Boosting AND/OR-Based Computational Protein Design: Dynamic Heuristics and Generalizable UFO (Extended Background)

Bobak Pezeshki[1] **Radu Marinescu**[2] **Alexander Ihler**[1] **Rina Dechter**[1]

[1]University of California, Irvine
[2]IBM Research

Abstract

In addition to providing a glossary of terms, abbreviations, and notation, this document aims to provide readers with background on topics that are foundational to the concepts that are discussed in the main paper. The most up-to-date version of this document - as well as other supplemental materials - can be found on the Dechter Lab publications page.

CONTENTS

i. Glossary **3**

ii. Abbreviations **6**

iii. Notation **7**

iv. Algorithms **9**

1 Background: Graphical Models **10**

1.1 Discrete Graphical Models . . . 10
1.2 Notation . . . 10
1.3 Primal Graph . . . 10
1.4 Simple Example . . . 11
1.5 Some Well-Known and Important Graphical Model Tasks . . . 12
1.5.1 Definitions . . . 12
1.5.2 Difficulty . . . 12
1.6 Variable Elimination . . . 13
1.7 Bucket Elimination . . . 14
1.8 Induced Width (w*) . . . 15
1.9 Pseudo Tree . . . 15

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

GLOSSARY

τ-underflow
See **τ-underflowed function** and glstau-underflowed-model.

τ-underflowed function
(f_τ) A function such that output values less than τ are replaced with 0.0.

$$f_{\alpha_\tau}(\boldsymbol{a} \in D_\alpha) = \begin{cases} f_\alpha(\boldsymbol{a}), & f_\alpha(\boldsymbol{a}) \geq \tau \\ 0, & \text{otherwise.} \end{cases}$$

, 3, 4, 5, 8

τ-underflowed model
($\mathcal{M}_\tau = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}_\tau \rangle$) A model with each of its functions replaced with the corresponding **τ-underflowed function**.

bucket
Data structure used in **bucket elimination**, each corresponding to a particular variable to be eliminated, that collects functions to be processed during an elimination step. Processing of a bucket involves applying an elimination operator (such as maximization or marginalization) over the bucket's variable to the combined functions of the bucket. , 14

bucket tree
In the context of **bucket elimination**: A directed tree with nodes corresponding to buckets and directed edges corresponding to the origin and destination of computed messages. , 15

bucket message
The function resulting after processing of a bucket during a **bucket elimination** elimination step. , 14

bucket elimination
A variable elimination framework that uses a dynamic programming approach leveraging commutability of mathematical expressions. , 3, 14

computational protein design
The process of using computational methods and algorithms to create novel protein sequences or structures with desired properties. , 6

configuration
A joint assignment to a set of variables. (See **full configuration** and **partial configuration** for more specific uses of the term). , 10

conformation
The three-dimensional orientation of a molecule.

constrained ordering
An elmination ordering that satisfies given constraints for some variables to be eliminated before others. , 13

***de novo* protein design**
The process of creating novel proteins in order to achieve desired criteria without relying on existing protein templates, using computational or experimental approaches.

elimination order
Order in which to process and eliminate variables during variable elimination inference. , 13

full configuration
A joint assignment to all the variables of a graphical model. , 3, 10

graphical model
($\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$). Mathematical tool for modeling complex systems composed of a set of variables $\boldsymbol{X}$, a set of domains $\boldsymbol{D} = \{D_X | X \in \boldsymbol{X}\}$ for each variable X, and a set of functions $\boldsymbol{F}$ with each function defined over a subset of the model's variables $\alpha \subseteq \boldsymbol{X}$. , 7, 8, 10

induced primal graph

With respect to a variable elimination procedure: the primal graph of a graphical model augmented with additional edges corresponding to the scope of messages created through the variable elimination procedure. , 4

induced graph

Induced primal graph.

induced width

(w^*) One less than the largest clique size of the **induced primal graph**. , 15

marginal maximum a-posteriori

(MMAP). The marginal likelihood associated with the configuration of a target subset of variables $\boldsymbol{Q}$ that maximizes their marginal likelihood.
In the context of discrete graphical models without evidence, with $\boldsymbol{Q} \subset \boldsymbol{X}$, $\boldsymbol{S} = \boldsymbol{X} \setminus \boldsymbol{Q}$ be the variables to sum over, $\boldsymbol{F_Q} = \{f_\alpha \mid \alpha \subseteq \boldsymbol{Q}\}$ be the set of functions defined only over $\alpha \in \boldsymbol{Q}$, and $\boldsymbol{F_S} = \boldsymbol{F} \setminus \boldsymbol{F_Q}$ be functions that include some $X \in \boldsymbol{S}$ in their scope,

$$MMAP = \max_{\boldsymbol{Q}} \sum_{\boldsymbol{S}} \prod_{f_\alpha \in \boldsymbol{F}} f_\alpha(\boldsymbol{q} \cup \boldsymbol{s}) \tag{1}$$

$$= \max_{\boldsymbol{Q}} \prod_{f'_\alpha \in \boldsymbol{F_S}} f'_\alpha(\boldsymbol{q}) \sum_{\boldsymbol{S}} \prod_{f''_\alpha \in \boldsymbol{F_S}} f''_\alpha(\boldsymbol{q} \cup \boldsymbol{s}) \tag{2}$$

