Further experiments and comparisons for PNS algorithms

Sigeng Chen, Jeffrey S. Rosenthal, Aki Dote, Hirotaka Tamura, Ali Sheikholeslami

April 29, 2023

1 QUBO Revisited: Choices for the Partial Neighbors

1.1 Systematic Or Random

In the QUBO question from the sampling paper, we naively used two systematically pre-selected neighbor sets $\mathcal{N}_0$, $\mathcal{N}_1$. However, there can be various ways to select the Partial Neighbor Sets, all of which, as long as the Partial Neighbor selected satisfies the PNS Convergence Theorem, will converge to the target density π. To start with, we compare two ways of choosing partial neighbor sets here: systematic and random. For simplicity, assume that we have N neighbors for all states, and we use partial neighbor sets of size n. Therefore, we have $\binom{N}{n}$ different partial neighbor sets. For systematic method, we choose $\mathcal{I}$ partial neighbor sets $\{\mathcal{N}_i\}_{i=1}^{\mathcal{I}}$, where $\cup_{i=1}^{\mathcal{I}}\mathcal{N}_i(x) = \mathcal{N}(x)$. We proceed with each Partial Neighbor Set for L_0 original samples and then move on to the next until we reach the $\mathcal{I}$-th one and then go back to the first one. We use the notation $\mathcal{N}_i(x)$ for systematic Partial Neighbor Sets because $\mathcal{N}_i(x)$ is pre-determined for $i = 1, 2, \ldots, \mathcal{I}$. On the other hand, for random Partial Neighbor Sets, we choose a new set $\mathcal{N}_k$ from all $\binom{N}{n}$ potential Partial Neighbor Sets after each L_0 original samples. We use the notation $\mathcal{N}_k(x)$ for random partial neighbor sets because $\mathcal{N}_k(x)$ can be different for every PNS step, and the subscript k represents the special partial neighbor set for step k. For both methods, $\mathcal{Q}_i(x, y), \mathcal{Q}_k(x, y) \propto \mathcal{Q}(x, y)$ for $y \in \mathcal{N}_i(x)$, and $\mathcal{Q}_i(x, y) = \mathcal{Q}_k(x, y) = 0$ otherwise.

To compare the above two methods of selecting Partial Neighbor Sets, we apply them to the previous 16×16 QUBO question, and we test the following four scenarios:

1. two systematic partial neighbor sets where the first set considers flipping the first half of the bits, and the second set considers flipping the second half of the bits;
2. four systematic partial neighbor sets where each set considers flipping a quarter of the bits;
3. random partial neighbor sets with $\frac{N}{2}$ partial neighbors; that is, each set considers flipping a random set of bits with size $\frac{N}{2}$;
4. random partial neighbor sets with $\frac{N}{4}$ partial neighbors; that is, each set considers flipping a random set of bits with size $\frac{N}{4}$;

The result is shown in Figure 1; for this case, systematic Partial Neighbor Sets are better than random Partial Neighbor Sets. However, random Partial Neighbor Sets can be better when we run the same code with a different random seed. After running this simulation for 100 different random seeds, the systematic neighbor sets are better 56 times. Thus, we conclude that the performance of these two Partial Neighbor Sets is close to each other. We will continue using the systematic Partial Neighbor Sets in our later simulation.

1.2 The choice of the Partial Neighbor Sets sizes

In previous simulations, we naively use we used $|\mathcal{N}_i(x)| = 4$ or 8 and $L_0 = 100$ in previous examples. What is the optimal choice for $|\mathcal{N}_i(x)|$? We want to compare $|\mathcal{N}_i(x)| = 2, 4, 6, 8, \ldots, 14$ by the QUBO

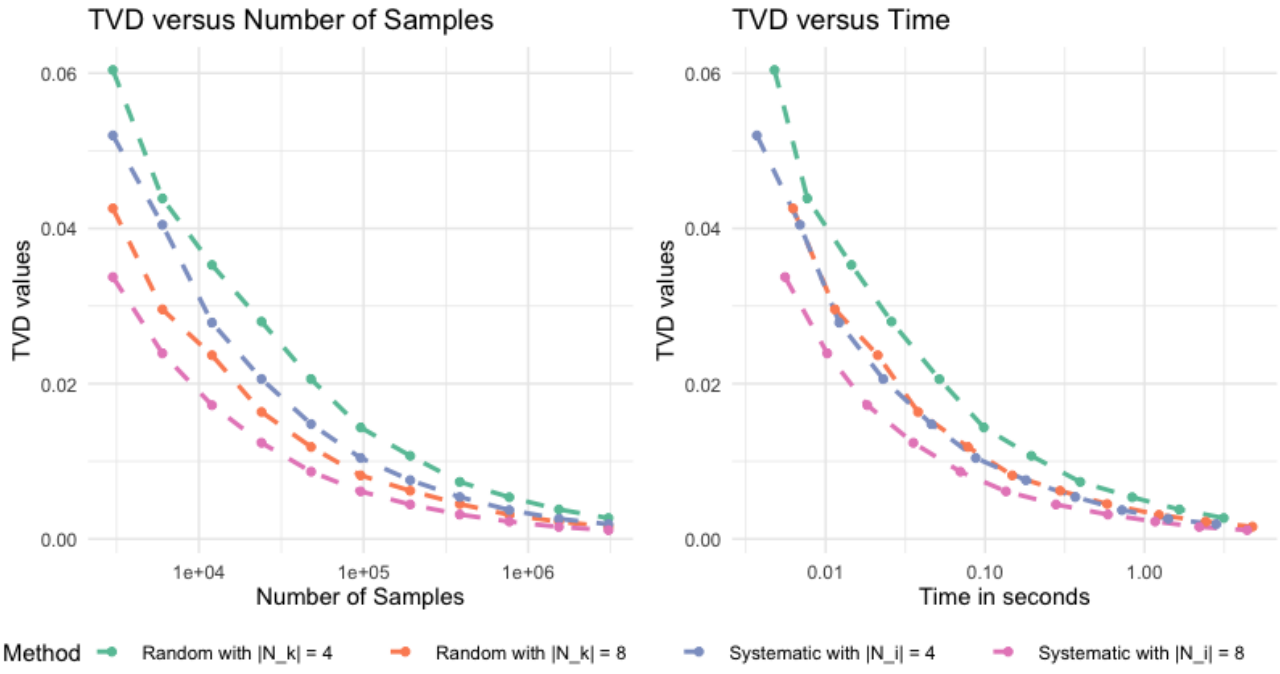

Figure 1: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for four scenarios: Systematic PNS and Random PNS, each with Partial Neighbor Set sizes of 4 and 8. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. For all PNS, we used $L_0 = 100$.*

question. In previous cases, we only used the systematic Partial Neighbor Set size n that can be divided evenly by N. For other n's that cannot be divided evenly such as 14, we create the Partial Neighbor Sets in loops to make the sets have the same size and include all entries for the same amount of time. For example, we create the following 8 Partial Neighbor Sets for $|\mathcal{N}_i(x)| = 14$:

- $\mathcal{N}_1(x) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$; that is, we consider flipping entries 1 to 14 for $\mathcal{N}_1(x)$;
- $\mathcal{N}_2(x) = \{15, 16, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$;
- $\mathcal{N}_3(x) = \{13, 14, 15, 16, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$;
- $\mathcal{N}_4(x) = \{11, 12, 13, 14, 15, 16, 1, 2, 3, 4, 5, 6, 7, 8\}$;
- $\mathcal{N}_5(x) = \{9, 10, 11, 12, 13, 14, 15, 16, 1, 2, 3, 4, 5, 6\}$;
- $\mathcal{N}_6(x) = \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1, 2, 3, 4\}$;
- $\mathcal{N}_7(x) = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1, 2\}$;
- $\mathcal{N}_8(x) = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$;

Figure 2 shows the results for comparing the Unbiased PNS with $|\mathcal{N}_i| \equiv 2$, 4, 6, 8, 10, 12, and 14 for $\forall X \in \{0, 1\}^{16}$. Every other simulation setting is the same as the previous simulations for the QUBO question. The choice of L_0 is still 100. According to the left plot, we can say that given the same amount of original samples, the Markov chain from $|\mathcal{N}_i| = 14$ is the least biased. On the other hand, from the right plot, we can conclude that, given the same amount of CPU time, the sample quality from $|\mathcal{N}_i| = 14$ is the best. In addition, the performances are close to each other for all cases where $|\mathcal{N}_i| \geq 8$. Note that a single-core implementation makes all these comparisons by the CPU time, and parallelism hardware can provide speedups. Intuitively, the more tasks that can be calculated simultaneously, the greater the

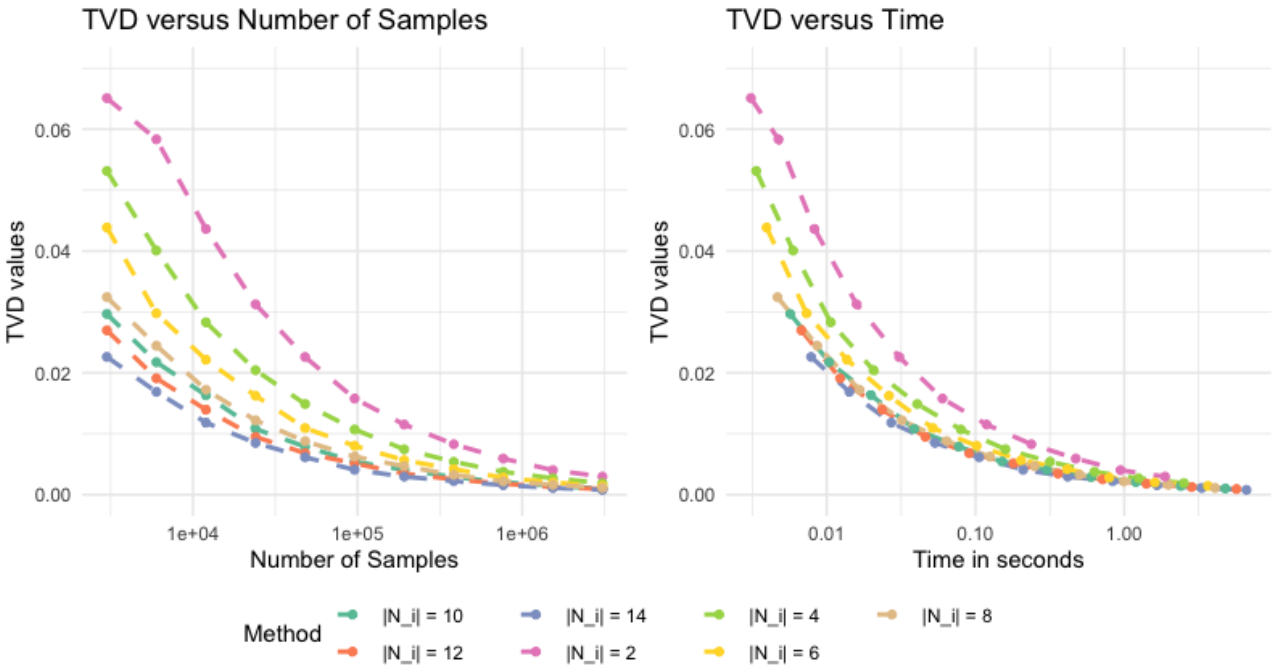

Figure 2: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for four scenarios: Unbiased PNS with different partial neighbor set sizes {2, 4, 6, ..., 14}. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. For all PNS, we used $L_0 = 100$.*

speedup. Thus, if we apply our Unbiased PNS on parallelism hardware with a limited number of parallel tasks that can be computed simultaneously, we should choose the largest possible partial neighbor set size $|\mathcal{N}_i|$.

