Thin Junction Tree Filters

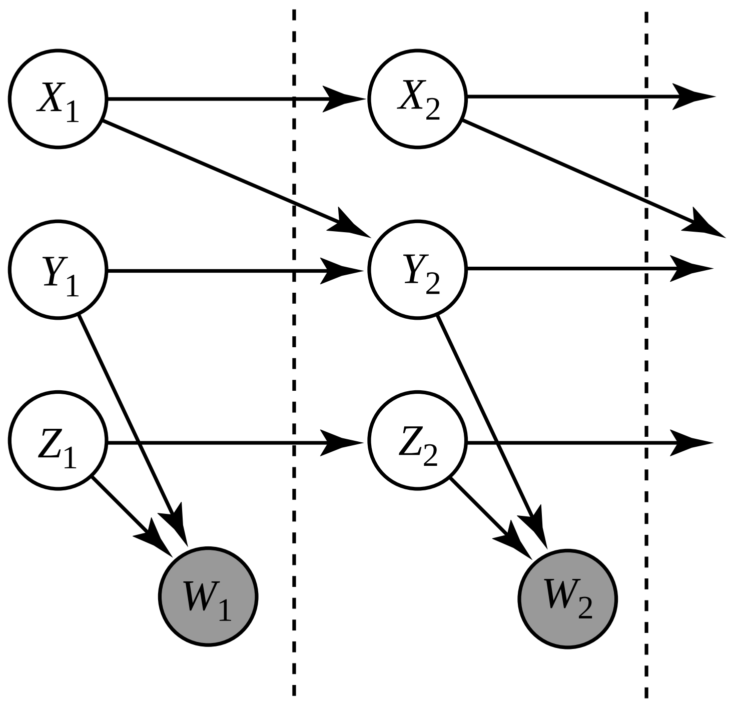
Mark Paskin
Computer Science Division
University of California, Berkeley
mark@paskin.org

Summary

- Thin junction tree filtering (TJTF) is an approximate filtering technique for dynamic Bayesian networks.
- TJTF is an assumed density filter where at each time step the belief state is projected to a density of bounded treewidth.
- The approximation is **adaptive**: the error induced by each of a set of projections is efficiently computed and the best one is executed.
- Thin junction tree filters generalize the popular Boyen & Koller (1998) algorithm beyond products-of-marginals approximations.
- TJTF advances the state-of-the-art in Simultaneous Localization and Mapping, a fundamental problem in Robotics.

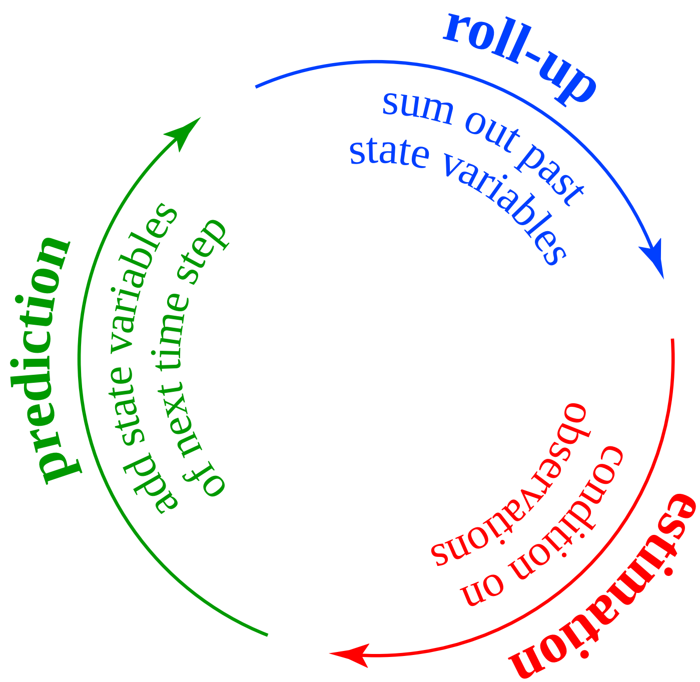
Filtering in dynamic Bayesian networks

A dynamic Bayesian network (DBN) is a **compact** and **modular** representation of a complex discrete-time stochastic process.



The **filtering task** is to compute the *a posteriori* density of some subset of the current state variables, e.g., $p(x_t \mid w_{1:t})$ or $p(x_t, y_t \mid w_{1:t})$.

