Bit-Blasting Probabilistic Programs

Poorva Garg [1] **Steven Holtzen** [2] **Guy Van den Broeck** [1] **Todd Millstein** [1]

Probabilistic programming languages (PPLs) have emerged as a prominent area of research in recent years due to their ability to democratize probabilistic modeling. PPLs are domain specific languages which have probability distributions as first class members and enable conditioning on available information. One of the key tasks of a PPL is to perform inference on a given probabilistic model. These probabilistic models are often a representation of real-life applications having instances of both discrete and continuous distributions and are referred to as *hybrid probabilistic models*. Some common examples include modeling skill of the players in tournaments [10] or modeling supply chain variables with gaussian distributions [11].

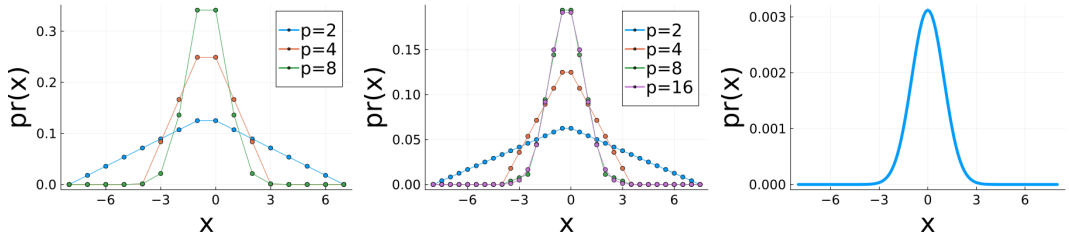
Current PPLs can be divided into two broad classes which either do not support or do not scale well on hybrid probabilistic programs. On one hand, we have special purpose PPLs which are fast for probabilistic programs they support but impose non-trivial constraints. For example, *Dice* only supports discrete probabilistic programs [6], Psi only supports exact inference on probabilistic programs with closed form algebraic representations [3] and Stan only supports continuous latent random variables [2]. On the other hand, there are universal Turing-complete PPLs like WebPPL and Anglican that can model any computable distribution, but this generality comes at a cost [4, 14]. Their sampling-based inference algorithms come with the risk of not scaling, obtaining not enough samples in limited time. These limitations make it non-trivial to use these probabilistic programming languages for hybrid probabilistic models.

We present *HyBit*, an approximate inference algorithm that provides better support and scalability for hybrid probabilistic programs and expand the class of programs that a special-purpose PPL can support. Through HyBit, we introduce bit blasting [16, 13] in the world of PPLs and enable the discrete probabilistic programming system *Dice* to be used for hybrid probabilistic models. *HyBit* first obtains a relaxed discrete abstraction of the hybrid probabilistic program. The continuous distributions in the program are limited to a finite range and discretized using a finite number of bits, leading to a discrete distribution over fixed point numbers. We represent these distributions over fixed point numbers as distributions over their binary encoding and further as distributions over Boolean random variables (or bits). This is referred to as *bit blasting*. Bit blasting enables use of traditional binary arithmetic operations and is particularly natural on top of recent knowledge compilation approaches to discrete inference. These approaches reduce probabilistic inference to weighted model counting over a weighted boolean formula represented using compact data structures like binary decision diagrams [6]. Once we have this relaxed discrete abstraction, we harness the power of these existing discrete PPLs to perform exact inference.

The bit blasting approach poses its own set of challenges. A naive approach to discretizing a continuous distribution using b bits would be to associate 2^b probabilities with their fixed point number represented by the bits. But as you increase the number of bits to achieve higher accuracy, it would not always be possible to enumerate the exponentially increasing number of values. To counter this problem, we present an efficient way to approximate a discrete distribution using linear piece-wise distributions. Such approximations can be written down compactly, using a number of independent Boolean random variables (or flips) that is only linear in the number of bits and pieces. Figure 1 shows different approximations possible for a normal distribution for different number of bits and linear pieces. It can be clearly seen that as we increase the number of linear pieces for a fixed bitwidth, the discretized approximation gets closer to the continuous normal distribution. Figure 1(c) shows an approximation of a normal distribution using 7 bits and 128 linear pieces which seems visually indistinguishable from a continuous normal distribution. It is important to note here that the discretized approximation corresponds to the discretized cumulative probability. That is, the probability of a point, say x_i in the approximation is the integration of density with limits x_i and x_{i+1} where x_{i+1} is the next representable number using b bits. If there are enough linear pieces, i.e. 2^{b-1}, we would be representing the exact discretization using b bits leaving discretization error as the only source of approximation.

