Reducing required randomness for parallel Gibbs sampling

Gellert Toth[1][0009−0002−9340−4807], Johan Kwisthout[1][0000−0003−4383−7786]

Radboud University, 6525 XZ Nijmegen, Netherlands

Abstract. Current state of the art Gibbs samplers parallelize sampling through several means, one being graphs colorings. However, parallelism is not the only relevant factor in the performance of a Gibbs sampler. Random number generation can be quite costly in terms of energy and hardware usage. Thus the number of random values needed, for carrying out the Gibbs update step, should also be optimized. This paper proposes a technique for reusing random numbers, which allows Gibbs samplers to reduce the required randomness without sacrificing any parallelism. We also introduce a novel coloring algorithm that is capable of producing multiple colorings along the curve of the trade-off between parallelism and required randomness. Between these colorings, a selection can be made given the parameters of the environment to further optimize performance.

Keywords: Parallel Gibbs sampling · Graph coloring · Chromatic Sampler · RNG

1 Introduction

Probabilistic graphical models (PGMs) represent a set of probabilistic variables and the conditional dependencies between them in the form of a directed (Bayesian Network) or undirected (Markov Random Fields) graph. The aim of these networks is to compute the probability distribution of the variables given that we know the state of some variables in the network.

In order to make use of PGMs, an efficient method for calculating the posterior distribution of the variables is needed. However, it has been shown [1] that exact inference is intractable (i.e., NP-hard), thus approximation methods are needed.

1.1 Gibbs sampler

The goal of the Gibbs sampler is to approximate the posterior probability distributions of the variables in the network. This is done by repeatedly drawing samples of the distribution at random under the assumption that given enough repetitions the distribution of the drawn samples will estimate the real distribution [4].

The original Gibbs sampler constructs a Markov chain of samples, where each element of the chain depends only on the previous element [4]. The distribution of values in the chain will converge to the real distribution. For a variable in the PGM a new value can be added to the chain by sampling the variable. This sample will depend on the last value of the neighbours of the variable in the network.

In particular this means that the Gibbs sampler executes this update step using the values from the previous step for every variable one-by-one. This is repeated for some predefined amount of iterations or until a certain exit criterion is met.

1.2 The Chromatic sampler

Sampling to a satisfactory approximation of the true distribution can require many samples and take a long time. In order to speed up sampling, the Gibbs update can be parallelized using a graph coloring. This method was proposed as the Chromatic sampler by Gonzalez et al. (2011) [5]. Their paper focuses on Markov Random Fields (MRFs), which are described by an undirected graph, but is applicable to Bayesian Networks as well. Their algorithm relies on the fact that two nodes can be sampled in parallel, when there is no edge between them in the MRF. A graph coloring provides exactly this: an assigned color to every node, where no two nodes of the same color are directly adjacent. Meaning, that given a coloring nodes of the same color can be sampled in parallel. Thus an iteration of the update step on a computer with p processors can be done in $O(\frac{n}{p} + k)$, for a MRF with n variables, given a k-coloring. A formal proof of this can be found under Proposition 3.1 of [5].

1.3 Independent random numbers

Although, the chromatic sampler achieves a significant improvement from the linear Gibbs sampler, there are further advancements to be made. Both algorithms use a distinct random number for each of the variables in the update step. However, we can reuse these random numbers to save on the costs of random number generation [7]. As even with recent improvements [2], random number generators (RNGs) remain quite costly in terms of resource usage, by reusing the random numbers we could save on hardware and energy usage. This is a criterion that was not taken into account by the Chromatic sampler. This paper will focus on creating sampling algorithms that take both the amount of parallelism achievable and the required number of random values into account.

More precisely, the sampler can use the same random number when two nodes of the MRF have at least two intermediate nodes on the shortest path between them. In such case the two nodes depend on completely different variables in the network. On the contrary, if the shortest path only contains one node, thus the sampled nodes share a common neighbour, an independent random number is needed. In practice this means that we need at least one independent random

value per color and possibly more if nodes within that color share neighbours. This means that the number of random values needed is at least the chromatic number, but in practice this is a very weak lower bound. This proposes a trade off between the amount of parallelism achievable and the amount of independent random values needed. Since there possibly could be a coloring which uses more colors but uses less independent random numbers. An example of the trade off is presented in Figure 1.

