Algorithms on Junction Trees

Huizhen Yu

janey.yu@cs.helsinki.fi
Dept. Computer Science, Univ. of Helsinki

Probabilistic Models, Spring, 2010

Outline

Junction Trees

Outline

Cluster Trees

The clustering approach for complex networks:

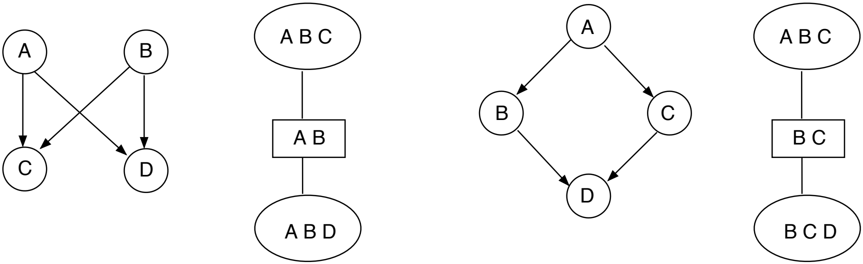
- By belief propagation studied in the previous lectures, we can only obtain the marginal distributions of each variable for singly connected networks.
- For more complex networks as well as for computing the marginal distributions of multiple variables, a natural approach is to cluster the variables and to arrange them in a graph with a simpler structure.

Let U be a set of variables.

- A *cluster tree* $\mathcal{T}$ over U is an undirected tree of clusters of variables from U. The nodes are subsets of U, and the union of all nodes is U.
- Each edge between two adjacent nodes C_1 and C_2 in a cluster tree $\mathcal{T}$ is labeled with $C_1 \cap C_2$, called the *separator*.
- Denote the set of nodes of $\mathcal{T}$ by $\mathcal{C}$ and the set of edges of $\mathcal{T}$ by $\mathcal{S}$.
- There may be multiple edges labeled with the same separator. We index each edge by the associated separator, nevertheless; and we allow $\mathcal{S}$ to contain such repetitions, for notational simplicity.

Cluster Trees

Examples of cluster trees for two DAGs. Clusters of variables are shown inside the nodes, while separators are shown in the squares along each edge.



We assoicate

- each node and separator of the cluster tree $\mathcal{T}$ with a function $\phi_C(x_C)$ and $\phi_S(x_S)$ of the variables in their variable sets, respectively.

A Technique with Invariance Property

A seemingly trivial technique that we will rely on:

- If a can be expressed as b/c, then a can also be expressed as

$$a = b'/c', \quad \text{where} \quad b' = b \cdot c'/c,$$

  assuming $c' \neq 0$. (This will be carefully extended later to the case where c, c' can be zero.)

We think of (b, c) and (b', c') as different representations for a. (We may not know a, but we have access to these representations.)

The technique provides a way to keep a unchanged when we want to apply certain modifications to some part of its representation.

It will be applied to functions associated with clusters of variables. A simple example is given next.

An Example of Varying Representations

Suppose a joint distribution $P(X, Y, Z)$ is strictly positive with

$$p(x, y, z) = f(x, y)\frac{1}{h(y)}g(y, z). \tag{1}$$

It is desirable to obtain the marginal distribution $p(x, y)$ and $p(y, z)$. Then, define

$$h^*(y) = \sum_z g(y, z), \qquad f^*(x, y) = f(x, y) \cdot h^*(y)/h(y),$$

and we can express $p(x, y, z)$ as

$$p(x, y, z) = f(x, y)\frac{1}{h(y)}g(y, z) = f^*(x, y)\frac{1}{h^*(y)}g(y, z).$$

Indeed $f^*(x, y) = p(x, y)$. Similarly, define

$$h^\dagger(y) = \sum_x f^*(x, y), \qquad g^\dagger(y, z) = g(y, z) \cdot h^\dagger(y)/h^*(y),$$

and we can further express $p(x, y, z)$ as

$$p(x, y, z) = f^*(x, y)\frac{1}{h^*(y)}g(y, z) = f^*(x, y)\frac{1}{h^\dagger(y)}g^\dagger(y, z).$$

Indeed $g^\dagger(y, z) = p(y, z), h^\dagger(y) = p(y)$. Thus, starting with the representation for p in Eq. (1), we finished with a representation for p as

$$p(x, y, z) = p(x, y)\frac{1}{p(y)}p(y, z).$$

Local Modifications with Local Consistency and Global Invariance Properties

The method in the previous slide looks promising for obtaining marginal distributions of clusters of variables by local modifications:

- Suppose two adjacent nodes C_1 and C_2 of $\mathcal{T}$ with separator S exchange information by sending a "message" from C_1 to C_2:

$$\phi^*_S(x_S) = \sum_{x_{C_1 \setminus S}} \phi_{C_1}(x_{C_1}), \qquad \phi^*_{C_2}(x_{C_2}) = \phi_{C_2}(x_{C_2}) \cdot \phi^*_S(x_S)/\phi_S(x_S),$$

  and next, a "message" from C_2 to C_1:

$$\phi^\dagger_S(x_S) = \sum_{x_{C_2 \setminus S}} \phi^*_{C_2}(x_{C_2}), \qquad \phi^\dagger_{C_1}(x_{C_1}) = \phi_{C_1}(x_{C_1}) \cdot \phi^\dagger_S(x_S)/\phi^*_S(x_S).$$

