Adaptive Importance Tempering: A flexible approach to improve computational efficiency of Metropolis Coupled Markov Chain Monte Carlo algorithms on binary spaces

R. Alexander Valencia-Sanchez [1], Jeffrey S. Rosenthal[1],
Yasuhiro Watanabe [2], Hirotaka Tamura[3] & Ali Sheikholeslami[4]

Abstract

Based on the algorithm Informed Importance Tempering (IIT) proposed by Li et al. (2023) we propose an algorithm that uses an adaptive bounded balancing function. We argue why implementing parallel tempering where each replica uses a rejection free MCMC algorithm can be inefficient in high dimensional spaces and show how the proposed adaptive algorithm can overcome these computational inefficiencies. We present two equivalent versions of the adaptive algorithm − A-IIT and SS-IIT − and establish that both have the same limiting distribution, making either suitable for use within a parallel tempering framework. To evaluate performance, we benchmark the adaptive algorithm against several MCMC methods: IIT, Rejection free Metropolis-Hastings (RF-MH) and RF-MH using a multiplicity list. Simulation results demonstrate that Adaptive IIT identifies high-probability states more efficiently than these competing algorithms in high-dimensional binary spaces with multiple modes.

Keywords: Markov chain Monte Carlo, adaptive Markov chain, computational methods, Informed proposals, Parallel Tempering.

1 Introduction

Consider a probability mass function (PMF) π over a finite discrete state space $\mathcal{S}$. Given a measurable function g, we want to compute an estimate of $\mathbb{E}_\pi(g(X))$. Another way to look at this problem is considering that we have a non-negative function π defined over a finite discrete space $\mathcal{S}$ and we want to obtain samples from a probability distribution which PMF is proportional to the function π for which we don't know the normalizing constant.

A standard approach to approximate the desired value is to use a Markov Chain Monte Carlo (MCMC) algorithm which proposes to build a Markov chain $\{X_n\}$ that converges to the target distribution π, get N samples from the chain and use them to create a Monte Carlo estimate.

[1]Department of Statistical Sciences, University of Toronto, Canada
[2]Fujitsu Limited, Kanagawa, Japan
[3]DXR Laboratory Inc., Kanagawa, Japan
[4]Department of Electrical and Computer Engineering, University of Toronto, Canada

An example of a MCMC algorithm is Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) which defines a way to define a Markov chain that converges to the desired target distribution by using the un-normalized function $\pi(x)$ and a proposal distribution $\mathcal{Q}(y \mid x)$.

Rejection of some proposed states is an important part of this algorithm, in some settings it's been shown that an optimal implementation of the algorithm will reject a bit more than 75% of the proposals (Gelman et al., 1997; Roberts & Rosenthal, 2001; Neal et al., 2012). The rejection rate can also be very high for high-dimensional state spaces which results in low efficiency for the algorithm (Li et al., 2023).

These rejections might be seen as an inefficiency of the algorithm as we are performing computations that don't translate in new information for the Markov chain. To address this situation, it is possible to define a Rejection-Free version of the MH algorithm (Rosenthal et al., 2020). This algorithm defines a new Markov chain that always accepts a move to a proposed state and allows to make inference of the target distribution π. The algorithm identifies the set of neighbor states of the current state, evaluates the target distribution on each neighbor state and modify the proposal probability with the computed information.

This Rejection-Free MH algorithm is one instance of the class of Informed Importance Tempering (IIT) algorithms proposed by Zanella (2020) as it uses information of the local neighbors of the current state to modify the proposal distribution. The advantage of IIT become clear when we have access to parallel computing. It has been shown that this class of algorithms performs well in high dimensional settings (Zhou & Smith, 2022) and that it is very simple to implement an IIT version of different MCMC algorithms (Li et al., 2023)

In this work we explore how we can create an algorithm that combines IIT with Parallel Tempering (PT) and show it's effectiveness using simulations in binary multi modal state spaces. The most direct way of combining the two algorithms was already discussed by Rosenthal et al. (2020, sec. 6) where they implement parallel tempering where each replica uses a rejection free Metropolis Hastings algorithm and they modify the replica swap probability to correct the bias generated by the rejection free chains. This implementation, although straightforward, increases the number of computations needed to try a replica swap in a factor proportional to the dimension of the space.

In order to address this computational inefficiency we propose an algorithm called Adaptive IIT (A-IIT). This algorithm creates a chain with Markovian adaptations (Roberts & Rosenthal, 2007) that uses the information of the neighbors of the current state to propose a rejection free move to the next state but makes the computation of the probability of a replica swap more efficient as it doesn't need additional factors to compute the replica swap probability.

1.1 Overview of the paper

In section 2 we present a brief summary of MCMC, rejection free algorithms and Informed Importance Tempering that will be useful for the definition of the adaptive algorithm.

In section 3 we present the adaptive algorithm (A-IIT) and we prove that the Markov chain defined by this algorithm converges to a probability distribution and we show how to use this adaptive Markov chain to obtain an estimator for $\mathbb{E}_\pi(f(X))$.

In section 4 we briefly discuss why using a rejection-free Markov chain with parallel tempering creates additional computations for the computation of the replica swap probability and how we can avoid these extra computations without affecting the convergence of the chain.

In section 5 we present more details on the implementation of the algorithms in binary multimodal spaces. We present the second version of the algorithm: Single step IIT (SS-IIT), we explain why it's equivalent to A-IIT and how we can use both in an implementation with Parallel Tempering to reduce the number of computations needed to update the Markov chain. We present a discussion on the chosen balancing function $h(r) = \sqrt{r}$.

In section 6 we present the details of the simulations performed and show the results in binary multimodal state spaces. We compare the performance of A-IIT with other MCMC algorithms. We show that in low dimensional state spaces MH still performs better than A-IIT in terms of convergence in Total Variation Distance (TVD) and in terms of how many iterations it takes for the chains to visit the modes. We present results in a binary space of dimension 3000 where A-IIT performs better than MH and IIT, we discuss how we can attribute this improvement to both the use of the balancing function $h(r) = \sqrt{r}$ and the use of a multiplicity list to reduce the number of computations needed to update the Markov chain. Additional details of these simulations and results on state spaces of other dimension (1000, 5000, 7000) can be found in the appendix.

2 MCMC algorithms

To build an estimate of $\mathbb{E}_\pi(f(X))$ Markov Chain Monte Carlo proposes to build a Markov chain $\{X_n\}$ that converges to the target distribution π, get N samples from that distribution and then use the Monte Carlo method to estimate the desired value as follows.

$$\mathbb{E}_\pi(f(X)) \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

in general we can write this estimator in the following form.

$$\widehat{\mathbb{E}_\pi(f(X))} = \frac{\sum_{i=1}^{N} w_i(X_i) f(X_i)}{\sum_{i=1}^{N} w_i(X_i)} \tag{1}$$

Where $w_i(X_i)$ is the weight assigned to the state $X_i \in S$ when the Markov chain visits that state at iteration i.

To build this estimator we can see that we need two components: First a process to explore the state space and while exploring, a function to define the weight of each visited state.

Metropolis-Hastings algorithm, for example, assigns $w(X_i) = 1$ to every state X_i every time that the chain is visited. The exploration of the state space depends on the proposal distribution and the estimation of the weight depends on the rejections happening during that exploration.

If we modify the exploration process then the weight estimation also has to be modified so the chain still converges to the desired target distribution.

2.1 Rejection free algorithms

As stated before, an optimal implementation of Metropolis-Hastings algorithm will consistently reject proposed states. Rosenthal et al. (2020) present a Rejection-free version of MH algorithm. Consider a Markov chain $\{X_n\}$ with transition probabilities $P(y|x) = P(X_{n+1} = y|X_n = x)$. Then we can build the corresponding jump Markov chain $\{J_n\}$ that has transition probabilities given by

$$\widehat{P}(y|x) = P(J_{n+1} = y|J_n = x) = \frac{P(y|x)}{Z(x)}$$

Where $Z(x)$ represents the probability that the chain "escapes" state x in a single iteration.

$$Z(X) = \sum_{y \neq x} P(y|x) = 1 - P(x|x)$$

The jump chain inherits important properties from the original chain, such as being irreducible and reversible. Moreover if the original chain's stationary distribution is π then the stationary distribution of the jump chain is given by $\widehat{\pi}(x) = Z(x)\pi(x)$.

Along with the jump chain authors define the multiplicity list $\{M_k\}$ that represents the number of iterations the original chain stayed in its current state before accepting a move to a different state. The distribution of M_k given J_k is equal to the distribution of $1 + G$ where $G \sim \text{Geometric}(Z(x))$.

Authors present two possible ways to define the weight of a state when using a rejection free algorithm. In particular theorem 8 and 12 of Rosenthal et al. (2020) prove that using $w_k(X_k) = M_k$ or $w_k(X_k) = 1/Z(X_k)$ in equation 1 creates a consistent estimator for $\mathbb{E}_\pi(f(X))$.

The main advantage of using the multiplicity list is that it allows to recover samples of the original Markov chain. This aspect of the multiplicity list is an important part of the implementation of A-IIT that will be presented in the next section.

2.2 Informed Importance Tempering

A generalization of rejection-free algorithms is presented by Zanella (2020) introducing the concept of balancing function. This function measures the importance of neighbor states and uses this information to modify the way the chain explores the state space.

To build a Markov chain that follows IIT we will start with a proposal distribution $\mathcal{Q}(y|x)$ that defines the set of neighbor states of a state x as $\mathcal{N}_x = \{y : \mathcal{Q}(y|x) > 0\}$. Then we consider a balancing function, which is a function $h : (0, \infty) \to (0, \infty)$ that satisfies $h(r) = rh(r^{-1})$. For the states y in the neighborhood of x we consider a function $\alpha(x, y) \geq 0$ that represents the preference of choosing y, which then is used to modify the proposal weight of that state $\eta(y|x)$.

$$\eta_h(y|x) = \mathcal{Q}(y|x)\alpha_h(x,y), \qquad \alpha_h(x,y) = h\left(\frac{\pi(y)\mathcal{Q}(x|y)}{\pi(x)\mathcal{Q}(y|x)}\right) \tag{2}$$

Algorithm 1 shows the implementation of IIT.

Algorithm 1 Informed importance tempering (Naive IIT)

```
Initialize $X_0$
for $k$ in 0 to $K$ do
    For each $Y \in \mathcal{N}_{X_k}$, calculate
        $$\eta_h(Y|X_k) = \mathcal{Q}(Y|X_k)h\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right)$$
    Calculate
        $$Z_h(X_k) = \sum_{Y\in\mathcal{N}_{X_k}} \eta_h(Y|X_k)$$
    Set $w_k(X_k) \leftarrow 1/Z_h(X_k)$.
    Choose the next state $Y \in \mathcal{N}_{X_k}$ such that
        $$P(X_{k+1} = Y|X_k) \propto \mathcal{Q}(Y|X_k)h\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right)$$
end for
```

This is very similar to how rejection-free Metropolis-Hastings algorithm is implemented but we can choose to use a different function, as long as the function satisfies the balancing function equality this guarantees that the resulting chain is reversible and hence converges to the desired target distribution.

