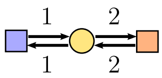
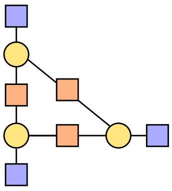
Lifted Approximate Inference



instead of



Kristian Kersting

technische universität dortmund

2014 SMLA WROCŁAW

1

Goals

- Loopy Belief Propagation and Linear Programming can be made aware of computational symmetries
- This can result in great speed-ups
- Computational symmetries can be detected using the Weisfeiler-Lehman (WL) algorithm
- WL computes fractional automorphisms in quasi-linear time; essentially no overhead!
- Few lines of Matlab code realize WL (with flooding) using sparse-matrix operations
- Strong connections to community detection, role discovery, graph kernels, clustering, …

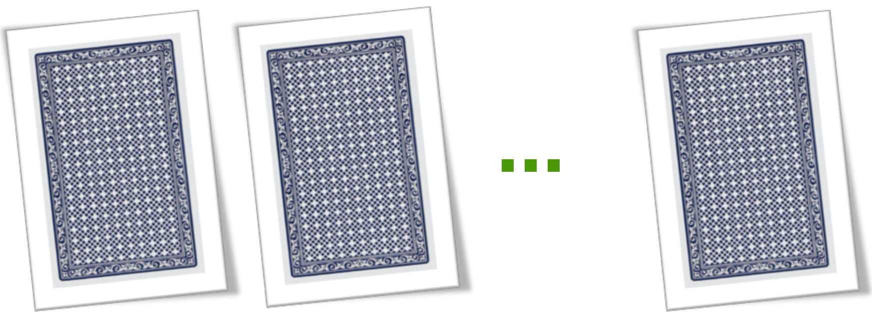
St. Paul's Cathredal, London, UK

General Take-Away Message

- **Sparseness and Tree-width are not enough**
- **We need to be aware of Symmetries**

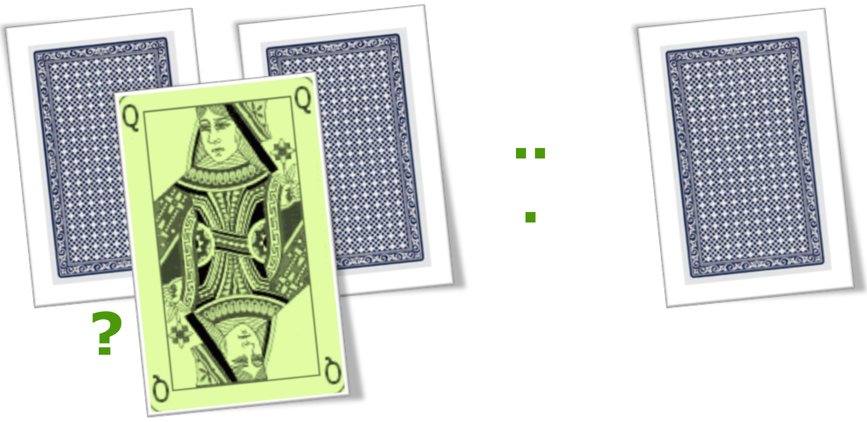
A Simple AI Problem

Thanks to Guy Van den Broeck



There are 52 cards (2-10, J, Q, K, A per color) and we would like to compute some basic probabilities

A Simple AI Problem



What is the probability that the first card will be a Queen if we reveal the first card, put it back, shuffle the deck, reveal the first card ....

The probability is
4/52=1/13

A Simple AI Problem

Probability 13/52=1/4

A Simple AI Problem

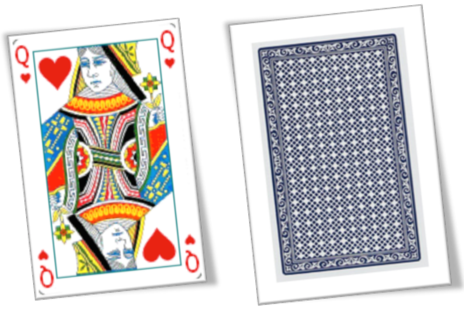
?

...

What if we knew already that its color is red?

Probability 13/26=1/2

A Simple AI Problem



...

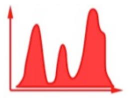
?

Probability13/51

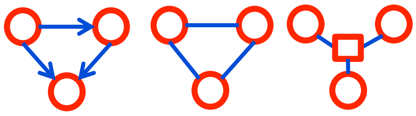
How does AI do this?

Graphical Model **+ Inference Algorithm**

Node=
Random Variable (RV)
Edge=
Dependency between RVs



$$P(\mathbf{X}) = \prod_{x \subseteq \mathbf{X}} \phi_j(x)$$

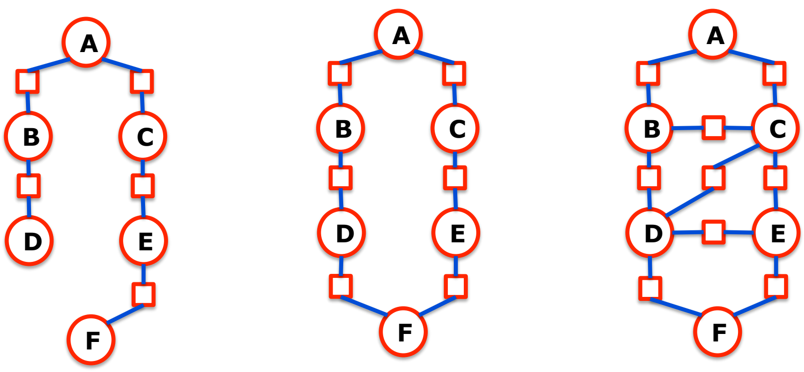


exact
Variable Elimination
Junction Tree

$P(X_i|E=e)$

Loopy Belief Prop.
MPLP
approximate

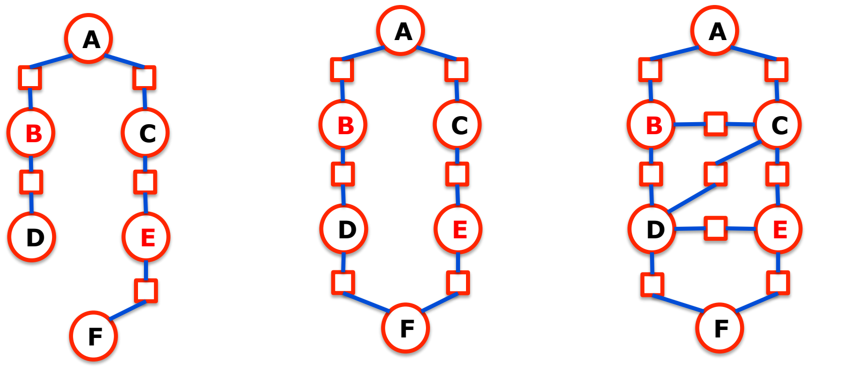
When is this efficient?



The more similar to a tree, i.e., small tree-width!