  Then, assuming there is no trouble with divisions, we have

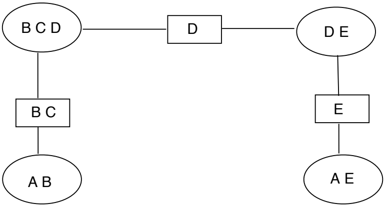
$$\begin{aligned} \sum_{x_{C_1 \setminus S}} \phi^\dagger_{C_1}(x_{C_1}) &= \sum_{x_{C_1 \setminus S}} \phi_{C_1}(x_{C_1}) \cdot \phi^\dagger_S(x_S)/\phi^*_S(x_S) \\ &= \phi^\dagger_S(x_S) = \sum_{x_{C_2 \setminus S}} \phi^*_{C_2}(x_{C_2}), \end{aligned}$$

  and the edge S is said to be *sum-consistent*.

Cluster Trees Can Fail Global Consistency

Applying such local modifications as in the previous slide to the tree $\mathcal{T}$ edge by edge, we wish to eventually obtain global consistency, i.e., for each variable x_A, the margin $\sum_{x_{C\setminus A}} \phi_C(x_C)$ is the same for all $C \supseteq A$. But this cannot be guaranteed with a cluster tree, as shown below.

Example: A cluster tree over binary variables. All variables except for A are in state y, while A in the bottom-left node is in state y and A in the bottom-right node can be in either states.



This motivates us to consider:

- cluster trees in which a variable does not appear at "isolated" locations,

so that global consistency can be obtained eventually by local modifications.

Definition of Junction Tree

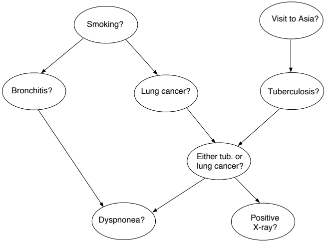
Definition: A cluster tree $\mathcal{T}$ is called a *junction tree* if, for each pair of nodes C_1, C_2 of $\mathcal{T}$, $C_1 \cap C_2$ is contained in every node on the unique path in $\mathcal{T}$ between C_1 and C_2.

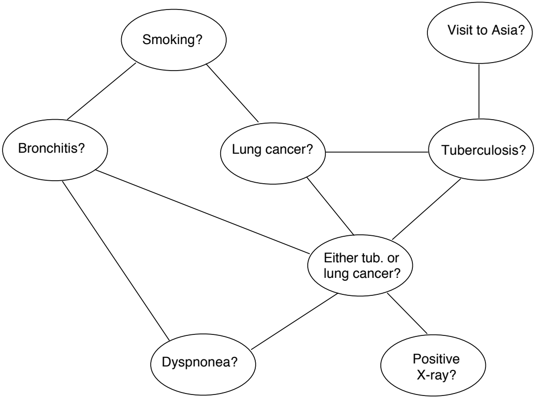
- The definition is equivalent to that, for all $u \in U$, the set of C in $\mathcal{C}$ containing u induces a connected subtree of $\mathcal{T}$.

Junction trees are also called *join trees* in the literature.

The heuristic arguments we just discussed will be developed rigorously on junction trees, in a rather general way.

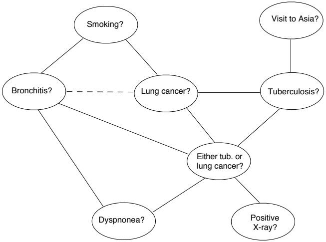
Example: Asia Network and its Moral Graph

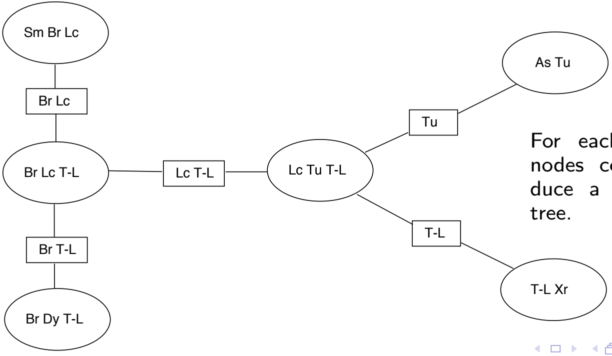




Asia Example: Triangulated Moral Graph and Junction Tree

As : Visit to Asia? Sm : Smoking?
Br : Bronchitis? Lc : Lung Cancer?
Tu : Tuberculosis? T-L : Either Tuberculosis or lung cancer?
Dy : Dyspnonea? Xr : Positive X-ray?





For each variable, the nodes containing it induce a connected subtree.

Outline

Junction Trees

Motivation: Cluster Trees and Heuristic Arguments

Representations with Potentials

Flow Passing Algorithm: Sum-Flows

Flow Passing Algorithm: Max-Flows

Graph-Theoretic Properties and Building Junction Trees

Definitions

Let $\mathcal{T}$ be a junction tree over U with the vertex set $\mathcal{C}$ and edge (separator) set $\mathcal{S}$.

