CHAPTER 9

Bounding Inference by Iterative message-passing schemes

As noted, the BTE algorithm, when applied to singly-connected problems becomes a very efficient propagation algorithm that solves the queries (marginal or mpe) exactly. This is why it is called belief propagation. The algorithm is well defined even if the graph is not tree-structured. When the graphical model is not tree-structured, then BP is no longer exact. If we initialize all of the messages in our system to some fixed value, e.g. the value "1", $\lambda_{i\to j} = 1$, then the message updates are still valid. Iteratively applying these message updates on the dual graph yields the approximate inference method known as Loopy Belief Propagation (BP, or Iterative Belied propagation (IBP). IBP is an iterative method that is not guaranteed to converge. And, when the algorithm does converge, the resulting beliefs may poorly approximate the true marginals, in the case of the marginal inference task, or give a poor approximation to the mpe assignment, in the case of the mpe inference task. However, despite this lack of guarantees the algorithms performs quite well in many cases (e.g., coding networks) and in other cases.

This section discusses Iterative bounded inference algorithms such as iterative belief propagation and iterative Join-Graph Propagation. One motivation for designing this algorithm is to combine the anytime feature of Mini-Clustering (MC) and the iterative virtues of Iterative Belief Propagation (IBP).

9.1 ITERATIVE JOIN-GRAPH PROPAGATION

Mini-clustering is partially an anytime algorithm because it is not anyspace. Namely it is limited by the available memory. It works on tree-decompositions and it converges in two passes, so iterating doesn't change the messages. IBP is an iterative algorithm that converges in many cases, and when it converges it does so very fast. Allowing it more time doesn't improve its accuracy. The immediate question is if we can exploit the anytime property of MC obtained using its i-bound, with the iterative qualities of IBP. Algorithm Iterative Join-graph Propagation (IJGP) was designed to benefit from both these directions. It works on a general join-graph which may contain cycles. The cluster size of the graph is user adjustable by the *i-bound*, and the cycles in the graph allow iterating.

The algorithm applies message computation over a join-graph decomposition, which has all the ingredients of a join-tree, except that the underlying graph may have cycles.

Definition 9.1 join-graph decompositions. A *join-graph decomposition* for $BN =< X, D, G, P >$ is a triple $D =< JG, \chi, \psi >$, where $JG = (V, E)$ is a graph, and χ and ψ are labeling functions which associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ such that:

1. For each $p_i \in P$, there is *exactly* one vertex $v \in V$ such that $p_i \in \psi(v)$, and $scope(p_i) \subseteq \chi(v)$.
2. (connectedness) For each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected subgraph of G, a property also called the running intersection property.

We will refer to a node and its CPT functions as a *cluster* (note that a node may be associated with an empty set of CPTs) and use the term *join-graph-decomposition* and *cluster graph* interchangeably. A *join-tree-decomposition* or a *cluster tree* is the special case when the join-graph *JG* is a tree.

It is clear that one of the problems of message propagation over cyclic join-graphs is *over-counting*. Various schemes were proposed to overcome this problem. We will describe here a scheme that avoids cycles relative to a single variable. Recall that in a dual graph some arcs may be redundant and can be removed, even reducing to a join-tree. We next define the notion arc-minimality that captures this concept.

Definition 9.2 arc-minimality. A join-graph decomposition D is *arc-minimal* if none of its arcs can be removed while still satisfying the connectedness property of Definition 2.1.

If a join-graph decomposition is not arc-minimal it is easy to remove some of its arcs until it becomes arc-minimal. The property of arc-minimality is *not* sufficient to ensure such acyclicity though. What is required is that, for every variable X, the arc-subgraph that contains X be a tree.

Example 9.3 The example in Figure 2.1a shows an arc minimal join-graph which contains a cycle relative to variable 4, with arcs labeled with separators. Notice however that if we remove variable 4 from the label of one arc we will have no cycles (relative to single variables) while the connectedness property will still be maintained.

We next refine the definition of join-graph decompositions, when arcs can be labeled with a subset of their separator.

Definition 9.4 ((minimal) arc-labeled join-graph decompositions.) An *arc-labeled decomposition* for $BN =< X, D, P_G, \prod$ is a graph, χ and ψ associate with each vertex $v \in V$ the sets $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ and θ associates with each edge $(v, u) \subset E$ the set $\theta((v, u)) \subseteq X$ such that:

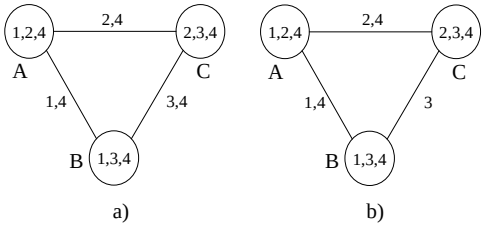


Figure 9.1: An arc-labeled decomposition

1. For each function $p_i \in P$, there is *exactly* one vertex $v \in V$ such that $p_i \in \psi(v)$, and $scope(p_i) \subseteq \chi(v)$.
2. (arc-connectedness) For each arc (u, v), $\theta(u, v) \subseteq sep(u, v)$, such that $\forall X_i \in X$, any two clusters containing X_i can be connected by a path whose every arc's label includes X_i.

Finally, an arc-labeled join-graph is *minimal* if no variable can be deleted from any label while still satisfying the arc-connectedness property.

Recall the following definitions of separators and eliminators.

Definition 9.5 separator, eliminator. Given two adjacent vertices u and v of JG, the *separator* of u and v is defined as $sep(u, v) = \theta((u, v))$, and the *eliminator* of u with respect to v is $elim(u, v) = \chi(u) - \theta((u, v))$.

Arc-labeled join-graphs can be made *label-minimal* by removing variables from their labels while maintaining connectedness (if an arc label becomes empty, the arc can be deleted altogether). It is easy to see that,

Proposition 9.6 *A* minimal arc-labeled join-graph *does not contain any cycle relative to any single variable. That is, any two clusters containing the same variable are connected by exactly one path labeled with that variable.*

Notice that every minimal arc-labeled join-graph is arc-minimal (no arc can be deleted), but not vice-versa. The mini-clustering approximation presented inChapter 3, works by relaxing the join-tree requirement of exact inference into a collection of join-trees having smaller cluster size. As we explained, it introduces some independencies in the original problem via node duplication and applies exact inference on the relaxed model requiring only 2 phases of message passings along the mini-cluster tree. For the iterative class of algorithms which we now introduce and call *IJGP*, the idea is to relax the tree-structure requirement and use instead join-graphs, which do not introduce new independencies, allowing iterative message-passing on the resulting cyclic structure. The intuition is that if IBP often works well in practice, even though we cannot predict when, having larger clusters can improve the accuracy of the same type of scheme.

Indeed, it can be shown that any join-graph of a belief network does not introduce any new independencies to the problem, namely it is an I-map (independency map [109]) of the underlying probability distribution relative to node-separation. Since we plan to use minimally arc-labeled join-graphs to address over-counting problems, the question is what kind of independencies are captured by such graphs.

Definition 9.7 **(edge-separation in (arc-labeled) join-graphs)** Let $D = \langle JG, \chi, \psi, \theta \rangle$, $JG = (V, E)$ be an edge-labeled decomposition of a Bayesian network $\mathcal{B} = \langle X, D, P_G, \prod \rangle$. Let $N_W, N_Y \subseteq V$ be two sets of nodes, and $E_Z \subseteq E$ be a set of edges in JG. Let W, Y, Z be their corresponding sets of variables ($W = \cup_{v \in N_W} \chi(v)$, $Z = \cup_{e \in E_Z} \theta(e)$). We say that E_Z *edge-separates* N_W and N_Y in D if there is no path between N_W and N_Y in the JG graph whose edges in E_Z are removed. In this case we also say that W is *separated* from Y given Z in D, and write $\langle W|Z|Y \rangle_D$. Edge-separation in a regular join-graph is defined relative to its separators.

Theorem 9.8 *Any arc-labeled join-graph decomposition $D = \langle JG, \chi, \psi, \theta \rangle$ of a belief network $\mathcal{B} = \langle X, D, G, P \rangle$ is an I-map of P relative to edge-separation. Namely, any edge separation in D corresponds to conditional independence in P.*

For a proof see [99].

9.1.1 ALGORITHM IJGP

Applying CTE iteratively to minimal arc-labeled join-graphs yields algorithm *Iterative Join-Graph Propagation (IJGP)* described in Figure 2.2. One iteration of the algorithm applies message-passing in a topological order over the join-graph, forward and back. When node u sends a message (or messages) to a neighbor node v it operates on all the CPTs in its cluster and on all the messages sent from its neighbors excluding the ones received from v. First, all individual functions that share no variables with the eliminator are collected and sent to v. All the rest of the functions are *combined* in a product and summed over the eliminator between u and v.