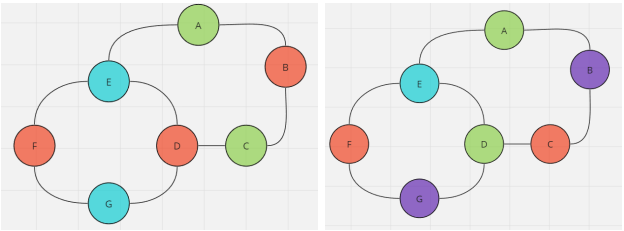

Fig. 1. The coloring on the left uses only 3 colors, but needs 6 independent random numbers (since E-G, F-D, A-C and B-D share a common neighbour). Whereas, the coloring on the right uses 4 different colors, but needs only 5 independent random numbers, as only the A-D pair requires additional independence.

Mapping out the curve of the trade off The goal is to map out the curve of this trade off and choose the point along the curve that fits our needs the best, based on the parameters of the environment. However, mapping out the exact curve is NP-hard, which we can easily show by noting that finding the chromatic number of a given graph is a by-product of mapping the curve out. Since at one of the extremes on the curve, one would have a coloring with the lowest theoretical amount of colors (which is the chromatic number). Meaning that if we find the curve, we also found the chromatic number, thus since finding the chromatic number is NP-hard [6] so is finding the curve of this trade off. Thus feasible solutions for mapping out the curve would only include approximation algorithms.

Ideal coloring algorithm The run time of the Chromatic sampler depends on the number of colors used in the coloring. Additionally the number of independent random numbers needed has to be considered as this affects hardware usage and energy costs. Without knowing what environment the sampling algorithm is running on it is impossible to say what is an ideal coloring. Since it is influenced by a lot of factors such as: number of processors, number of different RNGs available or the actual cost of getting a random number. Thus, an ideal coloring algorithm provides more than one coloring along the curve of the trade off. This way a selection can be made given the exact parameters of the environ-

ment out of the proposed options. Therefore, it makes sense to focus on coloring algorithms that produce more than one candidate.

1.4 Research Question

In this paper we propose an improved version of the Chromatic Gibbs sampler, which is equipped with our heuristic coloring algorithm chosen from a handful of candidates. We will discuss what improvements this method achieves compared to the original Chromatic sampler, in terms of number of random values needed compared to the possible parallelism.

1.5 Looking ahead

In the following Background section more technical details on the before mentioned notions and further motivation for the need for the proposed improvement will be provided. The Methods section will introduce our coloring algorithm and the means of comparison. Finally, the Results and Discussion sections will show how the algorithm performed.

2 Background

This chapter will dive deeper into when and how random numbers can be reused and will propose our changes to the chromatic sampler. Then we will shortly explain Recursive Largest First (RLF), a well known heuristic coloring algorithm that our most successful algorithm is based on.

2.1 Improving the chromatic sampler by reusing random values

Input: k-Colored MRF
`/* Iterate over all colors:` k_i `is the set of nodes in color` i `*/`
for $k_i : i \in \{1, ..., k\}$ **do**
  forall $X_j \in k_i$ **do in parallel**
    `/* Sample new value for node` j `depending on last value of neighboring nodes */`
    $X_j^{(t+1)} \sim \pi(X_j | X_{N_j}^{(t+1)} \in k_{<i}, X_{N_j}^{(t)} \in k_{>i})$ using random value sampled on demand
  end
end

Algorithm 1: Chromatic sampler [5]

Detailed explanation of the Chromatic Sampler Algorithm 1 shows the pseudo-code of the Chromatic sampler introduced by [5]. When sampling the

node j for time $(t+1)$ we make use of the values of its neighbours (N_j): for those that belong in an earlier color we take their $(t+1)^{th}$ sample, for those that belong in later colors we take their t^{th} sample.

As [5] argues the nodes in k_i (k_i is the set of nodes in in the i^{th} color) can be sampled in parallel as they are conditionally independent given the rest of the graph.

