Incremental Region Selection for Mini-bucket Elimination Bounds

Sholeh Forouzan
Department of Computer Science
University of California, Irvine
Irvine, CA, 92697

Alexander Ihler
Department of Computer Science
University of California, Irvine
Irvine, CA, 92697

Abstract

Region choice is a key issue for many approximate inference bounds. Mini-bucket elimination avoids the space and time complexity of exact inference by using a top-down partitioning approach that mimics the construction of a junction tree and aims to minimize the number of regions subject to a bound on their size; however, these methods rarely take into account functions' values. In contrast, message passing algorithms often use "cluster pursuit" methods to select regions, a bottom-up approach in which a predefined set of clusters (such as triplets) is scored and incrementally added. In this work, we develop a hybrid approach that balances the advantages of both perspectives, providing larger regions chosen in an intelligent, energy-based way. Our method is applicable to bounds on a variety of inference tasks, and we demonstrate its power empirically on a broad array of problem types.

1 INTRODUCTION

Mini-bucket elimination (MBE) (Dechter and Rish, 2003) is a popular bounding technique for reasoning tasks defined over graphical models. MBE is often used to develop heuristic functions for search and optimization tasks (Dechter and Rish, 2003; Kask and Dechter, 2001; Marinescu and Dechter, 2007; **?**; Marinescu et al., 2014), although it has also been used to provide bounds on weighted counting problems such as computing the probability of evidence in Bayesian networks (Rollon and Dechter, 2010; Liu and Ihler, 2011).

MBE obtains its bounds by approximating the variable or "bucket" elimination process. Rather than exactly eliminating each variable, MBE partitions the functions into smaller sets of bounded complexity; eliminating within each partition separately gives an upper or lower bound. A single control variable allows the user to easily trade off between accuracy and computation (memory and time). A recent extension called weighted mini-bucket (WMB) also serves to connect mini-bucket to the framework of variational bounds, allowing iterative reparameterization updates to improve the WMB bound.

The partitioning of a bucket into mini-buckets of bounded size can be accomplished in many ways, each resulting in a different accuracy. Viewed from a variational perspective, this corresponds to the critical choice of *regions* in the approximations, defining which sets of variables will be reasoned about jointly.

Traditionally, MBE is guided only by the graph structure, using a *scope-based* heuristic (Dechter and Rish, 2003) to minimize the number of buckets. However, this ignores the influence of the functions themselves on the bound. More recent extensions such as Rollon and Dechter (2010) have suggested ways of incorporating the function values into the partitioning process, with mixed success. A more bottom-up construction technique is the *relax-compensate-recover* (RCR) method of Choi and Darwiche (2010), which constructs a sequence of mini-bucket-like bounds of increasing complexity.

Variational approaches typically use a greedy, bottom-up approach termed *cluster pursuit*. Starting with the smallest possible regions, the bounds are optimized using message passing, and then new regions are added greedily from an enumerated list of clusters such as triplets (e.g., Sontag et al., 2008; Komodakis and Paragios, 2008). This technique is often very effective if only a few regions can be added, but the sheer number of regions considered often creates a computational bottleneck and prevents adopting large regions (e.g., Batra et al., 2011).

We propose a hybrid approach that is guided by the graph structure and connects to the mini-bucket construction, but takes advantage of the iterative optimization and scoring techniques of cluster pursuit. In practice, we find that our methods work significantly better than either the partitioning heuristics of Rollon and Dechter (2010), or a pure re-

gion pursuit approach. We also discuss the connections of our work to RCR (Choi and Darwiche, 2010). We validate our approach with experiments on a wide variety of problems drawn from recent UAI approximate inference competitions (Elidan et al., 2012).

2 PRELIMINARIES

Graphical models capture the dependencies among large numbers of random variables by explicitly representing the independence structure of the joint probability distribution. Consider a distribution

$$p(x) = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} f_\alpha(x_\alpha) \qquad Z = \sum_x \prod_\alpha f_\alpha(x_\alpha)$$

where x_α indicates a subset of variables and Z is the normalizing constant called the *partition function*. We associate $p(x)$ with a graph $G = (V, E)$ where each variable x_i is associated with a node $i \in V$ and is connected to x_j if both variables x_i and x_j are arguments of some function f_α. $\mathcal{I}$ is then a set of all cliques in G.

Common inference tasks include finding the most likely or MAP configuration of $p(x)$, a combinatorial optimization problem, or computing the partition function Z, a combinatorial summation problem. Computing Z, which corresponds to the probability of evidence in Bayesian networks, or the marginal probabilities of $p(x)$, are central problems in many learning and inference tasks.

2.1 MINI-BUCKET ELIMINATION

Unfortunately, inference tasks such as computing the partition function are often computationally intractable for many real-world problems. Exact inference, such as variable or "bucket" elimination (Dechter, 1999) is exponential in the tree-width of the model, leading to a spectrum of approximations and bounds subject to computational limits. In this section, we briefly describe bucket elimination and mini-bucket approximations.

Bucket Elimination (Dechter, 1999) is an exact algorithm that directly eliminates variables in sequence. Given an elimination order, BE collects all factors that include variable x_i as their earliest-eliminated argument in a *bucket* B_i, then takes their product and eliminates x_i to produce a new factor over later variables, which is placed in the bucket of its "parent" π_i, the earliest uneliminated variable:

$$\lambda_{i \to \pi_i}(x_{i \to \pi_i}) = \sum_{x_i} \prod_{f_\alpha \in B_i} f_\alpha(x_\alpha) \prod_{\lambda_{j \to i} \in B_i} \lambda_{j \to i}(x_{j \to i})$$

The functions $\lambda_{j \to i}$ constructed during this process can be interpreted as *messages* that are passed downward in a join-tree representation of the model (Ihler et al., 2012); see Figure 1(a)-(b).

