Supplementary Material for Identifying Norms from Observation using MCMC Sampling, IJCAI 2021

Stephen Cranefield[1*] and **Ashish Dhiman**[2]
[1]Department of Information Science, University of Otago, Dunedin, New Zealand
[2]Department of Aerospace Engineering, Indian Institute of Technology, Kharagpur, India
stephen.cranefield@otago.ac.nz, ashish1610dhiman@gmail.com

1 Correction

The discussion at the bottom of page 5 of the upper plot in Figure 4 states the following:

> Pink above green occurs in some cases where $p_{nn} \geq 0.25$. This indicates that an expression in the sample has a posterior higher than the true norm, and therefore is a better explanation of the observed behaviour due to the high rates of non-normative behaviour. In all such cases, this expression was the "No-norm" expression.

The last sentence above is incorrect. For two runs (trial 2 for $p_{nn} = 0.25$ and trial 3 for $p_{nn} = 0.45$), the sampled expression with a posterior higher than the true norm was not the "No-norm" expression, but rather a simpler version of the true norm containing the obligation but not the permission.

2 The Metropolis-Hastings algorithm

The 'vanilla' textbook version of the Metropolis-Hastings algorithm [Gelman *et al.*, 2013] is shown below:

```
 1: procedure METROPOLIS-HASTINGS($obs, n$)
 2:                 ▷ $obs$: observed data; $n$: num. samples desired
 3:     Sample $\theta^0 \sim p_0(\theta)$ such that $p(\theta^0|obs) > 0$
 4:     for $i = 1, \cdots, n$ do
 5:         Sample $\theta^* \sim J(\theta^*|\theta^{i-1})$
 6:         $r = \dfrac{p(\theta^*|obs)/J(\theta^*|\theta^{i-1})}{p(\theta^{i-1}|obs)/J(\theta^{i-1}|\theta^*)}$
 7:         $\theta^i = \begin{cases} \theta^* & \text{with probability } min(r,1) \\ \theta^{i-1} & \text{otherwise} \end{cases}$
 8:     end for
 9:     Return $\langle \theta^1, ..., \theta^n \rangle$
10: end procedure
```

In the context of Bayesian inference, we have from Bayes' Theorem that $p(\theta^*|obs) = (p(\theta^*)\, p(obs|\theta^*))\,/p(obs)$ and likewise for $p(\theta^{i-1}|obs)$. These equalities can be applied to line 6 of the algorithm, with the two occurrences of $p(obs)$ cancelling out:

$$r = \frac{p(\theta^*)\, p(obs|\theta^*)\;/\;J(\theta^*|\theta^{i-1})}{p(\theta^{i-1})\, p(obs|\theta^{i-1})\;/\;J(\theta^{i-1}|\theta^*)} \tag{1}$$

[*]Contact author

We use that Bayesian-specific version of the algorithm in the paper.

3 MCMC over a probabilistic grammar

Saad et al. note that if the grammar generates a countably infinite language, it is necessary to check that the recursive process of generating an expression from the grammar will terminate with probability 1, by constructing its expectation matrix and verifying that its largest eigenvalue has a modulus less than 1. This holds for our grammar in the paper.

Saad et al. do not present their version of MCMC as an instance of the Metropolis-Hastings (M-H) algorithm, but rather give a long and complex description from basic principles. In particular, they define their acceptance ratio with no motivation and then prove that it has the desired mathematical properties. We believe that the method becomes more approachable and simpler to understand when viewed as a Metropolis-Hastings, and here show how their acceptance ratio can be derived from that in the M-H algorithm.

