

# Exploiting Combinatorial Structure in CP: Going Beyond Domain Filtering to Model Counting and Marginal Distributions over Domains

Gilles Pesant

Polytechnique Montréal, Montreal, Canada  
`gilles.pesant@polymtl.ca`

CP Conference, FLoC 2026  
Lisboa, Portugal  
July 22, 2026

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - Sampling Solutions
  - Neuro-Symbolic AI
- 4 Conclusion

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

# Model-based combinatorial solving paradigms

## SAT

lots of

$$x_1 \vee x_2 \vee \overline{x_3}$$

## Integer Programming

lots of

$$3x_1 - 2x_2 + 5x_3 \leq 10$$

## Constraint Programming

not so many

constraints in heterogeneous syntax

Ex:  $\text{cumulative}(\langle x_1, \dots, x_k \rangle, \text{durations}, \text{demands}, \text{resource capacity})$

# Constraint Programming

**Q-** What is the distinctive driving force behind CP?

**A-** More direct access to problem structure from high-level constraints

# Constraint Programming

Q- What is the distinctive driving force behind CP?

A- More direct access to problem structure from high-level constraints

How does one nominate these high-level constraints?

- complex enough to provide structural insight
- simple enough for some desired computing tasks to remain tractable

# Constraint Programming

Q- What is the distinctive driving force behind CP?

A- More direct access to problem structure from high-level constraints

How does one nominate these high-level constraints?

- complex enough to provide structural insight
- simple enough for some desired computing tasks to remain tractable

What sort of thing does one wish to compute about constraints?

- satisfiability: "Is there any solution to constraint  $c$ ?"
- **domain filtering**: "Any solution to  $c$  s.t. variable  $x$  takes value  $d$ ?"
- ...
- "How many solutions are there to  $c$ ?"
- "How many solutions in which  $x = d$ ?"

# Constraint Programming

Q- What is the distinctive driving force behind CP?

A- More direct access to problem structure from high-level constraints

How does one nominate these high-level constraints?

- complex enough to provide structural insight
- simple enough for some desired computing tasks to remain tractable

What sort of thing does one wish to compute about constraints?

- satisfiability: “Is there any solution to constraint  $c$ ?”
- **domain filtering**: “Any solution to  $c$  s.t. variable  $x$  takes value  $d$ ?”
- ...
- “How many solutions are there to  $c$ ?” generalizes satisfiability
- “How many solutions in which  $x = d$ ?” generalizes domain filtering

# Constraints a.k.a. Combinatorial Substructures

How has this problem structure mostly been used so far?

To design domain-filtering (i.e. search-space reduction) algorithms and achieve *consistency*

$$A \in \{1, 2, 3, 4, 5\}$$

$$B \in \{1, 2, 3, 4, 5\}$$

$$C \in \{1, 2, 3, 4, 5\}$$

$$D \in \{1, 2, 3, 4, 5\}$$

$$E \in \{1, 2, 3, 4, 5\}$$

$\text{regular}(\langle A, B, C, D, E \rangle, \mathcal{M})$

$\text{alldifferent}(\{A, B, C, D, E\})$

# Constraints a.k.a. Combinatorial Substructures

How has this problem structure mostly been used so far?

To design domain-filtering (i.e. search-space reduction) algorithms and achieve *consistency*

$A \in \{1, 2, 3, 4, 5\}$

$B \in \{1, 2, 3, 4, 5\}$

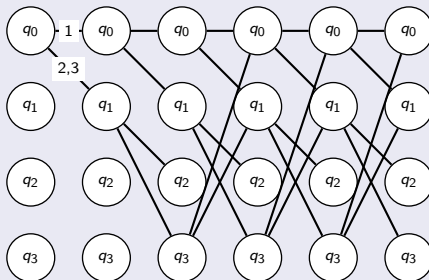
$C \in \{1, 2, 3, 4, 5\}$

$D \in \{1, 2, 3, 4, 5\}$

$E \in \{1, 2, 3, 4, 5\}$

$\text{regular}(\langle A, B, C, D, E \rangle, \mathcal{M})$

$\text{alldifferent}(\{A, B, C, D, E\})$



# Constraints a.k.a. Combinatorial Substructures

How has this problem structure mostly been used so far?

To design domain-filtering (i.e. search-space reduction) algorithms and achieve *consistency*

$$A \in \{1, 2, 3\}$$

$$B \in \{1, 3\}$$

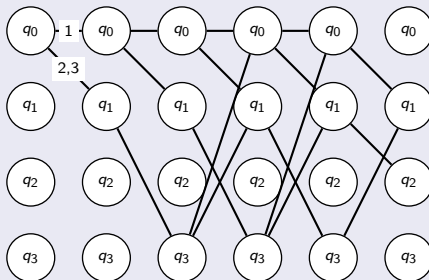
$$C \in \{2, 3\}$$

$$D \in \{1, 2, 3, 4\}$$

$$E \in \{2, 4, 5\}$$

$\text{regular}(\langle A, B, C, D, E \rangle, \mathcal{M})$

$\text{alldifferent}(\{A, B, C, D, E\})$

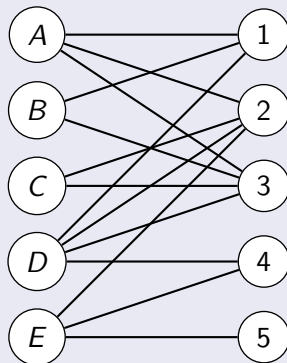


# Constraints a.k.a. Combinatorial Substructures

How has this problem structure mostly been used so far?

To design domain-filtering (i.e. search-space reduction) algorithms and achieve *consistency*

$A \in \{1, 2, 3\}$   
 $B \in \{1, 3\}$   
 $C \in \{2, 3\}$   
 $D \in \{1, 2, 3, 4\}$   
 $E \in \{2, 4, 5\}$   
 $\text{regular}(\langle A, B, C, D, E \rangle, \mathcal{M})$   
 $\text{alldifferent}(\{A, B, C, D, E\})$



# Constraints a.k.a. Combinatorial Substructures

How has this problem structure mostly been used so far?

To design domain-filtering (i.e. search-space reduction) algorithms and achieve *consistency*

$A \in \{1, 2\}$

$B \in \{1, 3\}$

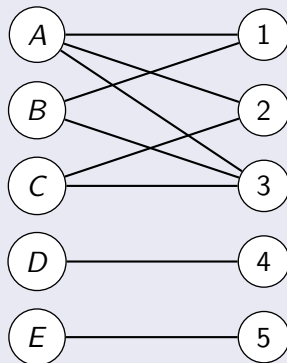
$C \in \{2, 3\}$

$D \in \{4\}$

$E \in \{5\}$

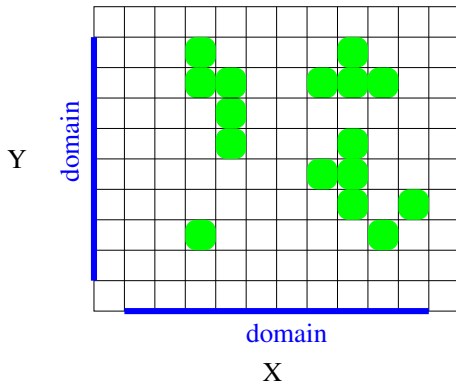
$\text{regular}(\langle A, B, C, D, E \rangle, \mathcal{M})$

$\text{alldifferent}(\{A, B, C, D, E\})$



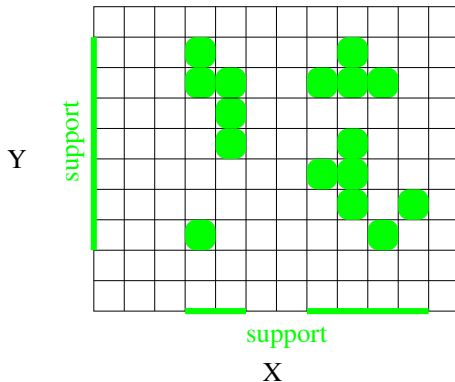
# Constraints a.k.a. Combinatorial Substructures

Consider a simple constraint on finite-domain variables  $X$  and  $Y$ .