Why? Conditional Independency

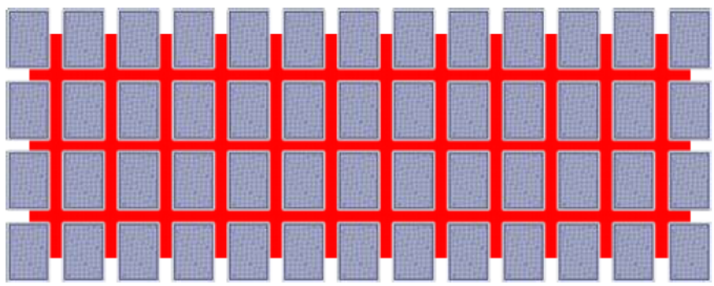
Dependencies simplify computations



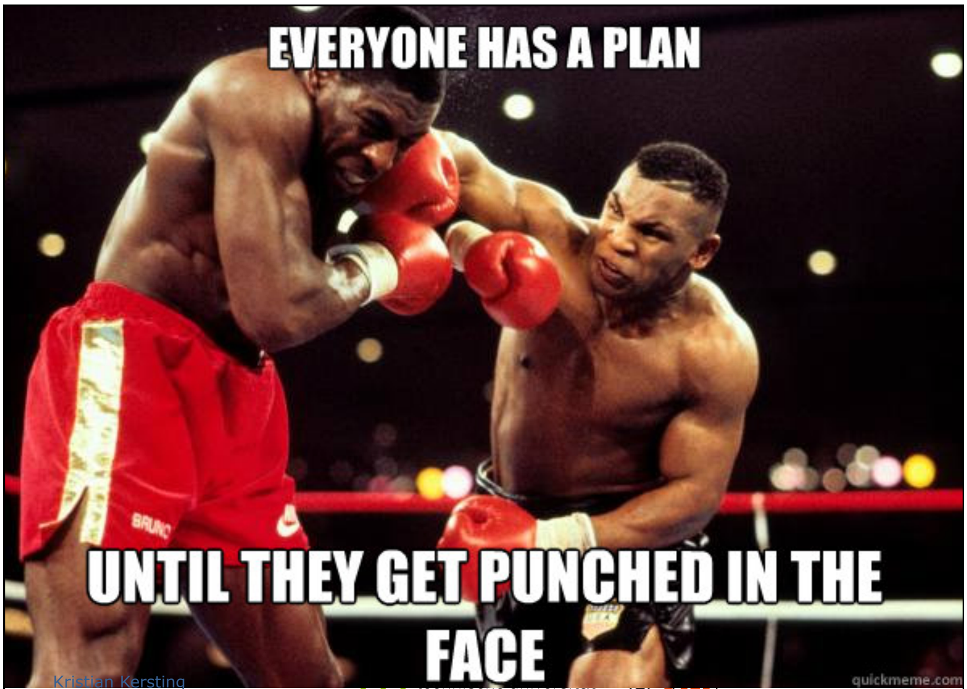
Tree	Graph	Graph
$P(A\|C,E) = P(A\|C)$	$P(A\|C,E) \neq P(A\|C)$	$P(A\|C,E) \neq P(A\|C)$
$P(A\|B,E,F) = P(A\|B,E)$	$P(A\|B,E,F) = P(A\|B,E)$	$P(A\|B,E,F) \neq P(A\|B,E)$

Back to the card problem

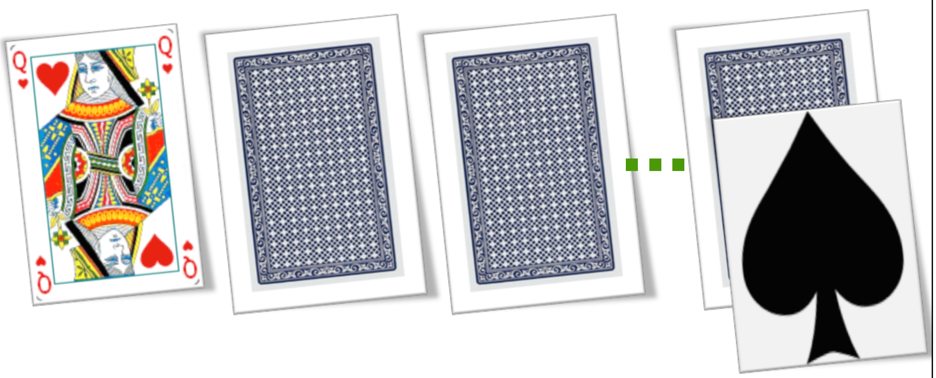
- Probabilistic propositional model is fully connected



- There are NO independencies
- Exact inference builds a table of $\geq 13^{52}$ Rows!
- Even approximate (for example, message passing) methods need to pass $\geq 13^{52}$) messages



What is the issue?



Probability 13/51

What is the issue?

Probability 13/51

What is the issue?

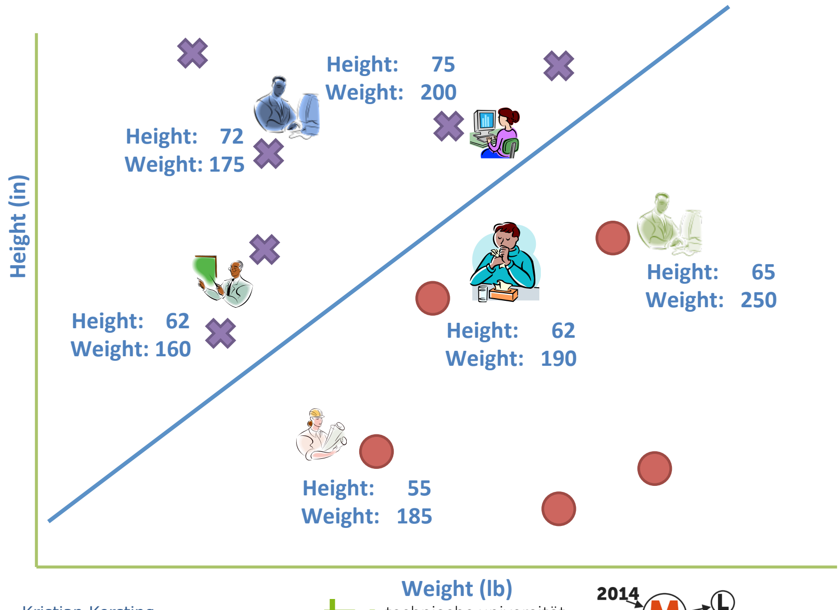
Probability 13/51

Tractable Probabilistic Inference

- Traditional Belief: Independence (Conditional/Contextual)
- Now: Symmetry (Exchangeability)

Where do the symmetries come from?

Traditional Machine Learning

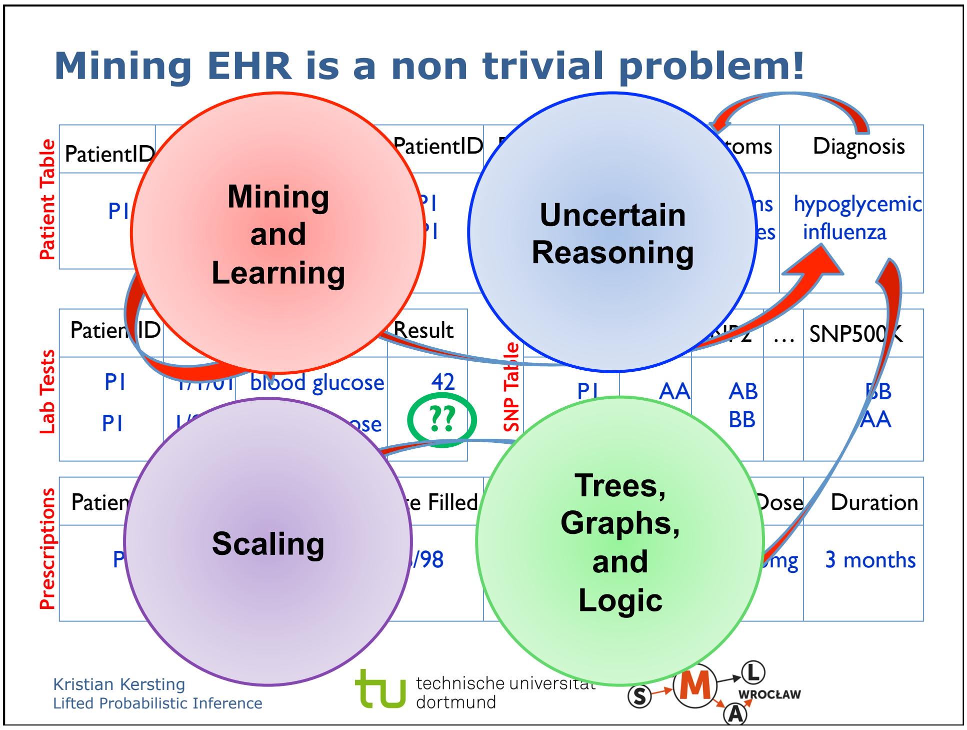


Standard Data Representation: Single Tables

Id	Age	Gender	Weight	BP	Sugar	LDL	Diabetes?
1	27	M	170	110/70	6.8	40	N
2	35	M	200	180/90	9.8	70	Y
3	21	F	150	120/80	4.8	50	N
				…			

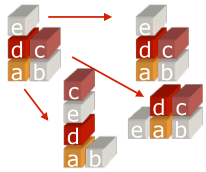
But nowadays data become richer and richter


Nat Rev Genet. 2012 May 2;13(6):395-405
Heart diseases and strokes – cardiovascular disease – are expensive for the world
According to the World Heart Federation, cardiovascular disease cost the European Union EURO169 billion in 2003 and the USA about EURO310.23 billion in direct and indirect annual costs. By comparison, the estimated cost of all cancers is EURO146.19 billion and HIV infections, EURO22.24 billion
nature REVIEWS
GENETICS
Electronic Health Records
A New Opportunity for AI to Save our Lifes
[Natarajan, Kersting, Joshi, Saldana, Ip, Jacobs, Carr IAAI 2013]


Mining EHR is a non trivial problem!
Patient Table
PatientID
PI
PatientID
PI
PI
Diagnosis
hypoglycemic
influenza
Mining and Learning
Uncertain Reasoning
Lab Tests
Result
blood glucose
42
??
SNP Table
PI
AA
AB
BB
SNP500K
AA
Prescriptions
Duration
3 months
Scaling
Trees, Graphs, and Logic
Kristian Kersting
Lifted Probabilistic Inference
technische universität dortmund
WROCŁAW

Endless Showcases !

