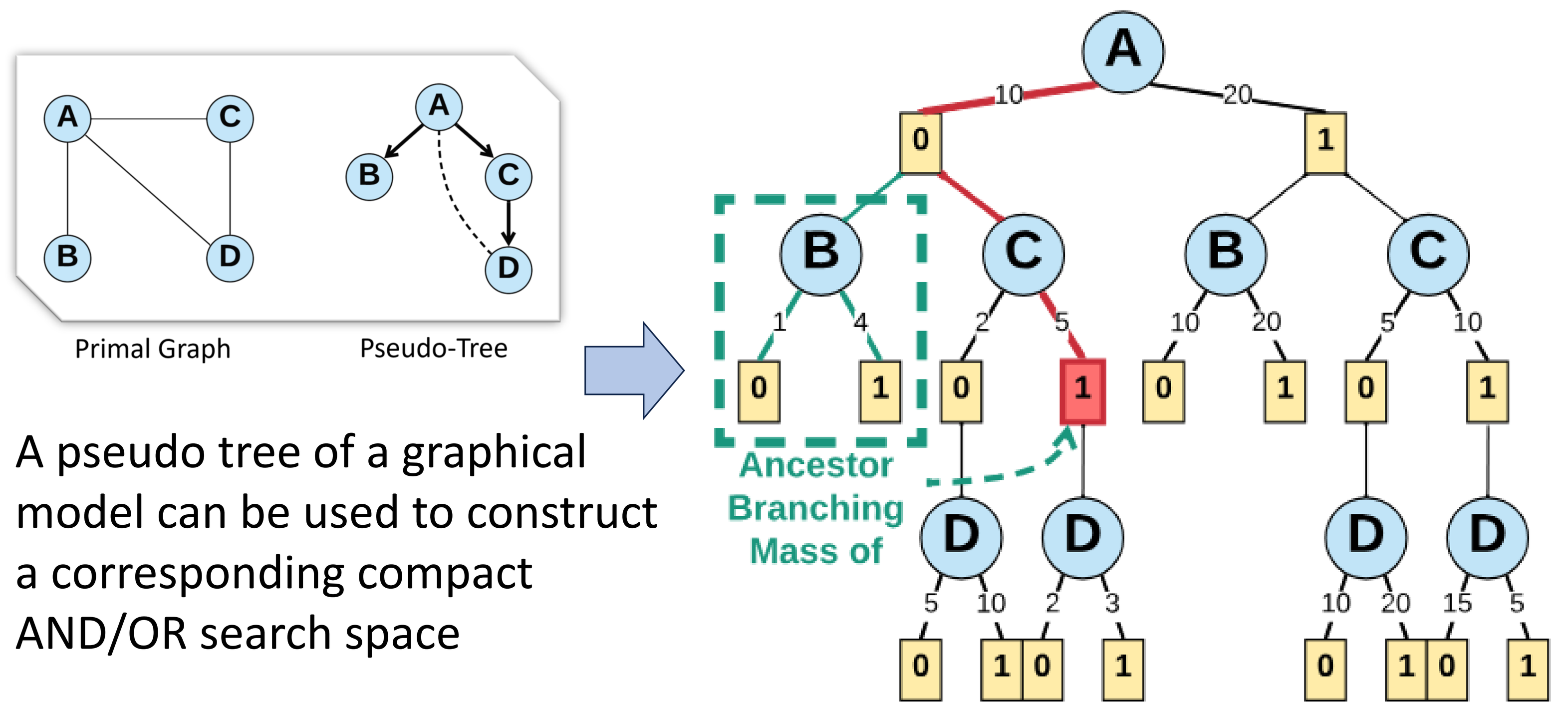
Value-Based Abstraction Functions for Abstraction Sampling

Bobak Pezeshki (pezeshkb@uci.edu), Kalev Kask (kkask@uci.edu), Alexander Ihler (ihler@ics.uci.edu), Rina Dechter (dechter@ics.uci.edu)

Abstract

Monte Carlo methods are powerful tools for solving problems involving complex probability distributions. Despite their versatility, these methods often suffer from inefficiencies, especially when dealing with rare events. As such, importance sampling emerged as a prominent technique for alleviating these challenges. Recently, a new scheme called Abstraction Sampling was developed that incorporated stratification to importance sampling over graphical models. However, existing work only explored a limited set of abstraction functions that guide stratification. This study introduces three new classes of abstraction functions combined with seven distinct partitioning schemes, resulting in twenty-one new abstraction functions, each motivated by theory and intuition from both search and sampling domains. An extensive empirical analysis on over 400 problems compares these new schemes highlighting several well-performing candidates.

