Interleave Variational Optimization with Monte Carlo Sampling: A Tale of Two Approximate Inference Paradigms

Qi Lou
University of California, Irvine
Irvine, CA 92697, USA
qlou@ics.uci.edu

Rina Dechter
University of California, Irvine
Irvine, CA 92697, USA
dechter@ics.uci.edu

Alexander Ihler
University of California, Irvine
Irvine, CA 92697, USA
ihler@ics.uci.edu

Abstract

Computing the partition function of a graphical model is a fundamental task in probabilistic inference. Variational bounds and Monte Carlo methods, two important approximate paradigms for this task, each has its respective strengths for solving different types of problems, but it is often nontrivial to decide which one to apply to a particular problem instance without significant prior knowledge and a high level of expertise. In this paper, we propose a general framework that interleaves optimization of variational bounds (via message passing) with Monte Carlo sampling. Our adaptive interleaving policy can automatically balance the computational effort between these two schemes in an instance-dependent way, which provides our framework with the strengths of both schemes, leads to tighter anytime bounds and an unbiased estimate of the partition function, and allows flexible tradeoffs between memory, time, and solution quality. We verify our approach empirically on real-world problems taken from recent UAI inference competitions.

Introduction

Probabilistic graphical models, including Bayesian networks and Markov random fields, are a set of powerful frameworks for representing and reasoning with probabilistic and deterministic information (Darwiche 2009; Dechter 2013; Dechter, Geffner, and Halpern 2010). Reasoning in a graphical model often requires computing the partition function, i.e., the normalizing constant of the underlying distribution. Exact computation of the partition function is known to be intractable (Valiant 1979) in general, leading to the development of many approximate schemes, the major categories of which are variational methods, Monte Carlo sampling, and search algorithms. Within these, techniques that provide some guaranteed confidence intervals on the correct value, and can be improved with additional computation, are valued for providing users with concrete information or certificates of accuracy within a reasonable amount of time, and allowing the user to decide the desired balance of quality versus time.

Variational bounds (Wainwright and Jordan 2008) and closely related approximate elimination methods (Dechter and Rish 2003; Liu and Ihler 2011) provide deterministic guarantees on the partition function. However, these bounds are not anytime; their quality often depends critically on the amount of memory available, and do not continue to improve (approach the correct value) without additional memory.

Monte Carlo methods (Liu 2008), such as importance sampling (Dagum and Luby 1997; Liu, Fisher, and Ihler 2015) gives probabilistic bounds that improve with more samples at a predictable rate; in practice this means bounds that improve rapidly at first, but can be slow to become very tight.

Search algorithms (Henrion 1991; Viricel et al. 2016; Lou, Dechter, and Ihler 2017a) explicitly enumerate over the space of configurations and eventually provide an exact answer; however, while some problems are well-suited to search, others only improve their quality very slowly with more computation.

Several algorithms combine two or more strategies. Approximate hash-based counting combines sampling (of hash functions) with CSP-based search (Chakraborty et al. 2014; Chakraborty, Meel, and Vardi 2016) or other MAP queries (Ermon et al. 2013; 2014), although these are not typically formulated to provide anytime behavior. Some recent works combine importance sampling with (partially) exact inference (Broka et al. 2018; Friedman and Van den Broeck 2018), though they do not have guarantees on their estimates. A number of works use some form of approximate elimination or variational bounds as search heuristics, such as Lou, Dechter, and Ihler (2017a), while Liu, Fisher, and Ihler (2015) develops an importance sampling proposal from variational bounds that provides strong probabilistic confidence intervals. More recently, Lou, Dechter, and Ihler (2017b) unifies these two works within the same framework.

However, a typical approach to incorporating variational methods is to build and optimize the variational bound via message passing first, waiting for this procedure to terminate before initiating search and/or sampling. While this works well enough at longer time scales (minutes to hours), it is undesirable if we want high-quality anytime bounds on faster time scales. Failing to build a high-quality variational bound can dramatically slow down the subsequent progress of search or sampling on more difficult problems, but for easier problems that would be solvable with a less optimized heuristic or proposal, solution time is dominated by this initial build time; unfortunately it is difficult to know beforehand into which regime a particular problem instance will fall. Our framework fills this gap by interleaving optimization of variational bounds with importance sampling from the very beginning, and provides an instance-specific balance designed to give rapid anytime improvement.

Our contributions. First, we propose a general framework that interleaves optimization of variational upper bounds with importance sampling to achieve anytime upper and lower bounds of the partition function, which is directly applicable to a set of convex variational bounds including TRW (Wainwright, Jaakkola, and Willsky 2005) and WMB (Liu and Ihler 2011). Second, we propose an effective adaptive policy that automatically balances the computational effort between these two processes on the fly in a problem-dependent fashion. Third, our experiments on real-world problems demonstrate that our interleaving framework with the adaptive policy is superior to several non-interleaving baselines and interleaving baselines with simple static policies in terms of anytime performance, gives competitive final bound quality, and is relatively insensitive to its hyperparameter, making it easy to use and automate in practice.

Background

In this section, we introduce some notations and background knowledge that are essential to present and understand our algorithm.

Let $X = (X_1, \dots, X_n)$ be a vector of random variables, where each X_i takes values in a discrete domain $\mathcal{X}_i$; we use lower case letters, e.g. $x_i \in \mathcal{X}_i$, to indicate a value of X_i, and x to indicate an assignment of X. A graphical model over X consists of a set of factors $\mathcal{F} = \{f_\alpha(X_\alpha) \mid \alpha \in \mathcal{I}\}$, where each factor f_α is defined on a subset $X_\alpha = \{X_i \mid i \in \alpha\}$ of X, called its scope.