In the paper we describe the jumping distribution of Saad et al. in a sequence of bullet points (pages 2–3). Here we present the probability of the jump from θ to θ^*, based on the exposition of Saad et al., but using a more concise notation, e.g. our $P_G(nt, \theta)$ corresponds to their $\mathsf{Expand}[\![E]\!](N)$.

$$J(\theta^*|\theta) = 1/|NI(\theta)| \sum_{n \in NI(\theta) \cap NI(\theta^*)} \mathbb{1}(\theta^* \backslash n = \theta \backslash n)\, P_G(nt(n,\theta), \theta^*[n])$$

where:

- $NI(\theta)$ is the set of node indices in the parse tree of θ, represented as paths from the root node. For example, $(1,2)$ represents the path from the root through the first child note of the root, and then to that node's second child node. The size of an expression θ is defined as the size of the node indices set: $|NI(\theta)|$. In our paper, we abbreviate this as $|\theta|$.
- $\mathbb{1}$ is the indicator function, mapping a Boolean expression to 1 (for true) or 0. This allows a concise presentation where certain terms, multiplied by a indicator expression, cancel out under a given condition.
- $\theta \backslash n$ denotes θ with the subterm at index n removed and replaced with a special "hole" symbol.

- $P_G(nt, \theta)$ is the probability of the grammar generating the expression θ starting from the non-terminal symbol nt.
- $nt(n, \theta)$ denotes the non-terminal symbol in the grammar that was used to generate the subexpression of θ at node index n. This is well defined, as each production rule in a tagged PCFG generates expressions beginning with a "tag", i.e. a symbol that is unique to that production.
- $\theta[n]$ denotes the subexpression of θ that has the node at index n as its root.

The equation above calculates the *average probability* of jumping from θ to θ^* across all node indices of θ, as a subterm substitution could be made at any of them. For a given index n, only θ^*s containing the same index are possible results, hence the set intersection in the range of the summation index. The argument to the indicator function asserts that θ and θ^* must be identical except for any differences in their subtrees at index n. The final term expresses the probability of the grammar expanding the non-terminal associated with $\theta[n]$ to produce the replacement subterm $\theta^*[n]$.

Finally, in Figure 1 we show how the analysis of Saad et al. [2019] can be adapted to derive the acceptance rate r from the standard formula for r in the Bayesian application of the Metropolis-Hastings algorithm:

$$r = \frac{|NI(\theta^{i-1})|\, p(obs|\theta^*)}{|NI(\theta^*)|\, p(obs|\theta^{i-1})} \tag{2}$$

4 Our norm syntax

Figure 1 in the paper shows the norm grammar used in our simulated robot scenario. The grammar included this production rule:

```
COND ::= (moved COL SHAPE ZONE COND)
```
$\overset{0.\dot{3}}{::=}$

In this rule, the final COND non-terminal symbol generates a nested condition. (There is a separate production rule to handle the base case for COND, which specifies the next move that the norm constrains.) The rule above generates a conjunction of conditions that is expressed in a non-standard way as a recursively nested set of s-expressions. This choice is due to the requirements of the tagged probabilistic context-free grammar notation used by Saad et al.: there must be a fixed number of non-terminal symbols on the right hand side of a production rule, and each production rule must have a unique tag symbol at the start of the s-expression in its right hand side. Also, a rule that has no non-terminal symbols after its tag will produce a terminal symbol drawn from a probability distribution to follow the tag. Thus, a rule cannot produce a tag along (hence (null-cond true)) in the example below). These restrictions make a traditional conjunction syntax, such as the example below, unwieldy:

```
(cond-conj (moved ...)
           (cond-conj (next-move ...) (null-cond true)))
```

It is expected that some of these restrictions could be removed in future work.

$$r = \frac{p(\theta^*)\, p(obs|\theta^*)\, 1/|NI(\theta^*)| \sum_{n\in NI(\theta^*)\cap NI(\theta^{i-1})} \mathbb{1}(\theta^{i-1}\backslash n = \theta^*\backslash n)\, P_G(nt(n,\theta^*), \theta^{i-1}[n])}{p(\theta^{i-1})\, p(obs|\theta^{i-1})\, 1/|NI(\theta^{i-1})| \sum_{n\in NI(\theta^{i-1})\cap NI(\theta^*)} \mathbb{1}(\theta^*\backslash n = \theta^{i-1}\backslash n)\, P_G(nt(n,\theta^{i-1}), \theta^*[n])}$$