AND/OR Search Spaces



A pseudo tree of a graphical model can be used to construct a corresponding compact AND/OR search space

Task

Estimate the normalizing constant of the graphical model:

$$Z = \sum_{X} \prod_{\alpha} f_{\alpha}(X_{\alpha})$$

Strategy

Obtain Monte Carlo estimate via AND/OR Abstraction Sampling

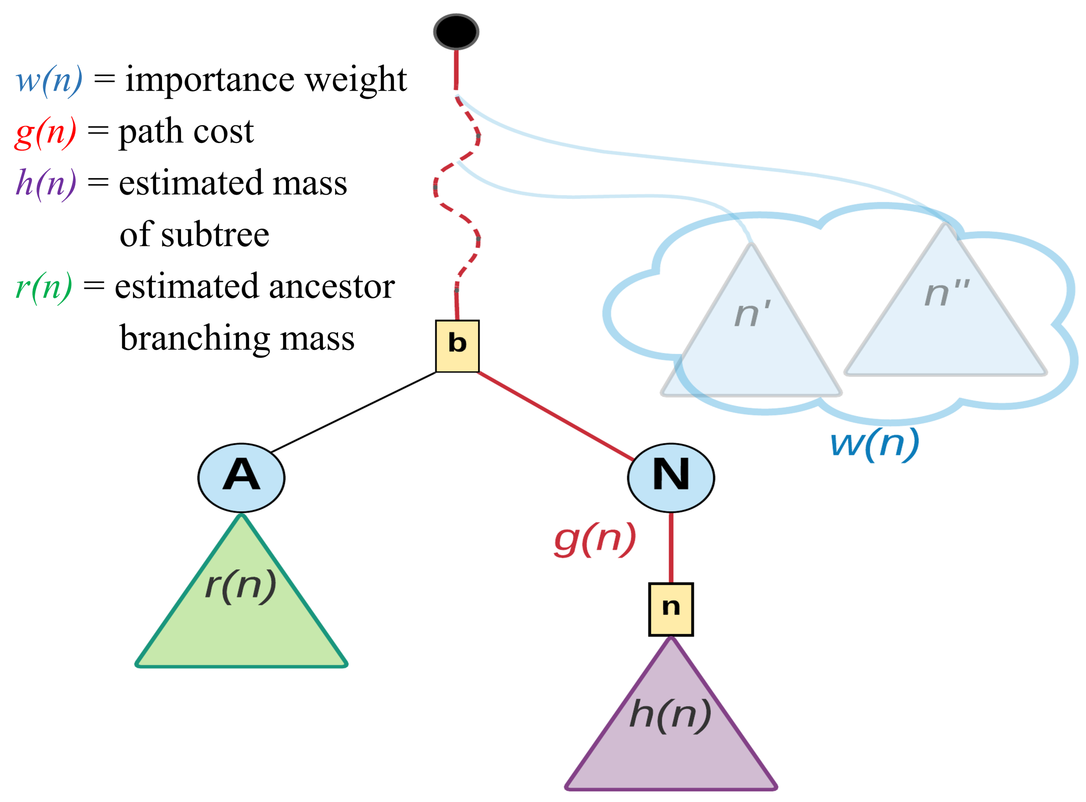
[Broka et al., 2018; Kask et al., 2020]