- A collection of non-negative functions $\Phi = \big(\{\phi_C, C \in \mathcal{C}\}, \{\phi_S, S \in \mathcal{S}\}\big)$ will be called a *charge* on $\mathcal{T}$.
- Individual functions in Φ will be called *potentials* (on the vertices C or separators S).
- The *contraction* of Φ is the function of x_U defined by

$$\frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)}, \tag{2}$$

  where the expression on the right-hand side is *interpreted to be* 0 *whenever the denominator is* 0.
- If a function f equals the contraction of Φ, i.e.,

$$f(x) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)}, \quad \forall x,$$

  then Φ will be called a *representation* for f on $\mathcal{T}$.

Examples of Potentials and Representations

Suppose G is an undirected graph, $\mathcal{C}$ is the set of cliques of G, and $P(X)$ factorizes according to G. Then for some constant $\alpha > 0$,

$$p(x) = \alpha \prod_{C \in \mathcal{C}} \phi_C(x_C), \quad \text{for some non-negative functions } \phi_C.$$

So $\Phi = (\{\phi_C, C \in \mathcal{C}\}, \{\phi_S, S \in \mathcal{S}\})$ with $\phi_S \equiv 1$ is a representation for p/α on $\mathcal{T}$, and correspondingly, p/α equals the contraction of Φ.

Suppose $G = (V, E)$ is a DAG, $P(X)$ factorizes recursively according to G:

$$p(x) = \prod_{v \in V} p(x_v \,|\, x_{\text{pa}(v)}),$$

and $\mathcal{C}$ is the set of cliques of G^m. Then $\Phi = (\{\phi_C, C \in \mathcal{C}\}, \{\phi_S, S \in \mathcal{S}\})$ is a representation for p on $\mathcal{T}$, where Φ is obtained by:

- first, let $\phi_C \equiv 1, \phi_S \equiv 1$ for all cliques C and separators S;
- then, for each v, choose exactly one clique C containing v and $\text{pa}(v)$, and multiply $\phi_C(x_C)$ by $p(x_v \,|\, x_{\text{pa}(v)})$.

Correspondingly, p is a contraction of Φ.

Examples of Potentials and Representations

Continue with the DAG example in the previous slide. Suppose also that the received evidence $\mathbf{e}$ is expressible in the factor form

$$\mathbf{e}(x) = \prod_{v \in V} \ell_v(x_v), \quad \text{where} \quad \ell_v(x_v) \in \{0, 1\},$$

(as in Lec. 9 and 10). Then

$$p(x \,\&\, \mathbf{e}) = \prod_{v \in V} p(x_v \,|\, x_{\text{pa}(v)})\, \ell_v(x_v).$$

And $\Phi = (\{\phi_C, C \in \mathcal{C}\}, \{\phi_S, S \in \mathcal{S}\})$ is a representation for the function $p(x \,\&\, \mathbf{e})$ on $\mathcal{T}$, where Φ is obtained similarly as in the DAG example by:

- first, let $\phi_C \equiv 1, \phi_S \equiv 1$ for all cliques C and separators S;
- next, for each v, choose exactly one clique C containing v and $\text{pa}(v)$, and multiply $\phi_C(x_C)$ by $p(x_v \,|\, x_{\text{pa}(v)})$;
- and finally, for each v, choose one (or multiple) C containing v, and multiply $\phi_C(x_C)$ by $\ell_v(x_v)$.

Correspondingly, the function $p(x \,\&\, \mathbf{e})$ is a contraction of Φ.

Outline

Junction Trees

Motivation: Cluster Trees and Heuristic Arguments

Representations with Potentials

Flow Passing Algorithm: Sum-Flows

Flow Passing Algorithm: Max-Flows

Graph-Theoretic Properties and Building Junction Trees

Basic Operations, Definitions and Simplified Notation

- If ϕ is a potential on V (i.e., a non-negative function of x_V) and ψ a potential on W, the sum and multiplication of the two

$$\phi + \psi, \qquad \phi\psi$$

stand for the functions of $x_{V \cup W}$ given, respectively, by

$$(\phi + \psi)(x_{V\cup W}) = \phi(x_V) + \psi(x_W), \quad (\phi\psi)(x_{V\cup W}) = \phi(x_V)\psi(x_W).$$

- Division is defined likewise, except when *dividing by zero*:

$$(\phi/\psi)(x_{V\cup W}) = \begin{cases} \phi(x_V)/\psi(x_W), & \psi(x_W) \neq 0; \\ 0, & \psi(x_W) = 0. \end{cases}$$

- For a potential $\phi(x_V)$ on V and $W \subseteq V$, we denote the function

$$\sum_{x_{V\setminus W}} \phi(x_V) \;\text{ by }\; \sum_{V\setminus W} \phi.$$

And we call it the *sum-margin* of ϕ on W.

- Similarly, under the above conditions, we denote the function

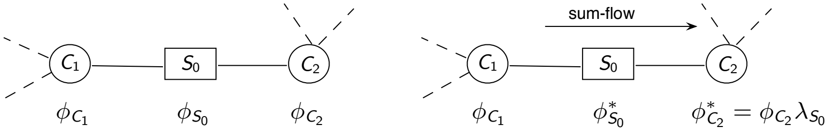
$$\max_{x_{V\setminus W}} \phi(x_V) \;\text{ by }\; \max_{V\setminus W} \phi.$$

And we call it the *max-margin* of ϕ on W.

Flow of Information between Adjacent Vertices

A *sum-flow* (flow, for short) refers to the following operation involving a pair of adjacent vertices C_1, C_2 of $\mathcal{T}$ and their separator S_0.

