Chapter 9
Bounding Inference by Iterative message-passing schemes

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. Partially, because, like the mini-bucket scheme, it may not be able to generate an exact solution when more time is available because it will run out of memory first. The scheme 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. However, allocating IBP 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 9.1.1 (join-graph decompositions) *A* join-graph decomposition *for* $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$, *where* $\mathcal{P} = \{P_1, ..., P_n\}$, *ia 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 reduce this problem. We will describe here a scheme that avoids cycles relative to a single variable using the notion of arc-minimality also referred to as edge-minimality.

Definition 9.1.2 (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 9.1.1.*

If a 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 9.1.3 The example in Figure 9.1a 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 9.1.4 ((minimal) arc-labeled join-graph decompositions.) *An* arc-labeled decomposition *for $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$ is a four-tuple $D =< JG, \chi, \psi, \theta >$, where $JG = (V, E)$ 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:*

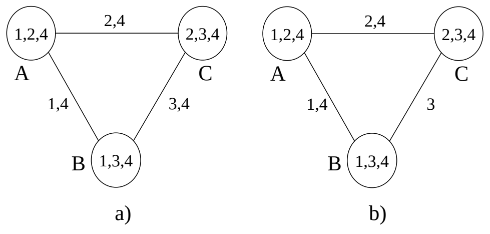

Figure 9.1: An arc-labeled decomposition

1. *For each function $P_i \in \mathbf{P}_G$, 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 9.1.5 (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 deleting variables from their labels while maintaining connectedness (if an edge label becomes empty, the edge can be deleted altogether). It is easy to see that,

Proposition 9.1.6 *A* minimal edge-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 edge-labeled join-graph is edge-minimal (no edge can be deleted), but not vice-versa. The mini-clustering approximation presented in the previous section, works by relaxing the join-tree requirement of exact inference into a collection of join-trees having smaller cluster size. It introduces some independencies in the original problem via node duplication and applies exact inference on the relaxed model requiring only 2 message passings. For the class of IJGP algorithms the idea is to relax the tree-structure requirement and use join-graphs, which do not introduce new independencies, and then utilize an iterative message-passing on the resulting cyclic structure.

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 [89]) of the underlying probability distribution relative to node-separation.

Since we plan to use minimally edge-labeled join-graphs to address over-counting problems, the question is what kind of independencies are captured by such graphs.

Definition 9.1.7 (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 $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$. 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 9.1.8 *Any arc-labeled join-graph decomposition $D = \langle JG, \chi, \psi, \theta \rangle$ of a belief network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$ is an I-map of P relative to edge-separation. Namely, any edge separation in D corresponds to conditional independence in The probability distribution represented by $\mathcal{B}$.*

For a proof see [80].

9.1.1 Algorithm IJGP

Applying CTE iteratively to minimal edge-labeled join-graphs yields algorithm *Iterative Join-Graph Propagation (IJGP)* described in Figure 9.2. 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.

```
Algorithm **Iterative Join Graph Propagation (IJGP)**
Input  An arc-labeled join-graph decomposition $\langle JG, \chi, \psi, \theta \rangle, JG = (V, E)$ for $\mathcal{B} = \langle X, D, G, P \rangle$. Evidence variables $var(e)$.
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 $P_i \in \psi(u)$ $\chi(u) := \chi(u) - var(e), \quad \forall u \in V$
    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:
       $h_{(u \to v)} = \sum_{elim(u,v)} \prod_{f \in A} f$
       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)

It is straightforward to show that:

Theorem 9.1.9 *We can show the following:*

1. *[74] 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. *[70] 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.*

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 $deg_u \cdot (|\psi(u)| + deg_u - 1) \cdot k^{|\chi(u)|}$, 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 $O(deg \cdot (n+N) \cdot k^{w+1})$.

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

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 will be provided 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, as we (may) discuss later.

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

9.1.2 The Case of Iterative Belief Propagation

Iterative belief propagation (IBP) is an iterative application of Pearl's algorithm for poly-trees [89], to any Bayesian network. We will describe IBP as an instance of join-graph propagation over a *dual graph.*

Definition 9.1.10 (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 edge-labeled edge subgraph of DG. A* minimal *dual join-graph is a dual join-graph for which none of the edge labels can be reduced while maintaining the connectedness property.*

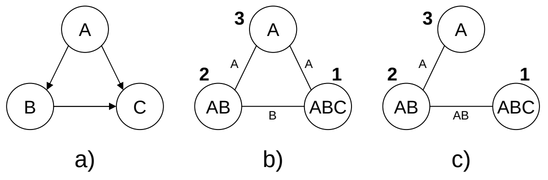

Figure 9.3: a) A belief network; b) A dual join-graph with singleton labels; c) A dual join-graph which is a join-tree

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 9.1.11 *Any Bayesian network has a minimal dual join-graph where each edge 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 edge between P_j and P_i by variable X_j, namely $l_{ij} = \{X_j\}$. It is easy to see that the resulting edge-labeled subgraph of the dual graph satisfies connectedness. The resulting labeled graph is a dual graph with singleton labels. ∎

Example 9.1.12 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 9.3a. Figure 9.3b shows a dual graph with singleton labels on the edges. Figure 9.3c 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 9.5, which 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