Planning

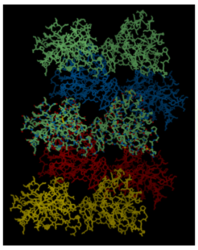


Imaging (medical)

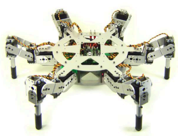
Social Networks

Scene interpretation/ segmentation

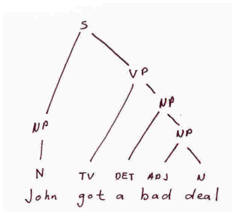
BioInformatics



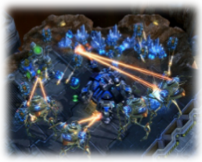
Robotics



Natural Language Processing



Games

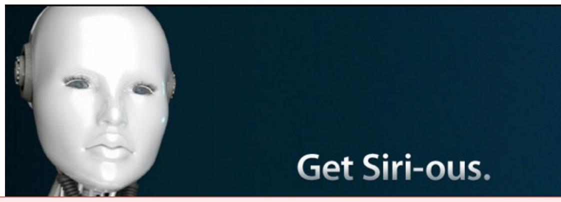


Big Data is not enough

- **The data soup**: most data is in logs, text, blogs, images, the web, databases, …
- **The knowledge soup:** next to the data, we may have background knowledge, often even competing theories
- **The reasoning soup**: pool of interacting tasks and algorithms

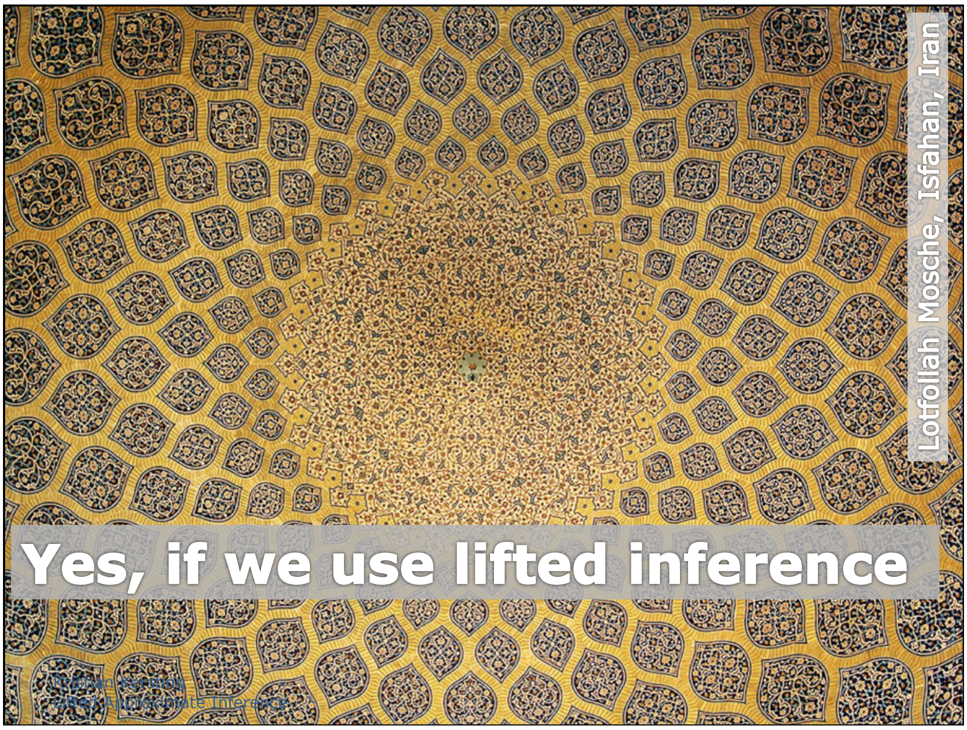
Even though computers can search the data for keywords, features, and models, they do not really understand most of it

Statistical Relational AI is serious business



However, efficient complex probabilistic reasoning becomes central! Can we make it faster?

- Probabilistic relational models are used by several million users.
- Many other applications such as entity resolution, information extraction, unsupervised semantic parsing, NELL, information broadcasting, …



So, what is lifted inference?

1. An inference algorithm that deals with "*groups*" of random variables at a first-order level
   - Takes a general first-order model as input
   - Automatically answers queries without computational waste
2. Reason over a large domains in time independent of the number of objects
3. Ability to carry out probabilistic inference in a relational probabilistic model without needing to reason about each individual separately
4. Try and perform inference at the first-order logic level and to ground out only when necessary

More views

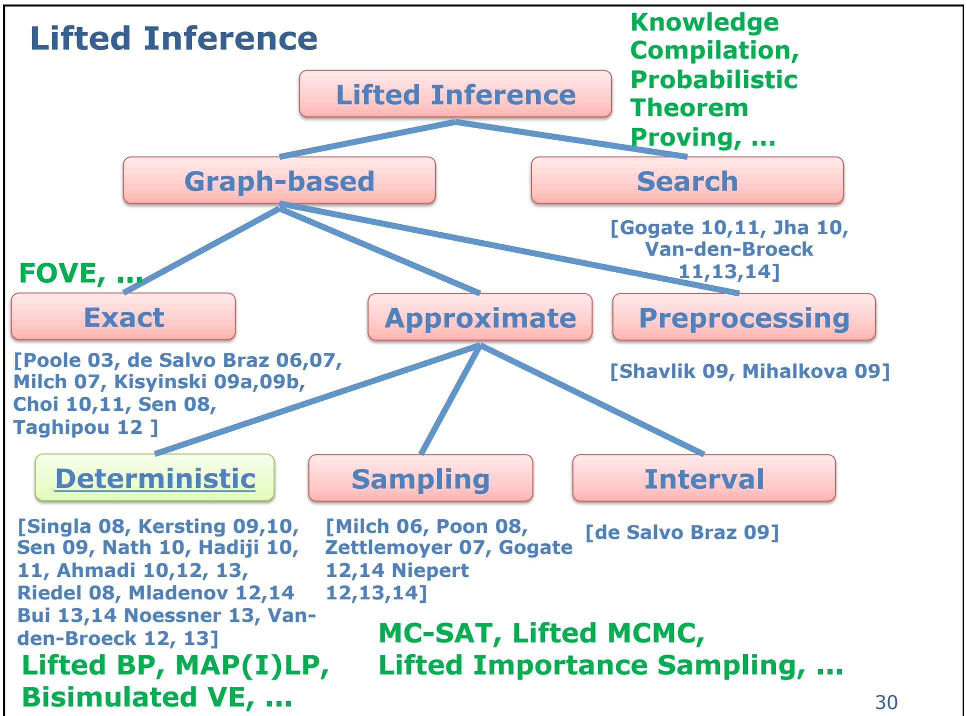
5. Algorithm that exploits interchangeability in the domain
6. Queries are answered without instantiating all the objects in the domain
7. Exploit shared correlations – Same uncertainties and correlations repeatedly occur in data
8. Exploit symmetries in the data and the model

Different definitions but similar goals

Least Common Denominator

- **Exploit symmetry = Identify similar groups of random variables**

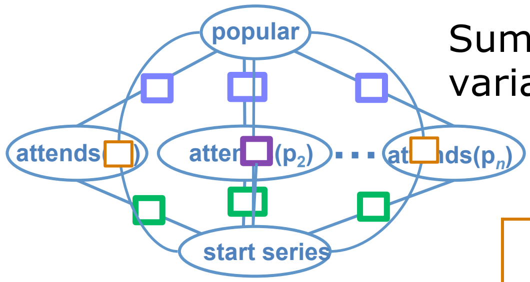
Lifted Inference



[Zhang, Poole 1994]

Variable Elimination

Example: Inviting ***n*** people to a workshop



Sum out non-query variables one by one

Time is linear in number of invitees *n*

$$\underbrace{\sum_{att(p_i)} \phi_1(pop, att(p_i))\ \phi_2(att(p_i), ser)}_{\phi'(pop,\ ser)}$$

Can't we do better?