Recursive filtering



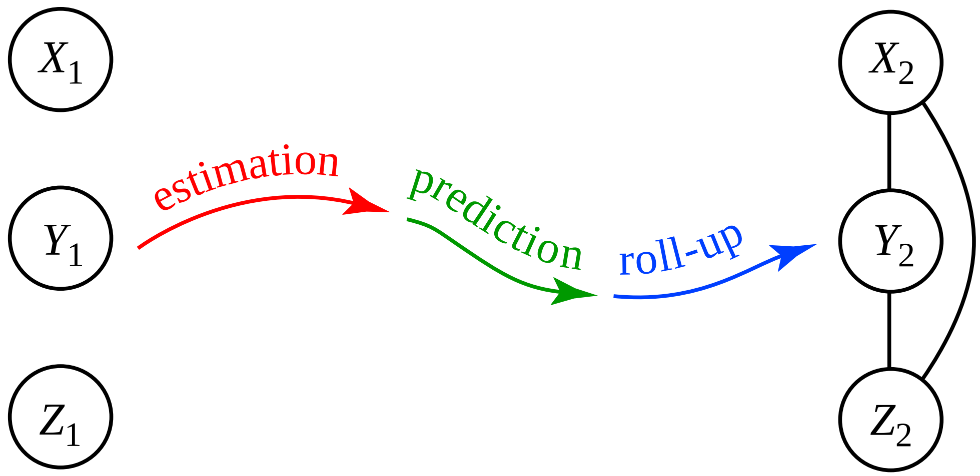
estimation: $p(x_t, y_t, z_t \,|\, w_{1:t}) \propto p(x_t, y_t, z_t \,|\, w_{1:t-1}) \times p(w_t \,|\, y_t, z_t)$

prediction: $$p(x_{t+1}, y_{t+1}, z_{t+1}, x_t, y_t, z_t \,|\, w_{1:t}) \;=\; p(x_t, y_t, z_t \,|\, w_{1:t}) \times p(x_{t+1} \,|\, x_t) \times p(y_{t+1} \,|\, x_t, y_t) \times p(z_{t+1} \,|\, z_t)$$

roll-up: $$p(x_{t+1}, y_{t+1}, z_{t+1} \,|\, w_{1:t}) = \sum_{x_t} \sum_{y_t} \sum_{z_t} p(x_{t+1}, y_{t+1}, z_{t+1}, x_t, y_t, z_t \,|\, w_{1:t})$$

Complexity of filtering in DBNs

Filter updates add edges to the belief state's graphical model:



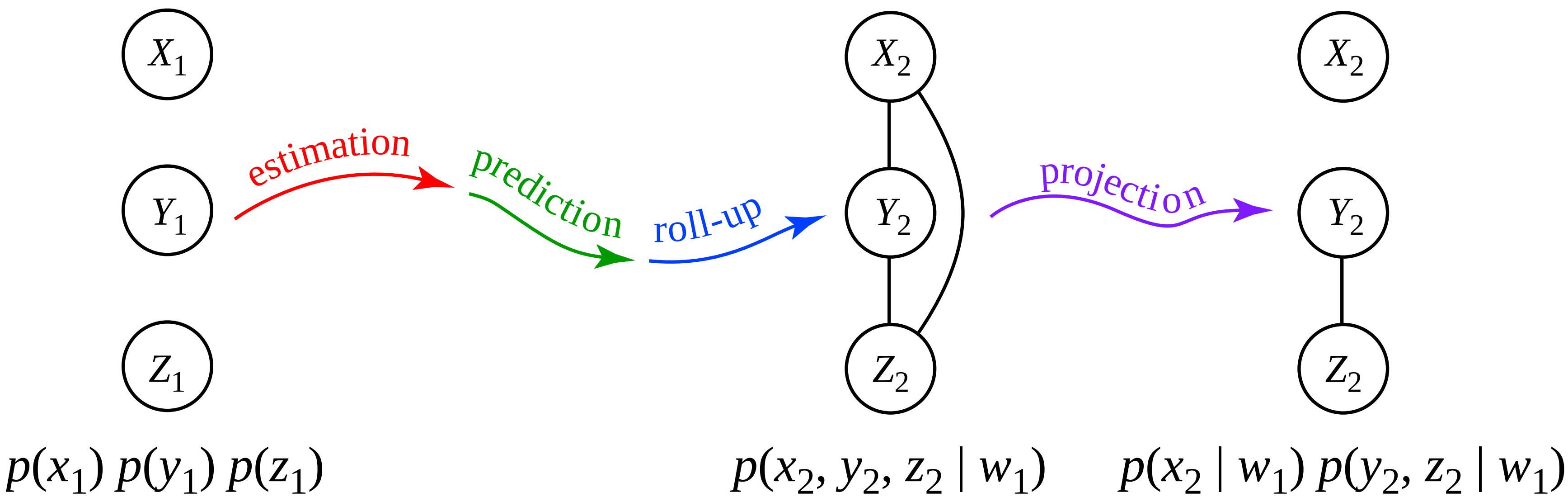
$p(x_1)\,p(y_1)\,p(z_1)$ $\qquad\qquad$ $p(x_2, y_2, z_2 \mid w_1)$

