Graphical Model Theory for Wireless Sensor Networks

William B. Davis

Lawrence Berkeley National Laboratory, MS 46A-1123B, One Cyclotron Road, Berkeley CA 94720.
WBDavis@lbl.gov

Abstract. Information processing in sensor networks, with many small processors, demands a theory of computation that allows the minimization of processing effort, and the distribution of this effort throughout the network. Graphical model theory provides a probabilistic theory of computation that explicitly addresses complexity and decentralization for optimizing network computation. The junction tree algorithm, for decentralized inference on graphical probability models, can be instantiated in a variety of applications useful for wireless sensor networks, including: sensor validation and fusion; data compression and channel coding; expert systems, with decentralized data structures, and efficient local queries; pattern classification, and machine learning. Graphical models for these applications are sketched, and a model of dynamic sensor validation and fusion is presented in more depth, to illustrate the junction tree algorithm.

1 A Model of Network Computation

Graphical model theory is a marriage of probability and graph theory that uses network graphs (with nodes and edges) to represent probability models. Probabilistic propositions are at least as expressive as logical propositions (containing them as extremal cases); the language of probability therefore includes the languages of all Turing computers, with probabilistic conditioning inclusive of logical implication. While Boolean propositions may only be true or false, probabilistic propositions may also assume any intermediate value. A probabilistic theory of analog computation may be *more* expressive than the Turing model.

Representing probability models as graphs has several advantages for wireless sensor networks. Inference, the essential probabilistic operation, is described as an algorithm directly on the probability graph, or on a graph derived from it. The probability graph contains the network hardware graph, so a graphical description of the algorithm facilitates its hardware implementation. Furthermore, the algorithm for exact inference on network graphs, the junction tree algorithm, has the following properties: it is minimum complexity; it is maximally decentralized (requiring only "local" information, in a well-defined sense); and it explicitly provides a communications protocol that specifies the information flows between pairs of nodes, and the order in which these information flows must occur. These properties are useful in network implementations.

Broadly, there are three steps to using a graphical model: (1) defining the model; (2) constructing the inference engine; and, (3) using the inference engine. The first step entails: specifying the random variables, as nodes in the graphical model, and assigning probability distributions to these nodes; and, specifying the dependencies between the variables, as edges in the graphical model. This step is described in section 2, with a static sensor fusion example.

The second and third steps, constructing and using an inference engine, are accomplished by the junction tree algorithm. The junction tree algorithm performs decentralized, efficient, exact Bayesian inference on any graphical model. Furthermore, the junction tree algorithm can be utilized as a subroutine for higher-order inference – for estimating probability distributions, and for determining structure in graphical models, when these are not known. The junction tree algorithm therefore provides a good introduction to graphical model theory, and is described in section 3 [3, 7-8, 10].

The claim that graphical models provide an expressive theory of network computation is given substantial support in section 4, which provides a variety of graphical models for wireless sensor networks. The suite of applications described is diverse and comprehensive, including: sensor validation and fusion; data compression and channel coding; distributed expert systems; pattern classification, and machine learning [4].

Section 5 develops the examples of graphical models for sensor fusion and validation in greater depth, with the objectives of illustrating the desirable properties of the junction tree algorithm, and describing decentralized network implementations.

Despite the computational advantages of the junction tree algorithm, exact inference can still be intractable for many models. Section 6 introduces faster algorithms for approximate inference, and extensions of graphical models and the junction tree algorithm to decision and game networks.

2 Graphical Models Represent How Joint Probability Distributions Factor

Constructing a graphical model G = ($\boldsymbol{X}$, $\boldsymbol{E}$) entails assigning nodes $\boldsymbol{X}$ to random variables, with a joint probability distribution function $\boldsymbol{p}(\boldsymbol{X})$, and including edges $\boldsymbol{E}$ to represent probabilistic dependencies between the nodes. If the edges are directed (and the graph has no cycles), the graphical model is known as a *Bayesian network*.[1] In this type of graphical model, the joint pdf can be factored

$$\boldsymbol{p}(\boldsymbol{X}) = \prod_{X \in \boldsymbol{X}} p(X \mid \boldsymbol{\pi}(X)) \tag{1}$$

into a product of conditional distributions, where each node is conditioned on its parent nodes $\boldsymbol{\pi}(X)$. Equivalently, the edges embody conditional independence assumptions on the local conditional probability distributions, namely that in Bayes nets, each node is conditionally independent[2] of all others, given its parents.

Consider the following sensor fusion example. The directed graph in figure 1 represents M sensors S_m observing the same object O.

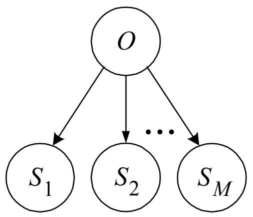


Fig. 2.1. Static sensor fusion directed graph

The sensors are conditionally independent given the object state, $S_m \perp S_n \mid O$. The joint pdf can therefore be factored according to eq. (1).

$$\boldsymbol{p}(O, S_1, S_2, \ldots, S_M) = p(O) \prod_{m=1}^{M} p(S_m \mid O)$$

A probability distribution may also be represented by an undirected graph, known as a *Markov network*. In this case, the joint pdf can be represented

$$\boldsymbol{p}(\boldsymbol{X}) = \frac{\prod_{C \in \boldsymbol{C}} \psi(C)}{\prod_{S \in \boldsymbol{S}} \phi(S)} \tag{2}$$

as a product of *clique potentials* ψ on the *cliques*[3] $\boldsymbol{C}$ of G, divided by the product of *separator potentials* ϕ on the *separators*[4] $\boldsymbol{S}$ of those cliques. In Markov nets, graph separation is equivalent to conditional independence, i.e., $X \perp Y \mid Z$ iff Z graph separates[5] X from Y.

[1] aka *causal network* or *belief network*.

[2] A random variable X is *conditionally independent* of Y given Z, $X \perp Y \mid Z$, iff $\boldsymbol{p}(X,Y \mid Z) = p(X \mid Z)\, p(Y \mid Z)$ whenever $p(z) > 0$, i.e., iff the joint pdf factors. An equivalent condition is $p(X \mid Y,Z) = p(X \mid Z)$ whenever $\boldsymbol{p}(y,z) > 0$.

[3] Two nodes are *connected* if there exists an edge between them. A set of nodes is *complete* if it is fully connected (where each node is connected to every other node). A *clique* is a maximal complete set of nodes – the largest

Continuing with the sensor fusion example, the undirected graph in figure 2 again represents M sensors S_m observing object O.

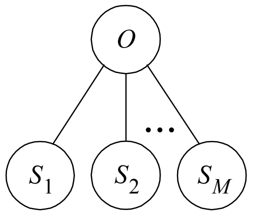


Fig. 2.2. Static sensor fusion undirected graph

The sensors are graph separated from one another by O, and are therefore conditionally independent given the object state, $S_m \perp S_n \mid O \,,\ \forall m \neq n$. The cliques in this graph are $\{S_1, O\}$, $\{S_2, O\}$,…,$\{S_M, O\}$, and the separator for these cliques is always $\{O\}$. The joint pdf can therefore be factored according to eq. (2),

$$p\left(O, S_1, S_2, ..., S_M\right) = \frac{\prod_{m=1}^{M} \psi(S_m, O)}{\prod_{m=1}^{M-1} \phi(O)} .$$

One example of potentials that satisfy this equation are marginal probabilities.

Missing edges in both directed and undirected graphical models represent conditional independence assumptions, and therefore describe how the joint probability distributions factor. The junction tree algorithm manipulates the joint pdf to perform exact inference. The algorithm utilizes the factorizations described in eqs. (1) and (2) to make these computations efficient, and can distribute the computational effort over nodes in the graph.