The space and time complexity of BE are exponential in the *induced width* of the graph given the elimination order. While good elimination orders can be identified using various heuristics (see e.g., Kask et al., 2011), this exponential dependence often makes direct application of BE intractable for many problems of interest.

Minibucket Elimination. To avoid the complexity of bucket elimination, Dechter and Rish (1997) proposed an approximation in which the factors in bucket B_i are grouped into partitions $Q_i = \{q_i^1, ..., q_i^p\}$, where each partition $q_i^j \in Q_i$, also called a *mini-bucket*, includes no more than $ibound+1$ variables. The bounding parameter $ibound$ then serves as a way to control the complexity of elimination, as the elimination operator is applied to each mini-bucket separately. Using the inequality

$$\sum_{x_i} \prod_{f_\alpha \in B_i} f_\alpha \quad \leq \quad \Big[\sum_{x_i} \prod_{f_\alpha \in q_i^1} f_\alpha\Big] \cdot \Big[\max_{x_i} \prod_{f_\alpha \in q_i^2} f_\alpha\Big],$$

MBE gives an upper bound on the true partition function, and its time and space complexity are exponential in the user-controlled $ibound$. Smaller $ibound$ values result in lower computational cost, but are typically less accurate. See Figure 1(c) for an illustration.

The resulting bound depends significantly on the partitionings $\{Q_i\}$; we discuss strategies for partitioning in Section 2.2.

Weighted Mini-bucket. A recent improvement to mini-bucket (Liu and Ihler, 2011) generalizes the MBE bound with a "weighted" elimination,

$$\sum_{x_i} \prod_{f_\alpha \in B_i} f_\alpha \leq \Big[\sum_{x_i} \prod_{f_\alpha \in q_i^1} f_\alpha^{\frac{1}{w_1}}\Big]^{w_1} \cdot \Big[\sum_{x_i} \prod_{f_\alpha \in q_i^2} f_\alpha^{\frac{1}{w_2}}\Big]^{w_2},$$

where $w_i > 0$ and $w_1 + w_2 = 1$.

Liu and Ihler (2011) also show that the resulting bound is equivalent to a class of bounds based on tree reweighted (TRW) belief propagation (Wainwright et al., 2005), or more generally conditional entropy decompositions (CED) (Globerson and Jaakkola, 2007), on a join-graph defined by the mini-bucket procedure (see Figure 1(d)). This connection is used to derive fixed point reparameterization updates, which change the relative values of the factors f_α while keeping their product constant in order to tighten the bound.

2.2 PARTITIONING METHODS

As discussed above, mini-bucket elimination and its weighted variant compute a partitioning over each bucket B_i to bound the complexity of inference and compute an upper bound on the partition function Z. However, different partitioning strategies will result in different upper

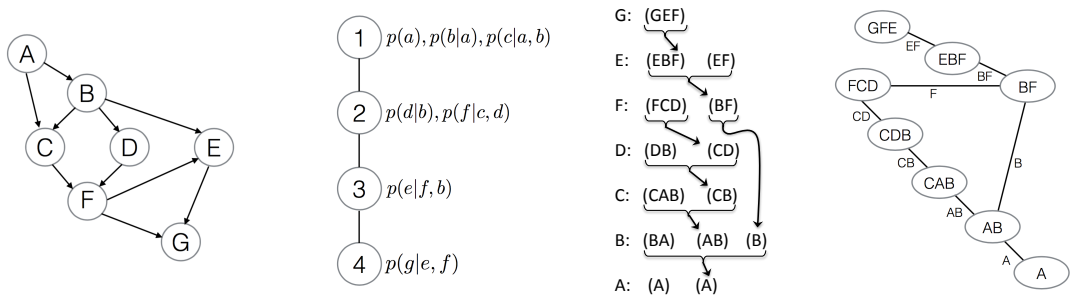

Figure 1: (a) A belief network $P(A, B, C, D, E, F, G) = P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|B) \cdot P(F|C, D) \cdot P(E|B, F) \cdot P(G|E, F)$; (b) a join-tree decomposition for exact inference; (c) a mini-bucket approximation ($ibound = 2$), with F eliminated approximately; (d) the region or join-graph associated with (c).

bounds. Rollon and Dechter (2010) proposed a framework to study different partitioning heuristics, and compared them with the original scope based heuristic proposed by Dechter and Rish (1997). Here we summarize several approaches.

Scope-based Partitions. Proposed in Dechter and Rish (1997), scope-based partitioning is a top-down approach that tries to minimize the number of mini-buckets in B_i by including as many functions as possible in each mini-bucket q_i^k. To this end, it first orders the factors in B_i by decreasing number of arguments. Starting from the largest, each factor f_α is then merged with the first available mini-bucket that satisfies the computational limits, i.e., where $|\text{var}(f) \cup \text{var}(q_i^j)| \leq ibound + 1$. If there are no mini-buckets available that can include the factor, a new mini-bucket is created and the scheme continues until all factors are assigned to a mini-bucket.