$$= \frac{p(obs|\theta^*)\, 1/|NI(\theta^*)| \sum_{n\in NI(\theta^*)\cap NI(\theta^{i-1})} \mathbb{1}(\theta^{i-1}\backslash n = \theta^*\backslash n)\, p(\theta^*)\, P_G(nt(n,\theta^*), \theta^{i-1}[n])}{p(obs|\theta^{i-1})\, 1/|NI(\theta^{i-1})|| \sum_{n\in NI(\theta^{i-1})\cap NI(\theta^*)} \mathbb{1}(\theta^*\backslash n = \theta^{i-1}\backslash n)\, p(\theta^{i-1})\, P_G(nt(n,\theta^{i-1}), \theta^*[n])} \quad \text{[Move priors inside summation]}$$

$$= \frac{p(obs|\theta^*)\, 1/|NI(\theta^*)| \sum_{n\in NI(\theta^*)\cap NI(\theta^{i-1})} \mathbb{1}(\theta^{i-1}\backslash n = \theta^*\backslash n) \left[\cancel{p(\theta^{i-1})}\, \frac{P_G(\cancel{nt(n,\theta^*),\theta^*[n]})}{P_G(\cancel{nt(n,\theta^*),\theta^{i-1}}[n])}\right] P_G(\cancel{nt(n,\theta^*),\theta^{i-1}[n]})}{p(obs|\theta^{i-1})\, 1/|NI(\theta^{i-1})| \sum_{n\in NI(\theta^{i-1})\cap NI(\theta^*)} \mathbb{1}(\theta^*\backslash n = \theta^{i-1}\backslash n)\, \cancel{p(\theta^{i-1})}\, P_G(\cancel{nt(n,\theta^{i-1}),\theta^*[n]})} \quad \text{[Saad et al. (Lemma 4.7) \& } nt(n,\theta^*) = nt(n,\theta^{i-1})\text{]}$$

$$= \frac{|NI(\theta^{i-1})|\, p(obs|\theta^*)}{|NI(\theta^*)|\, p(obs|\theta^{i-1})} \quad \text{[The generation of } \theta^* \text{ guarantees that } \theta^*\backslash n = \theta^{i-1}\backslash n\text{]}$$