, 6, 7, 8, 10, 12

maximum a-posteriori

(MAP). With respect to a graphical model, the likelihood value associated with the **most probable explanation** or **MPE**.
In the context of discrete graphical models without evidence, $MAP = \max_{\boldsymbol{Q}=\boldsymbol{X}} \prod_{f_\alpha \in \boldsymbol{F}} f_\alpha(\boldsymbol{q})$. , 6, 7, 8, 10, 12

most probable explanation

(MPE). Assignment to variables the variables of a graphical model that maximizes the conditional probability of the observed evidence.
In the context of discrete graphical models without any evidence, $MPE = \operatorname{argmax}_{\boldsymbol{Q}=\boldsymbol{X}} \prod_{f_\alpha \in \boldsymbol{F}} f_\alpha(\boldsymbol{q})$. , 4, 6, 10, 12

OSPREY

(Open Source Protein Engineering Software). A computational tool for protein design developed by the Bruce Donald Lab at Duke University, first released in 2010. [Hallen et al., 2018]

partial configuration

A joint assignment to a subset of the variables of a graphical model. , 3, 7, 10

partition function

(Z). A mathematical quantity that characterizes the distribution among a system's possible states and serves as a normalizing constant for calculating probabilistic measures associated with these states.
In the context of discrete graphical models, $Z = \sum_{\boldsymbol{X}} \prod_{f_\alpha \in \boldsymbol{F}} f_\alpha(\boldsymbol{x})$. , 6, 8, 10, 12

primal graph

The primal graph of a graphical model captures an underlying structure of the model, where each node corresponds to a random variable of the model and edges connect any two variables that are both part of a function's scope thus indicating direct dependencies between variables. , 10

protein re-design

The process of modifying existing protein sequences or structures to enhance or alter their function, stability, or other desired properties using computational or experimental approaches.

scope

The set of variables a function is defined over. Denoted as α in the general case. , 10

thresholded-underflow

See **τ-underflowed function** and glstau-underflowed-model.

underflow-threshold

(τ) A value $\tau \in (0, \infty)$ used to underflow functions (see τ**-underflowed function** and glstau-underflowed-model). , 8

variable elimination

(VE). An inference technique that involves an ordered computational processing of variables, each elimination step removing the processed variable from the resulting expression. , 6, 13

ABBREVIATIONS

CPD
: **Computational protein design**.

MAP
: **Maximum a-posteriori**. , 12

MMAP
: **Marginal maximum a-posteriori**. , 12

MPE
: **Most probable explanation**. , 4, 12

VE
: **Variable elimination**.

Z
: **Partition function**.

NOTATION

[capital letters] (ex. X)

Represents a variable of a **graphical model**.

[lower-case letters] (ex. x)

Represent assignments to variable corresponding to their capitalized form. For example, x represents a particular assignment to the variable represented by X, or $X \leftarrow x$.

[bold-faced capital letters] (ex. $\boldsymbol{X}$)

A set of variables of a **graphical model**. ($\boldsymbol{X}$, in particular, often refers to the set of all variables of a graphical model).

X

An abitrary variable of a **graphical model** (often indexed for more specificity). , 7

$\boldsymbol{X}$

The set of all variables X of a **graphical model**. , 7

$\boldsymbol{Q}$

In the context of the **maximum a-posteriori** or **marginal maximum a-posteriori** task, the [sub]set of the variables $\boldsymbol{Q}$ that are to be maximized over (know as "query" or "MAP" variables).

$\boldsymbol{S}$

In the context of the **marginal maximum a-posteriori** task, the subset of variables to be summed over. $\boldsymbol{S} = \boldsymbol{X} \setminus \boldsymbol{Q}\}$

x

An assignment to corresponding variable X.
$X \leftarrow x \in D_X$. , 7

$\boldsymbol{x}$

A full configuration, ie. assignment to all variables $\boldsymbol{X}$ of a **graphical model**.
$\boldsymbol{X} \leftarrow \boldsymbol{x} \in D_{\boldsymbol{X}}$.

α

Used to describe an arbitrary set of variables that makes up a function's scope (ie. the variables the function is defined over). , 7

D_X

The domain of variable X; ie. the set of all possible assignments x to X.

D_α

The set of all possible **partial configurations** to the variables of the set α. D_α is the Cartesian product of the domains of the variables in α. $D_\alpha = \bigotimes_{\{D_X | X \in \alpha\}} D_X$, where $\bigotimes$ is the Cartesian product operator.

$D_{\boldsymbol{Y}}$

The set of all possible **partial configurations** to the variables of the subset $\boldsymbol{Y} \subset \boldsymbol{X}$. $D_{\boldsymbol{Y}}$ is the Cartesian product of the domains of the variables in $\boldsymbol{Y}$. $D_{\boldsymbol{Y}} = \bigotimes_{\{D_Y | Y \in \boldsymbol{Y}\}} D_Y$, where $\bigotimes$ is the Cartesian product operator.

