

Junction Tree Algorithms for Inference in Dynamic Bayesian Networks (DBNs)

**Kevin Gimpel
September 2005**

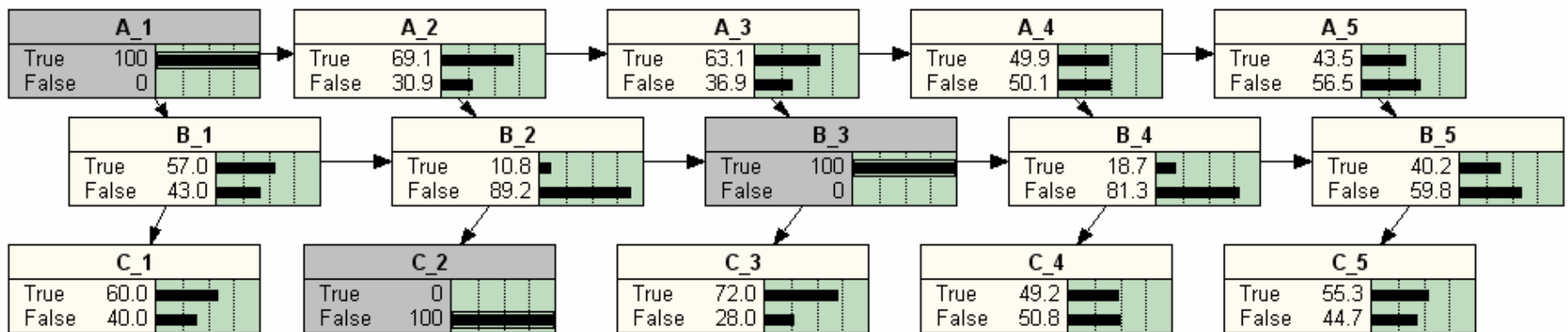
Outline

- Introduction to Inference in DBNs
- Junction Tree algorithms for DBNs currently implemented
 - Kevin Murphy's Interface Algorithm
 - Boyen-Koller (BK) Approximation

Introduction to Inference in DBNs

- Naïve Approach

- “Unroll” DBN for desired number of timesteps and treat as a static BN
- apply evidence at appropriate times and then run any static algorithm
- Example shown below in which evidence has been applied as shown and the current time is $t = 3$



- Inference Problems:

- Smoothing: querying from a previous time ($t < 3$) (e.g., $P(B_{1})$)
- Filtering: querying at the current time ($t = 3$) (e.g., $P(A_{3})$)
- Prediction: querying at a future time ($t > 3$) (e.g., $P(B_{5})$)

Introduction to Inference in DBNs

- Naïve Approach

- Pros:

- Simple, “already” implemented
 - Smoothing, filtering, and prediction have intuitive manifestations
 - Provides a way to validate other dynamic inference algorithms

- Cons:

- Typically unknown beforehand how many timesteps are “desired”
 - With many timesteps, unrolled DBN becomes huge; static inference algorithms run out of memory or take too long
 - We often want to do dynamic filtering, which requires rebuilding the entire time history of the process on every timestep

Introduction to Inference in DBNs

- Better Approach

- We don't need the entire unrolled DBN
- A DBN represents a process that is *stationary* and *Markovian*:
 - Stationary: the node relationships within a timeslice t and the transition function from timeslice t to timeslice $t+1$ do not depend on t
 - Therefore, we only need the initial timeslice and sufficient consecutive timeslices to show the transition function
 - Markovian: the transition function depends only on the immediately-preceding timeslice and not on any previous timeslices (e.g., no arrows go from timeslice t to timeslice $t+2$)
 - Therefore, the set of nodes in a timeslice d-separate the past from the future
- This is why we can represent a DBN using just a 2TBN that represents the first two timeslices of a process
 - We can do inference using just this structure too

Introduction to Inference in DBNs

- Better Approach (continued)

- Therefore, dynamic inference boils down to doing static inference on the 2TBN and then following some protocol for “advancing” forward one timestep

