



A Quantum Parallel Markov Chain Monte Carlo

Andrew J. Holbrook

To cite this article: Andrew J. Holbrook (2023): A Quantum Parallel Markov Chain Monte Carlo, Journal of Computational and Graphical Statistics, DOI: [10.1080/10618600.2023.2195890](https://doi.org/10.1080/10618600.2023.2195890)

To link to this article: <https://doi.org/10.1080/10618600.2023.2195890>



© 2023 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 21 Apr 2023.



Submit your article to this journal [↗](#)



Article views: 9



View related articles [↗](#)



View Crossmark data [↗](#)

A Quantum Parallel Markov Chain Monte Carlo

Andrew J. Holbrook

UCLA Biostatistics, Los Angeles, CA

ABSTRACT

We propose a novel hybrid quantum computing strategy for parallel MCMC algorithms that generate multiple proposals at each step. This strategy makes the rate-limiting step within parallel MCMC amenable to quantum parallelization by using the Gumbel-max trick to turn the generalized accept-reject step into a discrete optimization problem. When combined with new insights from the parallel MCMC literature, such an approach allows us to embed target density evaluations within a well-known extension of Grover's quantum search algorithm. Letting P denote the number of proposals in a single MCMC iteration, the combined strategy reduces the number of target evaluations required from $\mathcal{O}(P)$ to $\mathcal{O}(\sqrt{P})$. In the following, we review the rudiments of quantum computing, quantum search and the Gumbel-max trick in order to elucidate their combination for as wide a readership as possible.

ARTICLE HISTORY

Received September 2022
Accepted March 2023

KEYWORDS

Algorithms; Markov chain Monte Carlo (MCMC); Parallel computing; Statistical computing

1. Introduction

Parallel MCMC techniques use multiple proposals to obtain efficiency gains over MCMC algorithms such as Metropolis-Hastings (Metropolis et al. 1953; Hastings 1970) and its progeny that use only a single proposal. Neal (2003) first develops efficient MCMC transitions for inferring the states of hidden Markov models by proposing a “pool” of candidate states and using dynamic programming to select among them. Next, Tjelmeland (2004) considers inference in the general setting and shows how to maintain detailed balance for an arbitrary number P of proposals. Consider a probability distribution $\pi(d\theta)$ defined on \mathbb{R}^D that admits a probability density $\pi(\theta)$ with respect to the Lebesgue measure, that is, $\pi(d\theta) =: \pi(\theta)d\theta$. To generate samples from the target distribution π , we craft a kernel $P(\theta_0, d\theta)$ that satisfies

$$\pi(A) = \int \pi(d\theta_0)P(\theta_0, A) \quad (1)$$

for all $A \subset \mathbb{R}^D$. Letting θ_0 denote the current state of the Markov chain, Tjelmeland (2004) proposes sampling from such a kernel $P(\theta_0, \cdot)$ by drawing P proposals $\Theta_{-0} = (\theta_1, \dots, \theta_P)$ from a joint distribution $Q(\theta_0, d\Theta_{-0}) =: q(\theta_0, \Theta_{-0})d\Theta_{-0}$ and selecting the next Markov chain state from among the current and proposed states with probabilities

$$\pi_p = \frac{\pi(\theta_p)q(\theta_p, \Theta_{-p})}{\sum_{p'=0}^P \pi(\theta_{p'})q(\theta_{p'}, \Theta_{-p'})}, \quad p = 0, \dots, P. \quad (2)$$

Here, if $\Theta = (\theta_0, \dots, \theta_P)$ is the $D \times (P + 1)$ matrix having the current state and all P proposals as columns, then Θ_{-p} is the $D \times P$ matrix obtained by removing the p th column. Tjelmeland (2004) shows that the kernel $P(\theta_0, \cdot)$ constructed in such

a manner maintains detailed balance and hence satisfies (1). Others have since built on this work, developing parallel MCMC methods that generate or recycle multiple proposals (Frenkel 2004; Delmas and Jourdain 2009; Calderhead 2014; Yang et al. 2018; Luo and Tjelmeland 2019; Schwedes and Calderhead 2021; Holbrook 2021). Most recently, Glatt-Holtz et al. (2022) place all these developments in their natural measure theoretical context, allowing one to trivially apply (1) and (2) to distributions over discrete-valued random variables.

Taken together, these developments demonstrate the ability of parallel MCMC methods to alleviate inferential challenges such as multimodality and to deliver performance gains over single-proposal competitors as measured by reduced autocorrelation between samples. In the following, we focus on the specification presented in Algorithm 1 but note that the techniques we present may also be effective for generalizations of this basic algorithm.

One need not use parallel computing to implement Algorithm 1, but the real promise and power of parallel MCMC comes from its natural parallelizability (Calderhead 2014). Contemporary hardware design emphasizes architectures that enable execution of multiple mathematical operations simultaneously. Parallel MCMC techniques stand to leverage technological developments that keep modern computation on track with Moore's Law, which predicts that processing power doubles every two years. For example, the algorithm of Tjelmeland (2004) generates P conditionally independent proposals and then evaluates the probabilities of (2). One may parallelize the proposal generation step using parallel pseudorandom number generators (PRNG) such as those advanced in Salmon et al. (2011). The computational complexity of the target evaluations $\pi(\theta_p)$ is linear in the number of proposals. This presents a significant burden when $\pi(\cdot)$ is

Algorithm 1 Parallel (multiproposal) MCMC (Tjelmeland 2004)

Data: Initial Markov chain state $\theta^{(0)}$; total length of Markov chain S ; total number of proposals per iteration P ; routine for evaluating target density $\pi(\cdot)$; routines for drawing random samples from the proposal distribution $Q(\theta^{(s)}, \cdot)$ and from a $P + 1$ discrete distribution $\text{Discrete}(\pi)$ given some probability vector $\pi = (\pi_0, \dots, \pi_P)$.

Result: A Markov chain $\theta^{(1)}, \dots, \theta^{(S)}$.

```

for  $s \in \{1, \dots, S\}$  do
   $\theta_0 \leftarrow \theta^{(s-1)}$ ;  $\Theta_{-0} \leftarrow Q(\theta_0, \cdot)$ ; for  $p \in \{0, \dots, P\}$  do
     $\pi_p \leftarrow \pi(\theta_p) q(\theta_p, \Theta_{-p})$ 
  end
   $\hat{p} \leftarrow \text{Discrete}(\pi / \pi^T \mathbb{1})$ ;  $\theta^{(s)} \leftarrow \theta_{\hat{p}}$ 
end
return  $\theta^{(1)}, \dots, \theta^{(S)}$ .

```

computationally expensive, for example, in big data settings or for Bayesian inversion, but evaluation of the target density over the P proposals is again a naturally parallelizable task. Moreover, widely available machine learning software such as TENSORFLOW allows users to easily parallelize both random number generation and target evaluations on general purpose graphics processing units (GPU) (Lao et al. 2020). Finally, when generating independent proposals using a proposal distribution of the form $q(\theta_0, \Theta_{-0}) = \prod_{p=1}^P q(\theta_0, \theta_p)$, the acceptance probabilities (2) require the $\mathcal{O}(P^2)$ evaluation of the $\binom{P+1}{2}$ terms $q(\theta_p, \theta_{p'})$, but Holbrook et al. (2021a, 2021b) demonstrate the natural parallelizability of such pairwise operations, obtaining orders-of-magnitude speedups with contemporary GPUs. The proposed method directly addresses the acceptance step of Algorithm 1, while leaving the choice of parallelizing (or not parallelizing) the proposal step to the practitioner.

While parallel MCMC algorithms are increasingly well-suited for developing many-core computational architectures, there are trade-offs that need to be taken into account when choosing how to allocate computational resources. On one end of the spectrum, Gelman and Rubin (1992a, 1992b) demonstrate the usefulness of generating, combining and comparing multiple independent Markov chains that target the same distribution, and one may perform this embarrassingly parallel task by assigning the operations for each individual chain to a separate central processing unit (CPU) core or GPU work-group. In this multi-chain context, simultaneously running multiple parallel MCMC chains could limit resources available for the within-chain parallelization described in the previous paragraph. On the other end of the spectrum, one may find it useful to allocate resources to overcome computational bottlenecks within a single Markov chain that uses a traditional accept-reject step. In big data contexts, Holbrook et al. (2021a, 2021b) and Holbrook, Ji, and Suchard (2022a, 2022b) use multi- and many-core processing to accelerate log-likelihood and log-likelihood gradient calculations within single Metropolis-Hastings and Hamiltonian Monte Carlo (Neal 2011) generated chains. This big data strategy might again limit resources available for parallelization across proposals and target evaluations described above.

In the presence of these tradeoffs, it is worthwhile to develop additional parallel computing tools for parallel MCMC so that future scientists may be better able to flexibly assign sub-routines to different computing resources in a way that is tailored to their specific inference task. In particular, we show that quantum parallelization can be a useful tool for parallel MCMC when evaluation of the target density represents the computational bottleneck.

Quantum computers use quantum mechanics to store information and manipulate data. By leveraging the idiosyncracies of quantum physics, these computers are able to deliver speedups over conventional computing for a relatively small number of computational problems. Some of these speedups are very large. Quantum computers can achieve *exponential* speedups over conventional computers for some computational tasks.. Shor’s quantum algorithm for factoring an integer N (Shor 1994) is polynomial in $\log N$ compared to a super-polynomial classical optimum. The HHL algorithm for solving sparse N -dimensional linear systems (Harrow, Hassidim, and Lloyd 2009) is $\mathcal{O}(\log N)$ compared to a classical $\mathcal{O}(N)$. Other algorithms deliver a still impressive *polynomial* speedup. For example, the algorithms considered in the following (Section 2.2) achieve quadratic speedups over conventional computing, turning $\mathcal{O}(N)$ runtimes into $\mathcal{O}(\sqrt{N})$. Both the magnitude of quantum computing’s successes and the relative rarity of those successes mean that there is a general interest in leveraging quantum algorithms for previously unconsidered computational problems. We will show that—with the help of the Gumbel-max trick (Section B)—established quantum optimization techniques are actually useful for sampling from computationally expensive discrete distributions and apply this insight to achieving quadratic speedups for parallel MCMC.

We provide a short introduction to the most basic elements of quantum computing in Appendix A. The interested reader may look to Nielsen and Chuang (2010) for a canonical introduction to quantum algorithms or Lopatnikova and Tran (2021), Wang and Liu (2022) for surveys geared toward statisticians. In Section 2.1, we review and compare three different quantum search algorithms that enjoy quadratic speedups over conventional algorithms. In Section 2.2 we review the quantum minimization algorithm of Durr and Hoyer (1996) and investigate the use of warm-starting therein. We then combine the Gumbel-max trick (Section B) and recent insights in parallel MCMC with the quantum minimization algorithm to create the quantum parallel MCMC algorithm for both continuously-valued (Section 3.2) and discrete-valued (Section 3.3) targets.

