Exploiting Combinatorial Structure in Constraint Programming: Beyond Domain Filtering to Counting and Marginals

Gilles Pesant

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

Satisfiability: Theory, Practice, and Beyond
Simons Institute, UC Berkeley, USA
April 17-21 2023

Outline

1. Exposed Combinatorial Structure in CP
2. (Weighted) Counting
   - Compact representation of the solution set
   - Sampling (interleaved with domain filtering)
   - Use existing theoretical result
   - Domain relaxation
3. CP-BP Framework
   - A Small Example
   - Branching for Combinatorial Search
   - (Near-)Uniform Sampling
   - Neuro-Symbolic AI
4. Conclusion

Outline

1. Exposed Combinatorial Structure in CP
2. (Weighted) Counting
   - Compact representation of the solution set
   - Sampling (interleaved with domain filtering)
   - Use existing theoretical result
   - Domain relaxation
3. CP-BP Framework
   - A Small Example
   - Branching for Combinatorial Search
   - (Near-)Uniform Sampling
   - 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

Constraint Programming Models

Round-robin tournament (TTPPV)

```
array[Teams,Rounds] of var Teams:  opponent;
array[Teams,Rounds] of var 1..2:  venue;
forall (i in Teams, k in Rounds) (venue[i,k] = pv[i,opponent[i,k]]);
forall (i in Teams, k in Rounds) (opponent[i,k] ≠ i);
forall (i in Teams, k in Rounds) (opponent[opponent[i,k],k] = i);
forall (i in Teams) (alldifferent([opponent[i,k] | k in Rounds]));
forall (i in Teams) (regular( [venue[i,k] | k in Rounds], automaton));
```

Moving furniture

```
array[Objects] of var 0..availableTime:  start;
var 0..availableTime:  end;
cumulative(start, duration, handlers, availableHandlers);
cumulative(start, duration, trolleys, availableTrolleys);
forall (o in Objects) (start[o] + duration[o] ≤ end);
solve minimize end;
```

Constraint Programming

Q- What is the distinctive driving force behind CP?

A- Direct access to problem structure from high-level constraints

Constraint Programming

Q- What is the distinctive driving force behind CP?

A- 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- Direct access to problem structure from high-level constraints

How does one nominate these high-level constraints?

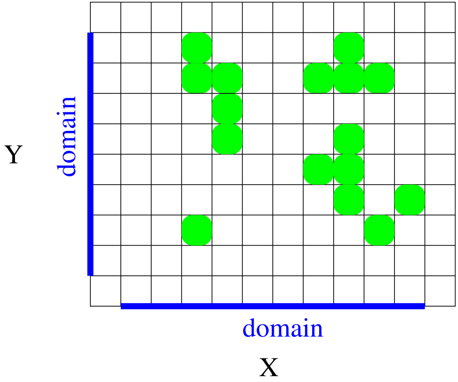
- complex enough to provide structural insight
- simple enough for some desired computing tasks to remain tractable

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?"
- $\cdots$
- "How many solutions are there to c?"
- "How many solutions in which $x = d$?"

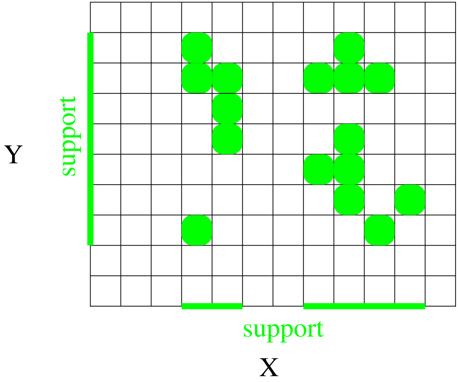
Using Global Constraints (a.k.a. Structure) in CP

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



Using Global Constraints (a.k.a. Structure) in CP

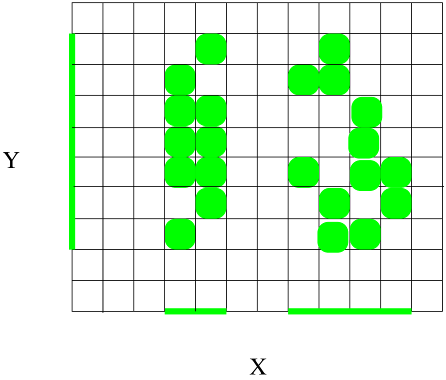
Consider a simple constraint on finite-domain variables X and Y.



domain filtering $\equiv$ projecting solutions on individual variables

Using Global Constraints (a.k.a. Structure) in CP

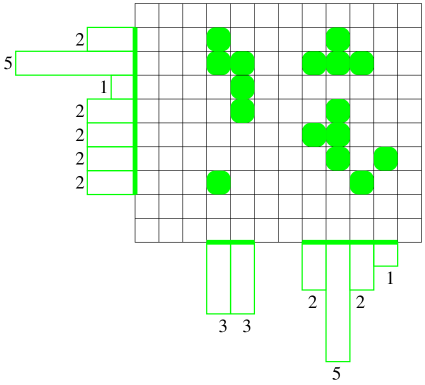
Consider a simple constraint on finite-domain variables X and Y.



