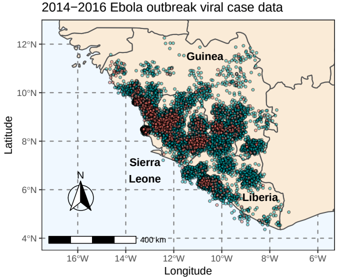
A quantum parallel Markov chain Monte Carlo

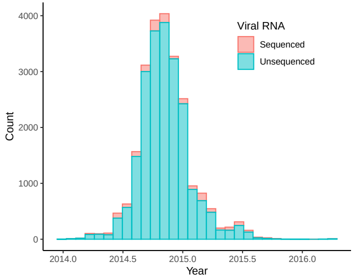
Andrew J. Holbrook

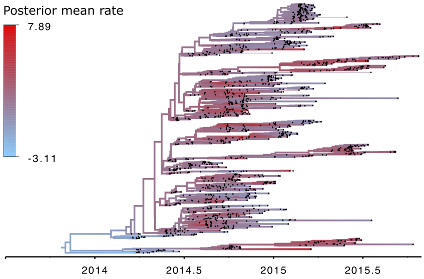
UCLA Biostatistics

March 11, 2022

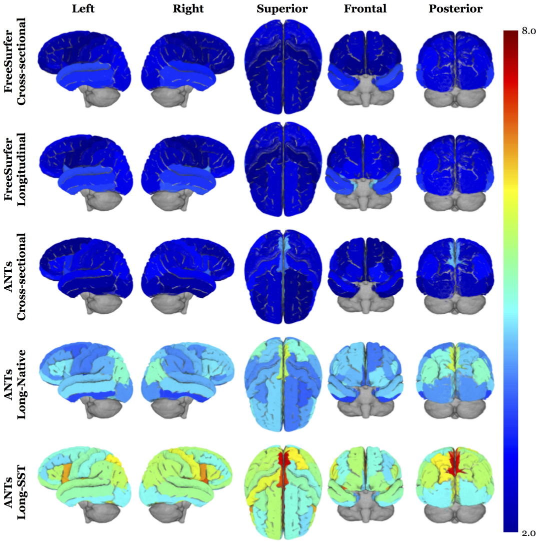
The data science perspective



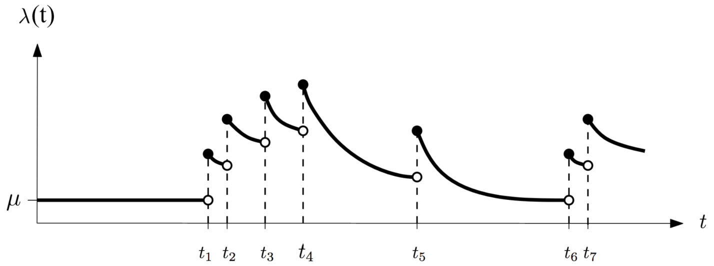




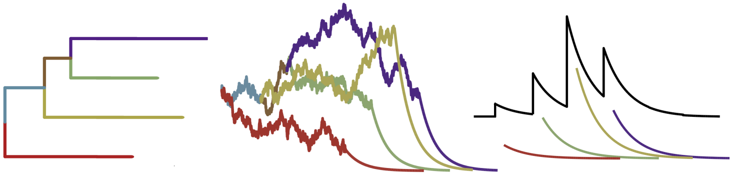
The data science perspective



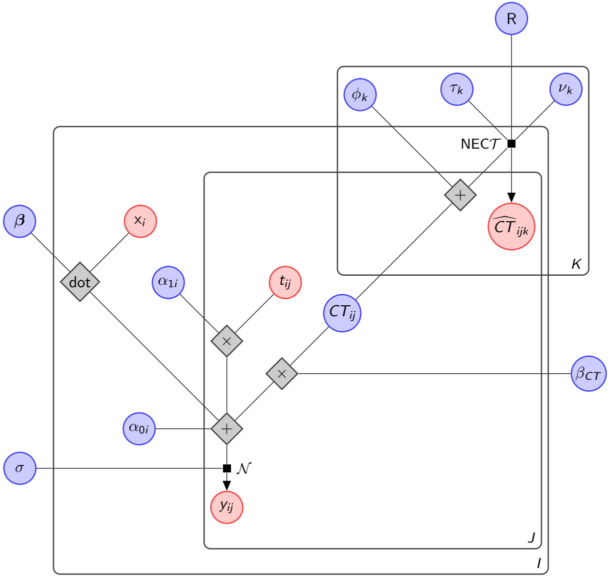
Hierarchical models for mixed-type data



Laub et al. 2015



Hierarchical models for mixed-type data



Bayesian data analysis

Assume data generated according to $\mathrm{y}_n \overset{\perp}{\sim} f(\mathrm{y}_n|\boldsymbol{\theta}, \mathrm{z}_n)$ with prior distributions $\boldsymbol{\theta} \sim p_\theta(\boldsymbol{\theta})$ and $(\mathrm{z}_1, \dots, \mathrm{z}_N) = \mathrm{Z} \sim p_z(\mathrm{Z})$.

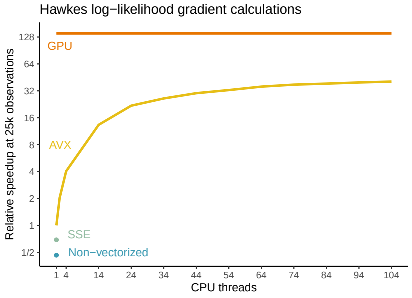
Bayes' theorem says:

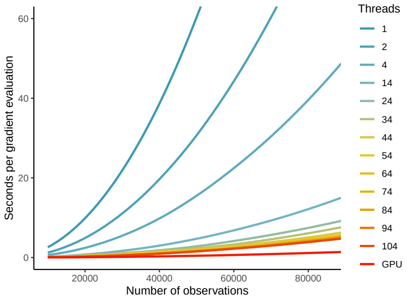