```
Algorithm IBP
Input: An edge-labeled dual join-graph $DG = (V, E, L)$ for a Bayesian network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$. 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: $h_u^v$ 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:
       $$h_u^v = \sum_{elim(u,v)} (p_u \cdot \prod_{\{h_i^u, i \in ne_v(u)\}} h_i^u)$$
  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 $DJ$,
  $P(F_i, e) = \alpha(\prod_{h_i^u, u \in ne(i)} h_i^u) \cdot p_u$;
  $P(X_i, e) = \alpha \sum_{scope(u) - \{X_i\}} P(F_i, e)$.
```

Figure 9.4: Algorithm Iterative Belief Propagation

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.

: Algorithm Iterative Belief Propagation

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 as long as the resulting clusters do not exceed the given i-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 9.7. 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 9.1.13 *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 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 9.1.14 Figure 9.8a shows the trace of procedure schematic mini-bucket(3) applied to the problem described in Figure 8.15a. The decomposition in Figure 9.8b is created by the algorithm 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 9.1.15 Figure 9.9 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. □

```
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 node 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).

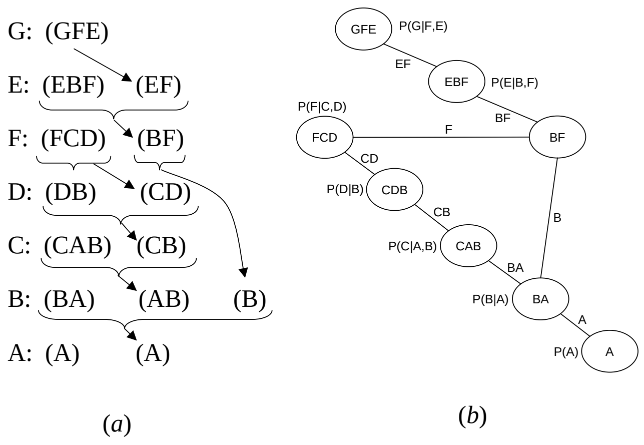

Figure 9.7: Join-graph decompositions.