2. Quantum Search and Quantum Minimization

Grover (1996) demonstrates how use a quantum computer to find a single marked element within a finite set of N items with only $\mathcal{O}(\sqrt{N})$ queries, and a result from Bennett et al. (1997) shows that Grover’s algorithm is optimal up to a multiplicative constant. Boyer et al. (1998) extend Grover’s algorithm to multiple marked items or solutions, further extend it to the case when the number of solutions is unknown and provide a rigorous bounds on the algorithms’ error rates. Finally, Durr and Hoyer (1996) use the results of Boyer et al. (1998) to obtain the minimum value within a discrete ordered set. Here, we briefly

Algorithm 2 Quantum search algorithm (Grover 1996)

Data: An oracle gate U_f taking $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ for a function $f(x) : \{0, \dots, N-1\} \rightarrow \{0, 1\}$ that satisfies $f(x_0) = 1$ for a single x_0 ; $n+1 = \log_2(N) + 1$ quantum states initialized to $|0\rangle^{\otimes n}|1\rangle$; an integer $R = \lceil \pi\sqrt{N}/4 \rceil$ denoting the number of Grover iterations.

Result: An n -bit binary string x_0 satisfying $f(x_0) = 1$ with error less than $1/N$.

$$\begin{aligned} |0\rangle^{\otimes n}|1\rangle &\longrightarrow \mathbf{H}^{\otimes n+1}|0\rangle^{\otimes n}|1\rangle = |h\rangle|-\rangle; & /* \quad |h\rangle &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle. & */ \\ |h\rangle|-\rangle &\longrightarrow \mathbf{G}^R |h\rangle|-\rangle \approx |x_0\rangle|-\rangle; & /* \quad \mathbf{G} &= \left(2|h\rangle\langle h| - \mathbf{I}\right)\left(\mathbf{I} - 2|x_0\rangle\langle x_0|\right) & */ \\ |x_0\rangle &\longrightarrow x_0; \text{ return } x_0. \end{aligned}$$

review these advances and provide illustrative simulations. In Section B, the Gumbel-max trick allows us to extend these results to the problem of sampling from a discrete distribution.

2.1. Quantum Search

Given a set of N items and a function $f : \{0, 1, \dots, N-1\} \rightarrow \{0, 1\}$ that evaluates to 1 for a single element, Grover (1996) develops an algorithm that uses quantum parallelism to score quadratic speedups over its classical counterparts. After only $\mathcal{O}(\sqrt{N})$ evaluations of $f(\cdot)$, Grover's algorithm returns the $x \in \{0, 1, \dots, N-1\}$ satisfying $f(x) = 1$ with high probability. Compare this to $\mathcal{O}(N)$ requirement for the randomized classical algorithm that must evaluate $f(\cdot)$ over at least $N/2$ items to obtain the same probability of detecting x . The algorithm takes the state $|0\rangle^{\otimes n}|1\rangle$ as input and applies Hadamard gates to each of the individual $N+1$ input qubits. The resulting state is

$$|h\rangle|-\rangle = \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle|-\rangle.$$

Next, we apply the oracle gate $U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ and note that

$$\begin{aligned} U_f |x\rangle|-\rangle &= U_f |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\ &= \frac{1}{\sqrt{2}} (|x\rangle|0 \oplus f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle) \\ &= -1^{f(x)} |x\rangle|-\rangle. \end{aligned}$$

Thus, U_f flips the sign for the state $|x_0\rangle$ for which $f(x_0) = 1$ but leaves the other states unchanged. If we suppress the ancillary qubit $|-\rangle$, then U_f is equivalent to the gate U_{x_0} defined as $U_{x_0}|x\rangle = -1^{\delta_{x_0}(x)}|x\rangle$. We may succinctly write this gate as the Householder matrix that reflects vectors about the unique hyperplane through the origin that has $|x_0\rangle$ for a normal vector:

$$U_{x_0} = \mathbf{I} - 2|x_0\rangle\langle x_0|.$$

The action of this gate on the state $|h\rangle$ takes the form

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} -1^{\delta_{x_0}(x)} |x\rangle.$$

Next, the algorithm reflects the current state about the hyperplane that has $|h\rangle$ as a normal vector and negates the resulting state:

$$\begin{aligned} &\left(2|h\rangle\langle h| - \mathbf{I}\right) \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} -1^{\delta_{x_0}(x)} |x\rangle \right) \\ &= \frac{1}{\sqrt{N}} \sum_x \left((-1)^{1-\delta_{x_0}(x)} + 2\frac{(N-2)}{N} \right) |x\rangle \\ &= \frac{(3N-4)}{N^{3/2}} |x_0\rangle + \sum_{x \neq x_0} \frac{(N-4)}{N^{3/2}} |x\rangle. \end{aligned}$$

The scientist who measures the state at this moment would obtain the desired state $|x_0\rangle$ with a slightly higher probability of $(3N-4)^2/N^3$ than the individual probabilities of $(N-4)^2/N^3$ for the other states. Each additional application of the *Grover iteration*

$$\mathbf{G} := -U_h U_{x_0} := \left(2|h\rangle\langle h| - \mathbf{I}\right) \left(\mathbf{I} - 2|x_0\rangle\langle x_0|\right) \quad (3)$$

increases the probability of obtaining $|x_0\rangle$ at the time of measurement in the computational basis and $R = \lceil \pi\sqrt{N}/4 \rceil$ iterations guarantees a probability of success that is greater than $1 - 1/N$.

In general, we may use Grover's search algorithm to find a solution when the number of solutions M is greater than 1. While the algorithmic details change little, the number of required Grover iterations

$$R = \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil \quad (4)$$

and the probability of success after those R iterations *do* change (Boyer et al. 1998). When M is much smaller than N , the success rate is greater than $1 - M/N$, and even for large M the success rate is more than $1/2$. Lower-bounds are useful for establishing mathematical guarantees, but it is also helpful to understand the quality of algorithmic performance as a function of M and R . Figure 1 shows success curves as a function of the number of Grover iterations (or oracle calls) applied. The search field contains $2^{14} \approx 16k$ elements, and the number of solutions M varies. Each curve represents an individual search task. The algorithm requires more iterations for smaller numbers and fewer iterations for larger numbers of solutions. The upper bound on error M/N is only an upper bound: the final probability of success for, for example, $M = 256$ is 0.997 compared to the bound of 0.984.

While Grover's algorithm delivers impressive speedups over classical search algorithms, it has a major weakness. Figure 1 hides the fact that the probability of success for Grover's algorithm is *not* monotonic in the number of iterations. Running the

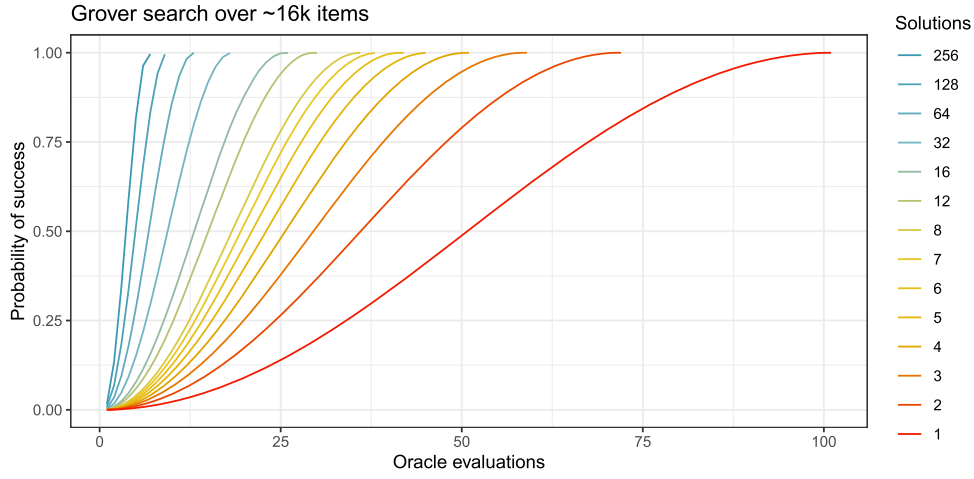


Figure 1. Curves depict the probability that Grover’s quantum search algorithm returns a solution as a function of the number of oracle evaluations. For the ranges depicted, the number of iterations required and the number of solutions inversely relate, but increasing the number of iterations past R (Equation (4)) can backfire and lead to extremely small probabilities of success.

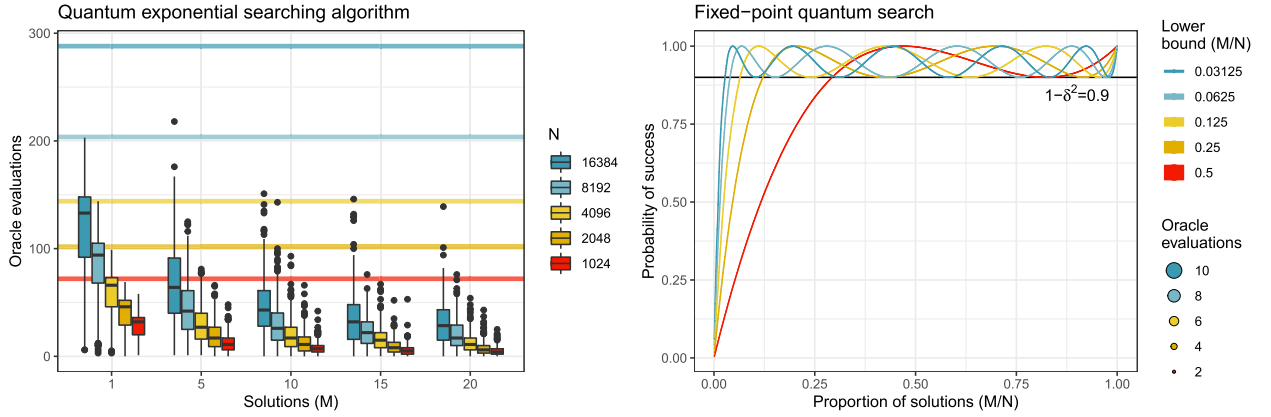


Figure 2. [Left] Total number of oracle evaluations required by quantum exponential searching algorithm (Algorithm 3) for different numbers of solutions M and search set sizes N from 500 independent simulations each. Horizontal lines at $\frac{9}{4}\sqrt{N}$ represent upper bounds on expected total number of evaluations to obtain a solution for the $M = 1$ problem. [Right] Probabilities of success for the fixed-point quantum search algorithm (Yoder, Low, and Chuang 2014) for different proportions of solutions $\lambda = M/N$ and selecting different lower bounds w on M/N with error tolerance δ^2 .

algorithm for more than R iterations can backfire. For example, running the algorithm for $\sqrt{2N}$ iterations when $M = 1$ results in a probability of success less than 0.095 (Boyer et al. 1998). The non-monotonicity of Grover’s algorithm becomes particularly problematic when we do not know the number of solutions M . Taking for example $N = 2^{20}$, Boyer et al. (1998) point out that 804 iterations provide an extremely high probability of success when $M = 1$ but a one-in-a-million chance of success when $M = 4$. To solve this problem and develop an effective search algorithm when M is unknown, those authors adopt the strategy of focusing on the expected number of iterations before success. In particular, they propose the quantum exponential search algorithm (Algorithm 3). When a solution exists, the algorithm returns a solution with expected total number of Grover iterations bounded above by $\frac{2}{2}\sqrt{N/M}$. Still better, this upper bound reduces to $\frac{2}{4}\sqrt{N/M}$ for the special case $M \ll N$, and simulations presented in Figure 2 show that even this bound is large. Such results come in handy when deciding whether to halt the algorithm’s progress if one believes it possible that no solutions exist. Indeed, this turns out to be useful in the context of quantum minimization (Section 2.2).