We associate an undirected graph $\mathcal{G} = (V, E)$ with $\mathcal{F}$, where each node $i \in V$ corresponds to a variable X_i and we connect two nodes, $(i, j) \in E$, iff $\{i, j\} \subseteq \alpha$ for some α. The set $\mathcal{I}$ then corresponds to cliques of $\mathcal{G}$. We can interpret $\mathcal{F}$ as an unnormalized probability measure, so that

$$f(x) = \prod_{\alpha \in \mathcal{I}} f_\alpha(x_\alpha), \qquad Z = \sum_x \prod_{\alpha \in \mathcal{I}} f_\alpha(x_\alpha).$$

Z is called the *partition function*, and normalizes $f(x)$. Computing Z is often a key task in evaluating the probability of observed data, model selection, or computing predictive probabilities.

Weighted Mini-bucket

In experiments, we apply our framework to a particular variational bound called weighted mini-bucket (WMB) (Liu and Ihler 2011), which we briefly introduce here to make our paper self-contained. WMB is an approximate elimination algorithm that generalizes its elimination-based predecessors (e.g., bucket elimination (Dechter 1999), mini-bucket elimination (Dechter and Rish 2003)) by relaxing the exact summation using a "power sum" operation during variable elimination, and controls the computational complexity using a user-specified parameter called the *ibound*.

Given an elimination order, WMB processes variables one by one. Upon reaching a variable, say, X_i, WMB first collects all factors (including those intermediately generated ones, termed "messages") with X_i in their scopes, which form a factor set called "bucket" $\mathcal{B}_i$. WMB then partitions $\mathcal{B}_i$ into several disjoint "mini-buckets" $\{\mathcal{B}_i^j\}$ to ensure that each $\mathcal{B}_i^j$ involves no more than $(ibound + 1)$ variables.

After assigning a "weight" ρ_{ij} to each $\mathcal{B}_i^j$, WMB eliminates X_i in factors of $\mathcal{B}_i^j$ using the power sum:

$$\lambda_{i \to \pi_{ij}} = (\sum_{x_i} \prod_{f_\alpha \in \mathcal{B}_i^j} f_\alpha^{\frac{1}{\rho_{ij}}})^{\rho_{ij}},$$

which sends a message $\lambda_{i \to \pi_{ij}}$ to $X_{\pi_{ij}}$, the earliest uneliminated variable in the scope of $\lambda_{i \to \pi_{ij}}$. This message $\lambda_{i \to \pi_{ij}}$ is then placed in the bucket of $X_{\pi_{ij}}$ for later processing.

When the weights ρ_{ij}'s are nonnegative and sum to one, Hölder's inequality guarantees that the product of the messages $\lambda_{i \to \pi_{ij}}$ is an upper bound of the sum, i.e.,

$$\sum_{x_i} \prod_{f_\alpha \in \mathcal{B}_i} f_\alpha \le \prod_j \lambda_{i \to \pi_{ij}}.$$

Therefore, WMB eventually returns an upper bound U of the partition function Z after it terminates. Liu and Ihler (2011) also shows that the resulting bound is equivalent to a class of bounds based on tree reweighted (TRW) belief propagation (Wainwright, Jaakkola, and Willsky 2005), or more generally conditional entropy decompositions (Globerson and Jaakkola 2007).

U can be tightened by cost shifting (a.k.a., reparameterization) and weight optimization, which can be implemented in a forward-backward message passing procedure. We refer readers to Liu and Ihler (2011) for details.

Weighted Mini-bucket Importance Sampling

This same relaxation of WMB can also be used to define a proposal distribution $q(x)$ (Liu, Fisher, and Ihler 2015) as follows:

$$q(x) = \prod_i \sum_j \rho_{ij} q_{ij}(x_i | x_{an_j(i)}), \tag{1}$$

where $X_{an_j(i)}$ are variables included in the mini-bucket $\mathcal{B}_i^j$ excluding X_i. The $q_{ij}(x_i | x_{an_j(i)})$'s are conditional distributions:

$$q_{ij}(x_i | x_{an_j(i)}) = \big(\prod_{f_\alpha \in \mathcal{B}_i^j} f_\alpha / \lambda_{i \to \pi_{ij}} \big)^{\frac{1}{\rho_{ij}}}.$$

One can show that importance weights derived from this proposal distribution are bounded and unbiased, i.e.,

$$f(x)/q(x) \le U, \quad \mathbb{E}\big[f(x)/q(x)\big] = Z, \tag{2}$$

which leads to finite-sample bounds of the partition function (see Corollary 3.2 of Liu, Fisher, and Ihler (2015)).

Example. Fig. 1 shows an example of running WMB on a pairwise graphical model to produce an upper bound of the partition function and a proposal distribution when an elimination order and an *ibound* are imposed.