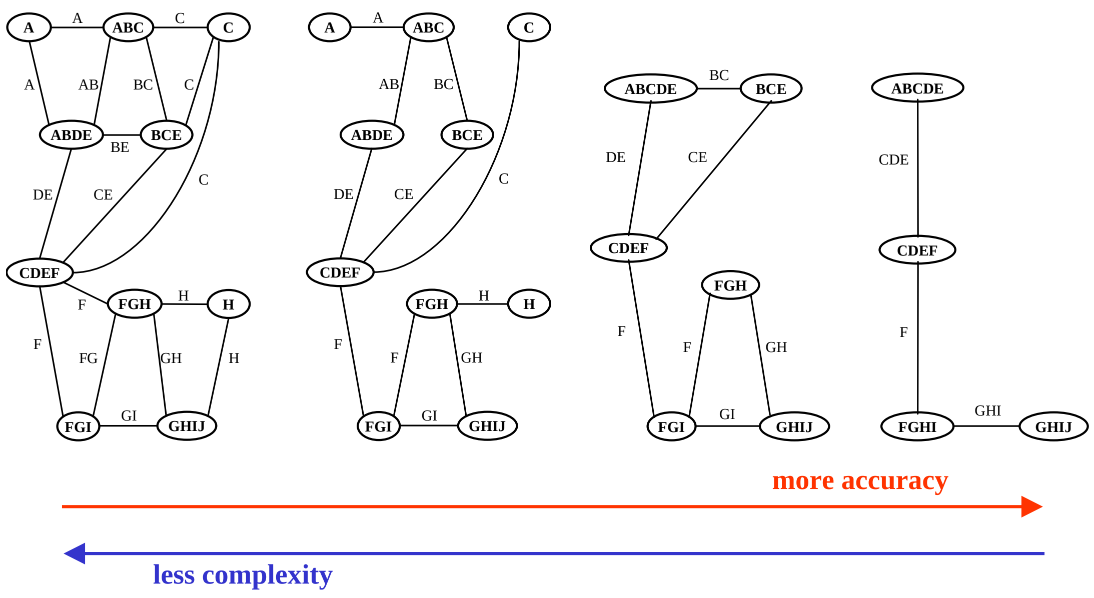

Figure 9.8: Join-graphs.

Bibliography

[1] Darwiche A. *Modeling and Reasoning with Bayesian Networks.* Cambridge University Press, 2009.

[2] Bar-Yehuda R A. Becker and D. Geiger. Random algorithms for the loop-cutset problem. In *Uncertainty in AI (UAI'99)*, pages 81–89, 1999.

[3] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[4] S. A. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT*, 25:2–23, 1985.

[5] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM J. Comput.*, 27(4):942–959, 1998.

[6] R. Bayardo and D. Miranker. A complexity analysis of space-bound learning algorithms for the constraint satisfaction problem. In *AAAI'96: Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 298–304, 1996.

[7] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *Uncertainty in AI (UAI'96)*, pages 81–89, 1996.

[8] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res. (JAIR)*, 12:219–234, 2000.

[9] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database ochemes. *Journal of the ACM*, 30(3):479–513, 1983.

[10] R.E. Bellman. *Dynamic Programming.* Princeton UNiversity Press, 1957.

[11] E. Bensana, M. Lemaitre, and G. Verfaillie. Earth observation satellite management. *Constraints*, 4(3):293–299, 1999.

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

[13] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming.* Academic Press, New York, 1972.

[14] B. Bidyuk and R. Dechter. On finding w-cutset in bayesian networks. In *Uncertainty in AI (UAI04)*, 2004.

[15] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming (Lecture Notes in Computer Science.* Springer-Verlag, 2004.

[16] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the Association of Computing Machinery*, 44, No. 2:165–201, 1997.

[17] H.L. Bodlaender. Treewidth: Algorithmic techniques and results. In *MFCS-97*, pages 19–36, 1997.

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

[19] Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 2012.

[20] M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1):199–227, 2004.

[21] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 125(1-2):5–41, 2001.

[22] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *arXiv preprint math/0307152*, 2003.

[23] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association of Computing Machinery*, 7(3), 1960.

[24] S. de Givry, J. Larrosa, and T. Schiex. Solving max-sat as weighted csp. *In Principles and Practice of Constraint Programming (CP-2003)*, 2003.

[25] 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.

[26] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[27] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[28] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Uncertainty in Artificial Intelligence (UAI'96)*, pages 211–219, 1996.

[29] R. Dechter. Bucket elimination: a unifying framework for processing hard and soft constraints. *CONSTRAINTS: An International Journal*, 2:51 – 55, 1997.

[30] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

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

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

[33] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[34] R. Dechter and Y. El Fattah. Topological parameters for time-space tradeoff. *Artificial Intelligence*, pages 93–188, 2001.

[35] R. Dechter and R. Mateescu. The impact of and/or search spaces on constraint satisfaction and counting. In *Proceeding of Constraint Programming (CP2004)*, pages 731–736, 2004.

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

[37] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987.

[38] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.

[39] R. Dechter and I. Rish. Directional resolution: The davis-putnam procedure, revisited. In *Principles of Knowledge Representation and Reasoning (KR-94)*, pages 134–145, 1994.

[40] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.