Choosing a different balancing function allows the algorithm to modify the way the Markov chain explores the state space. If we use the balancing function $h(r) = \min\{1, r\}$ then IIT is equivalent to Rejection-Free Metropolis-Hastings (Rosenthal et al., 2020), but there's no need to use a bounded balancing function, authors mention other valid balancing functions such as $h(r) = \sqrt{r}$ (*squared root*), $h(r) = 1 \wedge r$ (*min*), $h(r) = 1 \vee r$ (*max*), $h(r) = (1 \wedge re^{-c}) \vee (r \wedge e^{-c})$. If we choose a different balancing function, this modifies the proposal weight of the states η_h which then modifies the exploration of the state space.

In figure 1 we compare how these balancing functions measure the importance of neighbor states. Consider a uniform proposal distribution $\mathcal{Q}$ and two neighbor states y_1 and y_2 such that $\pi(y_1)/\pi(x) = 2$ and $\pi(y_2)/\pi(x) = 3$. Using the *min* balancing function both of these states have the same importance so the probability that the Markov chain chooses any of them in the next iteration is the same. However using any of the other two balancing functions the difference in probability is translated in one of the states being more likely to be chosen. In this example using *max* balancing function makes y_2 50% more likely to be chosen than y_1 while using *squared root* balancing function makes y_2 22% more likely to be chosen than y_1

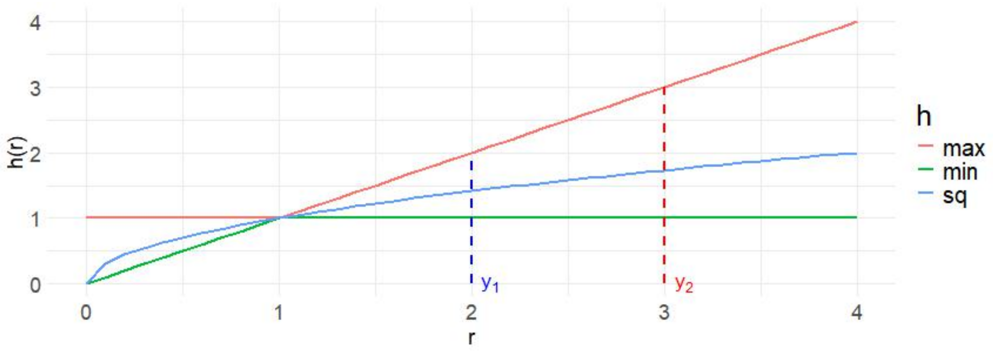

Figure 1: Comparison of balancing functions. Showing the possible values that the expression $h\left(\frac{\pi(y)}{\pi(x)}\right)$ may take for 2 neighbors of x: y_1 with lower probability and y_2 with higher probability.

3 Adaptive IIT

In general given a non-decreasing function $f(r)$ that we want to use to measure the importance of the states, we can create a version that is bounded and satisfies the balancing function property following what's presented by (Zanella, 2020).

For a fixed bounding constant $\gamma > 1$ we first take the bounded function $f_\gamma(r) = \frac{1}{\gamma}\min\{\gamma, f(r)\}$ which takes values in $(0, 1)$ and define:

$$h_\gamma(r) = \min\left\{f_\gamma(r), rf_\gamma\left(\frac{1}{r}\right)\right\} \tag{3}$$

Then h_γ is a balancing function taking values in $(0, 1)$.

The choice of bounding constant impacts the measurement of importance. Using a small constant may result in not fully using the information of the neighbors, while using a big constant may cause numerical errors during the implementation.

For a given state space $\mathcal{S}$, target probability distribution π and non-decreasing function f, we define the minimal fully informative constant as $\gamma^* = \max\{f(\pi(y)/\pi(x)); x \in \mathcal{S}, y \in \mathcal{N}(x)\}$ since this is the minimum constant that assigns a different weight to all states with different probabilities. Without previous knowledge of the problem, finding this constant would be unfeasible when the dimension of $\mathcal{S}$ is big. To overcome this problem we propose an adaptive algorithm that updates the bounding constant as it explores the state space.

For the adaptive algorithm first we consider the basis function f that will be used to measure the importance of the states. Following equation 3 and considering a constant γ we can create a bounded balancing function that can be used to define the Markov chain kernel P_γ as follows:

$$
\begin{aligned}
&P_\gamma(Y|X) = \mathcal{Q}(Y|X)h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right) \qquad \text{for } Y \neq X, \\
&P_\gamma(X|X) = 1 - \sum_{y\in\mathcal{N}_X} P_\gamma(y|X).
\end{aligned} \tag{4}
$$

With an adequate choice of proposal distribution $\mathcal{Q}$ this kernel is irreducible and aperiodic. For any fixed $\gamma > 0$, since h_γ is a balancing function then the chain is reversible and $\pi(x)$ is the stationary distribution.

Proposition 1. *Let π be the target probability distribution, $\gamma \geq 1$ be fixed and consider the Markov chain kernel P_γ defined in equation 4 where $\mathcal{Q}$ is a proposal distribution that makes the kernel irreducible and aperiodic and h_γ is a balancing function taking values in $(0,1)$. Then P_γ has π as a stationary distribution and it's ergodic for π.*

Proof. Since h_γ is a balancing function then $h_\gamma(r) = rh_\gamma(1/r)$. Using this property we obtain

$$
\begin{aligned}
\pi(X)P_\gamma(Y|X) &= \pi(X)\mathcal{Q}(Y|X)h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right) \\
&= \pi(X)\mathcal{Q}(Y|X)\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}h_\gamma\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right) \\
&= \pi(Y)\mathcal{Q}(X|Y)h_\gamma\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right) \\
&= \pi(Y)P_\gamma(X|Y)
\end{aligned}
$$

So P_γ has π as a stationary distribution and since the kernel is also aperiodic and irreducible over a finite state space $\mathcal{S}$ then $\mathbb{P}_\gamma$ it's ergodic for π □

We can then define the rejection-free kernel equivalent to equation 4 as follows:

$$
\begin{aligned}
\widehat{P}_\gamma(Y|X) &= \frac{1}{Z_h(X)}\mathcal{Q}(Y|X)h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right) \qquad \text{for } Y \neq X, \\
\widehat{P}_\gamma(X|X) &= 0.
\end{aligned} \tag{5}
$$

Where $Z_h(X)$ is the probability of escaping state X, defined as:

$$
Z_h(X) = \sum_{Y\in\mathcal{N}_X} \mathcal{Q}(Y|X)h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right) \tag{6}
$$

Proposition 2. *Let $\pi(\cdot)$ be the target probability distribution, $\gamma \geq 1$ be fixed and consider the rejection-free Markov chain kernel $\widehat{P}_\gamma$ defined in equation 5 where $\mathcal{Q}$ a proposal distribution that* *makes the kernel irreducible and aperiodic and h_γ is a bounded balancing function. Then $\widehat{P}_\gamma$ has $\pi^*(\cdot) = \pi(\cdot)Z_h(\cdot)$ as a stationary distribution and it's ergodic for π^*.*

Proof. The proof is very similar to the one before and relies on the property of a balancing function.

$$\begin{aligned}
\pi(X)Z_h(X)\widehat{P}_\gamma(Y|X) &= \pi(X)Z_h(X)\frac{1}{Z_h(X)}\mathcal{Q}(Y|X)h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right)\\
&= \pi(X)\mathcal{Q}(Y|X)\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}h_\gamma\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right)\\
&= \pi(Y)\mathcal{Q}(X|Y)h_\gamma\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right)\\
&= \pi(Y)Z_h(Y)\frac{1}{Z_h(Y)}\mathcal{Q}(X|Y)h_\gamma\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right)\\
&= \pi(Y)Z_h(Y)P_\gamma(X|Y)
\end{aligned}$$

So $\widehat{P}_\gamma$ has π^* as a stationary distribution and since the kernel is also aperiodic and irreducible over a finite state space $\mathcal{S}$ then $\widehat{P}_\gamma$ it's ergodic for π. □

Following the notation presented by Roberts & Rosenthal (2007), we consider a chain with Markovian adaptations $\{(X_n, \gamma_n)\}$ where at time n we use the Markov kernel defined by γ_n.

To define how the kernels will be adapted, we consider a function $M : \mathcal{S} \to \mathbb{R}$ defined as:

$$M(X) = \max_{Y\in\mathcal{N}(X)}\left\{f\left(\frac{\pi(Y)\mathcal{Q}(X|Y)}{\pi(X)\mathcal{Q}(Y|X)}\right), f\left(\frac{\pi(X)\mathcal{Q}(Y|X)}{\pi(Y)\mathcal{Q}(X|Y)}\right)\right\} \tag{7}$$

This function checks the possible values that the ratio of probabilities can take when evaluated using function f.

With this function we consider the set $\mathcal{Y} = \{M(X)|X \in \mathcal{S}\}$ and we consider all the Markov chain kernels P_γ for each $\gamma \in \mathcal{Y}$.

Using these kernels the transition probabilities of the adaptive algorithm are given by

$$P(X_{n+1} = y|X_n = x, \Gamma_n = \gamma) = P_\gamma(y|x) \tag{8}$$

We initialize the bounding constant at $\gamma_0 = 1$ and every time we visit a new state $X_n \in \mathcal{S}$ we update the constant considering the information of the neighbor states $\gamma_n = \max\{M(X_n), \gamma_{n-1}\}$ and use the corresponding adapted Markov kernel to choose the next state. Algorithm 2 shows the steps to implement this adaptive Markov chain.

Although the algorithm considers adaptations can happen every iteration, since we are considering a finite state space $\mathcal{S}$, then the set of bounding constants $\mathcal{Y}$ is finite so this becomes a finite adaptation algorithm. Finite adaptations are one of the safe ways to create adaptive Markov chains that are ergodic (Roberts & Rosenthal, 2007).

Algorithm 2 Adaptive IIT

```
Initialize $X_0$ and $\gamma_0 = 1$
for $k$ in 0 to $K$ do
                                            ▷ Step 1: Update bounding constant
    For each $Y \in \mathcal{N}_{X_k}$, calculate
        $$f\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right), f\left(\frac{\pi(X_k)\mathcal{Q}(Y|X_K)}{\pi(Y)\mathcal{Q}(X_k|Y)}\right)$$
    Compute $M(X_k)$ as shown in equation 7
    Update $\gamma_{k+1} = \max\{\gamma_k, M(X_k)\}$
    Define $h_\gamma$ using the updated constant $\gamma_{k+1}$ following equation 3
                          ▷ Step 2: Use the adapted Markov kernel to update the chain
    Calculate
        $$Z_h(X_k) = \sum_{y \in \mathcal{N}_{X_k}} \mathcal{Q}(Y|X_k) h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right)$$
    Set $w_k(X_k) \leftarrow 1 + \text{Geometric}(Z_h(X_k))$.
    Choose the next state $Y \in \mathcal{N}_{X_k}$ such that
        $$P(X_{k+1} = Y|X_k) = \frac{1}{Z_h(X_n)} \mathcal{Q}(Y|X_k) h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right)$$
end for
```

Proposition 3. *For a function $f : (0,\infty) \to (0,\infty)$ consider the chain with Markovian adaptations $\{X_n, \gamma_n\}$ defined over a finite state space $\mathcal{S}$ that follows algorithm 2 where $\mathcal{Y}$ is defined above and for each $\gamma \in \mathcal{Y}$ let P_γ be the Markov chain kernel as defined in equation 4 that uses the balancing function h_γ defined in equation 3. Assume that for each fixed $\gamma \in \mathcal{Y}$ the kernel P_γ is ergodic $\lim_{n\to\infty} ||P^n_\gamma(x,\cdot) - \pi(\cdot)|| = 0$ for all $\gamma \in \mathcal{Y}$ and $x \in \mathcal{S}$, then the adaptive algorithm is ergodic.*

Proof. Let $\mathcal{M} := \max_{\gamma\in\mathcal{Y}}\{\gamma\}$, there exist $X' \in \mathcal{S}$ such that $M(X') = \mathcal{M}$.