Kristian Kersting
Machines Reading the Data

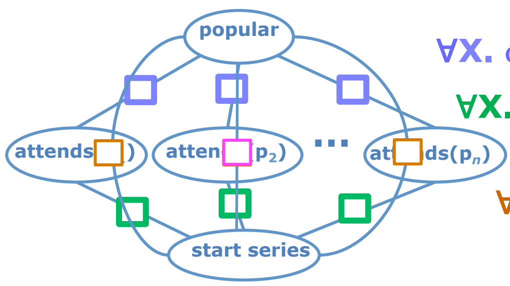
technische universität dortmund

2014 WROCŁAW

[Poole 2003; de Salvo Braz et al. 2005]

First-Order Variable Elimination

Based on logically parameterized factors



$\forall X.\ \phi_1(popular, attends(X))$

$\forall X.\ \phi_2(attends(X), series)$

⇩

$\forall X.\ \phi'(popular, series)$

⇩

$\phi'(popular, series)^n$

Idea: Exploit symmetries across factors, i.e., sum out all attends(X) variables at once

Time is constant in *n*

And there is often additional structure for optimization

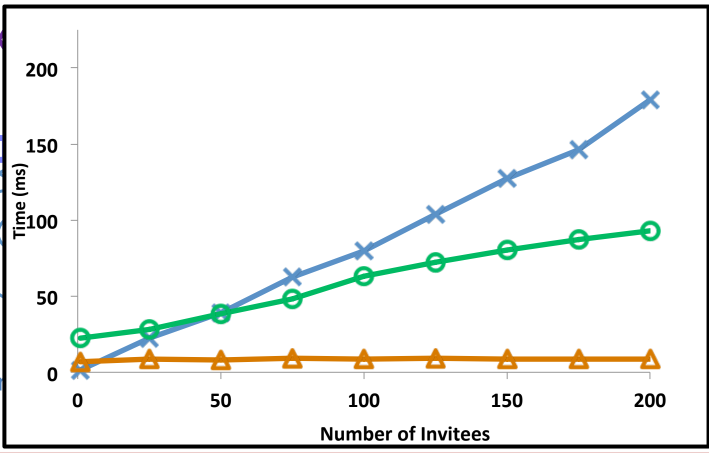
[Milch,Zettlemoyer, Haims, **Kersting**, Kaelbling AAAI08]

Exploiting Symmetries within Factors

∀W ∀X

hot(w_1)

att(p_1)



Can't su

joining

(W),

level

Actually, it turns out that lifted probabilistic inference is closely related to theorem proving

[Gogate, Domingos AAAI 2011, Van den Broeck et al. IJCAI 2011]

Probabilistic Theorem Proving (PTP)

TP(*KB*, *Query*)
  KB$_Q$ **← *KB* U {¬ *Query*}**
  return ¬SAT(CNF(*KB$_Q$*))

PTP(*PKB*, *Query*)
  PKB$_Q$ **← *PKB* U {(*Query,0*)}**
  return WMC(WCNF(*PKB$_Q$*))
    / WMC(WCNF(*PKB*))

All we need is lifted weighted model counting

First, however, we have to convert the PKB into (Lifted) CNF + Weights

WCNF(*PKB*) Clauses/Formula Potential
  for all (F_i, Φ_i) ∊ *PKB* s.t. $\Phi_i > 0$ **do**
    $PKB \leftarrow PKB \cup \{(\boldsymbol{F_i} \Leftrightarrow \boldsymbol{A_i}, 0)\} \setminus \{(F_i, \Phi_i)\}$
  $CNF \leftarrow \text{CNF}(PKB)$ **Hard formula as weight is 0**
  for all $\neg A_i$ literals **do** $W_{\neg Ai} := \Phi_i$
  for all other literals L **do** $W_L := 1$
  return (*CNF*, *weights*)

Lifted Weighted Model Counting

LWMC(CNF, substs, weights)
  if all clauses in WCNF are satisfied
    return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})^{n_A(substs)}$
  if CNF has empty unsatisfied clause **return** 0

Base Case

Lifted Weighted Model Counting

LWMC(*CNF, substs, weights*)

if all clauses in *WCNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})^{n_A(substs)}$

if *CNF* has empty unsatisfied clause **return** 0

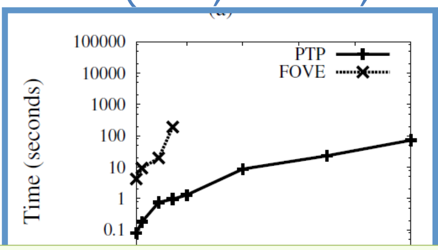
if there exists a lifted decomposition of CNF sharing no unifiable literals

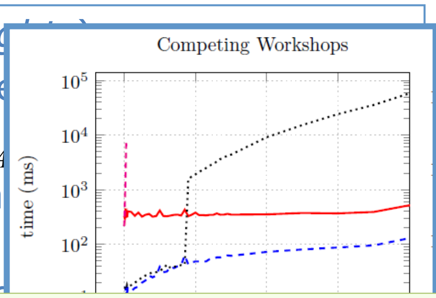
return $\prod_{i=1}^{k} [LWMC(CNF_{i,1}, substs, weights)]^{m_i}$

Decomp. Step

Lifted Weighted Model Counting

LWMC(*CNF, substs, wei*





What about approximate inference?

choose an atom *A*

Splitting Step

Main computational step. I am skipping the details. Nice connection to knoweldge compilation e.g. using first-order d-DNNF for efficient model counting. Is closely related to recursive conditioning

Introduction of lifted loopy belief propagation

Understanding its main ingredient: color-passing

Connections to AI, ML, and DM

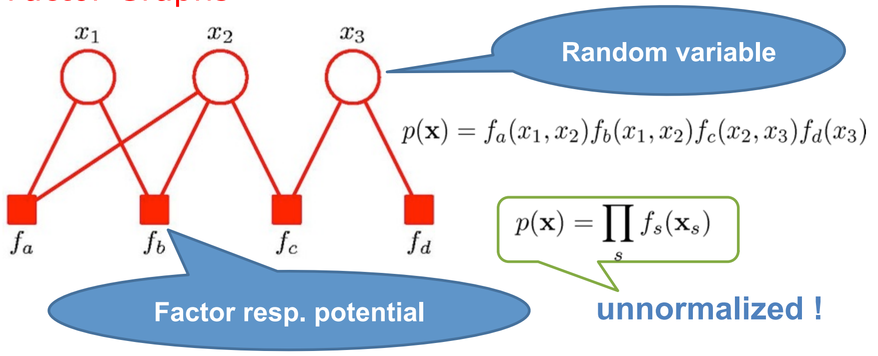
Lotfollah Mosche, Isfahan, Iran

Lifted / SYMMETRY-AWARE LOOPY BELIEF PROPAGATION

[Pearl, Koller, Friedman, Lauritzen, Spiegelhalter, …]

Reminder Factor Graphs

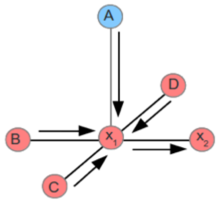
Distributions can naturally be represented as Factor Graphs

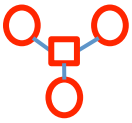


- There is an edge between a circle and a box if the variable is in the domain/scope of the factor

Approximate Inference using Loopy Belief Propagation

Random variables exchange Messages: "I *(A) believe that you (X_1) should be in state x_1* with probability … ."





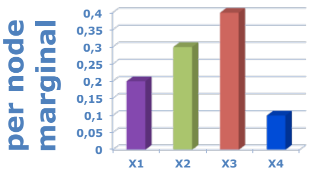
factor graph

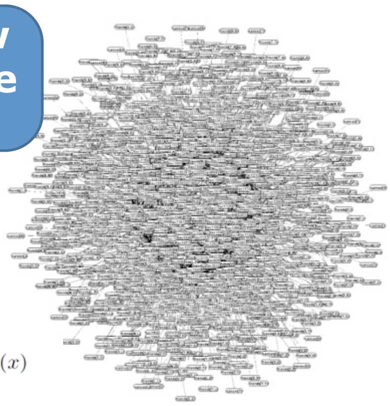
$$\mu_{X\to f}(x) = \prod_{h\in \mathrm{nb}(X)\setminus\{f\}} \mu_{h\to X}(x)$$

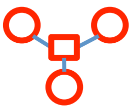
$$\mu_{f\to X}(x) = \sum_{\neg\{x\}} \left(f(\mathbf{x}) \prod_{y\in \mathrm{nb}(f)\setminus\{X\}} \mu_{y\to f}(y) \right)$$