Figure 1: Derivation of the formula for computing the acceptance rate r

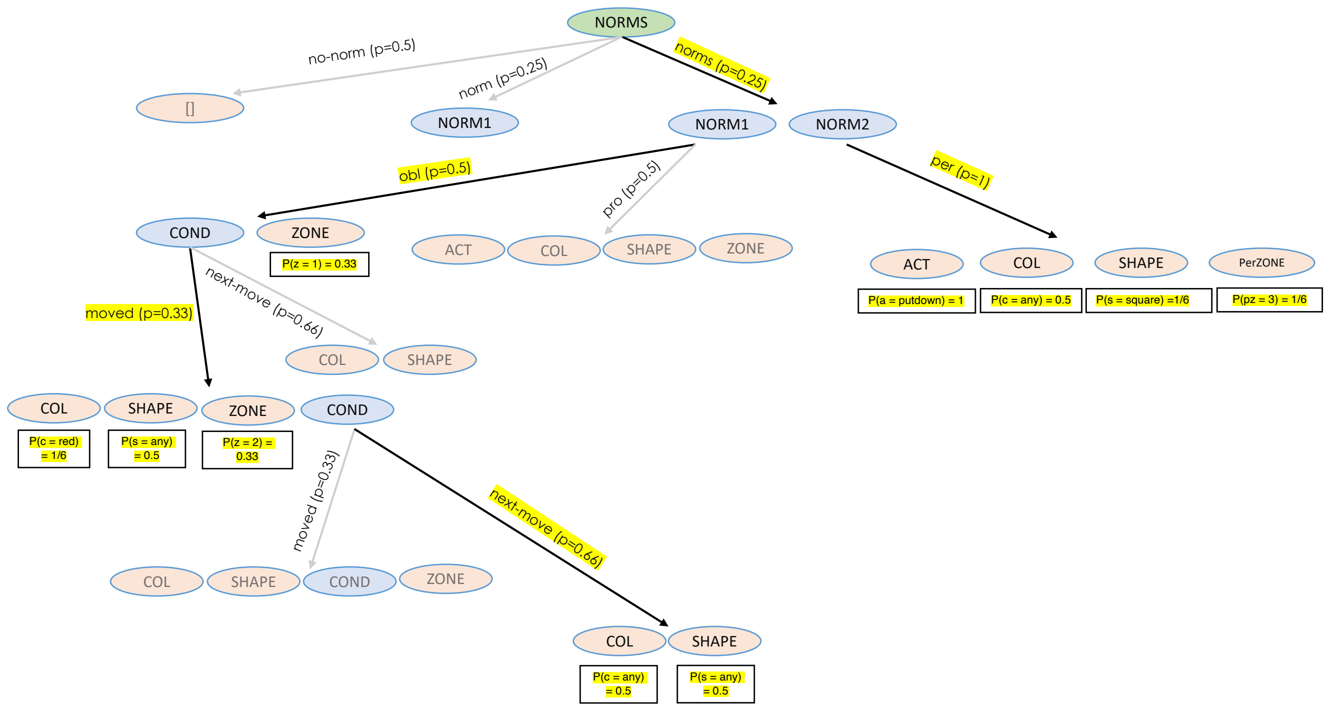

Figure 2: Illustration of the calculation of a prior probability

5 Prior calculation from the grammar

In this section, we want to present an example of how the prior is calculated for a given norm expression. To this end, we use the grammar described in Figure 1 in the paper and the true norm used in our experiments:

```
(norms
   (obl (moved (colour r) (shape any) (zone 1)
           (next-move (colour any) (shape any)))
        (zone 2))
   (per (action putdown) (colour any) (shape square)
        (per-zone 3)))
```

The prior calculation involves using the tags in the expression to tracing the path from the grammar's start symbol (NORMS) to the terminal symbols, while keeping track of the probability associated with each production rule and non-terminal symbol. This approach applied to the above expression yields the following probabilities:

```
[0.25, [0.5, [0.333, 0.166, 0.5, 0.333, [0.6666,
0.5, 0.5]], 0.333], [1, 1, 0.5, 0.166, 0.166]]
```

The prior probability is the product of these probabilities: 8.5787×10^{-7}. This is illustrated in Figure 2.

6 Comparison with the approach of Cranefield et al. [2016]

Due to lack of space, the paper omitted details of the six norm types in the scenario considered by Cranefield et al. [2016]. Informally, they are as follows. *Eventually* or *never* norms state that it is obligatory or (respectively) prohibited for an agent to pass through a given node (either unconditionally or conditionally once a given node has been reached). *Next* or *Not-next* norms state that it is obligatory or (respectively) prohibited to traverse between a specified pair of adjacent nodes. There is an instantiation of the unconditional *eventually* and *never* norm types for each node in the observed directed graph, and the conditional versions are instantiated for each pair of nodes. The *next* and *not-next* norm types have an instantiation for each edge in the graph. In addition, there was a *no-norm* null hypothesis. Cranefield et al. defined the prior odds of each non-null hypothesis over the null hypothesis to be 0.5.

Figure 3 shows the grammar we used to reproduce this norm language for the directed graph used by Cranefield et al. [2016], which has 30 nodes (named a to z then 0 to 3) and 36 edges. Note that, for example, the prior odds of `(next aw)` over `(no-norm true)` are $(36/1934 * 1/36)/(2/1934) = 0.5$, as required for compatibility with the experiments of Cranefield et al.