It is straightforward to show that:

Theorem 9.9

1. *If $IJGP$ is applied to a join-tree decomposition it reduces to join-tree clustering and it therefore is guaranteed to compute the exact beliefs in one iteration.*
2. *[99] The time complexity of one iteration of IJGP is $O(deg \cdot (n + N) \cdot k^{w+1})$ and its space complexity is $O(N \cdot k^{\theta})$, where deg is the maximum degree of a node in the join-graph, n is the number of variables, N is the number of nodes in the graph decomposition, k is the maximum domain size, w is the maximum cluster size and θ is the maximum label size.*

Algorithm **Iterative Join Graph Propagation (IJGP)**

Input An arc-labeled join-graph decomposition $\langle JG, \chi, \psi, \theta \rangle, JG = (V, E)$ for $\mathcal{B} = \langle X, D, P_G, \prod \rangle$. Evidence variables $var(e)$, evidence e.

Output An augmented graph whose nodes are clusters containing the original CPTs and the messages received from neighbors. Approximations of $P(X_i, e), \ \forall X_i \in X$.

Denote by $h_{(u \to v)}$ the message from vertex u to v, $ne_v(u)$ the neighbors of u in JG excluding v.
$cluster(u) = \psi(u) \cup \{h_{(v \to u)} | (v, u) \in E\}$.
$cluster_v(u) = cluster(u)$ excluding message from v to u.

• **One iteration of IJGP:**
For every node u in JG in some topological order d and back, do

1. **Process observed variables:**
   Assign relevant evidence to all $p_i \in \psi(u)$ $\chi(u) := \chi(u) - var(e), \ \ \forall u \in V$
2. **Compute individual functions:**
   Include in $H_{(u \to v)}$ each function in $cluster_v(u)$ whose scope does not contain variables in $elim(u, v)$. Denote by A the remaining functions.
3. **Compute and send to v the combined function:** $h_{(u \to v)} = \alpha \sum_{elim(u,v)} \prod_{f \in A} f$.
   Send $h_{(u \to v)}$ and the individual functions $H_{(u \to v)}$ to node v.

Endfor
• **Compute $P(X_i, e)$:**

For every $X_i \in X$ let u be a vertex in JG such that $X_i \in \chi(u)$.
Compute $P(X_i, e) = \alpha \sum_{\chi(u) - \{X_i\}} (\prod_{f \in cluster(u)} f)$

Figure 9.2: Algorithm Iterative Join-Graph Propagation (IJGP)

Proof. The number of cliques in the chordal graph G' corresponding to G is at most n, so the number of nodes in the join-tree is at most n. The complexity of processing a node u in the join-tree is $deg_u \cdot (|\psi(u)| + deg_u - 1) \cdot d^{|\chi(u)|}$, where deg_u is the degree of u. By bounding deg_u by deg, $|\psi(u)|$ by n and $\chi(u)$ by $w^* + 1$ and knowing that $deg < N$, by summing over all nodes, we can bound the entire time complexity by $O(deg \cdot (n + N) \cdot d^{w^*+1})$.

For each edge JTC records functions. Since the number of edges in bounded by n and the size of each message is bounded by d^{sep} we get space complexity of $O(n \cdot d^{sep})$. □

One question which we did not address at all in this section is why propagating the messages iteratively should help. Why is IJGP upon convergence, superior to IJGP with one iteration and is superior to MC? One clue can be provided when considering deterministic constraint networks which can be viewed as "extreme probabilistic networks". It is known that constraint propagation algorithms, which are analogous to the messages sent by belief propagation, are guaranteed to converge and are guaranteed to improve with convergence. The propagation scheme presented here works like constraint propagation relative to the flat network abstraction of P (where all non-zero entries are normalized to a positive constant), and propagation is guaranteed to be more accurate for that abstraction at least. Another explanation is provided [75] by showing a connection between the probability distribution generated by IJGP (upon convergence) and the distribution that minimize a distance function to the exact distribution.

Next we will demonstrate that the well-known algorithm IBP (also called loopy belief propagation) is a special case of IJGP.

9.1.2 ITERATIVE BELIEF PROPAGATION

Iterative belief propagation (IBP) is an iterative application of Pearl's algorithm for poly-trees [109] to any Bayesian network. We will describe IBP as an instance of join-graph propagation over a *dual graph*. We recap the definition of a dual graph.

Definition 9.10 dual graphs. Given a set of functions $F = \{f_1, ..., f_l\}$ over scopes $S_1, ..., S_l$, the dual graph of F is a graph $DG = (V, E, L)$ that associates a node with each function, namely $V = F$ and an edge connects any two nodes whose function's scope share a variable, $E = \{(f_i, f_j)|S_i \cap S_j \neq \phi\}$. L is a set of labels for the arcs, each being labeled by the shared variables of its nodes, $L = \{l_{ij} = S_i \cap S_j|(i, j) \in E\}$. A *dual join-graph* is an arc-labeled edge subgraph of DG. A *minimal* dual join-graph is a dual join-graph for which none of the edge labels can be further reduced while maintaining the connectedness property.

Interestingly, there may be many minimal dual join-graphs of the same dual graph. We will define Iterative Belief Propagation on a dual join-graph. Each node sends a message over an edge whose scope is identical to the label on that edge. Since Pearl's algorithm sends messages whose scopes are singleton variables only, we highlight minimal singleton-label dual join-graphs.

Proposition 9.11 *Any Bayesian network has a minimal dual join-graph where each arc is labeled by a single variable.*

Proof. Consider a topological ordering of the nodes in the acyclic directed graph of the Bayesian network $d = X_1, ..., X_n$. We define the following dual join-graph. Every node in the dual graph $\mathcal{D}$, associated with p_i is connected to node p_j, $j < i$ if $X_j \in pa(X_i)$. We label the arc between p_j and p_i by variable X_j, namely $l_{ij} = \{X_j\}$. It is easy to see that the resulting arc-labeled subgraph of

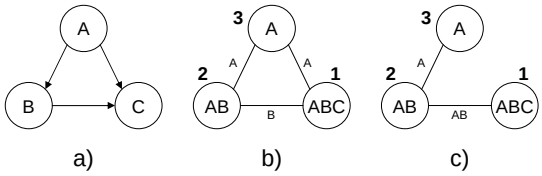


Figure 9.3: a) A belief network; b) A dual join-graph with singleton labels; c) A dual join-graph which is a join-tree

the dual graph satisfies connectedness. The resulting labeled graph is a dual graph with singleton labels. □

Example 9.12 Consider the belief network on 3 variables A, B, C with CPTs 1. $P(C|A,B)$, 2. $P(B|A)$ and 3. $P(A)$, given in Figure 2.3a. Figure 2.3b shows a dual graph with singleton labels on the edges. Figure 2.3c shows a dual graph which is a join-tree, on which belief propagation can solve the problem exactly in one iteration (two passes up and down the tree).

For completeness, we present algorithm IBP in Figure 2.4. It is a special case of IJGP. It is easy to see that one iteration of IBP is time and space linear in the size of the belief network. It is also easy to show that when IBP is applied to a minimal singleton-labeled dual graph it coincides with Pearl's belief propagation applied directly to the acyclic graph representation. Also, when the dual join-graph is a tree IBP converges after one iteration (two passes, up and down the tree) to the exact beliefs.

9.1.3 BOUNDED JOIN-GRAPH DECOMPOSITIONS

Since we want to control the complexity of join-graph algorithms, we will define it on decompositions having bounded cluster size. If the number of variables in a cluster is bounded by i, the time and space complexity of processing one cluster is exponential in i.

Given a join-graph decomposition $D = \langle JG, \chi, \psi, \theta \rangle$, the accuracy and complexity on the (iterative) join-graph propagation algorithm depends on the joinwidth of D defined as $max_{v \in V} |\chi(v)|$. Intuition also suggests that the accuracy depends on how far the join-graph is from a join-tree, which may be captured by the treewidth of JG which we would call *external width*.

We can now state our target decomposition as follows. Given a graph G, and a bounding parameter i we wish to find a join-graph decomposition D of G whose internal width is bounded by i and whose external width is minimized.

We can consider two classes of algorithms. One class is *partition-based*. It starts from a given tree-decomposition and then partitions the clusters until the decomposition has clusters bounded by i. An alternative approach is *grouping-based*. It starts from a minimal dual-graph-based join-graph decomposition (where each cluster contains a single CPT) and groups clusters into larger clusters

Algorithm **IBP**
Input: An edge-labeled dual join-graph $DG = (V, E, L)$ for a Bayesian network $\mathcal{B} = \langle X, D, P_G, \prod \rangle$. Evidence e.
Output: An augmented graph whose nodes include the original CPTs and the messages received from neighbors. Approximations of $P(X_i, e)$, $\forall X_i \in X$. Approximations of $P(F_i, e)$, $\forall F_i \in \mathcal{B}$.
Denote by: $h_{u\to v}$ the message from u to v; $ne(u)$ the neighbors of u in V; $ne_v(u) = ne(u) - \{v\}$; l_{uv} the label of $(u, v) \in E$; $elim(u, v) = scope(u) - scope(v)$.
• **One iteration of IBP**
For every node u in DJ in a topological order and back, do:
1. **Process observed variables**
Assign evidence variables to the each p_i and remove them from the labeled edges.
2. **Compute and send to v the function:**

$$h_{u\to v} = \alpha \sum_{elim(u,v)} (p_u \cdot \prod_{\{h_{i\to u}, i\in ne_v(u)\}} h_{i\to v})$$

Endfor
• **Compute approximations of $P(F_i, e)$, $P(X_i, e)$:**
For every $X_i \in X$ let u be the vertex of family F_i in DJ,
$P(F_i, e) = \alpha(\prod_{h_{i\to u}, u\in ne(i)} h_{i\to u}) \cdot p_u$;
$P(X_i, e) = \alpha \sum_{scope(u)-\{X_i\}} P(F_i, e)$.