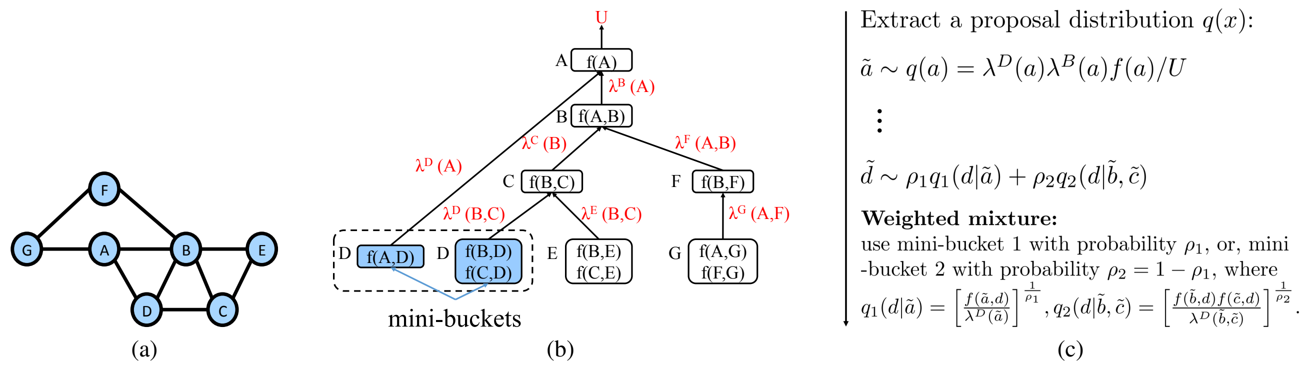

G
F
A
B
E
D
C
(a)
U
A f(A)
λ^B (A)
B f(A,B)
λ^C (B)
λ^D (A)
λ^F (A,B)
C f(B,C)
F f(B,F)
λ^D (B,C)
λ^E (B,C)
λ^G (A,F)
D f(A,D)
D f(B,D) f(C,D)
E f(B,E) f(C,E)
G f(A,G) f(F,G)
mini-buckets
(b)
Extract a proposal distribution $q(x)$:
$\tilde{a} \sim q(a) = \lambda^D(a)\lambda^B(a)f(a)/U$
$\tilde{d} \sim \rho_1 q_1(d|\tilde{a}) + \rho_2 q_2(d|\tilde{b},\tilde{c})$
Weighted mixture:
use mini-bucket 1 with probability ρ_1, or, mini -bucket 2 with probability $\rho_2 = 1 - \rho_1$, where
$q_1(d|\tilde{a}) = \left[\frac{f(\tilde{a},d)}{\lambda^D(\tilde{a})}\right]^{\frac{1}{\rho_1}}, q_2(d|\tilde{b},\tilde{c}) = \left[\frac{f(\tilde{b},d)f(\tilde{c},d)}{\lambda^D(\tilde{b},\tilde{c})}\right]^{\frac{1}{\rho_2}}.$
(c)

Figure 1: (a) A graphical model over 7 variables. (b) Given an elimination order G, F, E, D, C, B, A, and $ibound = 2$, WMB runs on this model to produce an upper bound of the partition function. (c) The same relaxation from (b) gives a proposal distribution with properties present in (2).

Interleaving Variational Optimization with Importance Sampling

In this section, we present a general scheme that interleaves optimization of the variational upper bound with importance sampling, derive finite-sample bounds for this scheme, and discuss interleaving policies.

A General Interleaving Framework

Our general scheme is quite simple: we interleave the two processes according to some policy. Since optimization of variational bounds is typically through a message passing procedure, we consider interleaving message passing with importance sampling in our algorithm. Procedurally, we first build up an initial bound within a given memory budget, and then interleave the two processes following a policy that we will discuss in the sequel. Alg. 1 presents details of our framework.

Our framework actually serves as a "meta-algorithm" from which any optimizable proposal with properties in (2) can benefit; in particular this includes convex variational bounds such as WMB and TRW (see Liu, Fisher, and Ihler (2015) for more details on why this holds for TRW).

Interleaving these two processes leads to rapid early bound improvement, since the improvement provided by the probabilistic bounds is not delayed until after variational optimization converges or is otherwise terminated (e.g., in Liu, Fisher, and Ihler (2015) and Lou, Dechter, and Ihler (2017b)); this results in significant improvements in anytime behavior as we demonstrate in the empirical evaluation. A critical point is that, by interleaving updates to the proposal, our samples are no longer identically distributed; in fact, their variance is decreasing as the proposal improves, and for best results we should up-weight these more accurate samples in our estimates. To this end, and analogous to that of Lou, Dechter, and Ihler (2017b), we define a weighted average estimator $\hat{Z}$ and apply an empirical Bernstein bound (Maurer and Pontil 2009) to derive finite-sample bounds on the error between Z and $\hat{Z}$ based on independent samples from a sequence of proposals satisfying (2):

Algorithm 1 A General Interleaving Framework

```
Require: memory budget, time budget, confidence parameter $\delta$, interleaving policy $P$.
Ensure: $N$, $U$, $\widehat{Z}$, $\mathrm{HM}(\boldsymbol{U})$, $\Delta$, $\widehat{\mathrm{Var}}$.
1:  Build an initial variational upper bound that fits the memory budget.
2:  while within the time budget do
3:      Generate $(R, S)$ via policy $P$.
4:      for $r \leftarrow 1$ to $R$ do
5:          Run one round of message passing.
6:          Update $U$.
7:      end for
8:      for $s \leftarrow 1$ to $S$ do
9:          Draw one sample from the current proposal.
10:         Update $N$, $\widehat{Z}$, $\mathrm{HM}(\boldsymbol{U})$, $\Delta$, $\widehat{\mathrm{Var}}$ via (3), (4), (6)
11:         and (7) respectively.
12:     end for
13:     Update policy $P$ if necessary.
14: end while
```

