Solving the N-queens problems via the Metropolis algorithm

Francesco Salvi, Neta Singer, Perrine Vantalon

***Abstract*—The N-queens problem demands a way to place N queens on an $N \times N$ chessboard so that no two queens attack each other (lie on the same row, column, or diagonal). In this paper, we discuss a Metropolis–Hastings algorithm for sampling a solution to the problem, for any board size. Moreover, we use this algorithm to estimate the number of solutions, by sequentially approximating the partition function. Our method finds a solution for $N \leq 1000$ in less than $20\,\mathrm{s}$ and approximates correctly the number of solutions for all the exact values up to $N = 26$. For large N, instead, our estimates scale as $N!/e^N$, coherently with previous literature.**

I. Finding a solution

A. Setup

The idea of the Metropolis algorithm is to design a Markov Chain - the Metropolis chain - whose stationary and limiting distribution approximates the uniform distribution over all the solutions to our problem, i.e. the distribution that we would ideally like to sample from. To create such a chain, we define the following elements:

1) The *state space* S, defined as the set of configurations with exactly one queen per row and column of the board. Formally, a valid configuration will thus be described by an N-dimensional vector σ, such that $\sigma_i \in \{1...N\}\,\forall i$ and $\sigma_i \neq \sigma_j \;\; \forall i \neq j$, whose element σ_i indicates the column associated to the queen occupying the i-th row. For example, with $N = 4$, the vector $\sigma = (3, 2, 4, 1)$ would correspond to a board with queens in positions (1, 3), (2, 2), (3, 4), and (4, 1). Therefore, valid configurations are in a one-to-one mapping to the set of permutations of N numbers, and we can formalize $S = \{1...N\}! = Sym(\{1...N\})$. Clearly, $|S| = N!$. This choice of S is convenient because it already excludes any conflicts along the rows and columns of the board, so that we will only need to check for conflicts along diagonals.
2) The *base chain* ψ, defined as the chain which swaps the columns of any two queens uniformly at random. Formally, we say that two valid configurations σ and τ are neighbors, and indicate that as $\sigma \sim \tau$, iff $\exists\, j, k$ such that:

$$\begin{aligned} &\sigma_i = j \quad \tau_i = k \\ &\sigma_l = k \quad \tau_l = j \\ &\sigma_r = \tau_r \quad \forall r \notin \{i, l\} \end{aligned} \tag{1}$$

Then, transition probabilities for the base chain are defined as:

$$\psi_{\sigma\tau} = \begin{cases} \frac{2}{N^2} & \text{if } \sigma \sim \tau \\ \frac{1}{N} & \text{if } \sigma = \tau \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Each configuration has exactly $\binom{N}{2}$ neighbors, so $\sum_{\tau \in S} \psi_{\sigma\tau} = \frac{2}{N^2} \cdot \frac{N(N-1)}{2} + \frac{1}{N} = 1,\ \forall \sigma \in S$. Note that the chain is irreducible, since all valid configurations can be reached from any starting configuration σ with at most $N - 1$ swaps, aperiodic, because self-loops are allowed, and symmetric. In particular, $\psi_{\sigma\tau} > 0 \iff \psi_{\tau\sigma} > 0$. Therefore, all the conditions of the Metropolis algorithm are satisfied, and we can easily sample from ψ by randomly generating two numbers between 1 and N, indicating the elements of σ to swap.
3) The *energy function* f, counting all the conflicts across the board, which we aim at minimizing. We define a "conflict" to be any attack between two (different) queens, so that:

$$\begin{aligned} f(\sigma) &= \#\text{conflicts of configuration } \sigma \\ &= \sum_{i>j} \mathbb{1}\{\text{queen } i \text{ attacks queen } j \text{ in } \sigma\} \end{aligned} \tag{3}$$

with $\mathbb{1}$ being the identity function. Finding a minimum of f, thus, means finding a solution to the N-queens problem. Ideally, we would like to be able to sample from a distribution π_∞ such that:

$$\pi(\sigma) = \begin{cases} \frac{1}{Z_\infty} & \text{if } f(\sigma) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

with Z_∞ being the number of states with 0 conflicts. Since this is infeasible, even as the limiting distribution of our Metropolis chain, we approximate π with π_β, for a given $\beta \geq 0$, so that:

$$\pi_\beta(\sigma) = \frac{e^{-\beta f(\sigma)}}{Z_\beta} \tag{5}$$

with $Z_\beta = \sum_{\sigma \in S} e^{-\beta f(\sigma)}$, and we use π_β in the algorithm as the target stationary distribution. Note that when $\beta = \infty$ this distribution reduces to the ideal but hard-to-sample one defined above, i.e. $\pi_\infty = \pi$. When $\beta = 0$, instead, π_0 is the uniform distribution across all states.
4) The *acceptance probabilities* a, such that:

$$a_{\sigma\tau} = \min\left(1, \frac{\pi_\tau \psi_{\tau\sigma}}{\pi_\sigma \psi_{\sigma\tau}}\right) = \min\left(1,\ e^{-\beta(f(\tau) - f(\sigma)}\right)$$

> This means that a move is always accepted if it decreases the number of conflicts. When the number of conflicts increases, instead, the move is only accepted with probability inversely proportional to the conflict difference.

With this set of choices, the *Metropolis chain* p is defined as:

$$p_{\sigma\tau} = \begin{cases} \dfrac{2}{N^2} & \text{if } \sigma \sim \tau,\ f(\tau) \leq f(\sigma) \\ \dfrac{2\, e^{\beta(f(\tau)-f(\sigma))}}{N^2} & \text{if } \sigma \sim \tau,\ f(\tau) > f(\sigma) \\ \dfrac{1}{N} + \displaystyle\sum_{\nu \neq \sigma} \psi_{\sigma\nu} a_{\sigma\nu}(1 - a_{\sigma\nu}) & \text{if } \sigma = \tau \\ 0 & \text{otherwise} \end{cases}$$

This chain is ergodic and reversible, with stationary and limiting distribution π_β.

B. Implementation choices

Since our state space is defined so that conflicts can only take place on diagonals (cf §I-A), we can devise an efficient algorithm to keep track of the number of conflicts when updating the board. As shown in Table 1, each diagonal is associated with a unique number, which can be computed given the coordinates of any of its cells.

2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8

(a) Increasing diagonals (row + column)

0	-1	-2	-3
1	0	-1	-2
2	1	0	-1
3	2	1	0

(b) Decreasing diagonals (row - column)

Table 1: Diagonal numbers in a 4x4 grid.

We can therefore store two hash maps Q^{incr} and Q^{decr}, respectively for increasing and decreasing diagonals, counting how many queens lie in each diagonal at any time. The number of conflicts is then computed as:

$$f(\sigma) = \sum_{i=2}^{2N} \binom{Q^{incr}[i]}{2} + \sum_{i=-N+1}^{N-1} \binom{Q^{decr}[i]}{2} \tag{6}$$

Given an initial state σ_0, both counters are initialized by looping once through σ_0, in an O(N) pass. At each update, i.e. for each swap of two queens, we only need to update the counts of the old and new diagonals where the moving queens lie. This accounts for at most 8 numbers, since each queen occupies two diagonals both before and after the swap. The number of conflicts is then trivially updated by taking into account the difference between the modified counts. The resulting scheme for performing an update, thus, runs in time $O(1)$.

C. Simulations and parameter tuning

The algorithm is successful in finding valid solutions to the N-queens problem, as can be seen in Figure 1 for a small grid. Figure 2 shows instead the time to find a solution as a function of N. For $N < 5000$, the growth seems to be close to a power law.