Reusing random numbers In the Gibbs sampling algorithm the value of a node is sampled according to the values of its neighbours. This is possible due to the definition of Markov Random Fields and Markov Blankets: the distribution of a variable is independent given its immediate neighbours from the rest of the graph. Formally:

$$\pi(X_i|X_{N_i}) = \pi(X_i|X_{\{1,\dots i-1,i+1,\dots n\}})$$

Thus nodes, whose Markov Blankets do not overlap can be sampled using the same random number, since their update depends on completely different nodes within the MRF. An example of this can be seen on Figure 2.

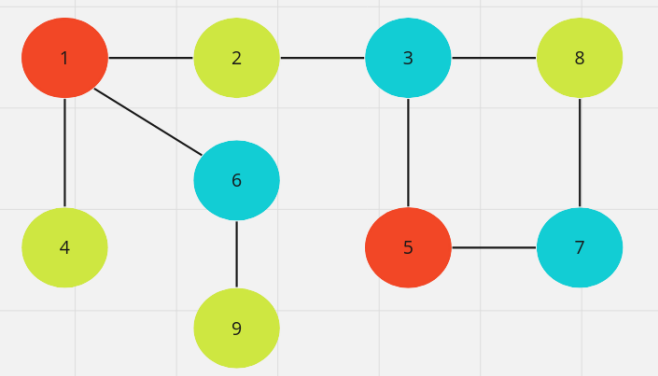

Fig. 2. In the graph above node 9 will be sampled based on the value of node 6, while node 2 will be sampled using node 1 and 3. Since there is no overlap between the nodes they depend on (their Markov blankets), node 2 and 9 can use the same random value for the update step. Similarly within the green color the (4,9) pair and (4, 8) pair could also use the same value.

We propose an algorithm that samples nodes of the same color in parallel and reuses random numbers needed between nodes within a color as much as possible. Nodes of different colors will always use independent random numbers (the alternative to this will be discussed later on). [5] showed that sampling can be parallelized given a graph coloring and we have showed that two nodes can reuse a random number if their Markov blankets do not overlap. However, we have not provided a proof that these two assumptions can be combined. We have not managed to obtain a formal proof, but a statistical test was conducted to

reinforce our intuition that this is indeed possible. This test will be explain in more detail later on.

So far we have only argued about when a pair of nodes could use the same random number, but this does not provide a full picture. In the example on Figure 2 the (2,9) pair could have the same random value and the (4,9) pair as well. However, this would mean that 4 and 2 also share the same value, which is not allowed. Thus we have to allocate these random values optimally, without breaching any of the restrictions. This will again be solved by obtaining a coloring on the graph constructed in the following manner (we will refer to this graph as **independence graph**):

- Take the set of nodes belonging to this color
- Add an edge between all pairs of nodes that are a distance of 2 apart (which is the case when they share a neighbour, thus their Markov blankets overlap)

Such a graph constructed on the green color from Figure 2 can be seen on Figure 3. Obtaining an optimal coloring for this graph gives the allocation for which nodes will use the same random values. Since coloring is an NP-hard problem one would use one of the many heuristics (two of which we will discuss in this section later) to obtain a (sub)optimal coloring.

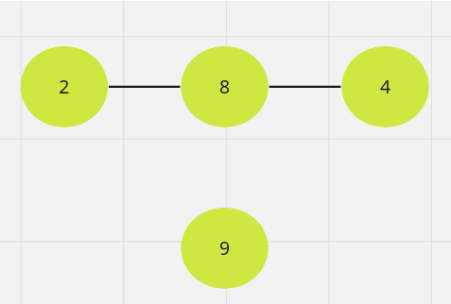

Fig. 3. The constructed independence graph for the green color on 2. The optimal coloring for this graph uses 2 colors, thus we will need 2 independent random values

There is also a known upper bound for the chromatic number of a graph: $\Delta + 1$, where Δ is the maximum degree in a graph. This can be easily proven the following way:

- choose a node without an assigned color, select the first color which none of its already colored neighbours have, repeat until all nodes are colored
- the upper limit for the color assigned is equal to the upper limit of the MEX (maximum excluded element) of a set with at most Δ elements (since Δ is the maximum degree), which is $\Delta + 1$
- thus a coloring with at most $\Delta + 1$ colors is always obtainable