# Constraints a.k.a. Combinatorial Substructures

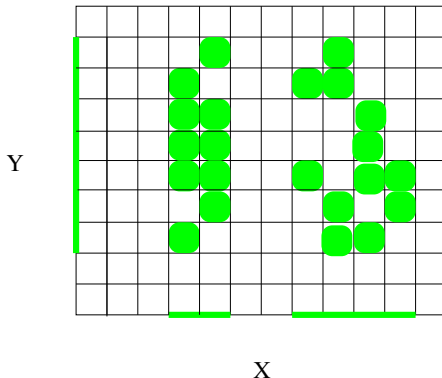
Consider a simple constraint on finite-domain variables  $X$  and  $Y$ .



domain filtering  $\equiv$  projecting solutions on individual variables

# Constraints a.k.a. Combinatorial Substructures

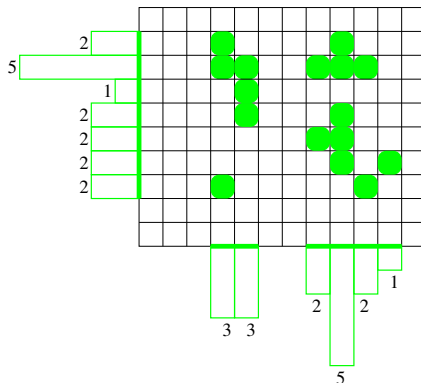
Consider a simple constraint on finite-domain variables  $X$  and  $Y$ .



same "outside information", but very different set of solutions

# Constraints a.k.a. Combinatorial Substructures

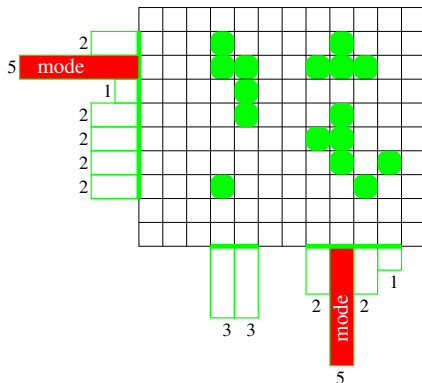
Now consider the set of solutions as a **multivariate discrete distribution**.



marginals  $\equiv$  projecting that distribution on individual variables

# Constraints a.k.a. Combinatorial Substructures

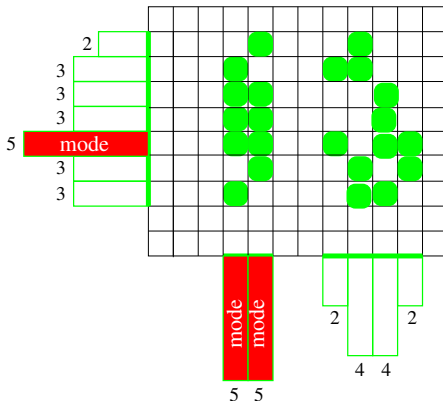
Now consider the set of solutions as a **multivariate discrete distribution**.



A possible branching heuristic: on a mode of the marginal distributions

# Constraints a.k.a. Combinatorial Substructures

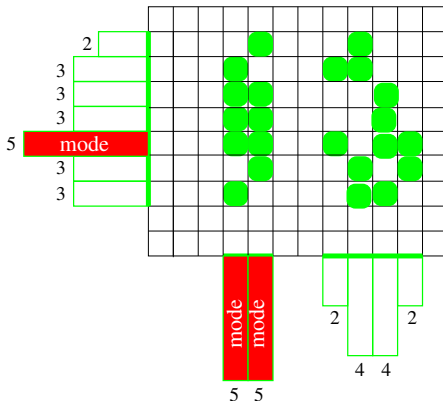
Now consider the set of solutions as a **multivariate discrete distribution**.



A possible branching heuristic: on a mode of the marginal distributions

# Constraints a.k.a. Combinatorial Substructures

Now consider the set of solutions as a **multivariate discrete distribution**.



Technically, we need to **count solutions**: 5 out of 22 solutions

# Outline

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

# Tasks over Constraints (Combinatorial Structures)

- domain filtering
- model counting
- weighted model counting
- marginal distribution over the domain of individual variables

And we'll see some common design patterns to perform these tasks.

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

Let's start slow...

the  $x \neq y$  constraint

## Definition

The  $x \neq y$  constraint holds if  $x$  and  $y$  take different values.

## Example

$D(x) = \{1, 2\}$ ,  $D(y) = \{1, 2, 3\}$  no domain filtering

$D(x) = \{1\}$ ,  $D(y) = \{1, 2, 3\}$

Let's start slow...

the  $x \neq y$  constraint

## Definition

The  $x \neq y$  constraint holds if  $x$  and  $y$  take different values.

## Example

$D(x) = \{1, 2\}$ ,  $D(y) = \{1, 2, 3\}$  no domain filtering

$D(x) = \{1\}$ ,  $D(y) = \{\cancel{1}, 2, 3\}$

Let's start slow...

the  $x \neq y$  constraint

## Definition

The  $x \neq y$  constraint holds if  $x$  and  $y$  take different values.

## Example

$D(x) = \{1, 2\}$ ,  $D(y) = \{1, 2, 3\}$  no domain filtering

$D(x) = \{1\}$ ,  $D(y) = \{\cancel{1}, 2, 3\}$

## domain filtering algorithm

**if**  $x$  is bound (to  $v$ ) **then**

    remove  $v$  from  $D(y)$

**if**  $y$  is bound (to  $v$ ) **then**

    remove  $v$  from  $D(x)$

# Model Counting on $x \neq y$

```
count  $\leftarrow$  0
foreach  $v \in D(x)$  do
  if  $v \notin D(y)$  then
    | count  $+= |D(y)|$ 
  else
    | count  $+= |D(y)| - 1$ 
return count
```

# Model Counting on $x \neq y$

```
count  $\leftarrow$  0
foreach  $v \in D(x)$  do
    if  $v \notin D(y)$  then
        | count  $+= |D(y)|$ 
    else
        | count  $+= |D(y)| - 1$ 
return count
```

Note that, even for this simple case, we do better than enumerating all solutions explicitly.

## Marginal distributions over $D(x)$ and $D(y)$ from $x \neq y$

```
// marginals  $\theta_x$  for  $x$ 
foreach  $v \in D(x)$  do
  if  $v \notin D(y)$  then
     $\theta_x(v) \leftarrow |D(y)|$ 
  else
     $\theta_x(v) \leftarrow |D(y)| - 1$ 
normalize  $\theta_x$ 
```

```
// marginals  $\theta_y$  for  $y$ 
foreach  $v \in D(y)$  do
  if  $v \notin D(x)$  then
     $\theta_y(v) \leftarrow |D(x)|$ 
  else
     $\theta_y(v) \leftarrow |D(x)| - 1$ 
normalize  $\theta_y$ 
```

# Weighted Model Counting on $x \neq y$

```
wcount  $\leftarrow$  0
foreach  $v \in D(x)$  do
  if  $v \notin D(y)$  then
    | wcount  $+= \omega_x(v) \times \sum_{v' \in D(y)} \omega_y(v')$ 
  else
    | wcount  $+= \omega_x(v) \times ((\sum_{v' \in D(y)} \omega_y(v')) - \omega_y(v))$ 
return wcount
```

given weights  $\omega_x(v)$  for each variable  $x$  and value  $v \in D(x)$

the weight of a solution is the product of the weights of its assignments

## Weighted Model Counting on $x \neq y$

```
wcount  $\leftarrow$  0
foreach  $v \in D(x)$  do
  if  $v \notin D(y)$  then
    | wcount  $+= \omega_x(v)$ 
  else
    | wcount  $+= \omega_x(v) \times (1 - \omega_y(v))$ 
return wcount
```

given **normalized** weights  $\omega_x(v)$  for each variable  $x$

i.e.  $\sum_{v \in D(x)} \omega_x(v) = 1$

## Marginal distributions over $D(x)$ and $D(y)$ from $x \neq y$

```
// marginals  $\theta_x$  for  $x$   
foreach  $v \in D(x)$  do  
  if  $v \notin D(y)$  then  
     $\theta_x(v) \leftarrow 1$   
  else  
     $\theta_x(v) \leftarrow 1 - \omega_y(v)$   
normalize  $\theta_x$ 
```

```
// marginals  $\theta_y$  for  $y$   
foreach  $v \in D(y)$  do  
  if  $v \notin D(x)$  then  
     $\theta_y(v) \leftarrow 1$   
  else  
     $\theta_y(v) \leftarrow 1 - \omega_x(v)$   
normalize  $\theta_y$ 
```

# Outline

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- **alldifferent** constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

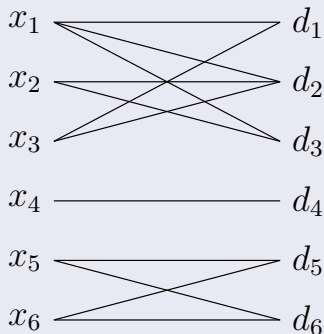
## 4 Conclusion

# alldifferent constraint

## Definition

The  $\text{alldifferent}(X)$  constraint holds if the values taken by the set of finite domain variables  $X = \{x_1, x_2, \dots, x_k\}$  are distinct.

## bipartite graph



## adjacency matrix

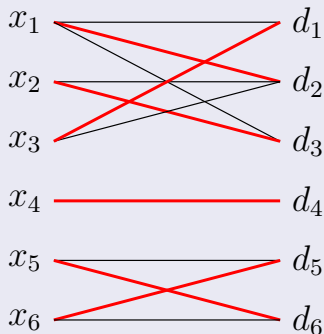
$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

# alldifferent constraint

## Definition

The  $\text{alldifferent}(X)$  constraint holds if the values taken by the set of finite domain variables  $X = \{x_1, x_2, \dots, x_k\}$  are distinct.