Content-based Partitions. Rollon and Dechter (2010), on the other hand, seeks to find a partitioning that is closest to the true bucket function, $g_i = \sum_{X_i} \prod_{\alpha \in B_i} f_\alpha$. This requires solving an optimization problem

$$Q* = \arg\min_Q dist(g_i^Q, g_i)$$

where Q is a partitioning of B_i with bounding parameter $ibound$ and

$$g_i^Q = \prod_{j=1}^{p} \sum_{X_i} \prod_{\alpha \in q_i^j} f_\alpha$$

is the function represented by the partitioning Q. The distance is minimized in a greedy fashion, and Rollon and Dechter (2010) studied the effectiveness of several different distance functions across multiple problem instances; however, no one distance was found to consistently outperform scope-based partitioning.

Relax-Compensate-Recover. Choi and Darwiche (2010) indirectly addresses the problem of partition selection within their *Relax, Compensate and Recover* framework, in which certain equality constraints in the graph are first relaxed in order to reduce complexity of inference. New auxiliary factors are then introduced to compensate for the relaxation and enforce a weaker notion of equivalence. The recovery process then aims to identify those equivalence constraints whose relaxation were most damaging and recover them. Choi and Darwiche (2010) proposed a number of recovery heuristics, including mutual information and residual recovery.

2.3 VARIATIONAL BOUNDS.

The variational viewpoint of inference corresponds to optimizing an objective function over a collection of *beliefs* constrained to lie within the marginal polytope, or set of marginal probabilities that can be achieved by some joint distribution. Efficient approximations are developed by relaxing these constraints to enforce only a subset of the constraints – that the beliefs be consistent between overlapping cliques. In the case of the log partition function, we also approximate the entropy term in the objective; for example, the CED bound is:

$$\log Z \leq \max_{b_\alpha \in \mathbb{L}} \sum_\alpha \mathbb{E}_{b_\alpha}[\log f_\alpha] + \sum_{i,\alpha} w_{i\alpha} H(x_i | x_{\alpha \setminus i}\,;\, b_\alpha)$$

where $\sum_\alpha w_{i\alpha} = 1$ for all i.

Like mini-bucket bounds, the quality of variational bounds depends significantly on the choice of regions, which determine what constraints will be enforced by the local polytope $\mathbb{L}$ as well as the form of the entropy approximation. Traditionally, variational approximations have focused more on the optimization of the bound through message passing than the region selection aspect. Often regions are chosen to match the original model factors, and then improved using methods like cluster pursuit, described next.

Cluster Pursuit. Sontag et al. (2008) studied the problem of region selection for MAP inference in the context of *cluster-based* dual decomposition relaxations. They developed a bottom-up approach in which regions (typically

cycles or triplets) are added incrementally: First, the dual decomposition bound is optimized through message passing. Then, a pre-defined set of clusters, such as triplets or faces of a grid, are scored by computing a lower bound on their potential improvement of the bound; the scoring function used measures the difference between independently maximizing each pairwise factor, versus jointly maximizing over the triplet. After adding the best-scoring cluster, the procedure repeats. Similar cycle repair processes were also proposed by Komodakis and Paragios (2008) and Werner (2008), and related cluster pursuit methods have also been applied to summation problems (Welling, 2004; Hazan et al., 2012). However, scoring all possible clusters often becomes a computational bottleneck; for example, Batra et al. (2011) proposed pre-selection heuristics to reduce the number of clusters considered. In practice, cluster pursuit is usually applied to add only a few, small regions; scoring sets of larger regions is typically considered prohibitive.

3 A HYBRID APPROACH

Mini-bucket elimination avoids the space and time complexity of exact inference by using a top-down partitioning approach that mimics the construction of a junction tree. In contrast, message passing algorithms often use "cluster pursuit" methods to select regions, a bottom-up approach in which a predefined set of clusters (such as triplets) is scored and incrementally added.

To balance the effectiveness of both approaches, our hybrid scheme, like mini-bucket, uses the graph structure to guide region selection, while also taking advantage of the iterative optimization and scoring techniques of cluster pursuit.

Cluster pursuit algorithms use the function values, and more concretely the bound produced by them, in order to select regions that tighten the upper bound more effectively. However, there are often prohibitively many clusters to consider: for example, in a fully connected pairwise model, there are $O(n^3)$ triplets, $O(n^4)$ possible 4-cliques, etc., to score at each step. For this reason, cluster pursuit methods typically restrict their search to a predefined set of clusters, such as triplets Sontag et al. (2008). Our proposed approach uses the graph structure to guide the search for regions, restricting the search to merges of existing clusters, within one bucket at a time. This allows us to restrain the complexity of the search and add larger regions more effectively.

In contrast, the content-based heuristics for region selection of Rollon and Dechter (2010) use the graph structure as a guide, but their scoring scheme only takes into account the messages from the earlier buckets in the elimination order. Our proposed hybrid approach uses iterative optimization on the junction tree in order to make more effective partitioning decisions.

Algorithm 1 Incremental region selection for WMBE

Input: factor graph $(\mathcal{G})$, bounding parameter $ibound$ and maximum number of iterations T
Initialize wmb to a join graph using e.g. a min-fill ordering o, uniform weights and uniform messages
for each bucket B_i following the elimination order **do**
  repeat
    $(q_i^m, q_i^n) \leftarrow$ SelectMerge(Q_i)
    $\mathcal{R} \leftarrow$ AddRegions(wmb, o, q_i^m, q_i^n)
    $wmb \leftarrow$ MergeRegions$(wmb, \mathcal{R})$
    for $iter = 1$ **to** T **do**
      // pass forward messages and reparameterize:
      $wmb \leftarrow$ msgForward(wmb)
      // pass backward messages:
      $wmb \leftarrow$ msgBackward(wmb)
    end for
  until no more merges possible
end for

Algorithm 1 describes the overall scheme of our hybrid approach, which is explained in detail next.

