# Hybrid Best First Search for Computing “Anytime” Partition Function

Clément Viricel[1,2], PhD directors: Sophie Barbe[1] and Thomas Schiex[2]

[1]Institut National des Sciences Appliquées, 135 Avenue de Rangueil, 31400 Toulouse
[2]Institut National de la Recherche Agronomique,24 Chemin de Borde Rouge, 31326 Castanet-Tolosan

## Definition

1. Set $X = \{X_1, \cdots, X_n\}$ of **variables**, with a **domain** $D_i$ containing values (booleen, interger, real).
2. Set $\Phi$ of **local functions** $\phi_S$ involving variables of $S \subset X$ (scope).
3. A **joint function** for a full assignement $t$ :

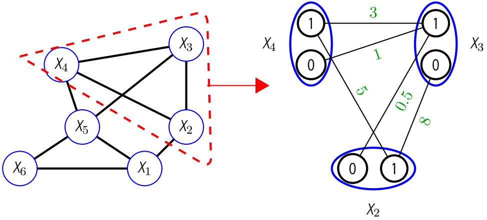
$$\Phi(t) = \bigoplus_{\phi_S \in \Phi} \phi_S(t[S])$$

### Definition

1. Set $X = \{X_1, \cdots, X_n\}$ of **variables**, with a **domain** $D_i$ containing values (booleen, interger, real).
2. Set $\Phi$ of **local functions** $\phi_S$ involving variables of $S \subset X$ (scope).
3. A **joint function** for a full assignement $t$ :

$$\Phi(t) = \bigoplus_{\phi_S \in \Phi} \phi_S(t[S])$$

### Toy Example



# Markov Random Field

## Probability of an assignment

The joint **probability** of a **complete assignment** is defined as:

$$p(t) = \frac{P(t)}{Z} = \frac{1}{Z} \prod_{\phi_S \in \Phi} \phi_S(t[S])$$

# Markov Random Field

## Probability of an assignment

The joint **probability** of a **complete assignment** is defined as:

$$p(t) = \frac{P(t)}{Z} = \frac{1}{Z} \prod_{\phi_S \in \Phi} \phi_S(t[S])$$

## Partition Function

The **normalizing constant or Partition Function**:

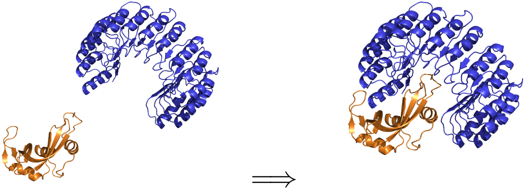
$$Z = \sum_t P(t) = \sum_t \prod_{\phi_S \in \Phi} \phi_S(t[S])$$

# Partition Function, yes ! But why ?

## Constante d'affinité

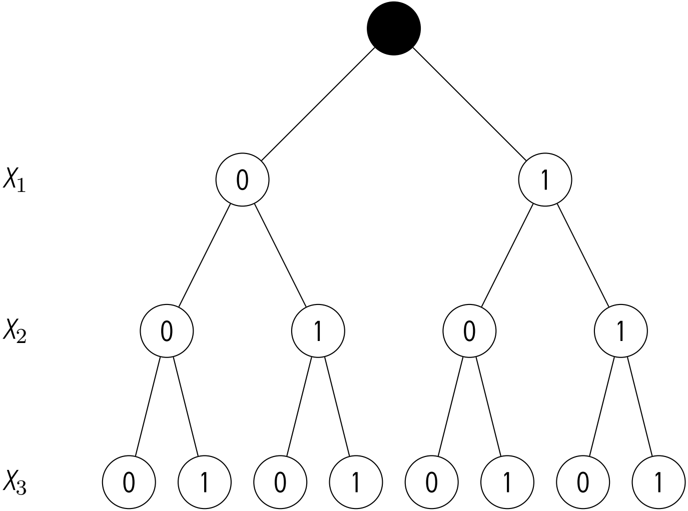
We can **approximate the affinity between two proteins** $P_1$ et $P_2$ forming a complex $C$ by:

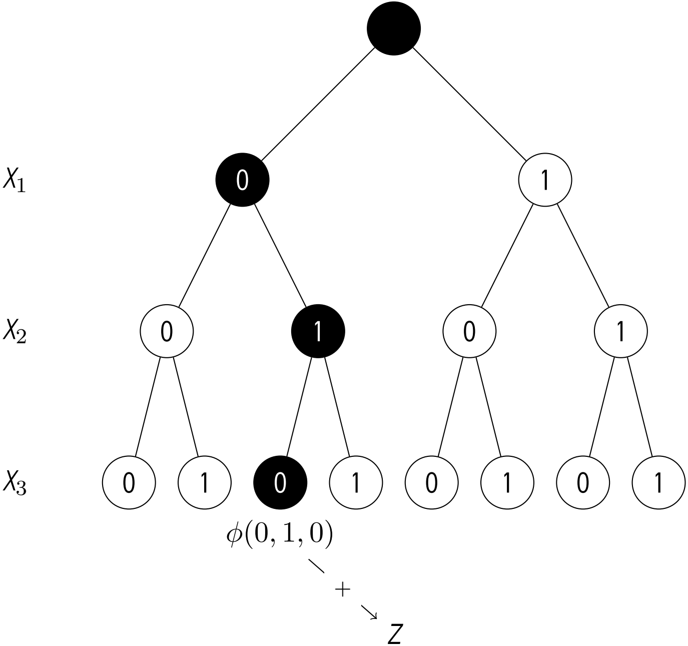
$$K_a = e^{(k_B T)} \frac{C^0}{8\pi^2} \frac{\sigma_{P_1}\sigma_{P_2}}{\sigma_C} \frac{Z(T, V, C)}{Z(T, V, P_1)\; Z(T, V, P_2)}$$