Using separate colorings for deciding parallelism and the allocation of random values So far we forced nodes that have different colors to use independent random numbers by default, as we only defined reusal within a color. However, this is not a necessity: two nodes need separate random numbers if their Markov blankets overlap. This means that one could create coloring on the original MRF to decide how to parallelize the sampling and a completely separate coloring on the independence graph which decides the allocation of the random values. In other words there is no need for us to find colorings that optimize on both aspects as the same time as we can optimize them independently. However, this approach presents with a different drawback: the random values would have to be readily available for us prior to the parallel Gibbs update which means pre-computation (or through extremely complicated scheduling). Even with recent advancements in random number generators achieved by [2], RNGs remain costly in the sense of hardware usage and energy consumption. Thus pre-computing these random values could not be parallelized to the same extent as the Gibbs update step, meaning that any advancements we make would be irrelevant.

In contrast with this, when there are no shared random values across colors the need for pre-computation is gone. Since one could sample the random values only when the variables belonging to that color are sampled. In this scenario, there are also less values needed in parallel, which could make it more manageable to compute using multiple RNGs in parallel.

It has to be noted that under certain circumstances (such as actual cost of an RNG, number of processors, number of colors etc. etc.) either approach could be more beneficial. We will not go into further details, as it is outside of the scope of our research. However, from here on we will only focus on the initially presented approach and assume that it performs better than the alternative.

2.2 Improved Chromatic Sampler

Algorithm 2 shows the pseudo code of the improved chromatic sampler. The only alteration made to the original Chromatic sampler (Algorithm 1) is that along with a coloring of the MRF Algorithm 2 is given a secondary coloring for each color's independence graph. During the parallel update, the random value corresponding to the color of the node in the independence graph can be used to execute the Gibbs update. These random values are sampled on demand either linearly if we only have access to one RNG or in parallel if we are given multiple RNGs.

To improve our confidence in the fact that this algorithm maintains ergodicity we ran the linear Gibbs sampler, the Chromatic sampler and our Improved Chromatic sampler and verified that they all converge to the same distribution. For the sake of simplicity we created a small MRF where all conditional distributions were Bernoulli distributions with randomly selected probabilities. We then ran the algorithms to obtain the individual distributions of each of the variables and asserted that values between the algorithms do not diverge significantly. This test can be found on this git repository along with all other codes produced during this project:

https://gitlab.socsci.ru.nl/gellert.toth/thesis-gellert-toth

Input: k-Colored MRF with a secondary coloring over the independence graph of each color
for $k_i : i \in \{1, ..., k\}$ **do**
  forall $X_j \in k_i$ **do in parallel**
    `/* Execute Gibbs Update: */`
    $X_j^{(t+1)} \sim \pi(X_j | X_{N_j}^{(t+1)} \in k_{<i}, X_{N_j}^{(t)} \in k_{>i})$ using the random value obtained on demand for this color in the independence graph of color i
  end
end

Algorithm 2: Improved Chromatic sampler

2.3 Recursive Largest First

The coloring algorithm that we will introduce in section 3 of this paper took inspiration of a well-known coloring heuristic: Recursive Largest First (RLF) originally proposed by [3]. In order to understand our algorithm it is important to be familiar with RLF. The original algorithm is as follows:

1. find a maximal independent set, assign it the first unused color
2. remove the elements of the set from the graph
3. repeat from step 1, until the graph is empty and all nodes have a color assigned

To find a maximal independent set in step 1, the following technique was proposed by [3]:

1. let S be an empty set
2. add the node with the largest degree
3. repeatedly add the node with the most adjacent nodes that are adjacent to any node already in S but itself is not adjacent to any node in S. In case of a tie select the node with the minimum number of adjacent nodes not in S

We will refer to this method as RLF-Basic.

The original form of RLF-Basic intuitively goes against our goal of minimizing the required random numbers. When creating the maximal independent set S the node with the most 2 away neighbours already in S is added to the set. Meaning that we are maximizing the number of edges in the independence graph of this color and graphs with higher density tend to have higher chromatic numbers. However, by altering only the method of finding a maximal independent set, different versions of this algorithm could be created that behave more in line with our expectations.