Figure 9.4: Algorithm Iterative Belief Propagation

as long as the resulting clusters do not exceed the given bound. In both methods one should attempt to reduce the external width of the generated graph-decomposition.

We will next present a partition-based approach which is based on the decomposition suggested by the mini-bucket scheme. Given a bound i, algorithm *Join-Graph Structuring(i)* applies the procedure *Schematic Mini-Bucket(i)*, described in Figure 2.6. The procedure only traces the scopes of the functions that would be generated by the full mini-bucket procedure, avoiding actual function computation. The procedure ends with a collection of mini-bucket trees, each rooted in the mini-bucket of the first variable. Each of these trees is minimally edge-labeled. Then, *in-edges* labeled with only one variable are introduced, and they are added only to obtain the running intersection property between branches of these trees.

Proposition 9.13 *Algorithm Join-Graph Structuring(i) generates a minimal edge-labeled join-graph decomposition having bound i.*

Proof. The construction of the join-graph specifies the vertices and edges of the join-graph, as well as the variable and function labels of each vertex. We need to demonstrate that 1) the connectedness property holds, and 2) that edge-labels are minimal.

Connectedness property specifies that for any 2 vertices u and v, if vertices u and v contain variable X, then there must be a path $u, w_1, ..., w_m, v$ between u and v such that every vertex on this path contains variable X. There are two cases here. 1) u and v correspond to 2 mini-buckets in

Algorithm **Join-Graph Structuring(*i*)**

1. Apply procedure schematic mini-bucket(i).
2. Associate each resulting mini-bucket with a node in the join-graph, the variables of the nodes are those appearing in the mini-bucket, the original functions are those in the mini-bucket.
3. Keep the edges created by the procedure (called out-edges) and label them by the regular separator.
4. Connect the mini-bucket clusters belonging to the same bucket in a chain by in-edges labeled by the single variable of the bucket.

Figure 9.5: Algorithm Join-Graph Structuring(i).

Procedure **Schematic Mini-Bucket(*i*)**

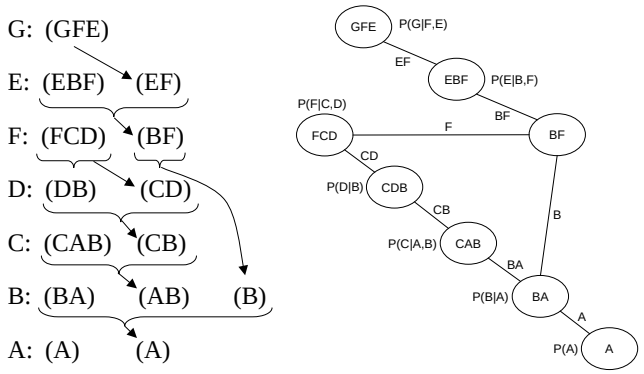
1. Order the variables from X_1 to X_n minimizing (heuristically) induced-width, and associate a bucket for each variable.
2. Place each CPT in the bucket of the highest index variable in its scope.
3. For $j = n$ to 1 do:
   Partition the functions in $bucket(X_j)$ into mini-buckets having at most i variables.
   For each mini-bucket mb create a new scope-function (message) f where $scope(f) = \{X | X \in mb\} - \{X_i\}$ and place scope(f) in the bucket of its highest variable. Maintain an edge between mb and the mini-bucket (created later) of f.

Figure 9.6: Procedure Schematic Mini-Bucket(i).

the same bucket, or 2) u and v correspond to mini-buckets in different buckets. In case 1 we have 2 further cases, 1a) variable X is being eliminated in this bucket, or 1b) variable X is not eliminated in this bucket. In case 1a, each mini-bucket must contain X and all mini-buckets of the bucket are connected as a chain, so the connectedness property holds. In case 1b, vertexes u and v connect to their (respectively) parents, who in turn connect to their parents, etc. until a bucket in the scheme is reached where variable X is eliminated. All nodes along this chain include variable X, so the connectedness property holds. Case 2 resolves like case 1b.

To show that edge labels are minimal, we need to prove that there are no cycles with respect to edge labels. If there is a cycle with respect to variable X, then it must involve at least one in-edge (edge connecting two mini-buckets in the same bucket). This means variable X must be the variable being eliminated in the bucket of this in-edge. Therefore, variable X is not contained in any of the parents of the mini-buckets of this bucket. Therefore, in order for the cycle to exist, another in-edge down the bucket-tree from this bucket must contain X. However, this is impossible as this would imply that variable X is eliminated twice. □

Example 9.14 Figure 2.7a shows the trace of procedure schematic mini-bucket(3) applied to the problem described in Figure 1.17a. The decomposition in Figure 2.7b is created by the algorithm



(*a*) (*b*)

Figure 9.7: Join-graph decompositions.

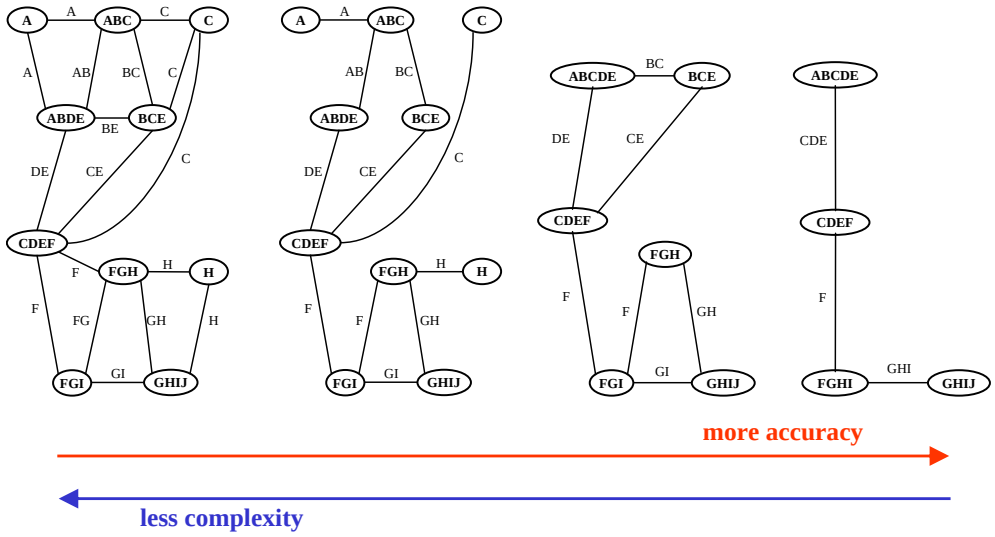


Figure 9.8: Join-graphs.

graph structuring. The only cluster partitioned is that of F into two scopes (FCD) and (BF), connected by an in-edge labeled with F.

Example 9.15 Figure 2.8 shows a range of edge-labeled join-graphs. On the left extreme we have a graph with smaller clusters, but more cycles. This is the type of graph IBP works on. On the right extreme we have a tree decomposition, which has no cycles but has bigger clusters. In between,

there could be a number of join-graphs where maximum cluster size can be traded for number of cycles. Intuitively, the graphs on the left present less complexity for join-graph algorithms because the cluster size is small, but they are also likely to be less accurate. The graphs on the right side are computationally more complex, because of larger cluster size, but are likely to be more accurate.

9.2 COST-SHIFTING FOR BETTER APPROXIMATION

All the decomposition bounds that we have shown can be improved, or tightened by what is sometime called *cost-shifting* and sometimes is called *re-parameterization* and fall under the class of variational schemes. The idea is to shift some costs from one local function to another in a way that does not change to global function, but can improve the decomposition bound. We will see several variants of these ideas, one is mini-bucket elimination with max-marginal matching (MBE-MM), which is a non-iterative algorithm that applies a single pass of cost-shifting during the mini-bucket construction bucket by bucket. The second approach, Join Graph Linear Programming (JGLP), iteratively applies cost-shifting updates to the full mini-bucket join-graph.