Algorithm 3 is not the only quantum search algorithm that is useful when the number of solutions M is unknown. Yoder, Low, and Chuang (2014) use *generalized Grover iterations* that extend (3) to include general phases (α_j, β_j) :

$$\begin{aligned} \mathbf{G}(\alpha_j, \beta_j) &:= -\mathbf{U}_h(\alpha_j)\mathbf{U}_{x_0}(\beta_j) \\ &:= \left((1 - e^{-i\alpha_j}) |h\rangle \langle h| - \mathbf{I} \right) \left(\mathbf{I} - (1 - e^{i\beta_j}) |x_0\rangle \langle x_0| \right). \end{aligned} \quad (5)$$

This form reduces to the original Grover iteration (3) when $\alpha = \pm\pi$ and $\beta = \pm\pi$. To select general phases, the method first fixes an error tolerance $\delta^2 > 0$ and a lower bound $w \leq \lambda = M/N$ and then selects an odd upper bound L on the total number of oracle queries $(L - 1)$ such that

$$L \geq \frac{\log(2/\delta)}{\sqrt{w}}.$$

Finally, letting $l = (L - 1)/2$, one obtains phases (α_j, β_j) for $j = 1, \dots, l$ that satisfy

$$\alpha_j = -\beta_{l-j+1} = 2 \cot^{-1} \left(\tan(2\pi j/L) \sqrt{1 - \gamma^2} \right),$$

Algorithm 3 Quantum exponential searching algorithm (Boyer et al. 1998)

Data: An oracle gate U_f taking $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ for a function $f(x) : \{0, \dots, N-1\} \rightarrow \{0, 1\}$ with unknown number of solutions; $n = \log_2(N)$.

Result: If a solution exists, an n -bit binary string x_0 satisfying $f(x_0) = 1$; if no solution exists, the algorithm runs forever.

```

m ← 1; γ ← 6/5; success ← FALSE; while success ≠ TRUE do
  j ← Uniform{0, ..., m-1}; |0⟩⊗n|1⟩ → H⊗n+1|0⟩⊗n|1⟩ = |h⟩|-⟩; |h⟩|-⟩ → Gj|h⟩|-⟩ = |x⟩|-⟩; /* j Grover
  iterations from Algorithm 2. */
  |x⟩ → x; /* Measure and check. */
  if f(x) = 1 then
    | x0 ← x; success ← TRUE;
  else
    | m ← min(γm, √N); /* Increase m in case of failure. */
  end
end
end
return x0.
    
```

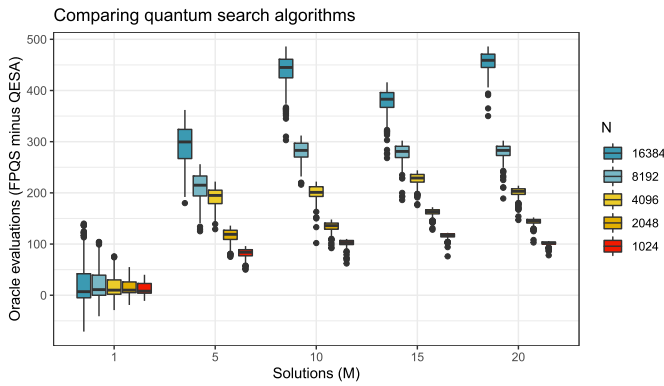


Figure 3. Total number of oracle evaluations required by fixed-point quantum search (FPQS) (Yoder, Low, and Chuang 2014) minus total required for quantum exponential searching algorithm (QESA) (Algorithm 3) for different numbers of solutions M and search set sizes N from 500 independent simulations each. FPQS underperforms partially due to a miniscule lower bound $w = 1/N$ on $\lambda = M/N$, a tuning decision motivated by the potential application within quantum minimization with warm-starting (Section 2.2).

for $\gamma^{-1} = \cos(\frac{1}{L} \cos^{-1}(\frac{1}{\delta}))$ is the reciprocal of the inverse L th Chebyshev polynomial of the first kind. Such phases mark the only algorithmic difference between the fixed-point quantum search and the original Grover search (Algorithm 2). The upshot is a search procedure that obtains a guaranteed probability of success $1 - \delta^2$ for any M such that there exists an $M_0 < M$ that also obtains the same success threshold (Figure 2). On the one hand, this algorithm avoids the exponential quantum search algorithm's need to perform multiple measurements of the quantum state. On the other hand, the exponential quantum search algorithm requires significantly fewer oracle evaluations when M is small (Figure 3), and it turns out that this is precisely the scenario that interests us.

2.2. Quantum Minimization

Given a function $f(\cdot)$ that maps the discrete set $\{0, \dots, N-1\}$ to the integers, we wish to find the minimizer

$$x_0 = \arg \min_{x \in \{0, \dots, N-1\}} f(x).$$

Durr and Hoyer (1996) propose a quantum algorithm for finding x_0 that iterates between updating a running minimum $F \in$

$f(\{0, \dots, N-1\})$ and applying the quantum exponential search algorithm (Algorithm 3) to find an element x such that $f(x) < F$. Having run these iterations a set number of times, the algorithm returns the minimizer x_0 with high probability. To this end, Durr and Hoyer (1996) show that their algorithm takes an expected total time less than or equal to $m_0 := \frac{45}{4} \sqrt{N} + \frac{7}{10} \log_2(N)$ to find the minimizer, where marking items with values less than the threshold value (Algorithm 4) requires $\log_2(N)$ time steps and each Grover iteration within the quantum exponential search algorithm requires one time step. From there, Markov's inequality says

$$\begin{aligned} \Pr \left(\text{total time to success} \geq \frac{m_0}{\epsilon} \right) \\ \leq \frac{E(\text{total time to success})}{m_0/\epsilon} \leq \epsilon, \end{aligned}$$

or that we must scale the minimization procedure's time steps by a factor of $1/\epsilon$ to reach a failure rate less than ϵ . Due to the iterative nature of the algorithm, one might suppose that it is beneficial to start at a value x_0 for which $f(x_0)$ is lower relative to other values. This turns out to be the case in theory and in practice, and the benefit of warm-starting is particularly useful in the context of parallel MCMC.

Proposition 1 (Warm-starting). Suppose that the quantum minimization algorithm begins with a threshold F_0 such that $f(x) < F_0$ for only $K-1 > 0$ items. Then the expected total number of time steps to find the minimizer is bounded above by

$$m_0^K = \left(\frac{5}{4} - \frac{1}{\sqrt{K-1}} \right) 9\sqrt{N} + \frac{7}{10} \log_2(K) \log_2(N),$$

and so the following rule relates the warm-started upper bound to the generic upper bound:

$$m_0^K = m_0 - 9\sqrt{\frac{N}{K-1}} + \frac{7}{10} \log_2\left(\frac{K}{N}\right) \log_2(N).$$

Proof. The proof follows the exact same form as Lemma 2 of Durr and Hoyer (1996). It relies on a theoretical algorithm called the *infinite algorithm* that runs until the minimum is found. In this case, Lemma 1 of that paper says that the probability that the r th lowest value is ever selected when searching among K items

Algorithm 4 Quantum minimum searching algorithm (Durr and Hoyer 1996)

Data: A quantum sub-routine capable of evaluating a function $f(\cdot)$ over $\{0, \dots, N-1\}$ with with unique integer values; a maximum error tolerance $\epsilon \in (0, 1)$; expected total time to success $m_0 = \frac{45}{4}\sqrt{N} + \frac{7}{10}\log_2(N)$.

Result: A $\log_2(N)$ -bit binary string x_0 satisfying $f(x_0) = \min f$ with probability greater than $1 - \epsilon$.

$s \leftarrow 0$; $x_0 \leftarrow \text{Uniform}\{0, \dots, N-1\}$; **while** $s < m_0/\epsilon$ **do**

Prepare initial state $\frac{1}{\sqrt{N}} \sum_x |x\rangle |x_0\rangle$; Mark all items x satisfying $f(x) < f(x_0)$; $s \leftarrow s + \log_2(N)$; Apply quantum exponential searching algorithm (Algorithm 3); /* I time steps */

$s \leftarrow s + I$; Obtain x' by measuring first register; **if** $f(x') < f(x_0)$ **then**

$x_0 \leftarrow x'$

end

end

return x_0 .

is $p(K, r) = 1/r$ for $r \leq K$ and 0 otherwise. For a warm-start at element K , the expected total time spent in the exponential search algorithm is

$$\begin{aligned} \sum_{r=2}^N p(K, r) \frac{9}{2} \sqrt{\frac{N}{r-1}} &= \sum_{r=2}^K p(K, r) \frac{9}{2} \sqrt{\frac{N}{r-1}} \\ &= \frac{9}{2} \sqrt{N} \sum_{r=1}^{K-1} \frac{1}{r+1} \frac{1}{\sqrt{r}} \\ &\leq \frac{9}{2} \sqrt{N} \left(\frac{1}{2} + \sum_{r=2}^{K-1} r^{-3/2} \right) \\ &\leq \frac{9}{2} \sqrt{N} \left(\frac{1}{2} + \int_{r=1}^{K-1} r^{-3/2} dr \right) \\ &= \left(\frac{5}{4} - \frac{1}{\sqrt{K-1}} \right) 9\sqrt{N}. \end{aligned}$$

An upper bound for the expected total number of time steps preparing the initial state and marking items $f(x) < f(x_0)$ follows in a similar manner. \square

Proposition 1 shows that, for example, if Algorithm 4 begins at the item with second lowest value, then the expected total time to success is bounded above by $m_0^2 = \frac{9}{4}\sqrt{N} + \frac{7}{10}\log_2(N)$, reducing the generic upper bound by $9\sqrt{N} - \frac{7}{10}\log_2\left(\frac{2}{N}\right)\log_2(N)$ time steps. When N equals 1000, say, the expected total time steps is $m_0^2 < 78.2$. Keeping $N = 1000$ but letting the algorithm begin at the third lowest value ($K = 3$), the expected total time steps before success is $m_0^3 < 165.6$. Raising N to 10,000, the two numbers increase to $m_0^2 < 234.4$ and $m_0^3 < 503.4$.

In practice, the number of time steps before finding the minimum is surprisingly small. Figure 4 corroborates the intuition that it is beneficial to start at a lower ranked element. To generate these results, we use an early stopping rule for the quantum minimization algorithm's exponential searching sub-routine, halting it after $\frac{9}{4}\sqrt{N}$ iterations. Even with this stopping rule in place, the error rate of the quantum minimization algorithm is less than 1%. We can increase the early stopping threshold to obtain even lower error rates, but this strategy would seem to be unnecessary in the context of parallel MCMC. The benefits of warm-starting are useful in the same context, when the current state $\theta^{(s)}$ usually inhabits the high-density region of the target distribution but the majority of proposals do not.

3. Quantum Parallel MCMC

With the rudiments of quantum minimization in hand, we present a quantum parallel MCMC (QPMCMC). The general structure of the algorithm the same as Algorithm 1: it constructs a Markov chain by iterating between generating many proposals and randomly selecting new Markov chain states among these proposals. Unlike classical single-proposal MCMC, parallel MCMC proposes many states and chooses one according to the generalized acceptance probabilities of (2). In general MCMC, evaluation of the target density function $\pi(\cdot)$ is the rate-limiting computational step, becoming particularly onerous in big data scenarios (Holbrook et al. 2021a). While parallel MCMC benefits from improved mixing as a function of MCMC iterations, the increased computational burden of P target evaluations at each step can lead to less favorable comparisons when accounting for elapsed wall-clock time.