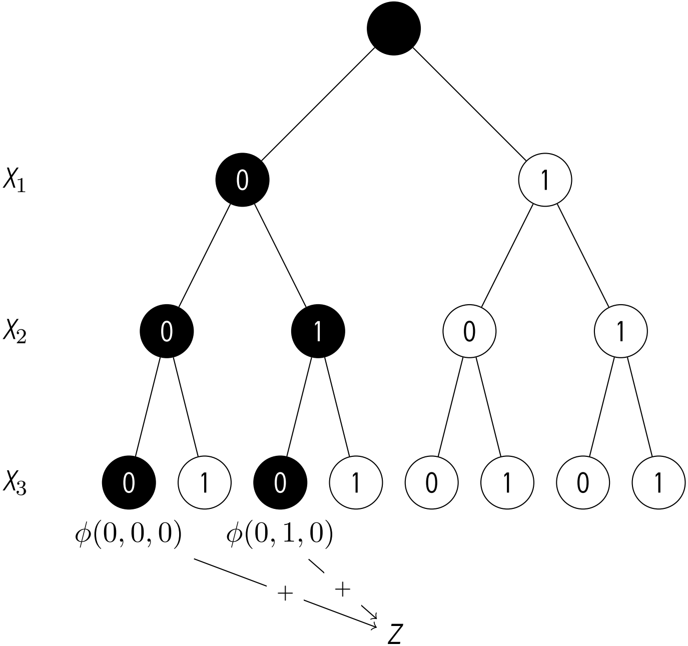
Unbound $\Longrightarrow$ Bound

# Solving by search tree

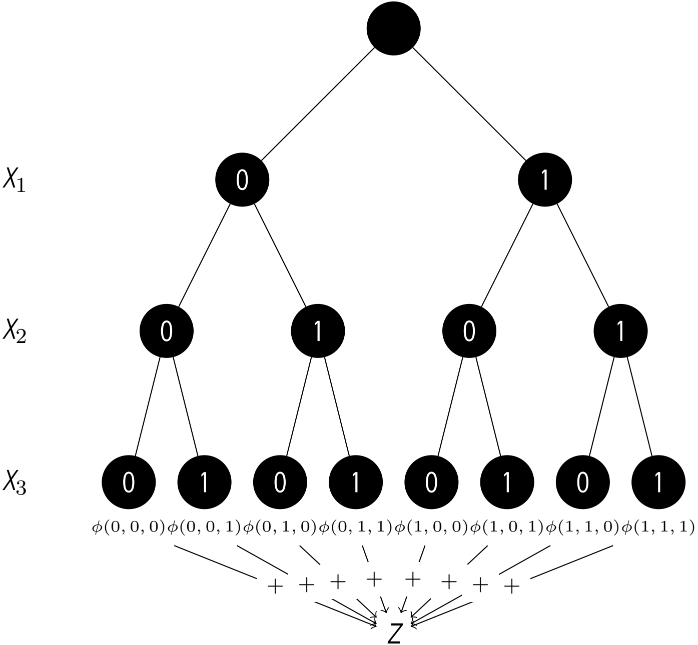



$X_1$
$X_2$
$X_3$
0
1
0
1
0
1
0
1
0
1
0
1
0
1
$\phi(0, 1, 0)$
$+$
$Z$

# Solving by search tree



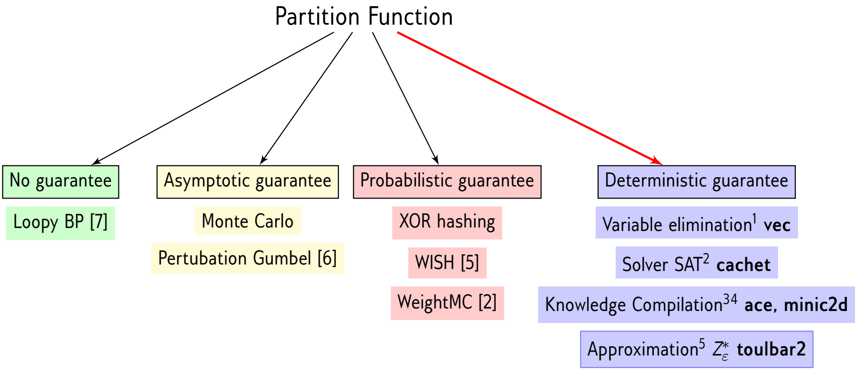
# Solving by search tree



## Combinatorial Explosion

Computing $Z$ is classified ad **#P-complet** problem so it is really hard

# Existing approaches for computing $Z$




---

[1] Rina Dechter. "Bucket Elimination: A Unifying Framework for Reasoning". In: *Artificial Intelligence* 113.1–2 (1999), pp. 41–85.

[2] Tian Sang, Paul Beame, and Henry A Kautz. "Performing Bayesian inference by weighted model counting". In: *AAAI*. vol. 5. 2005, pp. 475–481.

[3] Mark Chavira and Adnan Darwiche. "On probabilistic inference by weighted model counting". In: *Artificial Intelligence* 172.6 (2008), pp. 772–799.

[4] Umut Oztok and Adnan Darwiche. "A top-down compiler for sentential decision diagrams". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. *AAAI Press*. 2015.

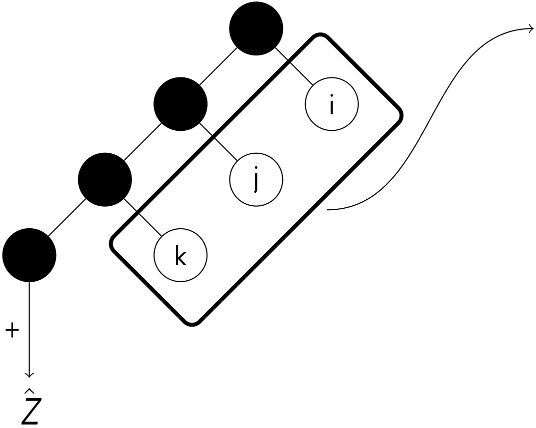
[5] Clément Viricel et al. "Guaranteed weighted counting for affinity computation: Beyond determinism and structure". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2016, pp. 733–750.