Using the adaptive chain we define the stopping time $\tau = \inf\{n : \mathbb{N}|X_n = X'\}$ which identifies the time of first visit to this state, which also corresponds to the time when the Markov chain kernel will stop changing. $\Gamma_n = \Gamma_\tau$ for all $n \geq \tau$.

Since the chain is irreducible and $\mathcal{S}$ is finite, the states are positive recurrent then $P(\tau < \infty|X_0 = x) = 1$ for any $x \in \mathcal{S}$. So this a finite adaptation algorithm, using Proposition 2 from Roberts & Rosenthal (2007) we know the adaptive algorithm converges to target distribution π. □

Algorithm 2 considers that at every step we update the bounding constant so the proof of convergence to the target distribution relies on the fact that the state space $\mathcal{S}$ is finite. If we want to extend this to an infinite discrete state space $\mathcal{S}^*$ we will need to modify the adaptive step of the algorithm. We can consider a reduction of the bounding constant towards 1, this reduction can be random, defining each step a probability to reduce the constant, or deterministic by reducing the constant after specific number of steps. We can also consider that step 1 is run only for a deterministic number of steps K and after that we use the balancing function h_γ defined with the biggest found bounding constant.

Proposition 3 proves that the Markov chain that follows algorithm 2 is ergodic for a distribution related to the target distribution π. Using theorem 8 from Rosenthal et al. (2020) we can see that if $\{X_k\}$ is the Markov chain constructed using A-IIT and $\{M_k\}$ is the corresponding multiplicity list then, for a suitable function g the estimator:

$$\frac{\sum_{k=1}^{N} M_k g(X_k)}{\sum_{k=1}^{N} M_k} \tag{9}$$

is a consistent estimator of $\mathbb{E}_\pi(g(X))$

Adaptive IIT explores the state space $\mathcal{S}$ in the same way that IIT would do it but it changes the way the weights are estimated. A-IIT can use a multiplicity list because the "escape probability" $Z_h(X)$ is bounded between $(0, 1)$ which was not the case in general for IIT. Using the multiplicity list will be useful when implementing the algorithm with Parallel Tempering as it allows to define a specific number of samples to obtain from the original (non rejection-free) Markov chain.

4 Rejection-free algorithms with Parallel Tempering

When implementing MCMC algorithms with multi modal target distributions it's often useful to use Metropolis-Coupled chains or Parallel Tempering (Geyer, 1991).

To implement Parallel Tempering we consider R independent copies (or replicas) of the chain X, each on the state space $\mathcal{S}$ and a set of inverse temperatures $\beta_1 > \beta_2 > ... > \beta_R \geq 0$. Replica i has $\pi_i \propto \pi^{\beta_i}$ as its stationary distribution.

Parallel tempering consist of two steps: First each replica explores the state space for a fixed number of iterations. The second step is to propose a swap between pairs of adjacent replicas $X^{(\beta_i)} \leftrightarrow X^{(\beta_{i+1})}$. This swap is accepted using the usual Metropolis probability

$$\min\left\{1, \frac{\pi_2(X^{(\beta_1)})\pi_1(X^{(\beta_2)})}{\pi_1(X^{(\beta_1)})\pi_2(X^{(\beta_2)})}\right\} \tag{10}$$

This replica swap probability preserves the product target measure of the original chain $\prod_i \pi(\cdot)^{(\beta_i)}$.

If we would like to implement parallel tempering with each replica following a rejection-free algorithm then the swap probability needs to be adjusted (Rosenthal et al., 2020) considering that the rejection-free chain have a different target distribution. The swap probability needs to be modified as follows:

$$\min\left\{1, \frac{Z_2(X^{(\beta_1)})\pi_2(X^{(\beta_1)})Z_1(X^{(\beta_2)})\pi_1(X^{(\beta_2)})}{Z_1(X^{(\beta_1)})\pi_1(X^{(\beta_1)})Z_2(X^{(\beta_2)})\pi_2(X^{(\beta_2)})}\right\} \tag{11}$$

This preserves the modified product measure $\prod_i Z(\cdot)\pi(\cdot)^{(\beta_i)}$ which corresponds to the target distribution where each replica is a rejection-free chain. Not adjusting the probability will induce bias in the chain that will affect the rate of convergence.

As stated before each of these Z factors requires to evaluate π for every neighbor of the state and apply the corresponding balancing function. For high dimension spaces $\mathcal{N}(X)$ can become very large and although rejection-free chains always move to a different state, the replica swap may be rejected and the additional computations won't translate in new information for the Markov chain.

To avoid these additional computations we use a rejection-free algorithm with multiplicity list so we can alternate between the independent replica updates and the replica swaps without inducing bias. As explained by Rosenthal et al. (2020), we define a number of samples L_0 to obtain from the original chain (with target distribution π). Then we proceed as follows.

1. Set the number of remaining repetitions to $L = L_0$
2. Find the next state for replica i, $X_{n+1}^{(i)}$ and the value of the multiplicity list $M_n^{(i)}$ corresponding to the current state $X_n^{(i)}$
3. If $M_n^{(i)} \geq L$ then replace $M_n^{(i)}$ by L and the chain stays in the current state $X_{n+1}^{(i)} = X_n^{(i)}$. Then return to step 1 for the next replica.
4. If $M_n^{(i)} < L$ then update $L = L - M_n^{(i)}$ and return to step 2 for the same replica

Using a rejection-free chain with a multiplicity list allows us to transform each single rejection-free iteration into M_n samples from the original chain. So we can use the steps described above to obtain L_0 samples from the original chain using a multiplicity list in all the replicas. This then allows us to use parallel tempering without creating bias and without additional computations when trying a replica swap. We can see it's straightforward to implement Parallel Tempering where each replica uses A-IIT following the four steps above. This is more efficient than using IIT in the replicas as we avoid the computation of the factors $Z_i(X^{(\beta_j)})$ every time we try a replica swap.

5 Implementation of Adaptive IIT in Parallel Tempering

5.1 Single step IIT

Algorithm 2 requires a lot of computations for each iteration. First the update of the bounding constant requires the computation of $\pi(\cdot)$ for each of the $\mathcal{N}(X)$ neighbors of the current state X. Then we need to apply the bounded balancing function to each of the probability ratios to compute the weight $Z_h(X_k)$ and then choose one of the proposed states.

When implementing A-IIT with parallel tempering it's possible to use a non-rejection free algorithm for some of the replicas as long as the target distribution is not modified. To achieve this, we can consider an algorithm that uses a bounded balancing function h and is equivalent to Adaptive IIT but creates just one sample of the original chain every iteration.

Algorithm 3 Single-step IIT (SS-IIT)

```
Initialize $X_0$ and $\gamma_0 = 1$
for $k$ in 0 to $K$ do
                                                  ▷ Step 1: Update bounding constant
    Propose a new state $Y \in \mathcal{N}(X_k)$ using the proposal distribution $\mathcal{Q}(\cdot|X_k)$
    Calculate
        $$B_k = \max\left\{ f\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right), f\left(\frac{\pi(X_k)\mathcal{Q}(Y|X_k)}{\pi(Y)\mathcal{Q}(X_k|Y)}\right)\right\}$$
    Update $\gamma_{k+1} = \max\{\gamma_k, B_k\}$
    Define $h_\gamma$ using the updated constant $\gamma_{k+1}$ following equation 3
                                  ▷ Step 2: Use the adapted Markov kernel to update the chain
    Set $w_k(X_k) \leftarrow 1$.
    let $U \sim Unif(0,1)$
    if $U < h_\gamma\left(\frac{\pi(Y)\mathcal{Q}(X_k|Y)}{\pi(X_k)\mathcal{Q}(Y|X_k)}\right)$ then
        $X_{k+1} \leftarrow Y$
    else
        $X_{k+1} \leftarrow X_k$
    end if
end for
```

As discussed in the previous section, when implementing parallel tempering with Adaptive IIT we define a number of samples of the original chain L_0 to obtain before trying a replica swap. We can also use Single-step IIT to obtain the desired number of samples.

The advantage of A-IIT is that, since it's rejection free, after every iteration the chain moves to a different state however this requires additional computations. The advantage of SS-IIT is that it doesn't require as many computations to propose a jump however we may reject the proposed state and the chain may not explore the state space as fast as A-IIT.

When implementing Parallel Tempering we can choose some replicas to use A-IIT and others to use SS-IIT. For some replicas with high values of β we expect more rejections whenever the chain moves to a state of high probability so we would prefer A-IIT. For a value of β close to zero we don't expect as many rejections so SS-IIT might be a better choice.

5.2 Choosing between A-IIT and SS-IIT

Now we have two adaptive algorithms that can generate samples from the target distribution π while using any balancing function h. To choose which of them to use for the different replicas in parallel tempering we will compare the number of computations needed for the chain to generate L_0 samples from the original chain. In this situation we consider that each evaluation of π in a neighbor state counts as one computation. SS-IIT always generates 1 sample of the original chain for each iteration of the algorithm which requires to evaluate π in the proposed neighbor state. A-IIT generates one sample from the rejection-free chain which is equivalent to M_n (a random number) of samples from the original chain but it requires to evaluate π for each state in the set

$\mathcal{N}_X$.

Consider the state space $\mathcal{S} = \{0,1\}^p$ with $p = 5000$ and a target distribution with 6 modes defined in a similar way than the target distributions presented in the next section i.e., $\pi(X) \propto \sum_{i=1}^{6} \exp(-\theta \, ||X - X_{(i)}||_1)$.

We run parallel tempering with 13 replicas $(X^{(\beta_1)}, \ldots, X^{(\beta_{13})})$. Each replica uses Adaptive IIT until the multiplicity list reaches a value of $L_0 = 800$ then tries a replica swap. We run 25 simulations and in each simulation each replica performs 100 swaps before stopping. We then compute the average number of rejection free steps each replica performed in-between replica swaps.

Table 1 contains the values of temperatures used and the average swap rate between β_i and β_{i+1}.

