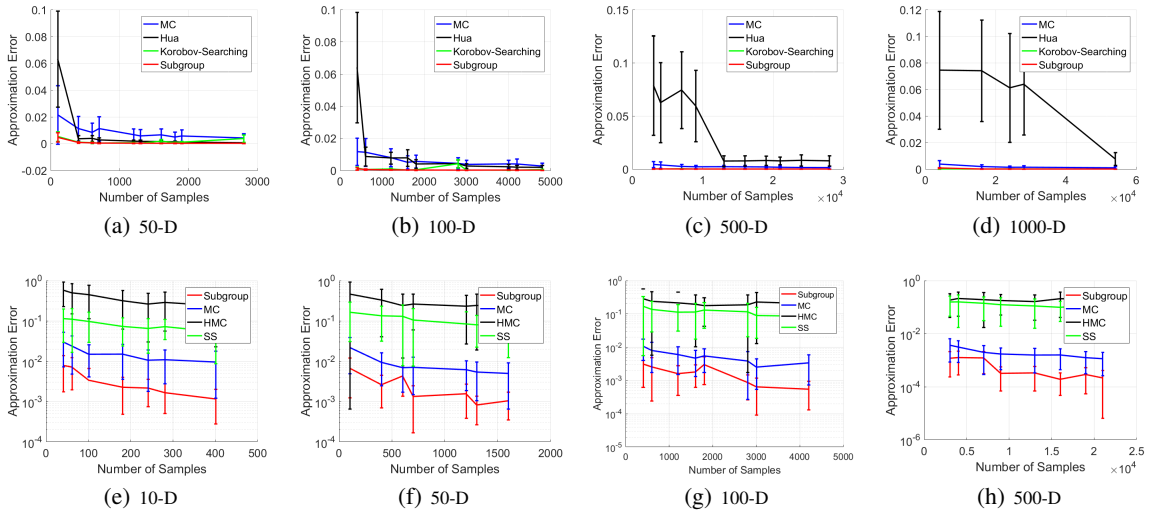

(a) 50-D (b) 100-D (c) 500-D (d) 1000-D

(e) 10-D (f) 50-D (g) 100-D (h) 500-D

We thank all the reviewers for their constructive comments! We will modify the paper by correcting all grammatical errors, adding the cross-references and the discussion of Hua's method.

Q1. (R1 & R2) Comparison of our subgroup rank-1 lattice with Hua and Korobov searching method on integral approximation problem in sec.5.2. The approximation errors are shown in Fig. (a)-(d). Hua's method obtains a smaller error than i.i.d Monte Carlo on the 50-D problem, however, it becomes worse than MC on 500-D and 1000-D problems. Our subgroup rank-1 lattice achieves similar performance to Korobov searching method and obtains a consistent smaller error on all the tested problems than Hua and MC.

Q2. (R2) Time Comparison of Korobov searching and our sub-group rank-1 lattice. The table below shows the time cost (seconds) for lattice construction. The run time for Korobov searching grows fast to hours. Our method can run in less than one second, achieving a $10^4\times$ to $10^5\times$ speed-up. The speed-up increases when n and d becomes larger.

d=500		n=3001	4001	7001	9001	13001	16001	19001	21001	24001	28001
	SubGroup	0.0185	0.0140	0.0289	0.043	0.0386	0.0320	0.0431	0.0548	0.0562	0.0593
	Korobov	34.668	98.876	152.86	310.13	624.56	933.54	1308.9	1588.5	2058.5	2815.9
d=1000		n=4001	16001	24001	28001	54001	70001	76001	88001	90001	96001
	SubGroup	0.0388	0.0618	0.1041	0.1289	0.2158	0.2923	0.3521	0.4099	0.5352	0.5663
	Korobov	112.18	1849.4	4115.9	5754.6	20257	34842	43457	56798	56644	69323

Q3. (All) More experiments: Approximation of the normalization constant of graphical model.

For Boltzmann Machines with continuous state in $[0,1]$, the energy function of $\boldsymbol{x} = [\boldsymbol{v}, \boldsymbol{h}] \in [0,1]^d$ is defined as $E(\boldsymbol{x}) = -(\boldsymbol{x}^\top \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x})/d$. The normalization constant is $Z = \int_{[0,1]^d} \exp\left(-E(\boldsymbol{x})\right) d\boldsymbol{x}$.

We evaluate our method on approximation of the normalization constant by comparing with i.i.d Monte Carlo (MC), slice sampling (SS) and Hamiltonian Monte Carlo (HMC). We generate the elements of $\boldsymbol{W}$ and $\boldsymbol{b}$ by sampling from standard Gaussian $\mathcal{N}(0,1)$. All the methods in comparison use the same $\boldsymbol{W}$ and $\boldsymbol{b}$. For SS and HMC, we use the *slicesample* function and *hmcSampler* function in MATLAB, respectively. We use the approximation of i.i.d MC with 10^7 samples as the pseudo ground-truth. The approximation errors $|\widehat{Z} - Z|/Z$ are shown in Fig.(e)-(h). our method consistently outperforms MC, HMC and SS on all cases. Moreover, our method is much cheaper than SS and HMC.

Q4. (R4) Comparison to sequential Monte Carlo. When the positive density region takes a large fraction of the entire domain, our method is very competitive (see Q3). When it is only inside a small part of a large domain, our method may not be better than sequential adaptive sampling. In this case, it is interesting to take advantage of both lattice and adaptive sampling. E.g., one can employ our subgroup rank-1 lattice as a rough partition of the domain to find high mass regions, then take sequential adaptive sampling on the promising regions with the lattice points as the start points. Also, it is interesting to consider progressively apply our subgroup rank-1 lattice to refine the partition.

Q5. (All) Benefits to NeurIPS community. Our subgroup rank-1 lattice performs good and robust. It does not have any hyperparameter and is very convenient and cheap for points set construction. It has potential applications at Bayesian inference, kernel approximation and the approximation of Wasserstein distance. It may also be able to combine with sequential MC as discussed in Q4. Readers may be inspired by or learned from our technique.