## bipartite graph



## adjacency matrix

$$\mathbf{A} = \begin{pmatrix} 1 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 0 & 1 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 1 \end{pmatrix}$$

# Domain filtering on alldifferent

## Domain filtering

bipartite graph matching + strongly-connected components

computed in polynomial time

# Domain filtering on alldifferent

## Domain filtering

bipartite graph matching + strongly-connected components

computed in polynomial time

But now counting solutions costs significantly more

# Model Counting on alldifferent

Its number of solutions is the same as...

- the number of perfect matchings in the bipartite graph
- the permanent of the adjacency matrix

$$\text{per}(A) = \sum_{\sigma \in S_n} \prod_i a_{i, \sigma(i)}$$

# Model Counting on alldifferent

Its number of solutions is the same as...

- the number of perfect matchings in the bipartite graph
- the permanent of the adjacency matrix

$$\text{per}(A) = \sum_{\sigma \in S_n} \prod_i a_{i,\sigma(i)}$$

## Remark

It is a  $\#P$ -complete problem, that is, it cannot be computed in polynomial time (under reasonable theoretical assumptions)

## Rasmussen's Estimator

```
if  $n = 0$  then  
  |  $X_A = 1$   
else  
  |  $W = \{j : a_{1,j} = 1\}$   
  | if  $W = \emptyset$  then  
    |  $X_A = 0$   
  | else  
    | Choose  $j$  u.a.r. from  $W$   
    | Compute  $X_{A_{1,j}}$   
    |  $X_A = |W| \cdot X_{A_{1,j}}$ 
```

# Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & \color{red}{1} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\begin{array}{l} |W| \\ 3 \end{array}$$

# Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} 0 & \color{red}{1} & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$|W|$$
$$\begin{matrix} 3 \\ 1 \end{matrix}$$

# Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} \color{red}{1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$|W|$$

|   |
|---|
| 3 |
| 1 |
| 1 |

## Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\begin{array}{c} |W| \\ 3 \\ 1 \\ 1 \\ 1 \end{array}$$

## Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & 1 & 1 \\ & & & 1 & 1 \end{pmatrix}$$

$$\begin{array}{c} |W| \\ 3 \\ 1 \\ 1 \\ 1 \\ 2 \end{array}$$

## Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} & & \\ & & \\ & & 1 \end{pmatrix}$$

$$\begin{array}{c} |W| \\ 3 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \end{array}$$

# Rasmussen's estimator

## Example

$$\mathbf{A} = \begin{pmatrix} & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{pmatrix}$$

$|W|$   
3  
1  
1  
1  
2  
1

$$X_A = 6$$

# Rasmussen's estimator properties

## Properties

- unbiased estimator:  $\mathbb{E}(X_A) = \text{per}(A)$  (for “almost” all dense matrices)
- Poor results in some special cases

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ & 1 & \dots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix}$$

# Rasmussen's estimator properties

## Properties

- unbiased estimator:  $\mathbb{E}(X_A) = \text{per}(A)$  (for “almost” all dense matrices)
- Poor results in some special cases

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ & 1 & \dots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix}$$

# Rasmussen's estimator properties

## Properties

- unbiased estimator:  $\mathbb{E}(X_A) = \text{per}(A)$  (for “almost” all dense matrices)
- Poor results in some special cases

$$\mathbf{U} = \begin{pmatrix} \color{red}{1} & 1 & \dots & 1 \\ & \color{red}{1} & \dots & 1 \\ & & \ddots & \vdots \\ & & & \color{red}{1} \end{pmatrix}$$

even simple consistency techniques can help avoid dead ends ( $W = \emptyset$ )

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ & 1 & \dots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

# Adding domain filtering

## Modified Rasmussen

**if**  $n = 0$  **then**

$X_A = 1$

**else**

    Domain filtering on  $A$

    Choose  $i$  u.a.r. from  $\{1 \dots n\}$

$W = \{j : a_{i,j} = 1\}$

**if**  $W = \emptyset$  **then**

$X_A = 0$

**else**

        Choose  $j$  u.a.r. from  $W$

        Compute  $X_{A_{i,j}}$

$X_A = |W| \cdot X_{A_{i,j}}$

# Model Counting and Marginals on alldifferent

## model counting

average  $X_A$  over multiple runs  $\approx per(A)$

## marginal distributions

$$\theta_{x_i}(d) \approx \frac{|S_{x_i,d}|}{|S|}$$

where  $S$  is the set of samples and  $S_{x_i,d}$  its subset s.t.  $x_i = d$

# Weighted Model Counting on alldifferent

## Weighted Modified Rasmussen

**if**  $n = 0$  **then**

|  $X_A = 1$

**else**

Domain filtering on  $A$

Choose  $i$  u.a.r. from  $\{1 \dots n\}$

$W = \{j : a_{i,j} > 0\}$

**if**  $W = \emptyset$  **then**

|  $X_A = 0$

**else**

Choose  $j$  from  $W$  randomly  
according to the distribution  
of weights

Compute  $X_{A_{i,j}}$

$X_A = (\sum_{j \in W} a_{i,j}) \cdot X_{A_{i,j}}$

$$\begin{pmatrix} .6 & .1 & .3 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 & 0 \\ .7 & .3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & .2 & .8 \\ 0 & 0 & 0 & 0 & .4 & .6 \end{pmatrix}$$

nonnegative matrix entries  $a_{i,j}$   
as weights

## Design pattern

### Sampling

another ex.: circuit constraint (Hamiltonian circuit on a digraph)

```
if  $n = 1$  then
   $X_G = \begin{cases} w_{(1,1)} & \text{if } (1,1) \in \mathcal{A} \\ 0 & \text{else} \end{cases}$ 
else
  Domain filtering on  $\mathcal{G}$ 
   $W = \{i > 1 : (1, i) \in \mathcal{A}\}$ 
  if  $W = \emptyset$  then
     $X_G \leftarrow 0$ 
  else
    Choose  $J \in W$  randomly according to the weight of each edge  $(1, J)$ 
     $X_G \leftarrow X_{G_{1,J}} \times \sum_{i \in W} w_{(1,i)}$ 
```

# Model Counting on alldifferent

But collecting enough samples may be too time consuming...

Fortunately the permanent is a well-studied problem:  
there are known upper bounds for the permanent of 0-1 matrices.

$\text{alldifferent}(X_1, X_2, X_3, X_4)$

|                       |               |      |       |   |   |   |   |
|-----------------------|---------------|------|-------|---|---|---|---|
| $X_1 \in \{a, b, c\}$ | $\Rightarrow$ | $A:$ |       | a | b | c | d |
| $X_2 \in \{b, d\}$    |               |      | $X_1$ | 1 | 1 | 1 | 0 |
| $X_3 \in \{b, d\}$    |               |      | $X_2$ | 0 | 1 | 0 | 1 |
| $X_4 \in \{a, c, d\}$ |               |      | $X_3$ | 0 | 1 | 0 | 1 |
|                       |               |      | $X_4$ | 1 | 0 | 1 | 1 |

# Model Counting on alldifferent

## Minc-Brègman

$$\text{perm}(A) \leq \prod_{i=1}^m (r_i!)^{1/r_i}$$

where  $r_i$  = number of 1's in row  $i$  (i.e.  $|D_i|$ )

## Liang-Bai

$$\text{perm}(A)^2 \leq \prod_{i=1}^m q_i (r_i - q_i + 1)$$

where  $q_i = \min\{\lceil \frac{r_i+1}{2} \rceil, \lceil \frac{i}{2} \rceil\}$

# Weighted Model Counting on alldifferent

`alldifferent`( $X_1, X_2, X_3, X_4$ )

|                       |                   |       |    |    |    |    |
|-----------------------|-------------------|-------|----|----|----|----|
| $X_1 \in \{a, b, c\}$ | $\Rightarrow A :$ |       | a  | b  | c  | d  |
| $X_2 \in \{b, d\}$    |                   | $X_1$ | .3 | .6 | .1 | 0  |
| $X_3 \in \{b, d\}$    |                   | $X_2$ | 0  | .2 | 0  | .8 |
| $X_4 \in \{a, c, d\}$ |                   | $X_3$ | 0  | .5 | 0  | .5 |
|                       |                   | $X_4$ | .4 | 0  | .3 | .3 |

Upper bound for the permanent of nonnegative matrices:

Soules ( $U^3$ )

$$\text{perm}(A) \leq \prod_{i=1}^m t_i \cdot g(s_i/t_i)$$

where  $s_i$  = sum of elements in row  $i$   
and  $t_i$  = maximum element in row  $i$

# Marginal Distributions on alldifferent

alldifferent( $X_1, X_2, X_3, X_4$ )

|                       |                                  |       |       |    |    |    |    |
|-----------------------|----------------------------------|-------|-------|----|----|----|----|
| $X_1 \in \{a, b, c\}$ |                                  |       |       | a  | b  | c  | d  |
| $X_2 \in \{b, d\}$    | $\xRightarrow{\theta_{X_1}(a)?}$ | $A :$ | $X_1$ | .3 | .6 | .1 | 0  |
| $X_3 \in \{b, d\}$    |                                  |       | $X_2$ | 0  | .2 | 0  | .8 |
| $X_4 \in \{a, c, d\}$ |                                  |       | $X_3$ | 0  | .5 | 0  | .5 |
|                       |                                  |       | $X_4$ | .4 | 0  | .3 | .3 |

Upper bound for the permanent of nonnegative matrices:

Soules ( $U^3$ )

$$\text{perm}(A) \leq \prod_{i=1}^m t_i \cdot g(s_i/t_i)$$

where  $s_i$  = sum of elements in row  $i$   
and  $t_i$  = maximum element in row  $i$

Design pattern

Use existing theoretical result

# In practice for alldifferent

- exact computation up to some threshold ( $\#vars$ )
- upper bounds otherwise

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- **table constraint**
- regular constraint (and mdd)
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

## Definition

The  $\text{table}(X, \mathcal{T})$  constraint holds if the combination of values taken by the sequence of finite-domain variables  $X = \langle x_1, x_2, \dots, x_k \rangle$  belongs to the finite set of  $k$ -tuples  $\mathcal{T}$ .