9.2.1 LINEAR PROGRAMMING METHODS

To introduce the idea assume for simplicity that the network consists only of pairwise functions $f_{ij}(X_i, X_j)$ and that the problem is max-sum, i.e., to compute $C^* = \max_{\mathbf{X}} \sum_{f_{ij} \in F} f_{ij}(X_i, X_j)$. A simple bound on the max-sum objective is then given by maxima of the individual functions, exchanging the sum and max operators, as is common in the mini-bucket scheme:

$$C^* = \max_{\mathbf{X}} \sum_{f_{ij} \in F} f_{ij}(X_i, X_j) \leq \sum_{f_{ij} \in F} \max_{X_i, X_j} f_{ij}(X_i, X_j) \tag{9.1}$$

As noted for the mini-bucket case we can interpret this operation as making an individual copy of each variable for each function, and optimizing over them separately. See, for example, the problem in Figure 2.9(a) with 3 variables and 3 functions. Figure 2.9(e). We can also derive a bound on the optimal cost by introducing a collection of functions $\{\lambda_{ij}(X_i), \lambda_{ji}(X_j)\}$ for each edge (ij) and requiring

$$\lambda \in \Lambda \quad \Leftrightarrow \quad \forall i, \quad \sum_j \lambda_{ij}(X_i) = 0 \tag{9.2}$$

Then, we have

$$\begin{aligned} C^* &= \max_{\mathbf{X}} \sum_{f_{ij}\in F} f_{ij}(X_i, X_j) \\ &= \max_{\mathbf{X}} \sum_{f_{ij}\in F} f_{ij}(X_i, X_j) + \sum_i \sum_j \lambda_{ij}(X_i) \\ &\leq \min_{\lambda\in\Lambda} \sum_{f_{ij}\in F} \max_{X_i, X_j} \left(f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j)\right) \end{aligned} \tag{9.3}$$

The last expression is obtained by distributing each λ_{ij} to its associated factor and applying the inequality (2.1).

The new functions $\tilde{f}_{ij} = f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j)$ define a *re-parametrization* of the original distribution, i.e., they change the individual functions without modifying the global function $F(\mathbf{X}) = \sum_{\mathbf{X}} f_{ij} = \sum_{\mathbf{X}} \tilde{f}_{ij}$. The λ_{ij} can be interpreted as "cost-shifting" operations that transfer cost from one function to another while preserving the overall cost [3] or as Lagrange multipliers enforcing consistency among the copies of X_i [75**?**]. In the former interpretation, the updates are called "soft arc-consistency" due to their similarity to arc-consistency for constraint satisfaction. Under the latter view, the bound corresponds to a *dual decomposition* solver for a linear programming (LP) relaxation of the original problem.

The main distinguishing feature among such dual decomposition approaches is the way in which the bound is tightened by updating the functions λ. This is done either by sub-gradient or gradient approaches or by coordinate descent updates that can be interpreted as "message passing" [**? ?**]. Without going into the details of these approaches, we refer to these iterative bound improvement updates as "LP-tightening" updates.

LP-tightening algorithm. Let's see a derivation of a particular simple, yet effective scheme that minimizes over λs the expression

$$\sum_{f_{ij}\in F} \max_{X_i, X_j} \left(f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j)\right) \tag{9.4}$$

tightening the upper bound on C^* (Eq. 2.3).

The LP-tightening scheme is initialized with all $\lambda_{ij}(X_i) = 0$. In the spirit of well-known coordinate descent approaches we iteratively minimize expression 2.4 with respect to each $\lambda_{ij}(X_i)$ separately, while fixing the values of all other λs. For simplicity of derivation, without loss of generality, lets assume that each variable X_i appears in the scope of at most two functions, $f_{ij}(X_i, X_j)$ and $f_{ik}(X_i, X_k)$. Identifying only the terms relevant to variable X_i, we get:

$$\begin{aligned} &\min_{\lambda_{ij}(X_i), \lambda_{ik}(X_i)} \left[\left[\max_{X_i, X_j} f_{ij}(X_i, X_j) + \lambda_{ij}(X_i)\right] + \left[\max_{X_i, X_k} f_{ik}(X_i, X_k) + \lambda_{ik}(X_i)\right]\right] \\ &= \min_{\lambda_{ij}(X_i), \lambda_{ik}(X_i)} \left[\max_{X_i} \left[\max_{X_j} f_{ij}(X_i, X_j) + \lambda_{ij}(X_i)\right] + \max_{X_i} \left[\max_{X_k} f_{ik}(X_i, X_k) + \lambda_{ik}(X_i)\right]\right] \end{aligned}$$

Figure 9.9: The mini-bucket procedure for a simple graph. (a) Primal graph; (b) the buckets and messages computed in MBE; (c) join-graph, dotted line corresponds to an edge absent from a mini-bucket tree; (d) interpreting MBE as variable duplication, X_1 is duplicated in each of two mini-buckets $q_1^1 = \{f_1(X_1, X_2)\}$ and $q_1^2 = \{f_3(X_1, X_3)\}$; (e) The graph on which FGLP runs with all variables duplicated and each factor processed separately.

Let us define the so-called "max-marginals" $\gamma_{ij}(X_i) = \max_{X_j} f_{ij}(X_i, X_j)$ and re-arrange the terms in the above expression, yielding the following bound:

$$\begin{aligned}
&\min_{\lambda_{ij}(X_i),\lambda_{ik}(X_i)} \Big[\max_{X_i} \big[\max_{X_j} f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) \big] + \max_{X_i} \big[\max_{X_k} f_{ik}(X_i, X_k) + \lambda_{ik}(X_i) \big] \Big] \\
&= \min_{\lambda_{ij}(X_i),\lambda_{ik}(X_i)} \Big[\max_{X_i} \big[\gamma_{ij}(X_i) + \lambda_{ij}(X_i) \big] + \max_{X_i} \big[\gamma_{ik}(X_i) + \lambda_{ik}(X_i) \big] \Big] \\
&\geq \min_{\lambda_{ij}(X_i),\lambda_{ik}(X_i)} \Big[\max_{X_i} \big[\gamma_{ij}(X_i) + \gamma_{ik}(X_i) + \lambda_{ij}(X_i) + \lambda_{ik}(X_i) \big] \Big]
\end{aligned} \tag{9.5}$$

We force that $\lambda_{ij}(X_i) + \lambda_{ik}(X_i) = 0$ to ensure that the total cost does not changed (Equation 2.2).

Many choices of λ_{ij} are known to achieve the minimum of the right-hand side of expression 2.5. Lets select the following one:

$$\lambda_{ij}(X_i) = \frac{1}{2}\big(\gamma_{ik}(X_i) - \gamma_{ij}(X_i)\big) = \frac{1}{2}\big(\max_{X_k} f_{ik}(X_i, X_k) - \max_{X_j} f_{ij}(X_i, X_j)\big) \tag{9.6}$$

Algorithm 6: LP-tightening (based on [**?**])

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, where $f_{\mathbf{S}_i}$ is a potential defined on variables $\mathbf{S}_i$
Output: Upper bound on the optimum value of $\max_{\mathbf{X}} \sum \mathbf{F}$
while *NOT converged* **do**
  for *any pair of function scopes* $\boldsymbol{S}_i$, $\boldsymbol{S}_j$ *such that* $\boldsymbol{S}_{ij} = \boldsymbol{S}_i \cap \boldsymbol{S}_j \neq \emptyset$ **do**
    Compute max-marginals:
    $\gamma_{\mathbf{S}_i}(\mathbf{S}_{ij}) = \max_{\mathbf{S}_i \setminus \mathbf{S}_{ij}} f_{\mathbf{S}_i}$;
    $\gamma_{\mathbf{S}_j}(\mathbf{S}_{ij}) = \max_{\mathbf{S}_j \setminus \mathbf{S}_{ij}} f_{\mathbf{S}_j}$;
    Update parametrization:
    $f_{\mathbf{S}_i} \leftarrow f_{\mathbf{S}_i} + \frac{1}{2}\big(\gamma_{\mathbf{S}_j}(\mathbf{S}_{ij}) - \gamma_{\mathbf{S}_i}(\mathbf{S}_{ij})\big)$;
    $f_{\mathbf{S}_j} \leftarrow f_{\mathbf{S}_j} + \frac{1}{2}\big(\gamma_{\mathbf{S}_i}(\mathbf{S}_{ij}) - \gamma_{\mathbf{S}_j}(\mathbf{S}_{ij})\big)$;

Iterating over all functions, while updating each function $\forall i, j, f_{ij}(X_i, X_j) \leftarrow f_{ij}(X_i, X_j) + \lambda_{ij}(X_i)$ and recalculating λ_{ij} at each step until convergence, yields a minimization procedure that can be interpreted as a *max-marginal* or *moment-matching* procedure on the functions $f_{ij}(X_i, X_j)$. Intuitively, we would like the updates to diminish in magnitude and converge to zero as fast as possible. To achieve that, taking into account Equation 2.6, we would like to make the max-marginals $\gamma_{ij}(X_i)$ and $\gamma_{ik}(X_i)$ of each variable X_i equal to each other. Algorithm 1 generalizes this update to higher-order functions $f_{\mathbf{S}_i}$ over scopes of variables $\mathbf{S}_i \subseteq \mathbf{X}$.