[41] R. Dechter and P. van Beek. Local and global relational consistency. *Theoretical Computer Science*, pages 283–308, 1997.

[42] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

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

[44] M. Fishelson, N. Dovgolevsky, and D. Geiger. Maximum likelihood haplotyping for general pedigrees. *Human Heredity*, 2005.

[45] M. Fishelson and D. Geiger. Exact genetic linkage computations for general pedigrees. *Bioinformatics*, 2002.

[46] Myan Fishelson and Dan Geiger. Optimizing exact genetic linkage computations. *RECOMB*, pages 114–121, 2003.

[47] Freuder. Partial constraint satisfaction. *Artificial Intelligence*, 50:510–530, 1992.

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

[49] 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.

[50] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[51] Nicola Leone Georg Gottlob and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artificial Intelligence*, pages 243–282, 2000.

[52] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Advances in Neural Information Processing Systems*, 21(1.6), 2007.

[53] V. Gogate. Sampling algorithms for probabilistic graphical models with determinism. Technical report, Ph.d. thesis, Information and Computer Science, Universiy of California, Irvine, 2009.

[54] Vibhav Gogate and Rina Dechter. Approximate inference algorithms for hybrid bayesian networks with discrete constraints. In *Proceeding of Uncertainty in Artificial Intelligence (UAI2005)*, 2005.

[55] Vibhav Gogate, Rina Dechter, Bozhena Bidyuk, Craig Rindt, and James Marca. Modeling transportation routines using hybrid dynamic mixed networks. In *UAI*, pages 217–224, 2005.

[56] Clifford Hildreth. A quadratic programming procedure. *Naval research logistics quarterly*, 4(1):79–85, 1957.

[57] 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.

[58] R. A. Howard and J. E. Matheson. *Influence diagrams.* 1984.

[59] F.V. Jensen. *Bayesian networks and decision graphs.* Springer-Verlag, New-York, 2001.

[60] V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *ICML*, 2010.

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

[62] K. Kask and R. Dechter. A general scheme for automatic search heuristics from specification dependencies. *Artificial Intelligence*, pages 91–131, 2001.

[63] K. Kask, R. Dechter, J. Larrosa, and G. Fabio. Bucket-tree elimination for automated reasoning. *Submitted -2001*, 2001.

[64] R. Dechter K. Kask and J. Larrosa. A general scheme for multiple lower bound computation in constraint optimization. *Principles and Practice of Constraint Programming (CP2001)*, pages 346–360, 2001.

[65] 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.

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

[67] N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. *Computer Vision–ECCV 2008*, pages 806–820, 2008.

[68] J Larrosa and R. Dechter. Dynamic combination of search and variable-elimination in csp and max-csp. *Submitted*, 2001.

[69] J. Larrosa and R. Dechter. Boosting search with variable-elimination. *Constraints*, pages 407–419, 2002.

[70] J Larrosa, K. Kask, and R. Dechter. Up and down mini-bucket: a scheme for approximating combinatorial optimization tasks. *Submitted*, 2001.

[71] Javier Larrosa and Rina Dechter. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, 8(3):303–326, 2003.

[72] J.-L. Lassez and M. Mahler. On fourier's algorithm for linear constraints. *Journal of Automated Reasoning*, 9, 1992.

[73] 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.

[74] 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.

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

[76] Radu Marinescu and Rina Dechter. Memory intensive and/or search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17):1492–1524, 2009.

[77] R. Mateescu. And/or search spaces for graphical models. Technical report, Ph.d. thesis, Information and Computer Science, Universiy of California, Irvine, 2007.

[78] R. Mateescu and R. Dechter. The relationship between AND/OR search and variable elimination. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence (UAI'05)*, pages 380–387, 2005.

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

[80] R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research (JAIR)*, 37:279–328, 2010.

[81] Robert Mateescu and Rina Dechter. And/or cutset conditioning. In *International Joint Conference on Artificial Intelligence (Ijcai-2005)*, 2005.

[82] Robert Mateescu, Kalev Kask, Vibhav Gogate, and Rina Dechter. Join-graph propagation algorithms. *J. Artif. Intell. Res. (JAIR)*, 37:279–328, 2010.