3.1 INITIALIZING A JOIN TREE

Given a factor graph G and a bounding parameter $ibound$, we start by initializing a join graph, using a min-fill elimination ordering (Dechter, 2003) $\mathbf{o} = \{x_1, ..., x_n\}$ and $ibound = 1$. For any given bucket B_i, this results in each mini-bucket $k \in B_i$ containing a single factor f_α. We denote the result of the elimination as $\lambda_{k\to l}$ which is sent to the bucket B_j of its first-eliminated argument in $\mathbf{o}$. Here, $l = \text{pa}(k)$ denotes the parent region of k which can be one of the initial mini-buckets in B_j if $var(\lambda_{k\to l}) \subseteq var(l)$, or be a new mini-bucket with $f_l = 1$. In our implementation we choose $l \in B_j$ to be the mini-bucket with the largest number of arguments,$|var(l)|$, such that $var(\lambda_{k\to l}) \subseteq var(l)$.

Using weighted mini-bucket for our elimination scheme, we initialize the mini-bucket weights w_r uniformly within each bucket B_i, so that for $r \in Q_i$, $w_r = \frac{1}{|Q_i|}$, which ensures $\sum_{r\in Q_i} w_r = 1$.

3.2 MESSAGE PASSING

We use iterative message passing on the join graph to guide the region selection decision. Having built an initial join graph, we use the weighted mini-bucket messages (Liu and Ihler, 2011) to compute forward and backward messages, and perform reparameterization of the functions f_α to tighten the bound.

Let r be a region of the mini-bucket join graph, and s its parent, $s = \text{pa}(r)$, with weights w_r and w_s, and $f_r(x_r$ the product of factors assigned to region r. Then we compute

the forward messages as,

Forward Messages:

$$\lambda_{r\to s}(x_s) = \Big[\sum_{x_r \setminus x_s} \big[f_r(x_r) \prod_{t:s=\text{pa}(t)} \lambda_{t\to s}(x_s) \big]^{\frac{1}{w_s}} \Big]^{w_s} \tag{1}$$

and compute the upper bound using the product of forward messages computed at roots of the join graph,

Upper bound on Z:

$$Z \leq \prod_{r:\text{pa}(r)=\varnothing} \lambda_{r\to\varnothing} \tag{2}$$

In order to tighten the bound, we compute backward messages in the join graph,

Backward Messages:

$$\lambda_{s\to r}(x_r) = \Big[\sum_{x_s \setminus x_r} \big[f_s(\cdot) \prod_t \lambda_{t\to s}(\cdot) \big]^{\frac{1}{w_s}} \big[\lambda_{r\to s}(\cdot) \big]^{-\frac{1}{w_r}} \Big]^{w_r}$$

where t indexes all neighbors (parent and children) of region s. We then use these incoming messages to compute a weighted belief at region r, and reparameterize the factors f_r for each region r in a given bucket B_i (e.g., $r \in Q_i$) to enforce a weighted moment matching condition:

Reparameterization:

$$b_r(x_i) = \sum_{x_r \setminus x_i} \big[f_r(x_r) \prod_t \lambda_{t\to r}(x_r) \big]^{\frac{1}{w_r}}$$
$$\bar{b}(x_i) = \big[\prod_{r\in Q_i} b_r(x_i) \big]^{1/\sum_r w_r}$$
$$f_r(x_r) \leftarrow f_r(x_r) \big[\bar{b}(x_i) / b_r(x_i) \big]^{w_r}$$

In practice, we usually match on the variables present in all mini-buckets r, e.g., $\cap_{r\in Q_i} x_r$, rather than just x_i; this gives a tighter bound for the same amount of computation.

3.3 ADDING NEW REGIONS

New regions are added to the initial join tree after one or more rounds of iterative optimization. To contain the complexity of the search over clusters, we restrict our attention to pairs of mini-buckets to merge within each bucket. To do so, we also use the elimination order o to guide our search, processing each bucket B_i one at a time in order.

Given bucket B_i and current partitioning $Q_i = \{q_i^1, ..., q_i^k\}$, we score the merge for each allowed pair of mini-buckets (q_i^m, q_i^n), e.g., those with $|\text{var}(q_i^m) \cup \text{var}(q_i^n)| \leq ibound + 1$, using an estimate of the benefit to the bound that may arise from merging the pair:

$$S(q_i^m, q_i^n) = \max_x \log \left[\lambda_{m\to\pi_m}(x_{\pi_m}) \times \lambda_{n\to\pi_n}(x_{\pi_n}) \div \lambda_{r\to\pi_r}(x_r) \right]$$

This score can be justified as a lower bound on the decrease in $\log Z$, since it corresponds to adding region π_r with weight $w_{\pi_r} = 0$, while reparameterizing the parents π_m, π_n to preserve their previous beliefs. This procedure leaves the bound unchanged except for the contribution of π_r; eliminating with $w_{\pi_r} = 0$ is equivalent to the $\max$ operation. For convenience, we set $S(q_i^m, q_i^n) = 0$ for pairs which violate the $ibound$ constraint. Then, having computed the score between all pairs, we choose the pair with maximum score to be merged into a new clique. In Algorithm 1, the function SelectMerge$(\cdot)$ denotes this scoring and selection process.