$D_{\boldsymbol{X}}$

The set of all possible partial configurations to the variables of the set $\boldsymbol{X}$. $D_{\boldsymbol{X}}$ is the Cartesian product of the domains of the variables in $\boldsymbol{X}$. $D_{\boldsymbol{X}} = \bigotimes_{\{D_X | X \in \boldsymbol{X}\}} D_X$, where $\bigotimes$ is the Cartesian product operator.

$\boldsymbol{D}$

The set of domains for every variable of a **graphical model**.

f_α

A function defined over the variables in the set α.
$f_\alpha : D_\alpha \rightarrow \mathcal{R}_{\geq 0}$, 7

f

A function f_α, but with its scope α omitted for simplicity.

v^*_f

The maximum output of f. $v^*_f = max_{\boldsymbol{X}} f(\boldsymbol{x})$

$\boldsymbol{F}$

The set of all functions of a **graphical model**.

$v^*_{\boldsymbol{F}}$

The maximum output of any function $f \in \boldsymbol{F}$. $v^*_{\boldsymbol{F}} = max_{f \in \boldsymbol{F}} max_{\boldsymbol{X}} f(\boldsymbol{x})$

τ

An **underflow-threshold** value $\tau \in (0, \infty)$.

f_τ

A function such that output values less than τ are replaced with 0.0. A τ**-underflowed function**.

$\boldsymbol{F}_\tau$

$\boldsymbol{F}_\tau = \{f_\tau \mid f \in \boldsymbol{F}\}$

$\boldsymbol{F}_{\boldsymbol{Q}}$

In the context of the **maximum a-posteriori** or **marginal maximum a-posteriori** task, the subset of functions defined only on a subset of the variables $\boldsymbol{Q}$ that are to be maximized over (know as "query" or "MAP" variables). $\boldsymbol{F}_{\boldsymbol{Q}} = \{f_\alpha \mid \alpha \subseteq \boldsymbol{Q}\}$

$\boldsymbol{F}_{\boldsymbol{S}}$

In the context of the **marginal maximum a-posteriori** task, $\boldsymbol{F}_{\boldsymbol{S}} = \boldsymbol{F} \setminus \boldsymbol{F}_{\boldsymbol{Q}} = \{f_\alpha \mid \exists X \in \alpha s.t. X \notin \boldsymbol{Q}\}$

$\boldsymbol{F}_{\boldsymbol{Q}_\tau}$

$\boldsymbol{F}_{\boldsymbol{Q}_\tau} = \{f_\tau \mid f \in \boldsymbol{F}_{\boldsymbol{Q}}\}$

$\boldsymbol{F}_{\boldsymbol{S}_\tau}$

$\boldsymbol{F}_{\boldsymbol{S}_\tau} = \{f_\tau \mid f \in \boldsymbol{F}_{\boldsymbol{S}}\}$

$\mathcal{M}$

Graphical model $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$ with non-negative functions.

$\mathcal{M}_\tau$

An altered **Graphical model** with non-negative functions such that each original function is replaced by its corresponding τ**-underflowed function**. $\mathcal{M}_\tau = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}_\tau \rangle$.

$\mathcal{M}_{\boldsymbol{S}_\tau}$

$\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}_{\boldsymbol{Q}} \cup \boldsymbol{F}_{\boldsymbol{S}_\tau} \rangle$

$MMAP(\mathcal{M}, \boldsymbol{Q})$

The **marginal maximum a-posteriori** of graphical model $\mathcal{M}$ maximizing over the subset of variables $\boldsymbol{Q}$.

$Z(\mathcal{M}, \boldsymbol{e})$

The **partition function** of graphical model $\mathcal{M}$ conditioned on an evidence assignment $\boldsymbol{E} \leftarrow \boldsymbol{e}$. $Z(\mathcal{M}, \boldsymbol{e}) = \sum_{\boldsymbol{y} \in D_{\boldsymbol{X} \setminus \boldsymbol{E}}} \prod_{f_\alpha} f_\alpha(\boldsymbol{y}, \boldsymbol{e})$

ALGORITHMS

BBK*

(Branch and Bound Over K*). A best-first search algorithm for protein re-design found in the computational protein design software OSPREY.

1 BACKGROUND: GRAPHICAL MODELS

Graphical models, such as a Bayesian or a Markov networks Pearl [1988], Darwiche [2009], Dechter [2013], are mathematical tools for modeling complex systems, each composed of a set of variables with defined domains and functions defined over subsets of the variables. The functions capture local dependencies of the subset of variables they are defined on, those variables known as the function's **scope**. The functions of a graphical model often represent a factorization of a global function over all the variables. An assignment to all of the variables (referred to as a **full configuration**) represents a possible state of the modeled system.

Graphical models are constructed not only to model a system, but also to provide a means of efficiently answering specific queries of interest via exploitation of the model's structure. Some common computational tasks are

- determination of the **partition function**: a normalization constant necessary for computing probabilistic quantities.
- determination of the **MPE** (**most probable explanation**): the most probable full configuration given a **partial configuration** (assignments to a subset of the variables) known as observation or evidence. Additionally, the associated likelihood corresponding to the MPE, known as the **MAP** (**maximum a-posteriori**) value, and can also be queried in kind.
- determination of the **MMAP** (**marginal maximum a-posteriori**) configuration: the configuration of a target subset of variables that maximizes their marginal likelihood.