This presents two problems:

1. The size of the belief state representation grows over time.
2. The cost of the filter updates (and inference) grows over time.

The Boyen & Koller (1998) Algorithm

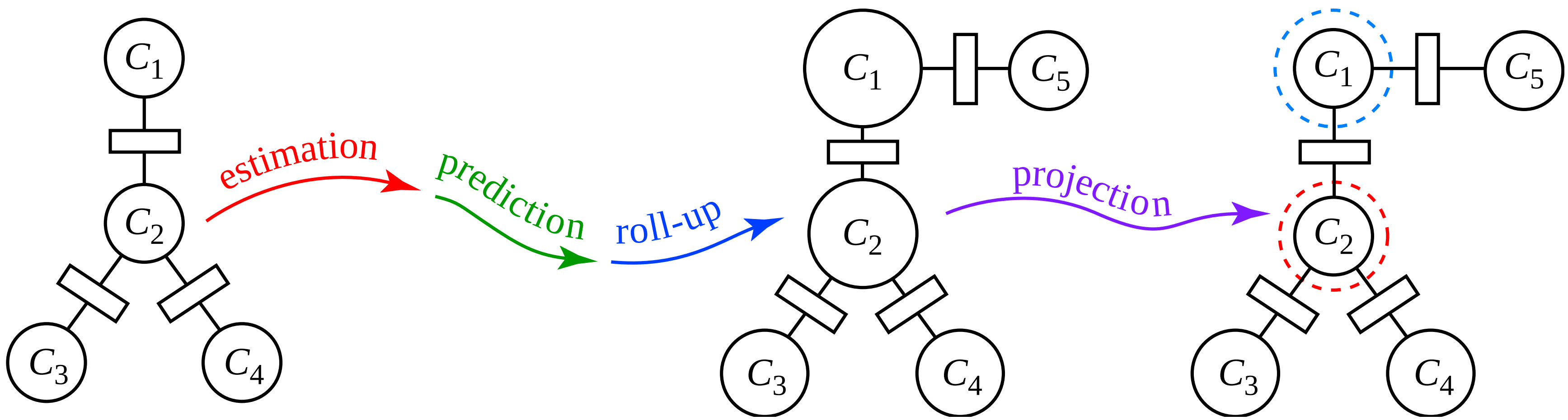
An **assumed density filter** (ADF) adds a **projection** step which chooses a tractable approximation of the belief state. The Boyen & Koller (1998) algorithm projects the belief state to a product of marginals:



This approximation makes filtering tractable, but it discards dependencies that help us make the best use of future observations.

Thin junction tree filters

We represent the belief state using a junction tree with bounded width:



The density represented by the junction tree is

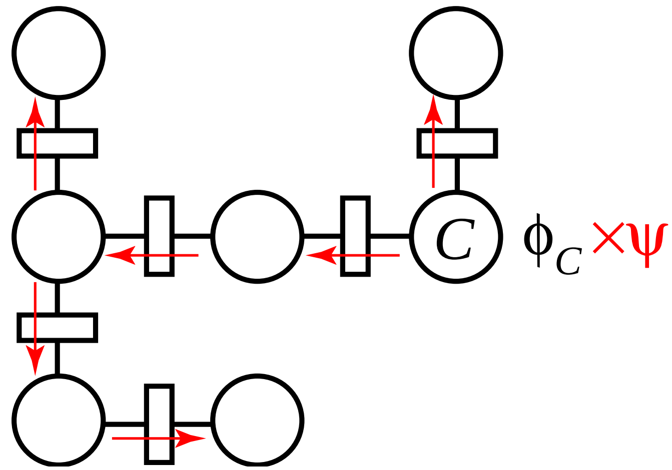
$$p \propto \frac{\prod_{C \in \mathcal{C}} \phi_C}{\prod_{S \in \mathcal{S}} \phi_S}$$

We keep the junction tree **calibrated**, so the cluster and separator potentials are proportional to marginals, i.e., $\phi_C \propto p_C$.