[83] L. G. Mitten. Composition principles for the synthesis of optimal multistage processes. *Operations Research*, 12:610–619, 1964.

[84] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7(66):95–132, 1974.

[85] K. P. Murphy. *Machine Learning; a probabilistic perspective.* 2012.

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

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

[88] P.Dagum and M.Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–155, 1993.

[89] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[90] L. Portinale and A. Bobbio. Bayesian networks for dependency analysis: an application to digital control. In *Proceedings of the 15th Conference on Uncertainty in Artifi cial Intelligence (UAI99)*, pages 551–558, 1999.

[91] 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.

[92] E. Bin R. Emek R. Dechter, K. Kask. Generating random solutions for constraint satisfaction problems. In *Proceedings of the National Conference of Artificial Intelligence (AAAI-02)*, 2002.

[93] R Dechter R. Mateescu and K. Kask. Tree approximation for belief updating. In *National Conference of Artificial Intelligence (AAAI-2002)*, pages 553–559, 2002.

[94] B. D'Ambrosio R.D. Shachter and B.A. Del Favero. Symbolic probabilistic inference in belief networks. In *National Conference on Artificial Intelligence (AAAI'90)*, pages 126–131, 1990.

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

[96] Irina Rish and Rina Dechter. Resolution versus search: Two strategies for sat. *J. Autom. Reasoning*, 24(1/2):225–275, 2000.

[97] E. Rollon and R. Dechter. New mini-bucket partitioning heuristics for bounding the probability of evidence. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI2010)*, pages 1199–1204, 2010.

[98] E. Rollon and J. Larrosa. Mini-bucket elimination with bucket propagation. *Principles and Practice of Constraint Programming-CP 2006*, pages 484–498, 2006.

[99] Emma Rollon and Rina Dechter. New mini-bucket partitioning heuristics for bounding the probability of evidence. In *AAAI*, 2010.

[100] D. Roth. On the hardness of approximate reasoning. 82(1-2):273–302, April 1996.

[101] 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.

[102] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Proc. IJCAI-99*, pages 542–547, 1999.

[103] L. K. Saul and M. I. Jordan. Learning in boltzmann trees. *Neural Computation*, 6:1173–1183, 1994.

[104] T. Schiex. Arc consistency for soft constraints. *Principles and Practice of Constraint Programming (CP2000)*, pages 411–424, 2000.

[105] R. Seidel. A new method for solving constraint satisfaction problems. In *International Joint Conference on Artificial Intelligece (Ijcai-81)*, pages 338–342, 1981.

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

[107] P.P. Shenoy. Valuation-based systems for bayesian decision analysis. *Operations Research*, 40:463–484, 1992.

[108] P.P. Shenoy. Binary join trees for computing marginals in the shenoy-shafer architecture. *International Journal of approximate reasoning*, pages 239–263, 1997.

[109] K. Shoiket and D. Geiger. A proctical algorithm for finding optimal triangulations. In *Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 185–190, 1997.

[110] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. *UAI*, 2008.

[111] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254, 2011.

[112] David Sontag and Tommi Jaakkola. Tree block coordinate descent for MAP in graphical models. In *AI & Statistics*, pages 544–551. JMLR: W&CP 5, 2009.

[113] 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.

[114] P. Thbault, S. de Givry, T. Schiex, and C. Gaspin. Combining constraint processing and pattern matching to describe and locate structured motifs in genomic sequences. In *Fifth IJCAI-05 Workshop on Modelling and Solving Problems with Constraints*, 2005.

[115] Bozhena Bidyuk Craig Rindt Vibhav Gogate, Rina Dechter and James Marca. Modeling transportation routines using hybrid dynamic mixed networks. In *Proceeding of Uncertainty in Artificial Intelligence (UAI2005)*, 2005.

[116] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *Information Theory, IEEE Transactions on*, 51(11):3697–3717, 2005.

[117] Jack Warga. Minimizing certain convex functions. *Journal of the Society for Industrial & Applied Mathematics*, 11(3):588–593, 1963.

[118] Yair Weiss and Judea Pearl. Belief propagation: technical perspective. *Commun. ACM*, 53(10):94, 2010.

[119] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report 2004-040, MERL, May 2004.

[120] N.L. Zhang and D. Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research (JAIR)*, 1996.