A well-known algorithm quite similar to the above LP-tightening in Algorithm 1 is a message-passing scheme called Max-Product Linear Programming (MPLP) [**?**]. Algorithm 2 presents a version of MPLP that we call Factor Graph Linear Programming (FGLP). At each step FGLP simultaneously updates all functions $f_{ij}(X_i, X_j)$ involving a single variable X_i. The messages sent by the algorithms on a factor graph are schematically illustrated in Figure 2.10. In this example variable X_i is in the scope of three functions: $f_{\mathbf{S}_t}(X_i, X_k, X_m)$, $f_{\mathbf{S}_p}(X_i, X_j)$ and $f_{\mathbf{S}_q}(X_i, X_n)$, where $\mathbf{S}_t = \{X_i, X_k, X_m\}$, $\mathbf{S}_p = \{X_i, X_j\}$ and $\mathbf{S}_q = \{X_i, X_n\}$. Note that in the figure we only show the messages involving X_i. The max-marginals are: $\gamma_{\mathbf{S}_q}(X_i) = \max_{X_n} f_{\mathbf{S}_q}(X_i, X_n)$, $\gamma_{\mathbf{S}_p}(X_i) = \max_{X_j} f_{\mathbf{S}_p}(X_i, X_j)$ and $\gamma_{\mathbf{S}_t}(X_i) = \max_{X_k, X_m} f_{\mathbf{S}_t}(X_i, X_k, X_m)$. The update messages from variable X_i back to the functions are:

$$\beta_{\mathbf{S}_q}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_q}(X_i)$$
$$\beta_{\mathbf{S}_p}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_p}(X_i)$$
$$\beta_{\mathbf{S}_t}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_t}(X_i)$$

Theorem 9.16 Complexity of FGLP. *The total time complexity of a single iteration of FGLP is* $O(n \cdot Q \cdot k^{Sc})$, *where* n *is the number of variables in the problem,* k *is the largest domain size,* $|\boldsymbol{F}|$ *is the number of functions,* Sc *bounds the largest scope of the original functions,* Q *is the largest*

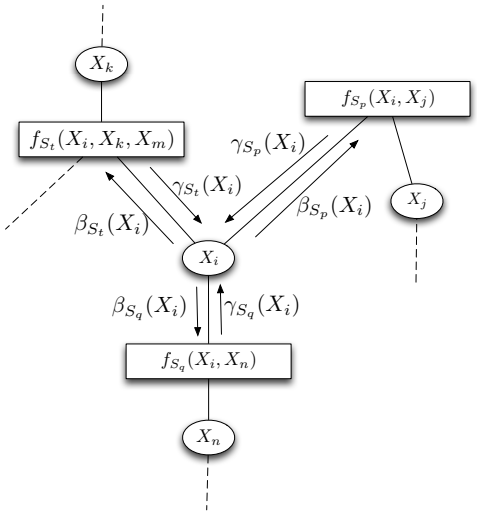


Figure 9.10: FGLP example: local messages involving variable X_i.

Algorithm 7: Factor Graph Linear Programming (FGLP, based on [**?**])

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, , \mathbf{F}, \sum \rangle$, variable ordering o
Output: Upper bound on the optimum value of MPE cost
while *NOT converged* **do**
  for *each variable* X_i **do**
    Get factors $F_i = f_{\mathbf{S}_k} : X_i \in \mathbf{S}_k$ with X_i in their scope;
    //for each function compute max-marginals γ marginalizing out all variables except for X_i:
    $\forall f_{\mathbf{S}_k}\ \gamma_{\mathbf{S}_k}(X_i) = \max_{\mathbf{S}_k \setminus X_i} f_{\mathbf{S}_k}$;
    // compute messages $\beta_{\mathbf{S}_k}(X_i)$ from X_i back to a function $f_{\mathbf{S}_k}$ correcting for the function's own max-marginal $\gamma_{\mathbf{S}_k}$:
    $\forall f_{\mathbf{S}_k}\ \beta_{\mathbf{S}_k} = \frac{1}{|F_i|} \sum_{\{\mathbf{S}_j | f_{\mathbf{S}_j} \in F_i\}} \gamma_{\mathbf{S}_j}(X_i) - \gamma_{\mathbf{S}_k}(X_i)$
    // update (re-parametrize) each function:
    $\forall f_{\mathbf{S}_k}, f_{\mathbf{S}_k} \leftarrow f_{\mathbf{S}_k} + \beta_{\mathbf{S}_k}$;

number of functions having the same variable X_j in their scopes. The space complexity is $O(|\boldsymbol{F}| \cdot k^{Sc})$.

9.2.2 JOIN GRAPH LINEAR PROGRAMMING

Join-Graph MBE Structuring. The mini-bucket procedure defines a mini-bucket tree. Each mini-bucket defines a cluster. Two mini-buckets are connected if there exists a message between them. Once the mini-buckets of the same variables are connected, the mini-bucket tree yields a join-graph, where each cluster has at most $i + 1$ variables, where i is the i-bound.

Join-Graph MBE Structuring (Algorithm 2.5) constructs a join-graph using the mini-bucket elimination. Note that the separators between the clusters corresponding to the mini-buckets of the same variable may be over more than a single variable, so the resulting graph may not be a join-graph in a strict sense, not satisfying the induced tree property. However, this minor point does not impact the use of the graph by our algorithms. The structure is the same as the mini-bucket-tree assuming variable duplication with additional arcs connecting mini-buckets of the same bucket in a line.

Figure 2.9 presents an example of a problem with 3 variables, whose primal graph is shown in Figure 2.9(a). In Figure 2.9(b) we see the trace of MBE on the problem, namely the buckets and the messages computed. Figure 2.9(c) shows the join-graph created by Join-Graph MBE Structuring. The dotted line corresponds to an edge absent from a mini-bucket tree and added during step at line 8.

Join-Graph Linear Programming. From the perspective of linear programming relaxation (Section 2.2.1) mini-bucket elimination can be interpreted as running a single pass of LP-tightening, sending messages top down only along edges of the spanning tree of the mini-bucket join graph. A straightforward extension of MBE can be an iterative procedure that repeatedly performs LP-tightening along all of the join graph edges. Algorithm 3 shows the resulting Join-Graph Linear Programming (JGLP) scheme. It constructs the join graph (line 1), calculated mini-bucket functions

Algorithm 8: Algorithm JGLP

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, , \mathbf{F}, \sum \rangle$, variable order $o = \{X_1, \ldots, X_n\}$, parameter i.
Output: Upper bound on the optimum value of MPE cost
//Initialize: Partition the functions in $\mathbf{F}$ into $\mathbf{B}_{X_1}, \ldots, \mathbf{B}_{X_n}$, where $\mathbf{B}_{X_p}$ contains all functions f_{S_j} whose highest variable is X_p;
Build mini-bucket join graph; // (Algorithm **??**)
Find the function of each mini-bucket q_k^p:
$F_k^p = \sum_{f_{\mathbf{S}_j} \in q_k^p} f_{\mathbf{S}_j}$
while *NOT converged OR NOT time limit reached* **do**
  for *all pairs of mini-buckets* q_k^p, q_k^l *connected by an edge* **do**
    Find the separators $S = Scope(q_k^p) \cap Scope(q_k^l)$;
    Find the max-marginals of each mini-bucket
    q_k^p: $\gamma_k^p = \max_{Scope(q_k^p)-S}(F_k^p)$;
    q_k^j: $\gamma_k^l = \max_{Scope(q_k^l)-S}(F_k^l)$;
    Update functions in both mini-buckets
    q_k^p: $F_k^p \leftarrow F_k^p - \frac{1}{2}(\gamma_k^p - \gamma_k^l)$
    q_k^l: $F_k^l \leftarrow F_k^l + \frac{1}{2}(\gamma_k^p - \gamma_k^l)$;

F_k^p (line 2) and performs re-parametrization updates to F_k^p (as in Algorithm 1) until convergence (lines 2-7). Note that once JGLP converges, performing mini-bucket elimination on the resulting graph will not change the bound value.

Theorem 9.17 Complexity of JGLP. *Given a problem with n variables with largest domains of size k, where at most Q functions have the same variable in their scope, and i-bound i, the time complexity of a single iteration of JGLP is $O(n \cdot Q \cdot k^i)$. The time complexity of join-graph construction step is $O(n \cdot k^{i+1})$. The overall space complexity is $O(n \cdot k^{i+1} + n \cdot k^i)$.*

Proof. The complexity of constructing a join-tree is the same as the complexity of running full mini-bucket elimination algorithm, namely $O(n \cdot k^{i+1})$. The time complexity of a single iteration of JGLP consists of performing for each edge the following steps: 1. compute max-marginals of a pair of clique functions, requires time equal to $O(2 \cdot k^{i+1-|S|})$, where $|S|$ is the size of a separator between the mini-buckets of the same variable. 2. compute the mean, which takes $O(2 \cdot k^{|S|})$ time. 3. update the clique functions, requiring $O(2 \cdot k^{i+1})$ time.