1.3 The choice of L_0

Furthermore, Figure 3 shows the results for comparing the Unbiased PNS with $L_0 = 10$, 50, 100, 500, and 1000. Again, every other setting of the simulation is the same, and $|\mathcal{N}_i|$ is still 8, $\forall x \in \{0,1\}^{16}$. The left plot shows that given the same samples, the Markov chain from $L_0 = 10$ is the least biased. However, the right plot shows that, given the same amount of CPU time, the TVD values are about the same except $L_0 = 10$. The case with $L_0 = 100$ is slightly better than the other cases, but the difference is not too large. $L_0 = 10$ becomes the worst since such L_0 has too many rejections (about one rejection for every ten samples). Thus, for a single-core implementation, the choice of L_0 is not that important as long as it is not extreme. In addition, for parallelism hardware, when we change the partial neighbors being chosen, we have to bring the new neighbors to the memory. This can be a waste of time if we are doing this very frequently. Thus, L_0 should not be too small for parallel computing.

1.4 The choice of $\mathbf{L_0}$ when the Partial Neighbor Sets have different sizes

In the previous section, we found the best choices for L_0 and Partial Neighbor Sets sizes by controlling the other parameter, and we also used Partial Neighbor Sets of the same size. Here, we try to explore the sampling efficiency with Partial Neighbor Sets of different sizes, and the best choice of the corresponding $\mathbf{L_0}$. Since Partial Neighbor Sets have different sizes, $\mathbf{L_0}$ is a vector with a size equal to the total number of all Partial Neighbor Sets. We only use two Partial Neighbor Sets $\mathcal{N}_1$ and $\mathcal{N}_2$ here. where $|\mathcal{N}_1| = n$, and $|\mathcal{N}_2| = N - n$. We can choose different $\mathbf{L_0}$ for different Partial Neighbor Sets. For example, we first let $n = 8$, then the $|\mathcal{N}_1| = |\mathcal{N}_2| = 8$, and we check the sampling efficiency when $\mathbf{L_0}(\mathcal{N}_1) = 100$ and

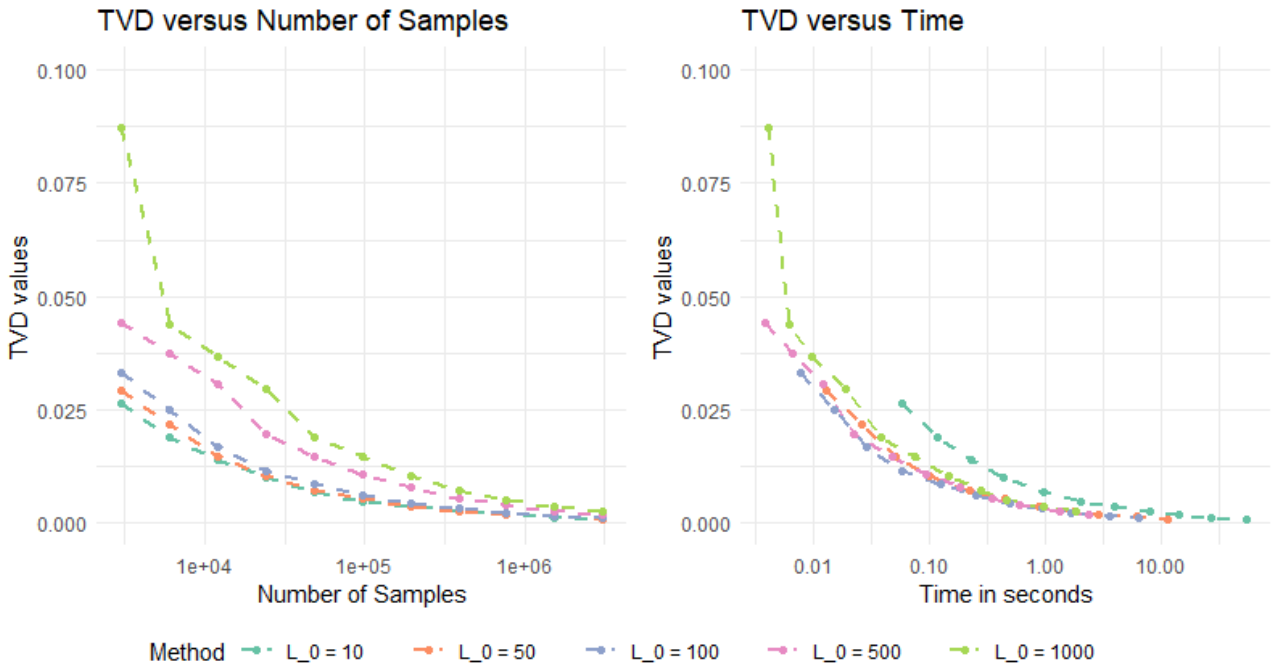

Figure 3: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased Partial Neighbor Search with different sizes of L_0. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{300, 600, 1200, 2400, \ldots, 3072000\}$. For all PNS, we used $|\mathcal{N}_i| = 8$.*

$\mathbf{L_0}(\mathcal{N}_2) = 100$. We also check the combinations for $\mathbf{L_0}$ like $\{100, 10\}$, $\{100, 50\}$, $\{200, 100\}$, etc. The result is shown in Figure 4. Similarly, we use $n = 4$, so $|\mathcal{N}_1| = 4$ and $|\mathcal{N}_2| = 12$, and do the same simulation again. The results are shown in Figure 5.

First of all, Figure 5 confirms that the Markov chain will converge to the correct stationarity given that the sizes of the Partial Neighbor Sets are different. In addition, by Figure 4 and Figure 5, the best strategies are both using $L_0 = 100$ and 500 accordingly for $\mathcal{N}_1$ and $\mathcal{N}_2$. However, logically, this is not true for all QUBO questions, and this result only occurs because of the randomness of the QUBO matrix Q.

To eliminate the influence from the randomness of the QUBO matrix Q, we create one thousand QUBO matrices Q. For each of these QUBO matrices Q, we compare the methods of choosing L_0 by 100 runs, and we average the results from these 1000×100 simulations to get the average TVD values as well as the average time in seconds used by the program. Since this simulation has much more simulation runs than before, we use an 8×8 QUBO matrix here, and the sizes for Partial Neighbor Sets are 2 and 6. The result is shown in Figure 6, and we can conclude that when comparing the TVD values by time in seconds, the choice of L_0 does not matter too much even when the sizes for the Partial Neighbor Sets are different.

1.5 The choice of Partial Neighbor Sets sizes given $\mathbf{L_0}$

We compared the sampling efficiency with different combinations for $\mathbf{L_0}$ when the Partial Neighbor Sets have different sizes in Section 1.4. On the other hand, if we have a vector $\mathbf{L_0}$ for different numbers of samples with respect to the original chain, what will be the best choice for the Partial Neighbor Set sizes? For both $\mathbf{L_0} = \{100, 100\}$ and $\mathbf{L_0} = \{100, 300\}$, we check the sampling efficiency for different combinations of the sizes for two Partial Neighbor Sets $\mathcal{N}_1$ and $\mathcal{N}_2$. The result is shown in Figure 7. In addition, the results for $L_0 = \{100, 300\}$ is shown in Figure 8.

From Figure 7 and Figure 8, the results are similar to what we had in Section 1.4. We conclude that, for given sizes L_0, the choice of PNS sets does not make a big difference. Similar to Section 1.4, we can

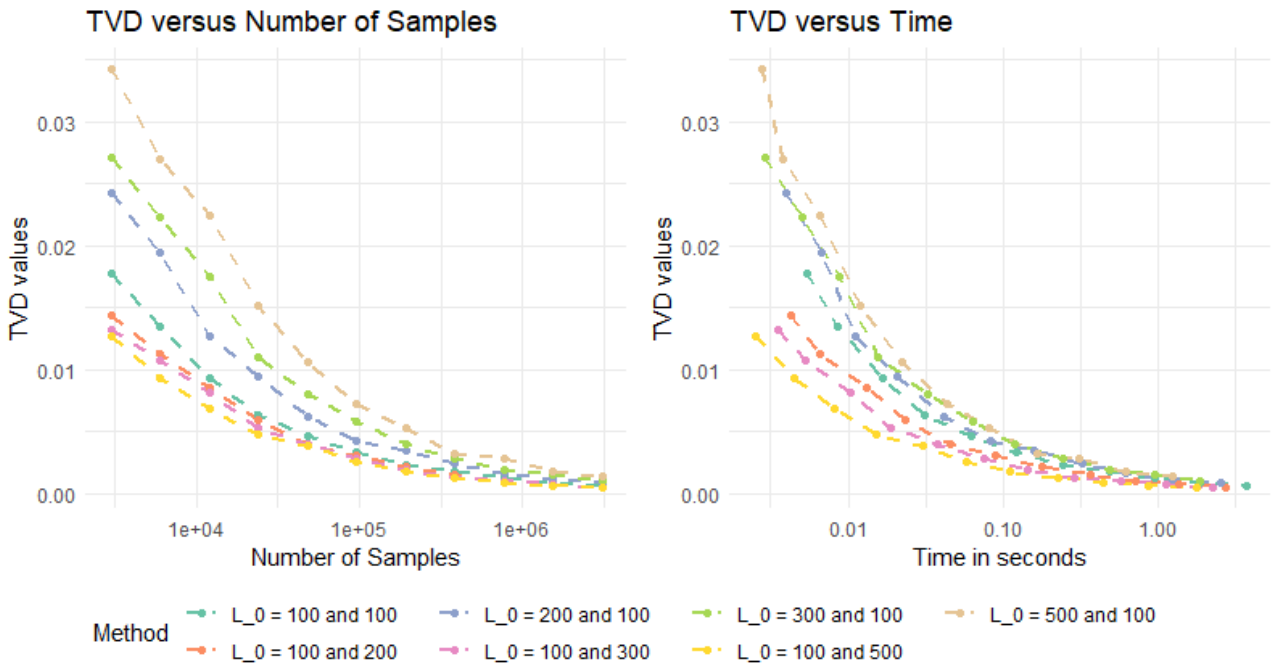

Figure 4: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with different L_0 values chosen for two partial neighbor sets. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 300 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. For all PNS, we used two partial neighbor sets, where each of them consider flipping 8 entries.*