Question

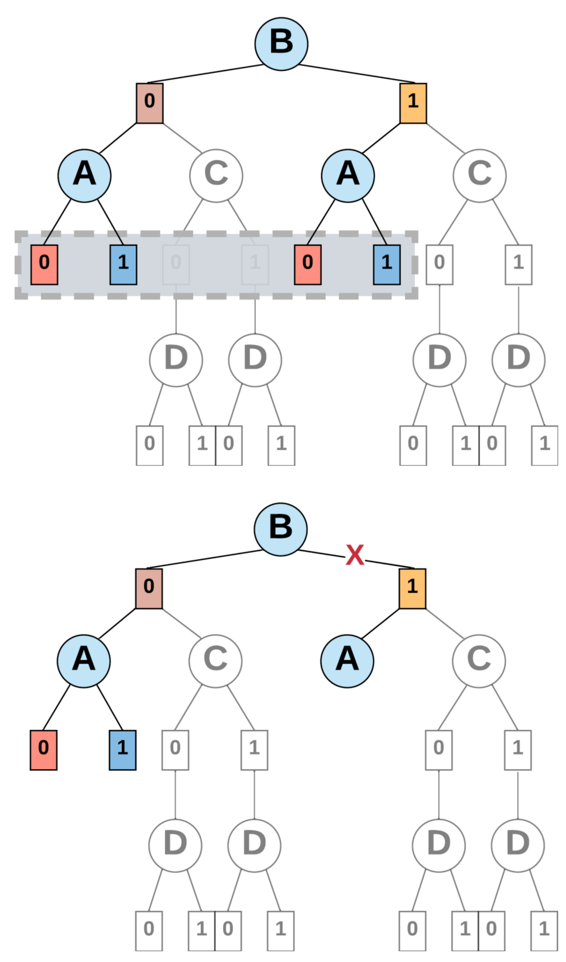
How should nodes be abstracted?

AND/OR Abstraction Sampling

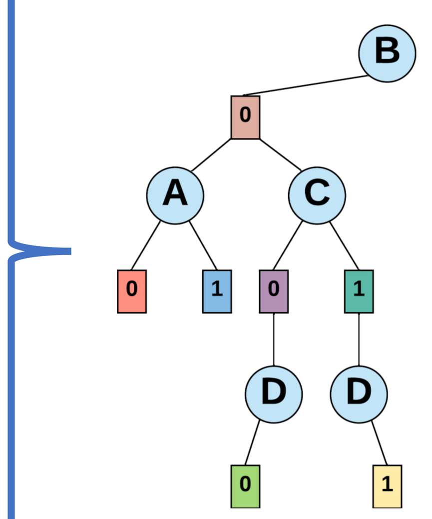
[Broka et al., 2018; Kask et al., 2020]



1. **Expansion:** Expand AND/OR search tree variable-by-variable following a DFS ordering of the guiding pseudo tree
2. **Form Abstract States:** Using an abstraction function, partition newly expanded nodes into an ***nAbs*** number of abstract states. (Prior state-of-the-art used graph-based abstraction functions).
3. **Select Representatives:** Using $p(n) \propto$ ***g(n) = w(n) h(n) r(n) g(n)***, stochastically select a representative from each abstract state and reweight it such that $w(n) \leftarrow w(n)/p(n)$
4. **Prune:** prune nodes not extending to valid configurations
5. **Backtrack:** When reaching a leaf, recursively backtrack until reaching an unexplored branch of the pseudo tree. Update parent node values while backtracking.
6. **Repeat:** Repeat until backtracking to the root node r



The result is a multi-configuration subtree (probe) that estimates Z: $\hat{Z} = w(r)\,\hat{Z}(r)$



$$\hat{Z} = \frac{1}{K}\sum_{k=1}^{K} Z'_k$$

Q-Based Abstraction Functions

Best performing variant of the newly introduced value-based abstraction schemes

Sort nodes based on *q(n) = w(n) h(n) r(n) g(n)*

- *The estimated contribution of a node towards the overall Z*
- *The unnormalized proposal measure*

Partition in an order-consistent manner…

Given $nAbs$ and a function $\mu : D_{\boldsymbol{X}} \to \mathbb{R}^+$, a partitioning function $\psi_\mu : D_{\boldsymbol{X}} \to \{A_1, A_2, \dots A_{nAbs}\}$, is order-consistent with μ relative to the $nAbs$ abstract states if for any $n_1 \in \boldsymbol{A_i}$ and $n_2 \in \boldsymbol{A_j}$, $i < j \Leftrightarrow \mu(n_1) \leq \mu(n_2)$.