Loopy Belief Propagation for Social Network Analysis

Inference often slow even for approximate inference



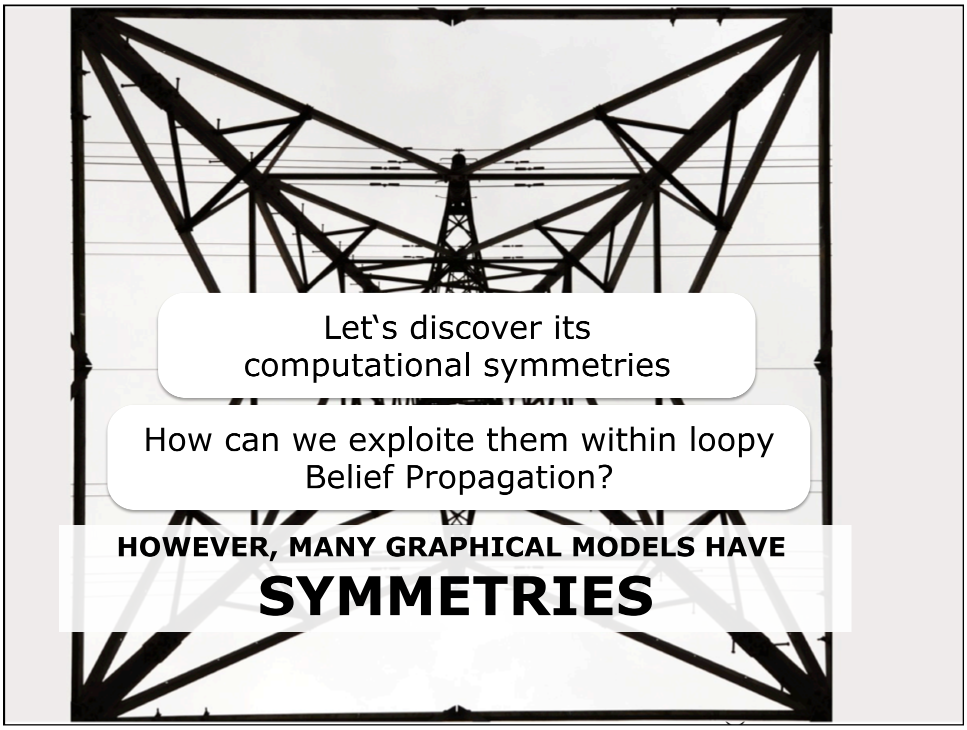




$$\mu_{X\to f}(x) = \prod_{h\in \mathrm{nb}(X)\setminus\{f\}} \mu_{h\to X}(x)$$

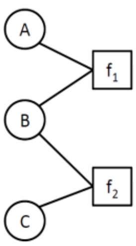
$$\mu_{f\to X}(x) = \sum_{\neg\{x\}} \left(f(\mathbf{x}) \prod_{y\in \mathrm{nb}(f)\setminus\{X\}} \mu_{y\to f}(y) \right)$$

factor graph + loopy belief propagation



[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

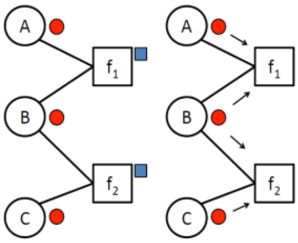
Step 1: Coloring the graph



- **Color nodes according to the evidence you have**
  - No evidence, say **red**
  - State „one", say **brown**
  - State „two", say **orange**
  - ...
- **Color factors distinctively according to their equivalence classes.** For instance, assuming f_1 and f_2 to be identical and B appears at the second position within both, say **blue**

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

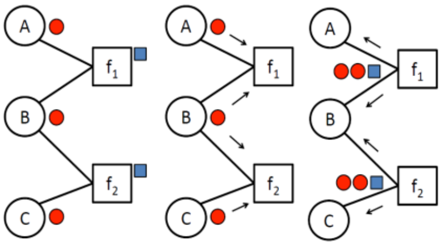
Step 2: Pass the colors around



1. Each factor collects the colors of its neighboring nodes

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

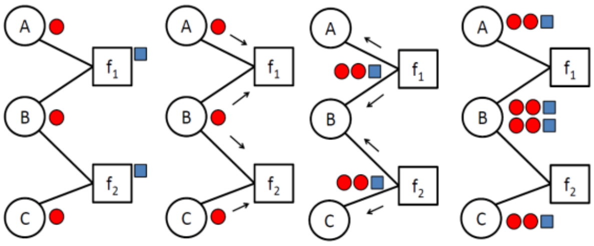
Step 2: Pass the colors around



1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" ist color signature with its own color

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

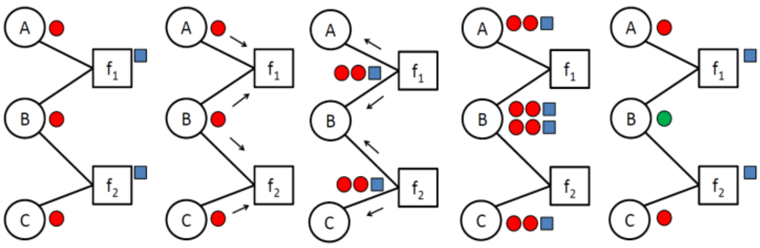
Step 2: Pass the colors around



1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" ist color signature with its own color
3. Each node collects the signatures of its neighboring factors

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

Step 2: Pass the colors around

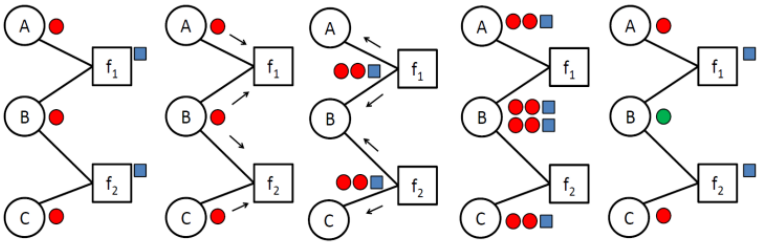


1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" ist color signature with its own color
3. Each node collects the signatures of its neighboring factors

4. Nodes are recolored according to the collected signatures

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

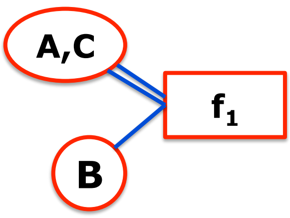
Step 2: Pass the colors around

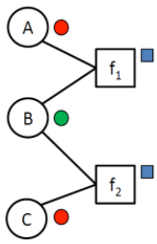


1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" ist color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures
5. If no new color is created stop, otherwise go back to 1

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

Step 3: Compress the factor

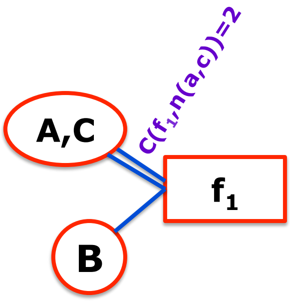




Essentially we just compute the so-called quotient factor graph

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

Step 4:
Run a modified Loopy Belief Propagation



$$b_i(x_i) = \prod_{\mathfrak{f}\in \mathrm{nb}(\mathfrak{X}_i)} \mu_{\mathfrak{f}\to\mathfrak{X}_i}(x_i)^{\boxed{c(\mathfrak{f},\mathfrak{X})}}$$

$$\mu_{\mathfrak{X}\to\mathfrak{f}}(x) = \mu_{\mathfrak{f}\to\mathfrak{X}}(x)^{\boxed{c(\mathfrak{f},\mathfrak{X})-1}} \cdot \prod_{\mathfrak{h}\in \mathrm{nb}(\mathfrak{X})\setminus\{\mathfrak{f}\}} \mu_{\mathfrak{h}\to\mathfrak{X}}(x)^{\boxed{c(\mathfrak{h},\mathfrak{X})}}$$

- Nodes are now groups of random variables
- The **counts** ensure that we send the same number of message as standard loopy belief propagation

[Singla, Domingos AAAI 2008; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]

Mind the counts! They also depend on the position of the variable within the factors

$$\mu_{\mathfrak{X}\to\mathfrak{f},p}(x) = \mu_{\mathfrak{f},p\to\mathfrak{X}}(x)^{c(\mathfrak{f},\mathfrak{X},p)-1} \prod_{\mathfrak{h}\in \mathrm{nb}(\mathfrak{X})} \prod_{\substack{q\in P(\mathfrak{h},\mathfrak{X}) \\ (\mathfrak{h},q)\neq(\mathfrak{f},p)}} \mu_{\mathfrak{h},q\to\mathfrak{X}}(x)^{c(\mathfrak{h},\mathfrak{X},q)}$$

$$\mu_{\mathfrak{f},p\to\mathfrak{X}}(x) = \sum_{\neg\{\mathfrak{X}\}} \left(\mathfrak{f}(\mathbf{x}) \prod_{\mathfrak{Y}\in \mathrm{nb}(\mathfrak{f})} \prod_{q\in P(\mathfrak{f},\mathfrak{Y})} \mu_{\mathfrak{Y}\to\mathfrak{f},q}(y)^{c(\mathfrak{f},\mathfrak{Y},q)-\delta_{\mathfrak{X}\mathfrak{Y}}\delta_{pq}} \right)$$

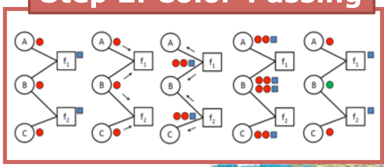
$$b_i(x_i) = \prod_{\mathfrak{f}\in \mathrm{nb}(\mathfrak{X}_i)} \prod_{p\in P(\mathfrak{f},\mathfrak{X}_i)} \mu_{\mathfrak{f},p\to\mathfrak{X}_i}(x_i)^{c(\mathfrak{f},\mathfrak{X}_i,p)}$$

