Markov Chains and Algorithmic Applications
Mini-Project: Solving the N-Queens Problem via the Metropolis Algorithm

Adway Girish, Megh Shukla
Team name: MCAA (Meghkov Chains and Adwayrithmic Applications)

December 19, 2022

Problem description

Consider an $N \times N$ chessboard. The problem is to place N queens on the chessboard such that no two queens may attack each other in that configuration (recall that a chess queen can move horizontally, vertically and diagonally). The solution to this problem, in general, is not unique. The number of solutions is known exactly only for $N \leq 27$, but very recently, the asymptotic growth rate has been shown to be approximately $(0.143N)^N$ [1, 2]. This mini-project consists of two parts:

1. to find a solution to the N-queens problem using the Metropolis algorithm, and
2. to estimate the number of solutions for large N.

1 Finding a solution

The Metropolis algorithm is a method to sample from a difficult distribution by constructing a Markov Chain that has this distribution as its stationary distribution. It can also be used to solve optimization problems by appropriately choosing the target distribution, e.g., to minimize a function f on S, defining the target distribution to be

$$\pi_\beta(x) = \frac{\exp[-\beta f(x)]}{Z_\beta} \text{ for } x \in S, \tag{1}$$

where $Z_\beta = \sum_{y \in S} \exp[-\beta f(y)]$, we have that as $\beta \to \infty$, π_β approaches a uniform distribution on the minima of f on S. Sampling from this distribution then gives us the solution to the minimization problem.

1.1 Theory

To solve the N queens problem, we formulate it as the solution to a minimization problem using the Metropolis algorithm as follows.

1. **State space.** A trivial attempt would be to let the state space be S_0, the set of all possible configurations of N queens on an $N \times N$ chessboard; note that $|S_0| = \binom{N^2}{N}$, which is of the order of $(eN)^N$ by Stirling's approximation. Observing that most of these states are completely uninteresting, we can reduce this to a more manageable size – in particular, all configurations where there is more than one queen in any row or column can be immediately rejected. We may thus reduce our state space to be simply the set of configurations of N queens on an $N \times N$ chessboard *with no row or column having more than one queen*, which is isomorphic to the set of permutations of $[N] \triangleq \{1, \ldots, N\}$. Formally, we have that the state space is given by
$$S = \{\sigma = (\sigma_1, \ldots, \sigma_N),\ \sigma \text{ is a permutation of } [N]\}. \tag{2}$$
Note that $|S| = N!$, which is of the order of $\left(\frac{N}{e}\right)^N \ll |S_0|$. A state $\sigma \in S$ can be interpreted as the configuration where for each $i \in [N]$, the cell indexed by the row-column pair (i, σ_i) is occupied.

2. **Base chain.** Pick two numbers uniformly at random from $[N]$ without replacement, and swap the columns of the queens in the corresponding rows. Mathematically, suppose the numbers picked are p and q satisfying $1 \le p < q \le N$, then starting from $\sigma = (\sigma_1, \ldots, \sigma_{p-1}, \sigma_p, \sigma_{p+1}, \ldots, \sigma_{q-1}, \sigma_q, \sigma_{q+1} \ldots, \sigma_N) \in S$, we end up with $\mu = (\sigma_1, \ldots, \sigma_{p-1}, \sigma_q, \sigma_{p+1}, \ldots, \sigma_{q-1}, \sigma_p, \sigma_{q+1} \ldots, \sigma_N)$, which is still in S, as it is still a permutation of $[N]$. This corresponds to a chain with transition probabilities

$$\psi_{\sigma\mu} = \begin{cases} \frac{1}{\binom{N}{2}} & \text{if } \sigma \sim \mu \triangleq \sigma \text{ and } \mu \text{ differ at exactly two positions,} \\ 0 & \text{else.} \end{cases} \tag{3}$$

That this is an irreducible chain is clear, as any target permutation can be obtained from any initial permutation, by simply going in order from the first element to the last and swapping to set it to the desired value. Since the state space is finite, this also implies that the base chain is positive-recurrent.

3. **Energy function.** Define the function $f : S \to \mathbb{Z}^+$, the set of nonnegative integers, by

$$\begin{aligned} f(\sigma) &= \text{the number of pairs of queens in conflict in configuration } \sigma \in S \\ &= \sum_{1 \le i < j \le N} \mathbb{1}\{\text{the queens at } (i, \sigma_i) \text{ and } (j, \sigma_j) \text{ are in conflict}\}. \end{aligned}$$

The only conflicts possible are if the queens happen to be along the same diagonal line (since our choice of state space eliminates the possibility of them being on the same horizontal or vertical line). This happens if the line connecting the cells has "slope" 1 or -1, i.e., if $\sigma_j - \sigma_i = i - j$ or $j - i$, or equivalently, $|\sigma_j - \sigma_i| = |j - i|$. Hence we have a concise formulation of the energy function,

$$f(\sigma) = \sum_{1 \le i < j \le N} \mathbb{1}\{|\sigma_j - \sigma_i| = |j - i|\}. \tag{4}$$