(More details about common graphical model queries to be provided in Section 1.5: Well Known Graphical Model Tasks).

1.1 DISCRETE GRAPHICAL MODELS

Considering the discrete space, a discrete graphical model can be defined as a 3-tuple $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$, where

- $\mathbf{X}$ is a set of variables for which the model is defined over
- $\mathbf{D} = \{D_X : X \in \mathbf{X}\}$ is a set of finite domains, one for each $X \in \mathbf{X}$, defining the possible values each X can be assigned
- Each $f_\alpha \in \mathbf{F}$ (sometimes denoted $f \in \mathbf{F}$ for simplicity) is a real-valued function defined over a subset of the model's variables $\alpha \subseteq \mathbf{X}$, known as the function's **scope**, for which the function defines local interactions. More concretely, if we let D_α denote the Cartesian product of the domains of the variables in α, then $f_\alpha : D_\alpha \rightarrow R_{\geq 0}$. These functions can be expressed as tables for which there is a real valued output associated with every possible input $d_\alpha \in D_\alpha$ (ie. every possible joint assignment - or **configuration** - to all of the variables in α).

1.2 NOTATION

Capital letters (X) represent variables and small letters (x) represent their values. Boldfaced capital letters ($\mathbf{X}$) denote a collection of variables, $|\mathbf{X}|$ its cardinality, $D_{\boldsymbol{X}}$ their joint domains, and $\boldsymbol{x}$ a particular realization in that joint domain. Abusing notation, operations denoted $\bigoplus_{\boldsymbol{X}}$ (ex. $\sum_{\boldsymbol{X}}$) imply...

$$\begin{aligned} \bigoplus_{\boldsymbol{X}} &\Longrightarrow \bigoplus_{\boldsymbol{x} \in D_{\boldsymbol{X}}} \\ &\Longrightarrow \bigoplus_{x_1 \in D_{X_1}} \bigoplus_{x_2 \in D_{X_2}} \cdots \bigoplus_{x_{|\boldsymbol{X}|} \in D_{X_{|\boldsymbol{X}|}}} \end{aligned} \tag{3}$$

Furthermore, given a function f_α with scope α, a super-set of variables β s.t. $\alpha \subseteq \beta$, a particular configuration $\mathbf{b}$ of β, and $\mathbf{a} := \{X \leftarrow x | X \leftarrow x \in \mathbf{b} \text{ and } X \in \alpha\}$ (ie. the subset of assignments in $\mathbf{b}$ corresponding to variables in α),

$$f_\alpha(\mathbf{b}) \implies f_\alpha(\mathbf{a}) \tag{4}$$

1.3 PRIMAL GRAPH

A **primal graph** $\mathcal{G} = \langle \mathbf{X}, \mathbf{E} \rangle$ of a graphical model $\mathcal{M}$ associates each variable of $\mathcal{M}$ with a corresponding node in a one-to-one fashion such that arcs $e \in \mathbf{E}$ connect nodes whose variables appear in the scope of the same local function. To simplify, we abuse notation by using the same symbols to refer to primal graph nodes as their corresponding variables in $\mathcal{M}$. (For those familiar, note that the primal graph corresponds to the Markov Random Field graph representation of the model). The primal graph is a useful tool for graphical model algorithms' exploitation of the model's local structure.

1.4 SIMPLE EXAMPLE

Consider a simple model that relates temperature and humidity to the chance of rain, and temperature and elevation to the chance of different oxygen levels. Let us choose binary variables $\mathbf{X} = \{T, H, R, E, O\}$ to represent these different levels and construct a corresponding graphical model $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$ where

- T has corresponding domain $D_T = \{high, low\}$ representing high and low temperature
- H has corresponding domain $D_H = \{high, low\}$ representing high and low humidity
- R has corresponding domain $D_R = \{yes, no\}$ representing the presence or absence of rain
- E has corresponding domain $D_E = \{high, low\}$ representing the high or low elevation
- O has corresponding domain $D_O = \{high, low\}$ representing the high or low oxygen levels

and having five functions

- $f_T(T)$ representing the marginal probability of the temperature being high or low, $p(T)$
- $f_H(H)$ representing the marginal probability of the humidity being high or low, $p(H)$
- $f_E(E)$ representing the marginal probability of the elevation being high or low, $p(E)$
- $f_{T,H,R}(T, H, R)$ representing the conditional probability of rain given levels of humidity and temperature, $p(R \,|\, T, H)$
- $f_{T,E,O}(T, E, O)$ representing the conditional probability of high vs. low oxygen concentrations given the temperature and elevation levels, $p(O \,|\, T, E)$

defined by the following tables, respectively:

T	$p(T)$
$high$	0.40
low	0.60

H	$p(H)$
$high$	0.25
low	0.75

E	$p(E)$
$high$	0.20
low	0.80

T	H	R	$p(R \,|\, T, H)$
$high$	$high$	yes	0.40
$high$	$high$	no	0.60
$high$	low	yes	0.05
$high$	low	no	0.95
low	$high$	yes	0.80
low	$high$	no	0.20
low	low	yes	0.10
low	low	no	0.90