The space complexity of the algorithm is dominated by the necessity of storing in memory the join-graph, whose size is bounded by $O(n \cdot k^{i+1})$ and the messages between the clusters of size $O(n \cdot k^i)$, yielding the overall space complexity of $O(n \cdot k^{i+1})$. □

Algorithm 9: Algorithm MBE-MM

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, , \mathbf{F}, \sum \rangle$, variable order $o = \{X_1, \ldots, X_n\}$, i-bound parameter i
Output: Upper bound on the optimum value of MPE cost
//Initialize:
Partition the functions in $\mathbf{F}$ into $\mathbf{B}_{X_1}, \ldots, \mathbf{B}_{X_n}$, where $\mathbf{B}_{X_k}$ contains all functions f_j whose highest variable is X_k;
//processing bucket $\mathbf{B}_{X_k}$
for $k \leftarrow n$ *down to 1* **do**
  Partition functions g (both original and messages generated in previous buckets) in $\mathbf{B}_{X_k}$ into the mini-buckets defined $Q_{X_k} = \{q_k^1, \ldots, q_k^t\}$, where each q_k^p has no more than $i+1$ variables;
  Find the set of variables common to all the mini-buckets of variable X_k: $\mathbf{S}_k = Scope(q_k^1) \cap \cdots \cap Scope(q_k^t)$;
  Find the function of each mini-bucket
  q_k^p: $F_k^p \leftarrow \prod_{g \in q_k^p} g$;
  Find the max-marginals of each mini-bucket
  q_k^p: $\gamma_{X_k}^p = \max_{Scope(q_k^p) \setminus \mathbf{S}_k} (F_k^p)$;
  Update functions of each mini-bucket
  $F_k^p \leftarrow F_k^p - \gamma_{X_k}^p + \frac{1}{t} \sum_{j=1}^t \gamma_{X_k}^j$;
  Generate messages $h_{X_k \rightarrow X_m}^p = \max_{X_k} F_k^p$ and place each in the bucket of highest in the ordering o variable X_m in $Scope(q_k^p)$;
return *All the buckets and the cost bound from* $\boldsymbol{B}_1$*;*

9.2.3 MBE-MM

While the iterative nature of JGLP yields more accurate bounds, in practice it can have a significant additional time and space overhead compared to MBE. The latter scheme does not need to store the entire functions computed in mini-buckets, requiring space of $O(nk^{(i+1)} + nk^i)$, only the messages between them, reducing the space complexity to $O(nk^i)$ [**?**]. The difference may seem insignificant worst-case wise, but makes a practical impact, as we will see in the empirical section.

We we next present a non-iterative scheme that performs re-parametrization between the mini-buckets of the same variable only. The algorithm mini-bucket elimination with max-marginal matching (MBE-MM, Algorithm 4) proceeds by following the standard mini-bucket downward pass. When each mini-bucket $q_k^p \in Q_k$ is processed, before eliminating variable X_k, we first perform an LP-tightening update (that is, a re-parametrization) to the mini-bucket functions $f_{q_k^p}$. For storage and computational efficiency reasons, we perform a single update on all mini-buckets of the same variable simultaneously, matching their max-marginals on their joint intersection. Alternatively, the updates can be done between all possible pairs of mini-buckets. Although any max-marginal matching step strictly improves the bound within each bucket (Equation 2.3), it is not guaranteed to increase the overall accuracy. However, it is reasonable to expect that the update will help, and in practice we find that the bounds are almost always significantly improved (see the experiments, Section **??**).

Theorem 9.18 Complexity of MBE-MM. *Given a problem with n variables having domain of size k and an i-bound i, the worst-case time complexity of MBE-MM is $O(n \cdot Q \cdot k^{i+1})$ and its space complexity is $O(n \cdot k^i)$, where Q bounds the number of functions having the same variable X_i in their scopes.*

Interestingly, our algorithms are related to known methods in constraint satisfaction. MBE(i) and JGLP(i), parametrized by an i-bound, are analogous to directional i-consistency and full i-consistency, respectively. Our algorithm MBE-MM represents an intermediate step between these two, and is analogous to an improvement of directional i-consistency with full iterative relational consistency schemes within each bucket [**?**].

CHAPTER 10

Conclusion

We covered the principles of *exact* algorithms in graphical models, organized along the two styles of reasoning: *inference* and *search*. We focused on methods that are applicable to general graphical models, whose functions can come from a variety of frameworks and applications (constraints, Boolean, probabilities, costs etc.). These include, constraint networks and SAT models, Bayesian networks, Markov random fields, Cost networks, and Influence diagrams. Therefore, the primary features that capture structure in a unified way across all these models are graph features. The main graph property is the *induced-width* also known as *treewidth*, but we also showed the relevance of related features such as *height* of pseudo trees, *cycle-cutsets, q-cutsets* and *separator width*. We showed that both inference and search scheme are bounded exponentially by any of these parameters, and some combination of those hint at how we can trade memory for time.

With the exception of constraints, we did not discuss internal function structure as a potential feature. These function-structure features are sometimes addressed as *language* (e.g., Horn clauses, linear functions, convex functions) and can lead to various tractable classes. Other terms used are *context-sensitive* or *context specific independence*. In the constraint literature, tractability based on the language of constraints was investigated thoroughly (see Dechter [50, Chapter 10].) Likewise, focus on language is a central research activity in probabilistic reasoning. An example of a structure exploited in probabilistic graphical models are the sub-modular functions [52].

The next thing on our agenda is to extend the book with a second part focusing on approximation schemes. This obviously is necessary since exact algorithms cannot scale-up to many realistic applications that are complex and quite large and appropriately, current research centered on developing approximation schemes. But, we believe that **in order to have effective approximation algorithms we have to be equipped with the best exact algorithms, first.**

Approximation algorithms can be organized along the dimensions of inference and search as well. Given a general algorithmic architecture (such as Adaptive AND/OR search with caching (AOC(q)), or, alternatively, AO-VEC(q), we can approximate either the inference part or the search part or both, systematically yielding an ensemble of candidates approximation algorithms that can be studied. We can view message-passing and variational algorithms such as generalized belief propagation, the mini-bucket and weighted mini-bucket schemes [43, 87] as approximations that bound inference. We can view Monte Carlo sampling methods, as approximations to search. The hybrid schemes can be used to focus on approximating only those portions of the problem instance that appear non-tractable for exact processing. Namely, for a given problem instances, it can suggest a balance between approximate and exact and the type of approximation that should be utilized.

One should note that approximate reasoning in graphical modeling with any guarantees was shown to be hard as well [32, 114]. Yet, algorithms that generate bounds or anytime schemes that can improve their bounds if allowed more time, and even get to an exact solution when time permits, are highly desirable. Pointers to some literature on approximations can be found in recent Ph.D. theses [80], [22], [69], [100], [94], and in a variety of articles in the field such as (on message-passing variational approaches) [99], [75, 89, 131, 132], [73, 88], and [125]. On Sampling and hybrid of sampling and bounded inference see Bidyuk and Dechter [18], Bidyuk et al. [20] and Gogate and Dechter [65, 66, 67]. On anytime schemes for optimization see Marinescu and Dechter [93], Otten and Dechter [108].

Bibliography

[1] Bar-Yehuda R. A. Becker and D. Geiger. Random algorithms for the loop-cutset problem. In *Uncertainty in AI (UAI)*, pages 81–89, 1999. DOI: 10.1613/jair.638.

[2] M. J. Wainwright and M. I. Jordan. Variational inference in graphical models: The view from the marginal polytope. In *Proceedings of the Annual Allerton congerence on communication control and computing*, volume 41, pages 961–971. Citeseer, 2003.

[3] T. Schiex. Arc consistency for soft constraints. *International Conference on Principles and Practice of Constraint Programming (CP)*, pages 411–424, 2000. xlii

[4] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511811357.

[5] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000. DOI: 10.1109/18.825794.

[6] D. Allen and A. Darwiche. New advances in inference by recursive conditioning. In *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 2–10, 2003.

[7] 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. xxvi

[8] S. A. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey. *BIT*, 25:2–23, 1985. DOI: 10.1007/BF01934985.

[9] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal of Computing*, 27(4):942–959, 1998. DOI: 10.1137/S0097539796305109.

[10] R. Bayardo and D. Miranker. A complexity analysis of space-bound learning algorithms for the constraint satisfaction problem. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI)*, pages 298–304, 1996.

[11] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *Uncertainty in AI (UAI)*, pages 81–89, 1996.

[12] A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research (JAIR)*, 12:219–234, 2000. DOI: 10.1613/jair.638.

[13] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983. DOI: 10.1145/2402.322389.

[14] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[15] E. Bensana, M. Lemaitre, and G. Verfaillie. Earth observation satellite management. *Constraints*, 4(3):293–299, 1999. DOI: 10.1023/A:1026488509554.

[16] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972. xviii

[17] B. Bidyuk and R. Dechter. On finding w-cutset in Bayesian networks. In *Uncertainty in AI (UAI)*, 2004.

[18] B. Bidyuk and R. Dechter. Cutset sampling for Bayesian networks. *Journal of Artificial Intelligence Research (JAIR)*, 28:1–48, 2007. DOI: 10.1613/jair.2149. lii

[19] G. Hardy, J. Littlewood, and G. Polya. Inequalities. Cambridge Mathematical Library. Cambridge University Press. 1952. x, xii

[20] B. Bidyuk, R. Dechter, and E. Rollon. Active tuples-based scheme for bounding posterior beliefs. *Journal of Artificial Intelligence Research (JAIR)*, 39:335–371, 2010. DOI: 10.1613/jair.2945. lii

[21] E. Rollon and R. Dechter. New Mini-Bucket Partitioning Heuristics for Bounding the Probability of Evidence Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. xxiv

[22] B. Bidyuk. Exploiting graph-cutsets for sampling-based approximations in Bayesian networks. Technical report, Ph.D. thesis, Information and Computer Science, Universiy of California, Irvine, 2006. lii

[23] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the Association of Computing Machinery*, 44(2):165–201, 1997. DOI: 10.1145/256303.256306.