4. **Acceptance probabilities.** Our target distribution is given by π_β, computed using the energy function f defined above. Since the chosen base chain is symmetric, we have that the acceptance probabilities are given by

$$\begin{aligned} a_{\sigma\mu} &= \min\left(1, \frac{\pi_\beta(\mu)\psi_{\mu\sigma}}{\pi_\beta(\sigma)\psi_{\sigma\mu}}\right) = \min\left(1, \frac{\pi_\beta(\mu)}{\pi_\beta(\sigma)}\right) = \min\left(1, \exp[-\beta(f(\mu) - f(\sigma))]\right) \\ &= \begin{cases} 1 & \text{if } f(\mu) \le f(\sigma), \\ \exp[-\beta(f(\mu) - f(\sigma))] & \text{else.} \end{cases} \end{aligned} \tag{5}$$

5. **Metropolis chain.** Using the quantities defined so far, we can finally compute the transition probabilities for our Metropolis chain, which are given by

$$p_{\sigma\mu} = \begin{cases} a_{\sigma\mu}\psi_{\sigma\mu} & \text{if } \sigma \ne \mu, \\ 1 - \sum_{\lambda \in S, \lambda \ne \sigma} a_{\sigma\lambda}\psi_{\sigma\lambda} & \text{else} \end{cases} = \begin{cases} \frac{1}{\binom{N}{2}} a_{\sigma\mu} & \text{if } \sigma \sim \mu, \\ 1 - \frac{1}{\binom{N}{2}} \sum_{\lambda \sim \sigma} a_{\sigma\lambda} & \text{if } \sigma = \mu, \\ 0 & \text{else,} \end{cases} \tag{6}$$

with $a_{\sigma\mu}$ as given above. Observe that this chain is irreducible, positive-recurrent (since the base chain is as well), and periodic (due the presence of self-loops, as $a_{\sigma\lambda}$ must be lesser than 1 for some pair (σ, λ)), which together imply that the chain is ergodic. Hence a stationary distribution exists, is given by π_β, and is also the limiting distribution.

6. **Metropolis algorithm.** The above formulation can be summarized as given in Algorithm 1 below. Observe that the algorithm stops as soon as we have found a solution, since that is the aim. If our goal was, say, to sample from π_β instead, we could modify it to continue running even after $f(\sigma)$ becomes zero, till it reaches some sort of steady state (which can be defined appropriately), then the samples that it produces will be distributed as π_β. Call this the modified Metropolis algorithm, **Algorithm 1′** (which will be of use in the next part).

Algorithm 1: To find a solution to the N-queens problem

Input: N, state space $S =$ permutations of $[N]$, energy function f, β
Output: $\sigma =$ a solution

Pick any permutation of $[N]$ uniformly at random and call it σ
while $f(\sigma) > 0$ **do**
  Pick a pair of numbers p, q uniformly at random from $[N]$ without replacement
  Construct μ with $\mu_i \leftarrow \sigma_i$ for $i \neq p, q$, $\mu_p \leftarrow \sigma_q$, and $\mu_q \leftarrow \sigma_p$
  Compute $a_{\sigma\mu} \leftarrow \begin{cases} 1 & \text{if } f(\mu) \leq f(\sigma), \\ \exp[-\beta(f(\mu) - f(\sigma))] & \text{else} \end{cases}$
  With probability $a_{\sigma\mu}$, set $\sigma \leftarrow \mu$
Output σ. `// is guaranteed to have` $f(\sigma) = 0$

1.2 Experiments

All our simulations were performed using MATLAB. Here are some additional practical considerations that may not be obvious from an initial reading of Algorithm 1.

1. **Efficiently updating the energy function?** Computing $f(\sigma)$ as given in Equation (4) takes around N^2 terms, which is not ideal since we will need to compute the energy of states at every iteration. From Algorithm 1, we only really require the value of $f(\mu) - f(\sigma)$, where σ and μ differ at exactly two positions p and q (with $1 \leq p < q \leq N$ w.l.o.g.). This gives us σ and μ as given in the paragraph on the base chain at §1.1.2. Defining $\alpha(i,j) = \mathbb{1}\{|\mu_j - \mu_i| = |j-i|\} - \mathbb{1}\{|\sigma_j - \sigma_i| = |j-i|\}$ for notational simplicity, observe that $\alpha(i,j) = 0$ if (1) neither i nor j is p or q, (2) $i = j$, or (3) $(i,j) = (p,q)$. Thus we have that $f(\mu) - f(\sigma)$ can be simplified as

$$\sum_{1 \leq i < j \leq N} \mathbb{1}\{|\mu_j - \mu_i| = |j-i|\} - \sum_{1 \leq i < j \leq N} \mathbb{1}\{|\sigma_j - \sigma_i| = |j-i|\} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \alpha(i,j)$$

$$= \sum_{\substack{i=1, \\ i \neq p,q}}^{N-1} \left[\sum_{j=i+1}^{N} \alpha(i,j) \right] + \sum_{j=p+1}^{N} \alpha(p,j) + \sum_{j=q+1}^{N} \alpha(q,j)$$