3.4 UPDATING GRAPH STRUCTURE

Having found which mini-buckets to merge, we update the join graph to include the new clique $r = q_i^m \cup q_i^n$. Our goal is to add the new region such that it affects the scope of the existing regions in the join tree as little as possible. Adding the new clique is done in two steps:

First we initialize a new mini-bucket in B_i with its scope matching var(r). Eliminating variable x_i from this new mini-bucket results in the message $\lambda_{r\to\pi_r}$. The earliest argument of $\lambda_{r\to\pi_r}$ in the elimination order determines the bucket B_j containing mini-buckets that can potentially be the parent, π_r, of the new region. To find π_r in B_j we seek a mini-bucket q_j^k that can contain r, i.e., var$(\lambda_{r\to\pi_r}) \subseteq$ var(q_j^k). If such a mini-bucket exists, we set π_r to q_j^k; otherwise, we create a new mini-bucket $q_j^{|Q_j|+1}$ and add it to Q_j, with a scope that matches var$(\lambda_{r\to\pi_r})$. The same procedure is repeated after eliminating x_j from $q_j^{|Q_j|+1}$ until we either find a mini-bucket already in the join tree that can serve as the parent, or var$(\lambda_{r\to\pi_r}) = \varnothing$ in which case the newly added mini-bucket is a root. Algorithm 2 describes these initial structural modifications.

Having added the new regions, we then try to remove any unnecessary mini-buckets, and update both the join tree and the function values of the newly added regions to ensure that the bound is improved. To this end, we update every new mini-bucket r that was added to the join tree in the previous step as follows. For mini-bucket $r \in Q_i$, we first find any mini-buckets $s \in Q_i$ that can be subsumed by r, i.e., var$(s) \subseteq$ var(r). For each of these mini-buckets s, we connect all of s's children (mini-buckets t such that pa$(t) = s$) to r, e.g., set pa$(t) = r$. We also merge the factors associated with r and s, so that $f_r \leftarrow f_r \times f_s$.

Next, we reparameterize several other functions in the join graph in order to preserve or improve the current bound

Algorithm 2 AddRegions: find regions to add for merge

Input: The join graph wmb, elimination order o, and mini-buckets q_i^m and q_i^n to be merged
Output: a list of newly added mini-buckets $\mathcal{R}$
Initialize new region q^r with $\text{var}(q^r) = \text{var}(q_i^m \cup q_i^n)$ and add it to Q_i
repeat
  Update $\mathcal{R} = \mathcal{R} \cup q^r$
  Set new clique $C = \text{var}(q^r) \backslash x_i$
  if $C = \varnothing$ **then**
    $done \leftarrow True$
  else
    Find B_j corresponding to the first un-eliminated variable in C based on elimination order o
    for each mini-bucket region $q_j^k \in Q_j$ **do**
      if $C \subseteq \text{var}(q_j^k)$ **then**
        // forward message fits in existing mini-bucket:
        $done \leftarrow True$
      end if
    end for
  end if
  if not $done$ **then**
    // Create a new region to contain forward message:
    Initialize new region q^r with $\text{var}(q^r) = C$ and add it to Q_j
  end if
until done

Algorithm 3 MergeRegions: merge and parameterize newly added regions to improve bound

Input: The join graph wmb and a list of newly added mini-buckets $\mathcal{R}$
for all $r \in \mathcal{R}$ **do**
  Initialize new region r in B_i with $f_r(x_r) = 1$
  Find regions $\{s \mid s \in Q_i \,\&\, \text{var}(s) \subseteq \text{var}(s)\}$
  // Remove / merge contained regions s:
  for all found regions s **do**
    Connect all children of s to r
    $f_r = f_r \cdot f_s$ *// merge factors and*
    // preserve belief at parent π_s:
    $f_{\pi_s} = f_{\pi_s} \times \lambda_{s \to \pi_s}$
    $f_{\pi_r} = f_{\pi_r} \div \lambda_{s \to \pi_s}$
    Remove s from Q_i
  end for
end for

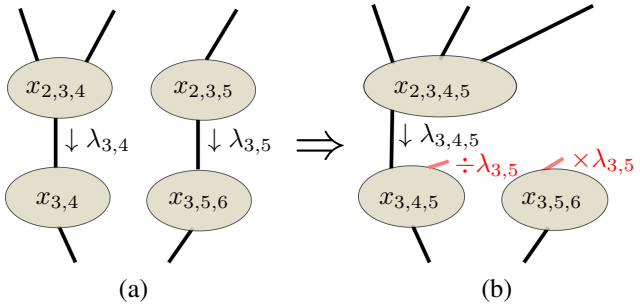

Figure 2: Merge and post-merge reparameterization operations. (a) A portion of a join-graph corresponding to the elimination of x_2 and x_3, each with two mini-buckets. (b) Merging cliques $(2,3,4)$ and $(2,3,5)$ produces a new clique $(3,4,5)$, which subsumes and removes clique $(3,4)$. Having removed parent $(2,3,5)$, we reparameterize the new clique functions by the original message $\lambda_{3,5}$ (red) to preserve the original belief at $(3,5,6)$ and ensure that the bound is tightened. See text for more detail.

value. Specifically, removing s changes the incoming, forward messages to its parent, $\pi_s = \text{pa}(s)$, which changes the bound. By reparameterizing the factor at π_s,

$$f_{\pi_s} \leftarrow f_{\pi_s} \times \lambda_{s \to \pi_s} \qquad f_{\pi_r} \leftarrow f_{\pi_r} \div \lambda_{s \to \pi_s}$$

we keep the overall distribution unchanged, but ensure that the bound is strictly decreased.

Finally we remove s from Q_i, completing the merge of mini-buckets s and r. This process is given in Algorithm 3 and depicted in Figure 2 for a small portion of join-graph.