Estimation and prediction

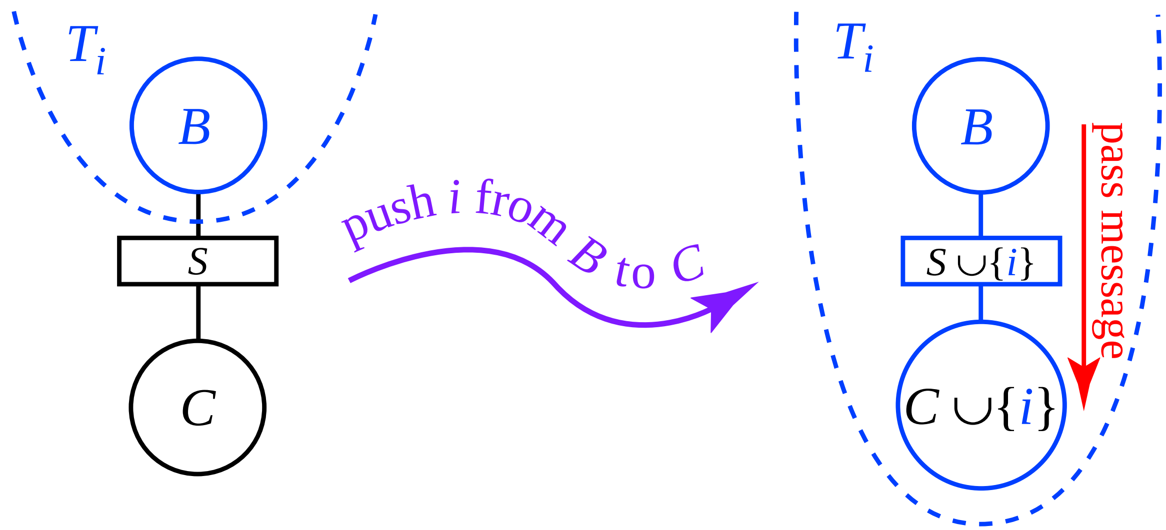
- Estimation and prediction consist of multiplying local potentials (e.g., $p(w_t \mid y_t, z_t)$ or $p(x_{t+1} \mid x_t)$) into the belief state.
- To multiply in a new potential ψ into the junction tree and calibrate:

    1. Find a cluster C that **covers** the potential's variables.
    2. Multiply ψ into ϕ_C.
    3. Distribute evidence from C.



- If there is no cluster that covers the variables of ψ, we must create one.