3 Methods

In this chapter we will first discuss how we calculate the attributes of interest then introduce our best novel heuristic algorithm and our novel method for post processing a coloring.

3.1 Coloring evaluation

Score coloring To score a coloring we have to determine the two relevant attributes:

- **chrom**: the number of colors used in the coloring
- **random_count**: an upper bound on the number of random values needed for sampling

Extracting **chrom** is very straightforward: simply take the size of the set of colors. On the other hand **random_count** is a bit more tricky, as normally it would require us to find the chromatic number of all the independence graphs for every color, which can only be done in exponential time due to coloring being an NP-hard problem. Alternatively, in practice one would run a heuristic coloring which will give a (sub) optimal solution under an acceptable run time. However, we decided to use the previously mentioned $\Delta + 1$ upper bound for the sake of simplicity.

3.2 RLF-Remove-Worst

RLF-Basic is one of the best (inexpensive) coloring heuristics, but intuitively it is less good in optimizing random_count. However, by first removing the nodes that have the largest potential contribution to random_count, then running RLF-Basic, the final random_count could be greatly reduced. For this we propose the following algorithm:

1. create an empty set S
2. select the node that does not neighbour any nodes already in S in the graph with the most secondary neighbours (nodes that are reachable in two steps)
3. repeat step 2 until no node can be added
4. calculate a possible coloring for the parts of the graph that have no color yet using RLF-Basic and save this candidate, then repeat from step 1

This method will be referred to as RLF-Remove-Worst from here on. By incrementally creating more and more independent sets using our method then finishing the rest with the original, we can effectively find points along the trade off curve.

3.3 Post process coloring

As all potential coloring algorithms are heuristic there is generally room for improvement. For this purpose we created a post-processing method that all heuristics can rely upon to improve the coloring and achieve further easy gains. This is done as follows:

1. try all pair of colors and try to find a possible better allocation in regards to **random_count**. More details on this later.
2. reassign the colors of the pair that offers the most improvement on **random_count**
3. repeat step 1 until no improvements possible

To find improvements on the allocation of nodes between two colors one can either iterate through all possible allocations or randomly sample allocations to try out. The former only works when the set of nodes between the two colors is small, thus in practice this method would randomly sample a set amount of allocations. In our implementation this was set to 128. We will refer to this method of reallocating a pair of colors as two-color.

Optimally, one would precalculate a table containing the gain for every pair and after a reassignment of a pair only recalculate the relevant parts of the table. If implemented this way one iteration of this algorithm will call two-color $O(k)$ times and requires $O(k^2)$ calls to two-color in the precalculation step.

We will show in the later sections that this method has lead to drastic improvements on several of our heuristics. We will refer to this method as post-process.

3.4 Evaluation

bnlearn - bnrepository The Bayesian networks uploaded to bnrepository were used for bench-marking, as these networks are extensively used in research and provide a wide range of networks with sizes ranging from only a couple nodes to several hundred.

One first has to convert these Bayesian networks into Markov Random Fields through a method called moralization. This is done by adding an edge between all pairs of nodes that share a child in the graph of the network. Then we remove the directions of the edges. Since, we only care about the structure of the graph we do not have to worry about adjusting the conditional probability distributions which normally would be a part of moralization.

Comparison through Pareto curves The set of colorings produced by an algorithm can be reduced to an array of pairs of the two relevant attributes: [(chrom, random_count)]. From this the Pareto curve can to be constructed. The curve expresses the theoretical lower bound that this algorithm finds for a given graph. This curve can be found by simply sorting all pairs and filter out points for which there exists another point that is strictly better. A coloring is

clearly better than another one, if it has a smaller random_count and an an equal or smaller chrom. Meaning that it uses less random numbers, while using the same or less colors. Approaching this from the other direction, a coloring A will be part of the curve, if the the coloring with the highest chrom that is still smaller than the chrom of A, does not use less random numbers than A.