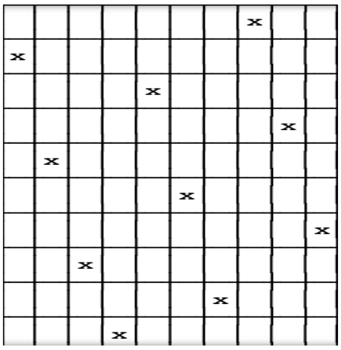


Fig. 1: Example solution with N=10.

Since the algorithm depends critically on the choice of β, we experimented with different strategies on a board with $N = 1000$:

- Keeping a fixed β. When β is chosen to be small ($\beta \leq 5$), the algorithm never converges to a solution, because the probability to accept unfavorable moves is too high. For medium values ($5 < \beta \leq 10$), instead, the algorithm converges, but with an average runtime that decreases as we increase β. For large values ($\beta \geq 10$), finally, the runtime stabilizes around the values shown in Figure 2. For such values, unfavorable moves are basically never accepted, so all the βs in this range are virtually equivalent. Note that those ranges refer to configurations with $N = 1000$, but will vary for other board sizes, since the number of iterations needed to get to a solution is in general a function of N.
- Using *simulated annealing*. In particular, we start with $\beta = 0.01$ and we double β every 1000 iterations. Note that this has a concrete impact only on the first 10 000 iterations, since afterwards $\beta > 10$. Other stepsizes and multiplicative factors were also considered, but didn't lead to any improvements in performance.

The two approaches resulted in comparable runtimes, without any apparent benefit coming from the usage of an annealing scheme. In fact, while simulated annealing is helpful in problems that benefit from an initial exploration phase, followed by an exploitation phase to get to a solution, in the N-queens problem exploration is generally not needed. This is because all the solutions are reachable from any state in S, and because the states are highly symmetrical: the only thing that matters for convergence is the current number of conflicts. For this reason, increasing $f(\sigma)$ is always a "bad" move, and the optimal strategy is to only make use of exploitation. The plot of Figure 2, therefore, was only generated with fixing $\beta = 20$.

II. Estimating the number of solutions

A fundamental problem that may be addressed through Metropolis sampling is estimating the number of N-queens solutions for a given board size. In general, counting the number of solutions is computationally very expensive, and

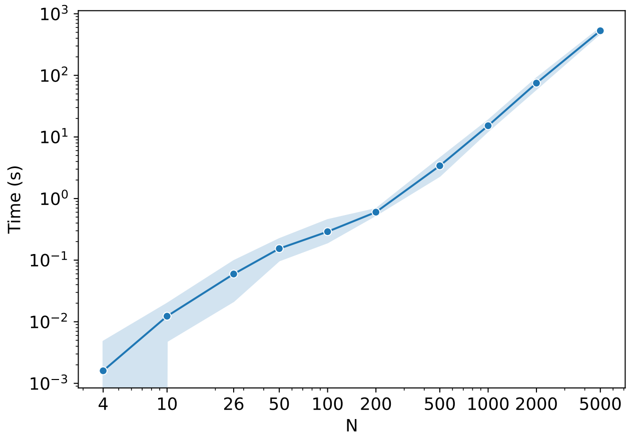


Fig. 2: Time to find a solution of the N-queens problem, as a function of N, in a log-log scale. The dots show the average runtime over 10 iterations, and the error bars represent 95% non-parametric confidence intervals.

exact values are only known for $N \leq 26$. This is explained by the extremely rapid growth (factorial) of the set of valid configurations: with $N = 8$, there are only 92 different solutions, but already with $N = 20$ the number blows up to more than 1 billion. Estimating this quantity corresponds to estimating the partition function Z_∞ in (5), which is equal to the number of 0-conflict configurations, i.e. of solutions to the problem. An easy way to get such an estimate would be to sample from π_0 (the uniform distribution over all configurations) and count the number of configurations with 0 conflicts, dividing it by the total number of samples to estimate the ratio $\frac{Z_\infty}{Z_0}$. Clearly, since $Z_0 = |S| = N!$, this would give a direct estimate of $\frac{Z_\infty}{Z_0}$. However, when N is large this ratio will be extremely small, and the variance of our estimator will be huge. Therefore, we will instead discuss an alternative method to estimate the ratio with low variance, leveraging the Metropolis algorithm.