$$p(\boldsymbol{\theta}|\mathrm{Y}) = \frac{f(\mathrm{Y}|\boldsymbol{\theta})\, p_\theta(\boldsymbol{\theta})}{f(\mathrm{Y})} = \frac{\int_{\mathrm{Z}} f(\mathrm{Y}|\mathrm{Z}, \boldsymbol{\theta}) p_z(\mathrm{Z}) \mathrm{dZ}\ p_\theta(\boldsymbol{\theta})}{\int_\Theta \left(\int_{\mathrm{Z}} f(\mathrm{Y}|\mathrm{Z}, \boldsymbol{\theta}) p_x(\mathrm{Z}) \mathrm{dZ}\right) p_\theta(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}},$$

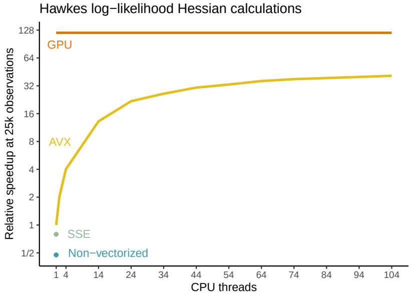
where $f(\mathrm{Y}|\boldsymbol{\theta}, \mathrm{Z}) = \prod_n^N f(\mathrm{y}_n|\boldsymbol{\theta}, \mathrm{z}_n)$ is the *likelihood* function and $f(\mathrm{Y}|\boldsymbol{\theta})$ is the *marginal likelihood*.

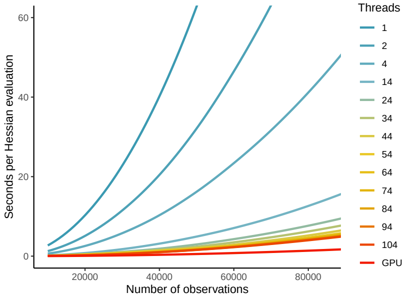
Instead of integrating, we use MCMC to generate samples from $p(\mathrm{Z}, \boldsymbol{\theta}|\mathrm{Y})$, but this is computationally demanding.

Parallelizing within-chain computations



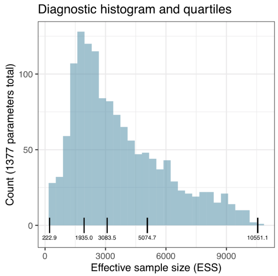


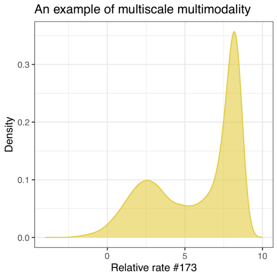


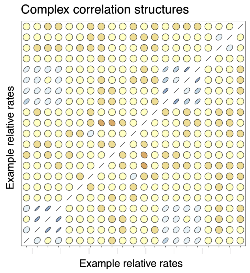


Parallelizing within-chain computations

I use Hamiltonian (hybrid) Monte Carlo with adaptive precoditioning to generate 100 million Markov chain states ($\sim$ 3.5 million samples/day on Nvidia GV100) in 1 month.

