_j/n_{j-1}\) is asymptotically
the variance of $\log \tilde{w}^{(j-1)}$. Therefore, if the dependency were ignored in and between the samples, then $\text{RV}_2/n_1 + \cdots + \text{RV}_j/n_{j-1}$ would approximately be the variance of $\log \tilde{w}^{(1)} + \cdots + \log \tilde{w}^{(j-1)}$, an estimator of $\log(Z_j/Z_1)$.

Figure 5 shows the cumulative relative variances $\text{RV}_2 + \cdots + \text{RV}_j$ and the effective sample fractions $\text{ESS}_j/n_{j-1}$, with $\text{RV}_j$ averaged from 100 repeated simulations. Compared with the case of 5 temperatures, using 9, 12, or 18 temperatures significantly reduces the relative variances and leads to effective sample fractions 80% or higher. However, the part of Monte Carlo variability from importance resampling, represented by $v_1[r\{h_t - E_2(g)\}]$ in (1), can still be large. See Appendix IV for discussion.

Figure 6 shows the Monte Carlo means and standard deviations for the estimates of $-U/K$ and $C/K$ at temperature $T = 1.413^{-1}$, starting from which the mode corresponding to the ordered phase becomes non-negligible for the spin system (see

Figure 6: Summary of estimates of $-U/K$ in top plots and $C/K$ in bottom plots at the temperature $T = 1.413^{-1}$ for $m = 5, 9, 12, 18$ tempering distributions. A horizontal line is placed at the Monte Carlo mean of Swendsen-Wang for $m = 5$. The results for Metropolis and Swendsen-Wang for $m = 9, 12, 18$ are based on shorter iterations than for $m = 5$, similarly to the other algorithms.
Figure 3). The algorithms are used with their best strategies of subsampling studied in Section 3.3.3: ITER with $\ell_j = 1000$ for generalized resample-move, SUB with $\ell_j = 1000$ for resampling MCMC labeled 4–6, and SUB with $\ell_j = 10$ for parallel tempering. The results are based on 100 repeated simulations.

Among the algorithms labeled 3–7, parallel tempering appears to yield the most accurate estimates, for $m = 5$ as seen in Section 3.3.3 and for larger $m = 9, 12, 18$. In fact, parallel tempering performs almost the same, in terms of both Monte Carlo means and variances, for $m$ ranging from 5 to 18. The performances of the 4 resampling MCMC algorithms including generalized resample-move show no significant improvement or even deteriorate with greater variances as $m$ increases from 5 to 18. Therefore, it seems not worthwhile using more tempering distributions, at the cost of a smaller sample size per distribution, in this example.

In Appendix IV, we provide additional discussion on the comparison of algorithms between the Potts model and the Gaussian mixture example.

4 Summary

There are three main factors for both implementation and performances of resampling MCMC and parallel tempering: tempering distributions, Markov kernels, and resampling rates or swapping probabilities. Our basic analysis and simulation studies suggest a number of interesting findings in the difficult situation where new modes emerge in the tails of previous tempering distributions.

If the Markov kernels lead to fast mixing or even locally so toward restricted distributions, then generalized resample-move tends to perform the best among all the algorithms studied. If the Markov kernels lead to slow mixing, without even converging fast to restricted distributions, then parallel tempering tends to perform the best. The performance of independence Metropolis resampling MCMC is often a close second best. Importance resampling MCMC and equi-energy sampling perform similarly to each other, often worse than independence Metropolis resampling MCMC.

To improve accuracy in the case of slow-mixing Markov chains, different strategies of increasing iterations and subsampling seem to be effective for different algorithms. For resampling MCMC algorithms, it is effective to keep the size of resampling, in-
crease the lengths of Markov moving, and then perform subsampling for resampling MCMC algorithms. For parallel tempering, it is effective to keep the probability of swapping between adjacent chains, say 10% in our examples, increase the number of overall iterations, and then perform subsampling.

There are various open problems raised from our work. First, for generalized resample-move, further research is needed to develop adaptive choices of tempering distributions, Markov kernels, and resampling rates, using techniques from adaptive MCMC and SMC. In fact, there are parallel subchains simulated for each tempering distribution. This unique, rich structure remains to be fully exploited. Second, for resampling MCMC, it is interesting to study possible ways to exchange information between tempering distributions as in parallel tempering, instead of feeding information unidirectionally. Finally, different algorithms seem to have advantages in different settings as in our simulation studies. It is important to further study this phenomenon and develop appropriate guidelines.

**Supplementary Materials**

**Appendices:** (I) Technical details, (II) Example on multimodal bivariate Gaussian mixtures, (III) Additional results for Section 3.2, (IV) Additional discussion for Sections 3.2–3.3, (V) Additional figures for Sections 3.2–3.3. (RMCMC-appendix.pdf)

**Computer codes:** C codes to perform the simulations on the multivariate Gaussian mixture in Section 3.2 and the Potts model in Section 3.3. Documentations of the codes are also provided. (RMCMC-codes.tar.gz, GNU zipped tar file)

**References**

Andrieu, C., Jasra, A. and Doucet, A. (2011) “On nonlinear Markov chain Monte Carlo,” *Bernoulli*, 17, 987–1014.