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

Using Global Constraints (a.k.a. Structure) in CP

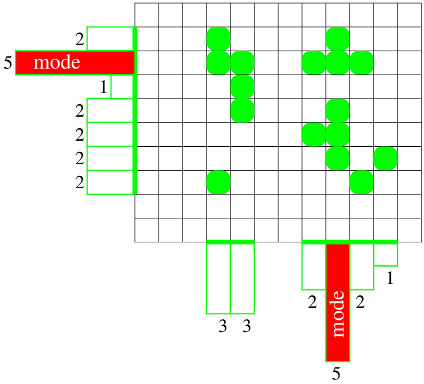
Now consider the set of solutions as a multivariate discrete distribution.



marginals ≡ projecting that distribution on individual variables

Using Global Constraints (a.k.a. Structure) in CP

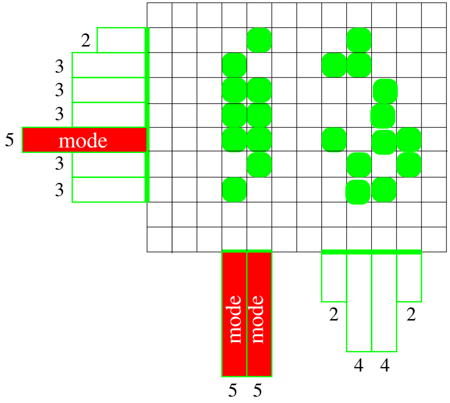
Now consider the set of solutions as a multivariate discrete distribution.



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

Using Global Constraints (a.k.a. Structure) in CP

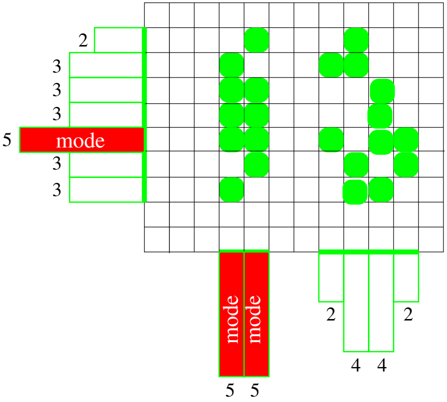
Now consider the set of solutions as a multivariate discrete distribution.



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

Using Global Constraints (a.k.a. Structure) in CP

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



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

Outline

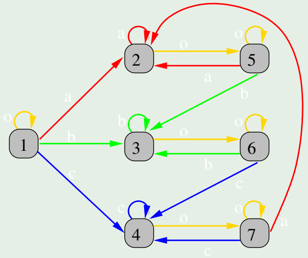
Outline

regular constraint

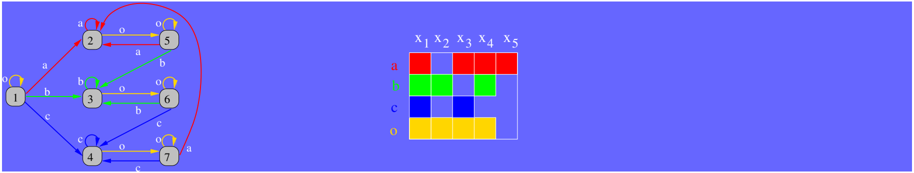
Definition

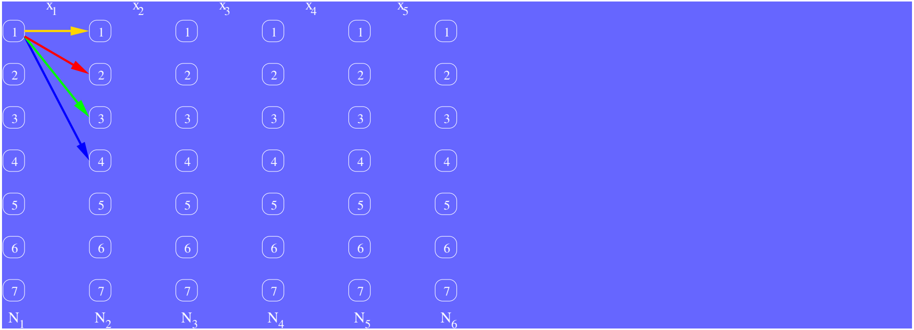
The regular(X, Π) constraint holds if the values taken by the (finite) sequence of finite-domain variables $X = \langle x_1, x_2, \ldots, x_k \rangle$ spell out a word belonging to the regular language defined by the deterministic finite automaton $\Pi = (Q, \Sigma, \delta, q_0, F)$