3 Exact Inference on Graphical Models: The Junction Tree Algorithm

Inference is a probabilistic operation wherein marginal and conditional probabilities are calculated from the joint pdf, possibly conditioning on the knowledge of the states of some observed nodes. Exact inference in Bayesian and Markov networks is accomplished via the Junction Tree Algorithm (JUNCTIONTREE). This can be described as a process of first constructing and then utilizing an inference engine, which makes these probabilistic calculations. The inference engine that JUNCTIONTREE constructs has several advantages for decentralized implementation in small, low-power, wireless sensor networks. JUNCTIONTREE exploits those conditional independencies that do exist to make the calculations as efficient and decentralized as possible, given the dependency structure of the sensor network. In addition to describing the calculations necessary at each node in the network, JUNCTIONTREE also specifies a message-passing protocol, which describes the timing and content of the information flow between nodes. JUNCTIONTREE is minimum complexity, maximally decentralized, and specifies the necessary message passing throughout the network. These properties will be demonstrated by example, after the algorithm is described. Because an inference engine can be used repeatedly once constructed (for example every time new sensor readings are obtained), and also because it is the inference engine itself, and its use, not its construction, that is efficient and decentralized, more detailed description and theoretical justification is provided for using than constructing the inference engine. The subroutines that comprise using the inference engine are also those whose implementation is discussed in the context of the sensor fusion and validation example in section 5.

complete set containing those nodes; strictly contained by no other complete set; i.e., there exist no additional nodes connected to every node in the set.

[4] A *separator* S of cliques C_1 and C_2 is those nodes they share in common, $S = C_1 \cap C_2$.

[5] *Z graph separates X* from Y iff all paths from X to Y intersect Z.

3.1 Overview of the Junction Tree Algorithm

JUNCTIONTREE performs exact inference in probability models, and operates on the graphical representation of a joint pdf. Following is an overview of this algorithm, which consists of five main subroutines. The MORALIZE, TRIANGULATE, and CONSTRUCTJT subroutines are graph-theoretic operations that construct the inference engine (the junction tree), while the INTRODUCEEVIDENCE and PROPAGATEPOTENTIALS subroutines use the junction tree to answer probabilistic queries.

```
JUNCTIONTREE
  Input graph $G$ with nodes $\boldsymbol{X}$ and edges $\boldsymbol{E}$; Distributions $\boldsymbol{p}(\boldsymbol{X})$; Evidence $X_E = x_E$
        MORALIZE (only if $G$ is directed)
          Input $G$; Output moral graph $G^M$
        TRIANGULATE
          Input moral graph $G^M$; Output triangulated graph $G^T$
        CONSTRUCTJT
          Input cliques $\boldsymbol{C}$ and separator cardinalities $|\boldsymbol{S}|$ of $G^T$;
          Output $JT$ a junction tree of cliques of $G^T$
        INTRODUCEEVIDENCE
          Initialize clique potentials $\psi(\boldsymbol{C})$, separator potentials $\phi(S)$ on $G^T$
          Input $X_E = x_E$; Output posterior clique potentials $\psi(\boldsymbol{C})$ on cliques of $G^T$
        PROPAGATEPOTENTIALS
          Start at root clique
                COLLECTEVIDENCE
                 For each child of clique,
                  UPDATEPOTENTIALS
                   Input clique, COLLECTEVIDENCE(child)
                   Output updated clique potentials $\psi^*(X_C)$
                DISTRIBUTEEVIDENCE
                 For each parent of clique,
                  UPDATEPOTENTIALS
                   Input clique, DISTRIBUTEEVIDENCE(parent)
                   Output updated clique potentials $\psi^{**}(X_C)$
  Output $p(X_F \mid X_E)$
```

This algorithm constructs an inference engine, in the form of a junction tree of clique potentials, then performs inference by achieving local consistency between adjoining clique potentials in the junction tree. Because of properties of the junction tree, when this algorithm terminates, the clique potentials equal pdfs conditioned on the evidence, from which inferential queries can be answered locally.

PROPAGATEPOTENTIALS adjusts the clique potentials to achieve this mutual consistency. PROPAGATEPOTENTIALS also describes the message-passing necessary between cliques in the junction tree. The calls to COLLECTEVIDENCE and DISTRIBUTEEVIDENCE are recursive – these subroutines repeatedly call themselves so long as a clique in the junction tree has children or a parent. The recursive calls ensure that all cliques send a message to parents only after they have received messages from all children. Messages are passed once "inward" from the leaves to the root, then once "outward" from the root to the leaves (there exist shorter sequences of message passing that yield the same result).

The MORALIZE, TRIANGULATE, and CONSTRUCTJT subroutines are graph-theoretic operations that guarantee that local consistency will result in global consistency, i.e., that PROPAGATEPOTENTIALS will result in all the potentials proportional to the local marginal probabilities, as desired. These subroutines are described directly in graph-theoretic terms – adjacency matrices afford a more efficient representation of graphs in implementation.

Constructing the Inference Engine: Moralization

MORALIZE converts a directed graphical representation to an undirected graphical representation (that encodes no more conditional independence assumptions, and possibly fewer). This step is therefore only necessary if G is directed.

```
MORALIZE
 Input directed graph $G$ with nodes $\boldsymbol{X}$ and directed edges $\boldsymbol{E}$
 For Each node $X \in \boldsymbol{X}$
  If $(P, X) \in \boldsymbol{E}$
   Then $\boldsymbol{\pi}(X) \leftarrow P$
  For Each node $P \in \boldsymbol{\pi}(X)$
   For Each node $Q \in \boldsymbol{\pi}(X)$ s.t. $Q \neq P$
    If $(P,Q) \notin \boldsymbol{E}$ And $(Q,P) \notin \boldsymbol{E}$
     Then $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup (P,Q)$
 For Each edge $(X,Y) \in \boldsymbol{E}$
  $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup (Y,X)$
 Output moral graph $G^M$
```

To MORALIZE a directed graph, fully connect all parent sets, and convert directed to undirected edges. Adding edges makes the family of probability distributions represented by the graph larger (with fewer conditional independence assumptions). Inference on the moral graph G^M includes inference on the original graph G.

Constructing the Inference Engine: Triangulation

A graph is *triangulated* iff all cycles of length four or more have a chord.[6] There exist a number of algorithms for triangulating a graph. All involve adding edges to the moral graph, when necessary to eliminate chordless cycles. There often exist, however, several triangulations of the same moral graph. Furthermore, the resulting triangulation is important in determining the complexity of PROPAGATEPOTENTIALS, which is exponential in the cardinality of the largest clique in the junction tree. Finding an efficient graph triangulation algorithm is still an open research question.

```
TRIANGULATE
 Input moral graph $G^M = (\boldsymbol{X}, \boldsymbol{E})$
 $\boldsymbol{E}^T \leftarrow \boldsymbol{E}$
 For Each node $X \in \boldsymbol{X}$
  $\mathrm{Ne}(X) \leftarrow \varnothing$
  For Each node $Y \in \boldsymbol{X}$
   If $(X,Y) \in \boldsymbol{E}$ Then $\mathrm{Ne}(X) \leftarrow \mathrm{Ne}(X) \cup Y$
  $\Delta E(X) \leftarrow \left\{ \frac{|\mathrm{Ne}(X)|\left[|\mathrm{Ne}(X)|-1\right]}{2} - \sum_{(X_1,X_2)\in\boldsymbol{E}} \mathrm{I}\left[X_1 \in \mathrm{Ne}(X) \;\&\; X_2 \in \mathrm{Ne}(X)\right] \right\}$
 $\boldsymbol{X}^T \leftarrow \boldsymbol{X}$
 $i \leftarrow 1$
 While $\boldsymbol{X}$ is not a clique
  $A \leftarrow \arg\min \Delta E(X)$
  For Each node $X \in \mathrm{Ne}(A)$
   For Each node $Y \neq X \in \mathrm{Ne}(A)$
    If $(X,Y) \notin \boldsymbol{E}$ Then $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup (X,Y)$ And $\boldsymbol{E}^T \leftarrow \boldsymbol{E}^T \cup (X,Y)$
  $C_i \leftarrow A \cup \mathrm{Ne}(A)$
  For Each node $X \in \mathrm{Ne}(A)$
   $\boldsymbol{E} \leftarrow \boldsymbol{E} \setminus (A,X)$
  $\boldsymbol{X} \leftarrow \boldsymbol{X} \setminus A$
  $i \leftarrow i+1$
 Output triangulated graph $G^T = (\boldsymbol{X}, \boldsymbol{E}^T)$ (and its cliques $\boldsymbol{C}$)
```