- Two algorithms that use the static junction tree algorithm for inference on the 2TBN:

1. *Frontier Algorithm* (Zweig, 1996)

- » The *frontier* is the full set of hidden nodes at the current timeslice which d-separate the process into two segments

2. *Interface Algorithm* (Murphy, 2002)

- » More efficient than frontier algorithm since it uses the *interface* (subset of the frontier) to d-separate the process

- » Currently implemented as the Dynamic Junction Tree Algorithm

- » Also called “1.5 Slice Junction Tree Algorithm” (PNL)

Outline

- Introduction to Inference in DBNs
- Junction Tree algorithms for DBNs currently implemented
 - Kevin Murphy's Interface Algorithm
 - Boyen-Koller (BK) Approximation

Junction Tree Algorithms

- Junction Tree algorithms for static Bayesian networks
 - Most widely-researched exact inference algorithm family for static BNs
 - Many variants have been developed
 - Variations include: particulars of secondary structure used, message-passing scheme, and message format (cluster-sepset absorption protocol)
- Junction Tree algorithms for dynamic Bayesian networks
 - Many variants, like the static case
 - All use a static junction tree algorithm as a subroutine
 - Any static variant can be used
 - Versions have been developed for every dynamic inference problem: smoothing, filtering, prediction, etc.
 - Allow approximation schemes
 - E.g., Boyen-Koller (BK) approximation

Junction Tree Algorithms

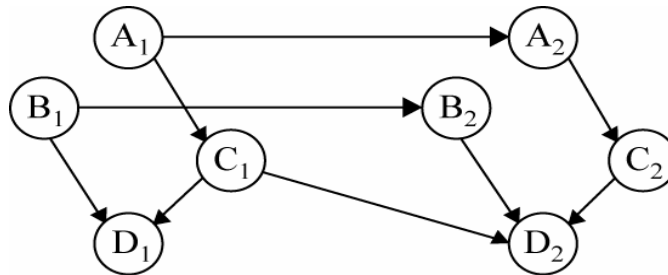
- Junction Tree algorithm for DBNs currently implemented
 - Follows Kevin Murphy's *Interface algorithm* from:
 - Kevin Murphy. "Dynamic Bayesian Networks (Draft)," November 2002. To appear in *Probabilistic Graphical Models*, Michael Jordan.
 - Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, U.C. Berkeley, 2002.
 - Both available from www.cs.ubc.ca/~murphyk/papers.html
- Current implementation
 - Simplified version which only does filtering
 - Evidence must be applied sequentially and queries can only be issued for nodes in the current timeslice
 - Possible extensions:
 - Smoothing
 - Support for networks with linear Gaussian relationships (requires static algorithm to be extended first)

Outline

- Introduction to Inference in DBNs
- Junction Tree algorithms for DBNs currently implemented
 - Kevin Murphy's Interface Algorithm
 - Boyen-Koller (BK) Approximation

Preliminaries

- **Definition: Outgoing Interface, I_t**
 - Set of nodes in timeslice t with children in timeslice $t+1$
 - E.g., $\{A_1, B_1, C_1\}$ is the outgoing interface of timeslice 1 below:



- **Theorem: I_t d-separates the past from the future (Murphy 2002)**
 - Where “past” = all nodes in timeslices before t and all non-interface nodes in timeslice t , and “future” = nodes in timeslices $t+1$ and later
 - Therefore, the outgoing interface encapsulates all necessary information about previous timeslices to do filtering