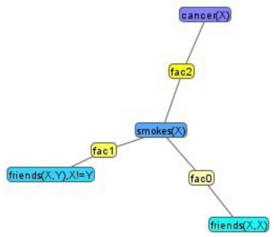
- Here, $P(\mathfrak{h},\mathfrak{X})$ denotes the position variables appear in factors
- The main difference is in the factor to variable messages. We now send only one message per „supernode" and position as expressed by the indicator functions
- For the lifting, we can turn the graph into a position- pairwise factor graph and then run color-passing

[Singla, Domingos AAAI 2008*; Kersting, Ahmadi, Natarajan UAI 2009; Ahmadi, Kersting, Mladenov, Natarajan MLJ 2013]
(*)Singla and Domingos actually proposed a relational variant that corresponds to color-passing on the ground network

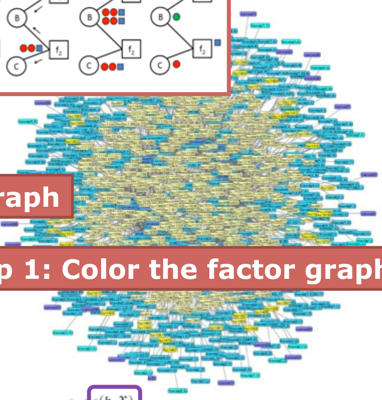
Lifted Loopy Belief Propagation

Step 2: Color-Passing

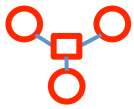




Step 3: Compress the factor graph



Step 1: Color the factor graph



$$b_i(x_i) = \prod_{f \in nb(X_i)} \mu_{f\to X_i}(x_i)^{\boxed{c(f,X)}}$$

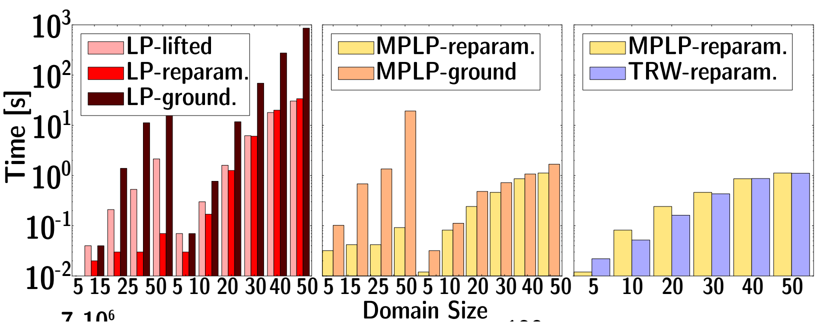
$$\mu_{X\to f}(x) = \mu_{f\to X}(x)^{\boxed{c(f,X)-1}} \cdot \prod_{h \in nb(X)\setminus\{f\}} \mu_{h\to X}(x)^{\boxed{c(h,X)}}$$

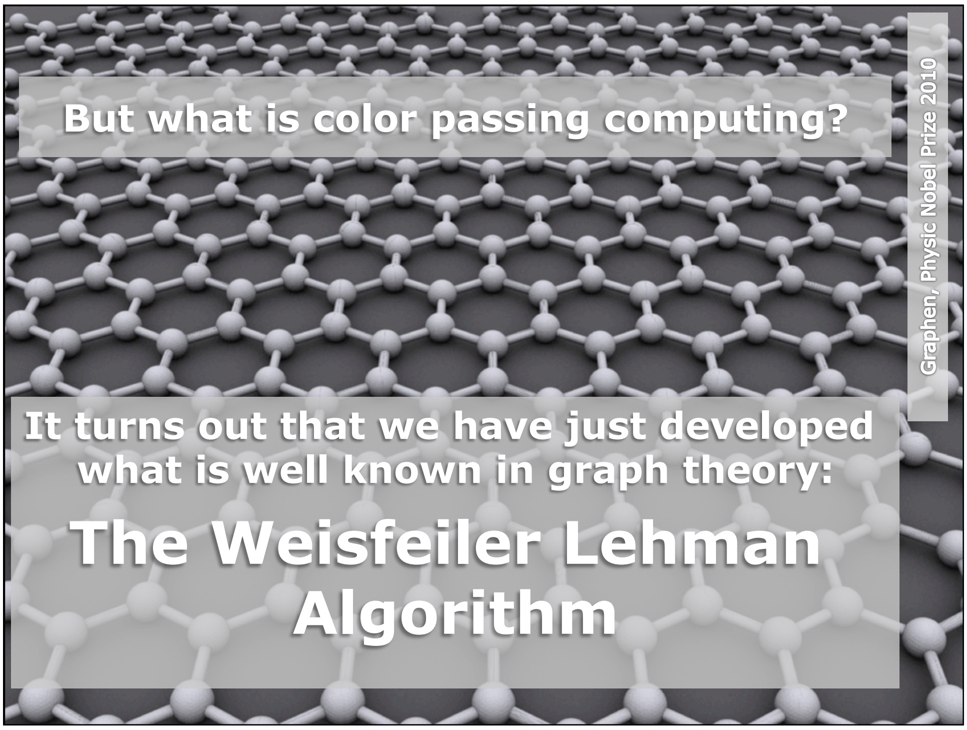
Step 4: Run modified Loopy Belief Propagation

53

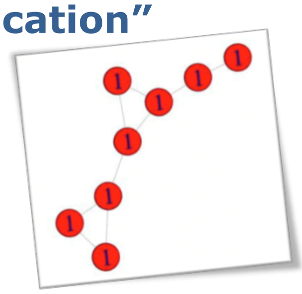
[Taken from Mladenov, Globerson, Kersting UAI 2014]

And these kinds of lifted message-passing approaches can be orders of magnitudes faster





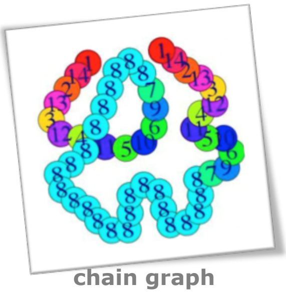
Weisfeiler-Lehman (WL) Algorithmus aka "naive vertex classification"

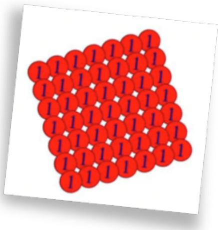


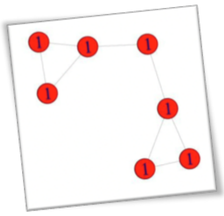
- Basic subroutine for graph isomorphism testing
- Computes so called fractional automorphisms:

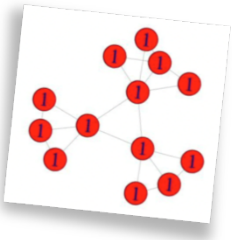
  Doubly stochastic matrices instead of permutation matrices
- Quasi-linear running time O((n+m)log(n)) when using asynchronous updates [Berkholz, Bonsma, Grohe ESA 2013]
- Part of graph tool SAUCY [See e.g. Darga, Sakallah, Markov DAC 2008]
- Can be extended to weighted graphs [Grohe, Kersting, Mladenov, Selman ESA 2014]

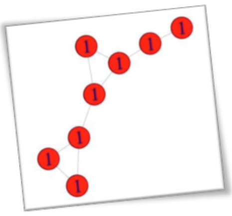
Examples for WL with flooding

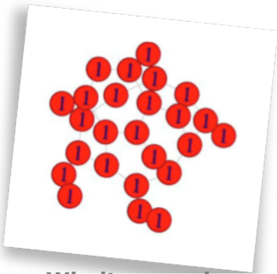

chain graph


grid graph


McKay graph

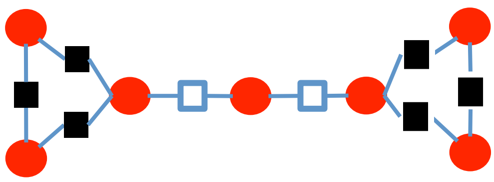

social rules


„walk" graph

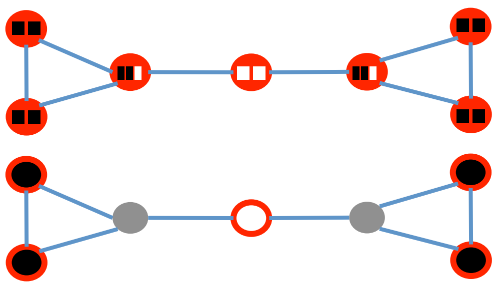

Whalter graph

The resulting partitions are called „coarsest equitable partitions" (CEPs)

But how do we get from factor graphs to graphs?