Example

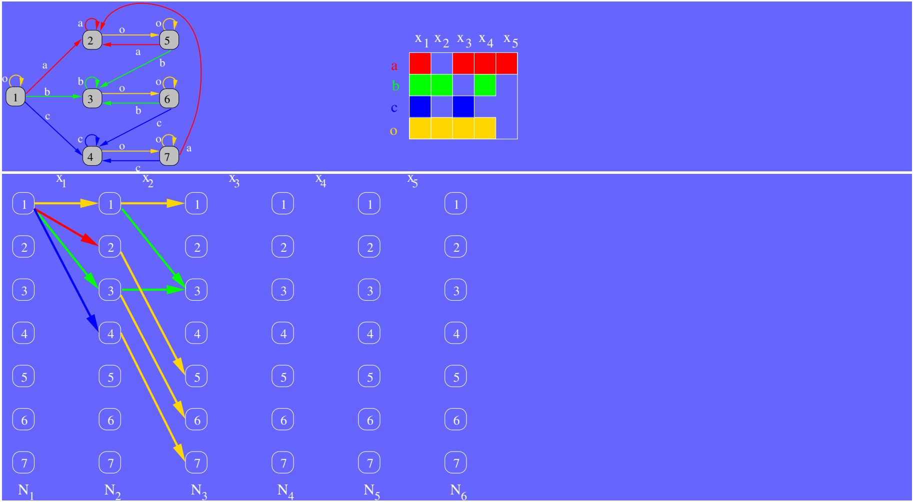


Domain Filtering on regular constraints

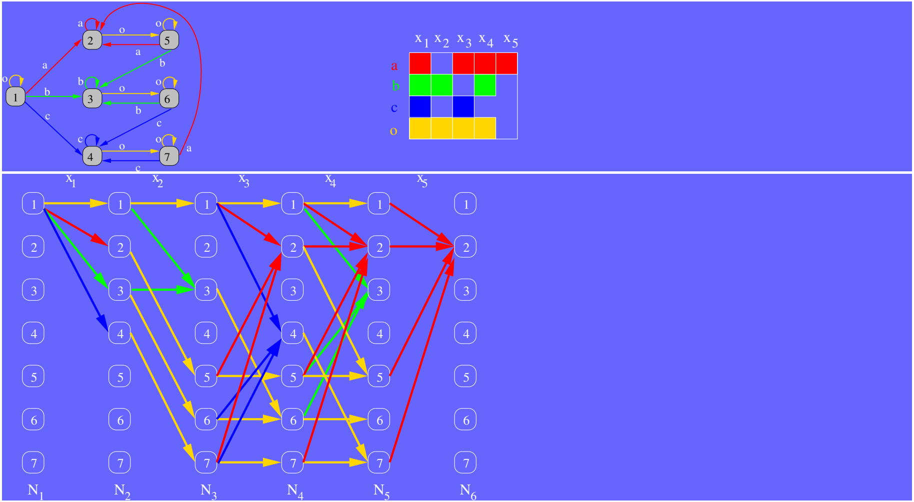




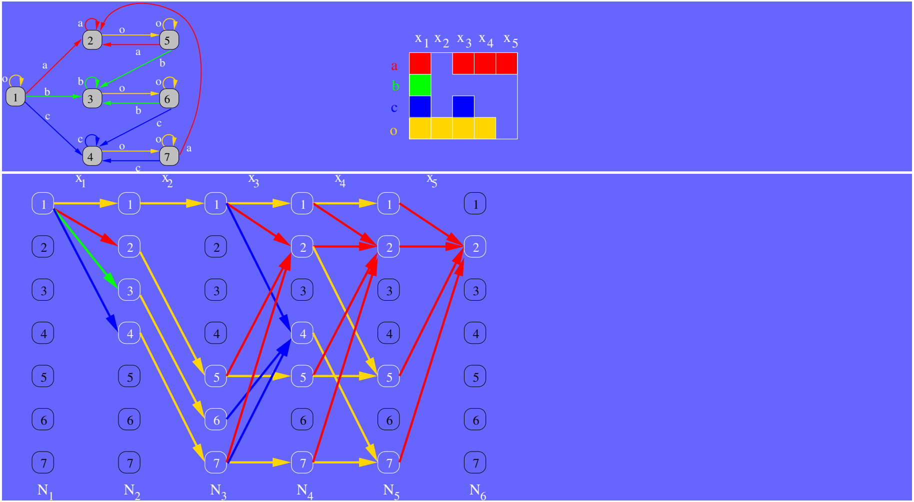
Domain Filtering on regular constraints



Domain Filtering on regular constraints



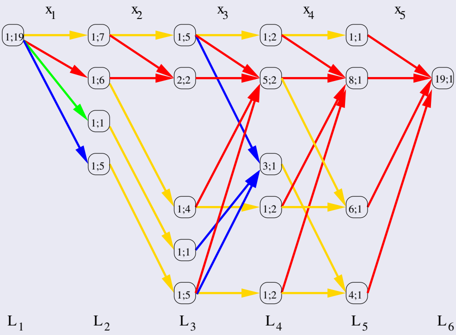
Domain Filtering on regular constraints



One-to-one correspondence between paths and solutions

Counting Solutions of `regular` constraints

Layered graph



Each node contains:

"#ip;#op"

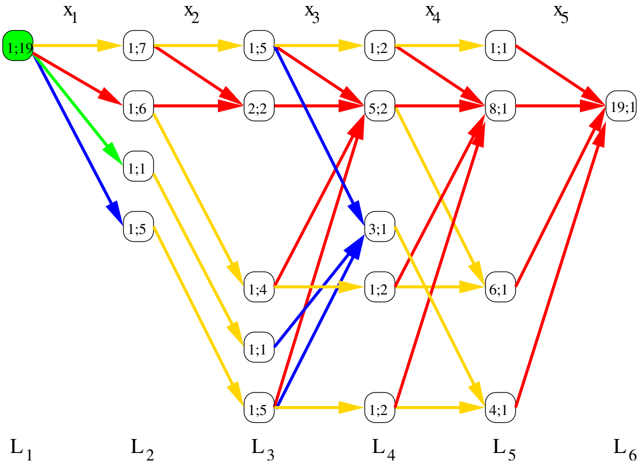
#ip nb of incoming paths from initial state

#op nb of outgoing paths to final state

Recurrence relation

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

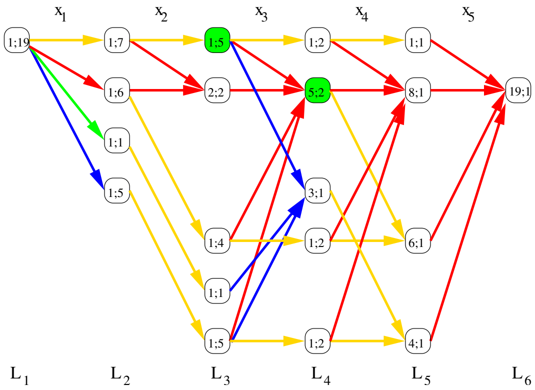
Counting All Solutions



Counting solutions such that x_3 =red (marginal, bias)

$\theta_{x_3}(red) =$

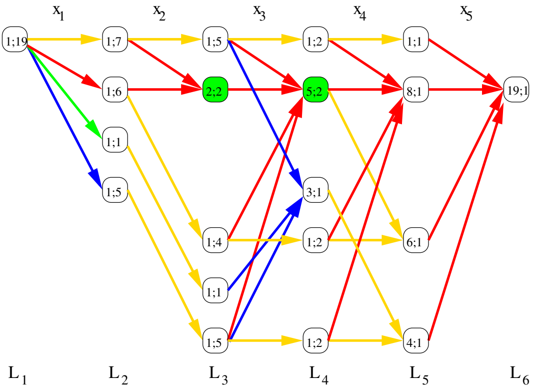
$\frac{2}{19}$



Counting solutions such that $x_3 =$red (marginal, bias)