$$= \sum_{\substack{i=1, \\ i \neq p,q}}^{N-1} \left[\left(\cancelto{0}{\sum_{\substack{j=i+1, \\ j \neq p,q}}^{N} \alpha(i,j)} \right) + \alpha(i,p)\mathbb{1}\{i < p\} + \alpha(i,q)\mathbb{1}\{i < q\} \right] + \sum_{j=p+1}^{N} \alpha(p,j) + \sum_{j=q+1}^{N} \alpha(q,j)$$

$$= \sum_{i=1}^{p-1} \alpha(i,p) + \sum_{\substack{i=1, \\ i \neq p}}^{q-1} \alpha(i,q) + \sum_{j=p+1}^{N} \alpha(p,j) + \sum_{j=q+1}^{N} \alpha(q,j)$$

$$= \sum_{i=1}^{p-1} \alpha(i,p) + \sum_{\substack{i=1, \\ i \neq p}}^{q-1} \alpha(i,q) + \sum_{i=p+1}^{N} \alpha(i,p) + \sum_{i=q+1}^{N} \alpha(i,q) \quad [\because \alpha(i,j) = \alpha(j,i);\ \text{relabelling summation index}]$$

$$= \sum_{i=1}^{N} \alpha(i,p) + \sum_{i=1}^{N} \alpha(i,q) = \sum_{i=1}^{N} [\alpha(i,p) + \alpha(i,q)], \qquad [\because \alpha(i,i) = 0, \alpha(p,q) = 0] \qquad (7)$$

   which is of a linear order in N. A vectorized implementation on MATLAB also makes it extremely efficient.

2. **Cooling schedule?** The parameter β may be thought of as an "inverse temperature". A small value of β (or a high temperature) gives us a distribution π_β that is close to the uniform distribution on all $N!$

states, while a large β (or a low temperature) gives us exactly the uniform distribution on the solutions. It may seem, then, that the value of β should be chosen to be as large as possible, but the problem then is that it becomes nearly impossible to escape from local minima, since the "effort" needed to overcome the barrier (which is exponential in β) is too large, so the algorithm could take longer to reach the global minima. A possible solution is *simulated annealing*, where we start from a small β (high temperature) and increase it gradually to a large value ("cool" it), to ensure better exploration at the initial stages. Some standard "cooling schedules" are linear, quadratic, exponential and logarithmic in the number of iterations, and we test all these possibilities against a constant value of β in Figure 1. The conclusion is that simulated annealing offers little, if any, improvement.

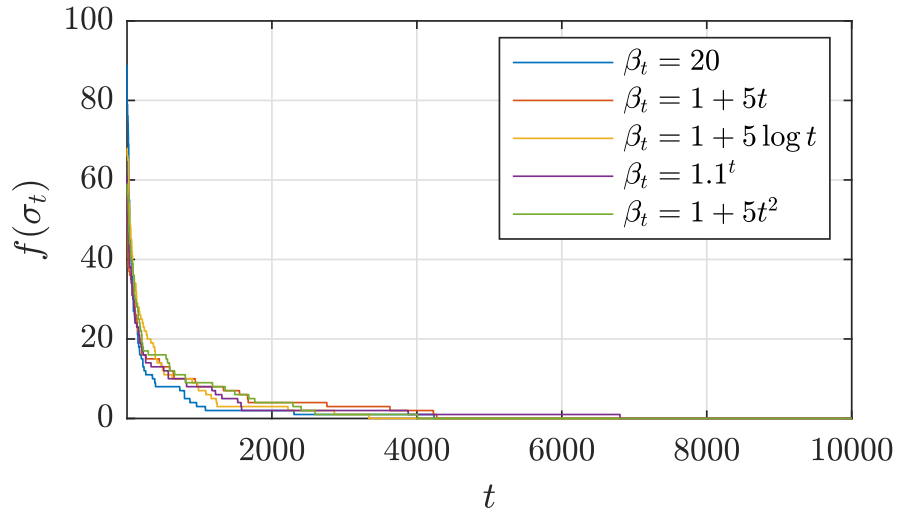


Figure 1: "Is it worth using simulated annealing here?": Seems not, as even a constant β performs at par with various cooling schedules for simulated annealing, i.e., it seems to reach a solution (and even steady state) in a similar amount of time. The plots are meant to be illustrative, even when repeated with different constants than 5, the results were of a similar nature. Results shown for $N = 100$.

3. **Value of β?** Being reasonably convinced that a constant β will suffice, we now look to find a good one. With a β that is too small, one would suspect that π_β is too close to the uniform distribution, and thus a solution may never be attained. The plots in Figure 2 confirm this, and $\beta = 20$ works for $N = 100$.

And finally, in Figure 4 are the plots showing that the code does indeed work for $N = 100$ and $N = 1000$, both obtained by taking $\beta = 20$. Since the actual solutions obtained for large N are incomprehensible visually, example solutions are shown in Figure 3 for small N.