T	E	O	$p(O \,|\, T, E)$
$high$	$high$	$high$	0.20
$high$	$high$	low	0.80
$high$	low	$high$	0.40
$high$	low	low	0.60
low	$high$	$high$	0.25
low	$high$	low	0.75
low	low	$high$	0.70
low	low	low	0.30

and where we make independence assumptions allowing the joint distribution $P(T, H, R, E, O)$ to factorize to the probability functions represented by the model such that:

$$\begin{aligned} p(T, H, E, R, O) &= p(T) \cdot p(H \,|\, T) \cdot p(E \,|\, T, H) \cdot p(R \,|\, T, H, E) \cdot p(O \,|\, T, H, E, R) \\ &= p(T) \cdot p(H) \cdot p(E) \cdot p(R \,|\, T, H) \cdot p(O \,|\, T, E) \\ &= f_T(T) \cdot f_H(H) \cdot f_E(E) \cdot f_{T,H,R}(T, H, R) \cdot f_{T,E,O}(T, E, O) \\ &= \prod_{f_\alpha \in \mathbf{F}} f_\alpha(\alpha) \end{aligned} \tag{5}$$

with primal graph:

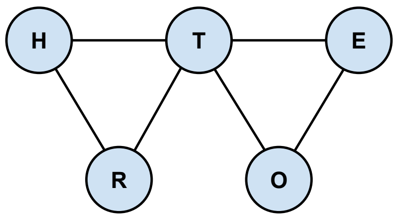


Figure 1: Primal graph of the example model described above.

You can see that the graph consists of nodes representing T, H, R, E, and O and has edges between each pair of $\{T, H, R\}$ since each pair appears together in at least one $f_\alpha \in \mathbf{F}$, and similarly between each pair of $\{T, E, O\}$.

With the model and primal graph defined, we can then use a variety of algorithms over graphical models to efficiently answer queries about the model. One such query could be to find the probability corresponding to the mode of our modeled distribution - namely to find the probability associated with the most likely full configuration. Throughout the next several sections, we will describe various queries and computational schemes commonly used with graphical models starting next by describing the general framework of variable elimination, which we will use to show one way to compute our example query.

1.5 SOME WELL-KNOWN AND IMPORTANT GRAPHICAL MODEL TASKS

There are a plethora of queries that a graphical model can lend itself to answering. In the context of our work in computational protein *re*-design, we will describe four related tasks in particular: determination of the

- **Partition function (Z)**
- **Maximum a-posteriori (MAP)**
- **Most probable explanation (MPE)**
- **Marginal maximum a-posteriori (MMAP)**

1.5.1 Definitions

Definition 1.5.1.1 below provides the formalization of these tasks respectively.

Definition 1.5.1.1 (Z, MAP, MPE, and MMAP)**.** *Given a graphical model* $\mathcal{M} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F})$,

$$Z = \sum_{\boldsymbol{X}} \prod_{\boldsymbol{F}} f(\boldsymbol{x}); \tag{6}$$

$$MAP = \max_{\boldsymbol{X}} \prod_{\boldsymbol{F}} f(\boldsymbol{x}); \tag{7}$$

$$MPE = \operatorname*{argmax}_{\boldsymbol{X}} \prod_{\boldsymbol{F}} f(\boldsymbol{x}); \tag{8}$$

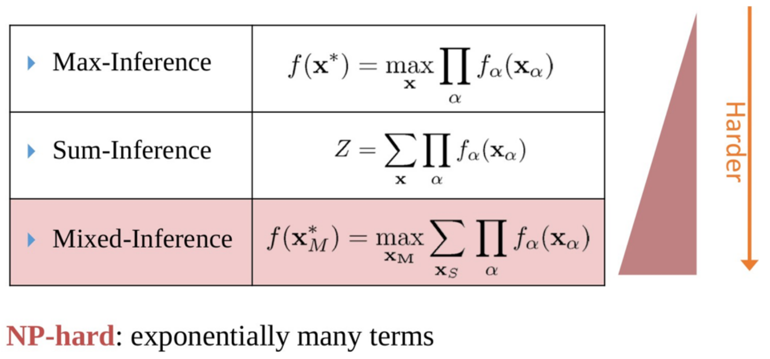
$$MMAP = \max_{\boldsymbol{Q} \subset \boldsymbol{X}} \sum_{\boldsymbol{S} = \boldsymbol{X} \setminus \boldsymbol{Q}} \prod_{\boldsymbol{F}} f(\boldsymbol{q} \cup \boldsymbol{s}) \tag{9}$$

The **partition function**, **Z**, is mathematical quantity that characterizes the distribution among a system's possible states. It is often used as a normalization constant for computing probabilities. **MPE** is a full configuration that maximizes the value of the model defined as the product of all of its functions, and **MAP** outputs that value. From a probabilistic model standpoint, this corresponds to finding the assignment to the variables that are most likely under the model, and the corresponding likelihood value, respectively. Given evidence (ie. a given assignment to a subset of the variables), the MPE (constrained to be consistent with the evidence) corresponds to finding the assignment to the rest of the variables that makes the evidence most likely to occur (thus the name "most probable explanation"). **MMAP** is similar to MAP with the exception that the model value is now defined with respect to the marginalization of a subset of the variables denoted as the "sum" or $\boldsymbol{S}$ variables, and so the maximization is with respect to the remaining set denoted as the "query" or $\boldsymbol{Q}$ variables. Although not commonly referred to, and thus omitted here, there can also be a corresponding MMPE task.