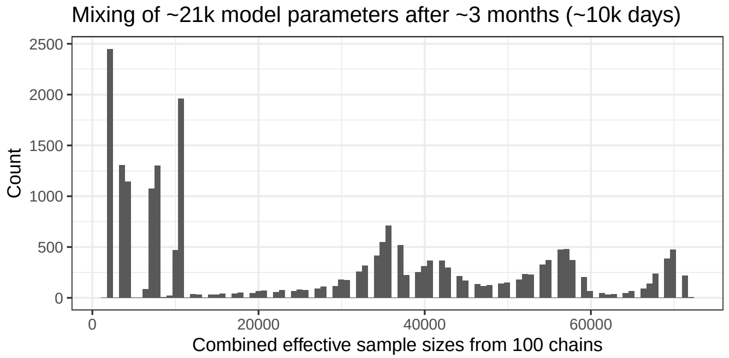






Parallelizing between-chain computations

Cortical thickness: 100 parallel chains using No-U-Turn with adaptive mass matrix.



Two questions

Neither within- nor between-chain parallelization is a “silver bullet.” Here, I am interested in two questions.

- Can we adjust the deeper *algorithmic* structure of MCMC to increase its parallelism?
- How can we leverage parallelism using emerging computational technologies?

Markov chain Monte Carlo

Consider a probability distribution $\pi(\mathrm{d}\boldsymbol{\theta})$ defined on $\mathbb{R}^D$ that admits a probability density $\pi(\boldsymbol{\theta})$ with respect to the Lebesgue measure, i.e., $\pi(\mathrm{d}\boldsymbol{\theta}) =: \pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$.

To generate samples from the target distribution π, we craft a kernel $P(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta})$ that satisfies

$$\pi(A) = \int \pi(\mathrm{d}\boldsymbol{\theta}_0) P(\boldsymbol{\theta}_0, A) \quad a.s.$$

for $A \subset \mathbb{R}^D$ for which $\pi(A) > 0$.

Markov chain Monte Carlo

The Metropolis-Hastings algorithm builds such a transition kernel $P(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta})$ by:

1. generating a proposal $\boldsymbol{\theta}$ according to the distribution $Q(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta}) =: q(\boldsymbol{\theta}_0, \boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$; and

2. accepting this proposal with probability

$$\pi = 1 \wedge \frac{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta}, \boldsymbol{\theta}_0)}{\pi(\boldsymbol{\theta}_0)q(\boldsymbol{\theta}_0, \boldsymbol{\theta})} \, .$$

This kernel maintains *detailed balance* and therefore leaves $\pi(\mathrm{d}\boldsymbol{\theta})$ invariant.

Parallel MCMC

The parallel MCMC algorithm (Tjelmeland, 2004) builds such a transition kernel $P(\boldsymbol{\theta}_0, \mathrm{d}\boldsymbol{\theta})$ by:

1. generating P proposals $\Theta_{-0} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_P)$ from a joint distribution $Q(\boldsymbol{\theta}_0, \mathrm{d}\Theta_{-0}) =: q(\boldsymbol{\theta}_0, \Theta_{-0})\mathrm{d}\Theta_{-0}$; and

2. selecting the next state with probabilities

$$\pi_p = \frac{\pi(\boldsymbol{\theta}_p)q(\boldsymbol{\theta}_p, \Theta_{-p})}{\sum_{p'=0}^{P} \pi(\boldsymbol{\theta}_{p'})q(\boldsymbol{\theta}_{p'}, \Theta_{-p'})}, \quad p = 0, \ldots, P\,.$$

This kernel *also* maintains detailed balance and therefore leaves $\pi(\mathrm{d}\boldsymbol{\theta})$ invariant.

Parallel MCMC

We can simplify these acceptance probabilities to remove a potentially $O(P^2)$ computational burden.