2 Estimating the number of solutions

When $\beta \to \infty$, the distribution π_β approaches the uniform distribution on the solutions to the N-queens problem. Hence Z_∞ is the required number of solutions. By using the Metropolis algorithm to sample from π_β, we cleverly avoided having to compute precisely this quantity, but now we attempt to estimate it.

2.1 Theory

Consider the algorithm given in Algorithm 2 to estimate the value of Z_∞.
It is easy to see that this algorithm is indeed estimating the right quantity as long as β^* is large enough so

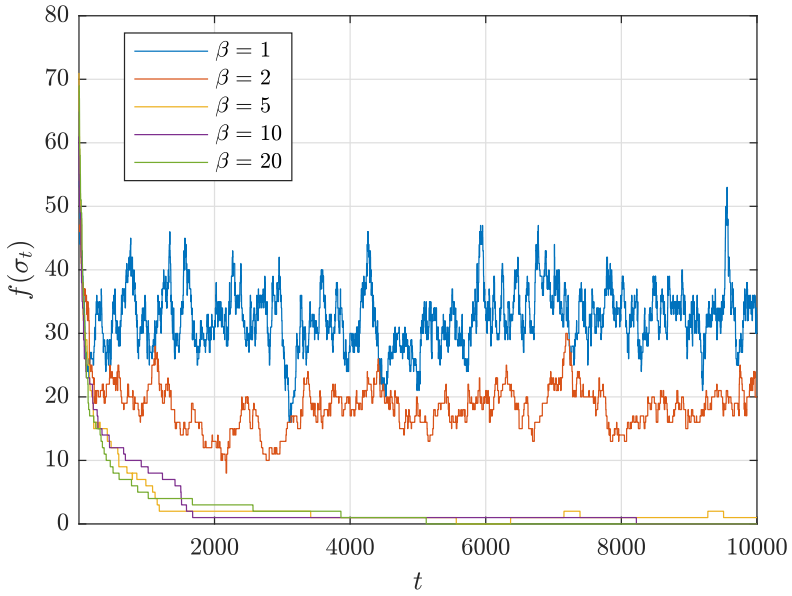


Figure 2: "What inverse temperature β should be chosen?": Observe that for small β (such as $\beta = 1, 2$), the limiting distribution π_β simply has too large a fraction of the total probability on configurations that are not solutions to the problem, leaving us in a situation where $f(\sigma_t)$ never becomes equal to zero. Beyond a certain point however (such as for $\beta = 20$), π_β has most of its probability concentrated on the solutions to the N-queens problem, which is exactly our goal. Results shown for $N = 100$.

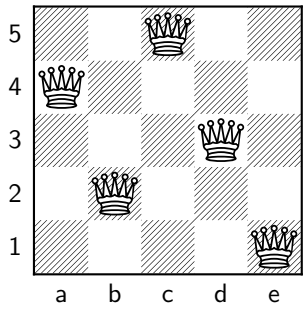
Algorithm 2: To estimate the number of solutions to the N-queens problem

Input: N, S = permutations of $[N]$, f, M, T, β^*, $\{(\beta_i)_{i=0}^T : 0 = \beta_0 < \beta_1 < \cdots < \beta_T = \beta^*\}$
Output: $\widehat{Z_\infty}$ = an estimate of Z_∞