[24] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming (Lecture Notes in Computer Science*. Springer-Verlag, 2004. DOI: 10.1007/b95712.

[25] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *MFCS*, pages 19–36, 1997. DOI: 10.1007/BFb0029946.

[26] C. Borgelt and R. Kruse. *Graphical Models: Methods for Data Analysis and Mining*. Wiley, April 2002.

[27] C. Cannings, E. A. Thompson, and H. H. Skolnick. Probability functions on complex pedigrees. *Advances in Applied Probability*, 10:26–61, 1978. DOI: 10.2307/1426718.

[28] M.-W. Chang, L.-A. Ratinov, and D. Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 2012. DOI: 10.1007/s10994-012-5296-5.

[29] P. Beam, H. Kautz, Tian Sang, F. Bacchus, and T. Piassi. Combining component caching and clause learning for effective model counting. In *SAT*, 2004.

[30] Z. Collin, R. Dechter, and S. Katz. On the feasibility of distributed constraint satisfaction. In *Proc. of the 12th International Conference of Artificial Intelligence (IJCAI)*, pages 318–324, Sidney, Australia, 1991.

[31] Z. Collin, R. Dechter, and S. Katz. Self-stabilizing distributed constraint satisfaction. *The Chicago Journal of Theoretical Computer Science*, 3(4), special issue on self-stabilization, 1999. DOI: 10.4086/cjtcs.1999.010.

[32] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard (research note). *Artificial Intelligence*, 60:141–153, 1993. DOI: 10.1016/0004-3702(93)90036-B. i, lii

[33] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 125(1-2):5–41, 2001. DOI: 10.1016/S0004-3702(00)00069-2.

[34] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association of Computing Machinery*, 7(3), 1960. DOI: 10.1145/321033.321034.

[35] S. de Givry, J. Larrosa, and T. Schiex. Solving max-sat as weighted CSP. In *Principles and Practice of Constraint Programming (CP)*, 2003. DOI: 10.1007/978-3-540-45193-8_25.

[36] S. de Givry, I. Palhiere, Z. Vitezica, and T. Schiex. Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques. In *ICLP Workshop on Constraint Based Methods for Bioinformatics*, 2005. DOI: 10.1007/978-3-540-45193-8_25.

[37] R. Dechter and Y. El Fattah. Topological parameters for time-space tradeoff. *Artificial Intelligence*, pages 93–188, 2001. DOI: 10.1016/S0004-3702(00)00050-3.

[38] R. Dechter and R. Mateescu. The impact of and/or search spaces on constraint satisfaction and counting. In *Proc. of Constraint Programming (CP)*, pages 731–736, 2004. DOI: 10.1007/978-3-540-30201-8_56.

[39] R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007. DOI: 10.1016/j.artint.2006.11.003.

[40] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987. DOI: 10.1016/0004-3702(87)90002-6.

[41] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989. DOI: 10.1016/0004-3702(89)90037-4.

[42] R. Dechter and I. Rish. Directional resolution: The Davis–Putnam procedure, revisited. In *Principles of Knowledge Representation and Reasoning (KR)*, pages 134–145, 1994.

[43] R. Dechter and I Rish. Mini-buckets: A general scheme for approximating inference. *Journal of the ACM*, pages 107–153, 2002. li

[44] R. Dechter and P. van Beek. Local and global relational consistency. *Theoretical Computer Science*, pages 283–308, 1997. DOI: 10.1016/S0304-3975(97)86737-0.

[45] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990. DOI: 10.1016/0004-3702(90)90046-3.

[46] R. Dechter. Constraint networks. *Encyclopedia of Artificial Intelligence*, pages 276–285, 1992. DOI: 10.1002/9780470611821.fmatter.

[47] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. 12th Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996. DOI: 10.1016/S0004-3702(99)00059-4.

[48] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999. DOI: 10.1016/S0004-3702(99)00059-4. i

[49] R. Dechter. A new perspective on algorithms for optimizing policies under uncertainty. In *International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 72–81, 2000.

[50] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003. li

[51] R. Dechter. Tractable structures for constraint satisfaction problems. In *Handbook of Constraint Programming, Part I*, chapter 7, pages 209–244, Elsevier, 2006. DOI: 10.1016/S1574-6526(06)80011-8.

[52] S. Dughmi. Submodular functions: Extensions, distributions, and algorithms. A survey. *CoRR*, abs/0912.0322, 2009. li

[53] S. Even. Graph algorithms. In *Computer Science Press*, 1979.

[54] S. Dalmo F. Bacchus and T. Piassi. Algorithms and complexity results for #sat and Bayesian inference. In *FOCS*, 2003.

[55] S. Dalmo F. Bacchus and T. Piassi. Value elimination: Bayesian inference via backtracking search. In *Uncertainty in AI (UAI)*, 2003.

[56] M. Fishelson and D. Geiger. Exact genetic linkage computations for general pedigrees. *Bioinformatics*, 2002. DOI: 10.1093/bioinformatics/18.suppl_1.S189.

[57] M. Fishelson and D. Geiger. Optimizing exact genetic linkage computations. *RECOMB*, pages 114–121, 2003. DOI: 10.1145/640075.640089.

[58] M. Fishelson, N. Dovgolevsky, and D. Geiger. Maximum likelihood haplotyping for general pedigrees. *Human Heredity*, 2005. DOI: 10.1159/000084736.

[59] E. C. Freuder and M. J. Quinn. The use of lineal spanning trees to represent constraint satisfaction problems. *Technical Report 87-41*, University of New Hampshire, Durham, 1987.

[60] E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982. DOI: 10.1145/322290.322292.

[61] E. C. Freuder. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 32(1):755–761, 1985. DOI: 10.1145/4221.4225.

[62] E. C. Freuder. Partial constraint satisfaction. *Artificial Intelligence*, 50:510–530, 1992. DOI: 10.1016/0004-3702(92)90004-H.

[63] M. R Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. In *W. H. Freeman and Company*, San Francisco, 1979.

[64] N. Leone, G. Gottlob, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, pages 243–282, 2000. DOI: 10.1016/S0004-3702(00)00078-3.

[65] V. Gogate and R. Dechter. On combining graph-based variance reduction schemes. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:257–264, 2010. lii

[66] V. Gogate and R. Dechter. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011. DOI: 10.1016/j.artint.2010.10.009. lii

[67] V. Gogate and R. Dechter. Importance sampling-based estimation over and/or search spaces for graphical models. *Artificial Intelligence*, 184-185:38–77, 2012. DOI: 10.1016/j.artint.2012.03.001. lii

[68] V. Gogate, R. Dechter, B. Bidyuk, C. Rindt, and J. Marca. Modeling transportation routines using hybrid dynamic mixed networks. In *UAI*, pages 217–224, 2005.

[69] V. Gogate. Sampling algorithms for probabilistic graphical models with determinism. Technical report, Ph.D. thesis, Information and Computer Science, University of California, Irvine, 2009. xxvi, lii

[70] H. Hasfsteinsson, H. L. Bodlaender, J. R. Gilbert, and T. Kloks. Approximating treewidth, pathwidth and minimum elimination tree-height. In *Technical Report RUU-CS-91-1*, Utrecht University, 1991. DOI: 10.1006/jagm.1995.1009.

[71] R. A. Howard and J. E. Matheson. *Influence Diagrams*. 1984.

[72] A. Ihler, J. Hutchins, and P. Smyth. Learning to detect events with Markov-modulated poisson processes. *ACM Transactions on Knowledge Discovery from Data*, 1(3):13, 2007. DOI: 10.1145/1297332.1297337.

[73] A. T. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *UAI*, pages 397–406, 2012. lii

[74] P. Meseguer, J. Larrosa, and M. Sanchez. Pseudo-tree search with soft constraints. In *European Conference on Artificial Intelligence (ECAI)*, 2002.

[75] W. T. Freeman, J. S. Yedidia, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transaction on Information Theory*, pages 2282–2312, 2005. DOI: 10.1109/TIT.2005.850085. xxxvi, xlii, lii

[76] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.

[77] R. J. Bayardo and R. C. Schrag. Using CSP look-back techniques to solve real world sat instances. In *14th National Conference on Artificial Intelligence (AAAI)*, pages 203–208, 1997.

[78] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree-decompositions for reasoning in graphical models. *Artificial Intelligence*, 166(1-2):165–193, 2005. DOI: 10.1016/j.artint.2005.04.004.

[79] H. Kamisetty, E. P Xing, and C. J. Langmead. Free energy estimates of all-atom protein structures using generalized belief propagation. In *Proc. of the International Conference on Research in Computational Molecular Biology*, pages 366–380, 2007. DOI: 10.1007/978-3-540-71681-5_26.