For example, the proposal mechanism

$$\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_P \overset{\perp}{\sim} \mathit{Normal}_D(\bar{\boldsymbol{\theta}}, \Sigma)\,, \quad \bar{\boldsymbol{\theta}} \sim \mathit{Normal}_D(\boldsymbol{\theta}_0, \Sigma)$$

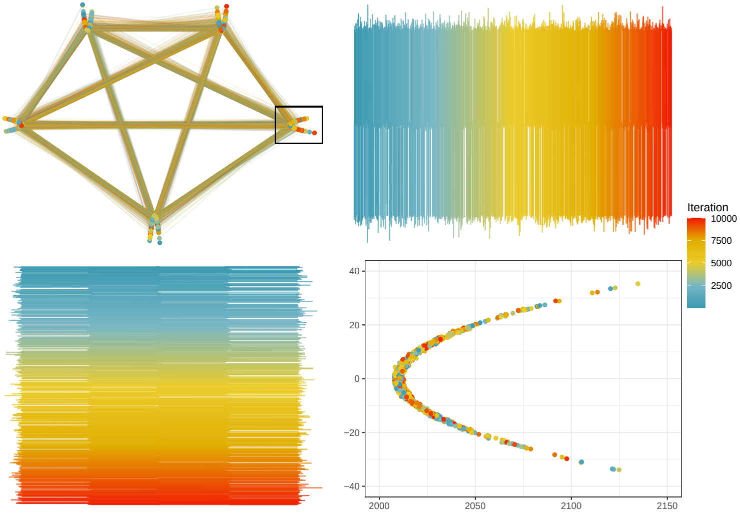
results in proposal densities that satisfy

$$q(\boldsymbol{\theta}_0, \Theta_{-0}) = q(\boldsymbol{\theta}_1, \Theta_{-1}) = \cdots = q(\boldsymbol{\theta}_P, \Theta_{-P})\,.$$

Thus, these simplified probabilities preserve detailed balance:

$$\pi_p = \frac{\pi(\boldsymbol{\theta}_p)}{\sum_{p'=0}^{P} \pi(\boldsymbol{\theta}_{p'})}\,, \quad p = 0, \dots, P\,.$$

Parallel MCMC



Parallel MCMC

Pro: fast and flexible on an iteration-by-iteration basis.

Con: each iteration requires $O(P)$ evaluations of the target $\pi(\cdot)$ in

$$\pi_p = \frac{\pi(\boldsymbol{\theta}_p)}{\sum_{p'=0}^{P} \pi(\boldsymbol{\theta}_{p'})}\,, \quad p = 0, \ldots, P\,.$$

But these evaluations do not depend on each other, so...
let's parallelize using a quantum computer!

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

Not so fast!

A quantum circuit could plausibly evaluate each $\pi(\boldsymbol{\theta}_p)$, for $p = 0, 1, \dots, P$ at the same time using quantum parallelism.

But having done this, the best we could hope for is to obtain a single $\pi(\boldsymbol{\theta}_p)$ upon measurement.

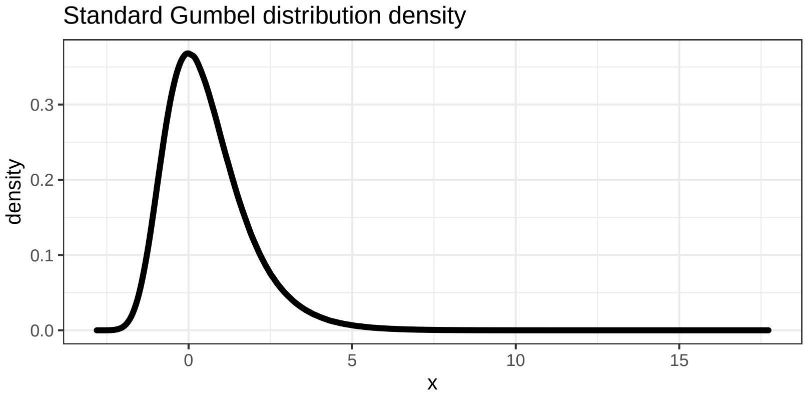
Hmmm, maybe we can also use the quantum circuit to sample

$$\hat{p} \sim \textit{Discrete}\left(\frac{\pi(\boldsymbol{\theta}_0)}{\sum_p \pi(\boldsymbol{\theta}_p)}, \frac{\pi(\boldsymbol{\theta}_1)}{\sum_p \pi(\boldsymbol{\theta}_p)}, \dots, \frac{\pi(\boldsymbol{\theta}_P)}{\sum_p \pi(\boldsymbol{\theta}_p)}\right)$$

Maybe? But doesn't the normalization step violate quantum parallelism?