[6] In an UG, two nodes are *connected* if there exists an edge between them. A *cycle* in an UG is a sequence of nodes each connected to the next, starting and finishing at the same node. A *chord* of a cycle is a connected pair of nodes not consecutive in the cycle.

The triangulation algorithm provided realizes good performance in applications, adding fewer edges than most alternatives. It is a simple greedy algorithm, that successively selects nodes in order of the smallest number of necessary edges to fully connect all neighbors of that node, adds the edges between neighbors, then absorbs the node and its edges (the edges absorbed are not the edges added). Let the absorbing node now represent a Cartesian product of itself with the absorbed node (nodes can represent vector as well as scalar random variables). Because its neighbors are fully connected, the absorbed node can not be factored from its clique. Factorization is represented by missing edges, and since none is possible, no missing edges from this node need be represented by the graph, and the node may be absorbed onto one of its neighbors without changing the conditional independence relations represented by the graph. When all nodes have been absorbed into one clique, the edges added during this process are added to the moral graph, resulting in a triangulated graph.

Constructing the Inference Engine: the Junction Tree

The junction tree is a tree of cliques of the triangulated graph G^T. Furthermore, a junction tree is a maximal spanning tree, where the edge weights are the cardinalities of the separator sets. With some initial machinery to determine the cliques and separators of G^T, any of a number of algorithms for finding a maximal spanning tree can be applied to construct a junction tree [2].

```
CONSTRUCTJT
  Input triangulated graph $G^T = \left(\boldsymbol{X}, \boldsymbol{E}^T\right)$
  $\boldsymbol{E}^{JT} \leftarrow \varnothing$
  For i= 1 to $|\boldsymbol{C}|$-1
    For $j = i + 1$ to $|\boldsymbol{C}|$
      $S_{ij} \leftarrow C_i \cap C_j$
  For k= 1 to $|\boldsymbol{C}|$-1
    $(x, y) \leftarrow \arg\max_{i,j} \left\{ |S_{ij}| \right\}$
    $\boldsymbol{E}^{JT} \leftarrow \boldsymbol{E}^{JT} \cup \left(C_x, C_y\right)$
  Output junction tree $JT = \left(\boldsymbol{C}, \boldsymbol{E}^{JT}\right)$
```

Using the Inference Engine: Introduction of Evidence

In sensor fusion and validation applications, evidence is sensor readings. In this case INTRODUCEEVIDENCE constrains the estimate of the state of the object being sensed to be consistent with the sensor readings.

```
INTRODUCEEVIDENCE
  Input $X_E = x_E$
  For each clique $C \in \boldsymbol{C}$
    $\psi(\boldsymbol{C}) \leftarrow 1$
  For each separator $S \in \boldsymbol{S}$
    $\phi(S) \leftarrow 1$
  For each $X \in X_E$
    $p\left(X_E \neq x_E\right) \leftarrow 0$
  While $\boldsymbol{X} \neq \varnothing$
    For each $X \in \boldsymbol{X}$
      For some $C \subset X$
        $\psi(\boldsymbol{C}) \leftarrow \psi(\boldsymbol{C}) p\left(X \mid C\right)$
      $\boldsymbol{X} \leftarrow \boldsymbol{X} \setminus X$
  Output initial clique potentials $\psi(\boldsymbol{C})$ on cliques of $G^T$
```

This subroutine first initializes clique and separator potentials to one, and sets all conditional probabilities not consistent with the evidence to zero. Then it multiplies all conditional probabilities of each node into the potential of some clique containing that node.

Using the Inference Engine: Propagation of Potentials

To illustrate how PROPAGATEPOTENTIALS proceeds, consider the simplest of junction trees, with two cliques and one separator (Figure 3.1.1). Designate C_1 as the root, and C_2 as the leaf. With each clique and separator is associated a potential.

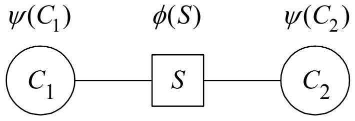


Fig. 3.1.1. Graph of the simplest junction tree

PROPAGATEPOTENTIALS proceeds recursively, with an inward pass from the leaves to the root, then an outward pass from the root to the leaves. In the above two-clique junction tree (with C_1 designated as the root), the JTA updates $\psi^*(C_1)$ based on $\psi(C_2)$, with an inward pass, then $\psi^*(C_2)$ on (the updated) $\psi^*(C_1)$ with an outward pass of messages through the junction tree.

<u>COLLECTEVIDENCE</u>

$$\phi^*(S) = \sum_{C_2 \backslash S} \psi(C_2) \qquad \psi^*(C_1) = \frac{\phi^*(S)}{\phi(S)}\psi(C_1)$$

<u>DISTRIBUTEEVIDENCE</u>

$$\phi^{**}(S) = \sum_{C_1 \backslash S} \psi^*(C_1) \qquad \psi^*(C_2) = \frac{\phi^{**}(S)}{\phi^*(S)}\psi(C_2)$$

PROPAGATEPOTENTIALS has several properties worth elucidating. Both the inward and outward passes leave the joint probability distribution, in the form of eq. (2), unchanged. After both passes the clique potentials are consistent with respect to their shared marginals. In particular, clique potentials (and thus separator potentials as well) are equal to marginal probabilities. Each of these properties is explained below.

PROPAGATEPOTENTIALS leaves the joint distribution in the graphical model unchanged. Recalling from eq. (2) that the joint distribution in an undirected graphical model is represented as the product of clique potentials divided by the product of separator potentials, after the inward pass,

$$\boldsymbol{p}(\boldsymbol{X}) = \frac{\psi^*(C_1)\psi(C_2)}{\phi^*(S)} = \frac{\phi^*(S)}{\phi(S)}\psi(C_1)\frac{\psi(C_2)}{\phi^*(S)} = \frac{\psi(C_1)\psi(C_2)}{\phi(S)},$$

and after the outward pass,

$$\boldsymbol{p}(\boldsymbol{X}) = \frac{\psi^*(C_1)\psi^*(C_2)}{\phi^{**}(S)} = \frac{\phi^*(S)}{\phi(S)}\psi(C_1)\frac{\phi^{**}(S)}{\phi^*(S)}\psi(C_2)\frac{1}{\phi^*(S)} = \frac{\psi(C_1)\psi(C_2)}{\phi(S)},$$

as claimed.