## Example

$\text{table}(\langle x_1, x_2, x_3 \rangle, \{(1, 2, 3), (1, 4, 5), (3, 2, 4), (2, 2, 2)\})$

$x_1 \in \{2, 3, 4, 5\} \quad x_2 \in \{1, 2, 3, 4, 5\} \quad x_3 \in \{1, 2, 3, 4, 5\}$

## Definition

The  $\text{table}(X, \mathcal{T})$  constraint holds if the combination of values taken by the sequence of finite-domain variables  $X = \langle x_1, x_2, \dots, x_k \rangle$  belongs to the finite set of  $k$ -tuples  $\mathcal{T}$ .

## Example

$\text{table}(\langle x_1, x_2, x_3 \rangle, \{(1, 2, 3), (1, 4, 5), (3, 2, 4), (2, 2, 2)\})$

$x_1 \in \{2, 3, \cancel{1}, \cancel{4}, \cancel{5}\} \quad x_2 \in \{\cancel{1}, 2, \cancel{3}, \cancel{4}, \cancel{5}\} \quad x_3 \in \{\cancel{1}, 2, \cancel{3}, 4, \cancel{5}\}$  domain consistent

# Domain Filtering on table

## Compact Table algorithm

Uses two data structures, as (reversible, sparse) bit-sets:

- **supports[i][v]**  
the set of tuples from  $\mathcal{T}$  such that  $x_i$  takes value  $v$ ;  
computed once at the beginning
- **supportedTuples**  
the set of tuples from  $\mathcal{T}$  currently supported by the domains of the  
variables;  
updated throughout execution

If  $\text{supportedTuples} \cap \text{supports}[i][v] = \emptyset$  then remove  $v$  from  $D(x_i)$

# Model counting on table

## model counting

**return** |supportedTuples|

i.e. the number of bits set to one

## marginals $\theta$

**foreach**  $x_i \in X$  **do**

**foreach**  $v \in D(x_i)$  **do**

$\theta_{x_i}(v) \leftarrow |\text{supportedTuples} \cap \text{supports}[i][v]|$

    normalize  $\theta_{x_i}$

# Weighted Model Counting on table

## weighted model counting

```
wcount  $\leftarrow$  0
foreach  $t \in \text{supportedTuples}$  do
    | wcount  $+= \prod_{1 \leq i \leq k} \omega_{x_i}(t[i])$ 
return wcount
```

## marginals $\theta$

precompute  $t.\text{weight}$  for each supported tuple, as above

```
foreach  $x_i \in X$  do
    | foreach  $v \in D(x_i)$  do
        |  $\theta_{x_i}(v) \leftarrow 0$ 
        | foreach  $t \in \text{supportedTuples} \cap \text{supports}[i][v]$  do
            |  $\theta_{x_i}(v) += t.\text{weight} / \omega_{x_i}(v)$ 
    | normalize  $\theta_{x_i}$ 
```

...and similarly for variants of the table constraint

negativeTable:     give forbidden tuples

shortTable:        starred tuples, e.g.  $(1, 2, \star, 3)$

Design pattern

(Compact) representation of the solution set

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- **regular constraint (and mdd)**
- sum constraint
- spanning\_tree constraint

## 3 What Is It Good For?

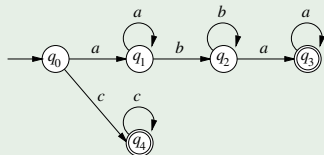
- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

## Definition

The  $\text{regular}(X, \Pi)$  constraint holds if the values taken by the (finite) sequence of finite-domain variables  $X = \langle x_1, x_2, \dots, x_k \rangle$  spell out a word belonging to the regular language defined by the deterministic finite automaton  $\Pi = (Q, \Sigma, \delta, q_0, F)$

## Example



with  $k = 4$ :

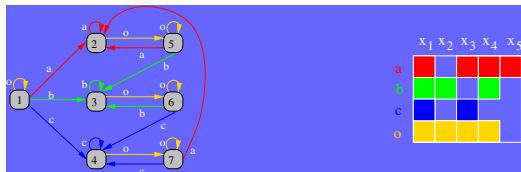
aaba

abba

abaa

cccc

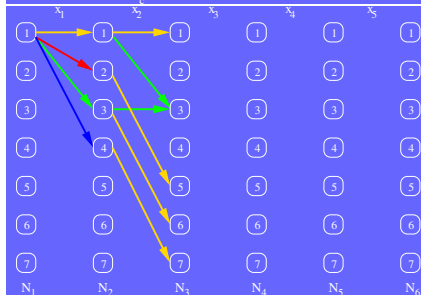
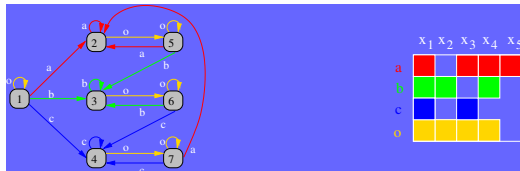
# Domain Filtering on regular



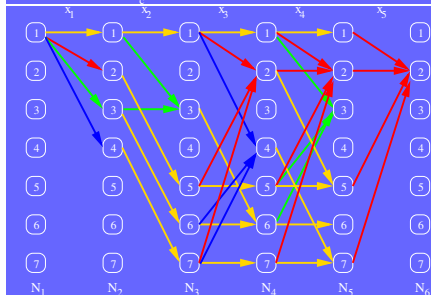
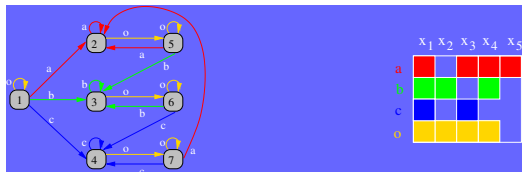
|   | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|-------|-------|-------|-------|-------|
| a |       |       |       |       |       |
| b |       |       |       |       |       |
| c |       |       |       |       |       |
| o |       |       |       |       |       |

|       | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|-------|-------|-------|-------|-------|-------|
| 1     | 1     | 1     | 1     | 1     | 1     |
| 2     | 2     | 2     | 2     | 2     | 2     |
| 3     | 3     | 3     | 3     | 3     | 3     |
| 4     | 4     | 4     | 4     | 4     | 4     |
| 5     | 5     | 5     | 5     | 5     | 5     |
| 6     | 6     | 6     | 6     | 6     | 6     |
| 7     | 7     | 7     | 7     | 7     | 7     |
| $N_1$ | $N_2$ | $N_3$ | $N_4$ | $N_5$ | $N_6$ |

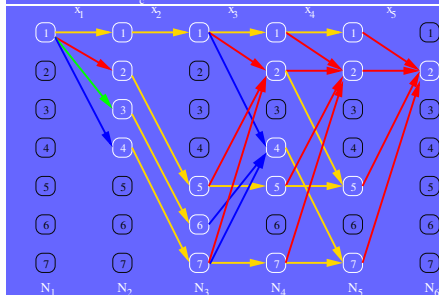
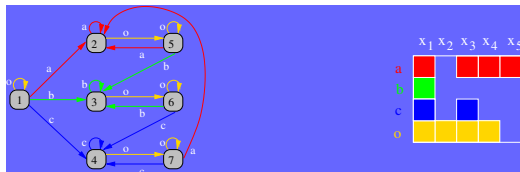
# Domain Filtering on regular



# Domain Filtering on regular



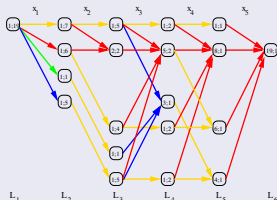
# Domain Filtering on regular



**One-to-one correspondence between paths and solutions**

# Adding information to the graph

## Layered graph



## Each node contains:

"#ip;#op"