Partitioning Scheme Examples

simpleVB: $1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 10, 100 \xrightarrow{simpleVB} \{1.0,\ 1.1\},\ \{1.2,\ 1.3\},\ \{1.4,\ 1.5\},\ \{10,\ 100\}$
(Evenly sort into equal cardinality abstract states)

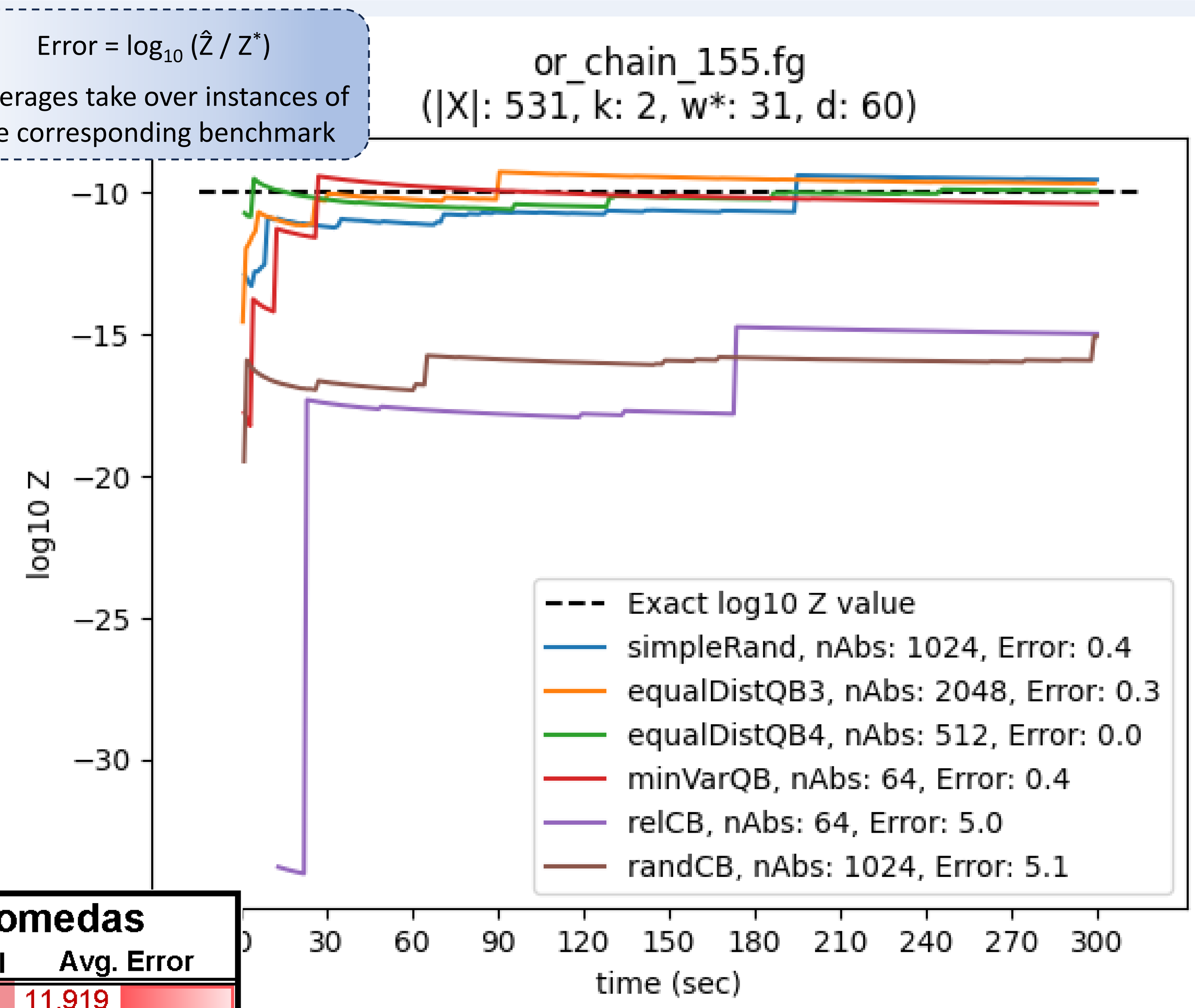
minVarVB: $1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 10, 100 \xrightarrow{minVarVB} \{1.0,\ 1.1,\ 1.2\},\ \{1.3,\ 1.4,\ 1.5\},\ \{10\},\ \{100\}$
(Minimize total with-in variance of the abstract states)