Table 1: Inverse temperature used

id	inverse temperature	swap rate
β_1	20000	0.412
β_2	19820	0.382
β_3	19539	0.426
β_4	19409	0.401
β_5	19282	0.384
β_6	19169	0.414
β_7	19112	0.333
β_8	18935	0.272
β_9	18724	0.341
β_{10}	18663	0.318
β_{11}	18586	0.256
β_{12}	18507	0.297
β_{13}	18485	-

In figure 2 we plot the number of rejection free iterations each replica with inverse temperature β_i needs so that their multiplicity list reaches the value $L_0 = 800$, i.e., before we try a replica swap.

We can observe that as the value of β decreases, the number of iterations increases as well.

For β_1 on average the chain performs 3 rejection-free steps before the replica swap. This means that single-step IIT algorithm would need, on average, 267 rejections before accepting a move. On the other hand for β_{13} the chain performs, on average, 26 rejection-free steps before the replica swap, single-step IIT algorithm would need, on average, 30 rejections before accepting a move.

The size of the neighbor set is the dimension of the state space $|\mathcal{N}(X)| = 5000$, this means that each rejection-free iteration needs to evaluate π 5000 times before moving to a new state no matter the value of β, meanwhile the number of computations needed for SS-IIT to move to a new state depends on the value of β, table 2 shows a summary of the % of computations needed for the chain to move to a different state for SS-IIT in comparison with A-IIT.

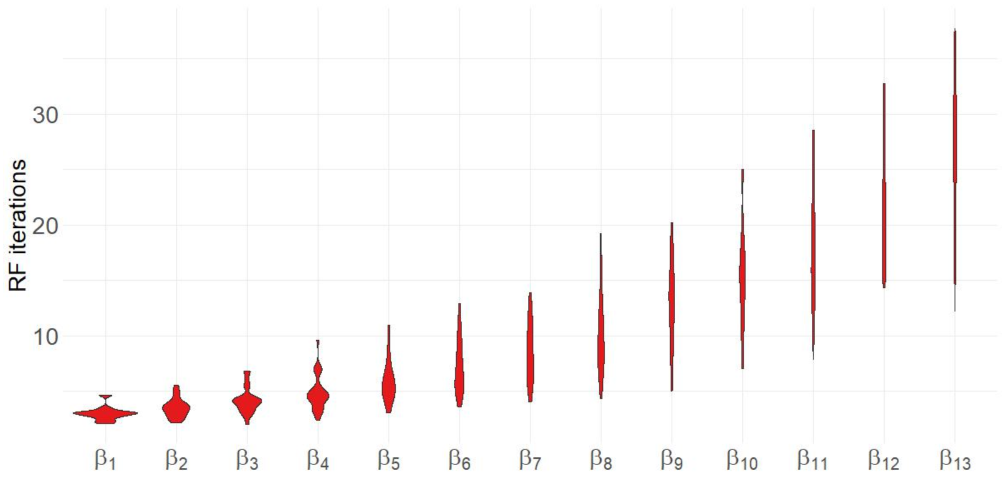

Figure 2: Number of Rejection Free iterations for replicas tempered at different temperatures

Algorithm	β_1	β_{13}
A-IIT	5000	5000
SS-IIT	267	30
%	5.3%	0.6%

Table 2: Comparison of number of computations needed for the Markov chain to move to a different state

In this example the the number of computations needed for the smallest β is 10 times bigger than the computations for the largest β. When implementing Adaptive IIT with parallel tempering we can consider running a first simulation to measure the average number of rejection-free steps that algorithm 2 would need in-between replica swaps and compare with the number of computations needed. With this information we can define which algorithm to use for each replica in such a way that we reduce the number of computations needed for a replica to visit a new state and, in this way, implement an algorithm that efficiently explores the state space.

5.3 General discussion of the balancing function

The proposed adaptive algorithms are defined in general and can be applied for any increasing function f. In their work Zhou & Smith (2022) gives particular emphasis to the balancing function $h(r) = \sqrt{r}$ so in this work we implement a bounded version of this function to use in the adaptive algorithm. In this section we present the process to create a bounded version of this balancing function.

For a fixed $\gamma > 0$ taking the function $f(r) = \sqrt{r}$ as a starting point, the bounded balancing function version of f takes the following form:

$$h_\gamma(r) = \min\left\{1, r, \frac{\sqrt{r}}{\gamma}\right\} = \begin{cases} r \text{ for } r \in \left[0, \frac{1}{\gamma^2}\right) \\ \frac{\sqrt{r}}{\gamma} \text{ for } r \in \left[\frac{1}{\gamma^2}, \gamma^2\right) \\ 1 \text{ for } r \in \left[\gamma^2, \infty\right) \end{cases} \tag{12}$$

Figure 3 shows the behaviour of this bounded balancing function for some values of γ.

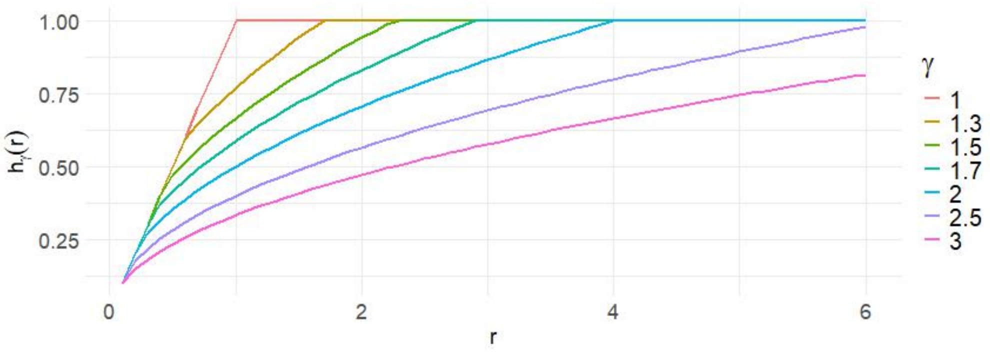

Figure 3: Bounded balancing function based on $h(r) = \sqrt{r}$ considering different values for the bounding constant γ.

Since we want to use h_γ to compute probabilities there's no usage of using a bound smaller than 1. We can observe that for $\gamma = 1$ $h_1(r) = \min\{1, r\}$ the balancing function is the same as the function used in Metropolis-Hastings algorithm.

The definition of algorithm 2 requires a double visit to the neighbor states. First we compute the ratio of probabilities of the neighbor states to update the bounding constant. Then, we apply the updated balancing balancing function to the target distribution evaluated at each neighbor state. This ensures that we're always using the latest information available (the latest bounding constant found).

To avoid this double visit to the neighbor states, we could swap steps 1 and 2 in the algorithm. So first we compute the ratio of probabilities $\pi(y)/\pi(x)$ for all the neighbor states, store these values, then apply the bounded balancing function (using a previously found bound) to compute the factor $Z_h(x)$. After this computation we can use the stored values of the ratio of probabilities, apply the function f and compute the new balancing function.

Changing the order of steps 1 and 2 in the algorithm helps to avoid the double visit to the neighbor states but we need extra storage capacity. If we use the squared root balancing function we can avoid this double visit in a more straightforward way.

Note that the definition of the bounded squared root balancing function (3), for a fixed $\gamma > 0$ $h_\gamma(r) = \sqrt{r}/\gamma$ as long as $r \in [1/\gamma^2, \gamma^2]$. So we can simplify the definition of the function as long as the input is between these two values.

For simplicity consider that we're using a uniform proposal distribution $\mathcal{Q}$ and consider $R_i = \pi(Y_i)/\pi(X)$. Note that the way that we update the bounding constant guarantees that $\gamma \geq R_i^{1/2}$

and $\gamma \geq R_i^{-1/2}$, this implies that $R_i \in [1/\gamma^2, \gamma^2]$ for all the neighbors of X, and this is ensured in every iteration of the algorithm.

Then we don't need to define a bounded balancing function to apply to each $\sqrt{R_i}$, to apply the bounded balancing function it suffices to divide all the values by γ i.e., $h_\gamma(R) = \sqrt{R}/\gamma$. Algorithm 4 shows how we can implement this more efficient version of Adaptive IIT when we use the balancing function $h(r) = \sqrt{(}r)$. This doesn't change the way the chain behaves but reduces the number of computations in each step of the algorithm.

Algorithm 4 Adaptive IIT (computationally efficient for $f(r) = \sqrt{r}$)

```
Initialize $X_0$ and $\gamma_0 = 1$.
Consider a uniform proposal distribution $\mathcal{Q}(Y|X) \propto 1$
for $k$ in 1 to $K$ do
    For each $Y_i \in \mathcal{N}_{X_{k-1}}$, calculate
        $$R_i(X_{k-1})^{1/2} = \left(\frac{\pi(Y_i)}{\pi(X_{k-1})}\right)^{1/2}$$
    $\gamma_c = \max_i\{R_i^{1/2}, R_i^{-1/2}\}$
    $\gamma_k = \max\{\gamma_{k-1}, \gamma_c\}$
    Calculate
        $$Z_h(X_{k-1}) = \frac{1}{|\mathcal{N}(X_{k-1})|} \sum_{y \in \mathcal{N}_{X_{k-1}}} \frac{1}{\gamma_k} \left(\frac{\pi(Y_i)}{\pi(X_{k-1})}\right)^{1/2}$$
    Set $w_k(X_{k-1}) \leftarrow 1 + \mathit{Geometric}(Z_h(X_{k-1}))$.
    Choose the next state $Y \in \mathcal{N}_{X_{k-1}}$ such that
        $$P(X_k = Y|X_{k-1}) \propto \left(\frac{\pi(Y)}{\pi(X_{k-1})}\right)^{1/2}$$
end for
```

6 Simulations

6.1 General framework

For the simulations we consider a state space $\mathcal{S} = \{0,1\}^p$ and a multimodal target distribution π defined as

$$\pi(X) = \frac{1}{\mathfrak{C}_\theta} \sum_{i=1}^{m} \exp(-\theta \, ||X - X_{(i)}||_1) \tag{13}$$

Where m is the number of modes, $X_{(i)}$ represents mode i, $\theta > 0$ and $||\cdot||_i$ is the L_1 distance, $\mathfrak{C}_\theta$ is the normalizing constant.

We implement non-reversible Parallel Tempering with R replicas (Syed et al., 2021) alternatinv between swapping even then odd indexed replicas. We compare the performance of Adaptive IIT with other 3 algorithms. All of them use non-reversible parallel tempering but the replicas in each of the implementation will use one of 4 different algorithms. For all of the algorithms we use a uniform proposal distribution ($\mathcal{Q}(y|x) \propto 1$ for all $y \in \mathcal{N}_X$) and the neighbor set is defined by all the states with L_1 distance 1: $\mathcal{N}_x = \{y \in \mathcal{S} : ||x - y||_1 = 1\}$.