Having successfully generated proposals $\theta_p, p = 1, \dots, P$ and evaluated the corresponding proposal densities $q(\cdot, \cdot)$, we would like to use quantum parallelism and an appropriate oracle gate to efficiently compute the $\pi(\theta_p)$ s but immediately encounter a Catch-22 when we seek to draw a sample $\theta^{(s+1)} \sim \text{Discrete}(\pi)$ for π the vector of probabilities $\pi_0, \pi_1, \dots, \pi_P$ defined in (2). We can use neither a quantum nor a classical circuit to draw the sample! On the one hand, drawing the sample within a quantum circuit would somehow require that all superposed states have access to all the $\pi(\theta_p)$ s at once to perform the required normalization. On the other hand, drawing the sample within the classical circuit would require access to all the $\pi(\theta_p)$ s, but measurement in the computational basis can only return one.

In light of this dilemma, we propose to use the Gumbel-max trick (Papandreou and Yuille 2011) to transform the generalized accept-reject step into a discrete optimization procedure (Section B). From there, we use quantum minimization to efficiently sample from $\text{Discrete}(\pi)$. Crucially, we get away with quantum parallel evaluation of the target $\pi(\cdot)$ over all superposed states because the Gumbel-max trick does not rely on a normalization step: each superposed state requires no knowledge of any target evaluation other than its own.

Once one has effectively parallelized the target evaluations $\pi(\theta_p)$, the new computational bottleneck is the calculations required to obtain the $P+1$ proposal density terms $q(\theta_p, \Theta_{-p})$, for $p = 0, \dots, P$. Under an independent proposal mechanism with a proposal distribution of the form $q(\theta_0, \Theta_{-0}) = \prod_{p=1}^P q(\theta_0, \theta_p)$, the acceptance probabilities (2) require the

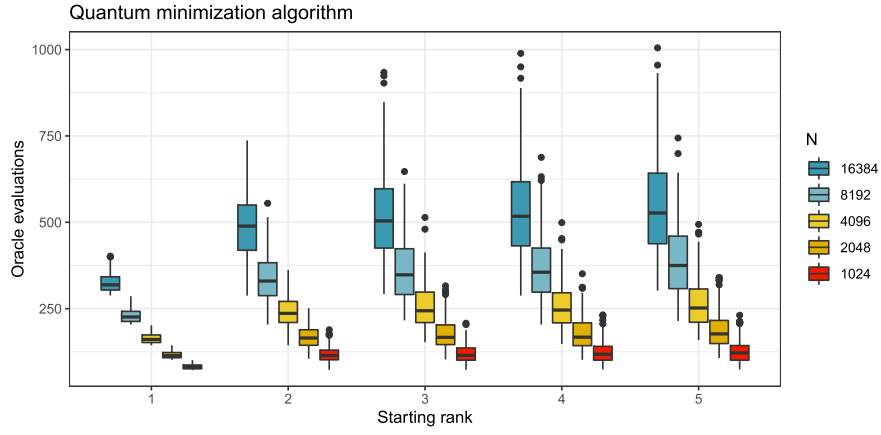


Figure 4. Total number of oracle evaluations required by quantum minimization algorithm (Durr and Hoyer 1996) for different starting ranks and search set sizes N from 500 independent simulations each. Here, less than 1% of the 12,500 instances fail to recover the true minimum on account of early stopping.

$\mathcal{O}(P^2)$ evaluation of the $\binom{P+1}{2}$ terms $q(\theta_p, \theta_{p'})$. To avoid such calculations, we follow Holbrook (2021) and use symmetric proposal distributions for which $q(\theta_p, \Theta_{-p}) = q(\theta_{p'}, \Theta_{-p'})$ for $p, p' = 0, \dots, P$.

Algorithm 5 The Gumbel-max trick

Data: A vector of unnormalized log-probabilities $\lambda^* = \log \pi + \log c$, for π a discrete probability vector with $P + 1$ elements.

Result: A single sample $\hat{p} \sim \text{Discrete}(\pi)$ satisfying $\hat{p} \in \{0, 1, \dots, P\}$.

for $p \in \{0, 1, \dots, P\}$ **do**

 | $z_p \leftarrow \text{Gumbel}(0, 1)$; $\alpha_p^* \leftarrow \lambda_p^* + z_p$;

end

$\hat{p} \leftarrow \arg \max_{p=0, \dots, P} \alpha_p^*$; **return** \hat{p} .

in the manner, but here we focus on Gaussian proposals without loss of generality. Finally, the simplicial sampler (Holbrook 2021) accomplishes the same simplified acceptance probabilities but incurs an $\mathcal{O}(D^3)$ computational cost.

One may also use a more limited strategy to enforce (8). Glatt-Holtz et al. (2022) show that the sampler using independence proposal $q(\theta_p, \Theta_{-p}) = \prod_{p' \neq p} q(\theta_{p'})$, where $q(\cdot)$ is not a function of θ_p , results in acceptance probabilities

$$\pi_p = \frac{\pi(\theta_p)/q(\theta_p)}{\sum_{p'=0}^P \pi(\theta_{p'})/q(\theta_{p'})}, \quad p = 0, \dots, P. \quad (9)$$

When $\pi(\cdot)$ and $q(\cdot)$ take continuous values on a topologically compact domain or discrete values on a finite domain, one may specify $q(\cdot)$ to be uniform and let (9) simplify to (6). This strategy proves useful in Section 3.3, in which we consider discrete-valued targets over Ising-type lattice models.

3.1. Simplified Acceptance Probabilities

Evaluation of the $P+1$ proposal density terms $q(\theta_p, \Theta_{-p})$ can be computationally burdensome as P grows large. Holbrook (2021) proves that two specific proposal mechanisms enforce equality across all $P+1$ terms and thus enable the simplified acceptance probabilities

$$\pi_p = \frac{\pi(\theta_p)}{\sum_{p'=0}^P \pi(\theta_{p'})}, \quad p = 0, \dots, P. \quad (6)$$

In Section 3.2, we opt for one of these proposal mechanisms, the centered Gaussian proposal of Tjelmeland (2004). This strategy first generates a Gaussian random variable $\bar{\theta}$ centered at the current state θ_0 and then generates P proposals centered at $\bar{\theta}$:

$$\theta_1, \dots, \theta_P \stackrel{\perp}{\sim} \text{Normal}_D(\bar{\theta}, \Sigma), \quad \bar{\theta} \sim \text{Normal}_D(\theta_0, \Sigma). \quad (7)$$

Holbrook (2021) shows that this strategy leads to the higher-order proposal symmetry

$$q(\theta_0, \Theta_{-0}) = q(\theta_1, \Theta_{-1}) = \dots = q(\theta_P, \Theta_{-P}) \quad (8)$$

and that the simplified acceptance probabilities (6) maintain detailed balance. In fact, other location scale families also suffice

3.2. Continuously Valued Targets

Algorithm 6 presents the details of QPMCMC for a continuously-valued target. The algorithm uses conventional (perhaps parallel) computing almost entirely, excluding two lines that are highlighted. The first of these loads proposals onto the quantum machine, and the second line that calls the quantum minimization algorithm presented in Algorithm 4. This subroutine seeks to find the minimizer of the function

$$f(p) = -(z_p + \log \pi(\theta_p)) \quad \text{over } p = 0, 1, \dots, P, \quad (10)$$

which combines the numerators of (2), the Gumbel-max trick and a trivial negation. As discussed in Section 2.2, the subroutine requires only $\mathcal{O}(\sqrt{P})$ oracle evaluations and, therefore, only $\mathcal{O}(\sqrt{P})$ evaluations of the target density $\pi(\cdot)$ using a quantum circuit. In theory, a quantum computer can perform the same computations as a classical computer, but the efficiency of the target evaluations would depend on a number of factors related to the structure of the density itself, the availability of fast quantum random access memory (Giovannetti, Lloyd, and Maccone 2008) and the ability of large numbers of quantum gates to act in concert with negligible noise (Kielpinski, Monroe, and Wineland 2002; Erhard et al. 2019).

Algorithm 6 Quantum parallel MCMC for a continuously-valued target

Data: Initial Markov chain state $\theta^{(0)}$; total length of Markov chain S ; total number of proposals per iteration P ; routine for evaluating target density $\pi(\cdot)$; routines for drawing random samples from a D -dimensional multivariate Gaussian $\text{Normal}_D(\mu, \Sigma)$ and the standard Gumbel distribution $\text{Gumbel}(0, 1)$.

Result: A Markov chain $\theta^{(1)}, \dots, \theta^{(S)}$.

```

for  $s \in \{1, \dots, S\}$  do  $\theta_0 \leftarrow \theta^{(s-1)}$ ;  $z_0 \leftarrow \text{Gumbel}(0, 1)$ ;  $\bar{\theta} \leftarrow \text{Normal}_D(\theta_0, \Sigma)$ ;           /* Proposal (7) */
for  $p \in \{1, \dots, P\}$  do  $\theta_p \leftarrow \text{Normal}_D(\bar{\theta}, \Sigma)$ ;  $z_p \leftarrow \text{Gumbel}(0, 1)$ ;  $|\theta_p\rangle \leftarrow \theta_p$ ;           /* Load proposal onto quantum computer. */

```

```

 $\hat{p} \leftarrow \arg \min_{p=0, \dots, P} (f(p) := -(z_p + \log \pi(\theta_p)))$ ;           /*  $\mathcal{O}(\sqrt{P})$  Algorithm 4 with warm-start at  $p=0$ . */
 $\theta^{(s)} \leftarrow \theta_{\hat{p}}$ ; return  $\theta^{(1)}, \dots, \theta^{(S)}$ .

```

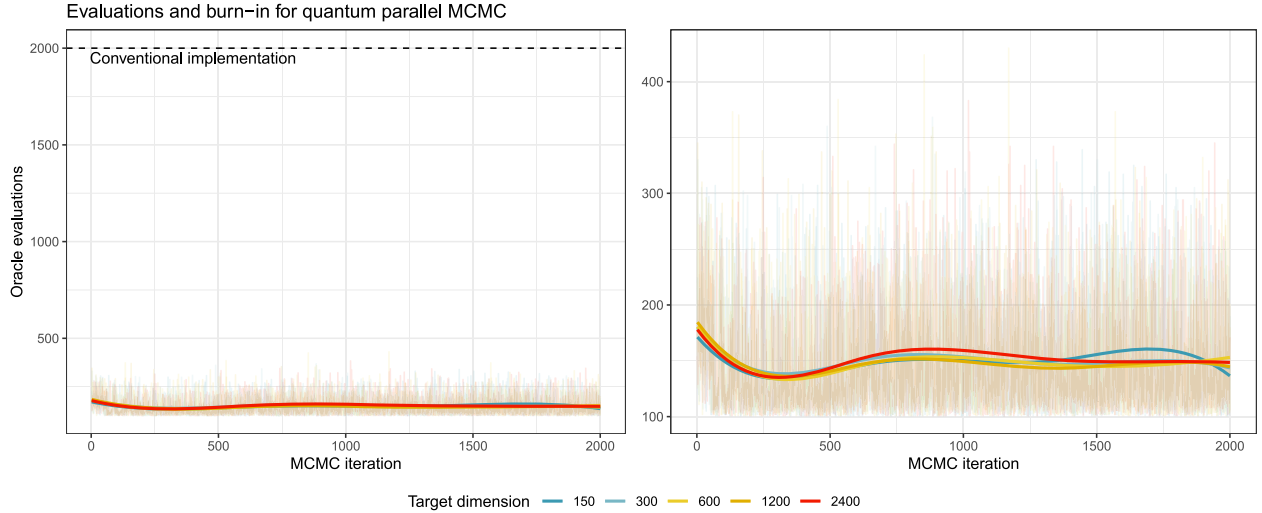


Figure 5. Total number of oracle evaluations required for each of 2000 quantum parallel MCMC (QPMCMC) iterations for standard multivariate normal targets of five different dimensionalities. Regardless of target dimension, the individual QPMCMC runs require roughly 7% of the usual 4 million target evaluations. Over 99.4% of the 10,000 MCMC iterations across all dimensions successfully sample from the discrete distribution with probabilities of (2). Burn-in iterations require moderately more evaluations because the current state occupies a lower density region and represents a “less good” warm-start.