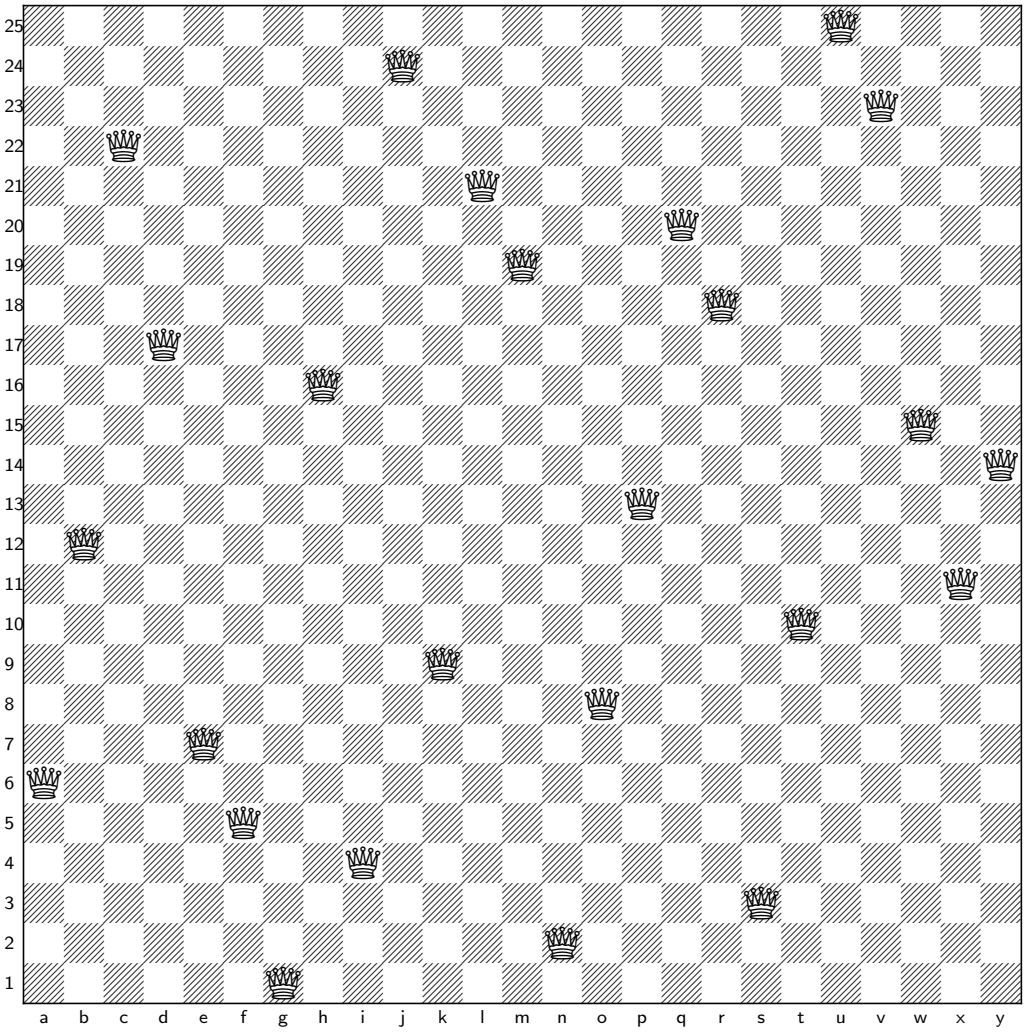
Compute $Z_0 \leftarrow N!$
for $t = 0, \ldots, T-1$ **do**
  Run the (modified) Metropolis algorithm, Algorithm 1′ ,to obtain $\pi_{\beta_t}(x) = \dfrac{\exp(-\beta_t f(x))}{\sum_{y \in S} \exp(-\beta_t f(y))}$
  Draw M i.i.d. samples $X_1, \ldots, X_M$ from π_{β_t}
  Compute $\Gamma_t \leftarrow \frac{1}{M} \sum_{k=1}^M \exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]$
Compute and output $\widehat{Z_\infty} \leftarrow Z_0 \prod_{t=0}^{T-1} \Gamma_t$.

that $Z_{\beta^*} \approx Z_\infty$, as

$$
\begin{aligned}
\mathbb{E}\left[\widehat{Z_\infty}\right] &= \mathbb{E}\left[Z_0 \prod_{t=0}^{T-1} \Gamma_t\right] \\
&= \mathbb{E}\left[Z_0 \prod_{t=0}^{T-1} \left\{\frac{1}{M} \sum_{k=1}^{M} \exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]\right\}\right] \\
&= Z_0 \prod_{t=0}^{T-1} \left\{\frac{1}{M} \sum_{k=1}^{M} \mathbb{E}\left[\exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]\right]\right\} \qquad [\because X_k \text{ are drawn i.i.d. at each } t; \text{linearity of } \mathbb{E}] \\
&= Z_0 \prod_{t=0}^{T-1} \left\{\sum_{x \in S} \exp\left[-(\beta_{t+1} - \beta_t) f(x)\right] \pi_{\beta_t}(x)\right\} \\
&= Z_0 \prod_{t=0}^{T-1} \frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} = Z_{\beta^*} \approx Z_\infty,
\end{aligned}
$$

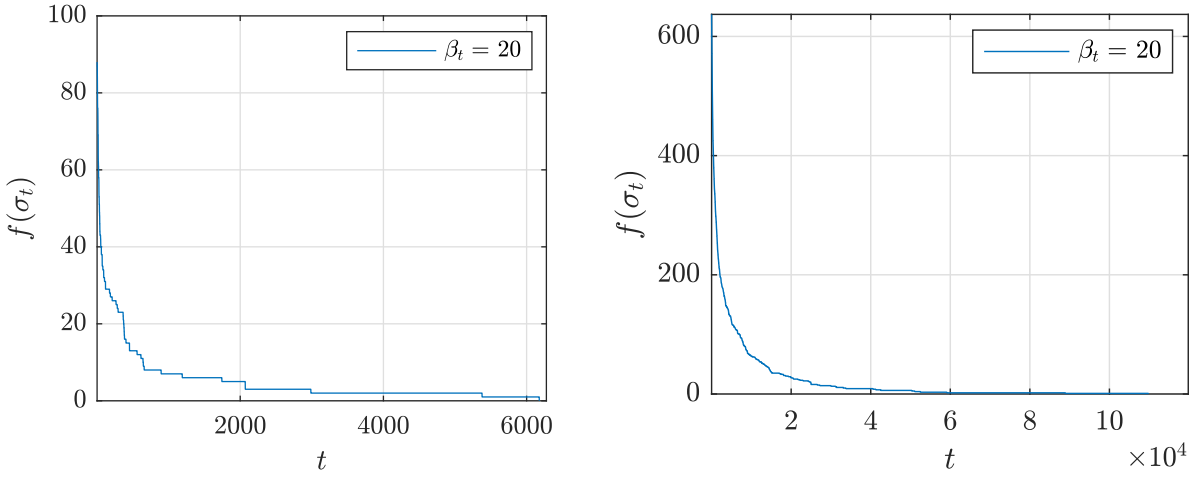


(a) $N = 5$



(b) $N = 25$

Figure 3: Solutions obtained using Algorithm 1; observe that no two queens can attack each other.