Pushing variables to create covers



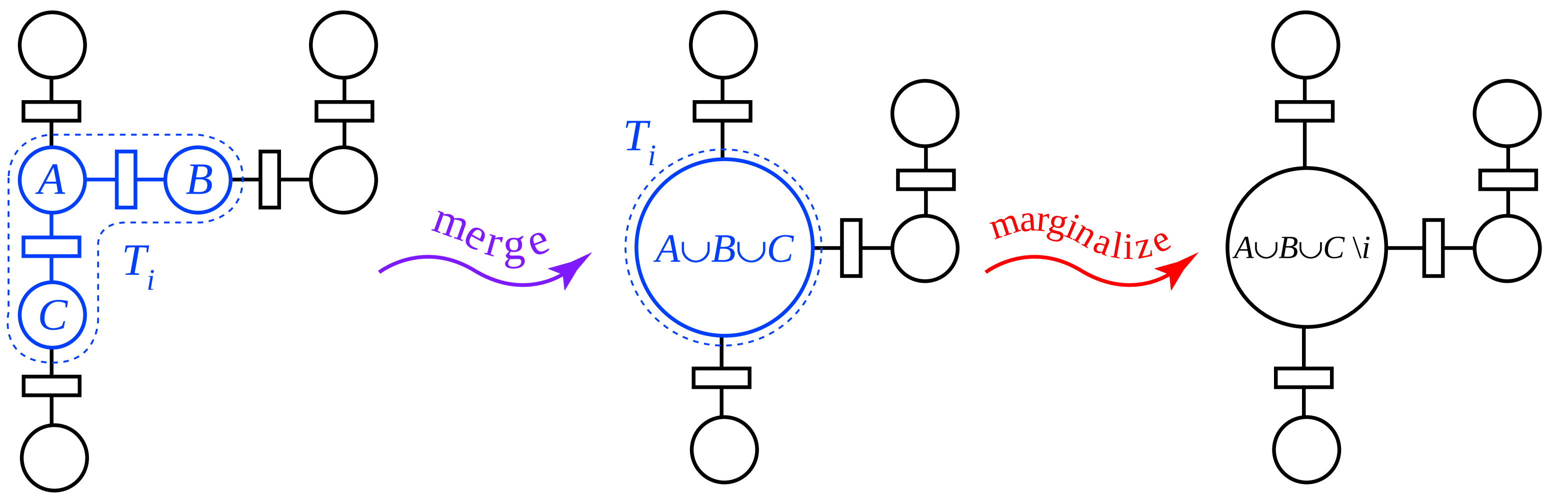
- Pushing yields a valid junction tree and preserves calibration.
- For non-adjacent B and C, push i along the unique path from B to C.
- To make a cluster C cover D, push each of the variables in D to C.

Roll-up

If X_i is present in only one cluster C, then we marginalize X_i out of ϕ_C:

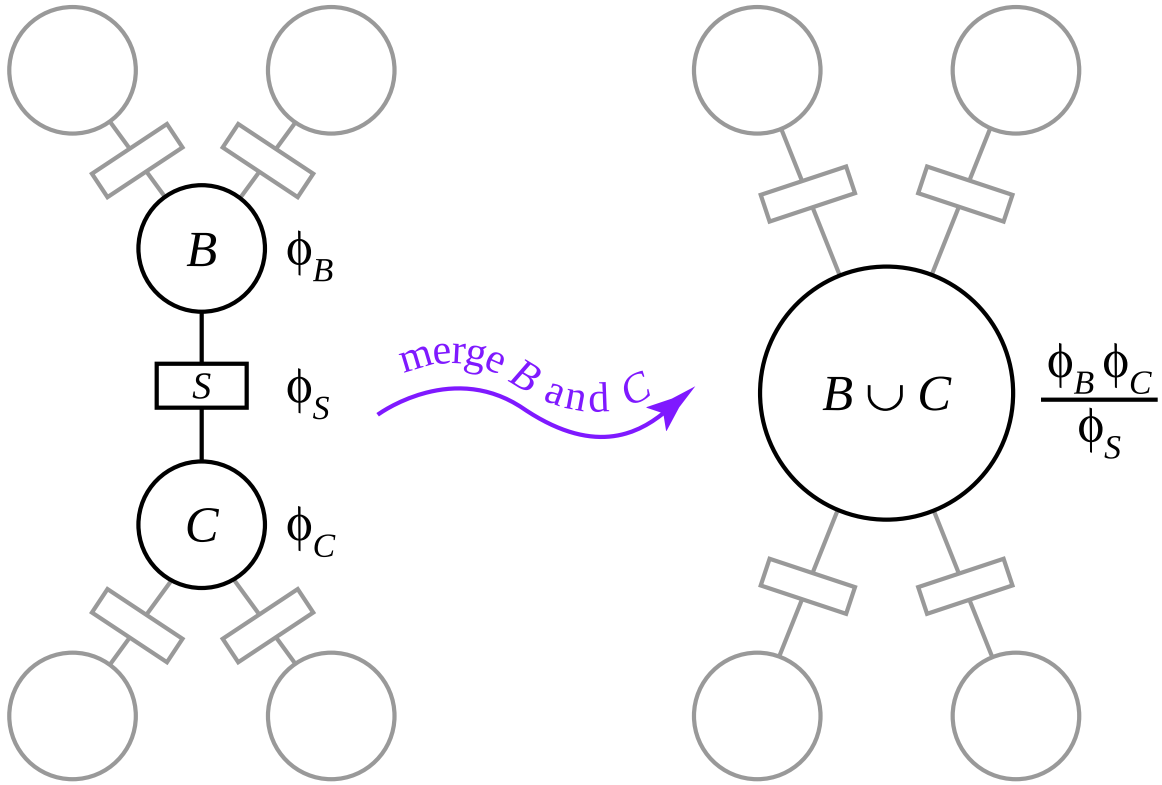
$$C^* \leftarrow C \setminus i \qquad \phi^*_C \leftarrow \sum_{x_i} \phi_C$$