Passing a sum-flow from the source C_1 to the sink C_2 changes *only* the potentials ϕ_{C_2} and ϕ_{S_0} to:



where

$$\phi^*_{S_0} = \sum_{C_1 \setminus S_0} \phi_{C_1}, \qquad \lambda_{S_0} = \phi^*_{S_0} / \phi_{S_0}, \quad \text{(called \textit{update ratio})}.$$

- Each flow affects only one vertex and one separator.
- Φ is unaffected by the passage of any sum-flow if and only if it is *sum-consistent*, i.e.,

$$\sum_{C \setminus S} \phi_C = \phi_S \quad \text{for any } C \text{ and neighboring } S.$$

Invariance of Contraction w.r.t. Sum-Flows

Fact: *The contraction of the charge Φ remains the same after a sum-flow.*

Implications:

- If Φ is a representation for f, then by passing flows we can find representations for f suitable for our problem without worrying about changing f.
- We can modify the charge by passing flows and use the property of a junction tree to obtain certain global consistency.

We next verify the above fact. It holds for cluster trees in general, and we won't need the junction tree property.

Let f be the contraction of Φ and f^* the contraction of Φ^* resulted after a sum-flow from C_1 to C_2 via the edge S_0:

$$f(x) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)}, \qquad f^*(x) = \frac{\prod_{C \in \mathcal{C}} \phi^*_C(x_C)}{\prod_{S \in \mathcal{S}} \phi^*_S(x_S)}.$$

Recall that Φ^* differs from Φ only in the potentials on C_2 and S_0.

Invariance of Contraction w.r.t. Sum-Flows

By the definition of a sum-flow, we have

$$\phi^*_{S_0} = \sum_{C_1 \setminus S_0} \phi_{C_1}, \qquad \lambda_{S_0} = \phi^*_{S_0}/\phi_{S_0}, \qquad \phi^*_{C_2} = \phi_{C_2}\lambda_{S_0},$$

$$f(x) = \frac{\prod_{C\in\mathcal{C}} \phi_C(x_C)}{\prod_{S\in\mathcal{S}} \phi_S(x_S)}, \qquad f^*(x) = \frac{\prod_{C\in\mathcal{C}\setminus\{C_2\}} \phi_C(x_C)}{\prod_{S\in\mathcal{S}\setminus\{S_0\}} \phi_S(x_S)} \cdot \phi^*_{C_2}(x_{C_2}) \cdot \frac{1}{\phi^*_{S_0}(x_{S_0})}.$$

For each x, consider the three possible cases:

Case (i): $\lambda_{S_0}(x_{S_0}) > 0$. Clearly, $f^*(x) = f(x)$.

Case (ii): $\phi_{S_0}(x_{S_0}) = 0$. Then $f(x) = 0$ by definition, while

$$\lambda_{S_0}(x_{S_0}) = 0 \;\Rightarrow\; \phi^*_{C_2}(x_{C_2}) = 0 \;\Rightarrow\; f^*(x) = 0.$$

Case (iii): $\phi_{S_0}(x_{S_0}) > 0$ but $\phi^*_{S_0}(x_{S_0}) = 0$. Then $f^*(x) = 0$ by definition, while

$$\phi^*_{S_0}(x_{S_0}) = 0 \;\Rightarrow\; \sum_{y_{C_1}:y_{S_0}=x_{S_0}} \phi_{C_1}(y_{C_1}) = 0 \;\Rightarrow\; \phi_{C_1}(x_{C_1}) = 0,$$

where the last step follows from ϕ_{C_1} being non-negative, so $f(x) = 0$.

This show $f^*(x) = f(x), \forall x$.

Active Flows and Flow Scheduling

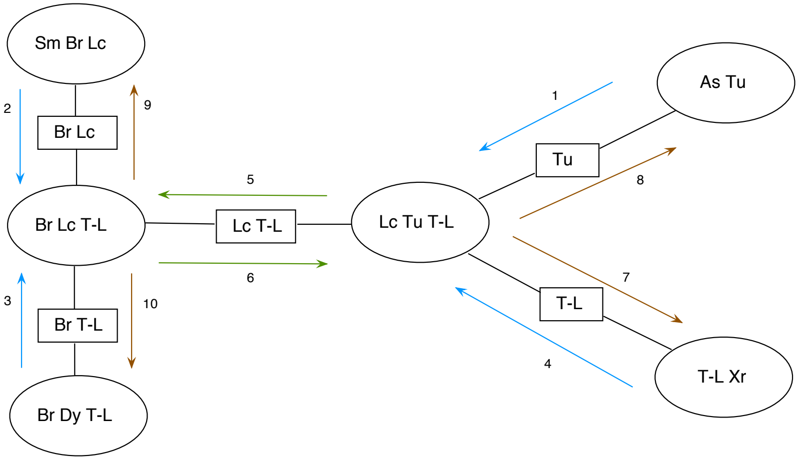
Terminologies:

- A subtree $\mathcal{T}'$of $\mathcal{T}$: a connected set of vertices of $\mathcal{T}$ with edges between them.
- A *neighbor* of a subtree $\mathcal{T}'$: a vertex C that is not a vertex of $\mathcal{T}'$, but is connected to a vertex of $\mathcal{T}'$ by an edge of $\mathcal{T}$.
- A *schedule* of flows: an ordered list of directed edges of $\mathcal{T}$, specifying which flows are to be passed and in what order.
- Relative to a schedule, a flow is called *active* if, before it is sent, the source has already received active flows from all its neighbors, with the possible exception of the sink; and it is the *first* flow (along the directed edge) in the list with this property.
- A schedule is *full* if it contains an active flow in each direction along every edge of $\mathcal{T}$. It is *active* if it contains only active flows, and *fully active* if it is both full and active.