equalDistVB: $1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 10, 100 \xrightarrow{equalDistVB4} \{100\},\ \{10\},\ \{1.5,\ 1.4,\ 1.3\},\ \{1.2,\ 1.1,\ 1.0\}$
(Attempt to equalize total values across abstract states)

randVB: $1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 10, 100 \xrightarrow{randVB} \{100\},\ \{10,\ 1.5,\ 1.4,\ 1.3\},\ \{1.2,\ 1.1\},\ \{1.0\}$
(Partition nodes into randomly sized abstract states)

iB-5, t-300sec, Exact			DBN		Grids		Pedigree		Promedas	
Class	Scheme	nAbs	Fail	Avg. Error	Fail	Avg. Error	Fail	Avg. Error	Fail	Avg. Error
QB	minVar	4	0	1.684	0	3.622	0	1.434	2	2.518
		64	0	0.180	0	1.897	0	0.210	1	1.062
		1024	0	0.060	0	1.566	0	0.479	2	1.837
	equalDist3	4	0	1.594	0	5.861	0	1.668	1	1.804
		64	0	0.236	0	2.570	0	0.221	0	0.570
		1024	0	0.051	0	1.844	0	0.155	0	0.462
	equalDist4	4	0	1.371	0	5.988	0	1.648	1	1.678
		64	0	0.215	0	2.438	0	0.231	0	0.596
		1024	0	0.150	0	1.891	0	0.150	0	0.455
CTX	rand	4	0	1.381	0	5.030	0	1.852	7	4.643
		64	0	1.763	0	5.950	0	0.598	1	2.659
		1024	0	2.007	0	5.513	0	1.114	1	2.442
	rel	4	0	1.850	0	5.933	0	1.332	10	5.729
		64	0	3.510	0	4.021	0	0.424	6	4.349
		1024	0	5.086	0	5.136	0	1.041	15	6.688
RAND	rand	4	0	1.018	0	4.329	0	1.705	2	2.947
		64	0	0.418	0	2.094	0	0.212	0	0.757
		1024	0	0.120	0	1.501	0	0.143	0	0.513

← Error = $\log_{10}(\hat{Z} / Z^*)$
Averages take over instances of the corresponding benchmark



iB-10, t-1200sec, LARGE		DBN			Grids			Linkage-Type4			Promedas		
Class	Scheme	nAbs	Fail	Avg. Error	nAbs	Fail	Avg. Error	nAbs	Fail	Avg. Error	nAbs	Fail	Avg. Error
QB	simple	1	0	6.540	16	0	197.931	2048	13	48.681	4	34	11.919
	minVar	2048	0	1.837	1024	0	28.423	256	31	93.058	16	13	5.403
	equalDist	512	0	5.423	2048	0	118.547	2048	22	46.196	512	15	5.960
	equalDist2	2048	0	3.813	2048	0	91.994	1024	21	40.310	2048	12	4.982
	equalDist3	**2048**	**0**	**1.645**	**2048**	**0**	**19.277**	**1024**	**20**	**37.490**	**256**	**5**	**2.560**
	equalDist4	**2048**	**0**	**1.643**	**2048**	**0**	**18.866**	**2048**	**16**	**30.512**	**512**	**5**	**2.476**
	rand	4	0	6.292	16	0	163.973	256	17	156.992	4	28	11.532
CTX	rand	64	0	5.710	512	0	111.104	2048	53	194.741	256	0	3.222
	rel	1	0	6.267	1024	0	80.633	1024	37	129.189	16	34	11.247
RAND	**rand**	**2048**	**0**	**2.123**	**2048**	**0**	**19.053**	**1024**	**19**	**33.804**	**1024**	**10**	**3.936**

Summary: Experiments show the ***QB scheme*** with ***equalDist3*** or ***equalDistQB4***, and the ***RAND*** scheme, performing best. These schemes significantly outperform the former state-of-the-art CTX based abstractions and tend to improve as the abstraction granularity (***nAbs***) increases. Our study suggests use of these schemes with the largest ***nAbs*** feasible.