Abstraction Sampling in Graphical Models

Rina Dechter, Filjor Broka

Donald Bren school of ICS, University of California, Irvine
Background

- Knuth 1965, 1975: Estimate the search space of Backtracking search, Knuth’s paper / Simple Sampling (SS)

- Chen’s Heuristic Sampling (HS) / Generic Sampling (HS) – Generalizations of Knuth’s algorithm

- Both works focused on estimating size of search trees
Chen’s HS - Problem Formulation

- Let $T$ be a tree with nodes/states $S$
- Let $f$ be an arbitrary function on $S$
- Goal of heuristic sampling is to produce estimates of the following sum without traversing the entire tree

$$\phi \stackrel{\text{def}}{=} \sum_{s \in S} f(s)$$
Chen’s HS - Problem Formulation

- The value of $\phi$ is a property of $T$, as captured by the definition of $f$.
- If $f(s)$ is the cost of processing node $s$, then $\phi$ is the total cost of traversing $T$.
- If $f(s)$ is the number of children of $s$, then $\phi$ is the total number of edges in $T$.
- If $f$ is an indicator function for some property $A'$, then $\phi$ is the number of nodes satisfying $A'$. 
Chen’s HS - Stratifier

- Heuristic sampling requires a stratifier
- Stratifier: a heuristic function \( h : S \rightarrow P \)
- Maps each state \( s \) to stratum \( h(s) \) in a poset \( \mathcal{P} \)
- \( h \) strictly decreasing along each edge of \( \mathcal{T} \)
Chen’s HS – Algorithm idea

- Sample each stratum $\alpha$ for a representative $s_\alpha$
- Simultaneously obtain an estimate $w_\alpha = w(s_\alpha)$ for the number of nodes in the stratum
- Compute an unbiased estimate of $\phi$ as follows:

$$\hat{\phi} \overset{\text{def}}{=} \sum_{\alpha} w_\alpha f(s_\alpha).$$
Algorithm HS

input: root of a tree and a stratifier $h$;
output: set of sample nodes $S$ and corresponding weights $W$;
data structure: a queue $Q$ with elements $(s, w)$ indexed by $h(s)$;

begin
    $Q \leftarrow \{(\text{root, 1})\}$;
    while ($Q$ not empty) do
        output an element $(s, w)$ of $Q$ with maximal $h(s)$;
        for each child $t$ of $s$ do
            $\alpha \leftarrow h(t)$;
            if $Q$ contains an element $(s_\alpha, w_\alpha)$ in stratum $\alpha$ then do
                $w_\alpha \leftarrow w_\alpha + w$;
                with probability $w/w_\alpha$ do
                    $s_\alpha \leftarrow t$;
            else
                insert a new element $(t, w)$ into $Q$;
        end.
3.2. Adding importance sampling. We will show that $\hat{\varphi}$ remains unbiased when we alter the child selection probability by replacing the clause

$$w_\alpha \leftarrow w_\alpha + w;$$

with probability $w/w_\alpha$ do

$$s_\alpha \leftarrow t;$$

in heuristic sampling (HS) with

$$w_\alpha \leftarrow w_\alpha/(1 - p);$$

with probability $p$ do

$$s_\alpha \leftarrow t;$$

$$w_\alpha \leftarrow w/p;$$
Graphical Models – Two views

**Factored representation using local functions.**
A graphical model represents a global function over assignments (or configurations) over a set of discrete variables. It is defined by a collection of local functions, over subsets of variables, called scopes.

4-tuple \( M = (X, D, F, \prod) \)  
\( X = \{X_i : i \in V\} \)  
\( D = \{D_i : i \in V\} \)

Each function in \( \psi_\alpha \in F \) is defined over a subset of the variables called its scope, \( X_\alpha \), where \( \alpha \subseteq V \) is the indices of variables in its scope and \( D_\alpha \) denotes the Cartesian product of their domains. Namely, \( \psi_\alpha : D_\alpha \rightarrow R^{\geq 0} \).