Theorem 1. Let $\{x^i\}_{i=1}^N$ be a series of samples drawn from proposal distributions $\{q_i(x)\}_{i=1}^N$ respectively via Alg. 1, with $\{\widehat{Z}_i = f(x^i)/q_i(x^i)\}_{i=1}^N$ the corresponding importance weights, and $\{U_i\}_{i=1}^N$ the corresponding variational upper bounds respectively. Let

$$\widehat{Z} = \frac{\mathrm{HM}(\boldsymbol{U})}{N} \sum_{i=1}^{N} \frac{\widehat{Z}_i}{U_i}, \tag{3}$$

where

$$\mathrm{HM}(\boldsymbol{U}) = \left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{U_i}\right]^{-1} \tag{4}$$

is the harmonic mean of $\{U_i\}_{i=1}^N$. then, $\widehat{Z}$ is an unbiased estimator of Z, i.e.,

$$\mathbb{E}\,\widehat{Z} = Z. \tag{5}$$

Define a deviation term

$$\Delta = \mathrm{HM}(\boldsymbol{U})\Big(\sqrt{\frac{2\widehat{\mathrm{Var}}\ln(2/\delta)}{N}} + \frac{7\ln(2/\delta)}{3(N-1)}\Big), \quad (6)$$

where

$$\widehat{\mathrm{Var}} = \frac{1}{N-1}\sum_{i=1}^{N}\Big[\frac{\widehat{Z}_i}{U_i} - \frac{\widehat{Z}}{\mathrm{HM}(\boldsymbol{U})}\Big]^2 \quad (7)$$

is the unbiased empirical variance of $\{\widehat{Z}_i/U_i\}_{i=1}^{N}$. Since the boundedness of each $\widehat{Z}_i$ guarantees that $\widehat{Z}_i/U_i \leq 1$ for all i, applying standard empirical Bernstein results we have

$$\begin{aligned} \Pr[Z \leq \widehat{Z} + \Delta] \geq 1 - \delta, \\ \Pr[Z \geq \widehat{Z} - \Delta] \geq 1 - \delta, \end{aligned} \quad (8)$$

i.e., $\widehat{Z} + \Delta$ and $\widehat{Z} - \Delta$ are upper and lower bounds of Z with probability at least $1 - \delta$, respectively.

Interleaving Policies

The key for our framework to work well is its interleaving policy. Our goal is to design a policy balancing the effort between message passing and sampling, so that the probabilistic bounds in (8) can improve as quickly as possible. Alg. 1 controls this balance by alternating between R steps of message passing, and S steps of sampling, within each iteration.

Optimize-First Policy. In most prior work, the variational bound is optimized first, as a separate pre-processing step. Within the framework of Alg. 1, this takes the form of setting $(R, S) = (1, 0)$ (all message passing) until some convergence or time-out criteria are satisfied, then changing the policy P so that $(R, S) = (0, 1)$ (all sampling). A simple strategy is to switch over after some fixed time. As noted previously, this approach suffers from several drawbacks. The deterministic variational bounds are considerably weaker than those provided by sampling, and the early work serves mainly to improve the quality of later sampling, so that quality does not improve in a smooth, anytime way. A related point is that this makes it difficult to know how much effort to put into the optimization process; too little, and the probabilistic bounds will improve only slowly; too much, and we waste time that could have been used for sampling.

Static Policy. Alternatively, we can interleave updates using a simple static policy, fixing (R, S) in Alg. 1 to some constants. While this choice will alternate between update types, giving smoother performance, it is also non-trivial to automate for different types of problems. In particular, it is difficult to know how any given (R, S) will perform a priori; the characteristics of the problem instance may make message updates more or less expensive and more or less effective, changing the desired balance between R and S.

Adaptive Policy. The main issue with static policies is that they are "blind" to the current status and behavior of the inference process on a particular problem instance. We propose an adaptive policy that is able to adjust its behavior based on information available on the fly. The basic idea of our adaptive policy is to select the action that is projected to have larger unit contribution to the probabilistic upper bound in each iteration. Details follow.

Suppose we have already drawn N samples so far. The deviation term Δ in (6) can be roughly approximated by Δ':

$$\Delta \approx \Delta' = \mathrm{HM}(\boldsymbol{U})\Big(\sqrt{\frac{\ln(2/\delta)}{2N}} + \frac{7\ln(2/\delta)}{3(N-1)}\Big). \quad (9)$$

Δ' is derived from Δ by substituting the empirical variance $\widehat{\mathrm{Var}}$ (see (7)) with $1/4$, which is an upper bound of the empirical variance in expectation according to Popoviciu's inequality (Popoviciu 1935). This means that Δ' is actually an upper bound of the expectation of Δ thanks to the concavity of the square root function. Therefore, the probabilistic upper bound $\widehat{Z} + \Delta$ in (8) can be approximated by $Z + \Delta'$ since $\widehat{Z}$ is an unbiased estimate of Z (see (5)).

Thus, we define a gain function that approximates improvement in the probabilistic upper bound (difference between current one and the projected one) if we draw N' samples from now with respect to the variational upper bound:

$$\begin{aligned} gain(N, U, N') &= (Z + \Delta') - (Z + \Delta'_{N'}) \\ &= \Delta' - \Delta'_{N'} \end{aligned}$$

where

$$\Delta'_{N'} = \Big[\frac{1}{N+N'}\Big(\frac{N'}{U} + \sum_{i=1}^{N}\frac{1}{U_i}\Big)\Big]^{-1}\Big(\sqrt{\frac{\ln(2/\delta)}{2(N+N')}} + \frac{7\ln(2/\delta)}{3(N+N'-1)}\Big).$$

Assuming that one sampling step takes time t_{is}, we can easily define the unit gain $gain_{is}$ of drawing one sample:

$$gain_{is} = gain(N, U, 1)/t_{is} \quad (10)$$

However, to define the gain of running one message passing step is more complicated: message passing does not directly contribute to the current probabilistic bound, but rather, affects the deterministic upper bound and quality of all later samples. We introduce a projected upper bound U' that one message passing step can achieve from now, and assume one message passing step will affect N_s later samples with this projected bound. We then define the unit gain $gain_{msg}$ of one message passing step:

$$gain_{msg} = gain(N, U', N_s)/t_{msg} \quad (11)$$

where t_{msg} is the time it takes to complete one message passing step. Note that U' is computed via a simple linear interpolation in our experiments.