7 Progression of the posterior in MCMC chains

Figures 4 to 6 present three plots (for different p_{nn} values), with subplots depicting progression of log posterior in the individual chains. Note that MCMC search does not require the posterior distribution to be normalised, so our log posterior probabilities are simply the sum of the log prior and log likelihood.

Figure 4 is for an experimental trial where $p_{nn} = 0.0$, Fig-

$$
\begin{aligned}
\mathsf{NORMS} &\overset{2/1934}{::=} (\texttt{no-norm}\ t)\\
&\overset{30/1934}{|}\quad (\texttt{eventually NODE})\\
&\overset{30/1934}{|}\quad (\texttt{never NODE})\\
&\overset{36/1934}{|}\quad (\texttt{next EDGE})\\
&\overset{36/1934}{|}\quad (\texttt{not-next EDGE})\\
&\overset{900/1934}{|}\quad (\texttt{cond-eventually COND NODE})\\
&\overset{900/1934}{|}\quad (\texttt{cond-never COND NODE})\\
\mathsf{COND} &::= (\texttt{cond}\ c)\\
\mathsf{NODE} &::= (\texttt{node}\ n)\\
\mathsf{EDGE} &::= (\texttt{edge}\ e)
\end{aligned}
$$

where

$$
\begin{aligned}
P(t = \texttt{true}) &= 1\\
P(c \in \{\texttt{a}, \ldots, \texttt{z}, \texttt{0}, \ldots, \texttt{3}\}) &= \tfrac{1}{30}\\
P(n \in \{\texttt{a}, \ldots, \texttt{z}, \texttt{0}, \ldots, \texttt{3}\}) &= \tfrac{1}{30}\\
P(e \in \{\texttt{aw}, \texttt{cg}, \ldots \textit{(36 edges)}\}) &= \tfrac{1}{36}
\end{aligned}
$$

Figure 3: PCFG grammar for the ECAI'16 scenario of Cranefield et al.

ure 5 is for a trial where $p_{nn} = 0.3$, and finally Figure 6 is for $p_{nn} = 0.55$. These plots illustrate how, for low values of p_{nn}, the no-norm hypothesis (red line) has a lower log posterior than the true norm (green line), but this trend is reversed for a high value of p_{nn}. This is primarily because with an increase in p_{nn}, the likelihood $p(obs|\theta)$ decreases (due to a decrease in the normative component of the likelihood), while the prior remains constant. At a certain threshold of p_{nn}, the gap in likelihoods will no longer be able to cover the gap in priors of the no-norm hypothesis (which has a higher prior according to the grammar). This is also the same p_{nn} threshold above which the true norm ceases to be the expression with the highest log posterior in ten chains. This threshold, is intrinsic to the specific instances of 'observed behaviour', and the constraints affected by the true norm on such behaviour.

In many cases, we can see that not all chains find the true norm (or an equivalent expression with the same or very close log posterior). This is because some chains get stuck in certain locales of the search space. To alleviate this, we have used over-dispersed expressions as the starting expressions in the chains: we generate 10 times as many expressions as there are chains, use the vector representation outlined in the paper to compute their distances from the mean vector, and choose chain starting expressions from this set in decreasing order of distance from the mean. This, each chain may initially explore a different region of the search space, thereby helping prevent the worst case in which most chains are stuck, without finding the true (or an equivalent) norm.

References

[Cranefield *et al.*, 2016] Stephen Cranefield, Felipe Meneguzzi, Nir Oren, and Bastin Tony Roy Savarimuthu. A Bayesian approach to norm identification. In *22nd European Conference on Artificial Intelligence*, pages 622–629. IOS Press, 2016.

[Gelman *et al.*, 2013] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, third edition, 2013.

[Saad *et al.*, 2019] Feras A. Saad, Marco F. Cusumano-Towner, Ulrich Schaechtle, Martin C. Rinard, and Vikash K. Mansinghka. Bayesian synthesis of probabilistic programs for automatic data modeling. *Proceedings of the ACM on Programming Languages*, 3: 37:1–37:32, 2019.

Figures 4 to 6 appear on the next three pages.