From any full schedule, a fully active schedule can be obtained by omitting inactive flows. For any tree $\mathcal{T}$, there exists a fully active schedule, as can be seen easily.

A Fully Active Schedule of Flows for the Asia Example

Numbers indicate the order for sending the flows. Flows with the same color can be passed in parallel.



Flow Passing Algorithm

Algorithm:

- Start with an initial representation Φ^0 for a function f on $\mathcal{T}$.
- Modify the representation Φ^t progressively by passing a sequence of flows according to some schedule.

To analyze this algorithm:

- We say a subtree is *live*, if it has received active flows from all of its neighbors.

More notation: For a subtree $\mathcal{T}'$ with vertices $\mathcal{C}'$ and separators $\mathcal{S}'$,

- the *base* U' of $\mathcal{T}'$ is the collection of variables associated with $\mathcal{T}'$, i.e., $U' = \big(\cup_{C \in \mathcal{C}'} C \big) \cup \big(\cup_{S \in \mathcal{S}'} S \big)$;
- for a charge $\Phi = \big(\{\phi_C, C \in \mathcal{C}\}, \{\phi_S, S \in \mathcal{S}\} \big)$, its *restriction to* $\mathcal{T}'$ is the sub-collection of potentials

$$\Phi_{\mathcal{T}'} = \big(\{\phi_C, C \in \mathcal{C}'\}, \{\phi_S, S \in \mathcal{S}'\} \big),$$

  and its *potential on* $\mathcal{T}'$ is the contraction of $\Phi_{\mathcal{T}'}$,

$$\frac{\prod_{C \in \mathcal{C}'} \phi_C}{\prod_{S \in \mathcal{S}'} \phi_S}, \quad \text{(which is a function of } x_{U'}\text{)}.$$

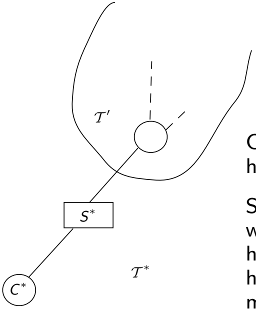
Reaching Equilibrium: Sum-Marginal Representation

Theorem 1: *Whenever a subtree $\mathcal{T}'$ is live, the potential on $\mathcal{T}'$ equals the sum-margin $f_{U'}$ of f on the base U' of $\mathcal{T}'$:*

$$f_{U'} = \sum_{U \setminus U'} f.$$

Proof: We will use induction. The contraction f is invariant w.r.t. the passage of flows, so the statement holds trivially if $\mathcal{T}' = \mathcal{T}$.

Consider any time when $\mathcal{T}'$ is live.



- Let C^* be the last neighbor of $\mathcal{T}'$ to have passed a flow (active or not) into $\mathcal{T}'$.
- Let $\mathcal{T}^*$ be the subtree obtained by adding C^* and the associated edge S^* to $\mathcal{T}'$.

Clearly, $\mathcal{T}^*$ is live: otherwise, its subtree $\mathcal{T}'$ could not have received active flows from all its neighbors.

Since this construction process can be repeated until we obtain the entire tree $\mathcal{T}$ for which the statement holds, we may by induction assume that the statement holds for $\mathcal{T}^*$. What we need to show now is that it must also hold for $\mathcal{T}'$.

Proof of Theorem 1 Cont'd

We have, just before the last flow from C^* into $\mathcal{T}'$,

$$f_{U^*} = \frac{\phi_{C^*}\alpha_{U'}}{\phi_{S^*}}, \text{ (potential on } \mathcal{T}^*\text{)}, \quad \text{where } \alpha_{U'} = \frac{\prod_{C\in\mathcal{C}'}\phi_C}{\prod_{S\in\mathcal{S}'}\phi_S}, \text{ (potential on } \mathcal{T}'\text{)}.$$

After the flow, $\alpha_{U'}$ is replaced by

$$\alpha^*_{U'} = \alpha_{U'}\lambda_{S^*}, \quad \text{where} \quad \lambda_{S^*} = \frac{\sum_{C^*\setminus S^*}\phi_{C^*}}{\phi_{S^*}}.$$

By the property of a junction tree,

- $S^* = C^* \cap U'$, so $C^* \setminus S^* = U^* \setminus U'$.

Therefore, for each $x_{U'}$,

$$f_{U'}(x_{U'}) = \sum_{U^*\setminus U'} f_{U^*}(x_{U'}, x_{U^*\setminus U'}) = \sum_{C^*\setminus S^*} f_{U^*}(x_{U'}, x_{C^*\setminus S^*})$$
$$= \alpha_{U'}(x_{U'}) \cdot \frac{\sum_{C^*\setminus S^*}\phi_{C^*}(x_{C^*\setminus S^*}, x_{S^*})}{\phi_{S^*}(x_{S^*})}.$$

If $\phi_{S^*}(x_{S^*}) > 0$, then $f_{U'}(x_{U'}) = \alpha^*_{U'}(x_{U'})$ clearly. If $\phi_{S^*}(x_{S^*}) = 0$, then $f_{U'}(x_{U'}) = 0$ by definition, while $\alpha^*_{U'}(x_{U'}) = 0$ because $\lambda_{S^*}(x_{S^*}) = 0$. This shows $f_{U'}(x_{U'}) = \alpha^*_{U'}(x_{U'}), \forall x_{U'}$. Since any possible, subsequent flows within $\mathcal{T}'$ do not change the potential on $\mathcal{T}'$ (which is a contraction of $\Phi_{\mathcal{T}'}$), the proof is complete.