If not, we must first merge all clusters containing X_i into a single cluster:



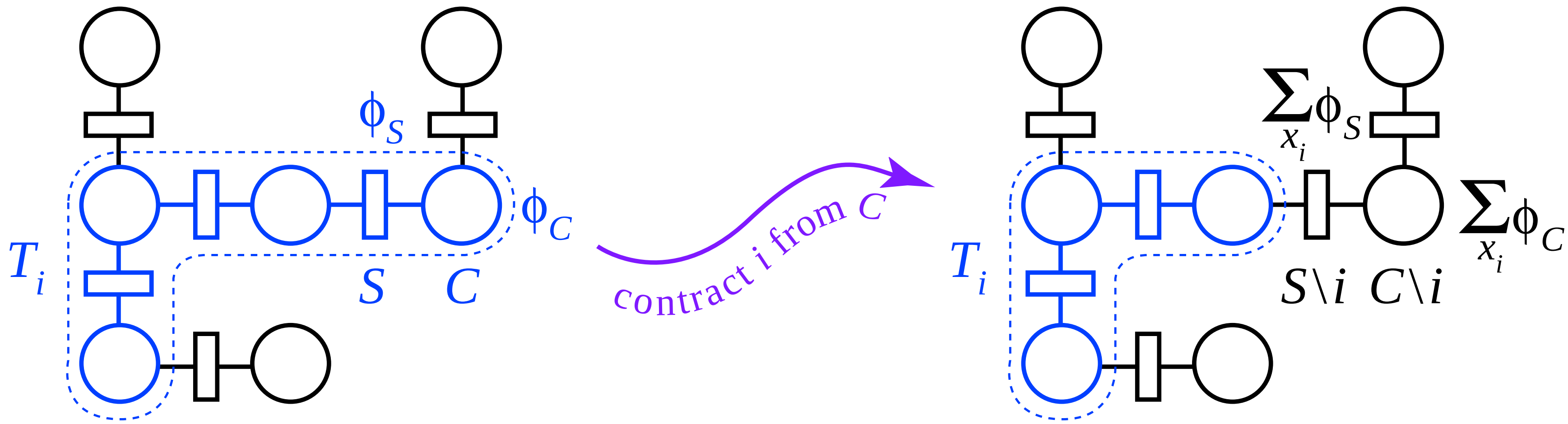
This is necessary to cover the elimination clique over the Markov blanket of X_i.

Merging clusters



- Merging clusters yields a valid junction tree and preserves calibration.
- We can merge all clusters containing X_i by a sequence of pairwise merges.

Variable contraction



- The result is a valid junction tree, but perhaps not for the original density.
- Variable contractions preserve calibration.
- Effect on the corresponding graphical model:

  Contracting i from C cuts all edges between X_i and X_{C-S}, the variables it no longer resides with.

Variable contraction is an efficient I-projection

Proposition. *If $\tilde{p}$ is the density obtained by contracting i from C, then*

$$\tilde{p} = \underset{\{q \,:\, X_i \perp\!\!\!\perp X_{C-S} \,|\, X_{S\setminus i}\}}{\arg\min} D(p \,||\, q)$$