A graphical model represents a global function, often a probability distribution, defined by
\[
Pr(X) \propto \prod_\alpha \psi_\alpha(X_\alpha)
\]

A popular task we will consider is to compute the normalizing constant, also known as a partition function
\[
Z = \sum_X \prod_\alpha \psi_\alpha(X_\alpha)
\]
Graphical Models – Two views

Weighted State Graph

- The states or nodes in the graph defined relative to an ordering of the variables, where each layer corresponds to one variable.
- Can be more compactly specified with an AND/OR state space graph
- We look at both OR and AND/OR graph representations
All Four Search Spaces

Full OR search tree
126 nodes

Context minimal OR search graph
28 nodes

Full AND/OR search tree
54 AND nodes

Context minimal AND/OR search graph
18 AND nodes

11/12/15
Weighted AND/OR Search Tree and Context Minimal Graph for Cost Networks

Figure 20: AND/OR search tree and context minimal graph

Figure 22: AOMDD for the weighted graph
Abstraction Sampling Idea

- Generalize Knuth-Chen’s methods to estimate for any recursive graph function (definition next slide)
- Current focus: partition function
- Nodes in state search space (tree or graph) partitioned through an abstraction function $a(s)$
- View nodes having the same value of abstraction function, as merged into one representative node in the style of decision diagrams
- Generalize to OR trees, AND/OR trees, AND/OR graphs
Recursive value functions

A value function $V_G(n)$ is recursive if for every node $n$ in the graph

$$V_G(n) = F[E(n), V_G(n_1), V_G(n_2), ..., V_G(n_b)]$$

Where $n_1, n_2, ..., n_b$ are child nodes of $n$.

$E(n)$ stands for a set of local properties characterizing the node $n$.

$F$ is an arbitrary combination function (Pearl book, page 57)

Examples: Expectation in influence diagrams
Abstraction Function

**Definition 3 (layered abstractions)** Given a weighted directed graph $S$, an abstraction function $\text{abs}$ is defined over $S$ $\text{abs} : S \rightarrow \{a_1, ..., a_r\}$. We require that if two states are at different depths in $S$ they map to different abstractions.

- Abstraction sampling: (one probe)
  
  1. go breadth-first from the root and sample a subset of nodes from each layer. One representative from each abstract state, yielding an explicit sampled graph, $S$
  
  2. Compute the function on $S$
Generalization to OR trees

- Two ways to generalize Knuth-Chen’s method to OR trees for partition function estimation

  - **Method 1:** Directly apply Heuristic Sampling by considering $f$ to be 0 in non-leaf nodes, and in leaf nodes it takes the value of the product of edge costs from the root to that particular leaf.

  - **Method 2:** Merge nodes having the same abstraction function value, generate a sample graph, compute estimated partition function.
Classic OR Search Space

Ordering: A B E C D F
OR Search Tree example

\[\text{dom}(A) = \{0, 1\}\]
\[\text{dom}(B) = \{0, 1\}\]
\[\text{dom}(C) = \{0, 1\}\]

<table>
<thead>
<tr>
<th>(P(A))</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(A = 0)</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>(A = 1)</td>
<td>0.6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(P(B \mid A))</th>
<th>(A = 0)</th>
<th>(A = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(B = 0)</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>(B = 1)</td>
<td>0.7</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(P(C \mid A))</th>
<th>(A = 0)</th>
<th>(A = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C = 0)</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>(C = 1)</td>
<td>0.8</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Method 1

Algorithm 3: Partition function estimation for weighted OR trees (Knuth-Chen style), a single probe

Input: root of a weighted OR tree $s^*$, abstraction function $a$.
Output: set of sampled nodes, corresponding weights and corresponding partial partition function $S$, estimated partition function $Z$.

1: $Q \leftarrow \{(s^*, 1, 1)\}$, $Q$ is a priority queue indexed by $a(s)$
2: $S \leftarrow \{\}$, $Z \leftarrow 0$
3: while $Q$ is not empty do
4:  $(s, w_s, z_s) = \text{dequeue}(Q)$
5:  $S \leftarrow S \cup \{(s, w_s, z_s)\}$
6:  if $s$ has no children then
7:    $Z \leftarrow Z + w_s \cdot z_s$
8:  for each child $t$ of $s$ do
9:    $\alpha \leftarrow a(t)$
10:   $z_t \leftarrow z_s \cdot c(s, t)$
11:   if $Q$ contains $(s_\alpha, w_\alpha, p_\alpha)$ s.t. $a(s_\alpha) = \alpha$ then
12:      $w_\alpha \leftarrow w_\alpha + w_s$
13:      with probability $w/w_\alpha$ do
14:         $s_\alpha = t$
15:         $z_\alpha = z_t$
16:      update $(s_\alpha, w_\alpha, z_\alpha)$ in $Q$
17:    else
18:      $Q \leftarrow Q \cup \{(t, w_s, z_t)\}$