In this work, we prove theoretically that as we increase the number of bits, the bit blasted representation converges to the continuous distribution. We further provide empirical evidence to show that bit blasting probabilistic programs can handle a broader class of programs and is more accurate than existing approaches within a given threshold of time.

[1]University of California, Los Angeles [2]Northeastern University. Correspondence to: Poorva Garg <poorvagarg@cs.ucla.edu>, Steven Holtzen <s.holtzen@northeastern.edu>, Guy Van den Broeck <guyvdb@cs.ucla.edu>, Todd Millstein <todd@cs.ucla.edu>.



(a) $b = 4$ (b) $b = 5$ (c) $b = 7, p = 128$

Figure 1. Discretizations of normal distribution for different bits and pieces

```
xs = [1.0, 2.0, 3.0, 4.0, 5.0]
ys = [2.023, 2.542, 3.001, 1.919, 0.712]
z1 = flip(0.5)
z2 = flip(0.5)
b1 = if z1 0 else N(0, 1) end
b2 = if z2 0 else N(0, 1) end
for (x, y) in zip(xs, ys)
    mean = x*b1 + b2
    observe(N(mean, 1) == y)
end
return z1
```

Figure 2. Spike and Slab Regression

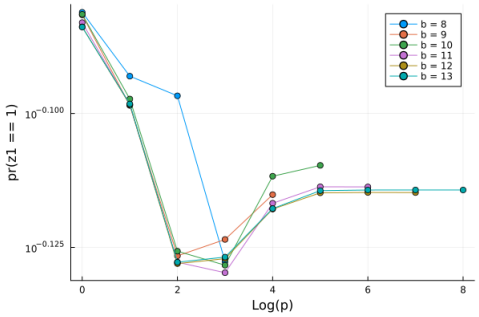


Figure 3. Output of *HyBit*

Table 1. Comparison of HyBit with existing PPLs. We report the least absolute error achieved by each approach within a time limit of 2 hours. A **bold** denotes the least absolute error among the systems compared.

Benchmark	*HyBit*	AQUA [8]	WebPPL(rejection) [4]	WebPPL(MCMC) [4]	Psi (Exact) [3]
zeroone [1]	**4.51E-04**	3.69E+00	1.39E+00	1.44E+00	Timeout
spacex [7]	**6.94E-04**	Not supported	2.00E-03	3.15E-03	Not supported
GPA [15]	2.22E-16	3.62E-01	1.95E-03	5.42E-03	**0.00E+00**
tug [8]	**4.47E-08**	Not supported	8.63E-05	6.99E-04	Not supported

Example Figure 2 shows an example which represents a linear regression model with two features having discrete-continuous mixtures as priors. These distributions are known as spike and slab priors where spike refers to the discrete distribution and slab refers to the continuous distribution. These distributions are used to induce sparsity in regression models to select relevant features from a large set of features [9]. Discrete-continuous mixtures also have other applications, like modeling quantities with threshold limits and the GPA problem is one such widely known example in the probabilistic programming literature [15].

Current PPLs do not cater well to discrete-continuous mixtures; their inference algorithms do not provide support for such distributions. For example, HMC and NUTS sampling impose requirements of almost-everywhere differentiability of posterior distribution [5]. To make models amenable to such sampling algorithms, spike-and-slab priors are replaced with horseshoe priors making models more complicated [12]. *HyBit* does not suffer from these limitations, it naturally supports discrete-continuous mixtures as an inference algorithm by replacing continuous density functions with bit-blasted probability mass functions.