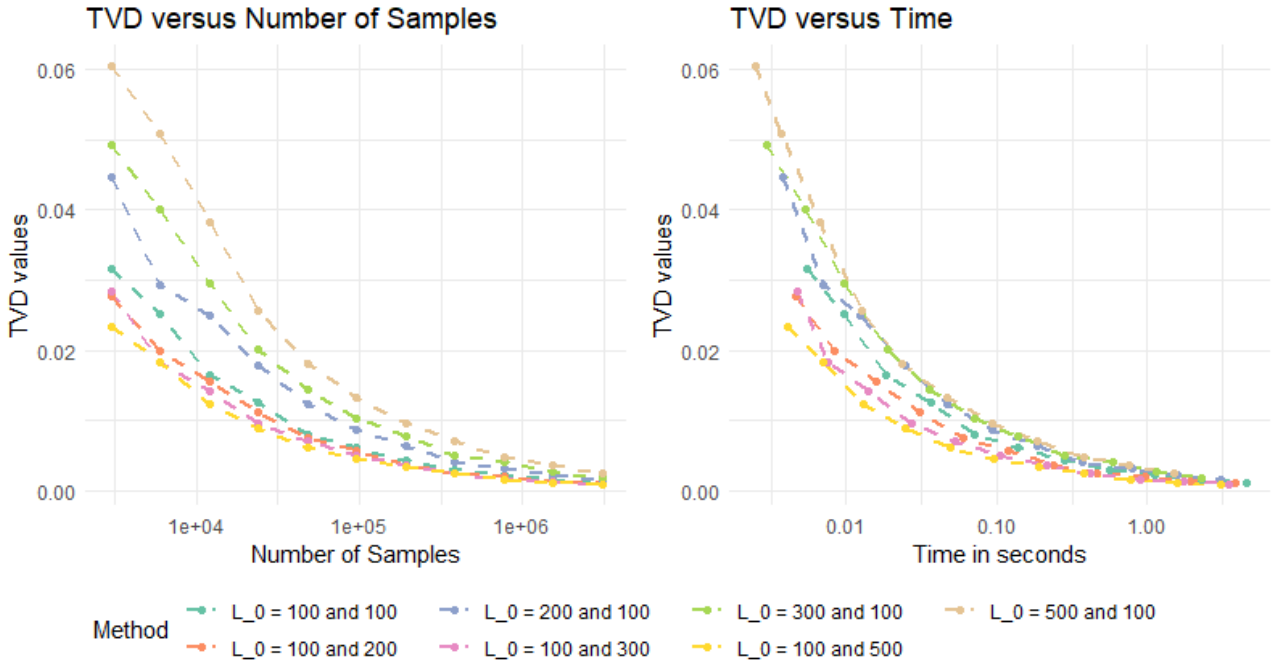

Figure 5: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with the sizes for Partial Neighbor Sets being 8 and 8. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 300 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$.*

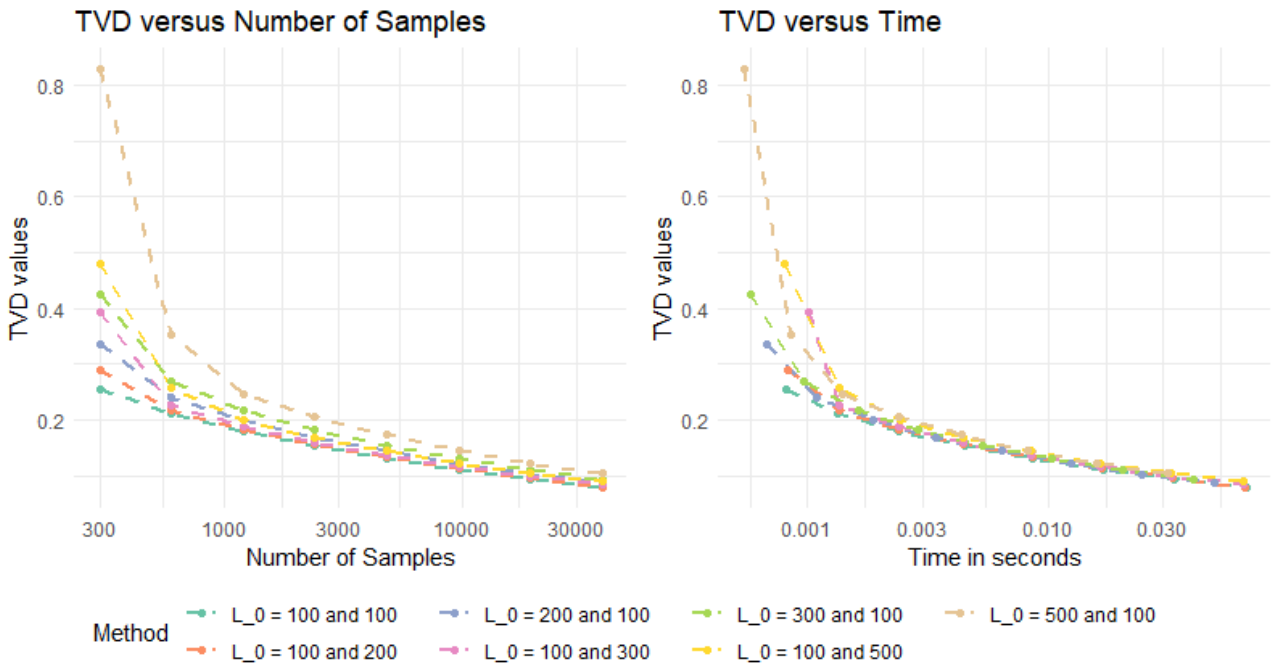

Figure 6: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with the sizes for Partial Neighbor Sets being 2 and 6. Random upper triangular 8×8 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 different QUBO matrix Q, and for each QUBO matrix Q, we did 100 simulation runs given a certain original sample size, where the sizes are* $\{3000, 6000, 12000, 24000, 48000, 96000\}$.

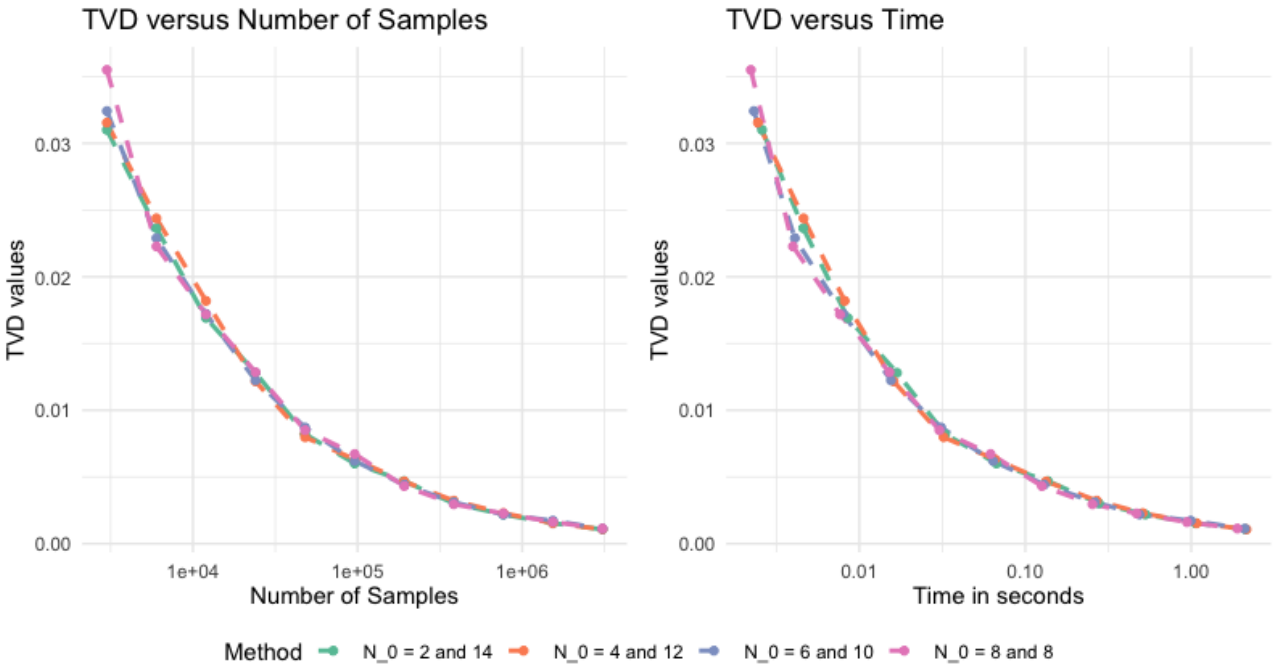

Figure 7: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with* $\mathbf{L_0} = \{100, 100\}$. *Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are* $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$.