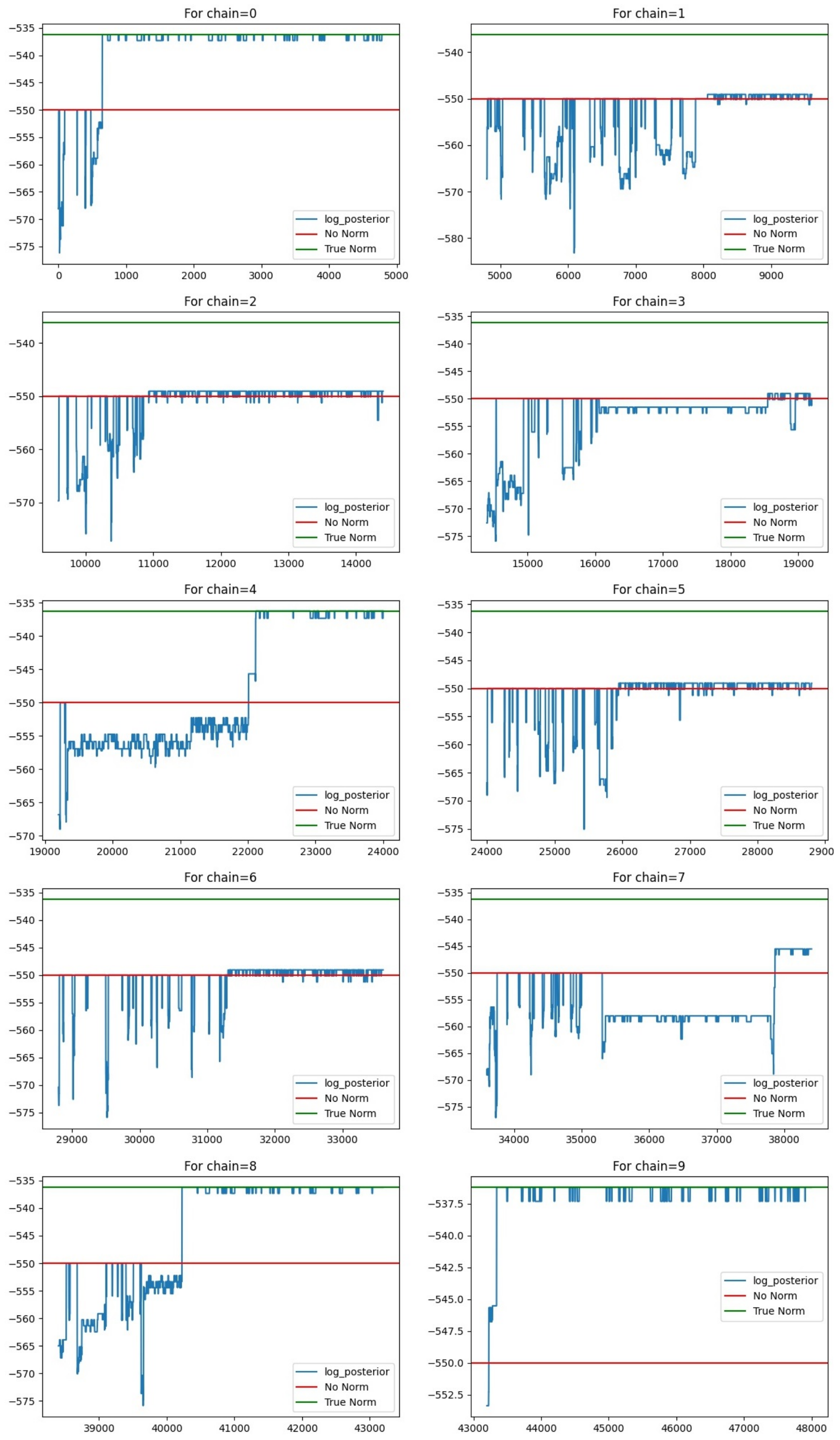

Figure 4: Chain plots for $p_{nn} = 0.0$