In general MCMC, one often calibrates the scaling of a proposal kernel to balance between exploring the target’s domain and remaining within high-density regions. Optimal scaling strategies may lead to a large number of rejected proposals (Roberts and Rosenthal 2001). Indeed, Holbrook (2021) shows that parallel MCMC algorithms are no exception. When using the Gumbel-max trick to sample from proposals, this means that the current state is often quite near optimality, representing a warm-start. Figure 5 shows how this warm-starting effects the number of oracle calls (and hence target evaluations) within the quantum minimization sub-routine over the course of an MCMC trajectory. We target multi-dimensional standard normal distributions of differing dimensions using the vector $(100, \dots, 100)$ as starting position. We fix the number of Gaussian proposals to be 2000 and use standard adaptation procedures (Rosenthal 2011) to target a 50% acceptance rate while guaranteeing convergence. Although no theoretical results validate this target acceptance rate, the rate is close to empirically optimal rates from Holbrook (2021). Across iterations, QPMCMC requires relatively few oracle calls compared to the 2000 target evaluations of parallel MCMC. We also witness the expected drop in the number of oracle evaluations as the chain approaches the target’s high-density region. Remarkably, the algorithm succeeds in sampling from the true discrete

distribution in 99.5% of the MCMC iterations. An independent simulation obtains similar results, requiring roughly 7% of the usual number of target evaluations. Figure 6 shows a quantile-quantile plot for all 100 dimensions of a multi-dimensional standard normal distribution. We see no appreciable impact from the rare failure of the quantum minimization sub-routine.

Finally, we compare convergence speed for QPMCMC using 1000, 5000, and 10,000 proposals to target a massively multimodal distribution: a mixture of 1000 two-dimensional, isotropic Gaussians with standard deviations equal to 1 and means equal to $10 \times (\mathbf{0}, \mathbf{1}, \dots, \mathbf{999})$. This target is particularly challenging because the distance between modes is significantly larger than standard deviations. In five independent simulations, we run each algorithm until it achieves an effective sample size of 100. Table 1 contains MCMC iterations and target evaluations required to achieve this effective sample size as well as relative speedups over sequential implementations and relative improvements over the 1000 proposal sampler.

3.3. Discrete-Valued Targets

We wish to sample from a target distribution defined over a discrete set $\{\theta_\alpha\}_{\alpha \in \mathcal{A}}$, for \mathcal{A} some finite or countably infinite set of indices. Glatt-Holtz et al. (2022) establish the broader

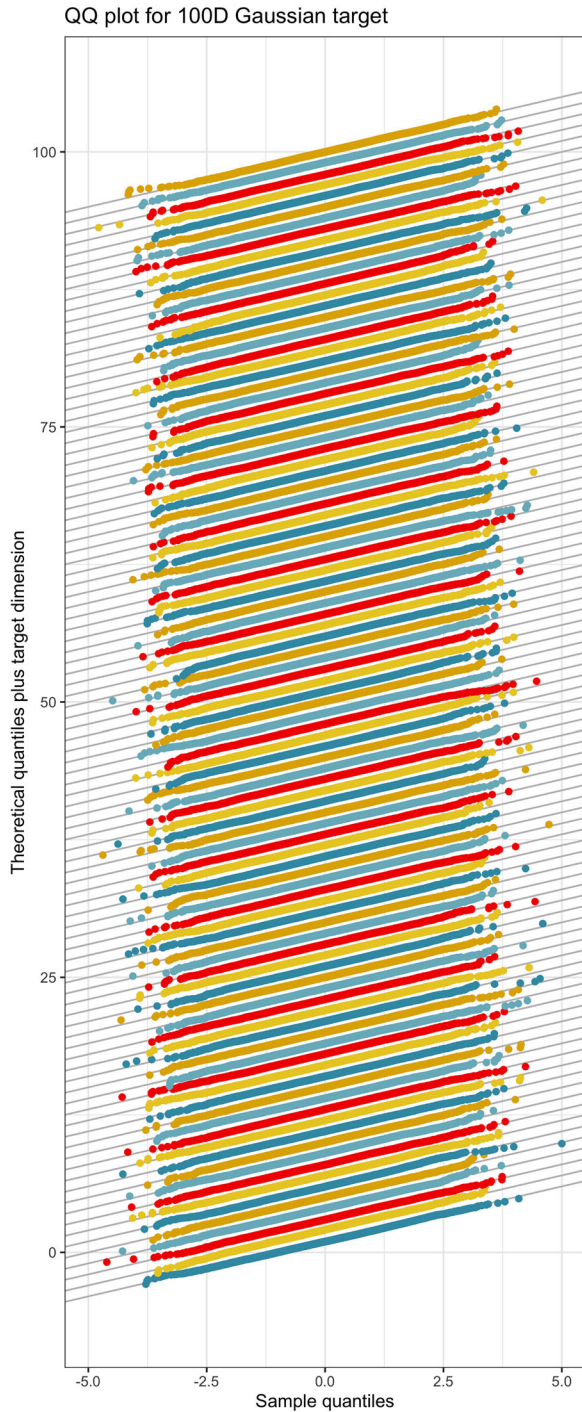


Figure 6. Empirical accuracy of quantum parallel MCMC (QPMCMC) for a 100 dimensional spherical Gaussian target. We generate 100,000 samples using 2000 proposals each iteration and remove the first 2000 samples as burn-in. The QQ (quantile-quantile) plot shows sample quantiles adhering closely to the theoretical quantiles. Similar to the independent simulation shown in Figure 5, here QPMCMC requires less than 7.2% of the usual number of target evaluations.

measure theoretical foundations for parallel MCMC procedures that use selection probabilities (2) to maintain detailed balance. In particular, when the target distribution has probability mass function $\pi(\cdot)$ defined with respect to the counting measure on the power set $2^{\mathcal{A}}$, then detailed balance results in the kernel $P(\cdot, \cdot)$ satisfying

$$\pi(\alpha) = \sum_{\alpha'} \pi(\alpha') P(\alpha', \alpha), \quad \forall \alpha, \alpha' \in \mathcal{A}.$$

Sections 3.3.1 and 3.3.2 consider targets defined over finite sets \mathcal{A} and make use of the uniform independence proposal scheme described in Section 3.1, thereby facilitating simplified acceptance probabilities (6). This scheme proposes single-flip updates to the current Markov chain state θ_0 , but it is worth noting that other multiple-flip schemes would also be amenable to QPMCMC.

3.3.1. Ising Model on a Two-Dimensional Lattice

Here, we are interested in an Ising-type lattice model over configurations $\theta = (\theta_1, \dots, \theta_D)$ consisting of D individual spins $\theta_d \in \{-1, 1\}$. In terms of the preceding section, we have $\mathcal{A} = \{-1, 1\}^D$ and $|\mathcal{A}| = 2^D$. In particular, we consider targets of the form

$$\pi(\theta|\rho) \propto \exp\left(\rho \sum_{(d,d') \in \mathcal{E}} \theta_d \theta_{d'}\right), \quad (11)$$

where $\rho > 0$ is the interaction term and \mathcal{E} is the lattice edge set. We specify a two-dimensional 500-by-500 lattice with nearest neighbors connections and fix $\rho = 1$. Since this latter setting encourages neighboring spins to equal each other, we begin our QPMCMC trajectories at the lowest-probability “checkerboard” state for an initial configuration. At each iteration, we generate collections of proposal states by uniformly flipping P individual spins θ_p , $p \in \{1, \dots, P\}$, and obtaining each proposal state θ_p by updating the current state θ_0 at the single location corresponding to θ_p . Figure 7 shows results from 10 independent runs using $P \in \{4, 8, 16, \dots, 2048\}$ proposals. Trace plots of unnormalized log-probabilities indicate that higher proposal counts enable discovery of higher probability configurations. Interestingly, QPMCMC appears to be particularly beneficial in this large P context, requiring less than one-tenth the target evaluations required using conventional computing when $P = 2048$.

3.3.2. Bayesian Image Analysis

We apply a Bayesian image classification model to an intensity map (Figure 8) of the newly imaged supermassive black hole, Sagittarius A*, located at the Galactic Center of the Milky Way (Akiyama et al. 2022). Whereas one cannot see the black hole itself, one may see the shadow of the black hole cast by the hot, swirling cloud of gas surrounding it. We extract the black hole from the surrounding cloud by modeling the intensity at each of the $D = 4076^2 = 16,613,776$ pixels as belonging to a mixture of two truncated normals with values y_d restricted to the intensity range $[0, 255]$. Namely, we follow Hurn (1997) and specify the latent variable model

$$\begin{aligned} y_d | (\mu_\ell, \sigma^2, \theta_d) &\stackrel{\text{iid}}{\sim} \text{Normal}(\mu_\ell, \sigma^2), \quad y_d \in [0, 255], \\ \theta_d &= \ell, \quad d \in \{1, \dots, D\}, \\ \mu_\ell &\stackrel{\text{iid}}{\sim} \text{Uniform}(0, 255), \quad \ell \in \{-1, 1\}, \\ \frac{1}{\sigma^2} &\sim \text{Gamma}\left(\frac{1}{2}, \frac{1}{2}\right), \end{aligned}$$

where $\theta = (\theta_1, \dots, \theta_D)$ share for a prior the Ising model (11) with edges joining neighboring pixels and interaction $\rho = 1.2$.

Table 1. Racing to a minimum (between the two dimensions) of 100 effective samples for a target with 1000 disjoint modes.

Proposals	MCMC iterations	Target evaluations	Speedup	Efficiency gain
1000	249,398 (200,998, 311,998)	24,988,963 (20,149,132, 31,265,011)	9.98 (9.98, 9.98)	1
5000	14,398 (12,998, 16,998)	3,314,560 (2,989,418, 3,916,281)	21.72 (21.70, 21.74)	7.58 (6.25, 9.71)
10,000	5998 (4998, 6998)	1,993,484 (1,662,592, 2,330,842)	30 (29.96, 30.26)	12.87 (8.64, 18.80)

NOTE: “Speedup” is ratio between target evaluations required for sequential and quantum implementations. “Efficiency gain” is ratio between target evaluations required for 1000 proposal and 5000/10,000 proposal implementations. We report means (minima, maxima) across five independent runs.