#	ID	Name	$h(r)$	$w(x)$	Algorithm
1	A-IIT	Adaptive IIT	$\sqrt{r}$	$1 + G(Z(x))$	2
				1	3
2	IIT	Naive IIT	$\sqrt{r}$	$1/Z(x)$	1
3	MH-mult	Metropolis Hastings with multiplicity list	$\min\{1, r\}$	$1 + G(Z(x))$	5
				1	7
4	RF-MH	Rejection Free Metropolis Hastings	$\min\{1, r\}$	$1/Z(x)$	6

Table 3: Summary of compared algorithms.

In table 3 is the summary of the algorithms compared. IIT and RF-MH use direct weight estimation so they require additional computations for the replica swap probability. The algorithms that we label as A-IIT and MH-mult are implementation of 2 different algorithms. For some of the replicas with high values of β we use a rejection free algorithm with a multiplicity list (algorithms 2 and 5) while for replicas with low values of β we use algorithms that allow rejections (algorithms 3 and 7), the number of replicas using a rejection-free algorithm is the same for both algorithms in any example.

Traditionally one should run the algorithms for the same amount of iterations to make a fair comparison of the performance of each of them. For this work is not straightforward to define the number of iterations to run each algorithm since algorithms (2) and (4) perform a fixed number of rejection free iterations while algorithms (1) and (3) perform a random number of rejection free iterations since they use a multiplicity list.

To define equivalent number of iterations in between replica swaps for all algorithms, we first define a number of rejection free iterations for algorithms (2) and (4) and then, for algorithms (1) and (3), we define a value of L_0 (the number of samples from the original chain to obtain before a replica swap) in such a way that, for the replica with the highest β, the average number of Rejection-Free steps in between replica swaps is as close as possible for all the algorithms. In this way the replicas get the same number of steps to explore the state space and the comparison is more fair.

For example for algorithms (2) and (4) we define 2 rejection-free iteration to try between replica swaps and to run the algorithms for a total of 2 million iterations, this translates in 1 million of swaps tried. For algorithms (1) and (3) defining $L_0 = 800$ equals to 1.64 and 1.2 average rejection free steps. So we use that L_0 and the algorithms tries a total of 1 million replica swaps. In this way we try to make the comparison as fair as possible.

The number of replicas and the distance between the β's is defined such that the proportion of accepted swap rates between adjacent replicas is close to 0.234 (Atchadé et al., 2011).

We measure the time it takes the replica with the highest β to reach all of the modes as we consider this is a measurement of how good is the chain's exploration capabilities to find states with high probability. This is presented with graphs simulating a ladder where in the x-axis we plot the time and the y-axis we plot the percentage of simulations that found all the modes of the target distribution at the defined time. Each jump in the plot represents one simulation finding all the modes of the target distribution, the faster the function reaches 100% the faster the algorithm is in finding all the modes.

For the low dimensional examples we also measure the evolution of Total Variation Distance (TVD). We present this in a chart where the x-axis is the time and the y-axis is the average TVD calculated of all the simulations ran. The faster the plot approaches 0 the better the algorithm is at converging to the desired target distribution.

The simulation results for the **low dimensional** examples show that the rejection free algorithms perform better both in terms of TVD and in speed to find the modes. This is expected as the rejection free algorithms efficiently use the information of neighbor states to identify direct paths to the modes. In this low dimensional example the additional computations needed by the algorithms using a multiplicity list (A-IIT and MH-mult) is a lot more than the computations needed by algorithms using direct weight estimation (IIT and RF-MH). This is due to the dimension of the state space being small and L_0 being large in comparison. So the number of computations in-between replica swaps for algorithms with multiplicity list is always larger than the computations for the other two algorithms and the additional computations needed during the replica swap process is not significant.

It is in the simulation results for the **high dimensional** example that we observe A-IIT performing better than the other algorithms. Since the dimension of the space is $p = 3000$ the computation of the probability of swapping adjacent replicas requires an additional $4 \times 3000 = 12000$ evaluation of the function π, and since we are performing non-reversible replica swaps each step of replica swaps requires $12000 \times 6 = 72000$ additional computations, this impacts the speed in which the parallel tempering algorithm explores the state space.

Additional to that we observe that the value of L_0 does not scale at the same rate as the dimension of the state space so, for the algorithms using multiplicity list, we are able to find an adequate choice of replicas that use the equivalent algorithm that permits rejections in such a way that we minimize the number of computations needed per replica swap.

In the simulation results, we can also see the difference in speed between A-IIT and MH-mult which can be attributed to using a different balancing function. Hence we observe the advantage of using A-IIT over Metropolis Hastings or IIT when using parallel tempering.

6.2 Low dimensional bi-modal example

For this example we consider a space of dimension $p = 16$ with two modes:

- $x_{(1)} = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)$
- $x_{(2)} = (0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1)$

The state space is $\mathcal{S} = \{0, 1\}^{16}$, we use $\theta = 6$ to define the target distribution, with these parameters, the normalizing constant is $\mathfrak{C}_\theta = 2(1+e^{-\theta})^{16}$, each mode accumulates 48% of the total probability mass. Since this space is small enough we can compute the true distribution π and use it to compute the Total Variation Distance.

We run all algorithms for a total of 1 million replica swaps, the algorithms using rejection free iterations try a replica swap after every iteration. Algorithm A-IIT uses $L_0 = 1000$ and algorithm MH-mult uses $L_0 = 100$. We run 100 simulations of each algorithm to obtain the results presented below.

For this simulations, all the replicas of A-IIT and MH-mult use the rejection free algorithm. In table 8 we see the average of rejection free swaps that each replica performs in-between replica swaps with the defined L_0.

Table 4: Average number of iterations between replica swaps

replica	A-IIT	MH-mult
β_1	1.246	1.489
β_2	24.067	10.942
β_3	91.075	25.027
β_4	218.88	42.84

In this particular problem we don't have benefits of using A-IIT as using IIT is faster both in terms of finding the modes and convergence to the target distribution. We attribute this to the small size of the state space, the number of additional computations needed by IIT both for each iteration and trying replica swaps does not hinder the performance.

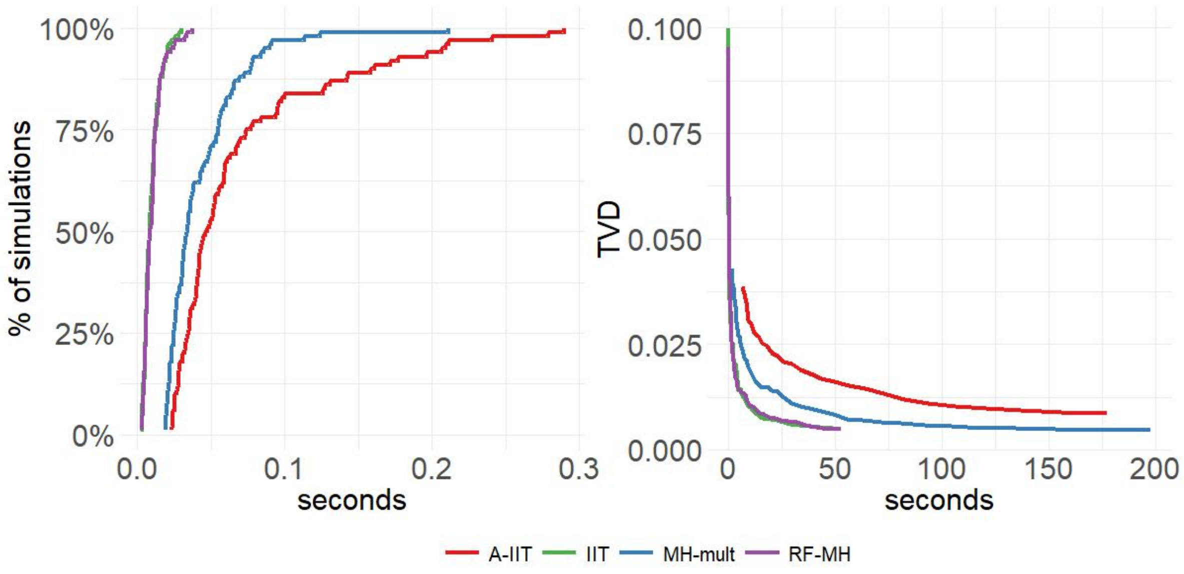

Figure 4: Comparison of performance of the four algorithms in the low dimensional bimodal problems. On the left, time to visit all the modes of the target distribution. On the right, evolution of Total Variation Distance

6.3 Low dimensional multi modal example

For this example we consider a space of dimension $p = 16$ with seven modes:

- $x_{(1)} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$
- $x_{(2)} = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)$
- $x_{(3)} = (0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1)$
- $x_{(4)} = (1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)$
- $x_{(5)} = (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1)$
- $x_{(6)} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$
- $x_{(7)} = (0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0)$

The state space is $\mathcal{S} = \{0, 1\}^{16}$, we use $\theta = 10$ to define the target distribution, with these parameters, the normalizing constant is $\mathfrak{C}_\theta = 7(1 + e^{-\theta})^{16}$, all modes accumulate 99.9% of the total probability mass. Since this space is small enough we can compute the true distribution π and use it to compute the Total Variation Distance.

We run all algorithms for a total of 1 million replica swaps, the algorithms using rejection free iterations try a replica swap after every 2 iterations while the algorithms using a multiplicity list uses an $L_0 = 1000$. We run 100 simulations of each algorithm to obtain the results presented below.

For this simulations, all the replicas of A-IIT and MH-mult use the rejection free algorithm. In table 10 we see the average of rejection free swaps that each replica performs in-between replica swaps with the defined L_0. Here we observe the effect of the adaptive constant. A bounding constant bigger than 1 means that the probability of escape $Z_h(X)$ is smaller which then translates in bigger values for the multiplicity list, reducing the number of rejection-free iterations needed to reach the defined L_0.

Table 5: Average number of iterations between replica swaps

replica	A-IIT	MH-mult
β_1	1.001	1.091
β_2	19.306	87.393
β_3	77.602	221.185

We observe a similar behavior as in the bi-modal problem. Using parallel tempering with IIT gives a faster algorithm to both find all the modes and estimating the the target distribution. We consider that the benefits of using A-IIT depend on the dimension of the state space and not on the number of modes the state space has.

We don't observe any significant differences in the performance of using IIT or Rejection Free Metropolis Hastings. As the dimension of the state space is small there's no significant advantage to using a different balancing function when choosing a state to move to.