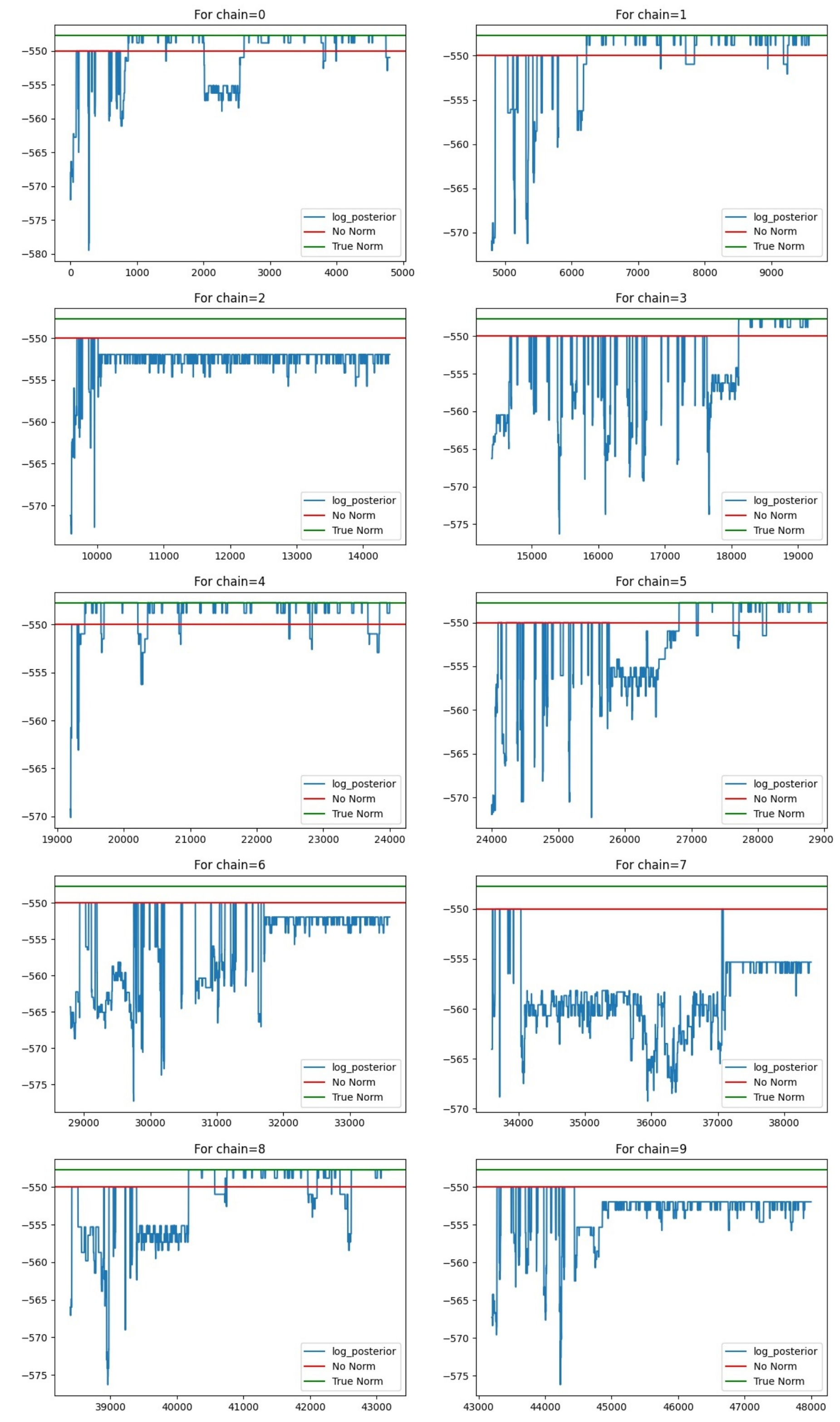

Figure 5: Chain plots for $p_{nn} = 0.3$