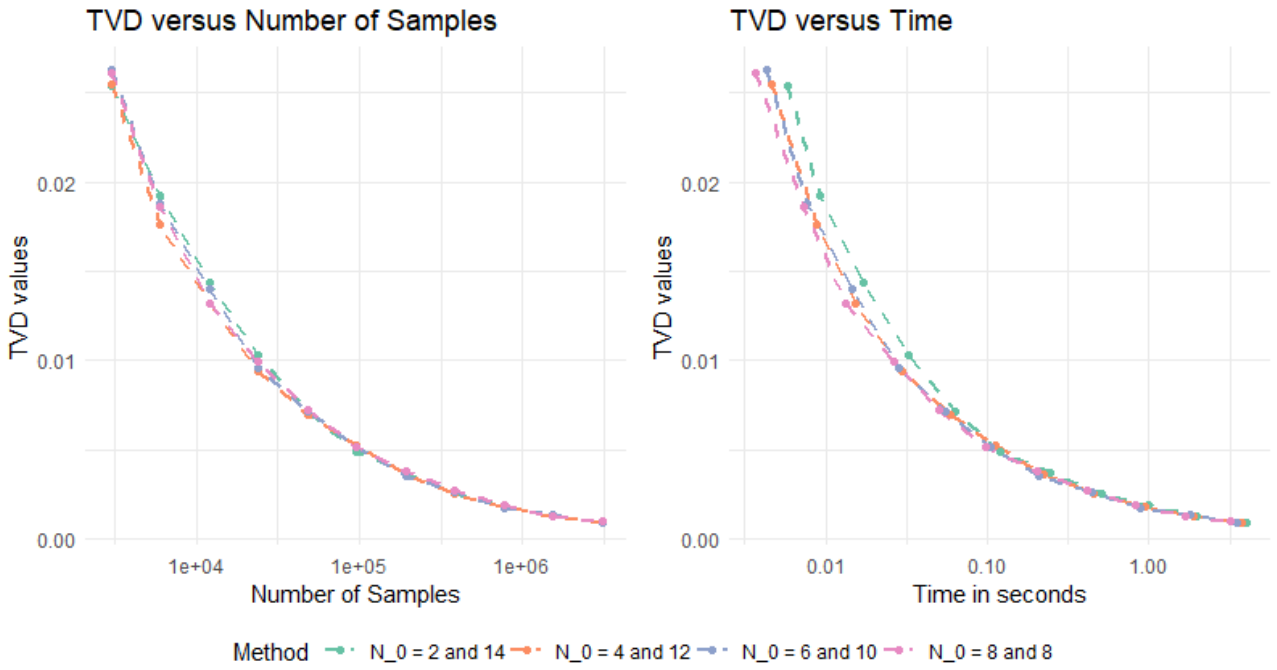

Figure 8: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with* $\mathbf{L_0} = \{100, 300\}$. *Random upper triangular* 16×16 *QUBO matrix is generated randomly by* $Q_{i,j} \sim N(0, 10^2)$ *for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are* $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$.

also try to eliminate the influence of the QUBO matrix Q. Again, we create one thousand QUBO matrices Q, and for each of these QUBO matrices Q, we compare the methods of choosing Partial Neighbor Sets by 100 runs. We average the results from these 1000×100 simulations to get the average TVD values as well as the average time in seconds used by the program. Again, we use an 8×8 QUBO matrix here, and the size for L_0 is 100 for both Partial Neighbor Sets. The result is shown in Figure 9, and we can conclude that when comparing the TVD values by time in seconds, the choice of Partial Neighbor Sets does not matter too as well.

1.6 Two Flips

In previous simulations, we used uniform proposal distributions among all neighbors where the neighbors are defined as binary vectors with Hamming distance 1. That is, $\mathcal{Q}(x, y) = \frac{1}{N}$ for $\forall y$ such that $|x - y| = \sum_{i=1}^{N} |x_i - y_i| = 1$, $\forall x, y \in \{0, 1\}^N$. Thus, the neighbors are all binary vectors different by one flip. We usually did not include two flips because of two reasons. First of all, the acceptance rate by the Metropolis-Hastings algorithm on two flips is low, since two flips usually have larger energy differences than one flip. In addition, there are too many of them. For example, for a 16×16 QUBO question, if we only consider one flip, then for each state, there are 16 neighbors. However, considering two flips, there will be 256 neighbors. Then for Rejection-Free, the number of neighbors is too many to be applied efficiently.

However, with the help of PNS, we can now try to include two flips into the Partial Neighbor Sets. For the first simulation, we use the same settings as before, except we choose the following proposal distribution and neighbors:

1. One Flip with four neighbors chosen randomly. That is, for each step, we randomly choose four entries from 1 to 16, and apply Partial Neighbor Search on these four neighbors.

2. One Flip with eight neighbors chosen randomly.

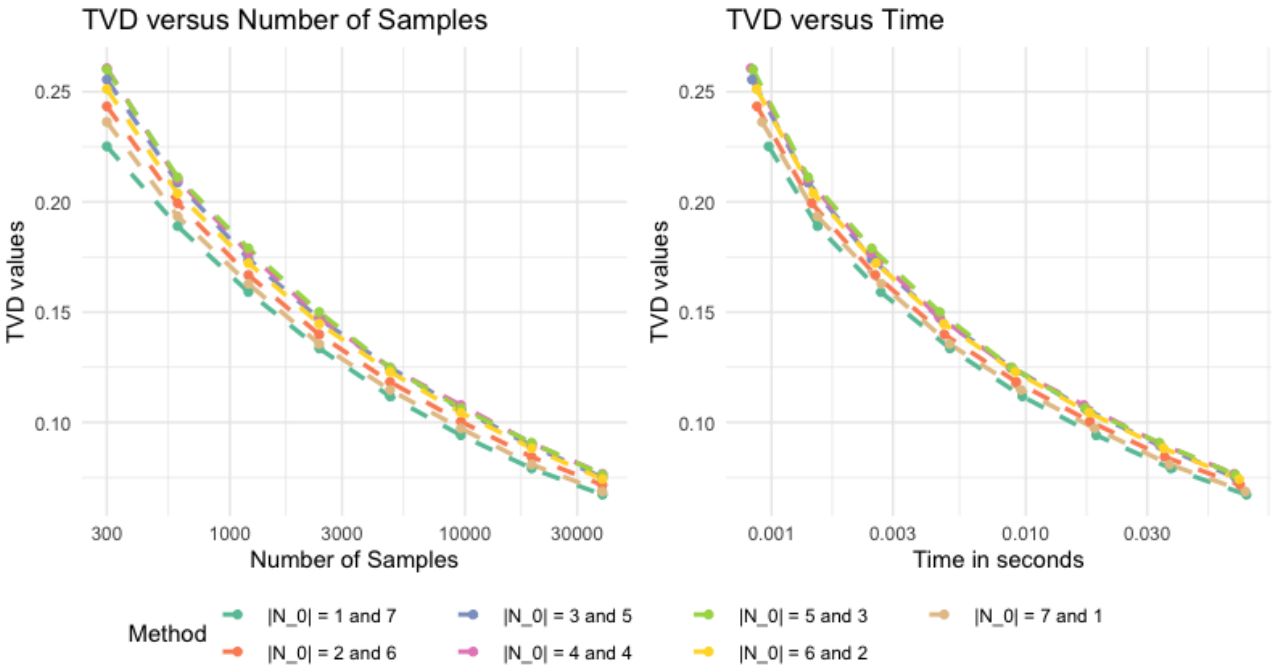

Figure 9: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with $L_0 = 100$ for both Partial Neighbor Sets. Random upper triangular 8×8 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 different QUBO matrix Q, and for each QUBO matrix Q, we did 100 simulation runs given a certain original sample size, where the sizes are $\{300, 600, 1200, 2400, 4800, 9600, 19200, 38400\}$.*

3. Two Flip with four neighbors chosen randomly. We first randomly choose eight entries from 1 to 16, and apply Partial Neighbor Search on these four neighbors, to make one flip with four neighbors first for L_0 original samples. Then we randomly choose four groups of two different entries from 1 to 16 and apply Partial Neighbor Search on these four groups of two flips.

4. Two Flip with eight neighbors chosen randomly.

Note that, we can only apply the pure two flips since the Markov chain will become reducible then, so we mix the one flip and two flips to make a hybrid chain. The results for the simulations are shown in Figure 10. Note that, the mean for the multiplicity list M_k (before considering L_0) is 7×10^{13}, which is super large. This is understandable since the energy difference for two flips is large. As a result, the two flip algorithms sample less efficiently than one flip. In addition, we did another simulation by generating $Q_{i,j} \sim N(0, 0.4^2)$ instead of using the standard deviation of 10. The simulation result is shown in Figure 11. It seems that even if the energy difference for two flips is not that large, two flips are still a little bit worse than one flip. Technically, we think two flips can be very helpful when escaping from the local mod, but in our simulation, we didn't find it to be more efficient.