- Encode the factor colors into the node colors



- Then run Weisfeiler-Lehman / Color-Passing just on the graph with these initial colors

[Berkholz, Bonsma, Grohe ESA 2013]

Quasi-linear running time

- Send color message asynchronously and keep a stack of active color classes
  - Initially only color 1 is active
  - Pop an active color C from the stack and send message to the neigbors of the corresponding color class members (refine)
  - Push all new colors on the stack in increasing order except we used C already before. If so, then push all new colors but the largest one
    -Hopcroft's trick resulting in a halfing argument-
  - Stop if the stack if empty
- Due to the halfing argument this can be shown to be of O((n+m)log(n))

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Problem: Find automorphism (*) of an undirected graph with adjacency matrix A

Find a permutation matrix P such that AP=PA

$$P^* = \arg\min_{P \in \mathcal{P}} S(P)$$

$$S(P) := ||A - A'||_F^2 = ||A - PAP^T||_F^2$$

$$= ||(A - PAP^T)P||_F^2 = ||AP - PA||_F^2$$

$$||A||_F^2 = \mathrm{tr}(A^T A) = \sum_{i,j} |A_{ij}|^2$$

(*) Automorphisms have recently received a lot of attention for lifted inference, see e.g. [Nieper UAI 2012; Bui, Huynh, Riedel UAI 2013; Niepert AAAI 2013; Apsel, Kersting, Mladenov AAAI 2014; Bui, Huynh, Sontag UAI 2014] and later in this tutorial. For many graphs they can be computed in polynomial time. For general graphs, however, the complexity is unknown; it might be NP-complete.

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Problem: Find automorphism (*) of an undirected graph with adjancency matrix A

Find a permutation matrix P such that AP=PA

$$P^* = \arg\min_{P \in \mathcal{P}} S(P)$$
$$= \arg\max_{P \in \mathcal{P}} F(P)$$

Complexity is unknown. Let's relax the probelm

$$:= \mathrm{tr}(APAP^T)$$

$$||A||_F^2 = \mathrm{tr}(A^T A) = \sum_{i,j} |A_{ij}|^2$$

(*) Automorphisms have recently received a lot of attention for lifted inference, see e.g. [Nieper UAI 2012; Bui, Huynh, Riedel UAI 2013; Niepert AAAI 2013; Apsel, Kersting, Mladenov AAAI 2014; Bui, Huynh, Sontag UAI 2014] and later in this tutorial. For many graphs they can be computed in polynomial time. For general graphs, however, the complexity is unknown; it might be NP-complete.

61

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Problem: Fractional Automorphism

Find a doubly stochastic matrix D s.t. AD=DA

$$D^* = \arg\max_{D \in \mathcal{D}} F(D)$$

$$D\mathbf{1} = \mathbf{1}$$
$$\mathbf{1}^T D = \mathbf{1}^T$$

Birkhoff Polytope consists of all doubly stochastic matrices. It is convex

Kristian Kersting
Lifted Approximate Inference

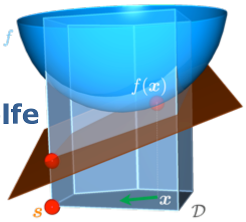
technische universität dortmund

2014 S M L A WROCŁAW

62

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Optimization over a convex set: Conditional Gradients aka Frank-Wolfe



- Start with $D^{(0)} \in \mathcal{D}$ — This lower bound is concave
- Lower bound F(D) with its FO Taylor series expansion

$$T^{(k)}(H) := F(D^{(k)}) + \langle \nabla F(D^{(k)}), H - D^{(k)} \rangle$$

- Maximize $T^{(k)}(H)$ s.t. $H \in \mathcal{D}$ — This is a Linear Assignement Problem

$$H^{(k)} = \max_{H \in \mathcal{D}} \langle \nabla F(D^{(k)}), H \rangle$$

- Since the polytope is convex, run a line search between $D^{(k)}$ and $H^{(k)}$ to find next $D^{(k+1)}$. Since the lower bound is concave, $H^{(k)}$ will be selected
- **Mission completed?**

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

NO!!

- This does not mimic WL at all!!!!
- It will just find the trivial solution (identity matrix). We do not employ the more-general-than relation among solutions!
- Replacing a quasi-linear approach by a sequence of cubic LAPs is stupid!

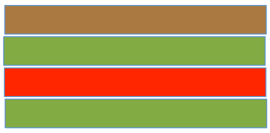
[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Exploit symmetries in the induced linear problems!

$$\nabla F(D^{(k)}) = 2AD^{(k)}A$$

- Intuitively, if two vertices of a graph have identical subgradients (rows in the gradient matrix) they are interchangeable

$$B_{ij} = \begin{cases} 1 & i\text{th vertex is in } j\text{th cluster of } \nabla F(D^{(k)}), \\ 0 & \text{otherwise} \end{cases}$$



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

65

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Do not solve the induced LPs at all

- B induces actually ascent directions.
  They are all what we need

$$H = BS^{-1}B^T$$

where S (the sizes of the clusters) takes care of normalization

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Conditional Gradients for Color Refinement/Weissfeiler Lehman

Algorithm 1: CGCR(A): CG for Color Refinement

Set $D^{(0)} = \frac{1}{n}\mathbf{1} \in \mathfrak{D}$, i.e., the flat partition matrix;
Set $k := 1$;
repeat
    $B := \textsc{CharactMat}\big(\nabla F(D^{(k)})\big)$;
    $S := \mathrm{diag}(B^T\mathbf{1})$ /* diagonal mat. of class sizes */;
    Update $D^{(k+1)} := BS^{-1}B^T$;
    Set $k := k + 1$;

Materializing $D^{(0)}$ breaks memory already for medium size graphs

Provably convergent to a local maximum of F in a linear number of iterations producing the same sequence of intermediate solutions as WL with flooding

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Matrix Multiplications for Color Refinement/Weissfeiler Lehman

Algorithm 4: CGCR(A): CG for Color Refinement

$B^{(0)} := \mathbf{1}$, i.e., the all 1 column vector;
$m^{(0)} := 1$ (the current maximal color) and $k := 1$;
repeat

But cubic running time !!!!

Just a few lines of matlab code !

Memory consumption scales well for sparse matrices

Provably convergent to a local maximum of F in a linear number of iterations producing the same sequence of intermediate solutions as WL with flooding

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Matlab code available at http://www-ai.cs.uni-dortmund.de/weblab/code.html

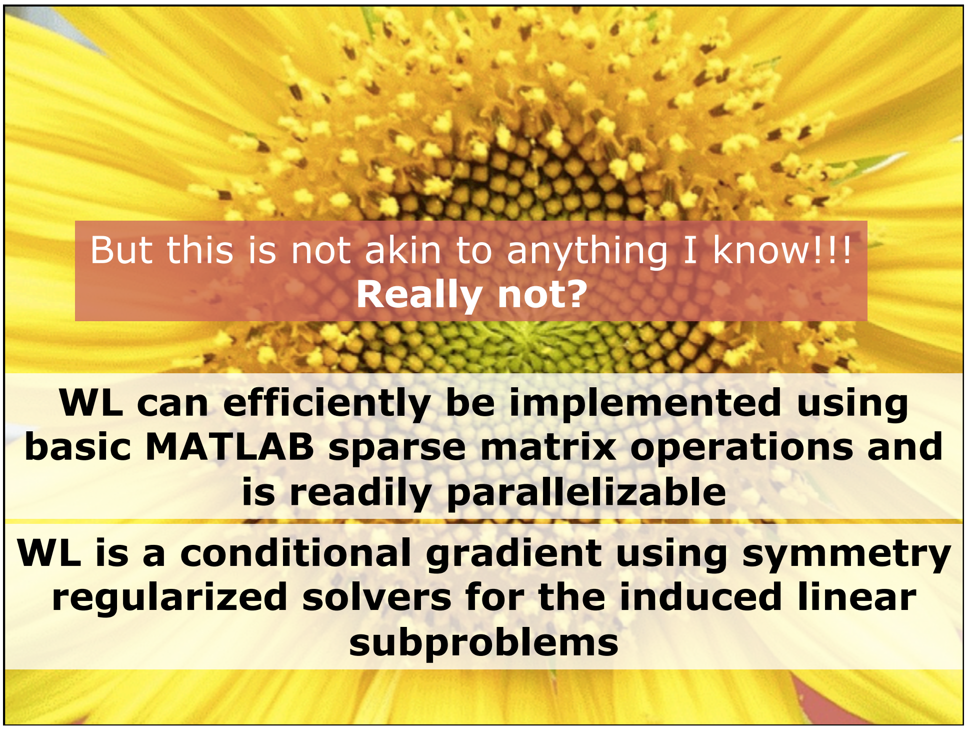
Perfectly Hashed Color Refinement/ Weissfeiler Lehman