Claim: This procedure gives an unbiased estimate for partition function. Shown by reduction to Knuth-Chen method.
Method 1 – OR Tree

\[ Z_{est} = 4 \times (0.6 \times 0.7 \times 0.8) + 4 \times (0.4 \times 0.1 \times 0.6) = 1.44 \]
Method 2

Claim: This procedure gives an **unbiased** estimate for partition function. To be shown formally.

---

**Algorithm 2: Improved Abstraction Sampling, AS1, a single probe**

**Require:** a weighted state space graph $\mathcal{S} = (X, D, G)$ over set of variables $X = \{X_1, ..., X_n\}$, defined implicitly. An abstraction function $\text{abs}$. A root $s_0$ of $\mathcal{S}$. $c(s, s')$ is the cost of the arc $(s, s')$.

**Ensure:** a sample explicit subgraph $\tilde{\mathcal{S}} = (N, E, C)$. Each node is a pair $<s, q>$. 

1. initialize $\text{OPEN} \leftarrow \langle s_0, 1 \rangle$, $\tilde{\mathcal{S}} \leftarrow \langle s_0, 1 \rangle$
2. while $\text{OPEN}$ is not empty do
3. $<s, q> \leftarrow \text{first in OPEN}$ where ordering is by depth. Remove it from OPEN
4. expand $s$, generating all its child nodes, denoted $\{s'\}$ and Place them at the end of OPEN.
5. for each child $s'$ of $s$ do
6. $q(s') \leftarrow q(s) \cdot c(s, s')$. 
7. $a \leftarrow a(s')$. (get the abstraction of $s'$)
8. if $\tilde{\mathcal{S}}$ contains a representative $s_a$ of abstraction $a$ then
9. $q(s_a) \leftarrow q(s_a) + q(s')$
10. with probability $q(s_a)$ replace $s_a$ by $s'$:
11. if $s'$ is selected then
12. update the graph and the $q$'s with the new representative:
13. $\text{olds}_a \leftarrow s_a, s_a \leftarrow s'$ ($s'$ is the new $s_a$)
14. $N \leftarrow N \cup \{s_a\} - \{\text{olds}_a\}$,
15. $E \leftarrow E \cup \{(s, s_a)\}, c(s, s_a) \leftarrow c(s, s')$. (add the new representative of $a$)
16. $E \leftarrow E \cup \{(\text{parent}(\text{olds}_a), s_a)\}, c(\text{parent}(\text{olds}_a), s_a) \leftarrow c(\text{parent}(\text{olds}_a), \text{olds}_a)$. (merge the old $\text{olds}_a$ into the new $s_a$
17. else
18. $E \leftarrow E \cup \{(s, s_a)\}$
19. $c(s, s_a) \leftarrow c(s, s')$
20. return $\tilde{\mathcal{S}} = (N, E, W)$, $N$ is the set of representative of abstractions and their $q$ values.
Method 2 – OR Tree

\[ Z_{\text{est}} = 0.756 + 0.244 = 1 \]
Imitation Tree

A = 0

B = 0
C = 0

A = 0
B = 1
C = 0

A = 0
B = 1
C = 1

A = 1
B = 0
C = 0

A = 1
B = 0
C = 1

A = 1
B = 1
C = 0

A = 1
B = 1
C = 1
Imitation Tree

A = 0

A = 1

B = 0

B = 1

C = 0

C = 1

0.6

0.4

0.3

0.7

0.1

0.9

0.4

0.6

0.2

0.8

0.4

0.6

0.2

0.8
Weighted AND/OR Tree for Bayesian Network

\[ P(E \mid A, B) \quad P(B \mid A) \quad P(C \mid A) \quad P(A) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>E=0</th>
<th>E=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.4</td>
<td>.6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.5</td>
<td>.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.7</td>
<td>.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.2</td>
<td>.8</td>
</tr>
</tbody>
</table>

\[
P(A) \quad P(B) \quad P(C) \quad P(D)\]

\[
P(D \mid B, C) \]