# How does it work ?

## Hybrid Best First Search

We adapt **HBFS**[a] search algorithm to **compute the partition function**

[a] David Allouche et al. "Anytime hybrid best-first search with tree decomposition for weighted CSP". . In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2015, pp. 12–29.



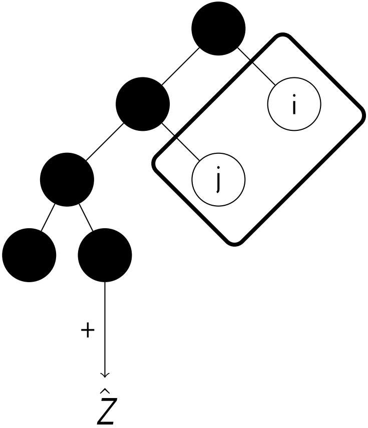

Open List :=

$$\begin{bmatrix} i & LB(i) & UB(i) \\ j & LB(j) & UB(j) \\ k & LB(k) & UB(k) \end{bmatrix}$$

## Hybrid Best First Search

We adapt **HBFS**[a] search algorithm to **compute the partition function**

[a] David Allouche et al. "Anytime hybrid best-first search with tree decomposition for weighted CSP". . In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2015, pp. 12–29.



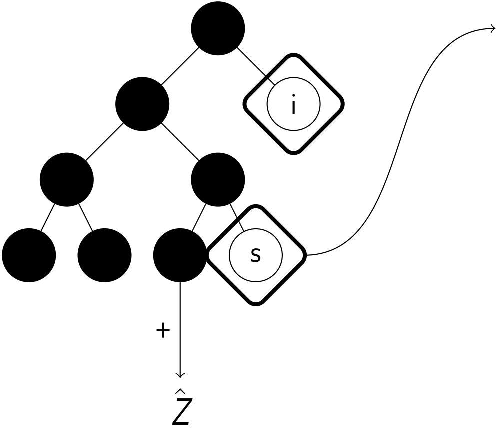

Open List :=

$$\begin{bmatrix} i & LB(i) & UB(i) \\ j & LB(j) & UB(j) \end{bmatrix}$$

## Hybrid Best First Search

We adapt **HBFS**[a] search algorithm to **compute the partition function**

[a] David Allouche et al. “Anytime hybrid best-first search with tree decomposition for weighted CSP”. . In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2015, pp. 12–29.




Open List :=

$$\begin{bmatrix} i & LB(i) & UB(i) \\ s & LB(s) & UB(s) \end{bmatrix}$$

**Function** HBFS-C

$Open = [(root, LB(root), UB(root))]$ ;

**while** $Open \neq \varnothing$ **do**

- $n = pop(Open)$ ;
- $restore(n)$ ;
- $\hat{Z} = DFS(n, \hat{Z}, k)$ ;

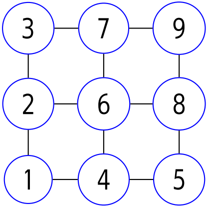
$$\hat{Z} + \sum_{n \in Open} LB(n) \ \leq \ Z \ \leq \ \hat{Z} + \sum_{n \in Open} UB(n)$$

We can have an anytime guarantee with:

$$\hat{Z} + \sum_{n \in Open} UB(n) \leq (1 + \varepsilon) \left( \hat{Z} + \sum_{n \in Open} LB(n) \right)$$

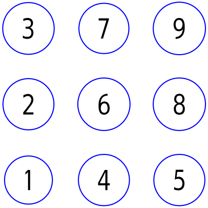
# Mean Field Lower Bound

**Original graph**




$$P \propto \prod_t \phi_t(x_t)$$

**Naive Mean Field**




$$Q \propto \prod_{i \in V} q_i(x_i)$$

$$\log(Z) \geq \sum_{i \in V} \sum_{x_i} [S_i(q_i) - q_i(x_i) E_i(x_i)] - \sum_{(i,j) \in E} \sum_{x_i, x_j} q_i(x_i) q_j(x_j) E_{ij}(x_i, x_j)$$