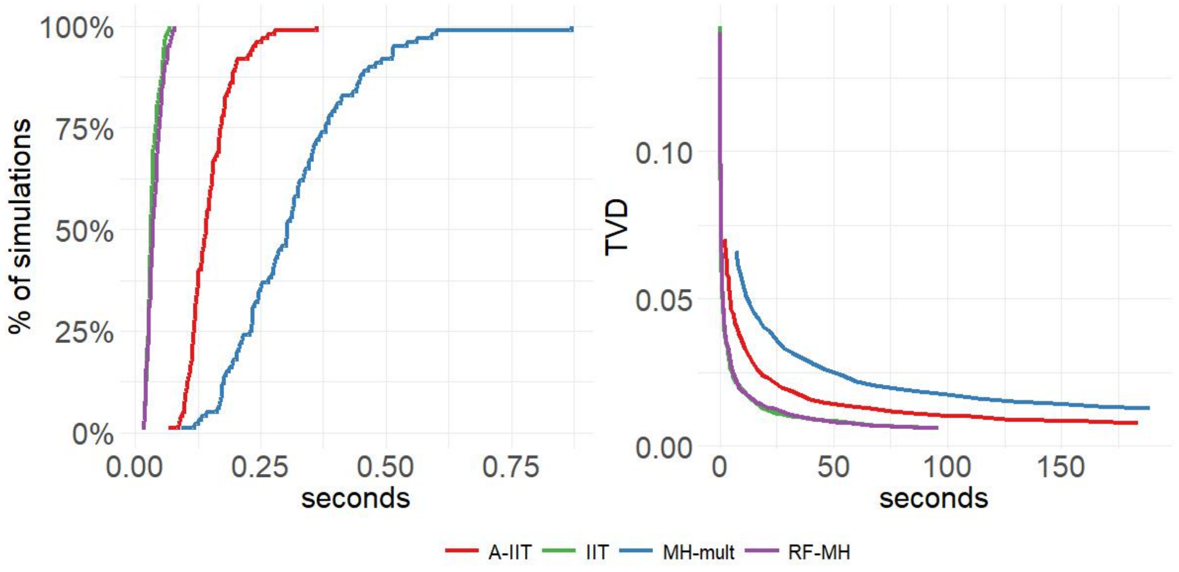

Figure 5: Comparison of performance of the four algorithms in the low dimensional multimodal problem. On the left, time to visit all the modes of the target distribution. On the right, evolution of Total Variation Distance

6.4 High dimensional example

For this example we consider a space of dimension $p = 3000$ and we consider 6 modes:

- $x_{(1)} = (1, 0, 1, 0, \ldots, 1, 0, 1, 0)$
- $x_{(2)} = (0, 1, 0, 1, \ldots, 0, 1, 0, 1)$
- $x_{(3)} = (1, 1, 1, 1, \ldots, 0, 0, 0, 0)$
- $x_{(4)} = (0, 0, 0, 0, \ldots, 1, 1, 1, 1)$
- $x_{(5)} = (0, 0, 0, 0, \ldots, 0, 0, 1, 1, \ldots, 1, 1, 0, 0, \ldots, 0, 0, 0, 0)$
- $x_{(6)} = (1, 1, 1, 1, \ldots, 1, 1, 0, 0, \ldots, 0, 0, 1, 1, \ldots, 1, 1, 1, 1)$

All modes satisfy $||x_{(i)}||_1 = \frac{p}{2}$ and are defined in such a way that $||x_{(i)} - x_{(j)}|| \geq \frac{p}{2}$

The state space is $\mathcal{S} = \{0, 1\}^{3000}$, we use $\theta = 0.001$ to define the target distribution.

We run all algorithms for a total of 1 million replica swaps, algorithms (2) and (4) try a replica swap after 2 iterations while the algorithms using a multiplicity list uses an $L_0 = 800$. For algorithms (1) and (3) 8 out of the 13 chains use the rejection free algorithm and the remaining 5 use the algorithm that allow rejections. We run 100 simulations of each algorithm to obtain the results presented below.

In table 12 we see the average number of iterations each replica performs in-between replica swaps with the defined L_0. For the 8 replicas using the rejection free algorithm this number corresponds to the average number of rejection-free steps performed before the multiplicity list reaches the value of L_0, for the other 5 replicas the number is fixed as the chain always performs L_0 iterations of the algorithm before trying a replica swap.

Table 6: Average number of iterations between replica swaps

replica	A-IIT	MH-mult
β_1	1.614	1.162
β_2	1.809	1.352
β_3	2.001	1.788
β_4	2.262	2.587
β_5	2.678	3.916
β_6	3.235	5.346
β_7	4.052	6.816
β_8	5.209	8.588
β_9	800	800
β_{10}	800	800
β_{11}	800	800
β_{12}	800	800
β_{13}	800	800

For algorithms (1) and (3) the burn-in period lasts until the multiplicity list reaches the value 8000. For algorithms (2) and (4) the burn-in period lasts 50 iterations. For algorithm (1) the bounding constant is updated only during the burn-in period. After the burn-in period we stop the adaptation and use the balancing function considering the latest bounding constant found.

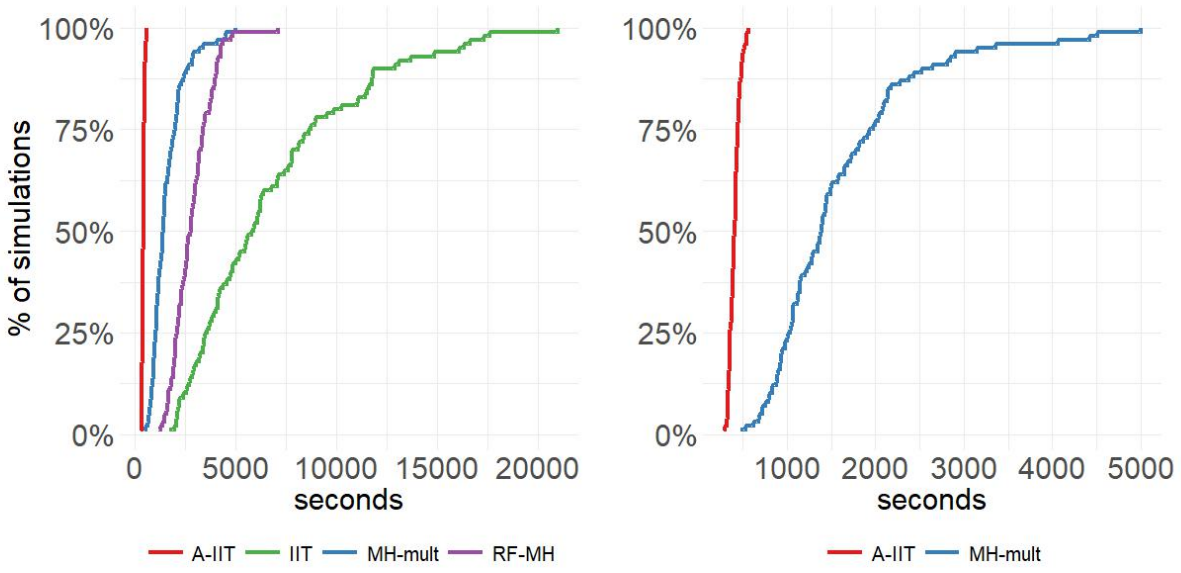

Figure 6: Comparison of time to visit all the modes of the target distribution in a space of dimension $p = 3000$. On the right is the same plot only focusin on algorithms A-IIT and MH-mult

The left plot of figure 6 presents the time each algorithm took to find all the modes. We can observe that A-IIT and MH-mult, the algorithms that use a multiplicity list, are faster than IIT and RF-MH, the rejection-free algorithms. The time for the algorithms using a multiplicity list could be improved in this case by using SS-IIT in more chains..

The right plot of figure 6 we observe the plot only for algorithms (1) and (3). Here we can better appreciate the difference in time. Adaptive IIT is 3 times faster than Rejection Free Metropolis-Hastings. We can attribute the difference in performance to the different balancing function used since that's the main difference between the algorithms.

In figure 7 we can see the plot of the different bounds found for each replica in the simulations. As we can see with the short burn-in period the algorithm was able to find different bounding constants for each replica. When choosing a state for the chain to move algorithm (3) assigns the same probability to all the states with higher probability than the current state, while algorithm (1) may assign different probabilities. This means that algorithm (1) will move quicker to states with high probability and in this setting where the target distribution is multimodal and each mode has a path of high probability leading to it, makes algorithm (1) reach the mode faster.

In figure 8 we ran similar simulations for algorithms (1) and (3) keeping track of the time it takes for the algorithms to find all the modes but with different number of replicas using the rejection free algorithm. In the x-axis is the number of replicas that used the rejection free algorithm, in the y-axis is the time in seconds it took for the simulations to find the modes.

If we increase the number of replicas using a rejection free algorithm the time to find the modes

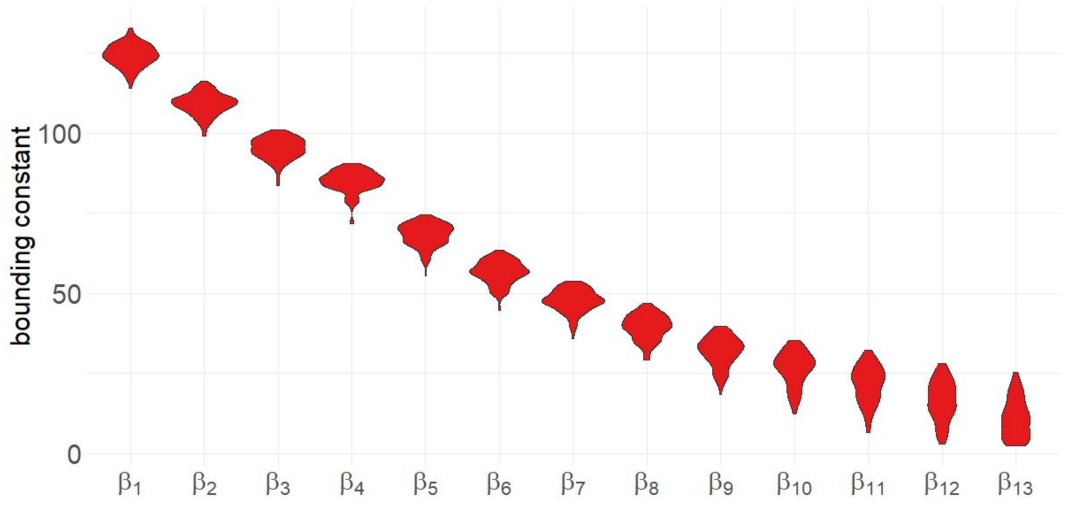

Figure 7: Summary of bounds used for each chain after the burn-in period for the 100 simulations.

increases, this is because each iteration requires more computations. However we can see that in all cases A-IIT is faster than MH-mult. This shows the advantage of using $h(r) = \sqrt{r}$ as the balancing function.

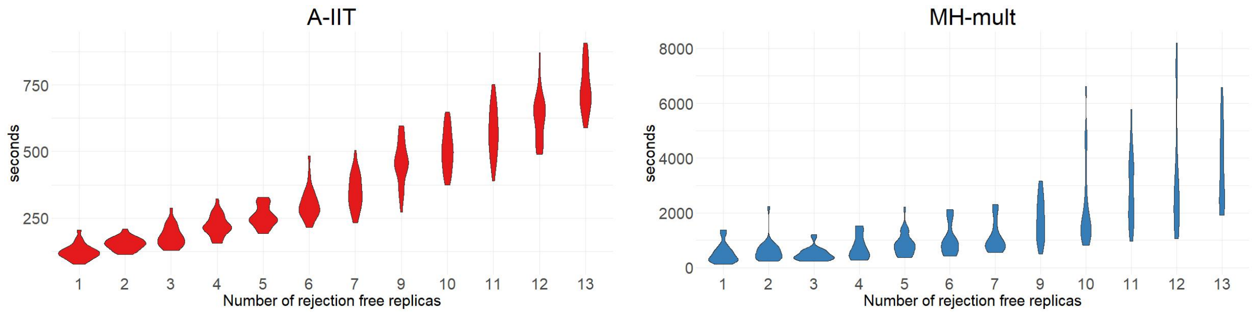

Figure 8: Comparison of time to find all modes for different number of rejection free replicas in 100 simulations.