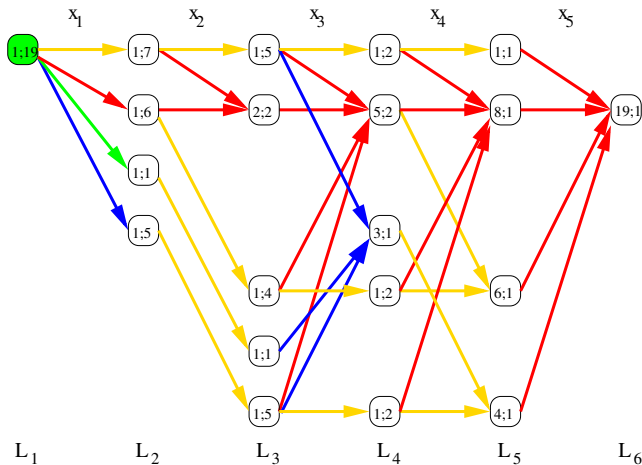
#ip nb of incoming paths  
from initial state

#op nb of outgoing paths to  
final state

## Recurrence relation

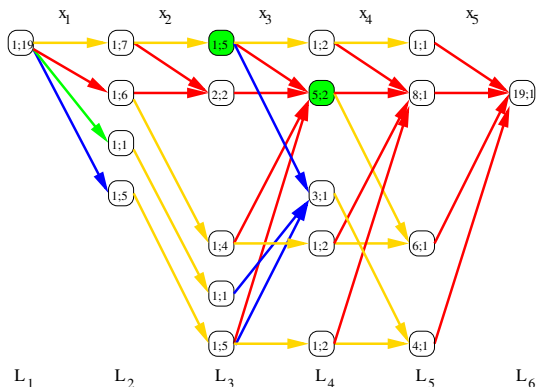
$$\begin{aligned}\#ip(1, q_0) &= 1 \\ \#ip(\ell + 1, q') &= \sum_{(v_{\ell, q}, v_{\ell+1, q'}) \in A} \#ip(\ell, q), \quad 1 \leq \ell \leq n\end{aligned}$$

# Model Counting on regular



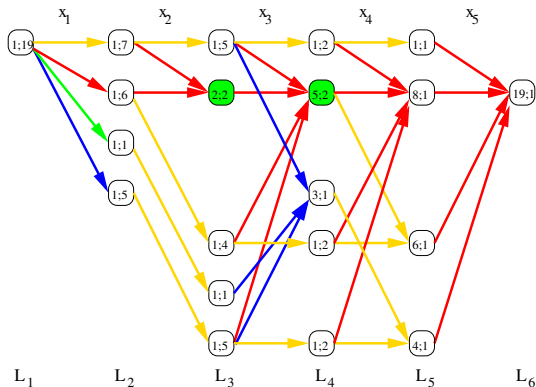
# Marginal Distribution: Counting solutions such that $x_3 = \text{red}$

$$\theta_{x_3}(\text{red}) = \frac{2}{19}$$



# Marginal Distribution: Counting solutions such that $x_3 = \text{red}$

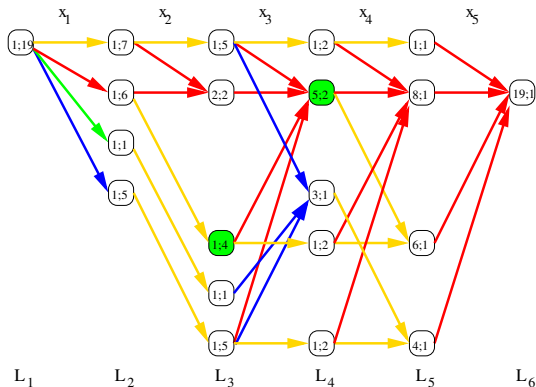
$$\theta_{x_3}(\text{red}) = \frac{2+4}{19}$$



# Marginal Distribution: Counting solutions such that $x_3 = \text{red}$

$$\theta_{x_3}(\text{red}) =$$

$$\frac{2+4+2}{19}$$

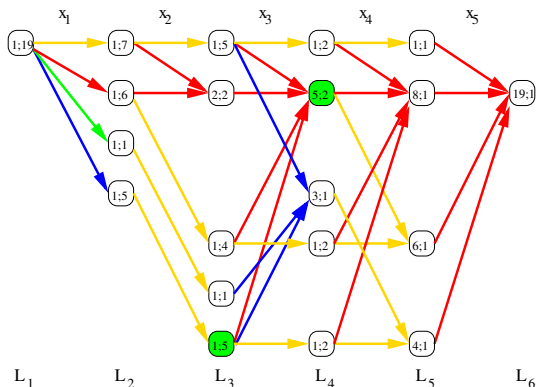


# Marginal Distribution:

Counting solutions such that  $x_3 = \text{red}$

$$\theta_{x_3}(\text{red}) =$$

$$\frac{2+4+2+2}{19} = \frac{10}{19}$$



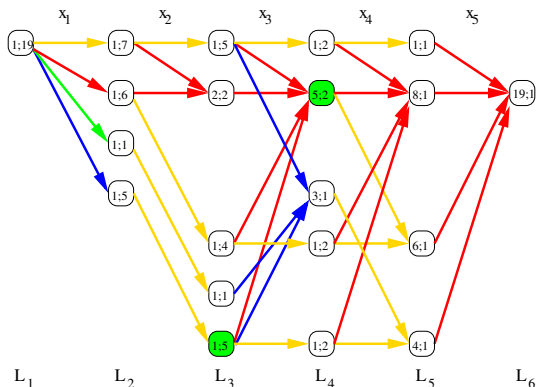
Marginal probability of  $x_3 = \text{red}$  in a solution chosen uniformly at random

# Marginal Distribution:

Counting solutions such that  $x_3 = \text{red}$

$$\theta_{x_3}(\text{red}) =$$

$$\frac{2+4+2+2}{19} = \frac{10}{19}$$

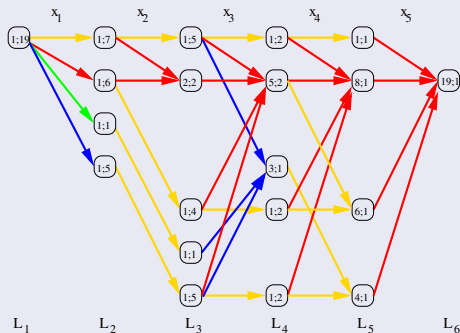


Marginal probability of  $x_3 = \text{red}$  in a solution chosen uniformly at random

**So, counting solutions doesn't cost much more here.**

# Weighted Model Counting on regular

## Layered graph



each arc  $a$  now has a positive weight  $w_a$   
weight of path = product of arc weights

Each node contains:

$\#ip$  sum of weighted incoming paths from initial state

$\#op$  sum of weighted outgoing paths to final state

## Recurrence relation

$$\#ip(1, q_0) = 1$$

$$\#ip(\ell + 1, q') = \sum_{a: (v_\ell, q, v_{\ell+1}, q') \in A} w_a \times \#ip(\ell, q), \quad 1 \leq \ell \leq n$$

another ex.: mdd constraint (multi-valued decision diagram)

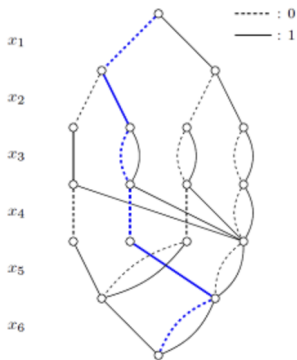


Figure 1: Exact BDD

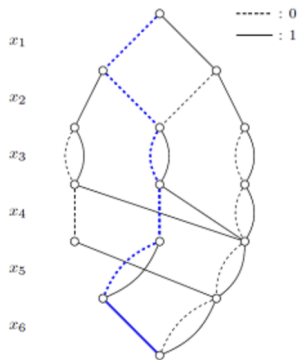


Figure 2: Relaxed BDD

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- **sum constraint**
- spanning\_tree constraint

## 3 What Is It Good For?

- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

# sum constraint

## Definition

The  $\text{sum}(\mathbf{x}, \mathbf{c}, \ell, u)$  constraint holds if  $\ell \leq \sum_{i=1}^n c_i x_i \leq u$ .

## domain filtering

To reach domain consistency (GAC) there is an algorithm<sup>a</sup> similar in spirit to the one for regular

---

<sup>a</sup>Trick, *A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints*, Ann. Oper. Res. (CPAIOR) 2003

So we can proceed as for regular constraints  
but it now runs in pseudo-polynomial time (w.r.t.  $\ell$  and  $u$ ).

Can still be fine if numerical values are not too large, and otherwise. ...