1.5.2 Difficulty

Summation tasks such as computing the partition function require consideration of the entire state-space (exponential in the number of variables) to compute accurately and are generally #P-hard. Since summation operations commute freely, when variable elimination algorithms are used for these tasks (with the variable ordering corresponding to the order in which variables are summed over in Equation 6), they can be used with any variable ordering. Pure homogeneous optimization such as MAP and MPE inference, whose solutions can be confirmed in polynomial time, are easier but still NP-Hard to compute. Since optimization operators can freely commute with others of their own kind (ex. max operators can commute with each other, or min operators can commute with each other), variable elimination for these tasks can also use of any variable ordering. Mixed inference tasks, however, are often more difficult to compute as they involve operators that do not commute. In the example of MMAP (Equation 9), the summation operations must be computed before maximization, and thus restricts the variable orderings that can be used. In practice, this **constrained ordering** can lead to inference over graphs of much greater widths (see Section 1.8 for more details) and thus are more difficult to compute.

This hierarchy of difficulties is summarized in Figure 2.



Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_\mathrm{M}} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$

- **NP-hard**: exponentially many terms

Figure 2: Hierarchy of difficulties for three classes of graphical model inference tasks.

1.6 VARIABLE ELIMINATION

Many PGM queries can be solved by an inference framework known as **variable elimination** (**VE**). Variable elimination involves an ordered computational processing of the variables of a model, at each step removing the processed variable from further computations (thus called an *elimination* step). Each elimination step corresponds to inference, transferring the effects of the eliminated variables over to the remaining variables (in practice, done by creating a newly inferred function over the remaining variables).

As an example, consider the query to find the mode of the distribution defined by our simple example above (Section 1.4). Formalizing this query, we want to solve the task:

$$\max_{T,H,E,R,O} p(t,h,e,r,o) \tag{10}$$

which, based on Equations 5, in terms of our model is equivalent to

$$\max_{T,H,E,R,O} p(t,h,e,r,o) = \max_{T,H,E,R,O}(\ f_T(t)\cdot f_H(h)\cdot f_E(e)\cdot f_{T,H,R}(t,h,r)\cdot f_{T,E,O}(t,e,o)\ ) \tag{11}$$

Using variable elimination to solve this query, we would first need an **elimination order** - order in which to process and eliminate variables during variable elimination inference. Suppose elimination order $o_{elim} = [r,o,e,h,t]$. Then, given this ordering, we express our query as

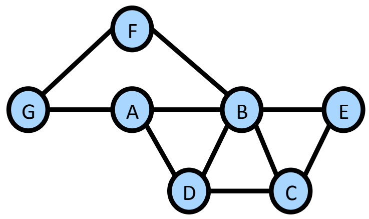
$$\max_T(\ \max_H(\ \max_E(\ \max_O(\ \max_R(\ f_T(t)\cdot f_H(h)\cdot f_E(e)\cdot f_{T,H,R}(t,h,r)\cdot f_{T,E,O}(t,e,o)\ )\ )\ )\ )\ ) \tag{12}$$

where the query can then be solved inside-to-out, variable-by-variable, via computations indicated by the parenthesis. The result from each step can be interpreted as the inference performed over its corresponding variable.

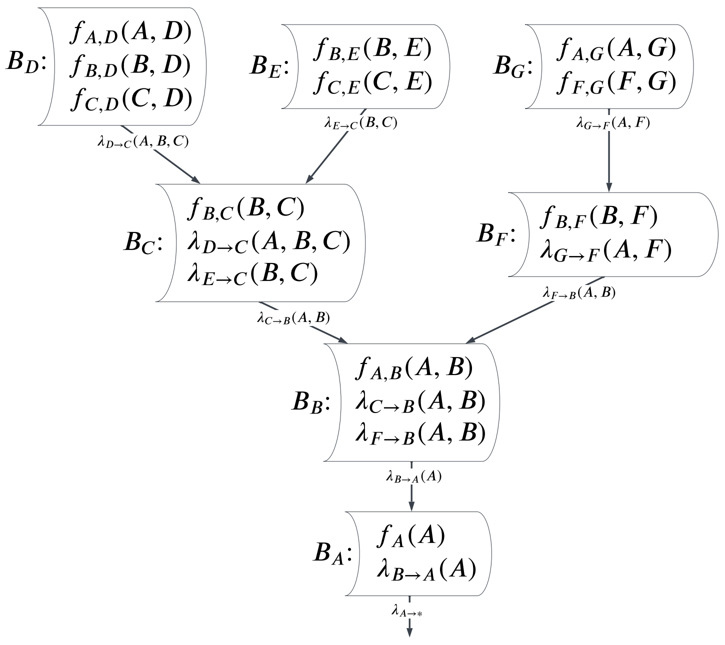
One power of variable elimination is its ability to simplify computation leveraging mathematical properties of the query. Note that in our example some of the model's functions are not dependent on the variable being immediately maximized over and so can be factored out of the respective maximization. Doing so recursively, we can rewrite our query with the same ordering instead as

$$\max_T(\ f_T(t)\cdot \max_E(\ f_E(e)\cdot \max_O(\ f_{T,E,O}(t,e,o)\ )\ )\cdot \max_H(\ f_H(h)\cdot \max_R(\ f_{T,H,R}(t,h,r)\ )\ )\ ) \tag{13}$$

which reduces the size of the terms being maximized over reducing complexity of the computations. (More on this shortly).