In addition to the previous comparison, we have more ways to apply two flips. We can define the neighbors to state with Hamming distance less or equal to 2. That is, $\mathcal{Q}(x, y) = \frac{1}{N^2}$ for $\forall y$ such that $|x - y| = \sum_{i=1}^{N} |x_i - y_i| \leq 2$, $\forall x, y \in \{0, 1\}^N$. Then we choose $|\mathcal{N}_i|$ neighbors as Partial Neighbor Set randomly from all N^2 neighbors. The result of the comparison is shown in Figure 12. Again, the two flips are worse than one flip. This is easy to understand as well since most of the two flips have higher energy differences, and thus the two flips are not making as much contribution as one flip when being included in the Partial Neighbor Sets. Moreover, we can try to adjust the proposal distribution $\mathcal{Q}$ to make our Partial Neighbor Sets consider one flip more. We choose $\mathcal{Q}(x, y) = \frac{1}{2N}$ for $|x - y| = \sum_{i=1}^{N} |x_i - y_i| = 1$ and $\mathcal{Q}(x, y) = \frac{1}{2(N^2 - N)}$ for $|x - y| = \sum_{i=1}^{N} |x_i - y_i| = 2$, and the result is shown in Figure 13. The difference

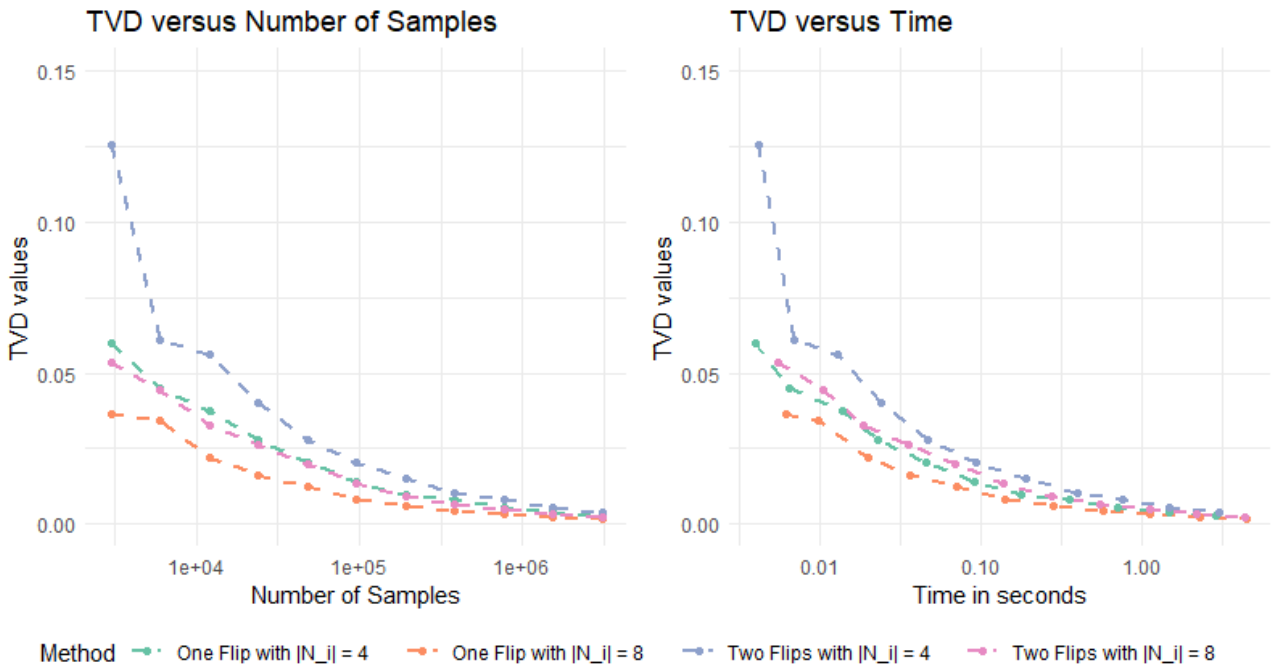

Figure 10: *Two-flip: Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with different selections of neighbor sets. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. $L_0 = 100$.*

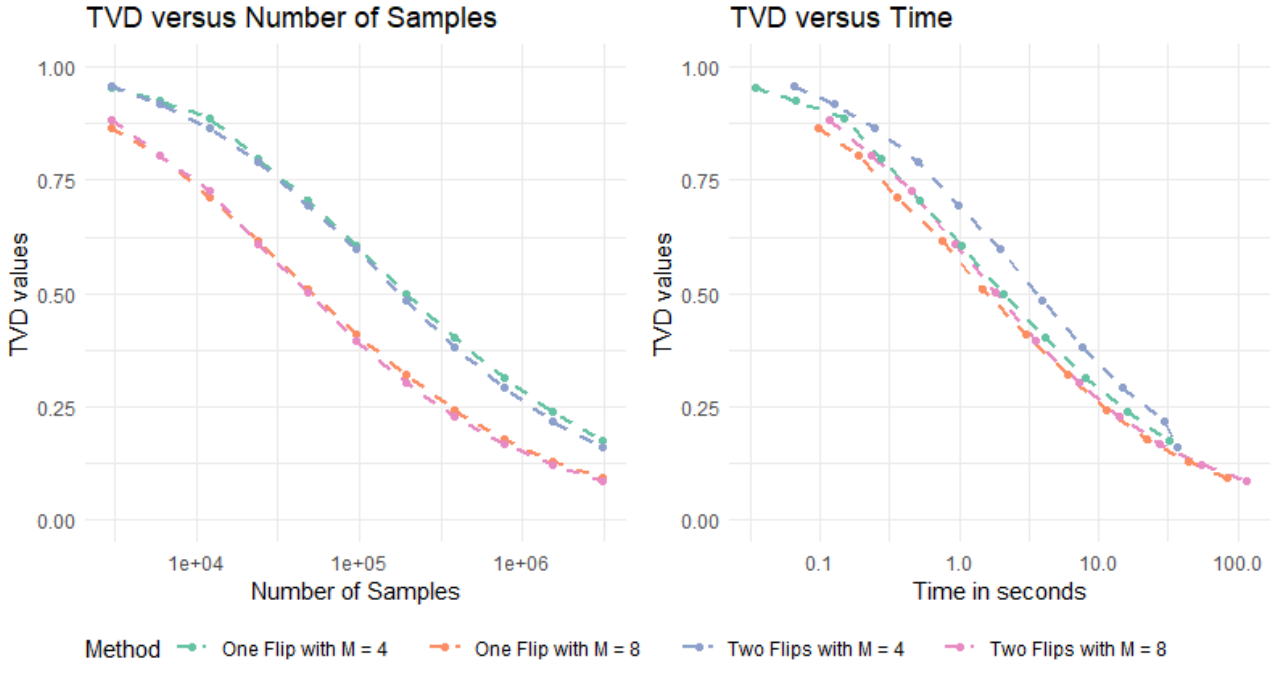

Figure 11: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with different selections of neighbor sets: one flip and two flips. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 0.4^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. $L_0 = 100$.*

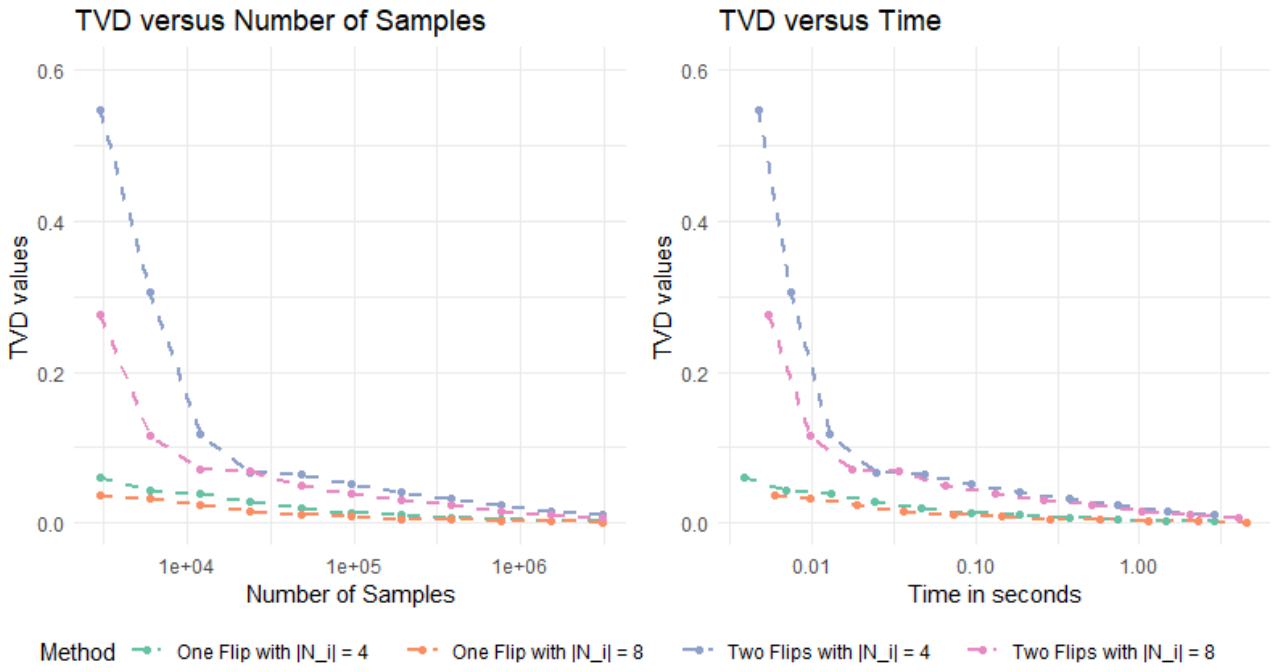

Figure 12: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with different selections of neighbor sets: one flip and two flips. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 0.4^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. $L_0 = 100$.*

between one flip and two flips is not as much as before, but two flips are still worse than one flip. Thus, here we conclude that for the situation we have here, the two flips may not be that useful.

2 Use the information of QUBO matrix

We already talked about using different proposal distributions other than a uniform proposal distribution on all neighbors in Section 1.6. We believe that uniform proposal distribution can be a good choice generally, but when we are given some extra information about the sampling question, there must exist some other efficient ways to create better proposal distribution. For example, for the QUBO question, maybe we can make use of the information of the QUBO matrix Q to get a better proposal distribution. First, we define a parameter weight w_i for each entry of x_i to be:

$$w_i \propto \sum_{j=1}^{N} [|Q_{i,j}| + |Q_{j,i}|] - |Q_{i,i}| \tag{1}$$

Then, based on the vector $\mathbf{w}$, we consider four types of Modified PNS

1. $\mathcal{Q}(X, Y) \propto \frac{1}{w_i}$ if $|X_i - Y_i| = 1$
2. $\mathcal{Q}(X, Y) \propto \frac{1}{w_i^2}$ if $|X_i - Y_i| = 1$
3. $\mathcal{Q}(X, Y) \propto w_i$ if $|X_i - Y_i| = 1$
4. $\mathcal{Q}(X, Y) \propto w_i^2$ if $|X_i - Y_i| = 1$

The result of the simulation is shown in Figure 14. From the top two figures, we can see that Modified 3 and Modified 4 are less efficient. To see the difference between regular PNS, Modified 1, and Modified