After the updates, $\psi^*(C_1)$ and $\psi^*(C_2)$ are consistent with respect to their intersection S,

$$\sum_{C_1 \backslash S} \psi^*(C_1) = \phi^{**}(S) = \frac{\phi^{**}(S)}{\phi^*(S)}\phi^*(S) = \frac{\phi^{**}(S)}{\phi^*(S)}\sum_{C_2 \backslash S}\psi(C_2) = \sum_{C_2 \backslash S}\frac{\phi^{**}(S)}{\phi^*(S)}\psi(C_2) = \sum_{C_2 \backslash S}\psi^*(C_2).$$

That is, summing out all nodes not in S results in the same marginal potentials for S regardless of whether the sum is performed on $\psi^*(C_1)$ or on $\psi^*(C_2)$. Thus local "consistency" or "agreement" between clique potentials is achieved by PROPAGATEPOTENTIALS.

Finally, it remains to be demonstrated that after running PROPAGATEPOTENTIALS, the potentials necessarily have the desired probabilistic interpretation. That this is at least possible is guaranteed by the conditional independence assumptions encoded in the graphical representation – $C_1 \perp C_2 \mid S$,

$$\boldsymbol{p}(\boldsymbol{X}) = \boldsymbol{p}(C_1, C_2, S) = \boldsymbol{p}(C_1, C_2 \mid S)\, p(S) = p(C_1 \mid S)\, p(C_2 \mid S)\, p(S)$$
$$= \frac{\boldsymbol{p}(C_1, S)}{p(S)} \frac{\boldsymbol{p}(C_2, S)}{p(S)} p(S) = \frac{\boldsymbol{p}(C_1)\boldsymbol{p}(C_2)}{p(S)}.$$

Because of the conditional independence assumptions encoded in the graphical model, the joint probability distribution can be represented as a product of clique marginals, divided by the (product of) separator marginals.

Furthermore, the potentials necessarily have this probabilistic interpretation after running PROPAGATEPOTENTIALS subroutine. After the inward pass,

$$\psi^*(C_1) = \frac{\psi(C_1)}{\phi(S)}\phi^*(S) = \frac{\psi(C_1)}{\phi(S)} \sum_{C_2 \setminus S} \psi(C_2) = \sum_{C_2 \setminus S} \frac{\psi(C_1)\psi(C_2)}{\phi(S)}$$
$$= \sum_{C_2 \setminus S} \boldsymbol{p}(C_1, C_2, S) = \boldsymbol{p}(C_1, S) = \boldsymbol{p}(C_1).$$

the first updated clique potential is the same as the marginal probability of that clique. After the outward pass,

$$\psi^*(C_2) = \frac{\phi^{**}(S)}{\phi^*(S)}\psi(C_2) = \frac{\sum_{C_1 \setminus S} \psi^*(C_1)}{\phi^*(S)}\psi(C_2) = \sum_{C_1 \setminus S} \frac{\psi^*(C_1)}{\phi^*(S)}\psi(C_2)$$
$$= \sum_{C_1 \setminus S} \boldsymbol{p}(C_1, C_2, S) = \boldsymbol{p}(C_2, S) = \boldsymbol{p}(C_2).$$

and the second updated clique potential is the same as the marginal probability of that clique. With these two potentials equal to their marginal probabilities, the (twice updated) separator potential must also be equal to its marginal probability, because

$$p(S) = \frac{\boldsymbol{p}(C_1)\boldsymbol{p}(C_2)}{\boldsymbol{p}(\boldsymbol{X})} = \frac{\psi^*(C_1)\psi^*(C_2)}{\boldsymbol{p}(\boldsymbol{X})} = \phi^{**}(S).$$

The above algebra only describes PROPAGATEPOTENTIALS on the simplest of junction trees, with two cliques, as depicted in Figure 2. To generalize this algorithm to more complex junction trees, care must be taken so that local consistency between a clique and one of its neighbors is not defeated by subsequent updates between that clique and its other neighbors. If a clique sends a message (distributes evidence) only after it has received messages (collected evidence) from all of its neighbors, then local consistency will be maintained. The recursive calls to COLLECTEVIDENCE and DISTRIBUTEEVIDENCE in PROPAGATEPOTENTIALS guarantee that this message-passing protocol is satisfied.

To demonstrate that PROPAGATEPOTENTIALS results in the desired clique-marginals for larger junction trees, first note that we can now write COLLECTEVIDENCE and DISTRIBUTEEVIDENCE in terms of clique and separator marginal probabilities as follows:

COLLECTEVIDENCE

$$\phi^*(S) = \sum_{C_2 \setminus S} \psi(C_2) \qquad\qquad p(C_1) = \frac{\phi^*(S)}{\phi(S)}\psi(C_1)$$

DISTRIBUTEEVIDENCE

$$p(S) = \sum_{C_1 \setminus S} p(C_1) \qquad\qquad p(C_2) = \frac{p(S)}{\phi^*(S)}\psi(C_2)$$

The case $|\boldsymbol{C}| = 2$ has already been demonstrated. To complete the induction, assume that PROPAGATEPOTENTIALS on a junction tree with $|\boldsymbol{C}| = N$ cliques results in potentials equal to marginal probabilities, then demonstrate that this implies the same for a junction tree with $|\boldsymbol{C}| = N + 1$ cliques. Because any tree can be built up by successively adding leaves, we can assume that the added clique is a leaf, and still span the set of all trees. The induction hypothesis then allows us to describe PROPAGATEPOTENTIALS in a manner similar to the $|\boldsymbol{C}| = 2$ case. Now the leaf is clique C, while the "root" is the junction tree with $|\boldsymbol{C}| = N$ cliques, and with nodes $\boldsymbol{X}$. The induction hypothesis provides $\boldsymbol{\psi}(\boldsymbol{X}) = \boldsymbol{p}(\boldsymbol{X})$.

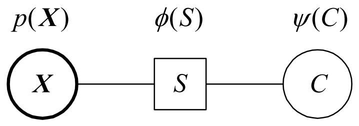


Fig. 3.1.2. Graph of a general junction tree

To complete the proof we need only describe PROPAGATEPOTENTIALS for clique C, and demonstrate that it results in $\phi(S) = p(S)$ and $\psi(C) = p(C)$ as desired. Because it is already assumed that PROPAGATEPOTENTIALS returns the correct marginals on $\boldsymbol{X}$,

$$\phi^{**}(S) = \sum_{\boldsymbol{X}\setminus S} \psi^*(\boldsymbol{X}) = \sum_{\boldsymbol{X}\setminus S} p(\boldsymbol{X}) = p(S).$$

Because DISTRIBUTEEVIDENCE leaves the joint probability distribution unchanged,

$$p(\boldsymbol{X}, C) = \frac{\psi(\boldsymbol{X})\psi(C)}{\phi(S)} = \frac{\psi^*(\boldsymbol{X})\psi^*(C)}{\phi^{**}(S)} = \frac{p(\boldsymbol{X})\psi^*(C)}{p(S)}.$$

Therefore,

$$\psi^*(C) = \frac{p(\boldsymbol{X}, C)p(S)}{p(\boldsymbol{X})} = p(C \mid \boldsymbol{X})p(S) = p(C \mid S)p(S) = p(C, S) = p(C),$$

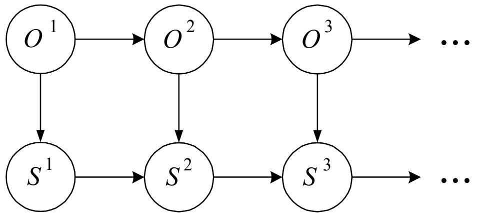
as desired. Each time INTRODUCEEVIDENCE and PROPAGATEPOTENTIALS are run, the result is a set of clique and separator potentials equal to marginal probabilities, which form a distributed database useful for answering probabilistic queries.