Formally, a pair ($chrom_i$, $random\_count_j$) will be part of the curve if and only if

$$\nexists j : chrom_j \leq chrom_i \wedge random\_count_j < random\_count_i$$

By allowing equality, the curve could have horizontal sections (when placing chrom on the x-axis), which represent that in this section of the chrom range there are no gains to be made in respect to random_count

By constructing an overall Pareto curve across all our algorithms, that we came up with in the process of writing this paper we can obtain an approximation of the actual curve, which is otherwise intractable to compute.

Given the overall curve and the curve for a specific algorithm, we can iterate through all the different values of **chrom** and calculate how much does the curve deviate from the overall curve (by diving their y values at that point). Averaging these values gives us the average proportional deviation from the theoretical limit.

This average deviation was used to compare our algorithms and enabled us to pick the best one to present in this paper.

4 Results

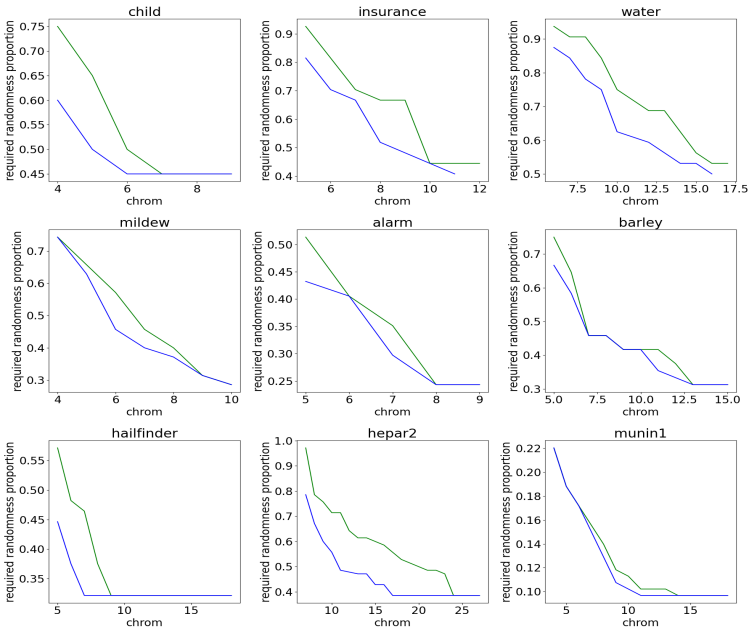

Fig. 4. The trade off curve produced by RLF-Remove-Worst on some of the benchmark networks. With the chrom range on the x-axis and the proportional need for randomness on the y-axis compared to the chromatic sampler. In green before post processing and in blue after post processing.

Table 1. The proportional difference to the overall curve across all Bayesian Networks that we run our algorithm on. The first two rows show the results before and after post-processing. The third row shows the gain achieved by post processing. The final column shows the average for all three rows.

	earthquake	cancer	survey	asia	sachs	child	insurance	water	mildew	alarm	barley	hailfinder	hepar2	win95pts	pathfinder	munin1	avg
before	1.0	1.0	1.067	1.083	1.0	1.11	1.221	1.195	1.143	1.104	1.125	1.092	1.296	1.164	1.0	1.048	1.103
after	1.0	1.0	1.0	1.083	1.0	1.0	1.053	1.055	1.067	1.028	1.073	1.007	1.033	1.025	1.0	1.015	1.027
gain %	0.0%	0.0%	6.2%	0.0%	0.0%	9.9%	13.7%	11.7%	6.6%	6.9%	4.7%	7.9%	20.3%	11.9%	0.0%	3.2%	6.4%

5 Discussion

This chapter will first discuss the gains compared to the chromatic sampler. Then we will shortly touch on the relevance of the post-processing method in

the pipeline. Then to conclude this paper we will discuss the limitations of our research along with what future research should focus on.

5.1 Gains from the improved chromatic sampler

Figure 4 shows the trade off curve for a few Bayesian networks. On the left extreme of the curves we have the scenario where we do not sacrifice parallelism compared to the chromatic sampler. In this case we can observe pure gains in our need for randomness as the improved version occasionally uses drastically less numbers. The difference is most prevalent in the case of MUNIN1, where with our method we need only 22% of the randomness the chromatic sampler would use.