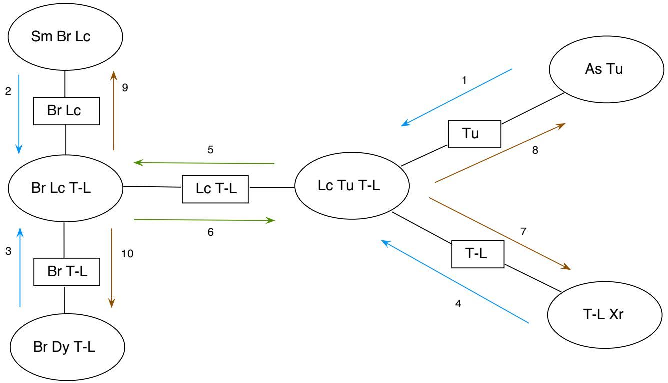
Reaching Equilibrium: Sum-Marginal Representation

Implications of Theorem 1:

- Whenever each node C is live, its potential is f_C.
- Any time after active flows have passed in both directions across an edge, the potential for the associated separator S is f_S.
- Any time after active flows have passed in both directions across an edge between C and D with associated separator S, the tree is sum-consistent along S.
- After passage of a full schedule of flows, the resulting charge is the *marginal charge* $\Phi_f = \big(\{f_C, C \in \mathcal{C}\}, \{f_S, S \in \mathcal{S}\}\big)$ of f, i.e.,

$$f(x) = \frac{\prod_{C \in \mathcal{C}} f_C(x_C)}{\prod_{S \in \mathcal{S}} f_S(x_S)}.$$

Asia Example



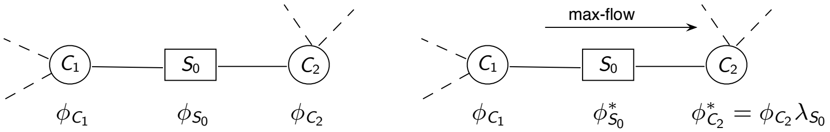
- Suppose no evidence is entered initially. After flows $1-4$, the potential on the subtree with nodes (Br, Lc, T-L), (Lc, Tu, T-L) is $p(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L})$. (The potential on the entire tree is always p.)
- suppose the evidence $\mathbf{e}: \{\text{Sm} = y, \text{Xr} = y, \text{As} = n\}$ is entered initially. Then after flows $1-4$, the potential on the subtree with nodes (Br, Lc, T-L), (Lc, Tu, T-L) is $p(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L}, \& \,\mathbf{e})$. (The potential on the entire tree is always $p(\cdot \,\& \,\mathbf{e})$.)

Outline

Passing Max-Flows

Replacing sum-flows with *max-flows*, we obtain a set of parallel results.

Passing a max-flow from the source C_1 to the sink C_2 changes *only* the potentials ϕ_{C_2} and ϕ_{S_0} to:



where

$$\phi^*_{S_0} = \max_{C_1 \setminus S_0} \phi_{C_1}, \qquad \lambda_{S_0} = \phi^*_{S_0}/\phi_{S_0}, \quad \text{(update ratio)}.$$

- Each max-flow affects only one vertex and one separator.
- Φ is unaffected by the passage of any max-flow if and only if it is *max-consistent*, i.e.,

$$\max_{C \setminus S} \phi_C = \phi_S \quad \text{for any } C \text{ and neighboring } S.$$

Algorithm and Analysis

The algorithm is the same as before with the flows being max-flows:

- Start with an initial representation Φ^0 for a function f on $\mathcal{T}$.
- Modify the representation Φ^t progressively by passing a sequence of max-flows according to some schedule.

The analysis of the algorithm is also almost the same – we only need to verify the following arguments:

- Invariance of contraction with respect to max-flows.
- Whenever a subtree is live, its potential is the max-margin of f.

The verification will be given after we state the theorem and its implications in the next slide.

Reaching Equilibrium: Max-Marginal Representation

Theorem 2: *Whenever a subtree $\mathcal{T}'$ is live, the potential on $\mathcal{T}'$ equals the max-margin $f_{U'}^{max}$ of f on the base U' of $\mathcal{T}'$:*

$$f_{U'}^{max} = \max_{U \setminus U'} f.$$

Implications:

- Whenever each node C is live, its potential is $f_C^{\max}$.
- Any time after active flows have passed in both directions across an edge, the potential for the associated separator S is $f_S^{\max}$.
- Any time after active flows have passed in both directions across an edge between C and D with associated separator S, the tree is max-consistent along S.
- After passage of a full schedule of flows, the resulting charge is the max-marginal charge $\Phi_f^{\max} = \left(\{f_C^{\max}, C \in \mathcal{C}\}, \{f_S^{\max}, S \in \mathcal{S}\}\right)$ of f, i.e.,

$$f(x) = \frac{\prod_{C \in \mathcal{C}} f_C^{\max}(x_C)}{\prod_{S \in \mathcal{S}} f_S^{\max}(x_S)}.$$