A. Sampling Method

In order to build an appropriate estimator, we first want to find a β^* that behaves like $\beta = \infty$ for the Metropolis chain, for which we will assume that $Z_\infty \approx Z_{\beta^*}$. Since π_{β^*} behaves as π_∞, the ratio $\frac{Z_{\beta^*}}{Z_0}$ will also be very small. Instead of estimating this value directly, we define the sequence $0 = \beta_0 < \beta_1 < ... < \beta_T = \beta^*$ and express the ratio as:

$$\frac{Z_{\beta^*}}{Z_0} = \prod_{t=0}^{T-1} \frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} \tag{7}$$

Each of the ratios in the product can in turn be rewritten as:

$$\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} = \sum_{x \in S} \exp\left(-(\beta_{t+1} - \beta_t) f(x)\right) \pi_{\beta_t}(x) \tag{8}$$

and estimated by sampling from the distribution π_{β_t} and averaging the summand over the samples. That is, given samples $X_1, ..., X_M \sim \pi_{\beta_t}$, we can approximate:

$$\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} \approx \frac{1}{M} \sum_{i=1}^{M} \exp(-(\beta_{t+1} - \beta_t) f(X_i)) \tag{9}$$

B. Parameter choices

The sampling method described in §II-A relies on three critical choices:

1) The choice of the value β^*: As discussed in §I-C, for all $\beta > 10$ the number of accepted swaps during a single run of the algorithm is on average very close to 0, which is what we expect for $\beta = \infty$. Therefore, we pick $\beta^* = 20$. Note that, with $\beta = 20$, there will be in expectation one accepted swap every half a billion iterations, while the typical running time of the algorithm is well below one million iterations.

2) The choice of the sequence $0 = \beta_0$, ..., $\beta_T = \beta^*$: Let $g(X_i) = \exp(-(\beta_{t+1} - \beta_t) f(X_i))$ and Y be the estimator of the ratio $\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}}$ as in (9), i.e. $Y = \frac{1}{M} \sum_{i=1}^{M} g(X_i)$. Then,

$$\mathbb{E}[Y] = \frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} \tag{10}$$

$$\begin{aligned} \operatorname{Var}(Y) &= \frac{1}{M} \operatorname{Var}(g(X_0)) \\ &\leq \frac{1}{M} (\mathbb{E}[g(X_0)] - \mathbb{E}[g(X_0)]^2) \\ &= \frac{1}{M} \left(\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} - \left(\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}} \right)^2 \right) \end{aligned} \tag{11}$$

where the inequality holds because $0 < g(X_0) \leq 1$. For ease of computation, we will work with logarithms to turn the product in (7) into a sum, and will hence estimate $\log(Y)$ instead of Y directly. Using a first-order Taylor approximation,

$$\begin{aligned} \operatorname{Var}(\log(Y)) &\approx \left(\frac{1}{\mathbb{E}[Y]} \right)^2 \operatorname{Var}(Y) \\ &\leq \frac{1}{M} \left(\frac{Z_{\beta_t}}{Z_{\beta_t+1}} \right) \end{aligned} \tag{12}$$

This means that the variance increases as the ratio becomes small, which also justifies the choice of not estimating $\frac{Z_\infty}{Z_0}$ directly. In order to get an efficient estimator, therefore, we want to choose a grid that keeps each ratio reasonably large. To do so, we run an estimator of $\frac{Z_{\beta_1}}{Z_{\beta_0}}, ..., \frac{Z_{\beta_{t+1}}}{Z_{\beta_t}}, ..., \frac{Z_{\beta_T}}{Z_{\beta_{T-1}}}$ on an initial grid $0 = \beta_0 < \beta_1 < ... < \beta_t < \beta_{t+1} < .. < \beta^*$ and we increase its granularity if some $\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}}$ comes up small, inserting an additional $\beta_+ = (\beta_t + \beta_{t+1})/2$ to get an updated grid $0 = \beta_0 < \beta_1 < ... < \beta_t < \beta_+ < \beta_{t+1} < .. < \beta^*$. We repeat this procedure iteratively until we get reasonably large values. Empirically, working with $N \leq 1000$ this led to the grid [0, 0.01, 0.1, 0.25, 0.375, 0.5, 0.625, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 4, 5, 10, 20].