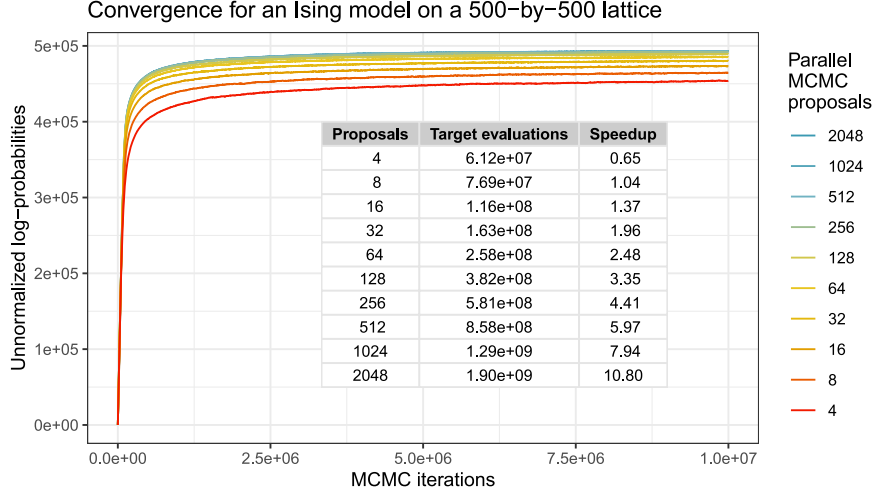


Figure 7. Convergence of log-posterior for a parallel MCMC sampler targeting an Ising model with uniform interaction $\rho = 1$ and no external field. All chains begin at minimal probability “checkerboard” state. Larger proposal counts allow discovery of higher probability configurations.

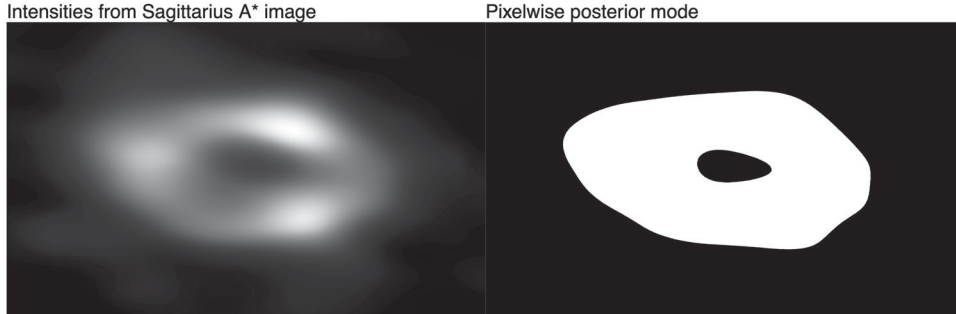


Figure 8. On the left is a 4076-by-4076 intensity map of the shadow of supermassive black hole Sagittarius A* (Akiyama et al. 2022). On the right is the pixelwise posterior mode of a Bayesian image classification model fit to intensity data. Within a Metropolis-in-Gibbs routine, quantum parallel MCMC using 1024 proposals requires less than one-tenth the posterior evaluations required by conventional parallel MCMC.

We use a QPMCMC-within-Gibbs scheme to infer the joint posterior of θ and the three mixture model parameters. For the former, we use the same QPMCMC scheme as in Section 3.3.1 with 1024 proposals at each iteration. For the latter, we use the closed-form updates presented in Hurn (1997). We run this scheme for 20 million iterations, discarding the first 10 million as burnin. We thin the remaining sample at a ratio of 1 to 40,000 for the latent memberships θ and 1 to 4000 for the three parameters μ_{-1} , μ_1 and σ^2 . Using the R package CODA (Plummer et al. 2006), we calculate effective sample sizes of the log-posterior (257.1), μ_{-1} (1,578.7), μ_1 (257.6) and σ^2 (2500.0), suggesting adequate convergence. Figure 8 shows both the intensity data and the pixelwise posterior mode of the latent membership vector θ . The QPMCMC requires only 1,977,553,608 target evaluations compared to the $1024 \times 20,000,000 = 2.048 \times 10^{10}$ evaluations required for the analogous parallel MCMC scheme implemented on a conventional computer, representing a 10.36-fold speedup.

4. Discussion

We have shown that parallel MCMC enjoys quadratic speedups by combining quantum minimization with the Gumbel-max trick. Within a QPMCMC iteration, the current state represents a warm-start for the sampling-turned-optimization problem, leading to increased efficiencies for the quantum minimization algorithm. Moreover, combining this approach with the Tjelmeland correction (Holbrook 2021) results in a total complexity reduction from $\mathcal{O}(P^2)$ to $\mathcal{O}(\sqrt{P})$. Preliminary evidence suggests that our strategy may find use when target distributions exhibit extreme multimodality. While the algorithm still must construct long Markov chains to reach equilibrium, generating each individual Markov chain state requires significantly fewer target evaluations.

There are major technical barriers to the practical implementation of QPMCMC. The framework, like other quantum machine learning (QML) schemes, requires on-loading and off-

loading classical data to and from a quantum machine. In the seminal review of QML, Biamonte et al. (2017) discuss what they call the “input problem.” Cortese and Braje (2018) present a general purpose quantum circuit for loading B classical bits into a quantum data structure comprising $\log_2(B)$ qubits with circuit depth $\mathcal{O}(\log(B))$. For continuous targets, QPMCMC requires reading $\mathcal{O}(P)$ real vectors $\theta_p \in \mathbb{R}^D$ onto the quantum machine at each MCMC iteration. If b is the number of bits used to represent a single real value, then reading $B = \mathcal{O}(PDb)$ classical bits into the quantum machine requires time $\mathcal{O}(\log(PDb))$. This is true whether one opts for fixed-point (Jordan 2005) or floating-point (Haener et al. 2018) representations for real values. The outlook can be better for discrete distributions. For example, the QPMCMC scheme presented in Section 4 only requires loading the entire configuration state once prior to sampling. A D -spin Ising model requires D classical bits to encode, and these bits load onto a quantum machine in time $\mathcal{O}(\log(D))$. Once one has encoded the entire system state, each QPMCMC iteration only requires loading the addresses of the $\mathcal{O}(P)$ proposal states. If one uses b bits to encode each address, then the total time required to load data onto the quantum machine is $\mathcal{O}(\log(Pb))$ for each QPMCMC iteration. On the other hand, the speedup for discrete targets assumes the ability to hold an entire configuration in QRAM. Conveniently, the “output problem” is less of an issue for QPMCMC, as only a single integer $\hat{p} \in \{0, \dots, P\}$ need be extracted within Algorithm 4.

This work leads to three interesting questions. First, what is the status of inference obtained by QPMCMC? The QPMCMC selection step relies on quantum minimization, an algorithm that only achieves success with probability $1 - \epsilon$. While empirical studies suggest that this error induces little bias, it would be helpful to use this ϵ to bound the distance between the target distribution and the distribution generated by QPMCMC. Such theoretical efforts would need to extend recent formalizations of the multiproposal based parallel MCMC paradigm (Glatt-Holtz et al. 2022) to account for biased MCMC kernels.

Second, can QPMCMC be combined with established quantum algorithms that make use of “quantum walks” on graphs to sample from discrete target distributions? Szegedy (2004) presents a quantum analog to classical ergodic reversible Markov chains and shows that such quantum walks sometimes provide quadratic speedups over classical Markov chains. Szegedy (2004) also points out that Grover’s search, a key component within QPMCMC, may be interpreted as performing just such a quantum walk on a complete graph. Wocjan and Abeyesinghe (2008) accelerate the quantum walk by using ancillary Markov chains to improve mixing and apply their method to simulated annealing. Given a quantum algorithm \mathbb{A} for producing a discrete random sample with variance σ^2 , Montanaro (2015) develops a quantum algorithm for estimating the mean of algorithm \mathbb{A} ’s output with error ϵ by running algorithm \mathbb{A} a mere $\tilde{\mathcal{O}}(\sigma/\epsilon)$ times, where $\tilde{\mathcal{O}}$ hides polylogarithmic terms. Importantly, this quadratic quantum speedup over classical Monte Carlo applies for quantum algorithms \mathbb{A} that only feature a single measurement such as certain simple quantum walk algorithms. Unfortunately, this assumption fails for quantizations of Metropolis-Hastings, in general, and QPMCMC, in particular. More promising for QPMCMC, Lemieux et al. (2020) develop a quantum circuit that applies Metropolis-Hastings to the Ising

model without the need for oracle calls. An interesting question is whether similar non-oracular quantum circuits exist for the basic parallel MCMC backbone to QPMCMC. In general, however, comparison between QPMCMC and other quantum Monte Carlo techniques is challenging because the foregoing literature (a) largely focuses on MCMC as a tool for discrete optimization, with algorithms that only ever return a single Monte Carlo sample or function thereof, and (b) restricts itself to a handful of simple, stylized and discrete target distributions. On the other hand, QPMCMC is a general inferential framework for sampling from general discrete and continuous distributions alike.

Third, there are other models and algorithms in statistics, machine learning and statistical mechanics that require sampling from potentially costly discrete distributions. Can one adapt our quantum Gumbel-max trick to these? Approximate Bayesian computation (Csilléry et al. 2010) extends probabilistic inference to contexts within which one only has access to a complex, perhaps computationally intensive, forward model. Having sampled model parameters from the prior, one accepts or rejects these parameter values based on the discrepancy between the observed and simulated data. If one succeeds in embedding forward model dynamics within a quantum circuit, then one may plausibly select from many parameter values using our quantum Gumbel-max trick. The trick may also find use within sequential Monte Carlo (Doucet, De Freitas, and Gordon 2001). For example, Berzuini and Gilks (2001) present a sequential importance resampling algorithm that uses MCMC-type moves to encourage particle diversity and avoid the need for bootstrap resampling. Multiproposals accelerated by the quantum Gumbel-max trick could add speed and robustness to such MCMC-resampling.

Large-scale, practical quantum computing is still a long way off, but quantum algorithms are ripe for mainstream computational statistics.

Appendix A. Limited Introduction to Quantum Computing

Quantum computers perform operations on unit-length vectors of complex data called quantum bits or *qubits*. One may write any qubit $|\psi\rangle$ as a linear combination of the *computational basis states* $|0\rangle$ and $|1\rangle$. In symbols,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad \text{for } \alpha, \beta \in \mathbb{C} \quad \text{and} \quad |\alpha|^2 + |\beta|^2 = 1.$$

This formula uses *Dirac* or *bra-ket* notation and obscures ideas that are commonplace in the realm of applied statistics. We make the unit-vector specification of $|\psi\rangle$ clear by writing

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{or} \quad |\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The full machinery of linear algebra is also available within this notation. The conjugate transpose of $|\psi\rangle$ is $\langle\psi|$. The inner product of $|\psi\rangle$ and $|\phi\rangle$ is $\langle\phi|\psi\rangle$. The outer product is $|\psi\rangle\langle\phi|$. Naturally, we can write $|\psi\rangle$ as a linear combination of any other such basis elements. Consider instead the vectors

$$|+\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \quad \text{and} \quad |-\rangle = \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle.$$

A little algebra shows that $|+\rangle$ and $|-\rangle$ are indeed unit-length and orthogonal to each other. A little more algebra reveals that, with respect

to this basis, ψ has coefficients α' and β' given by $(\alpha + \beta)/\sqrt{2}$ and $(\alpha - \beta)/\sqrt{2}$, respectively. A last bit of algebra shows that this representation is consistent with ψ 's unit length.

But linear algebra is not the only aspect of quantum computing that should come easily to applied statisticians. In addition to thinking of ψ as a vector, it is also useful to think of ψ as a (discrete) probability distribution over the computational basis states $|0\rangle$ and $|1\rangle$. The constraints on coefficients α and β mean that we can think of $|\alpha|^2$ and $|\beta|^2$ as probabilities that ψ is in state $|0\rangle$ or $|1\rangle$, respectively. Accordingly, $|+\rangle$ and $|-\rangle$ encode uniform distributions over the computational basis states. In the parlance of quantum mechanics, α , β and $\pm 1/\sqrt{2}$ are all *probability amplitudes*, and ψ , $|+\rangle$ and $|-\rangle$ are all *superpositions* of the computational basis states. Quantum *measurement* of ψ results in $|0\rangle$ with probability $|\alpha|^2$, but—in the following—we can safely think of this physical procedure as drawing a single discrete sample from ψ 's implied probability distribution.