(a) Example primal graph of a graphical model with 7 variables and model functions $F = \{f_A(A),\ f_{A,B}(A,B),\ f_{A,D}(A,D),\ f_{A,G}(A,G),\ f_{B,C}(B,C),\ f_{B,D}(B,D),\ f_{B,E}(B,E),\ f_{B,F}(B,F),\ f_{C,D}(C,D),\ f_{C,E}(C,E),\ f_{F,G}(F,G)\}$.


B_D: $f_{A,D}(A,D)$ $f_{B,D}(B,D)$ $f_{C,D}(C,D)$
B_E: $f_{B,E}(B,E)$ $f_{C,E}(C,E)$
B_G: $f_{A,G}(A,G)$ $f_{F,G}(F,G)$
$\lambda_{D\to C}(A,B,C)$
$\lambda_{E\to C}(B,C)$
$\lambda_{G\to F}(A,F)$
B_C: $f_{B,C}(B,C)$ $\lambda_{D\to C}(A,B,C)$ $\lambda_{E\to C}(B,C)$
B_F: $f_{B,F}(B,F)$ $\lambda_{G\to F}(A,F)$
$\lambda_{C\to B}(A,B)$
$\lambda_{F\to B}(A,B)$
B_B: $f_{A,B}(A,B)$ $\lambda_{C\to B}(A,B)$ $\lambda_{F\to B}(A,B)$
$\lambda_{B\to A}(A)$
B_A: $f_A(A)$ $\lambda_{B\to A}(A)$
$\lambda_{A\to *}$

(b) Bucket elimination schematic following an elimination order $o_{elim} = [D, E, G, C, F, B, A]$.

Figure 3: (a) A primal graph of a graphical model with 7 variables. (b) Illustration of *BE* with an ordering A B C E D F G.

1.7 BUCKET ELIMINATION

Bucket elimination Dechter [1999], or **BE**, is a variable elimination scheme that can be adapted for a myriad of graphical model tasks including those described in Section 1.5.

Bucket elimination performs variable elimination according to a given elimination order by processing what are called **buckets** one-by-one, each corresponding to a variable in the provided ordering. When reaching some variable X_i in the ordering, all unprocessed functions that contain X_i in their scope are placed in bucket B_i (this includes the model's original functions as well as messages generated during the bucket elimination process). As shown in Equation 1.7, the bucket is then processed by applying an elimination operation (generalized as $\bigoplus$) over X_i to the combination of the bucket functions (generalized as $\bigotimes$) resulting in a **bucket message** denoted $\lambda_{i\to j}$, or λ_i for short.

$$\lambda_{i\to j} = \bigoplus_{X_i} \bigotimes_{f_\alpha \in B_i} f_\alpha(\alpha) \tag{14}$$

In the context of computing the partition function, this corresponds to marginalizing X_i from the product of the functions

$$\lambda_{i\to j} = \sum_{X_i} \prod_{f_\alpha \in B_i} f_\alpha(\alpha) \tag{15}$$

or in the context of computing the MAP, maximizing the product of the functions over X_i

$$\lambda_{i\to j} = \max_{X_i} \prod_{f_\alpha \in B_i} f_\alpha(\alpha) \tag{16}$$

The i in $\lambda_{i\to j}$ refers to the bucket that generated the message. j indicates the bucket this message will be sent to; namely the next variable in the elimination ordering that is also found in the scope of the message.

The processed buckets and messages can then be used to compute result of the underlying query (ex. in the case of computing the partition function or MAP, the result is simply the combination of the final messages). Figure 3 shows a schematic of bucket elimination on a graphical model with variables indexed from A to G and with pair-wise functions over the pairs of variables that are seen connected by an edge in the underlying primal graph (Figure 3a), namely: $F = \{f_A(A),\ f_{A,B}(A,B),\ f_{A,D}(A,D),\ f_{A,G}(A,G),\ f_{B,C}(B,C),\ f_{B,D}(B,D),\ f_{B,E}(B,E),\ f_{B,F}(B,F),\ f_{C,D}(C,D),\ f_{C,E}(C,E),\ f_{F,G}(F,G)\}$.

Bucket elimination can be viewed as a 1-iteration message-passing algorithm along its **bucket tree** (Figure 3b. The nodes of the tree are the different buckets. Each bucket of a variable contains a set of the model's functions depending on the given order of processing. There is an arc from bucket B_i to a parent bucket B_j, if the function created at bucket B_i is placed in bucket B_j.