The other important aspect of our method is that we not only use less randomness out of the box, but we can also continue to trade off some of the parallelism. Our goal was to present that this trade off is generally there. It was not part of the scope of our research to figure out the nature of the trade off or the extent of it. We believe that Figure 4 displays that the possibility is there and gains could be significant.

5.2 Analysis of post-processing

Along with our main line of research for mapping the curve of the trade off and experimenting with novel heuristic algorithms, we also introduced a novel post-processing method that can improve any given coloring and further reduce the required randomness.

It is important to note that, the better a coloring algorithm performs the less post-processing can improve on it, as there is a hard limit. However, even on our best performing algorithm post-processing achieved an over 6% gain across all networks. This could potentially be further improved by using more sample allocations in our algorithm. This would lead to increased runtime, but luckily the algorithm can be very easily and effectively parallelized. By implementing a parallel version of our post-processing the number of samples tried could be greatly increased.

5.3 Limitations

Our research had certain limitations that prevented us from making more specific claims about our results. The most limiting factor is that we did not calculate the actual trade off curve, rather we used the merge off all algorithms as an approximation. This means, that only comparison between algorithms or comparison between before and after post-processing can be made. We cannot make any claims about the actual performance of RLF-Remove-Worst.

Simultaneously, we did not run our algorithms on a subset of all networks, rather we used a very specific selection of networks. This means, that our numbers are indicative and cannot be generalized to the population of all networks. However, we believe that the networks chosen are fairly representative and they are generally used in research.

5.4 Conclusions and future work

We have proposed a new version of the chromatic sampler [5], which not only takes into account the amount of parallelism achievable, but also the amount of random values needed for sampling. In order to use the least amount of random values while also achieving the most parallelism, we proposed a heuristic coloring algorithm that can be used to decide both which nodes are sampled in parallel and which nodes use the same random values.

Equipped with RLF-Remove-Worst the improved chromatic sampler already uses less (for some graphs drastically less) random values while achieving the same parallelism as the chromatic sampler. However, our algorithm also allows the possibility to trade off some of the parallelism in order to use less random values. We believe that the experimental research concluded here, shows that there are significant gains to be made with the outlined methods.

Future research should focus on with dealing with the limitations of our paper to be able to make general claims on performance. Alternatively, a method for selecting the point along the trade off curve is also needed for an actual implementation of our ideas.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cooper, G.F.: The computational complexity of probabilistic inference using bayesian belief networks. Artificial Intelligence **42**(2), 393–405 (1990). https://doi.org/https://doi.org/10.1016/0004-3702(90)90060-D, https://www.sciencedirect.com/science/article/pii/000437029090060D
2. İsmail Emir Yüksel, Olgun, A., Salami, B., Bostancı, F.N., Tuğrul, Y.C., Yağlıkçı, A.G., Ghiasi, N.M., Mutlu, O., Ergin, O.: Turan: True random number generation using supply voltage underscaling in srams (2022)
3. Frank, T.L.: A Graph Coloring Alogrithm for Large Scheduling Problems. Journal of research of the National Bureau of Standards **84**, 489–506 (1979). https://doi.org/10.1007/978-3-030-81054-2
4. Geman, D., Horowitz, J., Rosen, J.: A local time analysis of intersections of Brownian paths in the plane. The Annals of Probability pp. 86–107 (1984)
5. Gonzalez, J., Low, Y., Gretton, A., Guestrin, C.: Parallel Gibbs Sampling: From colored fields to thin junction trees. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 324–332. PMLR, Fort Lauderdale, FL, USA (11–13 Apr 2011), https://proceedings.mlr.press/v15/gonzalez11a.html
6. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972). https://doi.org/10.1007/978-1-4684-2001-2_9, https://doi.org/10.1007/978-1-4684-2001-2_9
7. Kwisthout, J.: Approximate inference in Bayesian Networks: Parameterized complexity results. International Journal of Approximate Reasoning **93**, 119–131 (2018). https://doi.org/https://doi.org/10.1016/j.ijar.2017.10.029, https://www.sciencedirect.com/science/article/pii/S0888613X17306680