# Marginal Distributions on sum

- Express one variable in terms of the others:  
 $\ell \leq \sum_{i=1}^n c_i x_i \leq u$  is rewritten as  
 $x_j = \frac{1}{c_j} (x_{n+1} - \sum_{i=1}^{j-1} c_i x_i - \sum_{i=j+1}^n c_i x_i)$  with  $x_{n+1} \in [\ell, u]$ .
- Relax domains to intervals
- Assume values in domains are equiprobable (uniform distribution)
- $x_j$  follows normal distribution (C.L.T.)

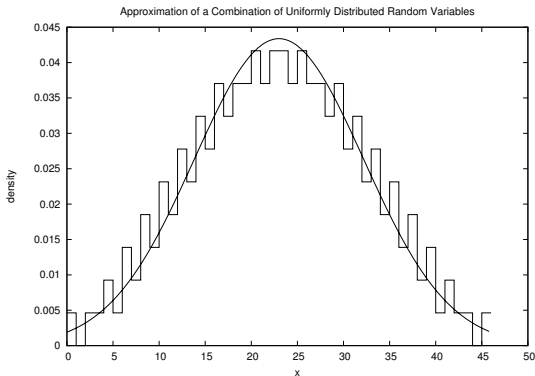
Note: that assumption doesn't hold for weighted counting

# Marginal Distributions on sum

## Example

Histogram is actual distribution of  $3x + 4y + 2z$  for  $x, y, z \in [0, 5]$ .

Curve is approximation given by Gaussian curve with mean  $\mu = 22.5$  and variance  $\sigma^2 = 84.583$ .



Design pattern

Domain relaxation

## 1 Motivation

## 2 Revisiting Some Common Constraints

- $x \neq y$  constraint
- alldifferent constraint (and circuit)
- table constraint
- regular constraint (and mdd)
- sum constraint
- **spanning\_tree** constraint

## 3 What Is It Good For?

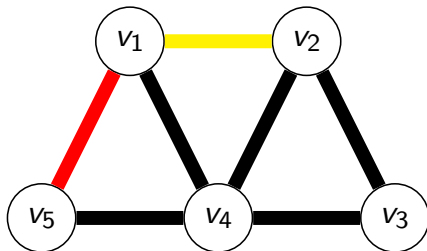
- CP-BP Framework
- Search Guidance
- Sampling Solutions
- Neuro-Symbolic AI

## 4 Conclusion

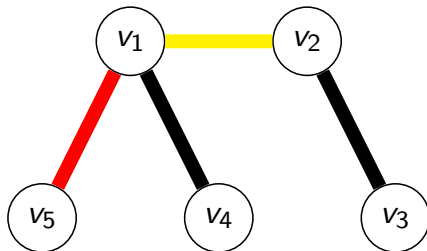
# spanning\_tree constraint

## Definition

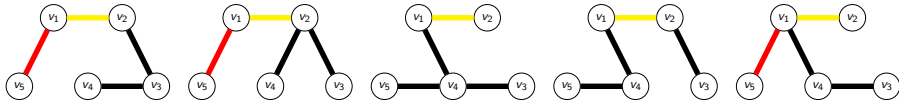
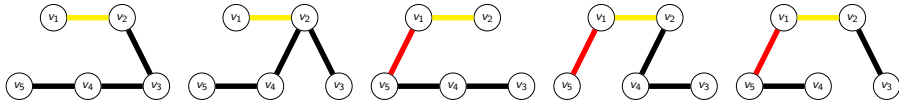
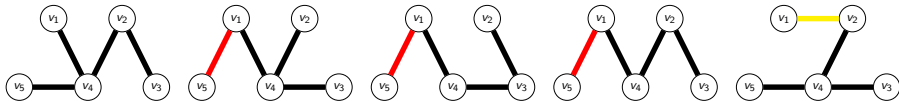
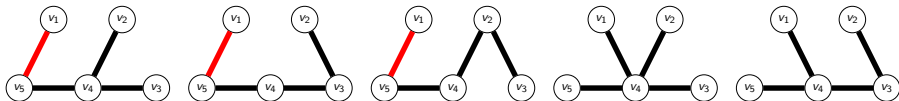
Given an undirected graph  $G(V, E)$  and set variable  $T \subseteq E$ , constraint  $\text{spanning\_tree}(G, T)$  restricts  $T$  to be a spanning tree of  $G$ .



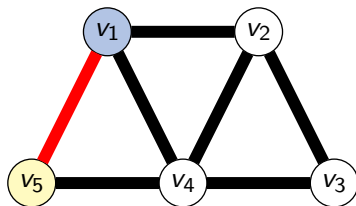
(a)  $G$



(b)  $T$



# Matrix-Tree Theorem



Laplacian matrix of the graph:

$$\begin{pmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{pmatrix}$$

# Model Counting on `spanning_tree`

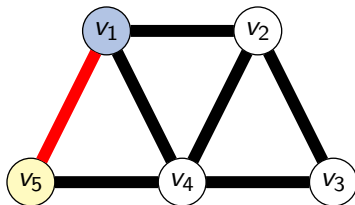
## Kirchhoff's Matrix-Tree Theorem

Any minor of the Laplacian is equal to the number of spanning trees (in absolute value)

$$\begin{vmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{vmatrix} = 21$$

# Marginal Distribution: Counting solutions excluding a given edge $(i,j)$

$$\frac{\#\text{spanning trees}(G \setminus \{(1,5)\})}{\#\text{spanning trees}(G)}$$



$$\begin{pmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

# Marginal Distribution: Counting solutions excluding a given edge $(i,j)$

let's take a minor with row/column  $i$  removed (here,  $i = 1$ ) :

$$\begin{vmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{vmatrix}$$

this determinant differs in only one entry from that for  $G$

# Marginal Distribution: Counting solutions excluding a given edge $(i, j)$

## Sherman-Morrison formula

$$\det(M') = (1 + e_j^\top M^{-1}(u - (M)_j))\det(M).$$

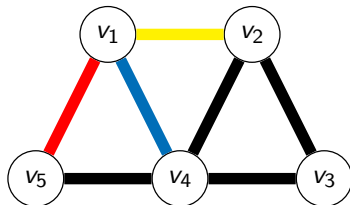
In our case this simplifies to  $\det(M') = (1 - m_{jj}^{-1})\det(M)$ .

So

$$\frac{\#\text{spanning trees}(G \setminus \{(i, j)\})}{\#\text{spanning trees}(G)} = \frac{(1 - m_{jj}^{-1})\det(M)}{\det(M)} = 1 - m_{jj}^{-1}$$

# Marginal Distribution: Counting solutions excluding a given edge $(i,j)$

One matrix inversion for all edges incident to a given vertex

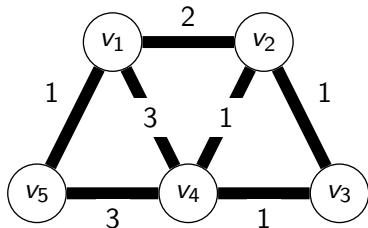


## Example

Let  $M$  be the sub-matrix of  $L$  obtained by removing its first row and

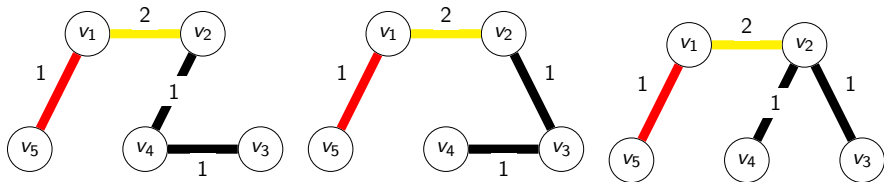
column as before. Then  $M^{-1} = \begin{pmatrix} 12/21 & 9/21 & 8/21 & 3/21 \\ 9/21 & 19/21 & 8/21 & 4/21 \\ 6/21 & 8/21 & 10/21 & 5/21 \\ 3/21 & 4/21 & 5/21 & 13/21 \end{pmatrix}$

# minimum\_spanning\_tree constraint

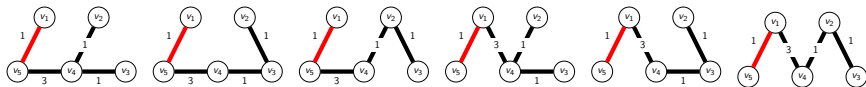


- 3 spanning trees of cost 5.
- 6 spanning trees of cost 6.
- 7 spanning trees of cost 7.
- 3 spanning trees of cost 8.
- 2 spanning trees of cost 9.