Algorithm Outline

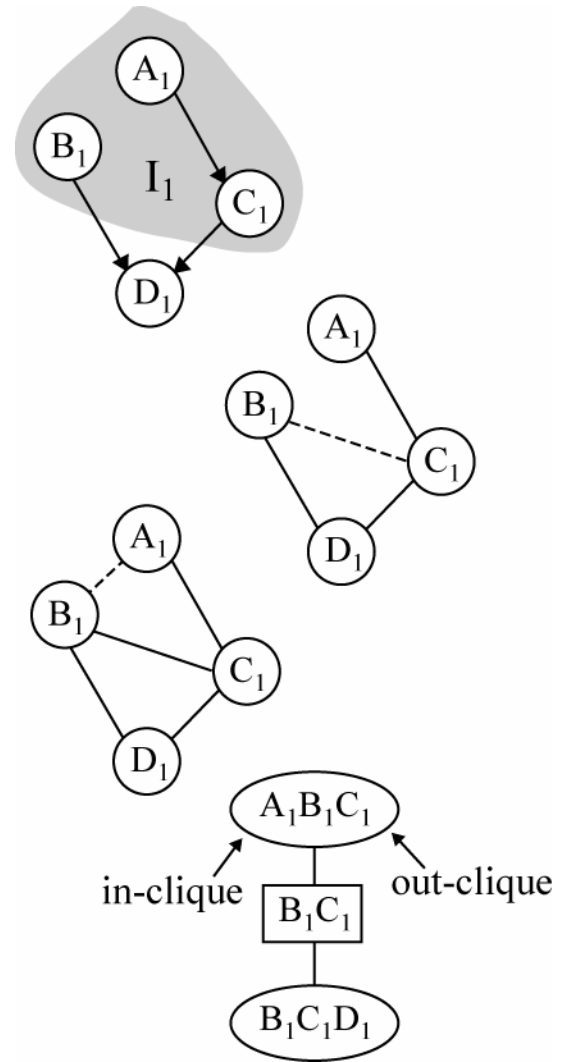
- Initialization
 - On initialization, we create two junction trees, J_1 and J_t
 - J_1 is the junction tree for the initial timeslice and is created from timeslice 1 of the 2TBN
 - J_t is the junction tree for each subsequent timeslice and is created from timeslice 2 of the 2TBN and the outgoing interface of timeslice 1
 - Time is initialized to 0
- Queries
 - Marginals of nodes at the current timeslice can be queried
 - If current time = 0, queries are performed on “_1” nodes in J_1
 - If current time > 0, queries are performed on “_2” nodes in J_t

Algorithm Outline (cont.)

- Evidence Application
 - Evidence can be applied to any node in the current timeslice
 - If current time = 0, evidence is applied to “_1” nodes in J_t
 - If current time > 0, evidence is applied to “_2” nodes in J_t
- Advance
 - Increment time counter
 - Use outgoing interface from active timeslice to do inference in next timeslice
 - Since the outgoing interface d-separates the past from the future, this ensures that when we do inference in the next timeslice we are taking everything that has occurred “so far” into account