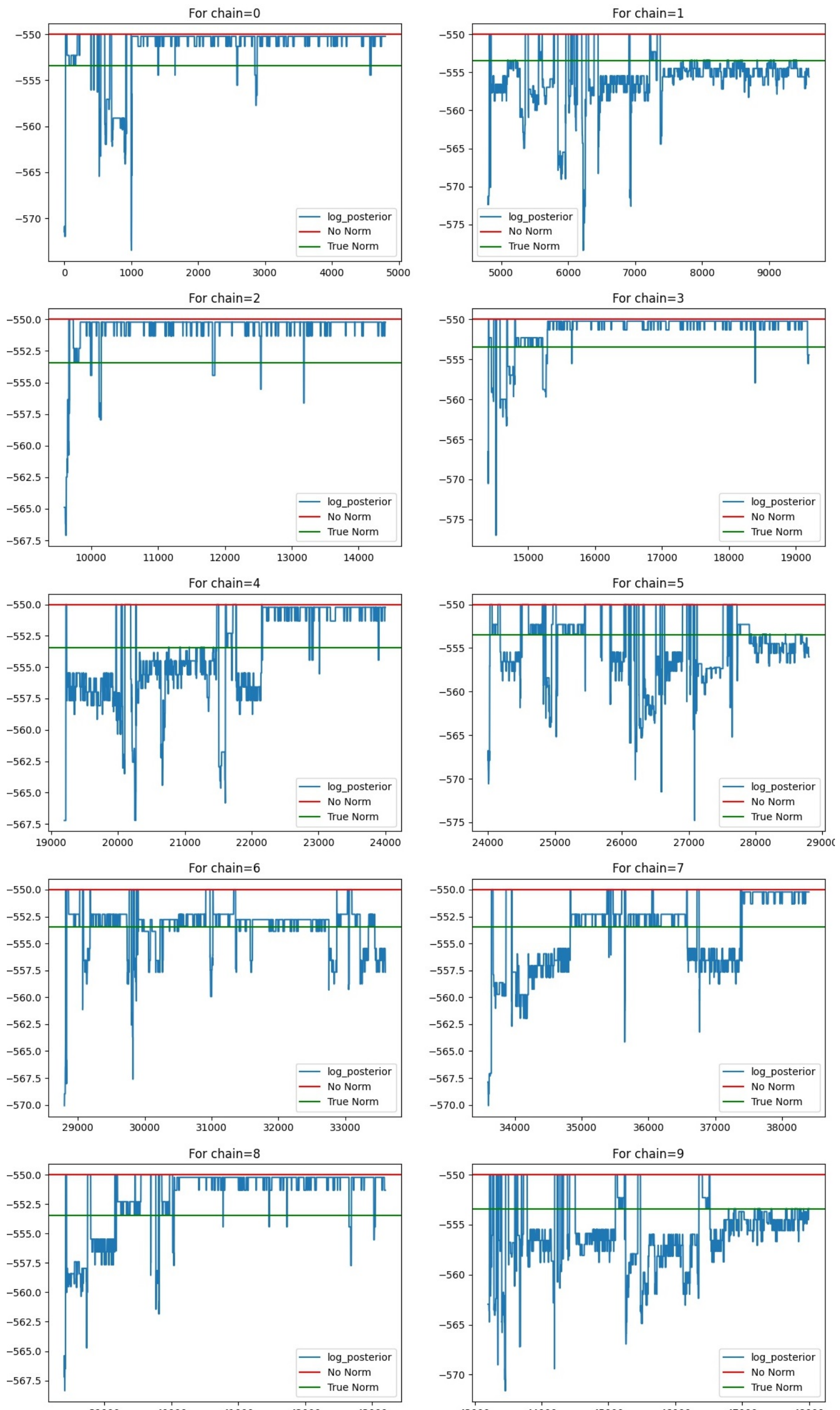

Figure 6: Chain plots for $p_{nn} = 0.55$