$\theta_{x_3}(red) =$

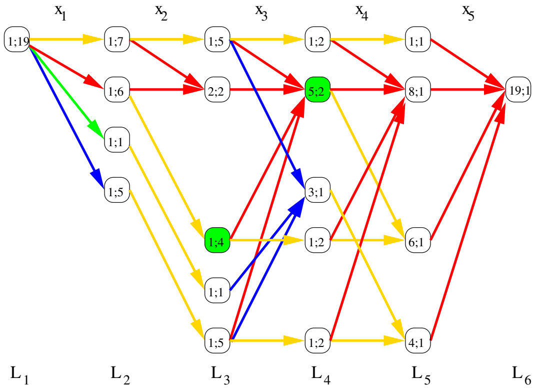
$\frac{2+4}{19}$



Counting solutions such that $x_3 =$red (marginal, bias)

$\theta_{x_3}(red) =$

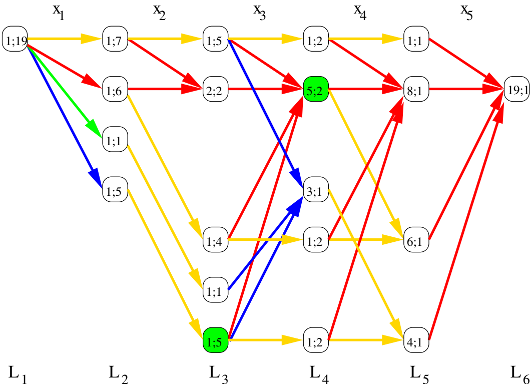
$\frac{2+4+2}{19}$



Counting solutions such that x_3 =red (marginal, bias)

$\theta_{x_3}(red) =$

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

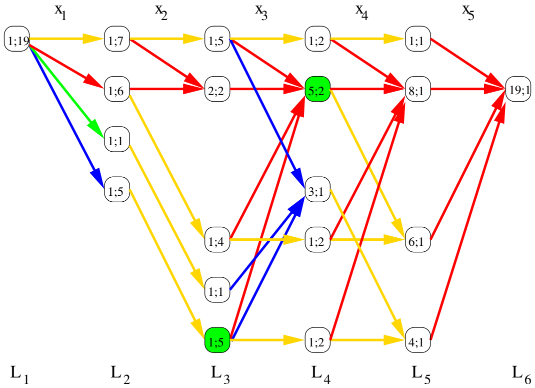


Marginal probability of x_3 =red in a solution chosen uniformly at random

Counting solutions such that x_3 =red (marginal, bias)

$\theta_{x_3}(red) =$

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

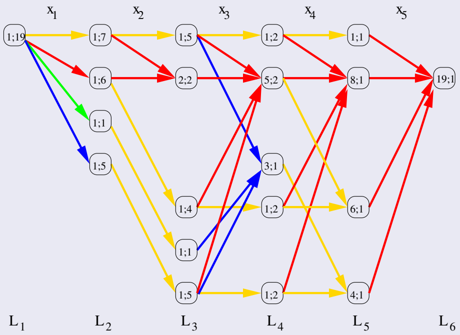


Marginal probability of x_3 =red in a solution chosen uniformly at random

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

Weighted Counting

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

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

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

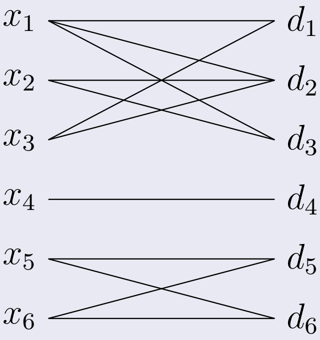
Outline

alldifferent constraint

Definition

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

Value graph



Adjacency Matrix

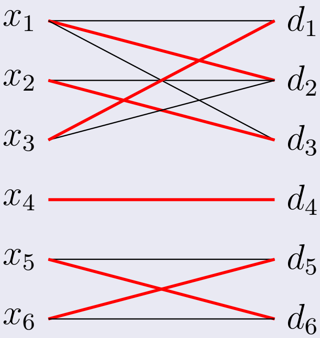
$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

alldifferent constraint

Definition

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

Value 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

Domain filtering

bipartite graph matching + depth-first search

`alldifferent` constraint

Domain filtering

bipartite graph matching + depth-first search

But now counting solutions cost significantly more

Counting with `alldifferent`

Its number of solutions is the same as...

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

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

Counting with `alldifferent`

Its number of solutions is the same as. . .

- the number of perfect matchings in the bipartite graph
- the permanent of the adjacency matrix
  $per(A) = \sum_{\sigma \in S_n} \prod_i a_{i,\sigma(i)}$

Remark

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

Sampling

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

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

$\|W\|$
3

Rasmussen's estimator

Example

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

$|W|$
3
1

Rasmussen's estimator

Example

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

$\lvert W \rvert$
3
1
1

Rasmussen's estimator

Example

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

$\lvert W \rvert$
3
1
1
1

Example

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

$\lvert W \rvert$
3
1
1
1
2

Rasmussen's estimator

Example

$$\mathbf{A} = \begin{pmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & 1 & \end{pmatrix} \qquad \begin{matrix} |W| \\ 3 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \end{matrix}$$

Rasmussen's estimator

Example

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

$\lvert W \rvert$
3
1
1
1
2
1

$X_A = 6$

Rasmussen's estimator properties

Properties

- It works well for "almost" all dense matrices
- Poor results in some special cases