4 Applications to Wireless Sensor Networks

JUNCTIONTREE has many potential applications in wireless sensor networks. In the following subsections are graphical models for: dynamic sensor validation and fusion; data compression and channel coding; pattern classification and machine learning. This overview is necessarily brief, providing: the graphs for the different applications; the factored probability distribution functions they represent; examples of inferential queries of interest that JUNCTIONTREE can perform, which are also applications of Bayes' theorem; and, references for readers interested more specifics of a particular application. All graphs are directed; therefore, their pdfs factor according to equation (1). The graphs presented are the simple building blocks from which more complex models can be developed.

4.1 Graphical Models for Sensor Validation and Fusion

Sensor validation and fusion are among the more fundamental functions a wireless sensor network should perform. Graphical models are especially efficient for dynamic models.



$$\begin{tikzcd}
O^1 \arrow[r] \arrow[d] & O^2 \arrow[r] \arrow[d] & O^3 \arrow[r] \arrow[d] & \cdots \\
S^1 \arrow[r] & S^2 \arrow[r] & S^3 \arrow[r] & \cdots
\end{tikzcd}$$

Fig. 4.1.1. One sensor, many time periods

Note how dynamic Bayesian networks encode the Markov property, that each node is conditionally independent of all earlier nodes given its immediate predecessor. This is evident in how the joint probability distribution factors, which is given by eq. (1),

$$p(\mathbf{O},\mathbf{S}) = p(O^1)\,p(S^1 \mid O^1)\prod_{t=2}^{T}\left[p(S^t \mid O^t, S^{t-1})\,p(O^t \mid O^{t-1})\right].$$

The conditional object distribution can be found by an application of Bayes' theorem,

$$p(\mathbf{O} \mid \mathbf{S}) = \frac{p(O^1)\,p(S^1 \mid O^1)\prod_{t=2}^{T}\left[p(S^t \mid O^t, S^{t-1})\,p(O^t \mid O^{t-1})\right]}{\sum_{O} p(O^1)\,p(S^1 \mid O^1)\prod_{t=2}^{T}\left[p(S^t \mid O^t, S^{t-1})\,p(O^t \mid O^{t-1})\right]},$$

and can also be calculated with JUNCTIONTREE.

The graphs for the models of dynamic sensor fusion and validation are similarly layered.

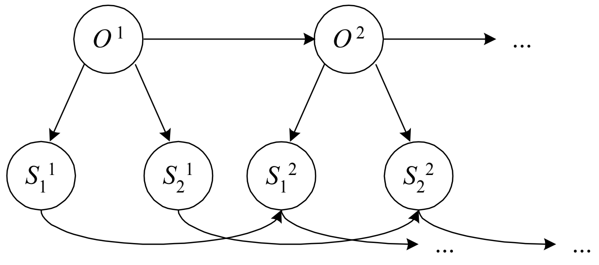


Fig. 4.1.2. Dynamic sensor fusion

The joint probability distribution is again given by eq. (1),

$$p(\mathbf{O},\mathbf{S}_1,\mathbf{S}_2) = p(O^1)\,p(S_1^1 \mid O^1)\,p(S_2^1 \mid O^1)\prod_{t=2}^{T}\left[p(S_1^t \mid O^t, S_1^{t-1})\,p(S_2^t \mid O^t, S_2^{t-1})\,p(O^t \mid O^{t-1})\right].$$

The conditional object distribution can be found by a more complex application of Bayes' theorem,

$$p(\mathbf{O} \mid \mathbf{S}_1,\mathbf{S}_2) = \frac{p(O^1)\,p(S_1^1 \mid O^1)\,p(S_2^1 \mid O^1)\prod_{t=2}^{T}\left[p(S_1^t \mid O^t, S_1^{t-1})\,p(S_2^t \mid O^t, S_2^{t-1})\,p(O^t \mid O^{t-1})\right]}{\sum_{O} p(O^1)\,p(S_1^1 \mid O^1)\,p(S_2^1 \mid O^1)\prod_{t=2}^{T}\left[p(S_1^t \mid O^t, S_1^{t-1})\,p(S_2^t \mid O^t, S_2^{t-1})\,p(O^t \mid O^{t-1})\right]}.$$

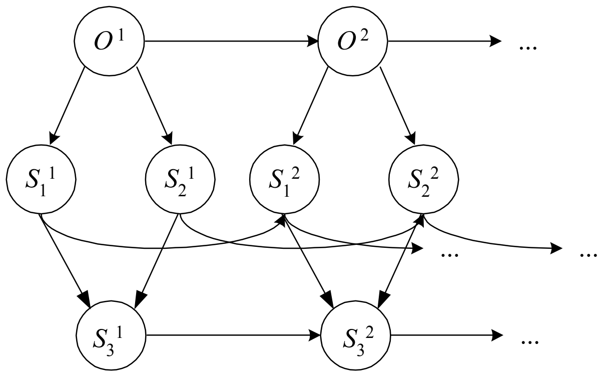


Fig. 4.1.3. Dynamic sensor fusion with validation

Again the joint probability distribution is given by eq. (1),

$$p(\mathbf{O},\mathbf{S}) = p(O^1)\,p(S_1^{\,1} \mid O^1)\,p(S_2^{\,1} \mid O^1)\,p(S_3^{\,1} \mid S_1^{\,1}, S_2^{\,1}) *$$
$$* \prod_{t=2}^{T}\left[p(O^t \mid O^{t-1})\,p(S_1^{\,t} \mid O^t, S_1^{\,t-1})\,p(S_2^{\,t} \mid O^t, S_2^{\,t-1})\,p(S_3^{\,t} \mid S_1^{\,t}, S_2^{\,t}, S_3^{\,t-1})\right].$$

Inference on this graphical model now includes a validation equation:

$$p(\mathbf{O} \mid \mathbf{S}) = \frac{p(\mathbf{O},\mathbf{S})}{\sum_{O} p(\mathbf{O},\mathbf{S})}; \qquad p(\mathbf{S}_1,\mathbf{S}_2 \mid \mathbf{S}_3) = \frac{\sum_{O} p(\mathbf{O},\mathbf{S})}{\sum_{O}\sum_{\mathbf{S}_1,\mathbf{S}_2} p(\mathbf{O},\mathbf{S})}.$$

These graphical models are developed in greater detail in section 5.

4.2 Graphical Models for Data Compression and Channel Coding

Digital communication can be partitioned into two stages: *data compression*, and *channel coding*.

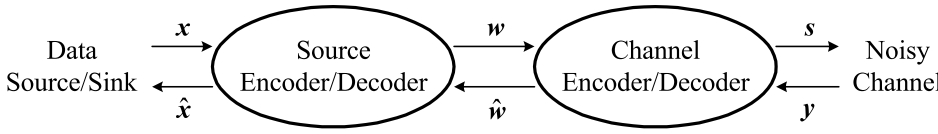


Fig. 4.2.1. Digital communications supergraph

A source coder accepts a pattern $\boldsymbol{x}$, and compresses the source at a rate $R \in (0, 1)$. Noiseless source coding, or lossless data compression, encodes the source pattern s.t. the expected codeword length $|\boldsymbol{w}|$ is as short as possible. A channel coder maps a codeword $\boldsymbol{w}$ to a signal $\boldsymbol{s}$, adding redundancy to protect against noise. After $\boldsymbol{s}$ is transmitted through a noisy channel, $\boldsymbol{y}$ is the signal received, and the channel coder reconstructs an estimate of the original codeword, then the source coder reconstructs an estimate of the decompressed source.