3) The procedure for sampling from π_{β_t}: Estimating each ratio as in (9) requires generating M samples from π_{β_t}, for each β_t in the grid. By running the Metropolis algorithm discussed in §I-A with a fixed β_t, we know that the chain will eventually converge to π_{β_t}, but we don't know how long it will actually take for convergence to occur. In general, $\beta_t < \beta_{t+1}$ $\forall t \in \{0...T-1\}$, meaning that π_{β_t} is closer than $\pi_{\beta_{t+1}}$ to the uniform distribution π_0. Therefore, convergence to π_{β_t} happens faster than convergence to $\pi_{\beta_{t+1}}$, and we can

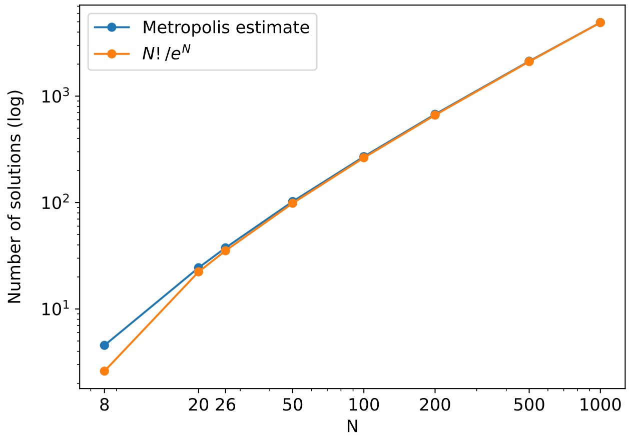


Fig. 3: log of the number of solutions as a function of N, in a log-log scale.

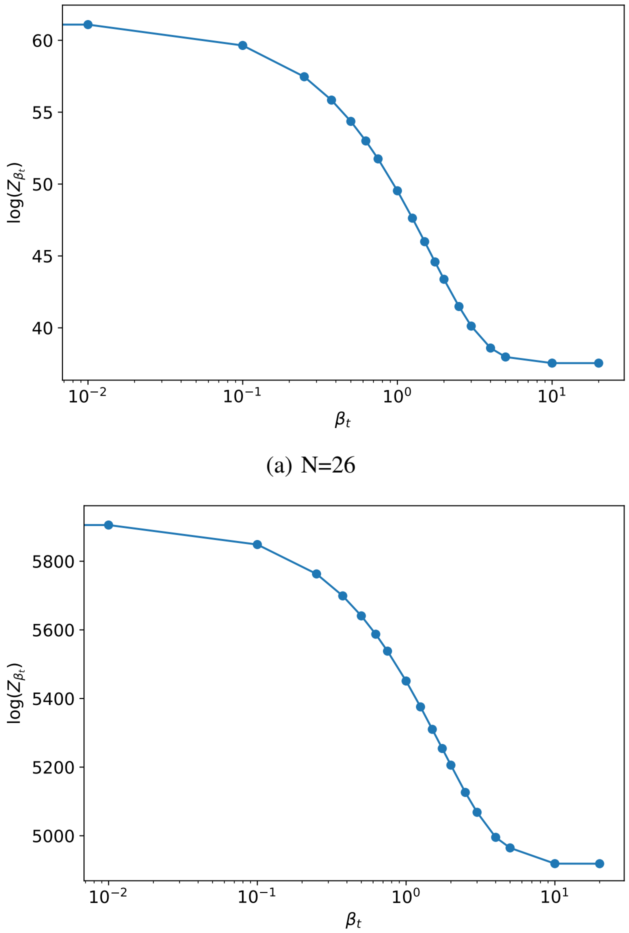
N	$\hat{Z}_\beta$	Z_∞
20	$(3.86 \pm 0.11) \times 10^{10}$	3.90×10^{10}
21	$(3.28 \pm 0.03) \times 10^{11}$	3.14×10^{11}
22	$(2.73 \pm 0.05) \times 10^{12}$	2.69×10^{12}
23	$(2.50 \pm 0.06) \times 10^{13}$	2.42×10^{13}
24	$(2.24 \pm 0.06) \times 10^{14}$	2.28×10^{14}
25	$(2.25 \pm 0.08) \times 10^{15}$	2.21×10^{15}
26	$(2.16 \pm 0.08) \times 10^{16}$	2.23×10^{16}
50	$(2.5 \pm 0.1) \times 10^{44}$	/
100	$(2.4 \pm 0.5) \times 10^{117}$	/
200	$(4 \pm 2) \times 10^{293}$	/
500	$(5 \pm 4) \times 10^{925}$	/
1000	$(1 \pm 1) \times 10^{2139}$	/