Detour: the Gumbel-max trick

The Gumbel distribution



If $z \sim \mathit{Gumbel}(0,1)$, then it has density and distribution functions

$$g(z) = \exp\big(-z - \exp(-z)\big) \quad \text{and} \quad G(z) = \exp\big(-\exp(-z)\big)\,.$$

Gumbel-max trick

We wish to sample from the discrete distribution $\hat{p} \sim \mathit{Discrete}(\boldsymbol{\pi})$ for $\hat{p} \in \{0, 1, \ldots, P\}$ and we only know $\boldsymbol{\pi}^* = c\boldsymbol{\pi}$ for some $c > 0$.

Define $\boldsymbol{\lambda}^* = \log \boldsymbol{\pi}^* = \log \boldsymbol{\pi} + \log c$ and suppose $z_0, z_1, \ldots, z_P \overset{\perp}{\sim} \mathit{Gumbel}(0, 1)$.

Finally, define $\alpha_p^* := \lambda_p^* + z_p$ and $\hat{p} = \arg\max_{p=0,\ldots,P} \alpha_p^*$.

Then the following holds (Papandreou and Yuille, 2011):

$$\Pr(\hat{p} = p) = \pi_p\,, \quad p = 0, 1, \ldots, P.$$

```
Data: A vector of unnormalized log-probabilities
      $\boldsymbol{\lambda}^* = \log \boldsymbol{\pi} + \log c$, for $\boldsymbol{\pi}$ a discrete probability vector
      with $P + 1$ elements.
Result: A single sample $\hat{p} \sim \mathit{Discrete}(\boldsymbol{\pi})$ satisfying
        $\hat{p} \in \{0, 1, \ldots, P\}$.
for $p \in \{0, 1, \ldots, P\}$ do
    $z_p \leftarrow \mathit{Gumbel}(0, 1)$;
    $\alpha^*_p \leftarrow \lambda^*_p + z_p$;
end
$\hat{p} \leftarrow \arg\max_{p=0,\ldots,P} \alpha^*_p$;
return $\hat{p}$ .
```

Algorithm 1: The Gumbel-max trick

End of detour.

```
**Data:** Initial Markov chain state $\boldsymbol{\theta}^{(0)}$; total length of Markov
       chain $S$; total number of proposals per iteration $P$.
**Result:** A Markov chain $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}$.
for $s \in \{1, \dots, S\}$ do
    $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}^{(s-1)}$;
    $\bar{\boldsymbol{\theta}} \leftarrow \mathit{Normal}_D(\boldsymbol{\theta}_0, \Sigma)$;
    $z_0 \leftarrow \mathit{Gumbel}(0, 1)$;
    for $p \in \{1, \dots, P\}$ do
        $\boldsymbol{\theta}_p \leftarrow \mathit{Normal}_D(\bar{\boldsymbol{\theta}}, \Sigma)$;
        $z_p \leftarrow \mathit{Gumbel}(0, 1)$;
    end
    $\hat{p} \leftarrow \arg\min_{p=0,\dots,P} \Big( f(p) := -\big(z_p + \log \pi(\boldsymbol{\theta}_p)\big) \Big)$;
    $\boldsymbol{\theta}^{(s)} \leftarrow \boldsymbol{\theta}_{\hat{p}}$;
end
return $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}$ .
```

Algorithm 2: An equivalent parallel MCMC

The main idea of QPMCMC

Use a quantum circuit to obtain

$$\hat{p} = \underset{p=0,\ldots,P}{\arg\min} \Big(f(p) := -\big(z_p + \log \pi(\boldsymbol{\theta}_p)\big) \Big)$$

in time $O(\sqrt{P})$.