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

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

helps avoiding dead ends
($W = \emptyset$)

Adding domain filtering

Modified Rasmussen

if $n = 0$ **then**
  $X_A = 1$
else
  Domain filtering on A
  Choose i u.a.r. from $\{1 \ldots n\}$
  $W = \{j : a_{i,j} = 1\}$
  if $W = \emptyset$ **then**
    $X_A = 0$
  else
    Choose j u.a.r. from W
    Compute $X_{A_{i,j}}$
    $X_A = |W| \cdot X_{A_{i,j}}$

helps avoiding dead ends
$(W = \emptyset)$

Number of solutions

$\#\texttt{alldiff}(x_1, \ldots, x_n) \approx E(X_A)$

Marginals by sampling

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

Weighted Counting with `alldifferent`

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

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

Outline

Counting with `alldifferent`

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

$$\begin{array}{lcl} X_1 & \in & \{a, b, c\} \\ X_2 & \in & \{b, d\} \\ X_3 & \in & \{b, d\} \\ X_4 & \in & \{a, c, d\} \end{array} \implies A: \begin{array}{c|cccc} & \text{a} & \text{b} & \text{c} & \text{d} \\ \hline X_1 & 1 & 1 & 1 & 0 \\ X_2 & 0 & 1 & 0 & 1 \\ X_3 & 0 & 1 & 0 & 1 \\ X_4 & 1 & 0 & 1 & 1 \end{array}$$

There are known upper bounds for the permanent of 0-1 matrices.

Counting with `alldifferent`

Minc-Brègman

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

where r_i = number of 1's in row i

Liang-Bai

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

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

Weighted Counting with alldifferent

alldifferent(X_1, X_2, X_3, X_4)

$$\begin{array}{lcl} X_1 & \in & \{a, b, c\} \\ X_2 & \in & \{b, d\} \\ X_3 & \in & \{b, d\} \\ X_4 & \in & \{a, c, d\} \end{array} \implies A:$$

	a	b	c	d
X_1	.3	.6	.1	0
X_2	0	.2	0	.8
X_3	0	.5	0	.5
X_4	.4	0	.3	.3

Upper bound for the permanent of nonnegative matrices:

Soules (U^3)

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

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

Weighted Counting with alldifferent

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

$$\begin{array}{lll} X_1 & \in & \{a, b, c\} \\ X_2 & \in & \{b, d\} \\ X_3 & \in & \{b, d\} \\ X_4 & \in & \{a, c, d\} \end{array} \quad \overset{\theta_{X_1}(a)?}{\Longrightarrow} \quad A:$$

	a	b	c	d
X_1	.3	.6	.1	0
X_2	0	.2	0	.8
X_3	0	.5	0	.5
X_4	.4	0	.3	.3

Upper bound for the permanent of nonnegative matrices:

Soules (U^3)

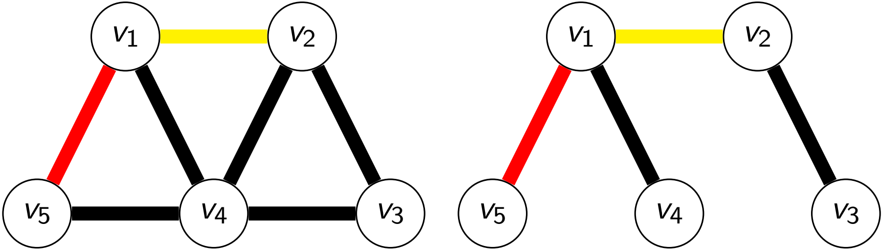
$$perm(A) \leq \prod_{i=1}^{m} t_i \cdot g(s_i/t_i)$$

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

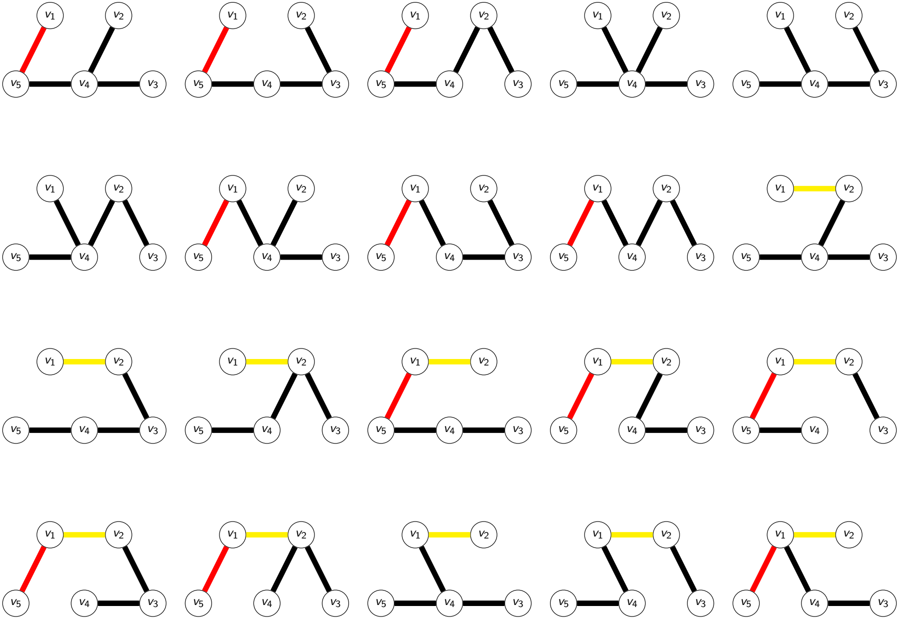
spanning_tree constraint

Definition

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

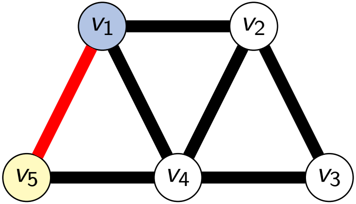


(a) G (b) T



v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3
v1 v2 v5 v4 v3

Matrix-Tree Theorem



Laplacian matrix of the graph:

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

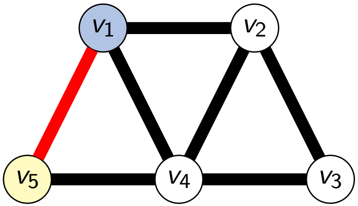
Counting all solutions

Kirchhoff's Matrix-Tree Theorem

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

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

Counting solutions excluding a given edge (i, j)



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

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

Laplacian(G) Laplacian($G \setminus \{(1,5)\}$)

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

Counting solutions excluding a given edge (i, j)

Sherman-Morrison formula

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

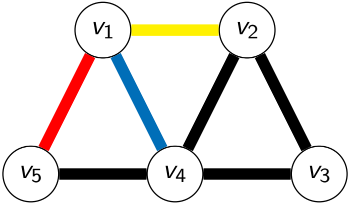
In our case this simplifies to $\det(M') = (1 - m_{jj}^{-1})\det(M)$.

So

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

Counting solutions excluding a given edge (i, j)

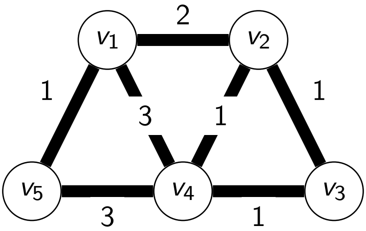
One matrix inversion for all edges incident to a given vertex



Example

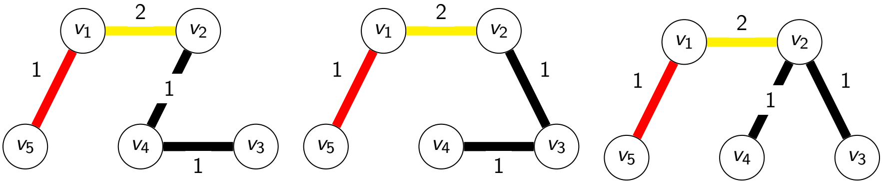
Let M be the sub-matrix of L obtained by removing its first row and column as before. Then $M^{-1} = \begin{pmatrix} 12/21 & 9/21 & 8/21 & 3/21 \\ 9/21 & 19/21 & 8/21 & 4/21 \\ 6/21 & 8/21 & 10/21 & 5/21 \\ 3/21 & 4/21 & 5/21 & 13/21 \end{pmatrix}$

minimum_spanning_tree constraint

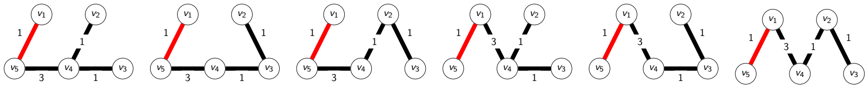


- 3 spanning trees of cost 5.
- 6 spanning trees of cost 6.
- 7 spanning trees of cost 7.
- 3 spanning trees of cost 8.
- 2 spanning trees of cost 9.

minimum_spanning_tree constraint

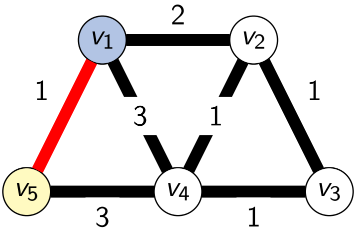


Trees of cost 5



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

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

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

Generalized Matrix-Tree Theorem

$$3x^{\mathbf{5}} + 6x^6 + 7x^7 + 3x^8 + 2x^9$$

- 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

$$3x^5 + 6x^{\mathbf{6}} + 7x^7 + 3x^8 + 2x^9$$

- 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.

Outline

knapsack constraint

Definition

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

To count solutions, we can proceed as for `regular` constraints (compact representation of solutions) but it now runs in pseudo-polynomial time (w.r.t. ℓ and u).

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

Counting for knapsack

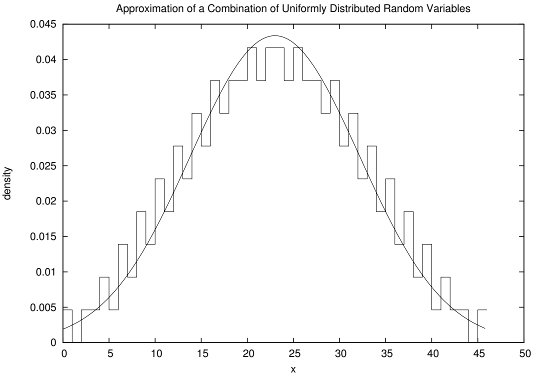
- Express variable in terms of other variables:
  $\ell \leq \sum_{i=1}^{n} c_i x_i \leq u$ is rewritten as
  $x_j = \frac{1}{c_j}(x_{n+1} - \sum_{i=1}^{j-1} c_i x_i - \sum_{i=j+1}^{n} c_i x_i)$ with $x_{n+1} \in [\ell, u]$.
- Relax domains to intervals
- Assume values in domains are equiprobable (uniform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

But our assumption doesn't hold for weighted counting

Counting for knapsack

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 $\sigma^2 = 84.583$.