In a nutshell, our adaptive policy compares $gain_{is}$ and $gain_{msg}$ and takes the action with larger unit gain in the following step. Note that neither $gain_{is}$ nor $gain_{msg}$ involves information from the samples themselves, which ensures that their independence is preserved and that Theorem 1 still applies.

Table 1: Statistics of the three evaluated benchmark sets.

	`pedigree`	`protein`	`promedas`
# instances	22	50	50
avg. # variable	917.14	99.96	682.12
avg. # of factor	917.14	355.84	682.12
avg. max domain size	4.95	77.94	2.00
avg. max scope	4.45	2.00	3.00
avg. induced width	25.50	11.24	25.76

Table 2: Notations and abbreviations used in figures and tables for the evaluated algorithms.

fixed $p\%$	optimize-first policy with the first $p\%$ of time for message passing, and the rest for sampling.
static (R, S)	static policy with some given (R, S).
equal time	static policy with $(1, t_{msg}/t_{is})$.
adaptive N_s	adaptive policy with N_s pseudo samples.

Empirical Evaluation

In this section, we present empirical results to demonstrate the usefulness of our framework and the effectiveness of our adaptive policy.

We evaluated on three benchmarks of real-world problem instances from recent UAI competitions. Our benchmarks include: `pedigree`, 22 genetic linkage instances from the UAI'08 inference challenge[1]; `protein`, 50 instances made from the "small" protein side-chains of (Yanover and Weiss 2002); `promedas`, 50 medical diagnosis expert systems (Wemmenhove et al. 2007). These three sets are selected to illustrate different problem characteristics. Table 1 shows some summary statistics of these benchmarks.

We adopted WMB as our variational bound, and allocated a maximum of 512MB memory, using the largest *ibound* that fit our memory budget. We set a maximum time budget of 600 seconds. The confidence parameter δ for our probabilistic bounds is set to 0.025. In the experiments, we also used $\widehat{Z}\delta$, a $(1-\delta)$ probabilistic lower bound by the Markov inequality (Gogate and Dechter 2011), and switched to our lower bound $\widehat{Z} - \Delta$ when it becomes non-trivial. We also replaced $\widehat{Z} + \Delta$ with the best deterministic upper bound reached so far if the latter is tighter.

Table 2 explains the evaluated algorithms, all of which share the same initial WMB structure. We test our adaptive approach against several non-interleaved strategies ("fixed-X") as well as statically interleaved strategies ("static-X"). The "equal time" strategy corresponds to "static $(1, t_{msg}/t_{is})$" instead of "static $(t_{is}/t_{msg}, 1)$" because message passing is usually much more expensive than sampling (typically, $t_{msg}/t_{is} > 10^3$). Note that "static (0, 1)" can also be viewed as an optimize-first (non-interleaving) strategy "fixed 0%", because it does not spend any time improving the variational bound after the initial bound construction. During one round of message passing, i.e., one forward-backward pass (see Liu and Ihler (2011)), we do cost-shifting and weight optimization simultaneously. All implementations are in C/C++. We ran experiments on AMD Opteron 6276 processors with a clock speed of 2.3 GHz.

Interleaving versus Non-interleaving

Fig. 2 shows anytime bounds from some typical instances of each benchmark for interleaving and non-interleaving strategies. We can observe from Fig. 2 that those non-interleaving strategies except "static (0, 1)" lack good anytime behavior compared to our adaptive interleaving strategy: they are unable to compute a lower bound until they quit message passing and start sampling; their early upper bounds correspond to the deterministic variational bounds, which typically do not improve as fast as the probabilistic upper bound of our adaptive variant. "static (0, 1)" responds quickly but often gives looser results at longer time scales; see Fig. 2(c) and 2(f) for example. It performs very well when the initial bound is already close to the ground truth (see Fig. 2(a)), but such information is usually not available to users beforehand.

To quantify the anytime performance of the methods in each benchmark, we use two measures: one is the area between the upper bound of $\log Z$ and the (estimated) ground truth of $\log Z$; the other is the area between the upper and lower bound of $\log Z$ as introduced in Lou, Dechter, and Ihler (2017b). The first facilitates comparison with those methods that do not provide lower bounds early on. These quantities are computed for each instance and method, and then normalized by those of "static (0, 1)". Finally, we take the (geometric) mean over these scores across each benchmark.

From Table 3, we observe that our adaptive policy performs significantly better in terms of anytime upper bound than any of the non-interleaving variants across all the benchmarks. We can also see differences in performance stemming from the different problem characteristics in the benchmarks: for example, the performance of the non-interleaved strategies degrades with higher time used for message passing on the `pedigree` benchmark, while this is less true of the other two benchmarks; this indicates the difficulty of deciding when to quit message passing and start sampling for good anytime behavior, especially when we do not know the time limit in advance. In contrast, our interleaving framework does not have such limitations.

We also examine performance at a fixed time limit, as opposed to anytime behavior. Table 5 shows the mean final gap between the upper bound and the (estimated) $\log Z$. We can see that our adaptive variants perform almost as well as the best method for each benchmark, which implies that although our algorithm is designed for a more responsive, anytime behavior, it does not sacrifice much in terms of long-term bound quality.