Every merge decision is followed by one or more iterations of message passing, followed by rescoring the mini-buckets in B_i. The process of message passing and merging continues until no more mini-buckets of B_i can be merged, while satisfying the bounding parameter $ibound$.

Continuing along the elimination order, the same procedure is repeated for the mini-buckets in each bucket B_i, and the final upper bound to the partition function is computed using Eq. (2).

4 DISCUSSION

Our method is similar to context-based mini-buckets, with the main difference being that message passing performed on the simpler graph is used to reparameterize the functions before the merge scores are computed.

Our method can also be viewed as a cluster pursuit approach, in which we restrict the clusters considered, to unions of the current minibuckets at the earliest bucket B_i, and merge up to our computational limit before moving on to later buckets. These restrictions serve to reduce the number of clusters considered, but in addition, appear to lead to better regions than a purely greedy region choice – in the experiments (Section 5), we compare our approach to a more "cluster pursuit-like" method, in which pairs of regions in *any* bucket B_i are considered and scored. Perhaps surprisingly, we find that this greedy approach actually gives significantly worse regions overall, suggesting

that processing the buckets in order can help by avoiding creating unnecessary regions.

Finally, our method is also closely related to RCR (Choi and Darwiche, 2010). From this perspective, we "relax" to a low-$ibound$ minibucket, "compensate" by variational message passing, and "recover" by selecting regions that will tighten the variational bound defined by the join graph. Compared to RCR, we find a number of differences in our approach: (1) RCR selects constraints to recover anywhere in the graph, similar to a greedy cluster pursuit; as noted, this appears to work significantly less well than an ordered recovery process. (2) RCR makes its recovery updates to the relaxed graph, then (re)builds a (new) join tree over the relaxed graph; in contrast, we incrementally alter the join graph directly, which avoids starting from scratch after each merge. (3) Our method is solidly grounded in the theory of variational bounds and message passing, ensuring that both the message passing and region merging steps are explicitly tightening the same bound. From this perspective, for example, it becomes clear that RCR's "residual recovery" heuristic is unlikely to be effective, since after message passing, the reparameterization updates should ensure that all mini-buckets containing a variable x_i will match on their marginal beliefs. In other words, residual recovery is making its structure (region) choices using a criterion that actually measures mismatches that can be resolved by message passing.

5 EMPIRICAL EVALUATION

To show our method's effectiveness compared to previous region selection strategies for MBE, we tested our incremental approach on a number of real world problems drawn from past UAI approximate inference challenges, including linkage analysis, protein side chain prediction, and segmentation problems. We compare our hybrid region selection method against the scope-based heuristic of Dechter and Rish (1997) and the content-based heuristic of Rollon and Dechter (2010).

Experimental Setup. For each set of experiments, we initialize a join tree using WMB elimination with $ibound = 1$. We use an elimination ordering found using the min-fill heuristic (Dechter, 2003) and set the weights uniformly in each bucket. As a result, each mini-bucket q_i^k contains a single factor f_α as described in section 3.1.

From this initial setup, we then use Algorithm 1 to merge mini-buckets incrementally and compute the upper bound as in Eq. (2).

Segmentation. To evaluate the different methods on pairwise binary problems we used a set of segmentation models from the UAI08 approximate inference challenge. These models have ≈ 230 binary variables and ≈ 850 factors. We used varying ibounds for comaprison and report the results on two values, $ibound \in [5, 10]$. Table 1 compares the upper bound on the log partition function for a representative subset of instances in this category, for two different computational limits, $ibound = 5$ and $ibound = 10$. Different columns show the bound achieved using different partitioning heuristics:
(1) **Scp** represents naïve scope-based partitioning;
(2) **Cont** represents the energy based heuristic of Rollon and Dechter (2010); and
(3) **Hyb** represents our hybrid approach, interleaving iterative optimization with partitioning.

The results show clear improvement in the upper bound using our hybrid approach, indicating the effectiveness of iterative message passing and optimization in guiding region selection. To further study the effectiveness of the merged regions in the context of message passing and optimization, we then fully optimized the join-graphs generated by the three region selection schemes using iterative message passing until convergence. The upper bounds after such optimization are denoted by *inst-opt* for each problem instance, *inst*. As might be expected, this additional optimization step improves the bounds of the scope-based and content-based heuristics more dramatically than our hybrid method; however, even after full optimization of the bounds, we find that the hybrid method's bounds remain better in all of the 6 instances except one, indicating that our method has identified fundamentally better regions than the previous approaches.

Linkage Analysis. To compare the various methods on models with non-pairwise factors and higher cardinalities of variables, we studied pedigree models. The pedigree linkage analysis models from the UAI08 approximate inference challenge have $\approx 300 - 1000$ variables, whose cardinalities vary in the range of $[2, ..., 5]$; the induced width of the models are typically $\approx 20 - 30$. We used varying ibounds for comaprison and report the results on two values $ibound \in [5, 10]$.

Table 2 shows the upper bounds on a subset of pedigree problems, again showing the effectiveness of the hybrid method: we find that again, the hybrid method consistently outperforms the other two region selection approaches, and results in better fully optimized bounds in all of the 22 instances when $ibound = 5$ and all but two cases when $ibound = 10$.