Outline

1. Exposed Combinatorial Structure in CP
2. (Weighted) Counting
   - Compact representation of the solution set
   - Sampling (interleaved with domain filtering)
   - Use existing theoretical result
   - Domain relaxation
3. CP-BP Framework
   - A Small Example
   - Branching for Combinatorial Search
   - (Near-)Uniform Sampling
   - Neuro-Symbolic AI
4. Conclusion

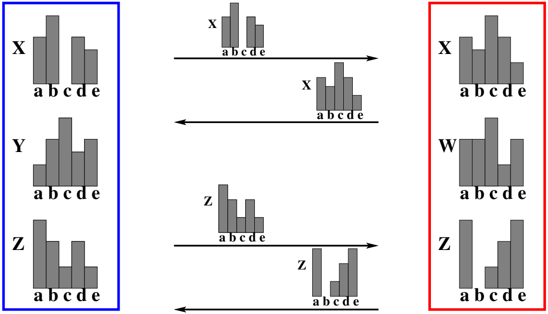
Moving beyond

standard support propagation

to

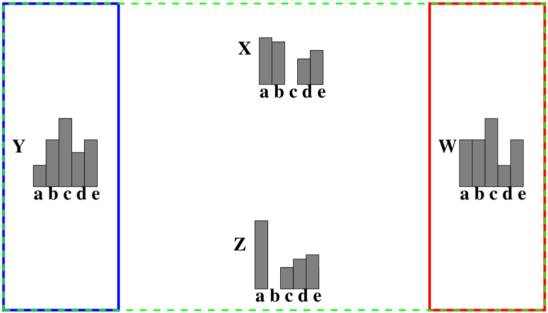
belief (marginal) propagation

Marginal (Belief) Propagation



- propagate marginal distributions over single variables
- iteratively adjust each constraint's marginals
- until some stopping criterion

Marginal (Belief) Propagation



- propagate marginal distributions over single variables
- iteratively adjust each constraint's marginals
- until some stopping criterion

How do we compute such marginal distributions?

Corresponds to weighted model counting *on each constraint*

Outline

Belief Propagation over the CP Model

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

1. $\texttt{alldifferent}(a, b, c)$
2. $a + b + c + d = 7$
3. $c \leq d$

$\theta_c^{iii}(3)$

support (solution)	weight
$d = 3$	1
$d = 4$	1
	$\sum = 2$

2 out of 10 solutions to *iii*

Belief Propagation over the CP Model

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

1. alldifferent(a, b, c)
2. $a + b + c + d = 7$
3. $c \leq d$

		1	2	3	4
a	θ_a^i	1/4	1/4	1/4	1/4
	θ_a^{ii}	10/20	6/20	3/20	1/20
b	θ_b^i	1/4	1/4	1/4	1/4
	θ_b^{ii}	10/20	6/20	3/20	1/20
c	θ_c^i	1/4	1/4	1/4	1/4
	θ_c^{ii}	10/20	6/20	3/20	1/20
	θ_c^{iii}	4/10	3/10	**2/10**	1/10
d	θ_d^{ii}	10/20	6/20	3/20	1/20
	θ_d^{iii}	1/10	2/10	3/10	4/10

Belief Propagation over the CP Model

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

1. alldifferent(a, b, c)
2. $a + b + c + d = 7$
3. $c \leq d$

		1	2	3	4
c	θ_c^i	1/4	1/4	1/4	1/4
	θ_c^{ii}	10/20	6/20	3/20	1/20
	θ_c^{iii}	4/10	3/10	2/10	1/10
	θ_c	**40/800**	**18/800**	**6/800**	**1/800**

Belief Propagation over the CP Model

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

1. $\texttt{alldifferent}(a, b, c)$
2. $a + b + c + d = 7$
3. $c \leq d$

		1	2	3	4
c	θ_c^{i}	1/4	1/4	1/4	1/4
	θ_c^{ii}	10/20	6/20	3/20	1/20
	θ_c^{iii}	4/10	3/10	2/10	1/10
	θ_c	**.62**	**.28**	**.09**	**.01**

Belief Propagation over the CP Model

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

1. `alldifferent`(a, b, c)
2. $a + b + c + d = 7$
3. $c \leq d$

Iteration 1

	1	2	3	4
θ_a	.50	.30	.15	.05
θ_b	.50	.30	.15	.05
θ_c	**.62**	**.28**	**.09**	**.01**
θ_d	.29	.34	.26	.11

Belief Propagation over the CP Model

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

1. $\texttt{alldifferent}(a, b, c)$
2. $a + b + c + d = 7$
3. $c \leq d$

		1	2	3	4
	$\vdots$				
d	θ_d^{ii}	10/20	6/20	3/20	1/20

$\theta_c^{iii}(3)$

support (solution)	weight
$d = 3$	$3/20$
$d = 4$	$1/20$
	$\sum = 4/20$

Belief Propagation over the CP Model

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

1. `alldifferent`(a, b, c)
2. $a + b + c + d = 7$
3. $c \leq d$

Iteration 10

	1	2	3	4
θ_a	.01	**.52**	**.46**	.01
θ_b	.01	**.52**	**.46**	.01
θ_c	**.98**	.02	.00	.00
θ_d	**.90**	.10	.00	.00

Belief Propagation over the CP Model

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

1. $\texttt{alldifferent}(a, b, c)$
2. $a + b + c + d = 7$
3. $c \leq d$

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

	1	2	3	4
θ_a	0	1/2	1/2	0
θ_b	0	1/2	1/2	0
θ_c	1	0	0	0
θ_d	1	0	0	0

Outline

Branching for Combinatorial Search

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

min-entropy

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

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

2. choose value maximizing the marginal:

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

Outline

1. Exposed Combinatorial Structure in CP
2. (Weighted) Counting
   - Compact representation of the solution set
   - Sampling (interleaved with domain filtering)
   - Use existing theoretical result
   - Domain relaxation
3. CP-BP Framework
   - A Small Example
   - Branching for Combinatorial Search
   - (Near-)Uniform Sampling
   - Neuro-Symbolic AI
4. Conclusion

(Near-)Uniform Sampling

sample solutions uniformly at random

Given true marginal distributions:

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

CP Belief Propagation could lead to near-uniform sampling.