Invariance of Contraction w.r.t. Max-Flows

Let f be the contraction of Φ and f^* the contraction of Φ^* after a max-flow from C_1 to C_2 via the edge S_0. We have

$$\phi^*_{S_0} = \max_{C_1 \setminus S_0} \phi_{C_1}, \qquad \lambda_{S_0} = \phi^*_{S_0} / \phi_{S_0}, \qquad \phi^*_{C_2} = \phi_{C_2} \lambda_{S_0},$$

$$f(x) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)}, \qquad f^*(x) = \frac{\prod_{C \in \mathcal{C} \setminus \{C_2\}} \phi_C(x_C)}{\prod_{S \in \mathcal{S} \setminus \{S_0\}} \phi_S(x_S)} \cdot \phi^*_{C_2}(x_{C_2}) \cdot \frac{1}{\phi^*_{S_0}(x_{S_0})}.$$

For each x, consider the three possible cases:

Case (i): $\lambda_{S_0}(x_{S_0}) > 0$. Clearly, $f^*(x) = f(x)$.

Case (ii): $\phi_{S_0}(x_{S_0}) = 0$. Then $f(x) = 0$ by definition, while

$$\lambda_{S_0}(x_{S_0}) = 0 \quad \Rightarrow \quad \phi^*_{C_2}(x_{C_2}) = 0 \quad \Rightarrow \quad f^*(x) = 0.$$

Case (iii): $\phi_{S_0}(x_{S_0}) > 0$ but $\phi^*_{S_0}(x_{S_0}) = 0$. Then $f^*(x) = 0$ by definition, while

$$\phi^*_{S_0}(x_{S_0}) = 0 \quad \Rightarrow \max_{y_{C_1} : y_{S_0} = x_{S_0}} \phi_{C_1}(y_{C_1}) = 0 \quad \Rightarrow \quad \phi_{C_1}(x_{C_1}) = 0,$$

where the last step follows from ϕ_{C_1} being non-negative, so $f(x) = 0$.

This show $f^*(x) = f(x), \forall x$.

Verification of a Proof Step for Theorem 2

The proof arguments on slide 25 for Theorem 1 apply here without changes. We now verify the next step in the proof (the counterpart for sum-flows is on slide 26).

By induction, we have, just before the last flow from C^* into $\mathcal{T}'$,

$$f_{U^*}^{\max} = \frac{\phi_{C^*}\alpha_{U'}}{\phi_{S^*}}, \quad (\text{potential on } \mathcal{T}^*), \quad \text{where} \quad \alpha_{U'} = \frac{\prod_{C\in\mathcal{C}'}\phi_C}{\prod_{S\in\mathcal{S}'}\phi_S}, \quad (\text{potential on } \mathcal{T}').$$

After the flow, $\alpha_{U'}$ is replaced by

$$\alpha_{U'}^* = \alpha_{U'}\lambda_{S^*}, \quad \text{where} \quad \lambda_{S^*} = \frac{\max_{C^*\setminus S^*}\phi_{C^*}}{\phi_{S^*}}.$$

By the property of a junction tree,

- $S^* = C^* \cap U'$, so $C^* \setminus S^* = U^* \setminus U'$.

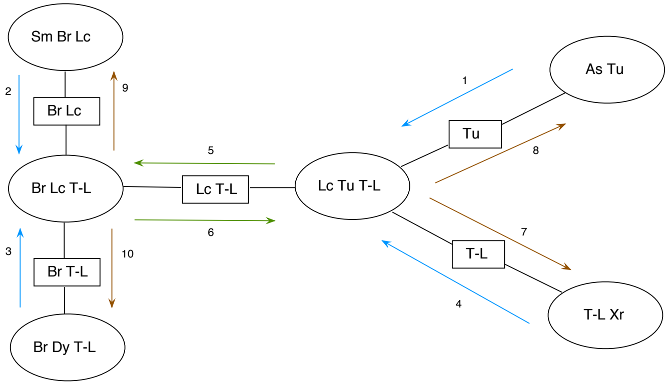
Therefore, for each $x_{U'}$,

$$f_{U'}^{\max}(x_{U'}) = \max_{U^*\setminus U'} f_{U^*}^{\max}(x_{U'}, x_{U^*\setminus U'}) = \max_{C^*\setminus S^*} f_{U^*}^{\max}(x_{U'}, x_{C^*\setminus S^*})$$
$$= \alpha_{U'}(x_{U'}) \cdot \frac{\max_{C^*\setminus S^*}\phi_{C^*}(x_{C^*\setminus S^*}, x_{S^*})}{\phi_{S^*}(x_{S^*})}.$$

If $\phi_{S^*}(x_{S^*}) > 0$, then $f_{U'}^{\max}(x_{U'}) = \alpha_{U'}^*(x_{U'})$ clearly. If $\phi_{S^*}(x_{S^*}) = 0$, then $f_{U'}^{\max}(x_{U'}) = 0$ by definition, while $\alpha_{U'}^*(x_{U'}) = 0$ because $\lambda_{S^*}(x_{S^*}) = 0$. This shows $f_{U'}^{\max}(x_{U'}) = \alpha_{U'}^*(x_{U'}), \forall x_{U'}$.