(a) $N = 100$ (b) $N = 1000$

Figure 4: Plots showing the evolution of the energy function evaluated at each iteration of the Metropolis algorithm (Algorithm 1) to obtain a solution to the N-queens problem, using a constant inverse temperature of 20; note that the algorithm stops when $f(\sigma_t)$ hits zero, i.e., a solution is reached.

i.e., it is nearly an unbiased estimator. But how good is it? The **variance** of $\widehat{Z_\infty}$ can be calculated as

$$\begin{aligned}
\operatorname{var}\left(\widehat{Z_\infty}\right) &= \operatorname{var}\left(Z_0 \prod_{t=0}^{T-1} \Gamma_t\right) \\
&= Z_0^2 \left\{ \mathbb{E}\left[\left(\prod_{t=0}^{T-1} \Gamma_t\right)^2\right] - \left(\mathbb{E}\left[\prod_{t=0}^{T-1} \Gamma_t\right]\right)^2 \right\} && [\because \operatorname{var}(aX) = a^2\operatorname{var}(X); \operatorname{var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2] \\
&= Z_0^2 \left\{ \mathbb{E}\left[\prod_{t=0}^{T-1} \Gamma_t^2\right] - \left(\mathbb{E}\left[\prod_{t=0}^{T-1} \Gamma_t\right]\right)^2 \right\} && [\text{by rearranging terms}] \\
&= Z_0^2 \left\{ \prod_{t=0}^{T-1} \mathbb{E}\left[\Gamma_t^2\right] - \left(\prod_{t=0}^{T-1} \mathbb{E}\left[\Gamma_t\right]\right)^2 \right\} && [\because \Gamma_t \text{ are independent}] \\
&= Z_0^2 \left\{ \prod_{t=0}^{T-1} \left(\operatorname{var}(\Gamma_t) + \mathbb{E}\left[\Gamma_t\right]^2\right) - \prod_{t=0}^{T-1} \mathbb{E}\left[\Gamma_t\right]^2 \right\}. && [\text{by rearranging terms; } \operatorname{var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2]
\end{aligned}$$

Consider $\Gamma_t = \frac{1}{M}\sum_{k=1}^{M} \exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]$, for which we have

$$\begin{aligned}
\operatorname{var}(\Gamma_t) &= \frac{1}{M^2} \operatorname{var} \sum_{k=1}^{M} \left(\exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]\right) && [\because \operatorname{var}(aX) = a^2\operatorname{var}(X)] \\
&= \frac{1}{M} \operatorname{var}\left(\exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]\right). && [\because \operatorname{var}(X+Y) = \operatorname{var}(X) + \operatorname{var}(Y) \text{ if } X \perp\!\!\!\perp Y;\ X_k \text{ are i.i.d.}]
\end{aligned}$$

Note that $0 \leq f(X_k) \leq N < \infty$, and $\beta_{t+1} - \beta_t > 0$, so $\exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]$ lies in $(0, 1]$. For a random variable bounded on $[a, b]$, the maximum possible variance is $\frac{(b-a)^2}{4}$. Here, we have $a = 0, b = 1$, which gives us

$$\operatorname{var}(\Gamma_t) = \frac{1}{M} \operatorname{var}\left(\exp\left[-(\beta_{t+1} - \beta_t) f(X_k)\right]\right) \leq \frac{1}{4M}.$$

Additionally, since $\beta_{t+1} > \beta_t$, we must have $\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} < 1$ (since $\exp[-\beta f(x)]$ is a nonincreasing function of β in general, and decreasing for all x such that $f(x) > 0$). Let the maximum value taken by this ratio over all $t \in \{0, \ldots, T-1\}$ be $r < 1$. Then we have,

$$\begin{aligned}
\operatorname{var}\left(\widehat{Z_\infty}\right) &= Z_0^2 \left\{ \prod_{t=0}^{T-1} \left(\operatorname{var}(\Gamma_t) + \mathbb{E}\left[\Gamma_t\right]^2 \right) - \prod_{t=0}^{T-1} \mathbb{E}\left[\Gamma_t\right]^2 \right\} \\
&\leq Z_0^2 \prod_{t=0}^{T-1} \left(\frac{1}{4M} + r^2 \right) = Z_0^2 \left(\frac{1}{4M} + r^2 \right)^T \\
&\approx \frac{Z_0^2}{M^T}, \qquad (8)
\end{aligned}$$

if M is chosen such that $r^2 \ll \frac{3}{4M}$, or equivalently, $M \ll \frac{3}{4r^2}$. While loose, it is still a useful upper bound.