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D=0</th>
<th>D=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.2</td>
<td>.8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.3</td>
<td>.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.5</td>
<td>.5</td>
</tr>
</tbody>
</table>

Evidence: E=0

Evidence: D=1
Weighted AND/OR Tree for Bayesian Network
(Sum-Product Networks)

**Evidence: E=0**

- **AND node:** Combination operator (product)
- **OR node:** Marginalization operator (summation)
- Value of node = updated belief for sub-problem below

**Result:** $P(D=1, E=0)$

---

**Evidence: D=1**

- **AND node:** Combination operator (product)
- **OR node:** Marginalization operator (summation)
- Value of node = updated belief for sub-problem below
AND-OR Tree generalizations

**Pseudo-tree**

(dom(A) = {0, 1})
(dom(B) = {0, 1})
(dom(C) = {0, 1})
(dom(D) = {0, 1})
Full AND-OR Tree
Algorithm 4: Partition function estimation for weighted AND-OR trees (Knuth-Chen style), a single probe

Input: - root of a weighted AND-OR tree $s^*$, 
- abstraction function for AND-nodes $a$, 
- ordering function $ord$ of OR-nodes based on depth-first-search traversal of pseudo tree associated with underlying graphical model.

Output: - set of sampled OR-nodes, corresponding parent pointer, corresponding forward and backward weights, corresponding forward and backward partial partition function $SO$ (example element: $(s, par, wf, wb, zf, zb)$), 
- set of sampled AND-nodes, corresponding parent pointer, corresponding forward and backward weights, corresponding forward and backward partial partition function $SA$ (example element: $(s, par, wf, wb, zf, zb)$), 
- estimated partition function $Z$.

1: $QO \leftarrow \{(s^*, \emptyset, 1.1.1.1)\}$. $QO$ is a priority queue indexed by $ord(s)$
2: $QA \leftarrow \{\}$, $QA$ is a priority queue indexed by $h(s)$
3: $SA \leftarrow \{\}$, $SO \leftarrow \{\}$, $Z \leftarrow 0$, curOrd $= ord(s^*)$
4: while hasNext($QO$, curOrd) or $QA$ is not empty do
5:   while hasNext($QO$, curOrd) do
6:     $(s, par_s, wf_s, wb_s, zf_s, zb_s) \leftarrow$ dequeue($QO$)
7:     $s' \leftarrow par_s$, $wb_s = zf_s$, $w_f_s = wb_s$, $z_f_s = zb_s$
8:     $SO \leftarrow SO \cup \{(s, par_s, wf_s, wb_s, zf_s, zb_s)\}$
9:     for each child $t$ of $s$ do
10:        $\alpha \leftarrow a(t)$
11:        $par_t = s$, $wb_t = wb_s$, $w_f_t = wf_s$, $z_f_t = zb_s \cdot c(s, t)$, $z_f_t = zb_t$
12:        if $QA$ contains $(s_a, par_a, wf_a, wb_a, zf_a, zb_a)$ s.t. $a(s_a) = \alpha$ then
13:           $wb_a \leftarrow wb_a + wb_s$
14:           with probability $wb_a/wb_a$
15:              $s_a = t$, $par_a = par_t$, $wf_a = wb_a$, $zf_a = zf_t$, $zb_a = zb_t$
16:           update $(s_a, w_a)$ in $QA$
17:        else
18:           $QA \leftarrow QA \cup \{(t, par_t, wf_t, wb_t, zf_t, zb_t)\}$
19:   while $QA$ is not empty do
20:      $(s, par_s, wf_s, wb_s, zf_s, zb_s) \leftarrow$ dequeue($QA$)
21:      $s' \leftarrow par_s$, $zb_s = zb_s \cdot c(s', s)$, $wb_s/wb_a$, $zf_s = zb_s$
22:      $SA \leftarrow SA \cup \{(s, par_s, wf_s, wb_s, zf_s, zb_s)\}$
23:      if $ord(s')$ is maximal (i.e. last variable visited) then
24:         $Z \leftarrow Z + zb_s$
25:      if $s$ has no children (i.e. leaf in AND-OR tree) then
26:         propagate($s$) (TBD: updates $wf$ and $zf$ on the path from root to $s$)
27:      for each child $t$ of $s$ do
28:         $par_t = s$, $wb_t = wb_s$, $w_f_t = wf_s$, $z_f_t = zb_s$, $z_f_t = zf_s$
29:         $QO \leftarrow QO \cup \{(t, par_t, wf_t, wb_t, zf_t, zb_t)\}$