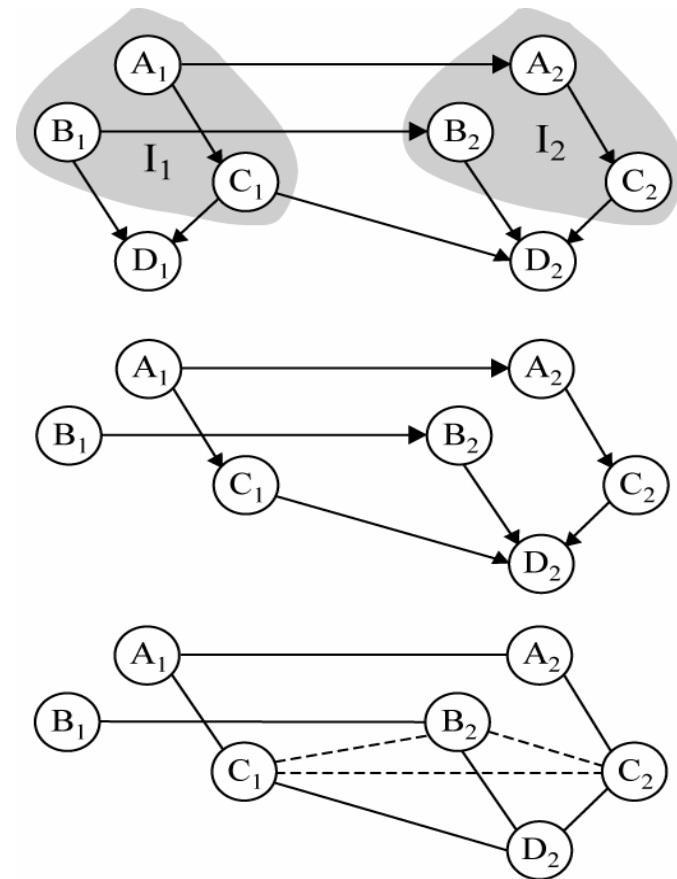
Initialization

- Initialization of J_1
 - Remove all nodes in timeslice 2 from the 2TBN
 - Identify nodes in outgoing interface of timeslice 1, call it I_1
 - Moralize (same as with static algorithm)
 - **Key change:** Add edges to make I_1 a clique
 - Triangulate, find cliques, form junction tree (same as static)
 - Find clique that contains I_1 , call it the *in-clique* (and *out-clique*)
 - Initialize clique potentials to 1's, multiply nodes' CPTs onto cliques (same as static)



Initialization

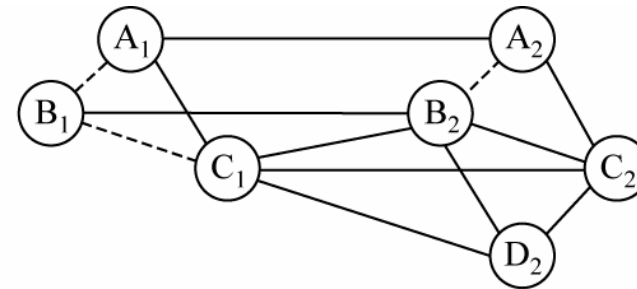
- Initialization of J_t
 - Starting with the whole 2TBN, identify nodes in outgoing interfaces of timeslices 1 and 2
 - call them I_1 and I_2 , respectively
 - Convert 2TBN to 1.5DBN (remove non-interface nodes in timeslice 1)
 - Moralize (same as static)



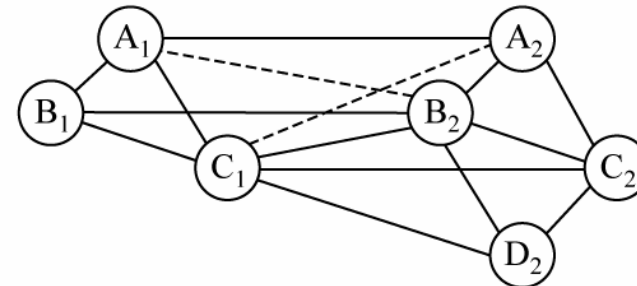
Initialization

- Initialization of J_t (continued)

- Complete outgoing interfaces
(add edges to make I_1 and I_2 each cliques)



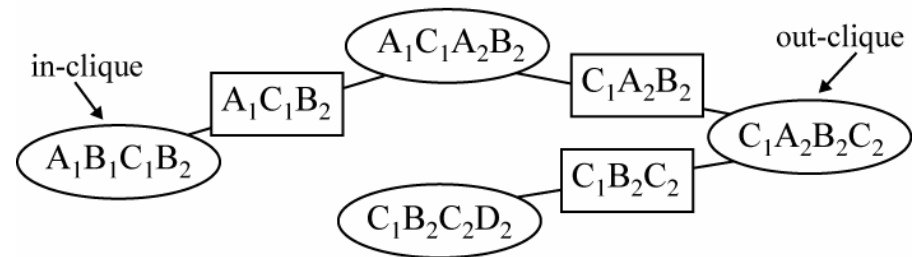
- Triangulate, find cliques, form junction tree (same as static)