Table 2: Estimated number of solutions for N, compared with the exact values known for $N \leq 26$. Error estimates correspond to standard errors, i.e. standard deviation of the mean, computed over 5 repetitions.

upper bound the convergence times of all the π_{β_t}s with the convergence time of $\pi_{\beta_T} = \pi_{\beta^*}$. Since π_{β^*} behaves like π_∞, we will consider that convergence has occurred as soon as the algorithm reaches one 0-conflict configuration. Once such a configuration is reached, in fact, the Markov chain accepts moves to higher conflict states so rarely that the chain is almost deterministically staying at 0 conflicts, as a sampling under π_∞ would. Working with $N = 1000$ as an upper bound, we observe empirically that a solution is essentially always found before 200 000 iterations, and we will hence use 200 000 as the stopping time for all values of β_t. Afterwards, we continue to run the algorithm to generate $M = 300\,000$ samples.

C. Results

The results of our simulations estimating the number of solutions are shown in Figure 3. We observe that our estimates are very well approximated by a scaling proportional to $\frac{N!}{e^N}$, which corresponds to what has previously been observed in the literature [1, 2]. In addition, Table 2 reports the precise values of our estimates compared with the known exact values. In almost all cases, the exact values fall within the error bounds of our estimates, showing that our method performs well.



(a) N=26

(b) N=1000

Fig. 4: Growth of $\log \beta_t$ as a function of β_t, in a semilogx-scale.

Finally, Figure 4 shows the estimated values of $\log(Z_{\beta_t})$ for each β_t in the grid, computed by adding $\log(Z_0)$ to a telescoping sum on the ratios $\log\left(\frac{Z_{\beta_{t+1}}}{Z_{\beta_t}}\right)$. The plots reveal that on a semilog scale $\log(Z_{\beta_t})$ behaves as a sigmoid, both for $N = 1000$ and $N = 26$. Moreover, the figure supports our choice of the sequence discussed in §II-B2: the grid is coarser at the extremes of the curve and finer in the middle, where the first derivative is higher. That is, our choice of β_ts is such that there are no sharp variations in the slope between two subsequent values, preventing the variance from blowing up.

References

[1] Igor Rivin, Ilan Vardi, and Paul Zimmermann. "The n-Queens Problem". In: *The American Mathematical Monthly* 101.7 (Aug. 1994), pp. 629–639. DOI: 10.1080/00029890.1994.11997004.

[2] Jordan Bell and Brett Stevens. "A survey of known results and research areas for n-queens". In: *Discrete Mathematics* 309.1 (Jan. 2009), pp. 1–31. DOI: 10.1016/j.disc.2007.12.043.