Adaptive versus Static

We next compare our adaptive interleaving strategy with several statically interleaved approaches. Fig. 3 shows anytime bounds of two instances per benchmark for various interleaving settings. From Fig. 3, we observe that in general, the

[1]http://graphmod.ics.uci.edu/uai08/Evaluation/Report/Benchmarks/

Table 3: Mean area between upper bounds and (estimated) ground truth $\log Z$, normalized by that of “static (0, 1)”, for each benchmark. Smaller numbers indicate better anytime upper bounds. The best for each benchmark is bolded.

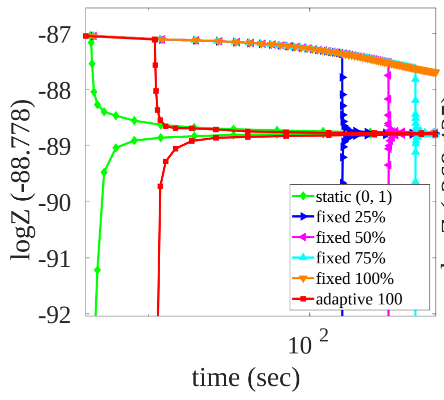
	static (0,1)	static (1,10)	static (1,100)	fixed 25%	fixed 50%	fixed 75%	fixed 100%	equal time	adaptive 10	adaptive 100
`pedigree`	1	2.436	1.262	2.084	3.027	3.904	4.655	0.947	**0.781**	0.786
`protein`	1	0.145	0.077	0.161	0.231	0.285	0.329	0.054	0.051	**0.050**
`promedas`	1	1.217	0.615	0.898	1.408	1.855	2.268	0.414	0.354	**0.349**

Table 4: Mean area between upper and lower bounds of $\log Z$, normalized by that of “static (0, 1)”, for each benchmark. Entries for some non-interleaved strategies are missing because they do not give lower bounds early on. Smaller numbers indicate better anytime bounds. The best for each benchmark is bolded.

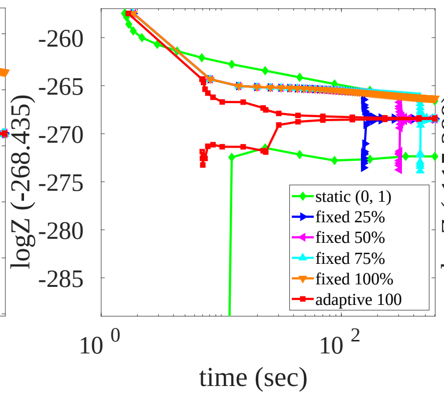
	static (0,1)	static (1,10)	static (1,100)	fixed 25%	fixed 50%	fixed 75%	fixed 100%	equal time	adaptive 10	adaptive 100
`pedigree`	**1**	4.624	2.213	-	-	-	-	2.061	1.544	1.529
`protein`	1	0.299	0.118	-	-	-	-	0.104	0.080	**0.078**
`promedas`	1	2.674	1.203	-	-	-	-	1.271	0.728	**0.726**

Table 5: Mean final gap between upper bound and (estimated) ground truth $\log Z$, normalized by that of “static (0, 1)”, for each benchmark. Smaller numbers are better; the best method for each benchmark is bolded.

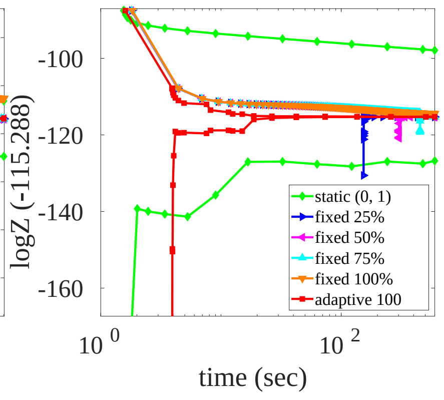
	static (0,1)	static (1,10)	static (1,100)	fixed 25%	fixed 50%	fixed 75%	fixed 100%	equal time	adaptive 10	adaptive 100
`pedigree`	1	2.113	0.804	**0.468**	0.480	0.581	6.576	0.534	0.554	0.572
`protein`	1	0.045	0.016	**0.004**	**0.004**	0.005	0.208	0.005	0.006	0.005
`promedas`	1	0.879	0.308	**0.121**	0.133	0.185	2.519	0.165	0.150	0.148