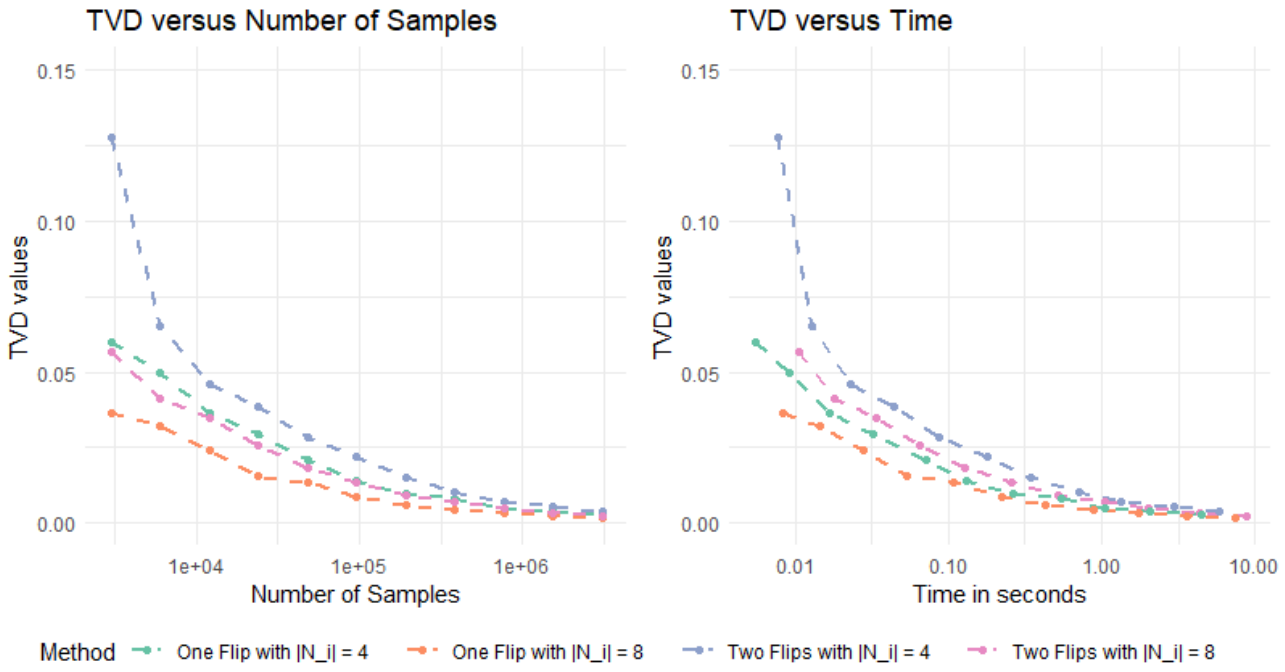

Figure 13: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with different selections of neighbor sets: one flip and two flips. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 0.4^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 1000 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$. $L_0 = 100$.*

2, we make two more plots for them on a smaller scale as the bottom two figures. From these two figures, we can see that, for this circumstance, the Modified PNS with proposal distribution being propositional to the inverse of the weights is better than the regular PNS. This is understandable, since usually for sampling questions, we care more about the states around the highest density values. For example, if we are sampling from space with only four density values $\{100, 99, 0.1, 0.09\}$, then the two small states do not matter too much. To get better samples, we probably want our Markov chain to be switching between state 100 and 99. A similar thing can happen here, for this QUBO question, we want our Markov chain to switch among states with large density values, and those large density values states are different by the flips with less weight. Therefore, the Modified versions of PNS can help.

Although the Modified PNS is better when we used a randomly generated QUBO matrix Q, we doubt that there can be some other problems for some special Q. For example, we define the following 4 by 4 QUBO matrix Q:

$$\begin{pmatrix} 2.5 & 3 & -5.0 & -6.0 \\ 0.0 & 4 & -6.0 & -7.0 \\ 0.0 & 0 & 5.5 & 6.5 \\ 0.0 & 0 & 0.0 & 7.0 \end{pmatrix} \tag{2}$$

For target density from this QUBO matrix, there are two separate local maximum $(1, 1, 0, 0)^T$ and $(0, 0, 1, 1)^T$. Then, in this case, we probably don't want to be stuck at one local maximum and thus Modified 1 and 2 may not be that good. Note that, the weights are $\{0.4232, 0.461, 0.077, 0.038\}$ accordingly.

The result of the simulation is shown in Figure 15. From the figures, Modified 1 and 2 are a little bit worse than the regular PNS. This shows that the Modified version of the PNS is not always better than the regular PNS. However, we still believe that the Modified PNS can be useful in most unimodal cases.

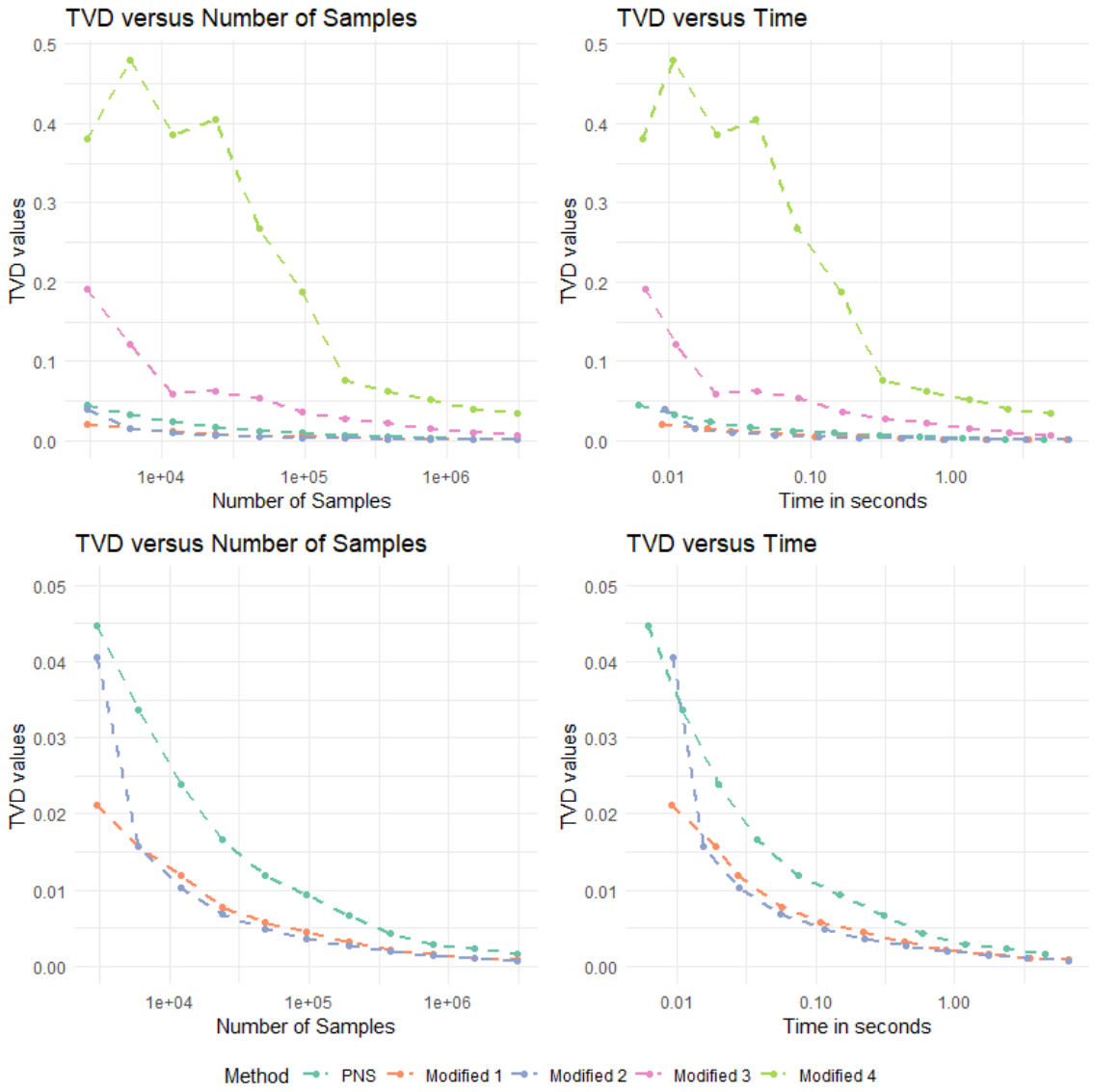

Figure 14: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with and without using the information of matrix Q. PNS Modified means proposal distribution based on the QUBO matrix Q as $w_i \propto \sum_{j=1}^{N}[|Q_{i,j}| + |Q_{j,i}|] - |Q_{i,i}|$. The upper two plots are the same as the bottom two plots except for the scales. Modified 1: $\mathcal{Q}(X,Y) \propto \frac{1}{w_i}$ if $|X_i - Y_i| = 1$; Modified 2: $\mathcal{Q}(X,Y) \propto \frac{1}{w_i^2}$ if $|X_i - Y_i| = 1$; Modified 3 : $\mathcal{Q}(X,Y) \propto w_i$ if $|X_i - Y_i| = 1$; Modified 4: $\mathcal{Q}(X,Y) \propto w_i^2$ if $|X_i - Y_i| = 1$. Random upper triangular 16×16 QUBO matrix is generated randomly by $Q_{i,j} \sim N(0, 10^2)$ for upper triangular elements. Each dot within the plot represents the average TVD value and time used for 100 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$.*