Cliques:
 $A_1B_1C_1B_2$
 $A_1C_1A_2B_2$
 $C_1A_2B_2C_2$
 $C_1B_2C_2D_2$

- Find cliques that contain I_1 and I_2 , call them the *in-clique* and *out-clique*

- Initialize clique potentials to 1's, multiply nodes' potentials onto cliques



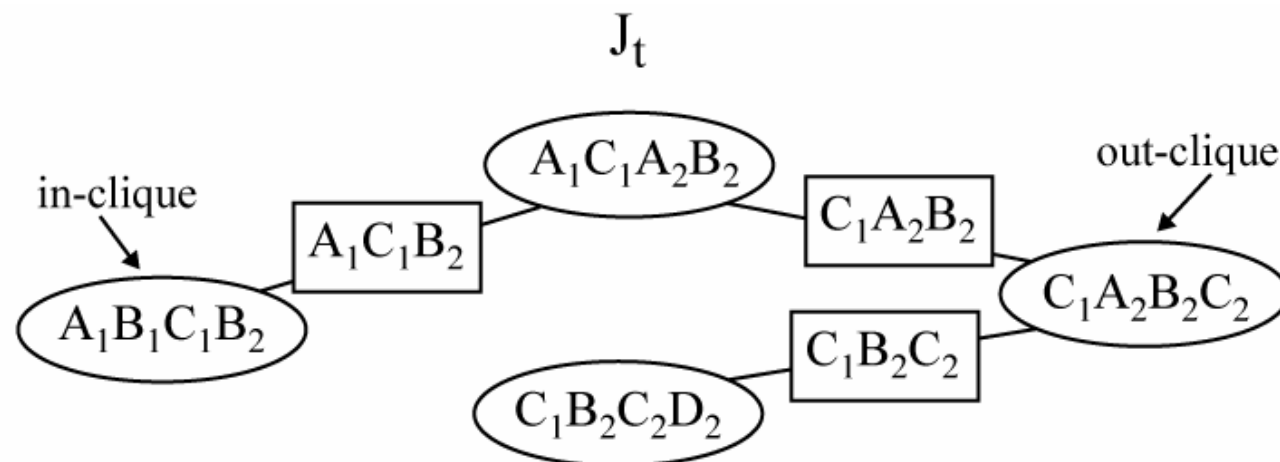
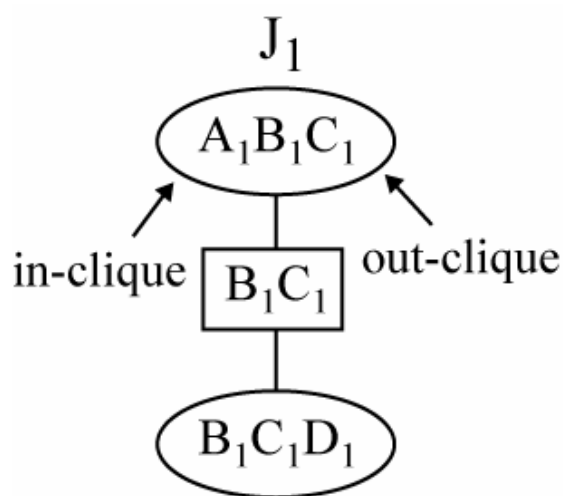
- **Key Change:** Only multiply timeslice 2 nodes' CPTs from a clique onto that clique's potential – Why?

Initialization

- Initialization of J_t (continued)
 - Why do we only multiply the CPTs of nodes in timeslice 2 onto the clique potentials?
 - J_t represents two timeslices, but actually only performs inference on the second
 - Evidence is applied and nodes are queried only in timeslice 2 of J_t
 - Timeslice 1 nodes are present to represent the execution of the process since its start
 - This is fully encapsulated in the potential on the outgoing interface of timeslice 1, which we obtain from the out-clique from the *previous timestep of the process*