In summary, bucket elimination uses the variable elimantion paradigm and dynamic programming to break a computational task into smaller subproblems, computing the result by processing buckets and sending resulting messages according to a provided elimination order.

Complexity. Both the time and space complexity of bucket elimination are exponential in the **induced width** of the model, which can be computed as a graph parameter based on the provided ordering and the underlying primal graph Dechter [2019]. In the context of bucket messages, the induced width is equal to the cardinality of the scope of the message with the largest scope. Given that its complexity is exponential in the induced width, bucket elimination becomes impractical if the induced width is large, and thus approximation schemes have been developed to address this Dechter and Rish [2002], Liu and Ihler [2011].

1.8 INDUCED WIDTH (W*)

The difficulty of answering a query using variable elimination can depend heavily on the elimination order being used, with some elimination orderings leading to efficient factorizations, whereas others may not and instead result large computations. We can capture this complexity graphically. As elimination computations are performed variable-by-variable pursuant to the ordering provided, the corresponding solutions (which themselves may be a function over some remaining variables) can be viewed as inducing new edges into the underlying primal graph in the same way the model's native functions did originally (see Section 1.3 for details). These new edges correspond to newly inferred relationships between variables not directly connected in the original graph. When adding all of the newly induced edges to the primal graph after processing of all of the variables, we end up with a new graph called the ***induced primal graph***, or ***induced graph*** for short. The complexity of variable elimination algorithms are typically with respect to the tree-width of the resulting induced graph - namely with respect to a quantity known as the ***induced width*** (or ***w****), which is one less than the largest clique-size of the induced graph.

Definition 1.8.0.1 (Induced Width (w*))**.** *The induced width of a graphical model $\mathcal{M}$ with respect to elimination order o_{elim} with induced primal graph $\mathcal{G}'$ is*

$$w^* = \max_{c \in clq(\mathcal{G}')} |c| - 1, \tag{17}$$

where $clq(\mathcal{G}')$ is the set of all cliques in $\mathcal{G}'$ and $|c|$ denotes the clique-size of clique c.

1.9 PSEUDO TREE

Given a variable ordering, directed tree called a ***pseudo tree*** $\mathcal{T} = (\mathbf{X}, \mathbf{E})$ can be constructed relative to a graphical model $\mathcal{M}$. Each node of the pseudo tree corresponds one-to-one with a node in $\mathcal{M}$. As before, to simplify, we abuse notation by using the same symbols to refer to pseudo tree nodes as their corresponding variables in $\mathcal{M}$. The tree will be structured such that the nodes follow the provided variable ordering - namely, each node is a descendant of only nodes that come before it in the provided ordering - and such that any branching in the tree corresponds to existing conditional dependencies in $\mathcal{M}$. A pseudo tree can be constructed by the following steps:

1. create a dummy root
2. add the first variable in the ordering $X_{i=1}$ as the child of the dummy root
3. for the next variable in the ordering X_{i+1}:
    i. choose an existing variable $X_p, p \in 0, ..., i$ in the partially constructed pseudo tree $\mathcal{T}'$ such that X_{i+1} is conditionally independent of all existing descendants of X_p in $\mathcal{T}'$ given X_p and its ancestors.
    ii. add X_{i+1} to $\mathcal{T}'$ as the child of X_p
4. repeat step 3 until all variables in the ordering (and thus $\mathcal{M}$) have been added to the tree

By convention, the dummy root often omitted in pseudo tree diagram representations.

Increasing Pseudo Tree Branching. In order to capture the maximal number of conditional independences given an ordering, step 3 i. can be altered to choose the earliest variable in the ordering that satisfies the said condition, thus leading to earlier branching in the tree.

Pseudo Tree Uses. One use of the pseudo tree is as a schematic of bucket elimination. Mores specifically, message passing from [mini] bucket elimination with an elimination ordering that is the reverse of o will follow a path from the leaves to the root of $\mathcal{T}$. (TODO: EXAMPLE). Furthermore, the pseudo tree can act as a blue print for constructing search space graphs of $\mathcal{M}$ as will be described in the next section. In combination, given an ordering o and corresponding search space and bucket elimination, the messages from the bucket elimination can act as heuristics guiding the search at each level.

References

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. doi: 10.2200/S00529ED1V01Y201308AIM023. URL http://dx.doi.org/10.2200/S00529ED1V01Y201308AIM023.

Rina Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms, second edition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13:1–199, 02 2019.

Rina Dechter and I Rish. Mini-buckets: A general scheme for approximating inference. *Journal of the ACM*, pages 107–153, 2002.

Mark Hallen, Jeffrey Martin, Adegoke Ojewole, Jonathan Jou, Anna Lowegard, Marcel Frenkel, Pablo Gainza, Hunter Nisonoff, Aditya Mukund, Siyu Wang, Graham Holt, David Zhou, Elizabeth Dowd, and Bruce Donald. Osprey 3.0: Open-source protein redesign for you, with powerful new features. *Journal of Computational Chemistry*, 39, 10 2018.

Qiang Liu and Alexander Ihler. Bounding the partition function using Hölder's inequality. In *International Conference on Machine Learning (ICML)*, pages 849–856. ACM, June 2011.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.