Adaptive approximation

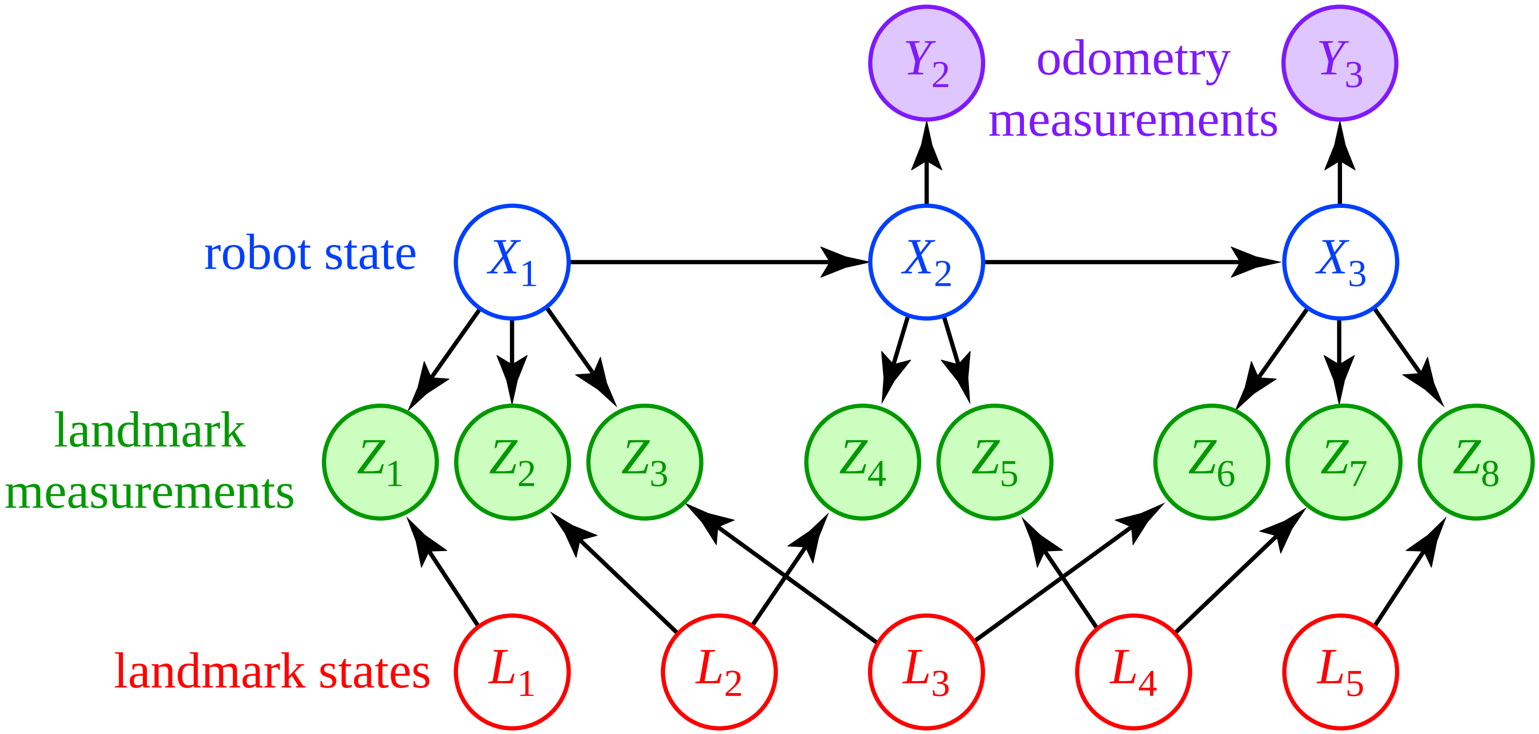
Proposition. *If $\tilde{p}$ is the density obtained by contracting i from C, then*

$$D(p \,||\, \tilde{p}) = I(X_i; X_{C-S} \,|\, X_{S \setminus i})$$

- This error can be computed locally using the cluster potential $\phi_C \propto p_C$.
- In Gaussian densities this error can be computed in $O(\dim(X_i)^3)$ time.
- To "thin" a large cluster C: perform the contraction with minimum error.
- Note: we may be using an approximate model to compute the error.

Simultaneous Localization and Mapping (SLAM)