7 Summary and future work

This work presented an adaptive MCMC algorithm for discrete state spaces that uses information of the neighbors of the current state to "temper" the choice of the next state to visit. We present both a rejection free version and what we call a single-step version of the algorithm.

We prove the convergence of these algorithms to the desired target distribution, and we show that they are equivalent in the sense that both converge to the same target distribution. This equivalence gives flexibility to the implementation with parallel tempering as we can choose the algorithm that more efficiently uses the available computational resources although the choice will depend on the specific problem to solve.

There are aspects of the algorithm that we don't explore in this work and can be the topic of future research.

- The choice of importance function f to define the bounded balancing function: As was mentioned in the original work of Li et al. (2023) the choice of balancing functions is problem specific.
- The schedule of adaptations: We mentioned the possibility of reducing the bounding constant towards 1 so we can consider a first stage of constantly updating the constant to improve the exploration of the state space and later reduce it to reduce the variance of the Monte Carlo estimator $\widehat{\mathbb{E}_\pi(X)}$
- The optimal choice of rejection free or single step replicas: In this work we made a choice on the number of rejection free replicas to use for the simulations trying to use half and half but there's an opportunity to define the optimal number of replicas for each algorithm.
- Extensions to other algorithms: In the work of Li et al. (2023) it's mentioned that IIT allows for other modifications for example to consider a smaller neighbor set at each iteration. We consider that A-IIT can also be adapted in a similar way.

Acknowledgments Thanks to Norma Garcia-Salazar and Marco Gallegos-Herrada for helpful discussions and remarks. This research was enabled in part by support provided by Compute Ontario (computeontario.ca) and the Digital Research Alliance of Canada (alliancecan.ca).
Funding The authors thank Fujitsu Limited and Fujitsu Consulting (Canada) Inc. for providing financial support.
Conflict of Interest The authors declare no competing interests.

References

Atchadé, Y. F., Roberts, G. O., & Rosenthal, J. S. (2011, October). Towards optimal scaling of metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, *21*(4), 555–568. Retrieved 2025-05-21, from http://link.springer.com/10.1007/s11222-010-9192-1 doi: 10.1007/s11222-010-9192-1

Gelman, A., Gilks, W. R., & Roberts, G. O. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, *7*(1), 110 – 120. Retrieved from https://doi.org/10.1214/aoap/1034625254 doi: 10.1214/aoap/1034625254

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. In *Computing science and statistics: Proceedings of the 23rd symposium on the interface* (pp. 156–163).

Hastings, W. K. (1970, 04). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97-109. Retrieved from https://doi.org/10.1093/biomet/57.1.97 doi: 10.1093/biomet/57.1.97

Li, G., Smith, A., & Zhou, Q. (2023). *Importance is important: A guide to informed importance tempering methods.* Retrieved from https://arxiv.org/abs/2304.06251v2

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953, 06). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087-1092. Retrieved from https://doi.org/10.1063/1.1699114 doi: 10.1063/1.1699114

Neal, P., Roberts, G., & Yuen, W. K. (2012). Optimal scaling of random walk Metropolis algorithms with discontinuous target densities. *The Annals of Applied Probability*, *22*(5), 1880 – 1927. Retrieved from https://doi.org/10.1214/11-AAP817 doi: 10.1214/11-AAP817

Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, *16*(4), 351 – 367. Retrieved from https://doi.org/10.1214/ss/1015346320 doi: 10.1214/ss/1015346320

Roberts, G. O., & Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of Applied Probability*, *44*(2), 458–475. Retrieved 2024-11-26, from http://www.jstor.org/stable/27595854

Rosenthal, J. S., Dote, A., Dabiri, K., Tamura, H., Chen, S., & Sheikholeslami, A. (2020). *Jump markov chains and rejection-free metropolis algorithms.*

Syed, S., Bouchard-Côté, A., Deligiannidis, G., & Doucet, A. (2021). Non-reversible parallel tempering: a scalable highly parallel mcmc scheme. *Journal of Royal Statistical Society, Series B*, *(Accepted)*.

Zanella, G. (2020). Informed proposals for local mcmc in discrete spaces. *Journal of the American Statistical Association*, *115*(530), 852–865. Retrieved from https://doi.org/10.1080/01621459.2019.1585255 doi: 10.1080/01621459.2019.1585255

Zhou, Q., & Smith, A. (2022, 28–30 Mar). Rapid convergence of informed importance tempering. In G. Camps-Valls, F. J. R. Ruiz, & I. Valera (Eds.), *Proceedings of the 25th international conference on artificial intelligence and statistics* (Vol. 151, pp. 10939–10965). PMLR. Retrieved from https://proceedings.mlr.press/v151/zhou22e.html

Appendix

A Details on the simulations

Here we present more details on the simulations. Specifically the values of β used, the average swap rate and we present other results obtained in spaces of dimension 1000, 5000 and 7000.

A.1 Low dimensional examples

Table 7: Inverse temperature used in the bimodal model

temp_id	A-IIT	MH-mult	IIT	RF-MH
β_1	1	1	1	1
β_2	0.49	0.49	0.26	0.26
β_3	0.33	0.33	0.06	0.06
β_4	0.22	0.22	NA	NA

Table 8: Average swap rate between replica β_i and β_{i+1} for the bimodal model

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.2352	0.2382	0.2352	0.2382
β_2	0.2423	0.2327	0.2423	0.2326
β_3	0.2521	NA	0.2521	NA

Table 9: Inverse temperature used in the 7 modes model

temp_id	A-IIT	MH-mult	IIT	RF-MH
β_1	1	1	1	1
β_2	0.31	0.31	0.15	0.155
β_3	0.21	0.21	0.002	0.002

Table 10: Average swap rate between replica β_i and β_{i+1} for the 7 modes model

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.2478	0.2327	0.2477	0.2506
β_2	0.2521	0.2539	0.2521	0.2391

A.2 High dimensional example

Table 11: Inverse temperature used in the problem of dimension 3k

temp_id	A-IIT	IIT	MH-mult	RF-MH
β_1	20000	20000	20000	20000
β_2	19517	15046	17899	15005
β_3	19029	13509	15895	13611
β_4	18535	12162	14353	12684
β_5	17744	11215	13057	12182
β_6	17125	10570	12234	11600
β_7	16568	10087	11631	11377
β_8	16037	9716	11093	11185
β_9	15571	9396	10578	11090
β_{10}	15273	9166	10109	10986
β_{11}	15075	9001	9409	10892
β_{12}	14786	8827	8951	10802
β_{13}	14595	8691	8417	10735

Table 12: Average swap rate between replica β_i and β_{i+1} in space of dimension 3000

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.4489	0.2509	0.3759	0.2161
β_2	0.4582	0.342	0.2893	0.3295
β_3	0.4333	0.2239	0.2381	0.3212
β_4	0.4009	0.222	0.1943	0.4041
β_5	0.3953	0.2743	0.2374	0.2964
β_6	0.3859	0.3078	0.2604	0.4368
β_7	0.3664	0.3336	0.2586	0.4125
β_8	0.3505	0.3144	0.2383	0.4716
β_9	0.3575	0.3507	0.2287	0.386
β_{10}	0.3942	0.4006	0.0984	0.3795
β_{11}	0.3503	0.3268	0.1639	0.3212
β_{12}	0.3703	0.4098	0.0946	0.4413

B Other results in high dimensions

We chose to present the results of the simulations of a problem in a state space of dimension 3000 as this showed clearly the advantages of A-IIT over other algorithms. Here we share some other results obtained from simulations run with similar target distributions π with the same number of modes but in other dimensions.

Although the results presented for the state space of dimension 3000 are not replicated in dimension 5000, we observe similar results for dimension 7000. We consider there is an opportunity to optimize some parameters in dimension 5000 so that we can show that A-IIT performs better than Metropolis Hastings

B.1 Dimension 1000

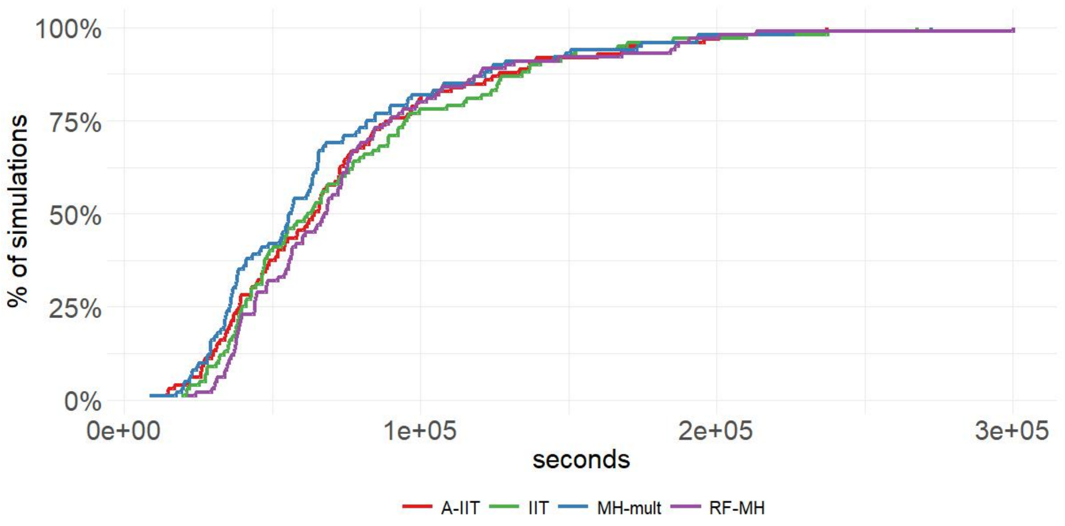

Figure 9: Comparison of time to visit all the modes of the target distribution in a space of dimension $p = 1000$.