Quantum logic gates perform linear operations on qubits like ψ and take the form of unitary matrices \mathbf{U} satisfying $\mathbf{U}^\dagger \mathbf{U} = \mathbf{I}$ for \mathbf{U}^\dagger the conjugate transpose of \mathbf{U} . One terrifically important single-qubit quantum gate is the *Hadamard gate*

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

One may verify that \mathbf{H} is indeed unitary and that its action maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$. In fact, the reverse is also true on account of the symmetry of \mathbf{H} . The Hadamard gate thus takes the computational basis states in and out of superposition, facilitating a phenomenon called *quantum parallelism*. Given a function $f : \{0, 1\} \rightarrow \{0, 1\}$, consider the two-qubit quantum *oracle gate* \mathbf{U}_f which takes $|x\rangle |y\rangle$ as input and returns $|x\rangle |y \oplus f(x)\rangle$ as output, where \oplus denotes addition modulo 2. Importantly, the output simplifies to $|x\rangle |f(x)\rangle$ for $y = 0$. We now consider a quantum circuit that acts on two qubits by first applying the Hadamard transform \mathbf{H} to the first qubit and then applying the oracle gate \mathbf{U}_f to both. Using the state $|0\rangle |0\rangle$ as input, we have

$$|0\rangle |0\rangle \longrightarrow \frac{1}{\sqrt{2}} |0\rangle |0\rangle + \frac{1}{\sqrt{2}} |1\rangle |0\rangle \longrightarrow \frac{1}{\sqrt{2}} |0\rangle |f(0)\rangle + \frac{1}{\sqrt{2}} |1\rangle |f(1)\rangle. \quad (\text{A.1})$$

The quantum circuit evaluates $f(\cdot)$ simultaneously over both inputs! Unfortunately, the scientist who implements this circuit cannot directly access \mathbf{U}_f 's output, and measurement will provide only $|0\rangle |f(0)\rangle$ or $|1\rangle |f(1)\rangle$ with probability $1/2$ each. Unlocking the potential of quantum parallelism requires more ingenuity.

Uncountably many single- and two-qubit quantum gates exist, but the real power of quantum computing stems from the development of complex quantum gates that act on multiple qubits simultaneously. To access this power, we need one more tool that also appears in the statistics literature. The *Kronecker* or *tensor product* between an L -by- M matrix \mathbf{A} and an N -by- O matrix \mathbf{B} is the LN -by- MO matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & \cdots & A_{1M}\mathbf{B} \\ \vdots & \ddots & \vdots \\ A_{L1}\mathbf{B} & \cdots & A_{LM}\mathbf{B} \end{bmatrix}. \quad (\text{A.2})$$

In statistics, the Kronecker product features in the definition of a matrix normal distribution and is sometimes helpful when specifying the kernel function of a multivariate Gaussian process (Werner, Jansson, and Stoica 2008). Here, the product is equivalent to the parallel action of individual quantum gates on individual qubits. Simple application of Formula (A.2) shows that

$$\mathbf{H}^{\otimes 2} = \mathbf{H} \otimes \mathbf{H} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad \text{and}$$

$$|0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (\text{A.3})$$

We may also denote the left product $\mathbf{H}^{\otimes 2}$ and the right product $|0\rangle^{\otimes 2}$, $|00\rangle$ or $|0\rangle |0\rangle$. One may therefore express (A.1) as a series of transformations applied to the 4-vector on the very right. Letting $|10\rangle$, $|01\rangle$ and $|11\rangle$ take on analogous meanings to $|00\rangle$, an immediate result of (A.3) is that

$$\mathbf{H}^{\otimes 2} |00\rangle = \frac{1}{2} \left(|00\rangle + |01\rangle + |10\rangle + |11\rangle \right). \quad (\text{A.4})$$

The action of $\mathbf{H}^{\otimes 2}$ transforms $|00\rangle$ into a superposition of the states $|00\rangle$, $|10\rangle$, $|01\rangle$ and $|11\rangle$, where the probability of selecting each is a uniform $(1/2)^2 = 1/4$. Writing so many 0s and 1s is tedious, so an alternative notation becomes preferable. Exchanging $|0\rangle$ for $|00\rangle$, $|1\rangle$ for $|01\rangle$, $|2\rangle$ for $|10\rangle$, and $|3\rangle$ for $|11\rangle$, (A.4) becomes the more succinct

$$\mathbf{H}^{\otimes 2} |00\rangle = \frac{1}{2} \left(|0\rangle + |1\rangle + |2\rangle + |3\rangle \right).$$

This formula extends generally to operations over n qubits. Now letting $N = 2^n$, we have

$$\mathbf{H}^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \left(|0\rangle + |1\rangle + \cdots + |N-1\rangle \right) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle =: |h\rangle,$$

and we call $|h\rangle$ a *uniform superposition* over the states $|0\rangle, \dots, |N-1\rangle$. The many-qubit analogue for the quantum parallelism of (A.1) is then

$$|0\rangle^{\otimes n} |0\rangle \longrightarrow \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \right) |0\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle.$$

Appendix B. Gumbel-max

We wish to randomly select a single element from the set $\{0, 1, \dots, P\}$ with probability proportional to the unnormalized probabilities $\boldsymbol{\pi}^* = (\pi_0^*, \pi_1^*, \dots, \pi_P^*)$. That is, there exists a $c > 0$ such that $\boldsymbol{\pi}^* = c\boldsymbol{\pi}$, for $\boldsymbol{\pi}$ a valid probability vector, but we only have access to $\boldsymbol{\pi}^*$. Define $\lambda := \log \boldsymbol{\pi}$ and $\lambda^* := \log \boldsymbol{\pi}^* = \log \boldsymbol{\pi} + \log c$, and assume that $z_0, \dots, z_P \stackrel{\text{iid}}{\sim} \text{Gumbel}(0, 1)$. Then, the probability density function $g(\cdot)$ and cumulative distribution function $G(\cdot)$ for each individual z_p are

$$g(z_p) = \exp(-z_p - \exp(-z_p)) \quad \text{and} \quad G(z_p) = \exp(-\exp(-z_p)).$$

Now, defining the random variables $\alpha_p^* := \lambda_p^* + z_p$, $\alpha_p := \lambda_p + z_p$ and

$$\hat{p} = \arg \max_{p=0, \dots, P} \alpha_p^*,$$

we have the result

$$\Pr(\hat{p} = p) = \pi_p.$$

In words, the procedure of adding Gumbel noise to unnormalized log-probabilities and taking the index of the maximum produces a random

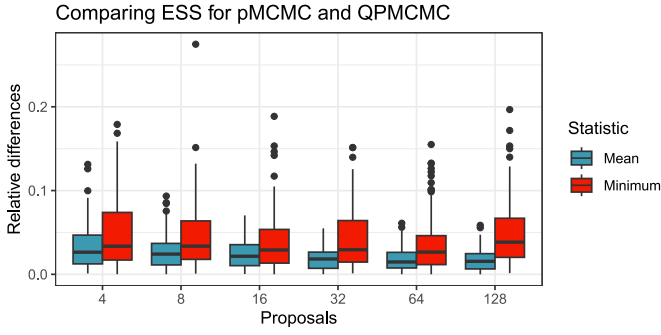


Figure B.1. Relative differences between effective sample sizes (ESS) for parallel MCMC (pMCMC) and quantum parallel MCMC (QPMCMC) across a range of proposal counts. We target a 10-dimensional standard normal distribution. For each algorithm and each proposal setting, we generate 100 independent chains of length 10,000 and calculate the mean and minimum ESS across dimensions.

variable that follows the discrete distribution over π . Moving from left to right:

$$\begin{aligned}
 \Pr(\hat{p} = p) &= \Pr(\alpha_p^* > \alpha_{p'}, \forall p' \neq p) \\
 &= \Pr(\alpha_p + \log c > \alpha_{p'} + \log c, \forall p' \neq p) \\
 &= \Pr(\alpha_p > \alpha_{p'}, \forall p' \neq p) \\
 &= \int_{-\infty}^{\infty} \prod_{p' \neq p} \Pr(\alpha_p > \alpha_{p'} | \alpha_p) g(\alpha_p - \lambda_p) d\alpha_p \\
 &= \int_{-\infty}^{\infty} \prod_{p' \neq p} G(\alpha_p - \lambda_{p'}) g(\alpha_p - \lambda_p) d\alpha_p \\
 &= \int_{-\infty}^{\infty} \prod_{p' \neq p} \exp(-\exp(\lambda_{p'} - \alpha_p)) \\
 &\quad \exp(-\alpha_p + \lambda_p - \exp(-\alpha_p + \lambda_p)) d\alpha_p.
 \end{aligned}$$

Recalling that $\lambda_{p'} = \log \pi_{p'}$, we exponentiate the logarithms, and the integral becomes

$$\begin{aligned}
 \pi_p \int_{-\infty}^{\infty} \prod_{p' \neq p} \exp(-\pi_{p'} \exp(-\alpha_p)) \exp(-\alpha_p - \pi_p \exp(-\alpha_p)) d\alpha_p \\
 = \pi_p \int_{-\infty}^{\infty} \exp(-\alpha_p) \exp\left(-\sum_{p'=0}^P \pi_{p'} \exp(-\alpha_p)\right) d\alpha_p \\
 = \pi_p \int_{-\infty}^{\infty} \exp(-\alpha_p) \exp(-\exp(-\alpha_p)) d\alpha_p = \pi_p,
 \end{aligned}$$

where the final equality follows easily from a change of variables.

Appendix C. Mixing of Parallel MCMC and QPMCMC

To ascertain whether QPMCMC mixes differently compared to conventional parallel MCMC, we run both algorithms for a range of proposal counts to sample from a 10-dimensional standard normal distribution. For each algorithm and proposal setting, we run 100 independent chains for 10,000 iterations and obtain effective sample sizes ESS_d for $d \in \{1, \dots, 10\}$. We then calculate the relative differences between the means and minima of one chain generated using parallel MCMC and one chain generated using QPMCMC; for example:

$$\begin{aligned}
 \text{Relative difference between means } \overline{ESS}_{(\cdot)} \\
 = \frac{|\overline{ESS}_{\text{pMCMC}} - \overline{ESS}_{\text{QPMCMC}}|}{\overline{ESS}_{\text{pMCMC}}}
 \end{aligned}$$

Figure B.1 shows results. In general, the majority relative differences are small. For both statistics, mean relative differences are less than 0.05, regardless of proposal count. Again for both statistics, more than 75% of the independent runs result in relative differences below 0.1. We note that relative differences between means (blue) appear to decrease with the number of proposals, but the same does not hold for relative differences between minima (red).

Appendix D. Note on Simulations

We use R (R Core Team 2021), PYTHON (van Rossum 1995), TENSORFLOW (Abadi et al. 2016) and TENSORFLOW PROBABILITY MCMC (Lao et al. 2020) in our simulations and the GGLOT2 package to generate all figures (Wickham 2016). In R, we use the package CODA (Plummer et al. 2006) to calculate effective sample sizes. In PYTHON, we use a built-in function from TENSORFLOW PROBABILITY MCMC. The primary color palette comes from Ram and Wickham (2018).