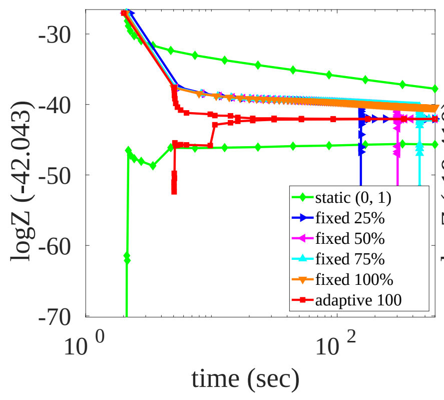
(a) `pedigree`/pedigree23



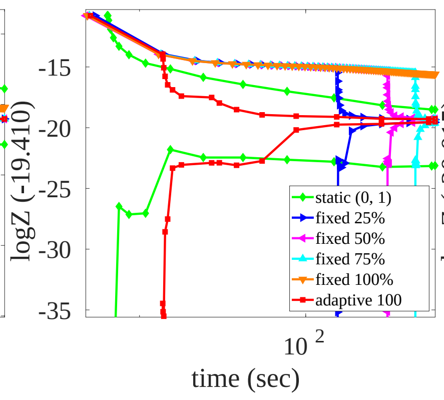
(b) `pedigree`/pedigree37



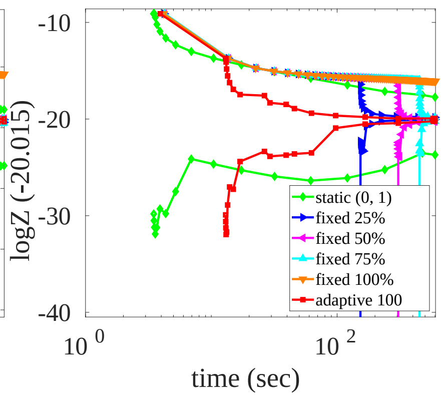
(c) `protein`/1qsq



(d) `protein`/1whi

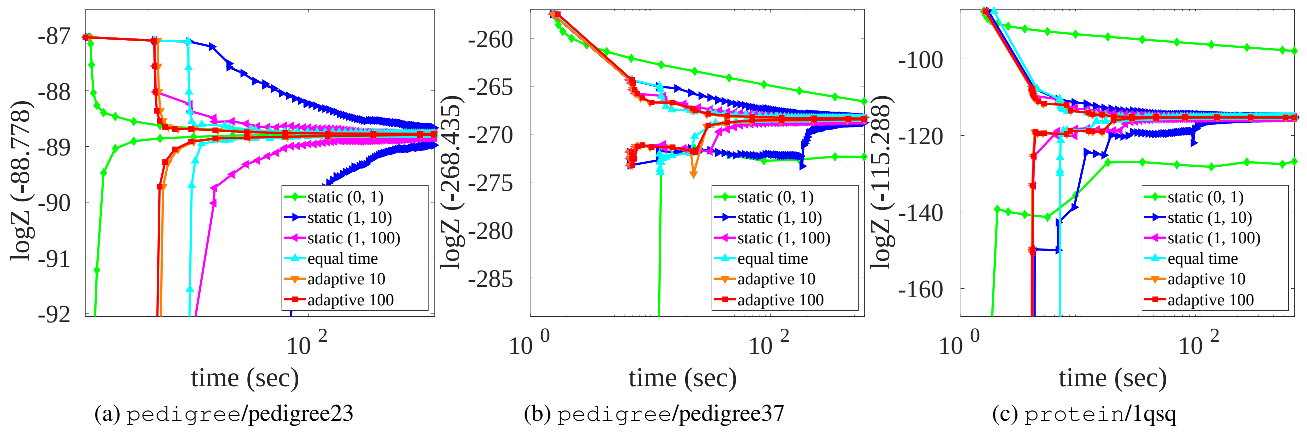


(e) `promedas`/or_chain_132.fg

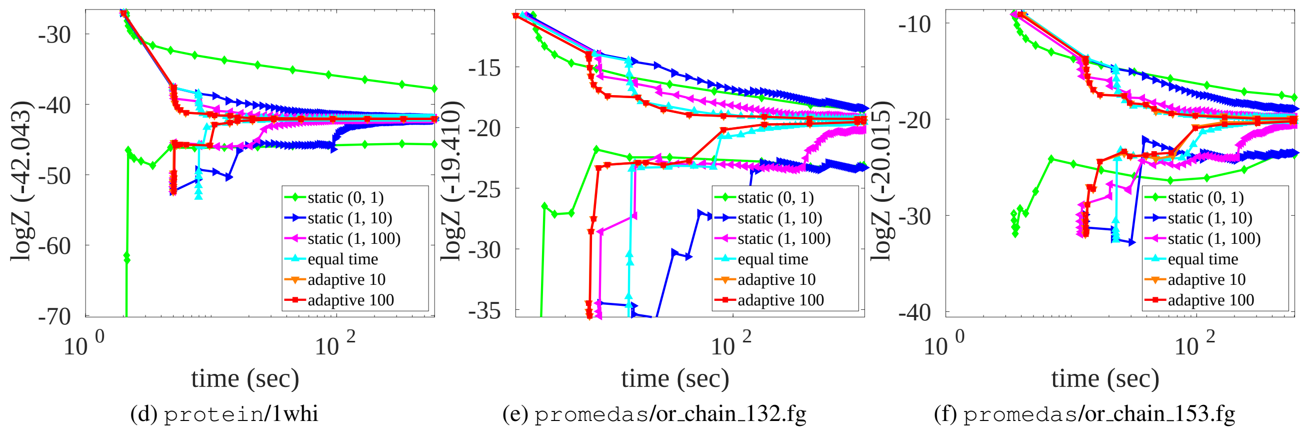


(f) `promedas`/or_chain_153.fg

Figure 2: Anytime bounds on $\log Z$ for two instances per benchmark, comparing our adaptive interleaving strategy with non-interleaved strategies, spending various fractions of the time (0% through 100%) optimizing the deterministic bound first (see Table 2; “static(0,1)” is equivalent to “fixed 0%”). The adaptive strategy strikes a balance between responsiveness (giving tighter bounds early) and long-term performance (tight bounds later).



(a) pedigree/pedigree23 (b) pedigree/pedigree37 (c) protein/1qsq



(d) protein/1whi (e) promedas/or_chain_132.fg (f) promedas/or_chain_153.fg

Figure 3: Anytime bounds on $\log Z$ for two instances per benchmark, comparing our adaptive strategy with statically interleaved strategies balancing message updates and sampling (see Table 2). The best fixed strategy is typically "equal time", but adaptivity is usually slightly better, as it is able to change its behavior dynamically based on the observed improvement in bounds from message updates.