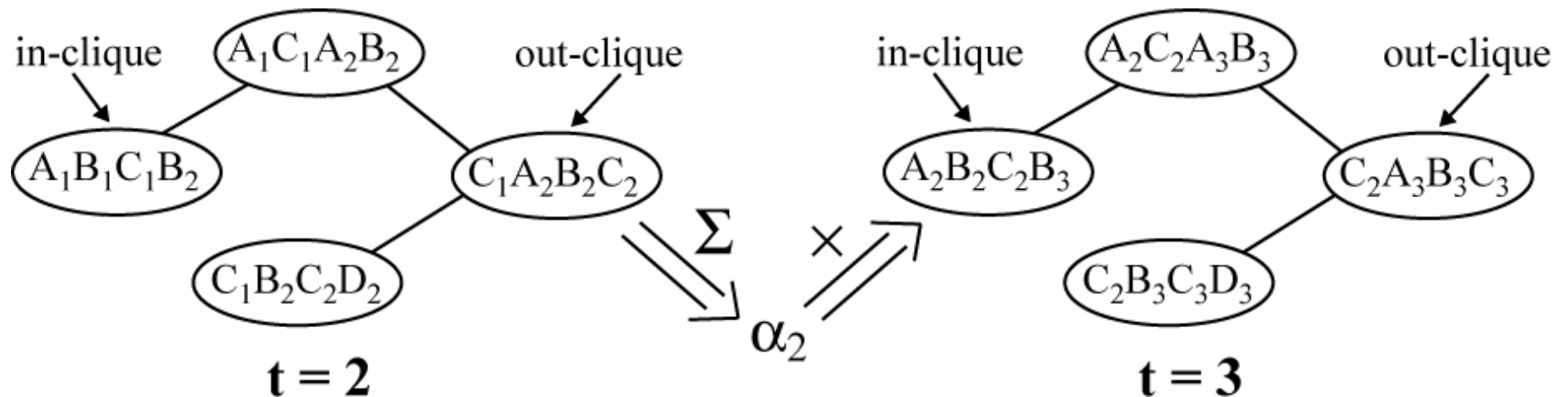
Initialization – Summary

- J_1 and J_t are initialized
 - Both are created from the input 2TBN
 - CPTs of “_1” nodes are multiplied onto clique potentials in J_1 , CPTs of “_2” nodes onto clique potentials in J_t
 - J_1 is used for the first timeslice and J_t is used for each additional timeslice
- Time is initialized to 0



Advance - Overview

1. Get α by marginalizing out-clique potential in current junction tree down to outgoing interface potential
2. Increment time
3. Multiply α onto in-clique in new junction tree



1. $\alpha_2 = \phi_{A_2B_2C_2} = \sum_{C_1} \phi_{C_1A_2B_2C_2}$

2. $t++$

3. $\phi_{A_2B_2C_2B_3} = \alpha_2 * \phi_{A_2B_2C_2B_3}$

Advance – Details

1. Before time is incremented:

- Get current junction tree (if time = 0, this is J_1 ; otherwise, this is J_t)
- Update beliefs in current junction tree if necessary
- Get α_t by marginalizing out-clique potential in current junction tree down to outgoing interface potential

2. Increment Time

3. After time is incremented:

- Get current junction tree (this will always be J_t since time > 0)
- Multiply α_t onto in-clique potential in current junction tree

The out-clique potential is the only quantity we need to keep when we advance from timestep to timestep

Outline

- Introduction to Inference in DBNs
- Junction Tree algorithms for DBNs currently implemented
 - Kevin Murphy's Interface Algorithm
 - Boyen-Koller (BK) Approximation