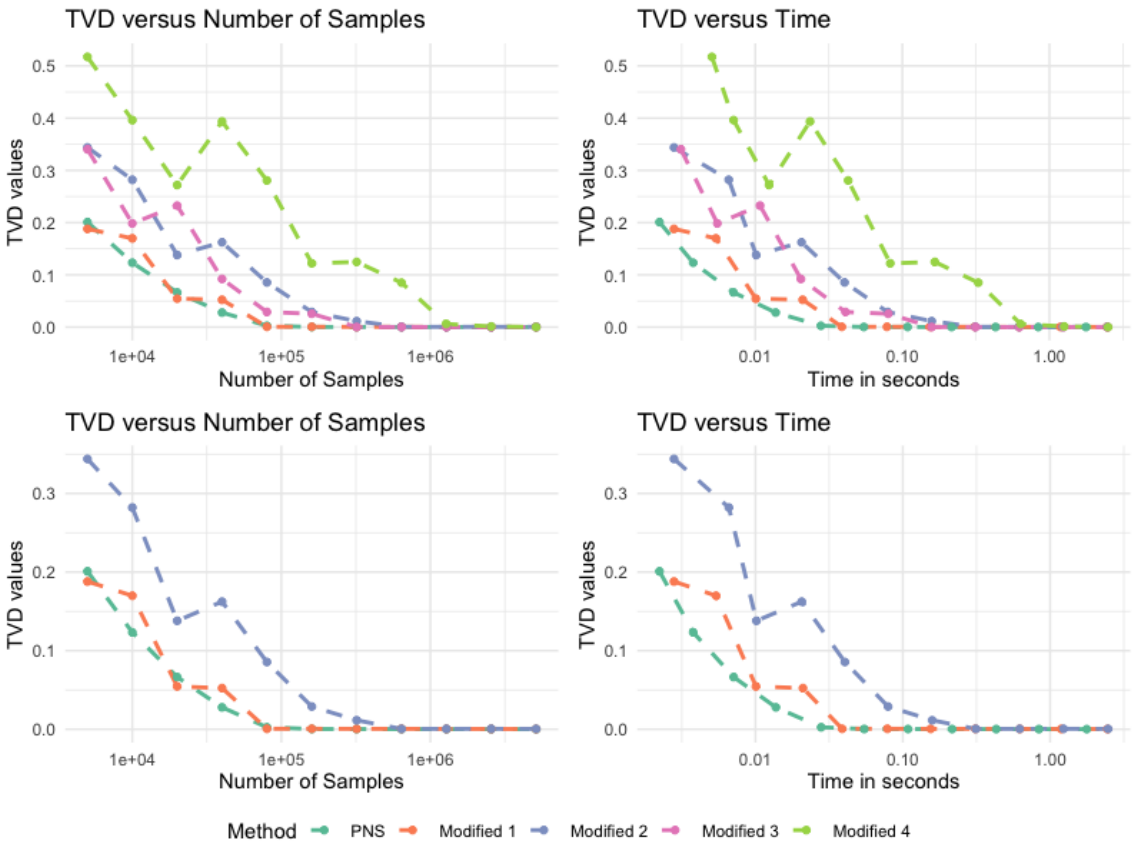

Figure 15: *Average values of TVD between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for Unbiased PNS with and without using the information of matrix Q. PNS Modified means proposal distribution based on the QUBO matrix Q as $w_i \propto \sum_{j=1}^{N}[|Q_{i,j}|+|Q_{j,i}|]-|Q_{i,i}|$. The upper two plots are the same as the bottom two plots except for the scales. Modified 1: $\mathcal{Q}(X,Y) \propto \frac{1}{w_i}$ if $|X_i - Y_i| = 1$; Modified 2: $\mathcal{Q}(X,Y) \propto \frac{1}{w_i^2}$ if $|X_i - Y_i| = 1$; Modified 3 : $\mathcal{Q}(X,Y) \propto w_i$ if $|X_i - Y_i| = 1$; Modified 4: $\mathcal{Q}(X,Y) \propto w_i^2$ if $|X_i - Y_i| = 1$. Random upper triangular 4×4 given QUBO matrix. Each dot within the plot represents the average TVD value and time used for 100 simulation runs given a certain original sample size, where the sizes are $\{3000, 6000, 12000, 24000, \ldots, 3072000\}$.*

3 Irreversible Assumptions for Partial Neighbor Sets

In the Donut example from the sampling paper, we compared PNS with the Metropolis-Hastings algorithm by a two-dimensional donut example. In that example, we used a reversible proposal. That is, for Partial Neighbor Set $\mathcal{N}_i$, $y \in \mathcal{N}_i(x) \iff x \in \mathcal{N}_i(y)$, $\forall x, y \in \mathcal{S}$. Thus, we chose the Partial neighbor Set $\mathcal{N}_i(x)$ as follows:

1. generate $\delta_1, \delta_2 \sim \text{Normal}(0, 1)$;
2. for state $x = (x_1, x_2)$, put $y = (x_1 + \delta_1, x_2 + \delta_2)$ into the Partial Neighbor Set $\mathcal{N}_i(x)$;
3. to ensure the reversibility, also put $y' = (x_1 - \delta_1, x_2 - \delta_2)$ into the Partial Neighbor Set $\mathcal{N}_i(x)$;
4. repeats the above steps 25 times to generate a total of 50 neighbors for the Partial Neighbor Set $\mathcal{N}_i(x)$.

We can prove that this reversible PNS chain will converge to the target density correctly. However, is reversibility necessary for the Partial Neighbor Set?

We use a four-dimension of donuts example to show the results from a PNS chain with irreversible Partial Neighbor Sets. Suppose we have four independent random variables μ, θ_1, θ_2, θ_3 where

$$\mu \sim \text{Normal}^+(\mu_0, \sigma^2),\ \theta_1, \theta_2, \theta_3 \sim \text{Uniform}[0, \pi). \tag{3}$$

Here, Normal^+ means the Truncated Normal distribution without the negative tail, and π in the Uniform distribution means the circular constant instead of the target density. Then we define two random variables X_1 and X_2 to be

$$\begin{aligned} X_1 &= \sqrt{\mu}\sin\theta_1\sin\theta_2,\ X_2 = \sqrt{\mu}\sin\theta_1\cos\theta_2, \\ X_3 &= \sqrt{\mu}\cos\theta_1\sin\theta_3,\ X_4 = \sqrt{\mu}\cos\theta_1\cos\theta_3. \end{aligned} \tag{4}$$

Thus we have

$$f_{X_1,X_2,X_3,X_4}(x_1, x_2, x_3, x_4) \propto \frac{1}{\sigma}\exp\left[-\frac{(x_1^2 + x_2^2 + x^3 + x^4 - \mu_0)^2}{2\sigma^2}\right], \tag{5}$$

In addition, the proposal distribution for the Metropolis-Hastings algorithm is defined to be the standard normal distribution for both dimensions. That is, for $x = (x_1, x_2, x_3, x_4)$, $y = (y_1, y_2, y_3, y_4) \in \mathbb{R}^4$, $\mathcal{Q}(x, y) = \phi(y_1 - x_1)\phi(y_2 - x_2)\phi(y_3 - x_3)\phi(y_4 - x_4)$ where ϕ is the density function of the standard normal distribution. Then for any $x \in \mathbb{R}^4$, we have $\mathcal{N}(x) = \mathbb{R}^4$.

For reversible Partial Neighbor Sets, we must have reversibility for all $\mathcal{N}_i(x)$, which means $y \in \mathcal{N}_i(x) \iff x \in \mathcal{N}_i(y)$, $\forall x, y \in \mathcal{S}$. Thus, we consider $50 + 50$ neighbors for each reversible Partial Neighbor Set $\mathcal{N}_i(x)$, and we pick our neighbors by repeating the following steps for 50 times:

1. generates $\delta_1, \delta_2, \delta_3, \delta_4 \sim \text{Normal}(0, 1)$
2. for state $x = (x_1, x_2, x_3, x_4)$, put $y = (x_1 + \delta_1, x_2 + \delta_2, x_3 + \delta_3, x_4 + \delta_4)$ into the Partial Neighbor Set $\mathcal{N}_i(x)$;
3. to ensure the reversibility, also put $y' = (x_1 - \delta_1, x_2 - \delta_2, x_3 - \delta_3, x_4 - \delta_4)$ into the Partial Neighbor Set $\mathcal{N}_i(x)$;

After repeating the above steps for 50 times, we can get a total of $50 + 50$ neighbors for the reversible Partial Neighbor Set $\mathcal{N}_i(x)$. For irreversible Partial Neighbor Sets, we don't need step 3. To make a fair comparison, we use two sizes of the Partial Neighbor Sets $|\mathcal{N}_i| = 50$ and 100.

The results are shown in Figure 16 and Figure 17. From the figures, we can see that the biases of the Markov chains by the irreversible Partial Neighbor Sets do decrease to near 0, which means such Markov chains do converge. In addition, it also performs better than the reversible ones. Does that mean we should use the irreversible Partial Neighbor Sets? Let's look at more simulation results.

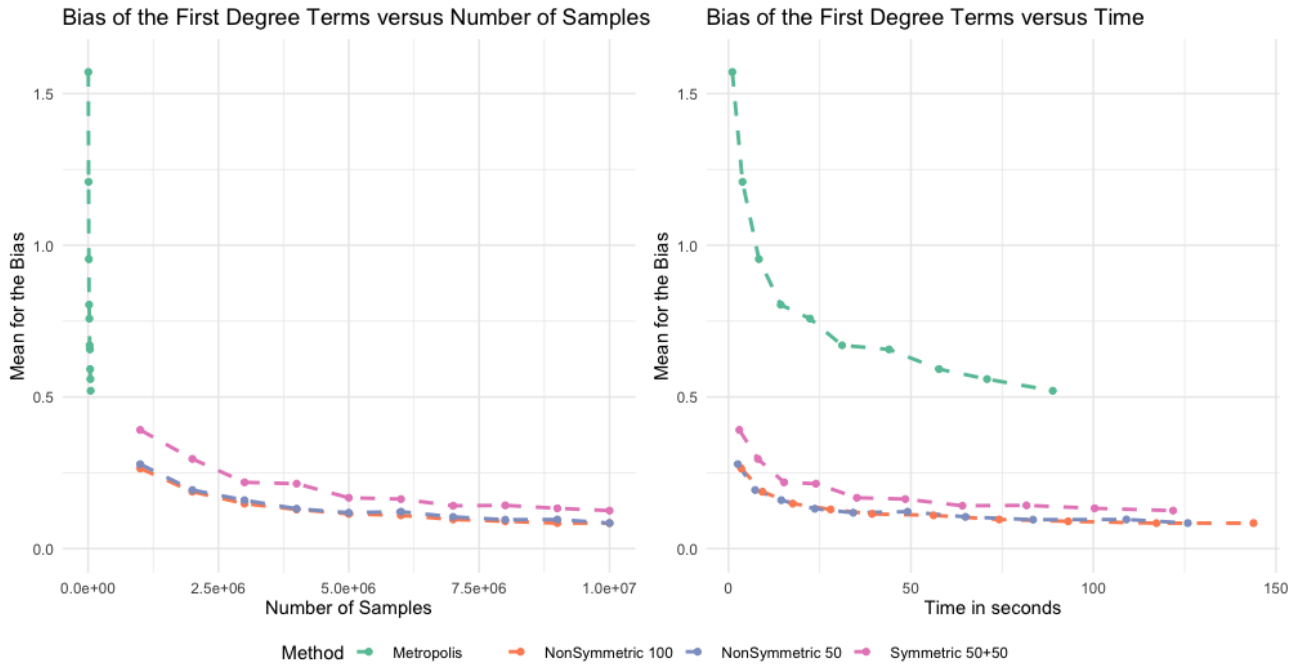

Figure 16: *Sum of the Average Bias of X_i's between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for two methods: Metropolis algorithm and Unbiased PNS with the reversible and non-reversible partial neighbor set. We used the Donuts example with $\mu_0 = 9$ and $\sigma = 0.1$. Each dot within the plot represents the result of the average bias value and time used for 30 simulation runs given certain original sample sizes. $L_0 = 1000$.*