# minimum\_spanning\_tree constraint

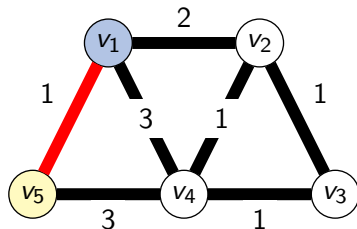


Trees of cost 5



Trees of cost 6

# Generalized Matrix-Tree Theorem



Generalized Laplacian matrix of the graph:

$$\begin{pmatrix} x^2 + x^3 + x^1 & -x^2 & 0 & -x^3 & -x^1 \\ -x^2 & x^2 + 2x^1 & -x^1 & -x^1 & 0 \\ 0 & -x^1 & 2x^1 & -x^1 & 0 \\ -x^3 & -x^1 & -x^1 & 2x^3 + 2x^1 & -x^3 \\ -x^1 & 0 & 0 & -x^3 & x^1 + x^3 \end{pmatrix}$$

# Generalized Matrix-Tree Theorem

$$\begin{vmatrix}
 x^2 + x^3 + x^1 & -x^2 & 0 & -x^3 & -x^1 \\
 -x^2 & x^2 + 2x^1 & -x^1 & -x^1 & 0 \\
 0 & -x^1 & 2x^1 & -x^1 & 0 \\
 -x^3 & -x^1 & -x^1 & 2x^3 + 2x^1 & -x^3 \\
 -x^1 & 0 & 0 & -x^3 & x^1 + x^3
 \end{vmatrix}$$

$$= 3x^5 + 6x^6 + 7x^7 + 3x^8 + 2x^9$$

# Generalized Matrix-Tree Theorem

$$3x^5 + 6x^6 + 7x^7 + 3x^8 + 2x^9$$

- 3 spanning trees of cost 5
- 6 spanning trees of cost 6
- 7 spanning trees of cost 7
- 3 spanning trees of cost 8
- 2 spanning trees of cost 9

# Generalized Matrix-Tree Theorem

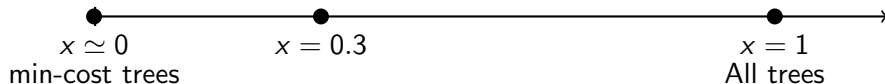
$$3x^5 + 6x^6 + 7x^7 + 3x^8 + 2x^9$$

- 3 spanning trees of cost 5
- 6 spanning trees of cost 6
- 7 spanning trees of cost 7
- 3 spanning trees of cost 8
- 2 spanning trees of cost 9

# Counting (good) solutions

In practice we don't compute determinants or inverses over matrices with polynomial entries:

we fix  $x$  to some real value in  $]0, 1]$ ...



... fall back to scalar entries and then invert some matrices.

Design pattern

Use existing theoretical result

and many more constraints...

### other constraints currently equipped in MiniCPBP

Absolute, Among, AtLeast, AtMost, BinPacking, Cardinality, CostRegular, Element, Equal, Exactly, Grammar, Inverse, IsEqual, IsEqualVar, IsLessOrEqual, IsLessOrEqualVar, IsOr, LessOrEqual, LexLess, Maximum, Minimum, Modulo, NValues, Or, Pow, Prod, Quotient, SumModP

...though some would deserve better than their current decomposition

...and famous others are missing (e.g. for scheduling)

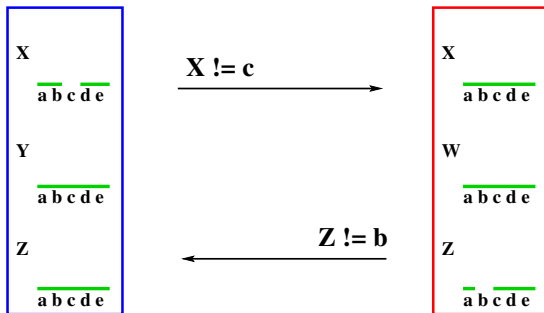
# Outline

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - Sampling Solutions
  - Neuro-Symbolic AI
- 4 Conclusion

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - Sampling Solutions
  - Neuro-Symbolic AI
- 4 Conclusion

Moving beyond  
the usual constraint propagation  
to  
belief (marginal) propagation

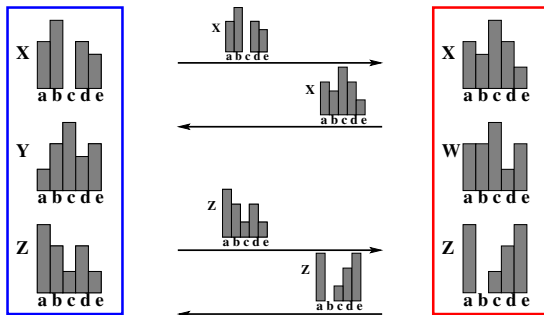
# Propagation



## constraint propagation

- propagate non-supported values for single variables
- each constraint revises its supports considering reduced domains
- iterate until fixpoint

# Propagation



## belief propagation

- propagate **marginal distributions** over single variables
- each constraint revises its marginals
- iterate until **some stopping criterion**

# Belief Propagation

Alternates between sending variable-to-constraint ( $\mu_{v \rightarrow c}$ ) and constraint-to-variable ( $\mu_{c \rightarrow v}$ ) messages

from variable to constraint

$$\mu_{v \rightarrow c}(d) = \prod_{c' \in N(v) \setminus \{c\}} \mu_{c' \rightarrow v}(d)$$

where  $d \in D(v)$

and  $N(v)$  are the constraints in which  $v$  appears

# Belief Propagation

Alternates between sending variable-to-constraint ( $\mu_{v \rightarrow c}$ ) and constraint-to-variable ( $\mu_{c \rightarrow v}$ ) messages

from constraint to variable

$$\mu_{c \rightarrow v}(d) = \sum_{\substack{x \models c \\ x \models (v=d)}} \prod_{\substack{v' \in N(c) \setminus \{v\} \\ x \models (v'=d')}} \mu_{v' \rightarrow c}(d')$$

where  $N(c)$  is the scope of  $c$   
and  $x$  a tuple from the Cartesian product of the domains of vars in  $N(c)$

Corresponds to *weighted model counting* on  $c$ .

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i `alldifferent(a, b, c)`

ii  $a + b + c + d = 7$

iii  $c \leq d$

What can we filter? ... nothing!

Still, let's see what we can do with CP-BP.

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

- i **alldifferent**( $a, b, c$ )
- ii  $a + b + c + d = 7$
- iii  $c \leq d$

$\theta_c^{iii}(3)$

| support (solution) | weight |
|--------------------|--------|
| $d = 3$            | 1      |
| $d = 4$            | 1      |
| <hr/>              |        |
| $\sum = 2$         |        |

2 out of 10 solutions to *iii*

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i  $\text{alldifferent}(a, b, c)$

ii  $a + b + c + d = 7$

iii  $c \leq d$

|     |                  | 1     | 2    | 3    | 4    |
|-----|------------------|-------|------|------|------|
| $a$ | $\theta_a^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_a^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
| $b$ | $\theta_b^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_b^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
| $c$ | $\theta_c^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_c^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
|     | $\theta_c^{iii}$ | 4/10  | 3/10 | 2/10 | 1/10 |
| $d$ | $\theta_d^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
|     | $\theta_d^{iii}$ | 1/10  | 2/10 | 3/10 | 4/10 |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i allifferent( $a, b, c$ )

ii  $a + b + c + d = 7$

iii  $c \leq d$

|     |                  | 1     | 2    | 3    | 4    |
|-----|------------------|-------|------|------|------|
| $a$ | $\theta_a^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_a^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
| $b$ | $\theta_b^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_b^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
| $c$ | $\theta_c^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_c^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
|     | $\theta_c^{iii}$ | 4/10  | 3/10 | 2/10 | 1/10 |
| $d$ | $\theta_d^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
|     | $\theta_d^{iii}$ | 1/10  | 2/10 | 3/10 | 4/10 |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i  $\text{alldifferent}(a, b, c)$

ii  $a + b + c + d = 7$

iii  $c \leq d$

|     |                  | 1      | 2      | 3     | 4     |
|-----|------------------|--------|--------|-------|-------|
| $c$ | $\theta_c^i$     | 1/4    | 1/4    | 1/4   | 1/4   |
|     | $\theta_c^{ii}$  | 10/20  | 6/20   | 3/20  | 1/20  |
|     | $\theta_c^{iii}$ | 4/10   | 3/10   | 2/10  | 1/10  |
|     | $\theta_c$       | 40/800 | 18/800 | 6/800 | 1/800 |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i **alldifferent**( $a, b, c$ )

ii  $a + b + c + d = 7$

iii  $c \leq d$

|     |                  | 1     | 2    | 3    | 4    |
|-----|------------------|-------|------|------|------|
| $c$ | $\theta_c^i$     | 1/4   | 1/4  | 1/4  | 1/4  |
|     | $\theta_c^{ii}$  | 10/20 | 6/20 | 3/20 | 1/20 |
|     | $\theta_c^{iii}$ | 4/10  | 3/10 | 2/10 | 1/10 |
|     | $\theta_c$       | .62   | .28  | .09  | .01  |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i **alldifferent**( $a, b, c$ )

ii  $a + b + c + d = 7$

iii  $c \leq d$

Iteration 1

|            | 1          | 2          | 3          | 4          |
|------------|------------|------------|------------|------------|
| $\theta_a$ | .50        | .30        | .15        | .05        |
| $\theta_b$ | .50        | .30        | .15        | .05        |
| $\theta_c$ | <b>.62</b> | <b>.28</b> | <b>.09</b> | <b>.01</b> |
| $\theta_d$ | .29        | .34        | .26        | .11        |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i  $\text{alldifferent}(a, b, c)$

ii  $a + b + c + d = 7$

iii  $c \leq d$

|     |                 | 1     | 2    | 3    | 4    |
|-----|-----------------|-------|------|------|------|
| $d$ | $\theta_d^{ii}$ | 10/20 | 6/20 | 3/20 | 1/20 |