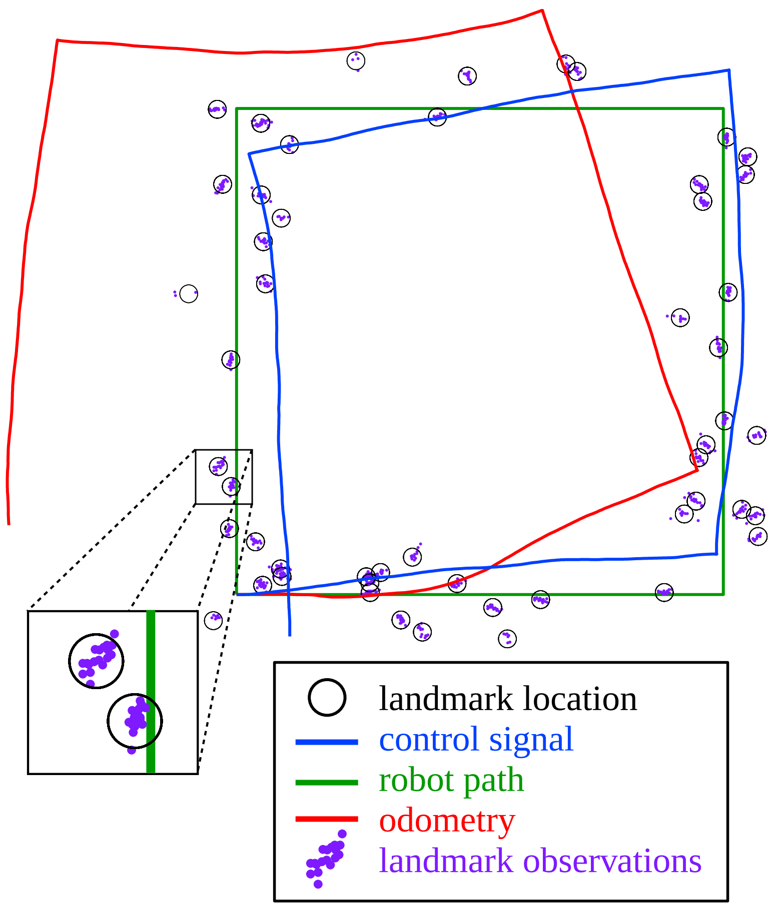
A mobile robot navigating in an unknown environment must incrementally build a map of its surroundings and localize itself within that map.

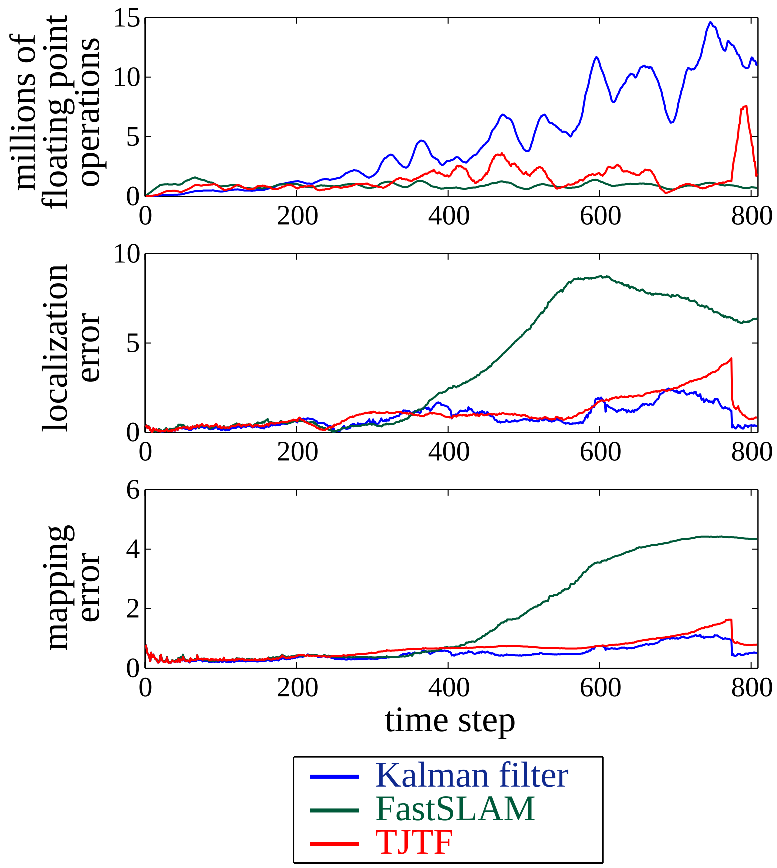


The SLAM belief state is **completely connected** after each filter update: it has no conditional independencies.

Thin junction tree filters for SLAM

The space and time complexity of the Kalman filter is **quadratic** in the number of observed landmarks, whereas TJTF is **linear**. Using **adaptive message passing**, the TJTF filter is often **constant time**.





http://www.cs.berkeley.edu/~paskin/slam

- the IJCAI 2003 paper
- a companion technical report
- movies and implementations of several types of SLAM filters