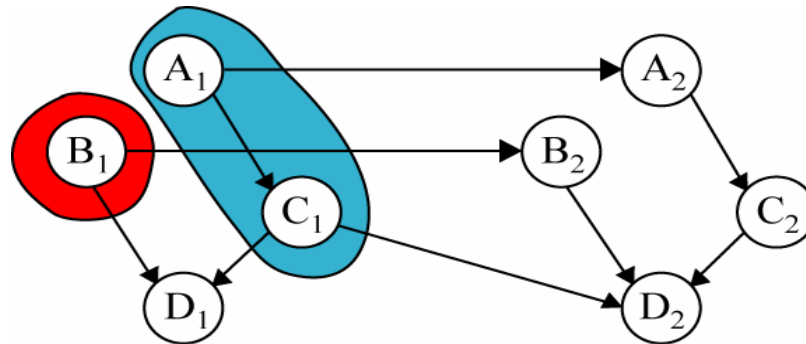
Boyen-Koller Approximation

- General approximation scheme for stochastic processes
 - From:
 - Xavier Boyen and Daphne Koller. “Tractable Inference for Complex Stochastic Processes.” In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 33–42, 1998.
 - This paper also shows how to use the BK approximation in the junction tree algorithm for DBNs
 - Requires less computation and storage than interface algorithm
 - Based on assumptions of independence between chosen sets of variables in outgoing interface
- Current implementation
 - Minor modifications to interface algorithm
 - User must specify sets among which independence is to be assumed

Preliminaries

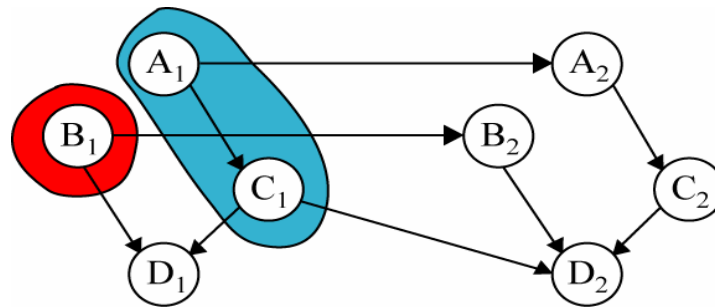
- **Definition: BK Clusters**

- Sets of nodes which form a partition of outgoing interface
- BK approximation scheme assumes independence among these sets
- E.g., $\{A_1, B_1, C_1\}$ is the outgoing interface of timeslice 1 below, and the set $\{\{A_1, C_1\}, \{B_1\}\}$ is a valid set of BK clusters:



Preliminaries

- Interface algorithm uses potential α_t on outgoing interface when advancing timesteps
 - BK approximates this potential as the product of the BK cluster potentials:



$$\alpha_1 = \phi_{A_1 B_1 C_1} \approx \phi_{A_1 C_1} \phi_{B_1}$$

- Multiplying two potentials is less expensive than working with the joint
- Best accuracy is obtained if no node in any one cluster is parent of a node in a different cluster within a single timeslice

Preliminaries

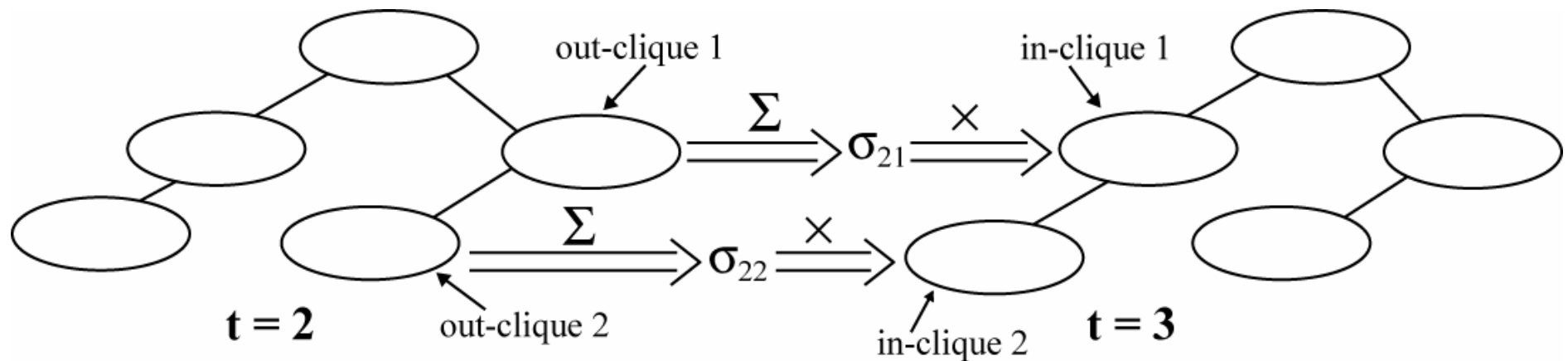
- How good of an approximation is it?
 - Couldn't error grow unboundedly over time as the process advances?
 - *BK Theorem*: error (KL-distance from truth) remains bounded indefinitely
 - Intuition: even though additional error is introduced at every timestep, the stochastic nature of the process and the informative nature of the evidence prevent the error from building up
 - Best case:
 - Transition from timestep to timestep is *maximally stochastic* (i.e., X_t does not depend on X_{t-1}) and observations are noiseless
 - Worst case:
 - Transition is deterministic and observations are not present or uninformative