$\theta_c^{iii}(3)$

| support (solution) | weight |
|--------------------|--------|
| $d = 3$            | 3/20   |
| $d = 4$            | 1/20   |
| $\Sigma = 4/20$    |        |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i **alldifferent**( $a, b, c$ )

ii  $a + b + c + d = 7$

iii  $c \leq d$

Iteration 10

|            | 1          | 2          | 3          | 4   |
|------------|------------|------------|------------|-----|
| $\theta_a$ | .01        | <b>.52</b> | <b>.46</b> | .01 |
| $\theta_b$ | .01        | <b>.52</b> | <b>.46</b> | .01 |
| $\theta_c$ | <b>.98</b> | .02        | .00        | .00 |
| $\theta_d$ | <b>.90</b> | .10        | .00        | .00 |

# Belief Propagation over the CP Model

## Example

constraints over variables  $a, b, c, d \in \{1, 2, 3, 4\}$ :

i **alldifferent**( $a, b, c$ )

ii  $a + b + c + d = 7$

iii  $c \leq d$

True marginals (solutions 2,3,1,1 and 3,2,1,1)

|            | 1 | 2   | 3   | 4 |
|------------|---|-----|-----|---|
| $\theta_a$ | 0 | 1/2 | 1/2 | 0 |
| $\theta_b$ | 0 | 1/2 | 1/2 | 0 |
| $\theta_c$ | 1 | 0   | 0   | 0 |
| $\theta_d$ | 1 | 0   | 0   | 0 |

# Outline

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - **Search Guidance**
  - Sampling Solutions
  - Neuro-Symbolic AI
- 4 Conclusion

# Branching for Combinatorial Search

Binary branching:  $x_i = d_j \quad \vee \quad x_i \neq d_j$

maxSD (no CP-BP, model counting)

- 1 choose variable maximizing the largest marginal:

$$i = \operatorname{argmax}_{x \in X} \max_{d \in D(x)} \theta_x(d)$$

- 2 choose value maximizing the marginal:

$$j = \operatorname{argmax}_{d \in D(x_i)} \theta_{x_i}(d)$$

# Branching for Combinatorial Search

Binary branching:  $x_i = d_j \quad \vee \quad x_i \neq d_j$

## min-entropy (CP-BP, weighted model counting)

- 1 choose variable minimizing the entropy of the marginal distribution over its domain:

$$i = \operatorname{argmin}_{x \in X} - \sum_{d \in D(x)} \theta_x(d) \log(\theta_x(d))$$

- 2 choose value maximizing the marginal:

$$j = \operatorname{argmax}_{d \in D(x_i)} \theta_{x_i}(d)$$

# Outline

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - **Sampling Solutions**
  - Neuro-Symbolic AI
- 4 Conclusion

# (Near-)Uniform Sampling

## sample solutions uniformly at random

Given true marginal distributions:

- pick any variable;
- pick a value at random according to its marginal distribution;
- adjust distributions and repeat.

## sample solutions **almost** uniformly at random

Given **estimated** marginal distributions from CP-BP solver:

- 
- ⋮
-

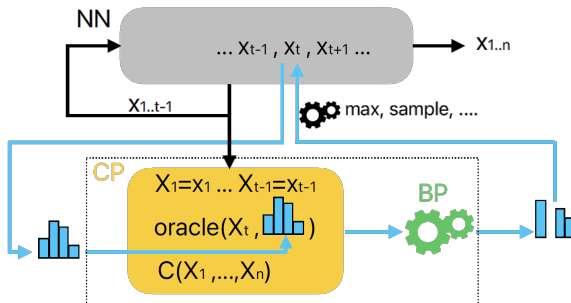
- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - Sampling Solutions
  - **Neuro-Symbolic AI**
- 4 Conclusion

Neural networks dealing with hard/deterministic structure  
(CP4ML perspective)

Ex: safe robotics, clinical text (vision, speech therapy)

Data-driven + Model-driven

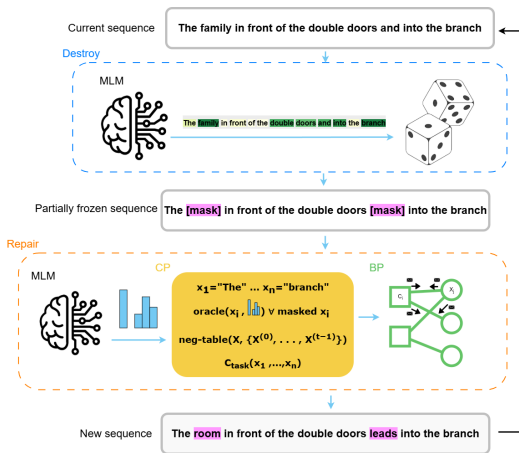
Marginals-augmented CP combines more naturally with NN outputs.



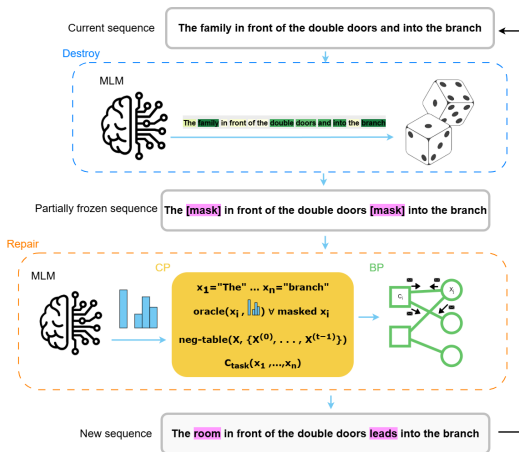
generated sequence satisfies constraints  
without straying too far from training data

Has been applied to generate text, music, and molecules.

# Neurosymbolic LNS

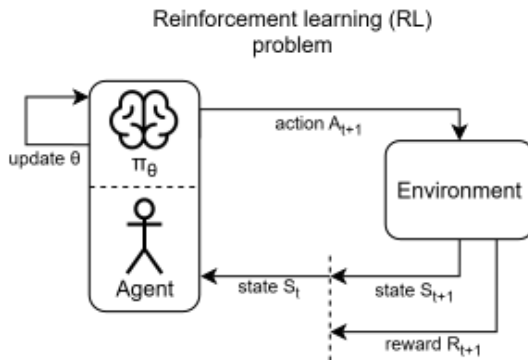


# Neurosymbolic LNS



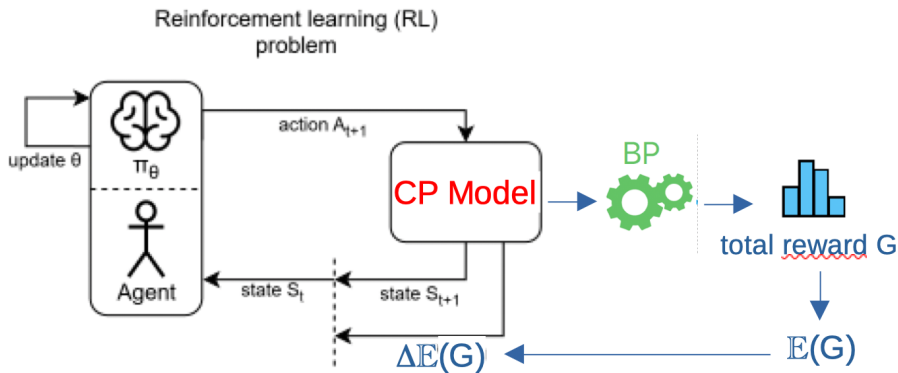
To know more... please attend Arnaud's talk today at 16:50

# Reward shaping in Reinforcement Learning



The problem of *sparse* rewards

# Reward shaping in Reinforcement Learning



# Deriving a loss function to train neural networks

## Semantic Loss

"...to fill in the gap between the continuous world of feedforward DNNs and the symbolic world of propositional logic"

Defined as **weighted model counting** on a sentence in propositional logic.

# Outline

- 1 Motivation
- 2 Revisiting Some Common Constraints
  - $x \neq y$  constraint
  - alldifferent constraint (and circuit)
  - table constraint
  - regular constraint (and mdd)
  - sum constraint
  - spanning\_tree constraint
- 3 What Is It Good For?
  - CP-BP Framework
  - Search Guidance
  - Sampling Solutions
  - Neuro-Symbolic AI
- 4 Conclusion

# Conclusion

**Q-** What is the distinctive driving force behind CP?

**A-** More direct access to problem structure from high-level constraints

Let's take advantage of this strength  
and exploit that structure in new ways.

MiniCPBP: <https://github.com/PesantGilles/MiniCPBP>