Table 13: Inverse temperature used in the problem of dimension 1k

temp_id	A-IIT	IIT	MH-mult	RF-MH
β_1	20000	20000	20000	20000
β_2	19209	18092	19175	18202
β_3	18544	16632	18496	16829
β_4	18030	15661	17962	15769
β_5	17441	14723	17333	14902
β_6	16904	13981	16728	14075
β_7	16480	13544	16322	13582
β_8	15969	12819	15876	12999
β_9	15483	12279	15397	12419
β_{10}	15115	11806	15052	11994
β_{11}	14768	11372	14716	11472
β_{12}	14376	10970	14393	11043
β_{13}	13910	10484	13968	10691

Table 14: Average swap rate between replica β_i and β_{i+1} in space of dimension 1000

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.333	0.2655	0.3258	0.2971
β_2	0.345	0.2824	0.3408	0.3201
β_3	0.369	0.3951	0.3631	0.3636
β_4	0.3397	0.3454	0.3275	0.3922
β_5	0.3417	0.3968	0.3199	0.3537
β_6	0.3651	0.5849	0.3678	0.5423
β_7	0.3287	0.3205	0.3468	0.4315
β_8	0.3254	0.4131	0.3255	0.3909
β_9	0.3576	0.4338	0.3648	0.4932
β_{10}	0.3583	0.4378	0.3617	0.3612
β_{11}	0.3324	0.4397	0.3603	0.4164
β_{12}	0.2914	0.3085	0.3092	0.4715

B.2 Dimension 5000

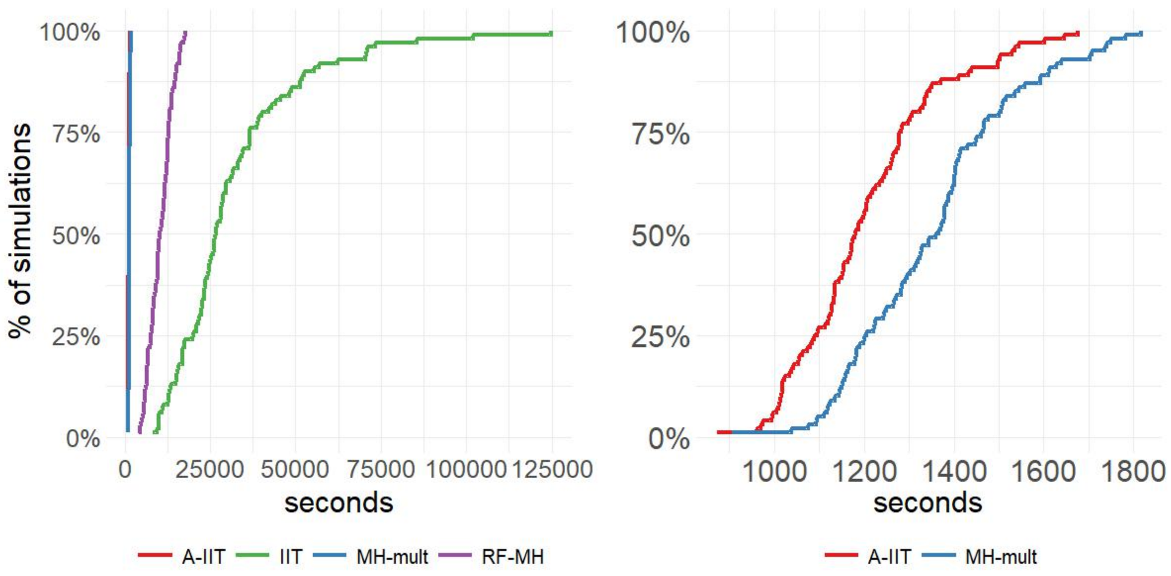

Figure 10: Comparison of time to visit all the modes of the target distribution in a space of dimension $p = 5000$. On the right is the same plot only focusin on A-IIT and MH-mult.

Table 15: Inverse temperature used in the problem of dimension 5k

temp_id	A-IIT	IIT	MH-mult	RF-MH
β_1	20000	20000	20000	20000
β_2	19813	11152	19199	10929
β_3	19524	9798	18256	9985
β_4	19387	8698	17010	9369
β_5	19254	7903	16246	8908
β_6	19135	7290	15526	8538
β_7	19075	6819	14787	8233
β_8	18891	6610	14075	7963
β_9	18680	6427	13404	7723
β_{10}	18616	6274	12786	7523
β_{11}	18536	6125	12388	7422
β_{12}	18454	6007	11783	7299
β_{13}	18421	5889	11230	7208

Table 16: Average swap rate between replica β_i and β_{i+1} in space of dimension 5000

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.4168	0.352	0.3636	0.2386
β_2	0.3777	0.3035	0.3773	0.3558
β_3	0.4079	0.171	0.3478	0.3633
β_4	0.3869	0.1459	0.3838	0.3457
β_5	0.3624	0.1368	0.3722	0.3313
β_6	0.3943	0.147	0.3776	0.3219
β_7	0.3077	0.3699	0.3593	0.269
β_8	0.2868	0.3774	0.3591	0.2445
β_9	0.3337	0.3816	0.3461	0.2021
β_{10}	0.3391	0.3427	0.3786	0.244
β_{11}	0.3133	0.3729	0.306	0.1981
β_{12}	0.3232	0.3278	0.294	0.2217

B.3 Dimension 7000

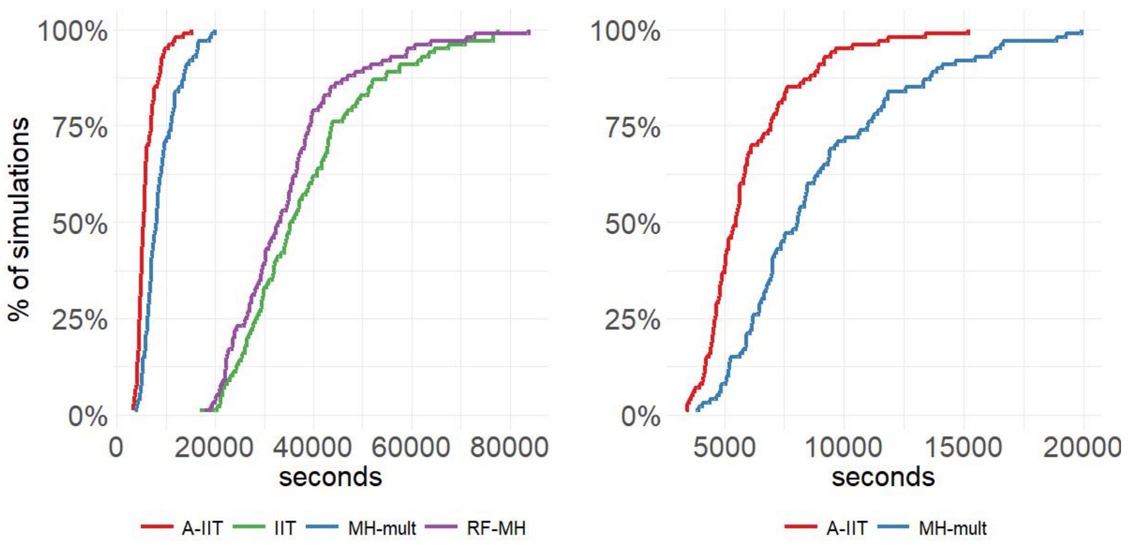

Figure 11: Comparison of time to visit all the modes of the target distribution in a space of dimension $p = 7000$. On the right is the same plot only focus in on A-IIT and MH-mult.

Table 17: Inverse temperature used in the problem of dimension 7k

temp_id	A-IIT	IIT	MH-mult	RF-MH
β_1	20000	20000	20000	20000
β_2	11710	10008	11457	9951
β_3	9796	8637	9845	8781
β_4	9154	7937	9005	8063
β_5	8692	7426	8348	7557
β_6	8327	7024	7813	7135
β_7	8023	6751	7377	6870
β_8	7738	6521	7049	6649
β_9	7490	6310	6741	6483
β_{10}	7274	6185	6472	6371
β_{11}	7069	6110	6239	6301
β_{12}	6878	6032	6029	6252
β_{13}	6713	5996	5858	6214

Table 18: Average swap rate between replica β_i and β_{i+1} in space of dimension 7000

replica	A-IIT	IIT	MH-mult	RF-MH
β_1	0.3965	0.363	0.3854	0.336
β_2	0.2348	0.206	0.2628	0.256
β_3	0.3199	0.2414	0.2614	0.2394
β_4	0.3341	0.2253	0.2418	0.2287
β_5	0.329	0.217	0.2207	0.1984
β_6	0.3428	0.2423	0.2226	0.2348
β_7	0.3314	0.2242	0.2412	0.2136
β_8	0.3336	0.1926	0.2274	0.2091
β_9	0.3264	0.2245	0.2272	0.2153
β_{10}	0.3182	0.257	0.2362	0.2371
β_{11}	0.3081	0.221	0.23	0.233
β_{12}	0.3226	0.295	0.2623	0.2666

C Algorithms

Algorithm 5 Metropolis Hastings with a multiplicity list

```
initialize $X_0$
for $k$ in 1 to $K$ do
    compute $Z(X_{k-1}) = \sum_{Y \in \mathcal{N}_{X_{k-1}}} \mathcal{Q}(Y|X_{k-1}) \min\left\{1, \frac{\pi(Y)\mathcal{Q}(X_{k-1}|Y)}{\pi(X_{k-1})\mathcal{Q}(Y|X_{k-1})}\right\}$
    simulate multiplicity list $M_{k-1} \leftarrow 1 + G$ where $G \sim \text{Geometric}(Z(X_{k-1}))$ with
    Set $w_{k-1}(X_{k-1}) \leftarrow M_{k-1}$
    choose the next state $X_k \in \mathcal{N}_{X_{k-1}}$ such that
        $$\widehat{\mathbf{P}}(X_k = Y \mid X_{k-1}) \propto \mathcal{Q}(Y|X_{k-1}) \min\left\{1, \frac{\pi(Y)\mathcal{Q}(X_{k-1}|Y)}{\pi(X_{k-1})\mathcal{Q}(Y|X_{k-1})}\right\}$$
end for
```

Algorithm 6 Rejection-Free Metropolis Hastings

```
initialize $X_0$
for $k$ in 1 to $K$ do
    compute $Z(X_{k-1}) = \sum_{Y \in \mathcal{N}_{X_{k-1}}} \mathcal{Q}(Y|X_{k-1}) \min\left\{1, \frac{\pi(Y)\mathcal{Q}(X_{k-1}|Y)}{\pi(X_{k-1})\mathcal{Q}(Y|X_{k-1})}\right\}$
    Set $w_{k-1}(X_{k-1}) \leftarrow \frac{1}{Z(X_{k-1})}$
    choose the next state $X_k \in \mathcal{N}_{X_{k-1}}$ such that
        $$\widehat{\mathbf{P}}(X_k = Y \mid X_{k-1}) \propto \mathcal{Q}(Y|X_{k-1}) \min\left\{1, \frac{\pi(Y)\mathcal{Q}(X_{k-1}|Y)}{\pi(X_{k-1})\mathcal{Q}(Y|X_{k-1})}\right\}$$
end for
```

Algorithm 7 Metropolis-Hastings algorithm

```
initialize $X_0$
for $k$ in 1 to $K$ do
    random $Y \in \mathcal{N}_{X_{k-1}}$ based on $\mathcal{Q}(\cdot|X_{k-1})$
    Set $w_{k-1}(X_{k-1}) \leftarrow 1$
    $X_k \leftarrow Y$ with probability $\frac{\pi(Y)\mathcal{Q}(Y,X_{k-1})}{\pi(X_{k-1})\mathcal{Q}(X_{k-1},Y)}$        ▷ Accept proposed move to $Y$
    Otherwise $X_k \leftarrow X_{k-1}$        ▷ Reject and stay at $X_{k-1}$
end for
```