Graphical models are useful for representing algorithms for data compression and channel coding.

(Insert data compression directed graph)

Fig. 4.2.2. Data compression directed graph

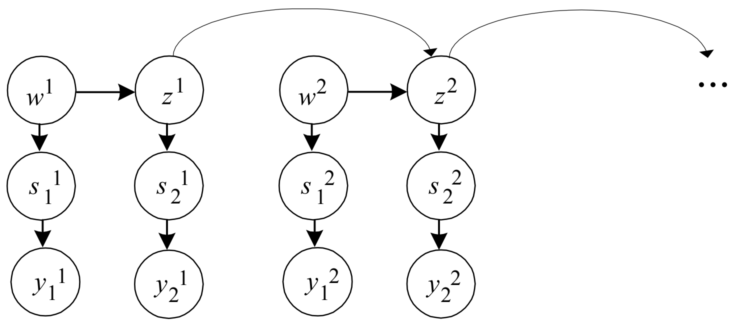


Fig. 4.2.3. Recursive convolutional coding directed graph

Figure 4.2.3. is a Bayesian network that describes an efficient channel code. To fully describe the probability model, the conditional distributions are specified. For details, see [4]. The JTA performs channel decoding, inferring $p\,(\boldsymbol{w} \mid \boldsymbol{y})$, and then

$$\hat{\boldsymbol{w}}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{y}) \qquad \text{or} \qquad \hat{\boldsymbol{w}}_{\text{B}} = \mathrm{E}_{\boldsymbol{w}|\boldsymbol{y}}[\boldsymbol{w}] = \int \boldsymbol{w}p(\boldsymbol{w} \mid \boldsymbol{y})\,d\boldsymbol{w}.$$

4.3 Graphical Models as Distributed Expert Systems

An expert system can be defined as a database with an inference engine [3]. We have already described the inference engine of graphical models. For the sensor fusion example, the data are the prior sensor and object distributions, and the sensor readings. As a result of INTRODUCEEVIDENCE, the data are represented as clique potentials in the junction tree. As a result of PROPOGATEPOTENTIALS, the set of all separator and clique potentials are a database in the form of marginal probabilities, which are useful in this form for inferential queries. Multiple simultaneous queries local to particular sensors can be answered in parallel.

4.4 Graphical Models for Pattern Classification and Machine Learning

One of the most active areas of research in graphical model theory is for pattern classification and machine learning [7-9]. For supervised pattern classification, both the pattern vector $\boldsymbol{x}$ and the class label c are observed as evidence. Generative classification involves fitting a classifier to data, then generating a class label prediction for new patterns.

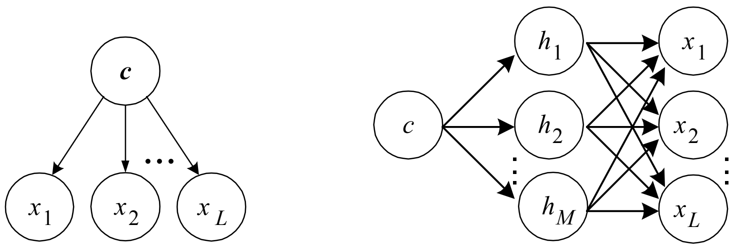


Fig. 4.4.1. Pattern classification and pattern classification with latent variables directed graphs

Note the similarity with figure 2.1. Sensor fusion *is* pattern classification, and this discipline can be applied to the problem of sensor fusion. For example, latent variable models for sensor fusion can be developed.

The joint pdf for the graphical models in figure 4.4.1 can again be factored according to eq. (1),

$$p(c,\boldsymbol{x}) = p(c)\prod_{l=1}^{L} p(x_l \mid c) \qquad \text{and} \qquad p(c,\mathbf{h},\mathbf{x}) = p(c)\prod_{m=1}^{M}\left[p(h_m \mid c)\prod_{l=1}^{L} p(x_l \mid \mathbf{h})\right].$$

The conditional class distributions are found by Bayes' theorem,

$$p(c \mid \boldsymbol{x}) = \frac{p(c)\prod_{l=1}^{L} p(x_l \mid c)}{\sum_c p(c)\prod_{l=1}^{L} p(x_l \mid c)} \qquad \text{and} \qquad p(c \mid \mathbf{x}) = \frac{\sum_{\mathbf{h}} p(c,\mathbf{h},\mathbf{x})}{\sum_c\sum_{\mathbf{h}} p(c,\mathbf{h},\mathbf{x})}.$$

In unsupervised learning, only the pattern vector $\boldsymbol{x}$ is observed, and is presumed generated by a set of latent variables, which may be layered,

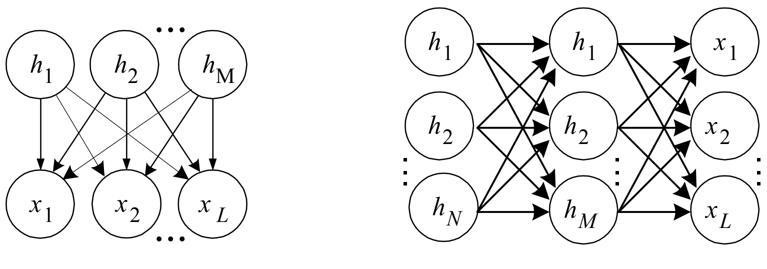


Fig. 4.4.2. Unsupervised learning and unsupervised learning with two hidden layers directed graphs

The pdf for these models again factor according to eq. (1),

$$p(\mathbf{h},\mathbf{x}) = \prod_{m=1}^{M}\left[p(h_m)\prod_{l=1}^{L}p(x_l \mid \mathbf{h})\right], \quad p(\mathbf{h}_N,\mathbf{h}_M,\mathbf{x}) = \prod_{n=1}^{N}\left\{p(h_n)\prod_{m=1}^{M}\left[p(h_m \mid \mathbf{h}_N)\prod_{l=1}^{L}p(x_l \mid \mathbf{h}_M)\right]\right\},$$

and the unconditional data distributions are,

$$p(\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{h},\mathbf{x}) \qquad \text{and} \qquad p(\mathbf{x}) = \sum_{\mathbf{h}_N}\sum_{\mathbf{h}_M} p(\mathbf{h}_N,\mathbf{h}_M,\mathbf{x}).$$

5 Decentralized, Efficient, Bayesian Sensor Fusion

This section uses the graphical models for sensor validation and fusion presented in section 4.1 as examples to illustrate the application of the junction tree algorithm. For decentralization of PROPOGATEPOTENTIALS, it is assumed that with each sensor is associated a processor.

Figure 2.2 is the moral graph of figure 2.1, the model of static sensor fusion. MORALIZE adds no edges. Figure 5.1.1 shows the moral graphs of figures 4.1.1 and 4.1.2, the output of MORALIZE. Both graphs are already triangulated.