**Function** MF-LB(n)

$t \leftarrow 0$;

Initialise $q^{(t)}$ ;

**while** $q^{(t)}$ converge **do**

**for** $i \in X(n)$ **do**

$$q_i^{(t+1)} \leftarrow \exp\left(-E_i(x_i) - \sum_{j \in \mathcal{N}(i)} \sum_{x_j} q_j(x_j) E_{ij}(x_i, x_j)\right) ;$$

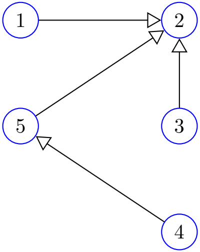
$$Z_i \leftarrow \sum_{x_i} q_i^{t+1}(x_i) ;$$

$$q_i^{(t+1)} \leftarrow \frac{q_i^{(t+1)}}{Z_i} ;$$

## Upper Bound by Maximum Spanning Tree

Define a **maximum spanning tree** $T \subset \Phi$ and by applying **dynamic programming** to $T' = T \cup \{E_S \in \Phi \ : \ |S| < 2\}$, we have an exact $Z_{T'}$ in **polynomial time.**

$$Z \leq Ub_T = \underbrace{\left( \sum_{t \in D^X} \prod_{\phi_S \in T} \phi_S(t) \right)}_{programmation \ \ dynamique} \cdot \left( \prod_{\phi_S \in \Phi \setminus T} \max_{t \in D^S} \phi_S(t) \right)$$



# Perspectives

## Extension

- Mixing BTD (Backtrack Tree decomposition) with HBFS-C
- Integrate $Z^*_\varepsilon$ pruning to HBFS

## Test

Run a battery of tests (HBFS-C; HBFS-C + $Z^*_\varepsilon$ ; HBFS-C+BTD; HBFS-C + BTD+ $Z^*_\varepsilon$) to see the dynamic of all the algorithm.

## Application

If these are improvement then try to solve large protein instances to predict affinity.

# Question(s) ?


WITH GREAT POWER
COMES
VIA 9GAG.COM
GREAT DIFFICULTY IN
FACTORIZING THE POLYNOMIAL
9GAG is your best source of fun.

David Allouche et al. "Anytime hybrid best-first search with tree decomposition for weighted CSP". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2015, pp. 12–29.

Supratik Chakraborty et al. "Distribution-aware sampling and weighted model counting for SAT". In: *arXiv preprint arXiv:1404.2984* (2014).

Mark Chavira and Adnan Darwiche. "On probabilistic inference by weighted model counting". In: *Artificial Intelligence* 172.6 (2008), pp. 772–799.

Rina Dechter. "Bucket Elimination: A Unifying Framework for Reasoning". In: *Artificial Intelligence* 113.1–2 (1999), pp. 41–85.

Stefano Ermon et al. "Taming the curse of dimensionality: Discrete integration by hashing and optimization". In: *arXiv preprint arXiv:1302.6677* (2013).

Tamir Hazan, Subhransu Maji, and Tommi Jaakkola. "On sampling from the Gibbs distribution with random maximum a-posteriori perturbations". In: *Advances in Neural Information Processing Systems*. 2013, pp. 1268–1276.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519.

Umut Oztok and Adnan Darwiche. "A top-down compiler for sentential decision diagrams". In: *Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press*. 2015.

Tian Sang, Paul Beame, and Henry A Kautz. “Performing Bayesian inference by weighted model counting”. In: *AAAI*. Vol. 5. 2005, pp. 475–481.

Clément Viricel et al. “Guaranteed weighted counting for affinity computation: Beyond determinism and structure”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2016, pp. 733–750.