We use a Matryoshka doll of established quantum algorithms:

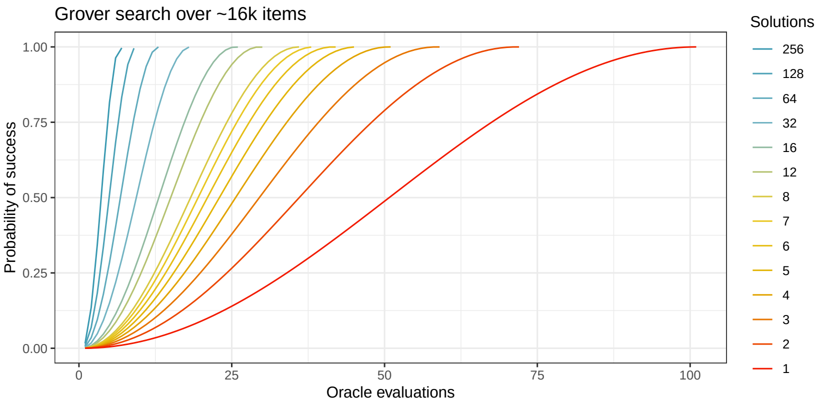
- Grover's search algorithm (Grover, 1996) embeds inside ...
- ... the exponential searching algorithm (Boyer et al., 1998) embeds inside ...
- the Quantum minimization algorithm (Dürr and Høyer, 1996).

Grover's search

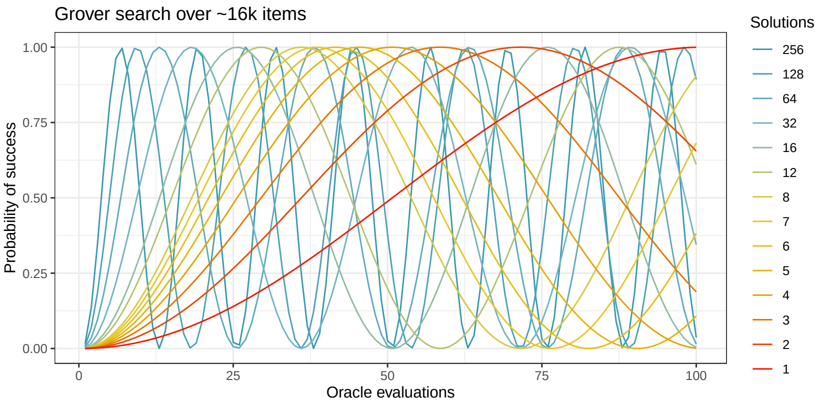
```
Data: An oracle gate $\mathrm{U}_f$ taking $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ for a
      function $f(x) : \{0,\ldots,N-1\} \to \{0,1\}$ that satisfies
      $f(x_0) = 1$ for a single $x_0$; $n+1 = \log_2(N)+1$ quantum
      states initialized to $|0\rangle^{\otimes n}|1\rangle$; an integer $R = \lceil \pi\sqrt{N}/4 \rceil$.
Result: An $n$-bit binary string $x_0$ satisfying $f(x_0) = 1$ with
        error less than $1/N$.
$|0\rangle^{\otimes n}|1\rangle \longrightarrow \mathrm{H}^{\otimes n+1}|0\rangle^{\otimes n}|1\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|-\rangle$;
$|h\rangle|-\rangle \longrightarrow \left((2|h\rangle\langle h| - \mathrm{I})(\mathrm{I} - 2|x_0\rangle\langle x_0|)\right)^R|h\rangle|-\rangle \approx |x_0\rangle|-\rangle$;
$|x_0\rangle \longrightarrow x_0$;
return $x_0$ .
```

Algorithm 3: Quantum search algorithm (Grover, 1996)