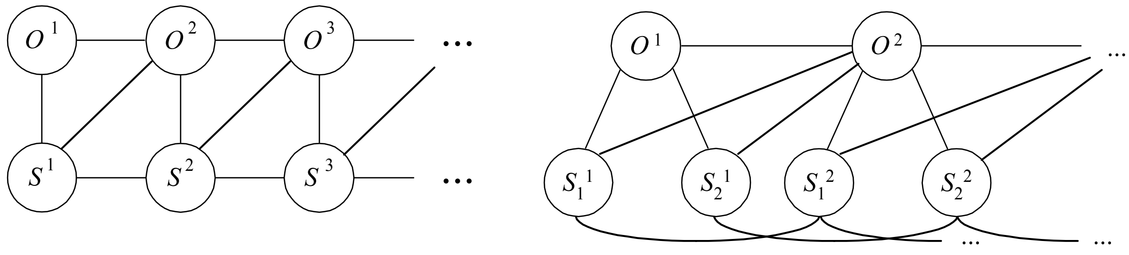


Fig. 5.1.1. Dynamic sensor and sensor fusion moral graphs

TRIANGULATE may add more edges than necessary. Unnecessary edges are not included in figures 5.1.1 and 5.1.2. It is not until so complex an example as that of dynamic sensor validation and fusion that it becomes necessary for edges to be added for the moral graph to be triangulated. Following are the moral and triangulate graphs of figure 4.1.3, for dynamic sensor validation and fusion. Without the added edges, there exist chordless 4-cycles in the moral graph.

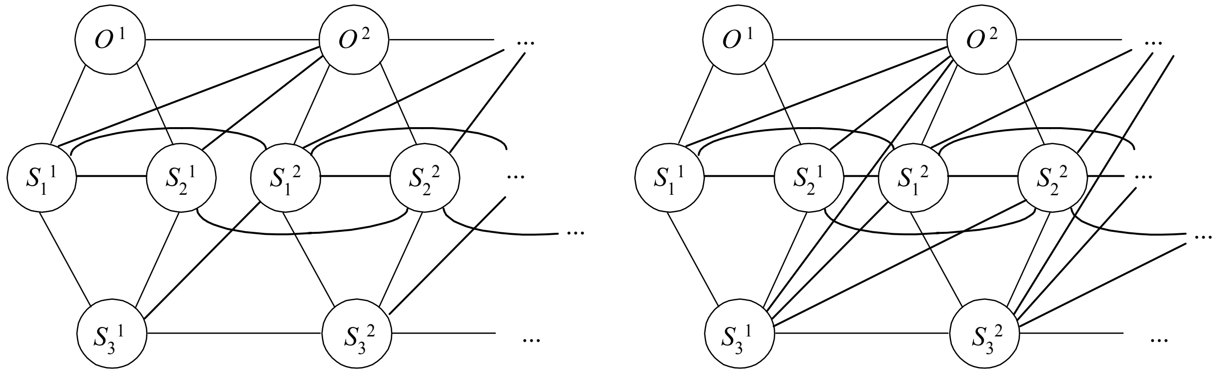


Fig. 5.1.2. Dynamic sensor validation and fusion moral and triangulated graphs

A final step is necessary to complete the inference engine, CONSTRUCTJT. This subroutine requires the separator cardinalities, which in turn requires the cliques of the triangulated graphs. These are the same as the elimination cliques provided by the TRIANGULATE subroutine, and are listed in table 5.1 for the models of dynamic sensor validation and fusion.

Table 5.1. Cliques and separator matrices of triangulated graphs in figures 5.1.1 and 5.1.2

Cliques	Separator matrices				
$\{O^1, S^1, O^2\}$	S_{ji}	C_1	C_2		
$\{S^1, O^2, S^2\}$	$= C_2$	$\{S^1, O^2\}$			
$\{O^2, S^2, O^3\}$	$= C_3$	$\{O^2\}$	$\{O^2, S^2\}$		
$\{S^2, O^2, S^3\}$	$= C_4$		$\{S^2\}$		
$\{O^1, S_1^{\ 1}, O^2\}$	S_{ji}	C_1	C_2	C_3	
$\{S_1^{\ 1}, S_1^{\ 2}, O^2\}$	$= C_2$	$\{S_1^{\ 1}, O^2\}$			
$\{S_2^{\ 1}, S_2^{\ 2}, O^2\}$	$= C_3$	$\{O^2\}$	$\{O^2\}$		
$\{O^2, S_1^{\ 2}, O^3\}$	$= C_4$	$\{O^2\}$	$\{O^2, S_1^{\ 2}\}$	$\{O^2\}$	
$\{S_1^{\ 2}, S_1^{\ 3}, O^3\}$	$= C_5$		$\{S_1^{\ 2}\}$		
$\{O^1, S_1^{\ 1}, S_2^{\ 1}, O^2\}$	S_{ji}	C_1	C_2	C_3	C_4
$\{S_1^{\ 1}, S_2^{\ 1}, S_3^{\ 1}, O^2, S_1^{\ 2}\}$	$= C_2$	$\{S_1^{\ 1}, S_2^{\ 1}, O^2\}$			
$\{S_2^{\ 1}, S_3^{\ 1}, S_1^{\ 2}, S_2^{\ 2}\}$	$= C_3$	$\{S_2^{\ 1}\}$	$\{S_2^{\ 1}, S_3^{\ 1}, S_1^{\ 2}\}$		
$\{S_3^{\ 1}, O^2, S_1^{\ 2}, S_2^{\ 2}, S_3^{\ 2}\}$	$= C_4$	$\{O^2\}$	$\{S_3^{\ 1}, O^2, S_1^{\ 2}\}$	$\{S_3^{\ 1}, S_1^{\ 2}, S_2^{\ 2}\}$	
$\{O^2, S_1^{\ 2}, S_2^{\ 2}, O^3\}$	$= C_5$	$\{O^2\}$	$\{O^2, S_1^{\ 2}\}$	$\{S_1^{\ 2}, S_2^{\ 2}\}$	$\{O^2, S_1^{\ 2}, S_2^{\ 2}\}$
$\{S_1^{\ 2}, S_2^{\ 2}, S_3^{\ 2}, O^3, S_1^{\ 3}\}$	$= C_6$		$\{S_1^{\ 2}\}$	$\{S_1^{\ 2}, S_2^{\ 2}\}$	$\{S_1^{\ 2}, S_2^{\ 2}, S_3^{\ 2}\}$
$\{S_2^{\ 2}, S_3^{\ 2}, S_1^{\ 3}, S_2^{\ 3}\}$	$= C_7$			$\{S_2^{\ 2}\}$	$\{S_2^{\ 2}, S_3^{\ 2}\}$

The cliques and separators of the triangulated graphs are formed into junction trees so that the separators have maximum cardinality. Junction trees for the sensor models are presented in figure 5.1.3.