Outline

Neuro-Symbolic AI

Neural networks (NN) dealing with hard/deterministic combinatorial structure

Ex: computer code generation, safe robotics, drug discovery

Data-driven + Model-driven

Combinatorial solvers

- can tell whether or not a NN output satisfies the constraints
- are expensive to run (answer an $\mathcal{NP}$-hard question)

Neuro-Symbolic AI

Neural networks (NN) dealing with hard/deterministic combinatorial structure

Ex: computer code generation, safe robotics, drug discovery

Data-driven + Model-driven

CP (among combinatorial solvers)

- can identify certain NN outputs that cannot satisfy the constraints
- runs in polytime because we don't ask for a SAT check

Neuro-Symbolic AI

Neural networks (NN) dealing with hard/deterministic combinatorial structure

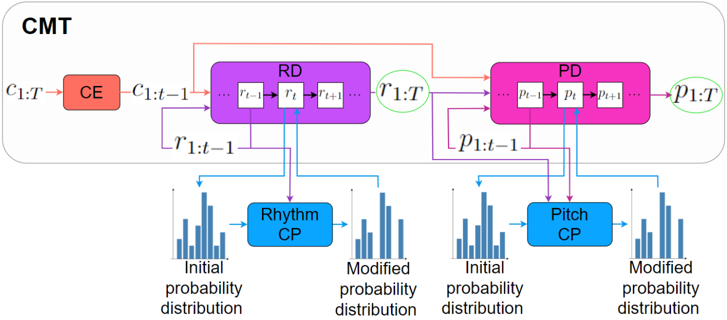
Ex: computer code generation, safe robotics, drug discovery

Data-driven + Model-driven

Marginals-augmented CP

- more discriminating between possible NN outputs
- combines more naturally with NN outputs
- runs in polytime as well

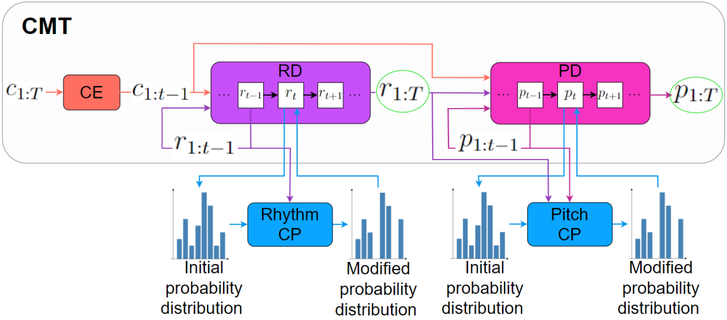
Inference: Adjusting NN's Learned PMF given Constraints



inputs to CP-BP solver

- constraints that you wish to enforce
- sequence so far (fixed variables)
- probability mass function for next token (unary constraint)

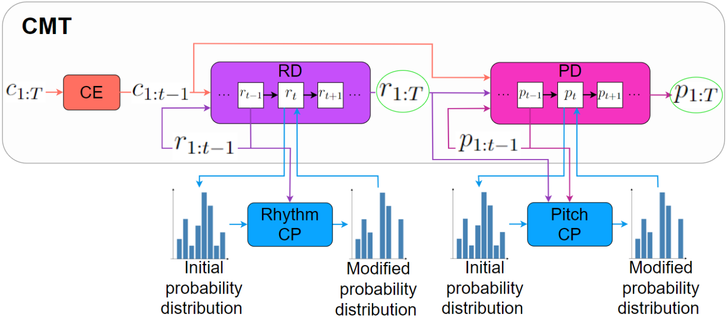
Inference: Adjusting NN's Learned PMF given Constraints



output of CP-BP solver, after iterated BP (no branching)

- adjusted probability mass function (marginals), from which the next token is sampled

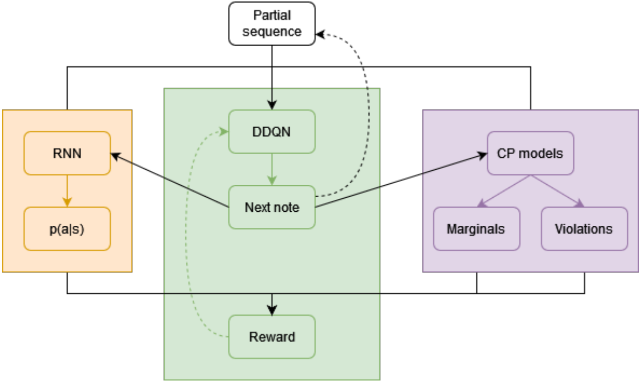
Inference: Adjusting NN's Learned PMF given Constraints



results

- generated sequence satisfies constraints without straying too far from training data

Training: Fine-Tuning an RNN given Constraints, using RL



reward function for action a

rnn+marginals+violations: $\log(p(a|s)) + c_1 \cdot (c_2 \cdot \hat{\theta}(a) - \sum v(a))$

Outline

1. Exposed Combinatorial Structure in CP
2. (Weighted) Counting
   - Compact representation of the solution set
   - Sampling (interleaved with domain filtering)
   - Use existing theoretical result
   - Domain relaxation
3. CP-BP Framework
   - A Small Example
   - Branching for Combinatorial Search
   - (Near-)Uniform Sampling
   - Neuro-Symbolic AI
4. Conclusion

Conclusion

Q- What is the distinctive driving force behind CP?

A- Direct access to problem structure from high-level constraints

What can we do with this knowledge?

- stronger search-space reduction
- better guidance to find solutions
- near-uniform sampling of solution set
- natural interface with neural networks
- ...

Acknowledgements

POLY MTL 150 ANS

CRSNG NSERC

IVADO

Mila

Alliance de recherche numérique du Canada | Digital Research Alliance of Canada