Grover's search



Grover's search

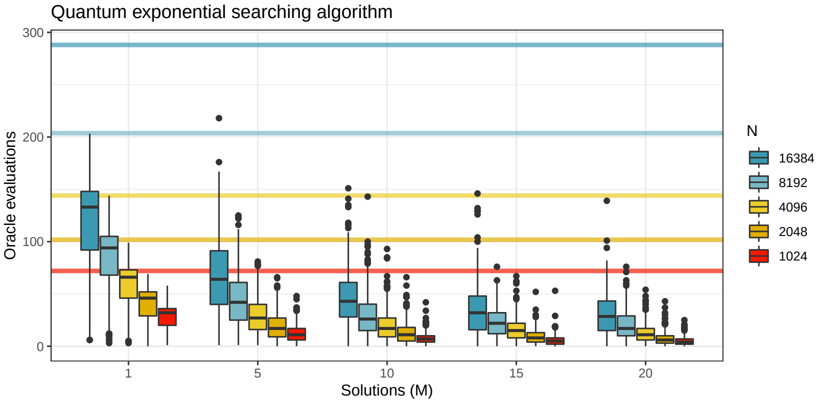


What do we do when we don't know M?

```
**Data:** An oracle gate $U_f$ taking $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ for a function
        $f(x) : \{0, \ldots, N-1\} \to \{0,1\}$ 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.
$m \leftarrow 1$;
$\gamma \leftarrow 6/5$;
success $\leftarrow$ FALSE;
**while** *success* $\neq$ *TRUE* **do**
    $j \leftarrow Uniform\{0, \cdots, m-1\}$;
    $|0\rangle^{\otimes n}|1\rangle \longrightarrow H^{\otimes n+1}|0\rangle^{\otimes n}|1\rangle = |h\rangle|-\rangle$;
    $|h\rangle|-\rangle \longrightarrow G^j|h\rangle|-\rangle = |x\rangle|-\rangle$;        /* $j$ Grover iterations.  */
    $|x\rangle \longrightarrow x$;                                     /* Measure and check.  */
    **if** $f(x) = 1$ **then**
        $x_0 \leftarrow x$;
        success $\leftarrow$ TRUE;
    **else**
        $m \leftarrow \min\left(\gamma m, \sqrt{N}\right)$;              /* If fail, increase $m$.  */
    **end**
**end**
**return** $x_0$ .
```

Algorithm 4: Exponential searching algorithm (Boyer et al., 1998)

Exponential searching algorithm



When $M << N$, the expected total number of Grover iterations is bounded above by $\frac{9}{4}\sqrt{\frac{N}{M}}$. Horizontal lines at $\frac{9}{4}\sqrt{N}$.

```
Data: A quantum sub-routine capable of evaluating a function $f(\cdot)$ over
      $\{0, \ldots, N-1\}$ 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 $1 - \epsilon$.
$s \leftarrow 0$;
$x_0 \leftarrow Uniform\{0, \ldots, N-1\}$;
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; /* $l$ time steps */
    $s \leftarrow s + l$;
    Obtain $x'$ by measuring first register;
    if $f(x') < f(x_0)$ then
        $x_0 \leftarrow x'$
    end
end
return $x_0$ .
```

Algorithm 5: Quantum minimization algorithm (Dürr and Høyer, 1996)

Quantum minimization and warm-starting

Proposition

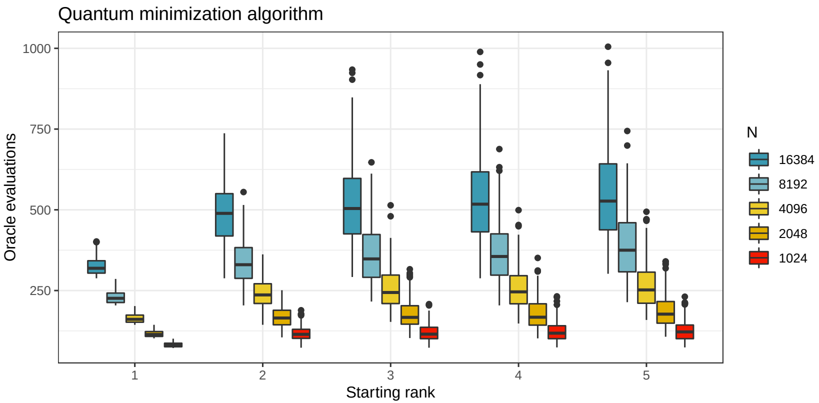
Suppose that the quantum minimization algorithm begins with a threshold F_0 such that $f(x) < F_0$ for only $K-1$ items. Then the expected total number of time steps to find the minimizer is bounded above by