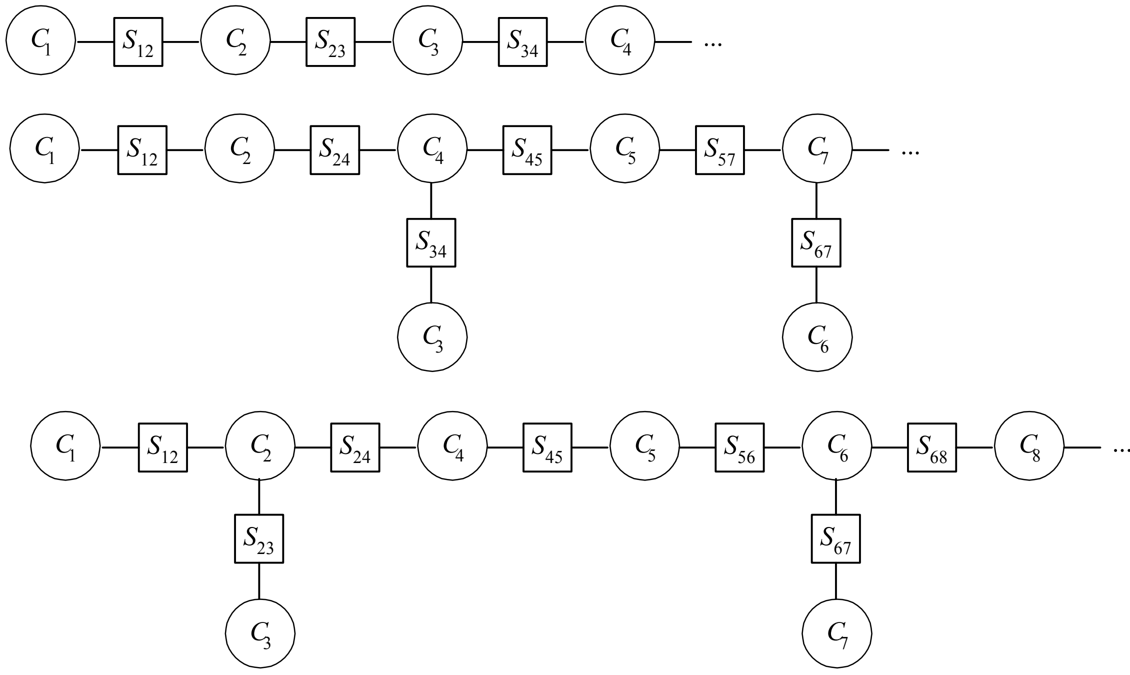


Fig. 5.1.3. Dynamic sensor, sensor fusion, and sensor validation junction trees

These junction trees are graphical inference engines, which perform sensor fusion and validation. With each clique and separator is associated a potential, which is initialized, and then made consistent with the sensor evidence. After the PROPAGATEPOTENTIALS subroutine achieves consistency between neighboring potentials in the junction tree, the potentials form a distributed database for querying the object being sensed.

For a given time period, the junction trees in figure 5.1.3 have two, three, and four cliques respectively. If a processor is associated with each sensor, then this is one more clique than sensor. Assigning the computational responsibility for maintaining, processing, and communicating clique potentials to processors contained in those cliques requires that one sensor be responsible for two cliques.

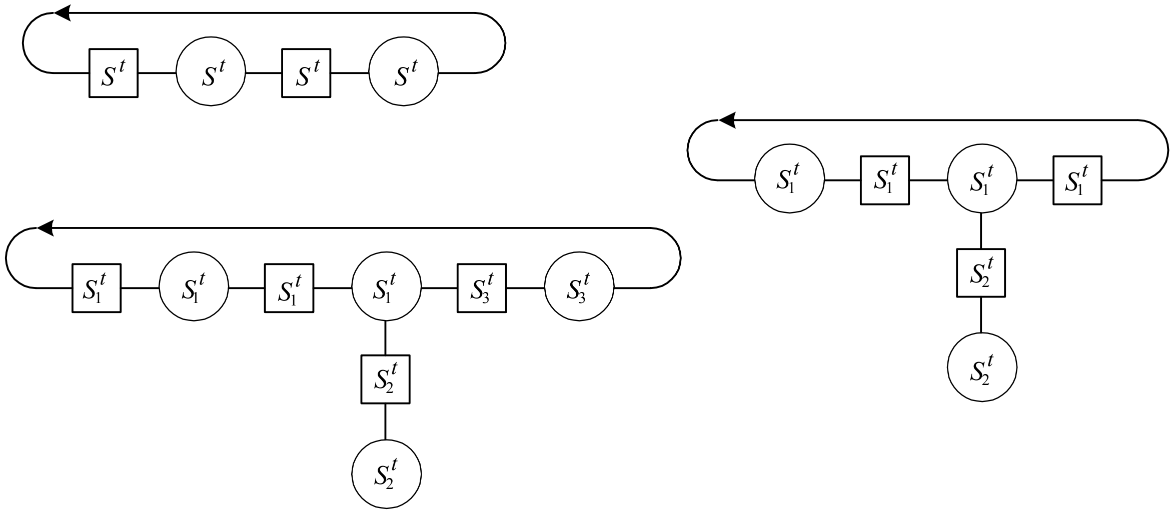


Fig. 5.1.4. Dynamic sensor, sensor fusion, and sensor validation distribution of computational effort

The cliques in the largest graph contain four and five elements (the separators contain three), while the cliques in the smaller graphs are all cliques with three elements (with separators with two elements). As these graphs suggest, if queries are only required for the current object state, then per-state computation and message passing can be reduced to an operation on the per-state subtrees in figure 5.1.4. Sensor asynchrony can also be handled with loops.

6 Summary and Extensions

The JUNCTIONTREE algorithm provides an efficient, decentralized algorithm for probabilistic inference on graphical probability models, useful for sensor fusion and validation, data compression and channel coding, pattern recognition and machine learning, among other applications in wireless sensor networks. The ability of graphical model theory to address issues of computational complexity and decentralization, and the fact that JUNCTIONTREE provides message passing protocols required for distributed computation, makes it useful in these applications. This paper has introduced graphical models and the JUNCTIONTREE algorithm, with applications to wireless sensor networks. The efficiency advantages of the JUNCTIONTREE algorithm are especially large for dynamic Markov models such as for sensor validation and fusion.

Still there exist complex graphical models (with few factorizations) for which JUNCTIONTREE is intractable. Researchers are developing related techniques based on graphical models for approximate inference, including sampling [12] and variational [9] methods.

With a little additional machinery, and slight modification, the JTA can perform (dynamic) expected utility maximization on decision graphs known as *influence diagrams* [6]. Game-theoretic extensions (*multi-agent influence diagrams*) are explored in [11].

These extensions can be hybridized to develop approximate decision- and game-theoretic solution concepts, which are of interest as computational and economic models of bounded rationality.

References

1. Alag, S. S. S.: A Bayesian Decision-Theoretic Framework for Real-Time Monitoring and Diagnosis of Complex Systems: Theory and Applications. University of California at Berkeley: Dept. of Mechanical Engineering Ph.D. Dissertation, 1996.
2. Cormen, T. H., C. E. Leiserson, R. L. Rivest: Introduction to Algorithms. MIT Press, Boston, 1990.
3. Cowell, R. G., A. P. Dawid, S. L. Lauritzen, D. J. Spiegelhalter: Probabilistic Networks and Expert Systems. Springer-Verlag, New York, 1999.
4. Frey, B. J.: Graphical Models for Machine Learning and Digital Communication. MIT Press, Boston, 1998.
5. Hall, D. L., J. L. Linas (eds.): Handbook of Multisensor Data Fusion. CRC Press, New York, 2001.
6. Jensen, F., F. V. Jensen, S. L. Dittmer, "From Influence Diagrams to Junction Trees." Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence, 1994.
7. Jordan, M. I., (ed.): Learning in Graphical Models. MIT Press, Boston, 1999.
8. Jordan, M. I., C. M. Bishop: An Introduction to Graphical Models. MIT Press, Boston, in preparation.
9. Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, L. K. Saul: "An Introduction to Variational Methods for Graphical Models," in Jordan, M. I., (ed.): Learning in Graphical Models. MIT Press, Boston, 1999.
10. Jordan, M. I., T. J. Sejnowski, (eds.): Graphical Models: Foundations of Neural Computation MIT Press, Boston, 2001.
11. Koller, D., B. Milch: "Multi-Agent Influence Diagrams for Representing and Solving Games." Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence, 2001.
12. MacKay, D. J. C.: "Introduction to Monte Carlo Methods," in Jordan, M. I., (ed.): Learning in Graphical Models. MIT Press, Boston, 1999.
13. Russell, S. J., P. Norvig: Artificial Intelligence: A Modern Approach. Prentice Hall, New Jersey, 1995.