HyBit needs three elements, 1) the hybrid probabilistic program 2) the number of bits (b) and 3) the number of linear pieces (p) to obtain the relaxed discrete abstraction. We need the number of bits as a specification of the precision of the discrete abstraction and the number of linear pieces to avoid enumerating all the values of the discretized distribution and come up with an efficient approximation. Using these parameters, continuous distributions are replaced with their bit blasted linear piece-wise approximation. Now we perform exact inference on the relaxed discrete abstraction, using weighted model counting supported by *Dice*, to obtain an approximate of the inference query. Figure 3 shows the output of using *HyBit* on the example of spike and slab regression with different number of bits and linear pieces. Table 1 shows some preliminary experimental results.

References

[1] P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, feb 2016. doi: 10.1111/rssb.12158. URL https://doi.org/10.1111%2Frssb.12158.

[2] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017. doi: 10.18637/jss.v076.i01. URL https://www.jstatsoft.org/index.php/jss/article/view/v076i01.

[3] T. Gehr, S. Misailovic, and M. Vechev. Psi: Exact symbolic inference for probabilistic programs. In *International Conference on Computer Aided Verification*, pages 62–83. Springer, 2016.

[4] N. D. Goodman and A. Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2022-10-26.

[5] M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo, 2011. URL https://arxiv.org/abs/1111.4246.

[6] S. Holtzen, G. Van den Broeck, and T. Millstein. Scaling exact inference for discrete probabilistic programs. *Proc. ACM Program. Lang. (OOPSLA)*, 2020. doi: https://doi.org/10.1145/342820.

[7] https://gist.github.com/canyon289. URL https://gist.github.com/canyon289/73890bab211c5cbaea41ad6f32df01a5.

[8] Z. Huang, S. Dutta, and S. Misailovic. Aqua: Automated quantized inference for probabilistic programs. In *International Symposium on Automated Technology for Verification and Analysis*, pages 229–246. Springer, 2021.

[9] G. Malsiner-Walli and H. Wagner. Comparing spike and slab priors for bayesian variable selection. 2018. doi: 10.48550/ARXIV.1812.07259. URL https://arxiv.org/abs/1812.07259.

[10] T. Minka, J. Winn, J. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill. /Infer.NET 0.3, 2018. Microsoft Research Cambridge. http://dotnet.github.io/infer.

[11] E. Moradi, M. R. Ghezel Arsalan, A. Naimi Sadigh, and H. Ghalb. Quantitative models in supply chain management. *Supply Chain Sustainability and Raw Material Management: Concepts and Processes*, pages 285–312, 01 2011. doi: 10.4018/978-1-61350-504-5.ch016.

[12] J. Piironen and A. Vehtari. Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2), jan 2017. doi: 10.1214/17-ejs1337si. URL https://doi.org/10.1214%2F17-ejs1337si.

[13] C. Smith, J. Hsu, and A. Albarghouthi. Trace abstraction modulo probability. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019. doi: 10.1145/3290352. URL https://doi.org/10.1145/3290352.

[14] D. Tolpin, J. W. van de Meent, H. Yang, and F. Wood. Design and implementation of probabilistic programming language anglican. *arXiv preprint arXiv:1608.05263*, 2016.

[15] Y. Wu, S. Srivastava, N. Hay, S. Du, and S. Russell. Discrete-continuous mixtures in probabilistic programming: Generalized semantics and inference algorithms, 2018. URL https://arxiv.org/abs/1806.02027.

[16] Y. Zohar, A. Irfan, M. Mann, A. Niemetz, A. Nötzli, M. Preiner, A. Reynolds, C. W. Barrett, and C. Tinelli. Bit-precise reasoning via int-blasting. In B. Finkbeiner and T. Wies, editors, *Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings*, volume 13182 of *Lecture Notes in Computer Science*, pages 496–518. Springer, 2022. doi: 10.1007/978-3-030-94583-1_24. URL https://doi.org/10.1007/978-3-030-94583-1_24.