Method 1 – AND/OR tree
Method 1 – AND/OR tree
Method 2 – AND/OR tree

Algorithm 3: AO-AS, a single probe

Require: a weighted AND/OR state space graph $S = (X, D, G)$ over set of variables $X = \{X_1, ..., X_n\}$, defined implicitly using a pseudo-tree $T$. An AND/OR graph data structure $G'$ initialized to $s_0$ that will output $\tilde{S}$. An abstraction function $abs$ for AND nodes only. A root $s_0$ of $S$, $c(s, s')$ is the cost of the arc $(s, s')$.

Ensure: a sample explicit subgraph $\tilde{S} = (N, E, C)$. Each node is a pair $< s, q >$. $q$ is the value of partial paths ended at $s$.

1: initialize OPEN $\leftarrow < s_0, 1 >$, $\tilde{S} \leftarrow < s_0, 1 >$
2: while OPEN is not empty do
3: $< s, q > \leftarrow$ first AND node in OPEN where ordering is by smallest depth, first.
4: Remove it from OPEN (we can keep only AND nodes in OPEN)
5: Expand $s$, generating all its OR child nodes, denoted by the set $A$.
6: for each OR node $X \in A$ do
7: Expand $X$ generating all its AND child nodes, denoted $\{s'\}$ and place them at the end of OPEN.
8: for each child $s'$ of $s$ do
9: $q(s') \leftarrow q(s) \cdot c(s, s')$.
10: $a \leftarrow a(s')$. (get the abstraction of $s'$)
11: if $\tilde{S}$ contains a representative $s_a$ of abstraction $a$ then
12: $q(s_a) \leftarrow q(s_a) + q(s')$
13: with probability $\frac{q(s')}{q(s_a)}$ replace $s_a$ by $s'$.
14: if $s'$ is selected then
15: update the graph $\tilde{S}$ and the $q$'s with the new representative:
16: old$s_a \leftarrow s_a$, $s_a \leftarrow s'$ ($s'$ is the new $s_a$)
17: $N \leftarrow N \cup \{s_a\} - \{old{s_a}\}$,
18: $E \leftarrow E \cup \{(s, s_a)\}$, $c(s, s_a) \leftarrow c(s, s')$. (add the new representative of $a$)
19: $E \leftarrow E \cup \{(parent(old{s_a}), s_a)\}$,
20: $c(parent(old{s_a}), s_a) \leftarrow c(parent(old{s_a}), old{s_a})$. (merge the old old$s_a$ into the new $s_a$)
21: else
22: $E \leftarrow E \cup \{(s, s_a)\}$
23: $c(s, s_a) \leftarrow c(s, s')$
24: return $\tilde{S} = (N, E, W)$, $N$ is the set of representatives of abstractions and their $q$ values.
Method 2 – AND/OR tree
Exploring Abstraction Functions

Pruned context abstractions

**Definition 4 (Pruned context abstractions)** Given a context $C_p$ of $X_p$, an abstraction at $X_p$ is called pruned-context if it is based on a subset $S \subseteq C_p$. A subset of a context, $S$, defines a collection of abstractions denoted $A_{(S,p)}$, for each $\bar{s}$:

$$A_{(S,p)}(\bar{s}) = \{ \bar{x}_p \mid \pi_S(\bar{x}_p) = \bar{s} \}$$

A 0-level abstraction is when $S_p = \{X_p\}$
The End

- Thanks you.
- How to design good abstraction functions