Effect of ibound. We further studied the results of the three partitioning methods across a range of ibounds to compare the effectiveness of our method when ibound is set to a range of values. Figure 3 shows the results for an instance of pedigree dataset. As shown here, our method is more effective on smaller ibounds, where there are a large number of possible merges and finding the best one results

Table 1: **UAI Segmentation Instances.** Different columns show the bound achived using each partitioning heuristic, where "Scp", "Cont" and "Hyb" represent the naïve scope based partitioning for MBE (Dechter and Rish, 1997), the context (or energy) based heuristic of Rollon and Dechter (2010) and our hybrid approach interleaving iterative optimization with partitioning, respectively. In all but one case, our proposed construction provides tighter bounds.

	$ibound = 5$			$ibound = 10$		
Instance	**Scp**	**Cont**	**Hyb**	**Scp**	**Ctxt**	**Hyb**
2-17-s	-31.3197	-33.4840	**-49.5670**	-38.9801	-42.1524	**-52.507**
2-17-s-*opt*	-46.9314	-45.4286	**-49.6432**	-48.661	-48.4306	**-52.5633**
8-18-s	-54.9884	-60.9899	**-85.6518**	-72.3045	-72.284	**-87.2385**
8-18-s-*opt*	-80.6527	-81.1391	**-85.6694**	-83.0398	-79.0921	**-87.2556**
9-24-s	-51.2897	-49.3903	**-55.6046**	-54.9325	-55.0151	**-55.615**
9-24-s-*opt*	**-56.0513**	-53.7852	-55.6241	-54.9325	-55.0151	**-55.615**
17-4-s	-58.7953	-59.0758	**-81.5415**	-80.267	-79.3323	**-85.3712**
17-4-s-*opt*	-71.7959	-76.8213	**-81.6079**	-83.2573	-82.9646	**-85.3865**
7-11-s	-59.8250	-57.2773	**-72.7178**	-71.1296	-70.1542	**-75.2869**
7-11-s-*opt*	-70.7037	-68.8255	**-72.9556**	-74.6424	-73.9855	**-75.2905**

Table 2: **UAI Pedigree Instances.** Different columns show the bound achieved using each partitioning heuristic; again, "Scp", "Cont" and "Hyb" are scope based partitioning (Dechter and Rish, 1997), the context-based heuristic (Rollon and Dechter, 2010) and our proposed, hybrid approach. In all cases, our proposed construction provides stronger bounds, both before and after full optimization using message passing.

	$ibound = 5$			$ibound = 10$		
Instance	**Scp**	**Cont**	**Hyb**	**Scp**	**Ctxt**	**Hyb**
ped23	-67.8848	-69.9015	**-71.9677**	-75.6057	-78.4033	**-79.4649**
ped23-*opt*	-71.6951	-71.6988	**-72.0670**	-76.0531	-78.7646	**-79.4669**
ped20	-35.6986	-40.1787	**-44.7230**	-51.2648	-54.4136	**-57.6506**
ped20-*opt*	-42.3024	-42.8980	**-44.7501**	-52.6043	-56.2193	**-57.7841**
ped42	-41.6656	-43.5206	**-51.0000**	-55.0681	-57.5755	**-61.3504**
ped42-*opt*	-49.0089	-50.0585	**-51.1018**	-57.3718	-59.2170	**-61.3560**
ped38	-79.4742	-89.6906	**-92.7643**	-98.6339	-101.1178	**-113.6004**
ped38-*opt*	-83.0351	-91.8510	**-93.0615**	-101.0715	-104.1031	**-113.8926**
ped19	-58.9234	-63.2737	**-80.6488**	-90.7840	-93.9027	**-100.3230**
ped19-*opt*	-72.3311	-77.7023	**-80.7167**	-92.8916	-96.2846	**-100.3388**

in a greater improvement to the upper bound. For larger ibounds, the upper bounds produced by all three heuristics are fairly close.

Protein Side-Chain Prediction. Finally, to examine models over high-cardinality variables, we look at a subset of the protein side chain prediction models, originally from Yanover and Weiss (2003) and Yanover et al. (2006). These models contain $\approx 300 - 1000$ variables with cardinalities between 2 and 81, with pairwise potential functions. For these problems, we only ran our experiments using $ibound = 2$, due to the high number of states for each variable. Table 3 shows the results of the three partitioning methods, which again agrees with the previous experiments: our hybrid method outperforms the other two in all 44 instances in this problem set, both before and after the bound is fully optimized.

Greedy vs. Elimination Order Based Merging. As discussed before, we restrict the clusters considered for merges to unions of the current minibuckets at the earliest bucket B_i, and merge up to our computational limit be-

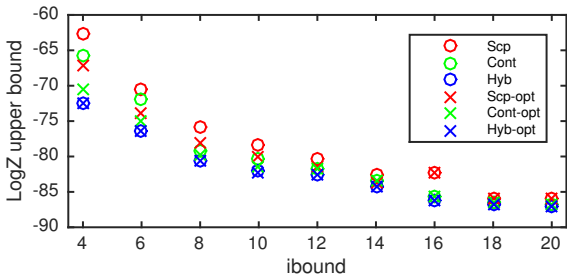

Figure 3: The upper bound achieved by the three partitioning heuristics for pedigree23 instance over ibound range between 4 to 20.

Table 3: **Protein side-chain prediction.** Here we show results for only $ibound = 2$, due to the high number of states in each variable. Our method often produces dramatically better partitionings than scope- or context-based mini-bucket partitions.