All simulations rely on the fact that the Grover iterations of (3) manipulate the probability amplitudes in a deterministic manner. For example, the following R code specifies N uniform probability amplitudes (`sqrtProbs`), performs I Grover iterations and measures a single state according to the resulting probabilities.

```

sqrtProbs <- rep(1/sqrt(N), N); i <- 1
while (i <= I) {
  sqrtProbs <- (1-2*marks)*sqrtProbs
  sqrtProbs <- -sqrtProbs + 2*mean
    (sqrtProbs)
  i <- i + 1
}
measurement <- sample(size=1,
  x=1:N, prob=sqrtProbs^2)
    
```

Of course, simulating these iterations on a classical computer requires precomputing the values of `marks` beforehand. All code is available online at <https://github.com/andrewjholbrook/qpMCMC>.

Disclosure Statement

There are no competing interests to declare.

Funding

This work was supported by grants NIH K25 AI153816, NSF DMS 2152774 and NSF DMS 2236854.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016), “{TensorFlow}: A System for {Large-Scale} Machine Learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283. [13]
- Akiyama, K., Alberdi, A., Alef, W., Algaba, J. C., Anantua, R., Asada, K., Azulay, R., Bach, U., Baczko, A.-K., Ball, D., et al. (2022), “First Sagittarius a* Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole in the Center of the Milky Way,” *The Astrophysical Journal Letters*, 930, L12. [9,10]
- Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. (1997), “Strengths and Weaknesses of Quantum Computing,” *SIAM Journal on Computing*, 26, 1510–1523. [2]
- Berzuini, C., and Gilks, W. (2001), “Resample-Move Filtering with Cross-Model Jumps,” in *Sequential Monte Carlo Methods in Practice*, eds. A. Doucet, N. Freitas, and N. Gordon, pp. 117–138, New York: Springer. [11]

- Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017), “Quantum Machine Learning,” *Nature*, 549, 195–202. [11]
- Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998), “Tight Bounds on Quantum Searching,” *Fortschritte der Physik: Progress of Physics*, 46, 493–505. [2,3,4,5]
- Calderhead, B. (2014), “A General Construction for Parallelizing Metropolis–Hastings Algorithms,” *Proceedings of the National Academy of Sciences*, 111, 17408–17413. [1]
- Cortese, J. A., and Braje, T. M. (2018), “Loading Classical Data into a Quantum Computer,” arXiv preprint arXiv:1803.01958. [11]
- Csilléry, K., Blum, M. G., Gaggiotti, O. E., and François, O. (2010), “Approximate Bayesian Computation (ABC) in Practice,” *Trends in Ecology & Evolution*, 25, 410–418. [11]
- Delmas, J.-F., and Jourdain, B. (2009), “Does Waste Recycling Really Improve the Multi-Proposal Metropolis–Hastings Algorithm? An Analysis based on Control Variates,” *Journal of Applied Probability*, 46, 938–959. [1]
- Doucet, A., De Freitas, N., and Gordon, N. J. (2001), *Sequential Monte Carlo Methods in Practice* (Vol. 1), New York: Springer. [11]
- Durr, C., and Hoyer, P. (1996), “A Quantum Algorithm for Finding the Minimum,” arXiv preprint quant-ph/9607014. [2,5,6,7]
- Erhard, A., Wallman, J. J., Postler, L., Meth, M., Stricker, R., Martinez, E. A., Schindler, P., Monz, T., Emerson, J., and Blatt, R. (2019), “Characterizing Large-Scale Quantum Computers via Cycle Benchmarking,” *Nature Communications*, 10, 1–7. [7]
- Frenkel, D. (2004), “Speed-up of Monte Carlo Simulations by Sampling of Rejected States,” *Proceedings of the National Academy of Sciences* 101, 17571–17575. [1]
- Gelman, A., and Rubin, D. B. (1992a), “Inference from Iterative Simulation Using Multiple Sequences,” *Statistical Science*, 7, 457–472. [2]
- (1992b), “A Single Series from the Gibbs Sampler Provides a False Sense of Security,” *Bayesian Statistics*, 4, 625–631. [2]
- Giovanetti, V., Lloyd, S., and Maccone, L. (2008), “Quantum Random Access Memory,” *Physical Review Letters*, 100, 160501. [7]
- Glatt-Holtz, N. E., Holbrook, A. J., Krometis, J. A., and Mondaini, C. F. (2022), “Parallel MCMC Algorithms: Theoretical Foundations, Algorithm Design, Case Studies,” arXiv preprint arXiv:2209.04750. [1,7,8,11]
- Grover, L. K. (1996), “A Fast Quantum Mechanical Algorithm for Database Search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219. [2,3]
- Haener, T., Soeken, M., Roetteler, M., and Svore, K. M. (2018), “Quantum Circuits for Floating-Point Arithmetic,” in *International Conference on Reversible Computation*, pp. 162–174, Springer. [11]
- Harrow, A. W., Hassidim, A., and Lloyd, S. (2009), “Quantum Algorithm for Linear Systems of Equations,” *Physical Review Letters*, 103, 150502. [2]
- Hastings, W. K. (1970), “Monte Carlo Sampling Methods Using Markov Chains and their Applications,” *Biometrika*, 57, 97–109. [1]
- Holbrook, A. J. (2021), “Generating MCMC Proposals by Randomly Rotating the Regular Simplex,” arXiv preprint arXiv:2110.06445. [1,7,8,10]
- Holbrook, A. J., Ji, X., and Suchard, M. A. (2022a), “Bayesian Mitigation of Spatial Coarsening for a Hawkes Model Applied to Gunfire, Wildfire and Viral Contagion,” *The Annals of Applied Statistics* 16, 573–595. [2]
- (2022b), “From Viral Evolution to Spatial Contagion: A Biologically Modulated Hawkes Model,” *Bioinformatics*, 38, 1846–1856. [2]
- Holbrook, A. J., Lemey, P., Baele, G., Dellicour, S., Brockmann, D., Rambaut, A., and Suchard, M. A. (2021a), “Massive Parallelization Boosts Big Bayesian Multidimensional Scaling,” *Journal of Computational and Graphical Statistics*, 30, 11–24. [2,6]
- Holbrook, A. J., Loeffler, C. E., Flaxman, S. R., and Suchard, M. A. (2021b), “Scalable Bayesian Inference for Self-Excitatory Stochastic Processes Applied to Big American Gunfire Data,” *Statistics and Computing*, 31, 1–15. [2]
- Hurn, M. (1997), “Difficulties in the Use of Auxiliary Variables in Markov Chain Monte Carlo Methods,” *Statistics and Computing*, 7, 35–44. [9,10]
- Jordan, S. P. (2005), “Fast Quantum Algorithm for Numerical Gradient Estimation,” *Physical Review Letters*, 95, 050501. [11]
- Kielpinski, D., Monroe, C., and Wineland, D. J. (2002), “Architecture for a Large-Scale Ion-Trap Quantum Computer,” *Nature*, 417, 709–711. [7]
- Lao, J., Suter, C., Langmore, I., Chimisov, C., Saxena, A., Sountsov, P., Moore, D., Saurous, R. A., Hoffman, M. D., and Dillon, J. V. (2020), “tfp.MCMC: Modern Markov Chain Monte Carlo Tools Built for Modern Hardware,” arXiv preprint arXiv:2002.01184. [2,13]
- Lemieux, J., Heim, B., Poulin, D., Svore, K., and Troyer, M. (2020), “Efficient Quantum Walk Circuits for Metropolis–Hastings Algorithm,” *Quantum*, 4, 287. [11]
- Lopatnikova, A., and Tran, M.-N. (2021), “An Introduction to Quantum Computing for Statisticians,” arXiv preprint arXiv:2112.06587. [2]
- Luo, X., and Tjelmeland, H. (2019), “A Multiple-Try Metropolis–Hastings Algorithm with Tailored Proposals,” *Computational Statistics*, 34, 1109–1133. [1]
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), “Equation of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, 21, 1087–1092. [1]
- Montanaro, A. (2015), “Quantum Speedup of Monte Carlo Methods,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471, 20150301. [11]
- Neal, R. (2011), “MCMC Using Hamiltonian Dynamics,” in *Handbook of Markov Chain Monte Carlo, Part 1*, eds. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, New York: Chapman & Hall/CRC. [2]
- Neal, R. M. (2003), “Markov Chain Sampling for Non-linear State Space Models using Embedded Hidden Markov Models,” arXiv preprint math/0305039. [1]
- Nielsen, M. A., and Chuang, I. (2010), *Quantum Computation and Quantum Information*, Cambridge University Press. [2]
- Papandreou, G., and Yuille, A. L. (2011), “Perturb-and-Map Random Fields: Using Discrete Optimization to Learn and Sample from Energy Models,” in *2011 International Conference on Computer Vision*, pp. 193–200, IEEE. [6]
- Plummer, M., Best, N., Cowles, K., and Vines, K. (2006), “Coda: Convergence Diagnosis and Output Analysis for MCMC,” *R News* 6, 7–11. [10,13]
- R Core Team. (2021), *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. [13]
- Ram, K., and Wickham, H. (2018), *wesanderson: A Wes Anderson Palette Generator*. R package version 0.3.6. [13]
- Roberts, G. O., and Rosenthal, J. S. (2001), “Optimal Scaling for Various Metropolis–Hastings Algorithms,” *Statistical Science*, 16, 351–367. [8]
- Rosenthal, J. S. (2011), “Optimal Proposal Distributions and Adaptive MCMC,” in *Handbook of Markov Chain Monte Carlo* 4, eds. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, New York: Chapman & Hall/CRC. [8]
- Salmon, J. K., Moraes, M. A., Dror, R. O., and Shaw, D. E. (2011), “Parallel Random Numbers: As Easy as 1, 2, 3,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. [1]
- Schwedes, T., and Calderhead, B. (2021), “Rao-blackwellised Parallel MCMC,” in *International Conference on Artificial Intelligence and Statistics*, pp. 3448–3456, PMLR. [1]
- Shor, P. (1994), “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134. [2]
- Szegedy, M. (2004), “Quantum Speed-Up of Markov Chain Based Algorithms,” in *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 32–41, IEEE. [11]
- Tjelmeland, H. (2004), “Using all Metropolis–Hastings Proposals to Estimate Mean Values,” Technical Report. Citeseer. [1,2,7]
- van Rossum, G. (1995), “Python Reference Manual,” Department of Computer Science [CS]. [13]
- Wang, Y., and Liu, H. (2022), “Quantum Computing in a Statistical Context,” *Annual Review of Statistics and Its Application*, 9, 479–504. [2]
- Werner, K., Jansson, M., and Stoica, P. (2008), “On Estimation of Covariance Matrices with Kronecker Product Structure,” *IEEE Transactions on Signal Processing*, 56, 478–491. [12]
- Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, New York: Springer-Verlag. [13]
- Wocjan, P., and Abeyesinghe, A. (2008), “Speedup via Quantum Sampling,” *Physical Review A*, 78, 042336. [11]
- Yang, S., Chen, Y., Bernton, E., and Liu, J. S. (2018), “On Parallelizable Markov Chain Monte Carlo Algorithms with Waste-Recycling,” *Statistics and Computing*, 28, 1073–1081. [1]
- Yoder, T. J., Low, G. H., and Chuang, I. L. (2014), “Fixed-Point Quantum Search with an Optimal Number of Queries,” *Physical Review Letters*, 113, 210501. [4,5]