"equal time" variant performs better than a static choice of interleaving rate, since it is able to take into account how computationally expensive the message updates are. Even so, both of our adaptive variants (corresponding to a shorter or longer horizon when estimating the impact of a message update) perform better most of the time; they are able to make more informed decisions about the current benefits of sampling versus message updates. These observations are also supported by the statistical results in Table 3 and Table 4. From Fig. 3, Table 3, and Table 4, we can also see that the two adaptive variants perform fairly similarly. In fact, we found in our experiments that our adaptive policy works reasonably well within a wide range of N_s, i.e., it is not very sensitive to the hyperparameter value, which simplifies its use in practice.

Conclusion

In this work, we propose a general framework that interleaves optimization of a variational upper bound with importance sampling based on its associated proposal, to obtain high-quality anytime bounds and estimates of the partition function. This framework can be viewed as a meta-algorithm for convex variational bounds such as TRW and WMB, giving them more responsive bounds without sacrificing long-term quality. Our proposed adaptive policy, which selects the action with larger unit gain for improving the probabilistic upper bound at each iteration, leads to excellent empirical anytime performance, both in comparison to simple non-interleaved baselines as well as simpler interleaved policies within our framework. Our approach is easy to use in practice since it does not appear to be sensitive to its hyperparameter in our experiments. For future work, one may consider incorporating importance sampling in the initial bound construction as well, to further boost the anytime performance.

Acknowledgements

We thank all the reviewers for their helpful feedback.

This work is sponsored in part by NSF grants IIS-1526842 and IIS-1254071, the U.S. Air Force (Contract FA9453-16-C-0508), and DARPA (Contract W911NF-18-C-0015).

References

Broka, F.; Dechter, R.; Ihler, A.; and Kask, K. 2018. Abstraction sampling in graphical models. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, UAI'18.

Chakraborty, S.; Fremont, D. J.; Meel, K. S.; Seshia, S. A.; and Vardi, M. Y. 2014. Distribution-aware sampling and weighted model counting for SAT. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI'14, 1722–1730.

Chakraborty, S.; Meel, K. S.; and Vardi, M. Y. 2016. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, 3569–3576.

Dagum, P., and Luby, M. 1997. An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence* 93(1-2):1–27.

Darwiche, A. 2009. *Modeling and reasoning with Bayesian networks*. Cambridge University Press.

Dechter, R., and Rish, I. 2003. Mini-buckets: A general scheme of approximating inference. *Journal of ACM* 50(2):107–153.

Dechter, R.; Geffner, H.; and Halpern, J. Y. 2010. *Heuristics, Probability and Causality. A Tribute to Judea Pearl*. College Publications.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1-2):41–85.

Dechter, R. 2013. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 7(3):1–191.

Ermon, S.; Gomes, C.; Sabharwal, A.; and Selman, B. 2013. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, ICML'13, 334–342.

Ermon, S.; Gomes, C.; Sabharwal, A.; and Selman, B. 2014. Low-density parity constraints for hashing-based discrete integration. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ICML'14, 271–279.

Friedman, T., and Van den Broeck, G. 2018. Approximate knowledge compilation by online collapsed importance sampling. *arXiv preprint arXiv:1805.12565*.

Globerson, A., and Jaakkola, T. 2007. Approximate inference using conditional entropy decompositions. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, AISTATS'07, 131–138.

Gogate, V., and Dechter, R. 2011. Sampling-based lower bounds for counting queries. *Intelligenza Artificiale* 5(2):171–188.

Henrion, M. 1991. Search-based methods to bound diagnostic probabilities in very large belief nets. In *Proceedings of the 7th conference on Uncertainty in Artificial Intelligence*, UAI'91, 142–150.

Liu, Q., and Ihler, A. 2011. Bounding the partition function using Hölder's inequality. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, 849–856.

Liu, Q.; Fisher, III, J. W.; and Ihler, A. 2015. Probabilistic variational bounds for graphical models. In *Advances in Neural Information Processing Systems*, NIPS'15, 1432–1440.

Liu, J. S. 2008. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

Lou, Q.; Dechter, R.; and Ihler, A. 2017a. Anytime anyspace AND/OR search for bounding the partition function. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, AAAI'17, 860–867.

Lou, Q.; Dechter, R.; and Ihler, A. 2017b. Dynamic importance sampling for anytime bounds of the partition function. In *Advances in Neural Information Processing Systems*, NIPS'17, 3198–3206.

Maurer, A., and Pontil, M. 2009. Empirical Bernstein bounds and sample variance penalization. In *Proceedings of the 22nd Conference on Learning Theory*, COLT'09.

Popoviciu, T. 1935. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica* 9:129–145.

Valiant, L. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8(2):189–201.

Viricel, C.; Simoncini, D.; Barbe, S.; and Schiex, T. 2016. Guaranteed weighted counting for affinity computation: Beyond determinism and structure. In *International Conference on Principles and Practice of Constraint Programming*, 733–750. Springer.

Wainwright, M., and Jordan, M. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1(1-2):1–305.

Wainwright, M.; Jaakkola, T.; and Willsky, A. 2005. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory* 51(7):2313–2335.

Wemmenhove, B.; Mooij, J. M.; Wiegerinck, W.; Leisink, M.; Kappen, H. J.; and Neijt, J. P. 2007. Inference in the promedas medical expert system. In *Conference on Artificial Intelligence in Medicine in Europe*, 456–460. Springer.

Yanover, C., and Weiss, Y. 2002. Approximate inference and protein-folding. In *Advances in Neural Information Processing Systems*, NIPS'02, 1457–1464.