Modifications to Interface Algorithm

- Initialization of J_1 and J_t
 - Instead of completing outgoing interface, complete each BK cluster (to ensure that each BK cluster is fully contained within a clique of the junction tree)
 - Keep track of multiple in-cliques and out-cliques, i.e., each BK cluster has its own in-clique and out-clique
- Advance
 - Since we have multiple out-cliques, we have to marginalize each of them down to their respective BK cluster potentials
 - This gives us a list of BK cluster potentials, which we call σ_t
 - The product of the potentials in σ_t is the approximation for α_t
 - After incrementing time and resetting J_t , multiply each of the BK cluster potentials in σ_t onto their respective in-cliques in J_t
 - Figure on next slide

Modifications to Interface Algorithm

- Example with two BK clusters:

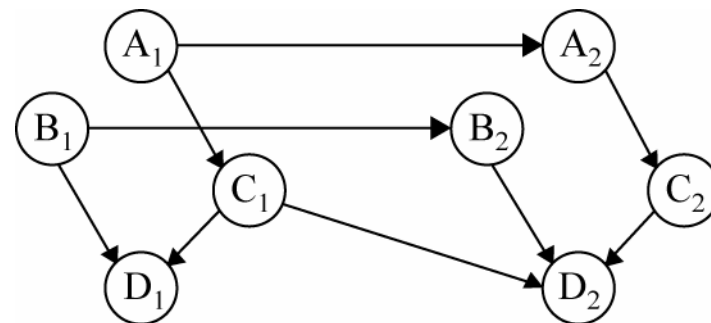


1. $\sigma_{21} = \sum \phi_{\text{out-clq1}}$
 $\sigma_{22} = \sum \phi_{\text{out-clq2}}$
2. $t++$
3. $\phi_{\text{in-clq1}} = \sigma_{21} * \phi_{\text{in-clq1}}$
 $\phi_{\text{in-clq2}} = \sigma_{22} * \phi_{\text{in-clq2}}$

BK Algorithm Example Results

Average KL-Divergence for Various BK Cluster Factorizations					
	ABC	AC B	AB C	A BC	A B C
KL-Divergence	0	6.796e-08	3.482e-04	3.518e-04	3.489e-04
Time (s)	8.2291	7.6949	7.9081	7.9741	7.3594

- Run for 10,000 timesteps, random evidence applied to node D on each timestep
- Nodes queried: A, B, C
- KL-Divergence shown is absolute value of KL-D from truth of queried distributions, averaged over A, B, and C
- Time given is for execution of all 10,000 timesteps and is averaged over 15 runs
- As shown, best speed obtained with fully-factored approximation and best accuracy obtained when no node in any one cluster is parent of a node in a different cluster within a single timeslice (ABC, AC | B)
- The interface algorithm is simply a special case of BK with no independence assumed (ABC)



Future Work

- Extensions to allow smoothing, prediction
- Many additional algorithm variants could be implemented, but there is currently no immediate need
 - Existing functionality should be solidified, rewritten as necessary, and well-documented