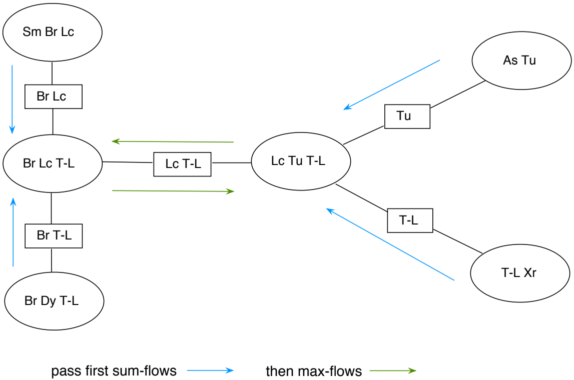
Since any possible, subsequent flows within $\mathcal{T}'$ do not change the potential on $\mathcal{T}'$ (which is the contraction of $\Phi_{\mathcal{T}'}$), the proof is complete.

Max-Flows for the Asia Example



- Suppose no evidence is entered initially. After flows $1-4$, the potential on the subtree with nodes (Br, Lc, T-L), (Lc, Tu, T-L) is $p^{\max}(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L})$. (The potential on the entire tree is always p.)
- suppose the evidence $\mathbf{e}: \{\text{Sm} = y, \text{Xr} = y, \text{As} = n\}$ is entered initially. Then after flows $1-4$, the potential on the subtree with nodes (Br, Lc, T-L), (Lc, Tu, T-L) is $p^{\max}(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L}, \&\,\mathbf{e})$. (The potential on the entire tree is always $p(\cdot\,\&\,\mathbf{e})$.)

A Mix of Sum/Max-Flows for the Asia Example



- suppose the evidence $\mathbf{e}: \{\text{Dy} = y, \text{Xr} = y, \text{As} = n\}$ is entered initially. Then after the sum-flows, the potential on the subtree with nodes $(\text{Br}, \text{Lc}, \text{T-L}), (\text{Lc}, \text{Tu}, \text{T-L})$ is $p(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L}, \& \,\mathbf{e})$. So after the max-flows, the potentials on the two cliques are

$$\max_{\text{Tu}} p(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L}, \& \,\mathbf{e}), \qquad \max_{\text{Br}} p(\text{Br}, \text{Lc}, \text{Tu}, \text{T-L}, \& \,\mathbf{e}),$$

respectively. From this, we can find the most probable configuration of the disease variables given the evidence, after *Sm being marginalized out*. (Note that the potential on the entire tree is always $p(\cdot \,\& \,\mathbf{e})$.)

Applications of the Algorithms

The algorithms are applicable to both undirected and directed graphs.

For a DAG, we can use them to find, for example,

- $P(\mathbf{e})$, $P(x_C \,|\, \mathbf{e})$ for any $C \in \mathcal{C}$;
- the most probable configuration x given $\mathbf{e}$, or the most probable configuration of a certain subset of variables given $\mathbf{e}$ (as in the previous Asia example);
- $P(X_A = x_A)$ for a subset $A \notin \mathcal{C}$:

  We treat x_A as the evidence $\mathbf{e}$, and then run the sum-flow algorithm to find $P(\mathbf{e})$.

Also,

- to sample from the posterior distribution $p(x \,|\, \mathbf{e})$:

  First, we run the sum-flow algorithm to find $p(x_C \,|\, \mathbf{e})$ for any C;
  next, according to this posterior distribution, we draw a sample $\hat{x}_C$;
  then, we include $\hat{x}_C$ in the evidence $\mathbf{e}$, and repeat the process for the variables whose values are yet to be assigned.

Besides the sum and max-flows, there are also other flows with interesting applications – see references [1], [2] given at the end.

Outline

Junction Trees of Cliques

Suppose $\mathcal{C}$ is the set of cliques of an undirected graph G. There are several results related to junction trees of cliques, for example:

Theorem: *There exists a junction tree of cliques for G if and only if G is decomposable.*

Theorem: *The followings are equivalent: (i) G is decomposable; (ii) G is chordal (or triangulated); and (iii) G admits a perfect numbering.*

- A *chord* of a cycle in G is a pair of non-adjacent vertices (α, β) of the cycle such that there is an edge between α and β in G.
- G is *chordal* if every one of its cycle of length ≥ 4 possesses a chord.
- A numbering of the vertices of G is called *perfect* if the neighbors of any vertex that have lower numbers induce a complete subgraph.

Building a junction tree involves:
(i) Triangulate and then find cliques of G (G^m if G is a DAG);
(ii) Find an ordering of the cliques that possesses the *running-intersection property*, which can be used to link the cliques into a junction tree.

For details and further study, see Chap. 4.3, 4.4 of Cowell et al. 2007.

Further Readings

The material of this lecture is based on

1. A. Philip Dawid. Applications of a general propagation algorithm for probabilistic expert systems, *Statistics and Computing*, No. 2, 25-36, 1992.
2. Robert G. Cowell et al. *Probabilistic Networks and Expert Systems*, Springer, 2007. Chap. 6.

For further study on building junction trees, see Chap. 4.3, 4.4 of [2].

For an introduction on the junction tree algorithm:

3. Finn V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996. Chap. 4.