Instance	Scp	Cont	Hyb
1crz	-242.2036	-284.865	**-451.598**
1crz -*opt*	-528.5348	-495.514	**-545.929**
2cav	71.2802	-26.0052	**-148.637**
2cav -*opt*	-156.5387	-240.289	**-272.606**
1kk1	89.5527	46.8216	**-121.723**
1kk1 -*opt*	-115.4737	-105.894	**-143.447**
1e4f	40.6686	-6.1785	**-190.943**
1e4f -*opt*	-212.4607	-202.547	**-240.27**
1ehg	71.3308	14.768	**-149.158**
1ehg -*opt*	-169.8435	-147.333	**-211.678**

fore moving on to later buckets, which serves to reduce the number of clusters considered. We compare our choice of clusters with a purely greedy region choice in which pairs of regions in *any* bucket B_i are considered and scored.

Interestingly the upper bounds achieved using the greedy approach was not better than the top down merging based on elimination order. The reason for this behavior is that the top-down approach allows large regions generated by mini-buckets early in the elimination ordering to be processed by buckets later in the order; the greedy approach disrupts this flow and results in extra regions that cannot be merged with any other region while respecting the $ibound$.

6 CONCLUSION

We presented a new merging heuristic for (weighted) mini-bucket elimination that uses message passing optimization of the bound, and variational interpretations, in order to construct a better heuristic for selecting moderate to large regions in an intelligent, energy-based way. Our approach inherits the advantages of both cluster pursuit in variational inference, and (weighted) mini-bucket elimination perspectives to produce a tight bound. We validated our approach with experiments on a wide variety of problems drawn from a recent UAI approximate inference competition. In practice, we find that our methods work significantly better than either existing partitioning heuristics for mini-bucket (Rollon and Dechter, 2010), or a pure region pursuit approach. We expect this construction to improve our ability to search and solve large problems. However, our method does involve additional computational overhead compared to, say, scope-based constructions, in order to to evaluate and make merge decisions. We did not focus here on any-time performance; a more nuanced balance of time, memory, and bound quality is one direction of potential future study.

Table 4: **Top-down vs. Greedy Merging.** We examine the effect of using a "fully greedy" merging procedure closer to standard cluster pursuit, in which we merge the best-scoring cluster in any bucket at each step. We find that following the top-down ordering actually results in significantly better bounds. Results shown are for $ibound = 5$.

Instance	Top-Down	Greedy
ped23	**-71.9677**	-67.9094
ped23-*opt*	**-72.0670**	-67.9094
ped20	**-44.7230**	-38.0717
ped20-*opt*	**-44.7501**	-38.0718
ped42	**-49.9955**	-37.8576
ped42-*opt*	**-50.0469**	-37.8582
ped38	**-92.7643**	-79.9144
ped38-*opt*	**-93.0615**	-79.9144
ped19	**-80.6488**	-48.6900
ped19-*opt*	**-80.7167**	-48.6904

Acknowledgements

This work is supported in part by NSF grants IIS-1065618 and IIS-1254071, and by the United States Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program.

References

D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for map-mrf inference: A local primal-dual gap based separation algorithm. *JMLR - Proceedings Track*, 15: 146–154, 2011.

Arthur Choi and Adnan Darwiche. Relax, compensate and then recover. In *New Frontiers in Artificial Intelligence - JSAI-isAI 2010 Workshops, LENLS, JURISIN, AMBN,*

ISS, Tokyo, Japan, November 18-19, 2010, Revised Selected Papers, pages 167–180, 2010.

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

Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(12):41 – 85, 1999.

Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1558608907.

Rina Dechter and Irina Rish. A scheme for approximating probabilistic inference. In *Proc. Uncertainty in Artificial Intelligence (UAI)*, pages 132–141, 1997.

G. Elidan, A. Globerson, and U. Heinemann. PASCAL 2011 probabilistic inference challenge. http://www.cs.huji.ac.il/project/PASCAL/, 2012.

A. Globerson and T.S. Jaakkola. Approximate inference using conditional entropy decompositions. In *In Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, 2007.

T. Hazan, J. Peng, and A. Shashua. Tightening fractional covering upper bounds on the partition function for high-order region graphs. In *Uncertainty in Artificial Intelligence*, 2012.

Alexander Ihler, Natalia Flerova, Rina Dechter, and Lars Otten. Join-graph based cost-shifting schemes. In *Uncertainty in Artificial Intelligence (UAI)*. August 2012.

K. Kask and R. Dechter. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence*, 129(1-2):91–131, 2001.

Kalev Kask, Andrew Gelfand, Lars Otten, and Rina Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *AAAI'11*, pages –1–1, 2011.

N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. pages 806–820, 2008.

Qiang Liu and Alexander Ihler. Bounding the partition function using hölder's inequality. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 849–856, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

R. Marinescu and R. Dechter. Best-first and/or search for most probable explanations. In *Uncertainty in Artificial Intelligence (UAI)*, 2007.

R. Marinescu, R. Dechter, and A. Ihler. AND/OR search for marginal MAP. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 563–572, 2014.

Emma Rollon and Rina Dechter. Evaluating partition strategies for mini-bucket elimination. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2010), Fort Lauderdale, Florida, USA, January 6-8, 2010*, 2010.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. In *Uncertainty in Artificial Intelligence*, pages 503–510, 2008.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. 51 (7):2313–2335, July 2005.

M. Welling. On the choice of regions for generalized belief propagation. In *Uncertainty in Artificial Intelligence*, pages 585–592, 2004.

T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimization (map-mrf). In *Computer Vision and Pattern Recognition*, 2008.

Chen Yanover and Yair Weiss. Approximate inference and protein-folding. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1457–1464. MIT Press, Cambridge, MA, 2003.

Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation - an empirical study. *Journal of Machine Learning Research*, 7: 1887–1907, 2006.