2.2 Experiments

Once again, all simulations were performed on MATLAB and here are some considerations that are of interest from a practical perspective.

1. As N gets large, $N!$ very quickly becomes impossibly large to deal with, so much so that MATLAB simply outputs `Inf` for $N > 200$. To avoid obtaining such unhelpful expressions, it is better to keep track of *the logarithm* of Z_0 and each Γ_t, and then simply add them all up to obtain an estimate of $\log Z_\infty$. This also makes the computations easier since the quantities involved are smaller in magnitude. (All logarithms involved as results, plots or otherwise are computed with respect to base 10.)

2. To obtain i.i.d. samples from π_{β_t}, the Metropolis algorithm must (1) run until it "reaches" the stationary distribution, and (2) produce *independent* samples. Simply taking the first M samples after reaching steady state will not give i.i.d. samples, since they will be correlated. One way to eliminate this correlation is to take only one in, say, 500 samples and discard the rest. As for reaching the stationary distribution, the ideal solution would be to calculate the mixing time, but an easier method is to simply look at the plots of $f(\sigma_t)$ vs. t as in Figures 1 and 2 for the desired ranges of N, and make an estimate of how long it takes to get close enough to steady state. For large N, we found that N^2 samples were sufficient, and for small N, $N^2\beta^*$ samples were needed. Of course, this is not an exact method, but as we shall see soon, these values are indeed enough to obtain good estimates.

3. Continuing from the variance analysis, we have the following. Even when $f(x) = 2$ for all $x \in S$, taking $\beta_{t+1} - \beta_t = 1$ gives us $r = \frac{1}{e^2} \approx 0.135$. However, $f(x)$ takes values all the way up to N. Hence, even with smaller values of $\beta_{t+1} - \beta_t$, we can expect to have $r > \frac{1}{e^2}$, which gives us $M \ll \frac{3e^4}{4} \approx 40$, so taking $M = 10$ should give a reasonable estimate with an appropriate choice of T such that $\frac{Z_0^2}{M^T} \ll 1$. Let $M = 10$, then choosing T such that $T \gg 2N \log N > \log Z_0^2$ will suffice (but is often not necessary).

4. As seen from the simulation results for the previous section, $\beta = 20$ seems to give a π_β that has most of its probability on the solutions to the N-queens problem, thus we take $\beta^* = 20$. An easy choice of β_t, then, is given by $\beta_t = \frac{t}{T}\beta^*$ for $t = 0, \ldots, T$.

One way to check if the algorithm is indeed estimating the number of solutions correctly is to plot $\log Z_{\beta_t}$ as a function of β_t. One would expect the plot to start at a high value at $\beta_0 = 0$, and then decrease as β_t increases. If our choice of $\beta_T = \beta^*$ were good enough, in the sense that $Z_{\beta^*} \approx Z_\infty$, $\log Z_{\beta_t}$ should become nearly constant as β_t gets closer to β^*. We would also want the plots to be as "smooth" as possible, since Z_β is a continuous function of β. If out plots are either not smooth or not saturating, there must have been a mistake in our choice of one of the time to reach steady state, M, or T. In Figure 5, we plot $\log Z_{\beta_t}$ vs. β_t for $N = 10, 15, 20, 25$, and observe that we do indeed get a continuous plot that saturates by taking $\beta^* = 20$, $M = 10$, and $T = 200$, which suggests that the algorithm is correctly estimating the number of solutions. In fact, for these N, the number of solutions in known exactly, and they are compared in Table 1.

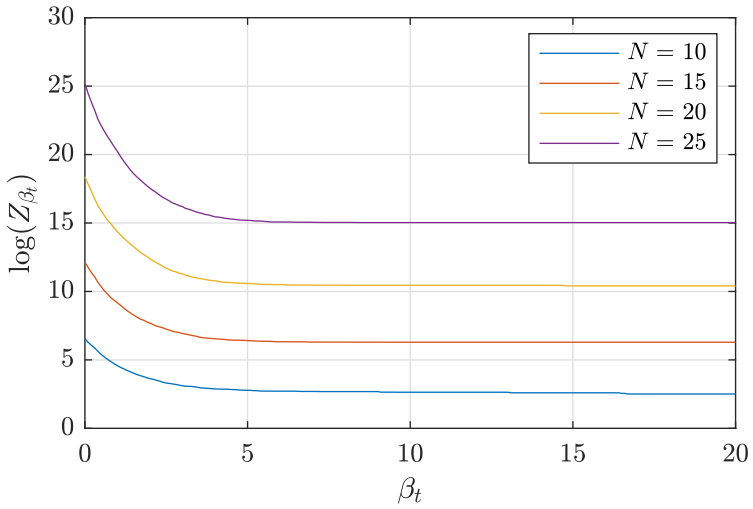


Figure 5: Plot showing $\log Z_{\beta_t}$ vs. β_t; observe that we get essentially continuous plots (in the sense of having no jumps) that saturate to $\log Z_\infty$ as expected, giving solutions that are very close to the actual values as seen in Table 1.