Atchadé, Y.F. (2009) “Resampling from the past to improve on Monte Carlo samplers,” *Far East Journal of Theoretical Statistics*, 27, 81-99.

Atchadé, Y.F. (2010) “A cautionary tale on the efficiency of some adaptive Monte Carlo schemes,” *Annals of Applied Probability*, 20, 841-868.
Bittner, E. and Jankel, W. (2011) “Parallel-tempering cluster algorithm for computer simulations of critical phenomena,” Physical Review E, 84, 036701.
Brockwell, A., Del Moral, P., and Doucet, A. (2010) “Sequentially interacting Markov chain Monte Carlo,” Annals of Statistics, 38, 3387–3411.
Chopin, N. (2004) “Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference,” Annals of Statistics, 32, 2385–2411.
Del Moral, P., Doucet, A., and Jasra, A. (2006) “Sequential Monte Carlo samplers,” Journal of the Royal Statistical Society, Ser. B, 68, 411-436.
Frantz, D.D., Freeman, D.L., and Doll, J.D. (1990) “Reducing quasi-ergodic behavior in Monte Carlo simulations by J-walking: Applications to atomic clusters,” Journal of Chemical Physics, 93, 2769–2784.
Gelman, A., Roberts, G.O., and Gilks, W.R. (1996) “Efficient Metropolis jumping rules,” in Bayesian statistics 5, eds, J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith, New York: Oxford University Press, 599–607.
Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2003) Bayesian Data Analysis (2nd edition), New York: Chapman & Hall.
Geyer, C.J. (1991) “Markov chain Monte Carlo maximum likelihood,” in Computing Science and Statistics: Proceedings of 23rd Symposium on the Interface, ed, E.M. Keramidas, Fairfax, VA: Interface Foundation, 156–163.
Gilks, W.R. and Berzuini, C. (2001) “Following a moving target – Monte Carlo inference for dynamic Bayesian models,” Journal of the Royal Statistical Society, Ser. B, 63, 127-146.
Gordon, N., Salmond, D.J., and Smith, A.F.M. (1993) “Novel approach to nonlinear/ non-Gaussian Bayesian state estimation,” IEE Proceedings F, 140, 107-113.
Hukushima K. and Nemoto, K. (1996) “Exchange Monte Carlo method and application to spin glass simulations,” Journal of Physical Society of Japan, 65, 1604–1608.
Jasra, A., Stephens, D.A., Doucet, A., and Tsagaris, T. (2011) “Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo,” Scandinavian Journal of Statistics, 38, 1-22.
Kong, A., Liu, J.S., and Wong, W.H. (1994) “Sequential imputations and Bayesian missing data problems,” *Journal of the American Statistical Association*, 89, 278-288.

Kou, S.C., Zhou, Q., and Wong, W.H. (2006) “Equi-energy sampler with applications in statistical inference and statistical mechanics” (with discussion), *Annals of Statistics*, 34, 1581-1619.

Liang, F. and Wong, W.H. (2001) “Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models,” *Journal of the American Statistical Association*, 96, 653-666.

Liu, J.S. (2001) *Monte Carlo Strategies in Scientific Computing*, New York: Springer.

Liu, J.S. and Chen, R. (1998) “Sequential Monte Carlo methods for dynamic systems,” *Journal of the American Statistical Association*, 93, 1032-1044.

Liu, J.S., Chen, R., and Wong, W.H (1998) “Rejection control and sequential importance sampling,” *Journal of the American Statistical Association*, 93, 1022-1031.

Machta, J. (2010) “Population annealing with weighted averages: A Monte Carlo method for rough free-energy landscapes,” *Physical Review E*, 82, 026704.

Newman, M.E.J. and Barkema, G.T. (1999) *Monte Carlo Methods in Statistical Physics*, New York: Oxford University Press.

Robert, C. and Casella, G. (2005) *Monte Carlo Statistical Methods*, New York: Springer.

Rubin, D.B. (1987) “A Noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm, Comment on Tanner and Wong (1987),” *Journal of the American Statistical Association*, 52, 543-546.

Swendsen, R.H. and Wang, J.S. (1987) “Nonuniversal critical dynamics in Monte Carlo simulations,” *Physical Review Letter*, 58, 86–88.

Tan, Z. (2006) “Monte Carlo integration with acceptance-rejection,” *Journal of Computational and Graphical Statistics*, 15, 735–752.

Woodard, D.B., Schmidler, S.C., and Huber, M. (2009) “Sufficient conditions for torpid mixing of parallel and simulated tempering,” *Electronic Journal of Probability*, 14, 780-804.

Wu, F.Y. (1982) “The Potts model,” *Reviews of Modern Physics*, 54, 235–268.