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

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

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

Quantum minimization and warm-starting

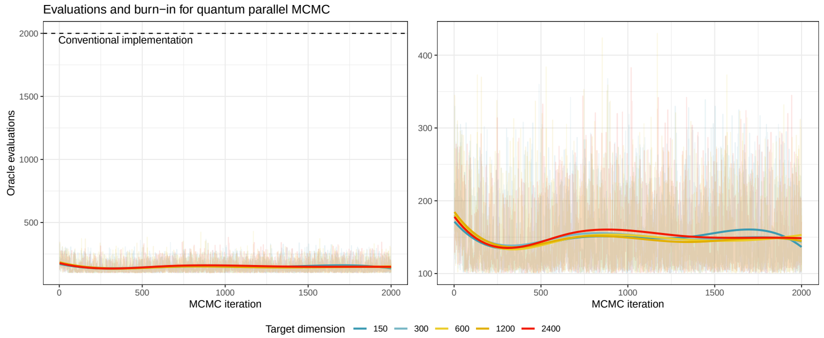


Using the same early stopping threshold within the exponential search sub-routine, we observe a failure rate less than 1%.

```
**Data:** Initial Markov chain state $\boldsymbol{\theta}^{(0)}$; total length of Markov
        chain $S$; total number of proposals per iteration $P$.
**Result:** A Markov chain $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}$.
for $s \in \{1, \dots, S\}$ do
    $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}^{(s-1)}$;
    $\bar{\boldsymbol{\theta}} \leftarrow \mathit{Normal}_D(\boldsymbol{\theta}_0, \Sigma)$;
    $z_0 \leftarrow \mathit{Gumbel}(0, 1)$;
    for $p \in \{1, \dots, P\}$ do
        $\boldsymbol{\theta}_p \leftarrow \mathit{Normal}_D(\bar{\boldsymbol{\theta}}, \Sigma)$;
        $z_p \leftarrow \mathit{Gumbel}(0, 1)$;
    end
    $\hat{p} \leftarrow \arg\min_{p=0,\dots,P} \Big( f(p) := -\big(z_p + \log \pi(\boldsymbol{\theta}_p)\big) \Big)$;
        /*    Quantum minimization starting at 0.    */
    $\boldsymbol{\theta}^{(s)} \leftarrow \boldsymbol{\theta}_{\hat{p}}$;
end
return $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}$ .
```

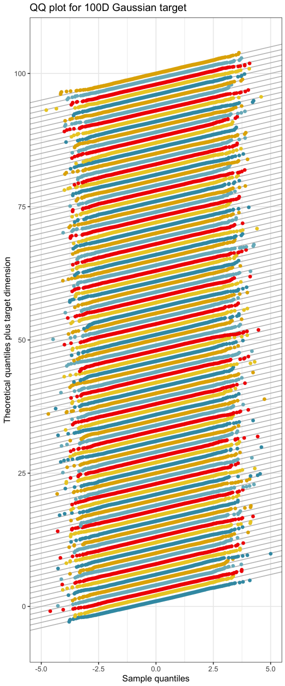
Algorithm 6: Quantum parallel MCMC

QPMCMC

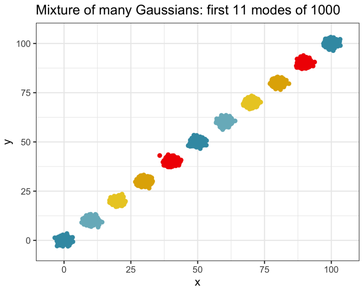


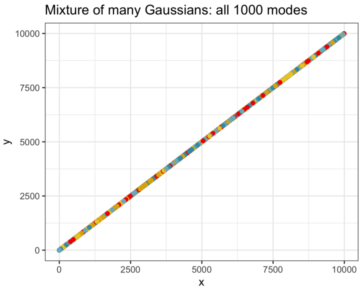
We use 7% of the conventional 2,000 target evaluations. Only 1 in 200 quantum minimizations fail.

QPMCMC


QQ plot for 100D Gaussian target
Theoretical quantiles plus target dimension
Sample quantiles
100
75
50
25
0
-5.0
-2.5
0.0
2.5
5.0

QPMCMC





Proposals	MCMC iterations	Target evaluations	Speedup	Efficiency gain
1,000	249,398 (200,998, 311,998)	24,988,963 (20,149,132, 31,265,011)	9.98 (9.98, 9.98)	1
5,000	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	5,998 (4,998, 6,998)	1,993,484 (1,662,592, 2,330,842)	30 (29.96, 30.26)	12.87 (8.64, 18.80)

Final thoughts

- Forget MCMC: just use quantum minimization and Gumbel-max to sample directly from the, e.g., Ising model.
- Don't forget MCMC: symmetric proposals can be made using discrete metrics for the, e.g., Ising model.
- Quantum minimization is "old" technology. Can we do better?

Thank you!