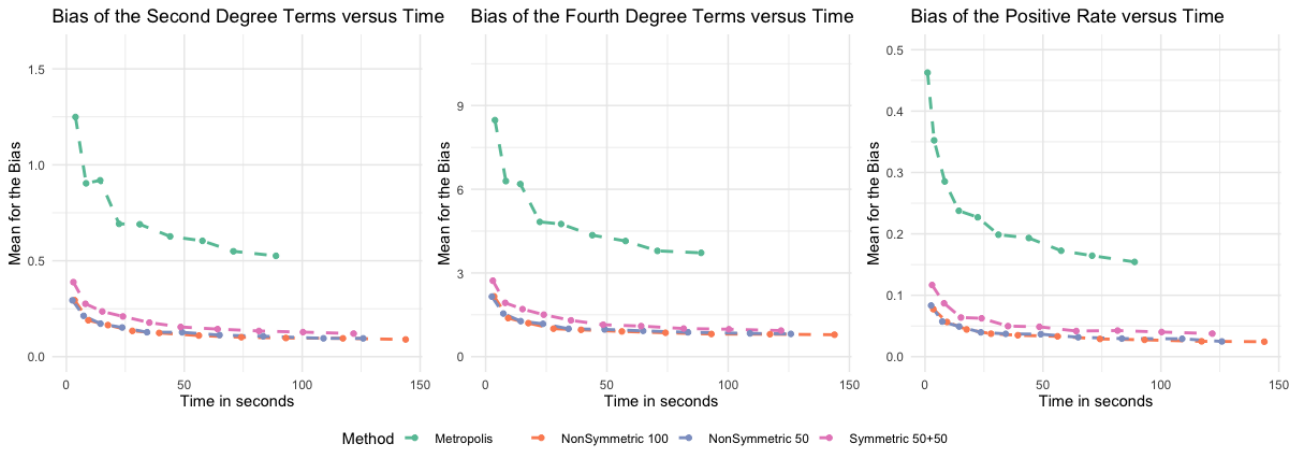

Figure 17: *Sum of the average bias from the second-degree terms (left), the fourth-degree terms (middle), and the positive rate (right) between sampling and target density π as a function of average time in seconds for two methods: Metropolis algorithm and Unbiased PNS with the reversible and non-reversible partial neighbor set. We used the Donuts example with $\mu_0 = 9$ and $\sigma = 0.1$. Each dot within the plot represents the result of the average bias value and time used for 30 simulation runs given certain original sample sizes. $L_0 = 1000$.*

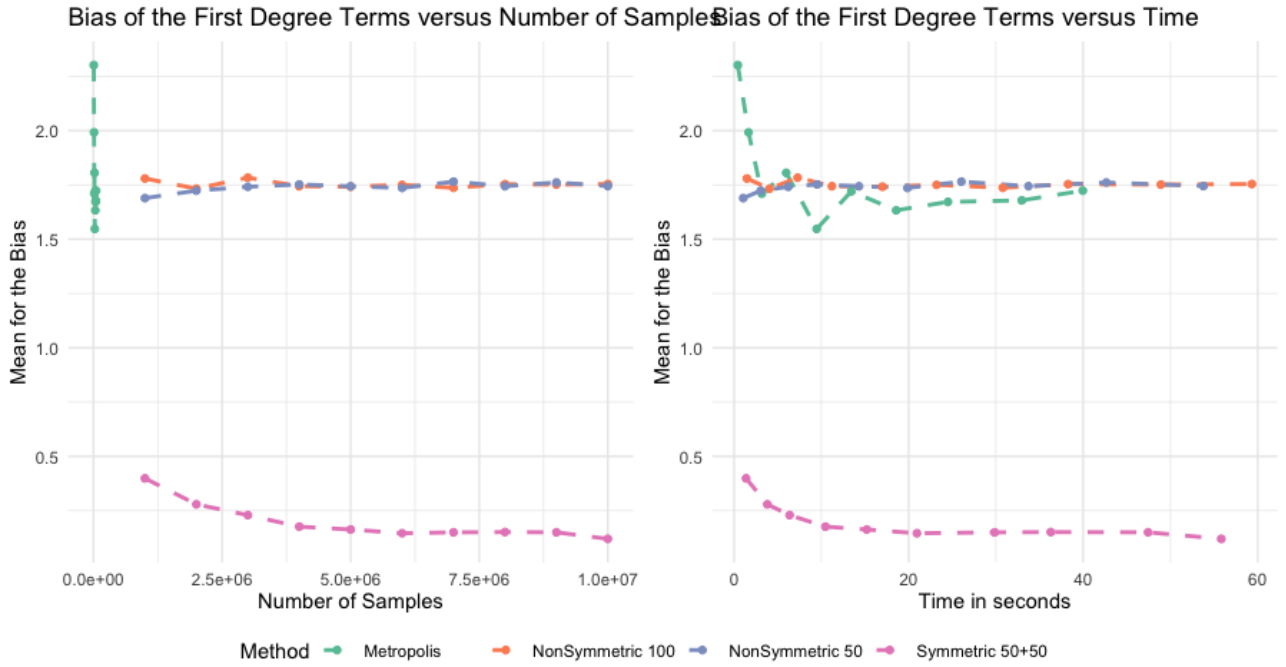

Figure 18: *Sum of the Average Bias of X_i's between sampling and target density π as a function of the number of iterations (left) and average time in seconds (right) for two methods: Metropolis algorithm and Unbiased PNS with the reversible and non-reversible partial neighbor set, where the proposal distribution is asymmetric with $\delta \sim Normal(0.1, 1)$. We used the Donuts example with $\mu_0 = 9$ and $\sigma = 0.1$. Each dot within the plot represents the result of the average bias value and time used for 30 simulation runs given certain original sample sizes. $L_0 = 1000$. According to the results here, we conclude that the non-reversible partial neighbor set is not always converging to the correct distribution.*

In the previous simulation, we used a symmetric proposal distribution for the increment part of the states. That is, we generated $\delta_1, \delta_2, \delta_3, \delta_4 \sim \text{Normal}(0, 1)$, which is symmetric around 0. How about other proposal distribution? For example, $\text{Normal}(0.1, 1)$ can be a workable proposal distribution for the Metropolis-Hastings algorithm. Although $\text{Normal}(0.1, 1)$ is not symmetric around 0, the Markov chain by the Metropolis-Hastings algorithm will still converge to the target density. Will the irreversible PNS chain converge correctly?

The results are shown in Figure 18 and Figure 19. From the figures, we can see that the biases of the Markov chains by the irreversible Partial Neighbor Sets are not converging to 0, which means such Markov chains do not converge. This result illustrates that such an irreversible PNS chain will not always converge correctly. People need to be very careful when using irreversible PNS. We include this simulation because naturally, people prefer irreversible PNS to reversible ones since reversible ones include one more step of including the negative increment, and such a step needs some time for the multicore hardware to communicate between cores. On the other hand, the first simulation shows that the irreversible PNS will converge to the target density under certain conditions, which can be another topic to explore in the future.

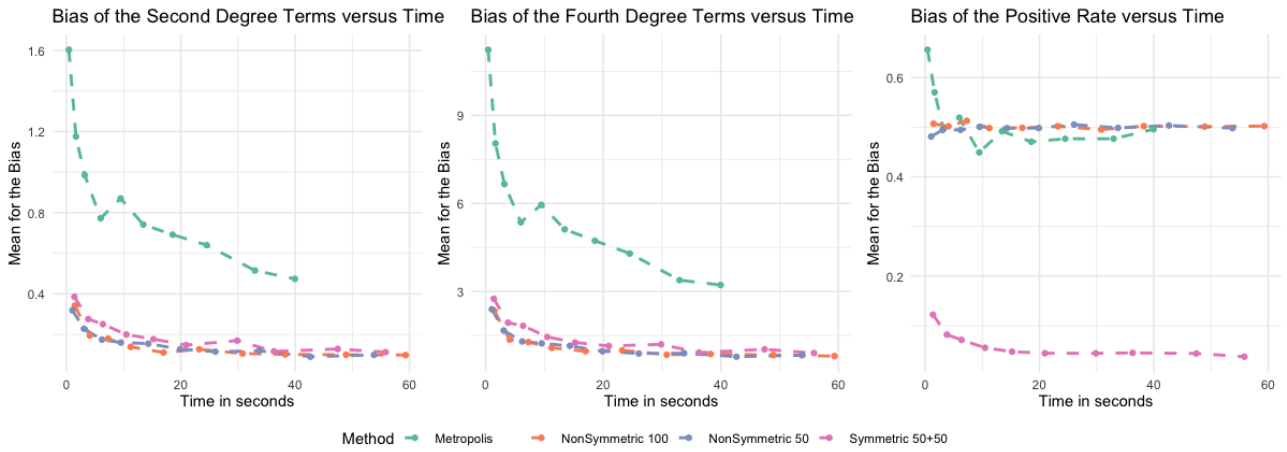

Figure 19: *Asymmetric Proposal* $\delta \sim Normal(0.1, 1)$*: Sum of the average bias from the second-degree terms (left), the fourth-degree terms (middle), and the positive rate (right) between sampling and target density* π *as a function of average time in seconds for two methods: Metropolis algorithm and Unbiased PNS with the reversible and non-reversible partial neighbor set, where the proposal distribution is asymmetric with* $\delta \sim Normal(0.1, 1)$*. We used the Donuts example with* $\mu_0 = 9$ *and* $\sigma = 0.1$*. Each dot within the plot represents the result of the average bias value and time used for 30 simulation runs given certain original sample sizes.* $L_0 = 1000$*.*