N	Exact number of solutions, Z_∞	Estimated number of solutions, $\widehat{Z_\infty}$	$\log Z_\infty$	$\log \widehat{Z_\infty}$
10	724	712	2.8597	2.8526
15	2,279,184	1,788,021	6.3578	6.2524
20	39,029,188,884	21,059,472,284	10.5914	10.3234
25	2,207,893,435,808,352	2,749,411,828,087,284	15.3440	15.4392

Table 1: Comparison of the estimated number of solutions versus the exact number of solutions when the value is known exactly, taken from the Wikipedia page [2]. The values of $\widehat{Z_\infty}$ are rounded to the nearest integer; the $\log Z_{\beta_t}$ vs. β_t plots corresponding to these estimated values are given in Figure 7.

Another check is to make use of the recently calculated asymptotic growth rate of $(0.143\ N)^N$ [1], and plot the number of solutions obtained in the same run as above against this curve, which we do in Figure 6, but this time, with larger values of $N = 100, 200, 300, 500$, with the parameters $\beta^* = 20$, $M = 10$, and $T = 500$. The associated plots of $\log Z_{\beta_t}$ vs. β_t are given in Figure 7.

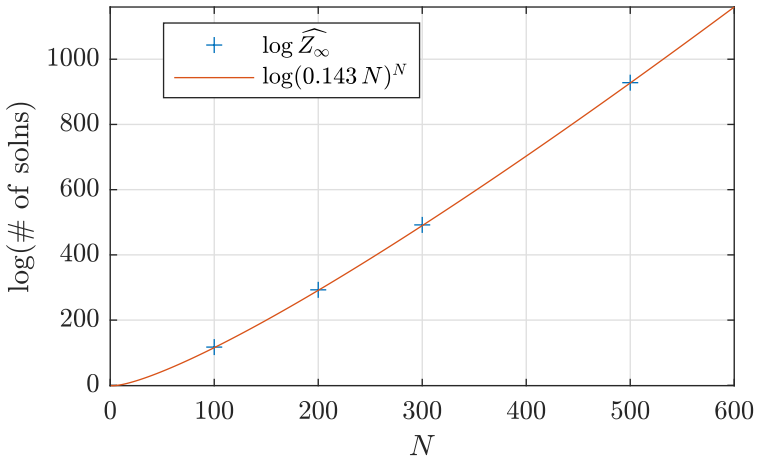


Figure 6: Comparison of the estimated number of solutions versus the asymptotic growth rate given by Simkin [1]; the $\log Z_{\beta_t}$ vs. β_t plots corresponding to these estimated values are given in Figure 7.

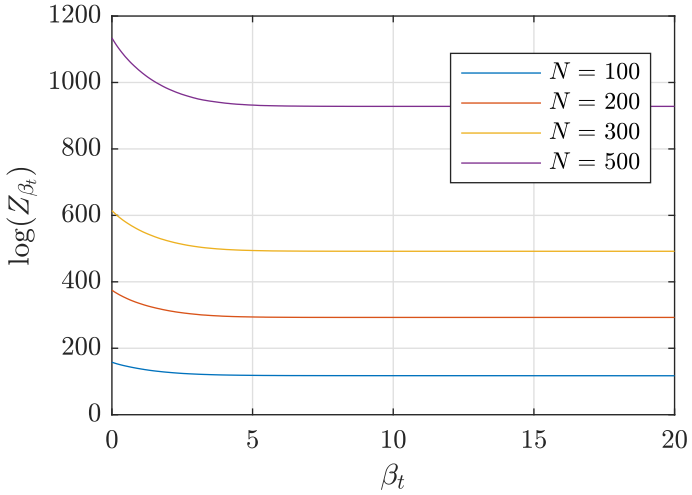


Figure 7: Plot showing $\log Z_{\beta_t}$ vs. β_t for large values of N; once again, observe that we get essentially continuous plots that saturate to $\log Z_\infty$ as expected, to obtain solutions that are virtually indistinguishable from what is expected theoretically as seen in Figure 6.

Conclusion

In Part 1, we used the Metropolis algorithm to find a solution to the N-queens problem. Using the information obtained from this exercise (such as the time needed to reach a steady state), in part 2, we estimated the number of solutions. Our estimates are close to the actual values where they are known, and also match

the asymptotic growth rate for large N.

Note: We have also implemented the Metropolis algorithm using Numba CUDA and is available in the attached Jupyter notebook. While the approximate running time is similar to a highly vectorized CPU implementation, the CUDA implementation should provide significant speedups for large values of N.

References

[1] Michael Simkin. *The number of n-queens configurations*. 2021. URL: https://arxiv.org/abs/2107.13460.

[2] Wikipedia contributors. *Eight queens puzzle — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-December-2022]. 2022. URL: https://en.wikipedia.org/wiki/Eight_queens_puzzle.