[80] K. Kask. Approximation algorithms for graphical models. Technical report, Ph.D. thesis, Information and Computer Science, University of California, Irvine, CA, 2001. lii

[81] U. Kjæaerulff. Triangulation of graph-based algorithms giving small total state space. In *Technical Report 90-09*, Department of Mathematics and Computer Science, University of Aalborg, Denmark, 1990.

[82] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[83] J. Larrosa. Boosting search with variable-elimination. *CP*, pages 291–305, 2000. DOI: 10.1023/A:1020510611031.

[84] J. Larrosa and R. Dechter. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, 8(3):303–326, 2003. DOI: 10.1023/A:1025627211942.

[85] J.-L. Lassez and M. Mahler. On Fourier's algorithm for linear constraints. *Journal of Automated Reasoning*, 9, 1992. DOI: 10.1007/BF00245296.

[86] S. L. Lauritzen and D. J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[87] Q. Liu and A. T. Ihler. Bounding the partition function using holder's inequality. In *ICML*, pages 849–856, 2011. xviii, li

[88] Q. Liu and A. T. Ihler. Variational algorithms for marginal map. *CoRR*, abs/1302.6584, 2013. lii

[89] T. Jaakola, M. J. Wainwright, and A. S. Willskey. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, pages 2313–2335, 2005. DOI: 10.1109/TIT.2005.850091. i, lii

[90] D. Maier. The theory of relational databases. In *Computer Science Press*, Rockville, MD, 1983.

[91] R. Marinescu and R. Dechter. AND/OR branch-and-bound for graphical models. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 224–229, 2005.

[92] R. Marinescu and R. Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16–17):1457–1491, 2009. DOI: 10.1016/j.artint.2009.07.003. ii

[93] R. Marinescu and R. Dechter. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16–17):1492–1524, 2009. DOI: 10.1016/j.artint.2009.07.003. ii, xxvi, lii

[94] R. Marinescu. AND/OR search strategies for optimization in graphical models. Technical report, Ph.D. thesis, Information and Computer Science, Universiy of California, Irvine, 2007. lii

[95] J. P. Marques-Silva and K. A. Sakalla. Grasp-a search algorithm for propositional satisfiability. *IEEE Transaction on Computers*, pages 506–521, 1999. DOI: 10.1109/12.769433.

[96] R. Mateescu and R. Dechter. The relationship between AND/OR search and variable elimination. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 380–387, 2005.

[97] R. Mateescu and R. Dechter. AND/OR cutset conditioning. In *International Joint Conference on Artificial Intelligence (Ijcai)*, 2005.

[98] R. Mateescu and R. Dechter. A comparison of time-space scheme for graphical models. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 2346–2352, 2007.

[99] R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *Journal Artificial Intelligence Research (JAIR)*, 37:279–328, 2010. DOI: 10.1613/jair.2842. xxxiv, lii

[100] R. Mateescu. AND/OR search spaces for graphical models. Technical report, Ph.D. thesis, Information and Computer Science, Universiy of California, Irvine, 2007. DOI: 10.1016/j.artint.2006.11.003. lii

[101] R. McEliece, D. Mackay, and J. Cheng. Turbo decoding as an instance of pearl's "belief propagation" algorithm. 16(2):140–152, February 1998.

[102] L. G. Mitten. Composition principles for the synthesis of optimal multistage processes. *Operations Research*, 12:610–619, 1964. DOI: 10.1287/opre.12.4.610.

[103] P. J. Modi, W. Shena, M. Tambea, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005. DOI: 10.1016/j.artint.2004.09.003.

[104] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7(66):95–132, 1974. DOI: 10.1016/0020-0255(74)90008-5.

[105] K. P. Murphy. *Machine Learning; A Probabilistic Perspective*. 2012.

[106] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall series in Artificial Intelligence, 2000.

[107] N. J. Nillson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

[108] L. Otten and R. Dechter. Anytime AND/OR depth-first search for combinatorial optimization. *AI Communications*, 25(3):211–227, 2012. DOI: 10.3233/AIC-2012-0531. lii

[109] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988. xxxiv, xxxvi

[110] L. Portinale and A. Bobbio. Bayesian networks for dependency analysis: An application to digital control. In *Proc. of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 551–558, 1999.

[111] A. Dechter, R. Dechter, and J. Pearl. Optimization in constraint networks. In *Influence Diagrams, Belief Nets and Decision Analysis*, pages 411–425, John Wiley & Sons, 1990.

[112] B. D'Ambrosio, R. D. Shachter, and B. A. Del Favero. Symbolic probabilistic inference in belief networks. In *National Conference on Artificial Intelligence (AAAI)*, pages 126–131, 1990.

[113] I. Rish and R. Dechter. Resolution vs. search; two strategies for sat. *Journal of Automated Reasoning*, 24(1/2):225–275, 2000. DOI: 10.1023/A:1006303512524.

[114] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*. DOI: 10.1016/0004-3702(94)00092-1. i, lii

[115] D. G. Corneil, S. A. Arnborg, and A. Proskourowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Discrete Mathematics*, 8:277–284, 1987. DOI: 10.1137/0608024.

[116] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Proc. IJCAI*, pages 542–547, 1999. DOI: 10.1016/S0004-3702(01)00159-X.

[117] L. K. Saul and M. I. Jordan. Learning in Boltzmann trees. *Neural Computation*, 6:1173–1183, 1994. DOI: 10.1162/neco.1994.6.6.1174.

[118] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. 47(1/2/3):7–42, April 2002.

[119] R. Seidel. A new method for solving constraint satisfaction problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 338–342, 1981.

[120] G. R. Shafer and P. P. Shenoy. Axioms for probability and belief-function propagation, vol. 4, 1990.

[121] P. P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40:463–484, 1992. DOI: 10.1287/opre.40.3.463.

[122] P. P. Shenoy. Binary join trees for computing marginals in the Shenoy–Shafer architecture. *International Journal of Approximate Reasoning*, pages 239–263, 1997. DOI: 10.1016/S0888-613X(97)89135-9.

[123] K. Shoiket and D. Geiger. A proctical algorithm for finding optimal triangulations. In *14th National Conference on Artificial Intelligence (AAAI)*, pages 185–190, 1997.

[124] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering object categories in image collections. In *Proc. International Conference on Computer Vision*, 2005.

[125] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for map using message passing. In *UAI*, pages 503–510, 2008. lii

[126] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal of Computation*, 13(3):566–579, 1984. DOI: 10.1137/0213035.

[127] C. Terrioux and P. Jegou. Bounded backtracking for the valued constraint satisfaction problems. In *Constraint Programming (CP)*, pages 709–723, 2003.

[128] C. Terrioux and P. Jegou. Hybrid backtracking bounded by tree-decomposition of constraint networks. In *Artificial Intelligence*, 2003. DOI: 10.1016/S0004-3702(02)00400-9.

[129] P. Thébault, S. de Givry, T. Schiex, and C. Gaspin. Combining constraint processing and pattern matching to describe and locate structured motifs in genomic sequences. In *5th IJCAI Workshop on Modelling and Solving Problems with Constraints*, 2005.

[130] P. Beam, H. Kautz, Tian Sang, F. Bacchus, and T. Piassi. Combining component caching and clause learning for effective model counting. In *SAT*, 2004.

[131] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. DOI: 10.1561/2200000001. lii

[132] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5):1120–1146, 2003. DOI: 10.1109/TIT.2003.810642. lii

[133] Y. Weiss and J. Pearl. Belief propagation: Technical perspective. *Communications ACM*, 53(10):94, 2010. DOI: 10.1145/1831407.1831430.

[134] A. Willsky. Multiresolution Markov models for signal and image processing. 90(8):1396–1458, August 2002.

[135] C. Yanover and Y. Weiss. Approximate inference and protein folding. In *Proc. of Neural Information Processing Systems Conference*, pages 84–86, 2002.

[136] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research (JAIR)*, 1996. DOI: 10.1613/jair.305.

Author's Biography

RINA DECHTER

Rina Dechter's research centers on computational aspects of automated reasoning and knowledge representation including search, constraint processing, and probabilistic reasoning. She is a Chancellor's Professor of Computer Science at the University of California, Irvine. She holds a Ph.D. from UCLA, an M.S. degree in applied mathematics from the Weizmann Institute, and a B.S. in mathematics and statistics from the Hebrew University in Jerusalem. She is the author of *Constraint Processing* published by Morgan Kaufmann (2003), and of *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms* published by Morgan and Claypool (2013). She has co-authored close to 200 research papers and has served on the editorial boards of: *Artificial Intelligence*, the *Constraint Journal*, *Journal of Artificial Intelligence Research (JAIR)*, and *Journal of Machine Learning Research (JMLR)*. She is a Fellow of the American Association of Artificial Intelligence since 1994, was a Radcliffe Fellow during 2005–2006, received the 2007 Association of Constraint Programming (ACP) Research Excellence Award, and became an ACM Fellow in 2013. She was a Co-Editor-in-Chief of *Artificial Intelligence* from 2011 to 2018 and is the conference chair-elect for IJCAI-2022.