Algorithm 5: HCGCR(A): Hashed CGCR

1 Let π an array where $\pi(i)$ equals to the ith prime;
2 $c^{(0)} := \mathbf{1}$, i.e., the all 1 column vector;
3 $m^{(0)} := 1$ (the maximal color) and $k := 1$;
4 **repeat**
5 $\quad c^{(k+1)} := \textsc{Colors}(c^{(k)} + A\log(\pi(c^{(k)})))$;

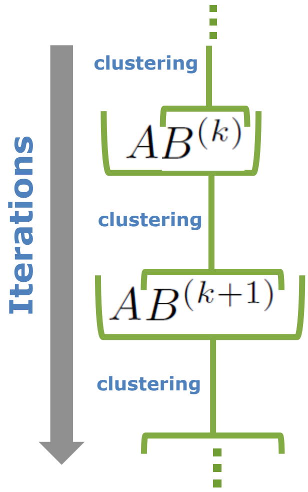
This is quadratic and can actually be turned into quasi-linear time using asynchronous updates

The fundamental theorem of artihmetic tells us that this is proveably correct



[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Sparse Matrix CP/WL



[Boldi et al. Theoretical Informatics and Applications 2006]

Theorem: Vertices in the same color class of CR have the same dot-product $x_{t+1}=Ax_t$ assuming $x_0(i)=x_0(j)$

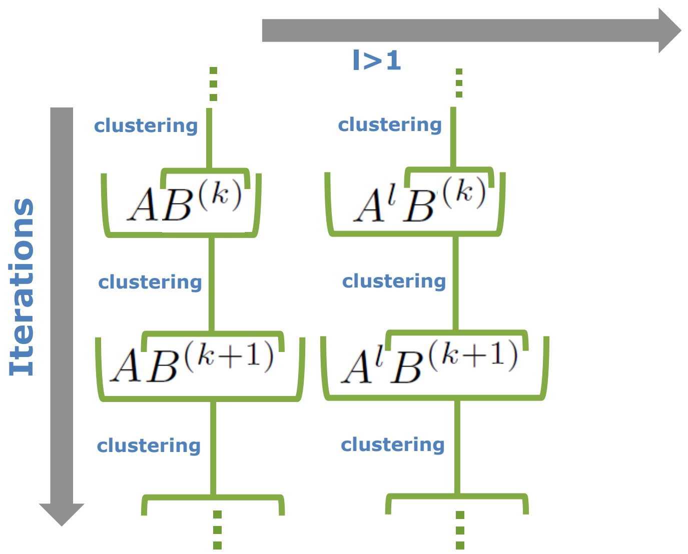
This is akin to Power Iteration

Lifted graph / fractional Automorphism

71

[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Power Iterated CP/WL

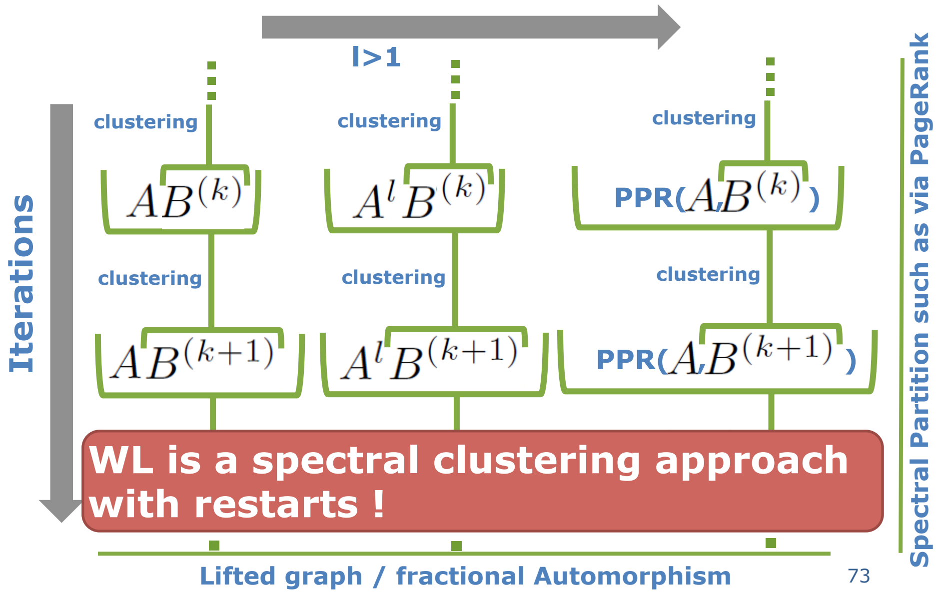


Lifted graph / fractional Automorphism

72

[R. Kannan, S. Vempala, A. Vetta: On clusterings: Good, bad and spectral. J. ACM 51(3): 497-515 (2004)]

Power Iterated CP/WL



[Kersting, Mladenov, Garnett, Grohe AAAI 2014]

Empirical Illustration using Matlab

Name / Description	# nodes	# edges	Avg. (5 reruns) time in sec. / median # CG iteration						S	C
			Hashing		PIfix		PIflex			
chain100001: Chain graph	100001	100,000	699.62	50002	420.18	•1252	•136.59	2892	< 0.01	49%
grid1000: Grid graph	1,000,000	1,998,000	96.26	501	77.44	•21	•23.87	41	0.43	13%
email-EuAll: Email comm. netw., EU res.	265,214	365,030	•0.32	8	6.09	•5	0.51	•5	0.05	81%
soc-Epinions1: Who-trusts-whom netw.	75,888	405,740	•0.08	5	1.60	5	0.14	•4	0.02	30%
web-Google: Web graph from Google	875,713	4,322,051	•5.61	17	163.49	•11	14.67	•11	1.03	40%
flickr: 2005 crawl of flickr.com by D. Gleich	820,878	9,837,214	•1.32	•5	59.70	6	3.47	•5	0.61	40%
lung2: Transp. in lung, Uni. Aukland	109,460	492,564	4.84	227	3.39	•10	•1.48	26	0.06	59%
xenon2: Complex zeolite, sodalite crystals	157,464	1,933,344	0.96	35	3.81	•5	•0.78	10	0.16	59%
		Total •	4	1	0	6	4	4		

Name / # graphs / avg. # nodes	Hashing	WL	CGCR	S
MUTAG / 188 / 17.93	•0.23	0.53	0.6	–
ENZYMES / 600 / 29.87	•0.64	3.46	2.08	–
NCI1 / 111 / 29.87	•5.25	16.07	93.81	–
Weighted MUTAG	–	–	•0.40	–
Weighted ENZYMES	–	–	•1.82	–
Weighted NCI1	–	–	•111.53	–

Why should I care?

- Well, this suggests to view lifting as an approach for clustering, community detection, and role discovery

Power Iteration Clustering with Restarts

- One iteration of Power Iterated WL using k-Means instead of exact clustering essentially mimics Power Iterated Clustering [Lin, Cohen ICML 2010]

Algorithm 6: PICGCR(A): Power Iterated CGCR

$B^{(0)} = \mathbf{1}$;
$m^{(0)} := 1$ (the current maximal color) and $k := 1$;
repeat
  $B^{(k+1)} :=$ **PIC(l+k,A,B$^{(k)}$)**
  Set $m^{(k+1)}$ to the number of columns of $B^{(k+1)}$;
  $k := k + 1$;
until **Clusters are not changing significantly anymore**
return $B^{(k)}$

kMeans with l clusters over all colored early state distribution

Empirical Illustration

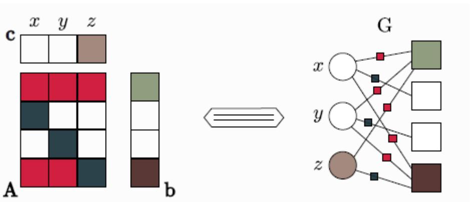
Dataset	k	PIC Purity	PIC NMI	PIC RI	PICWR Purity	PICWR NMI	PICWR RI
Iris	3	**0.9800**	**0.9306**	**0.9741**	**0.9800**	**0.9306**	**0.9741**
PenDigits01	3	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenDigits17	3	**0.7550**	**0.2066**	**0.6300**	**0.7550**	**0.2066**	**0.6300**
PolBooks	3	0.8000	0.4641	0.7702	**0.8667**	**0.6205**	**0.8405**
UBMCBlog	2	0.9480	0.7193	**0.9014**	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.9566	0.7426	0.9170	**0.9574**	**0.7492**	**0.9185**
20ngA	2	**0.9600**	**0.7594**	**0.9232**	**0.9600**	**0.7594**	**0.9232**
20ngB	2	0.8800	0.5563	0.7888	**0.9450**	**0.7042**	**0.8961**
20ngC	3	**0.6433**	**0.4955**	**0.6923**	0.6417	0.4932	0.6902
20ngD	4	0.5425	0.2979	0.6538	**0.5637**	**0.3283**	**0.6845**

Clusters are nothing but fix budget fractional automorphisms of datasets

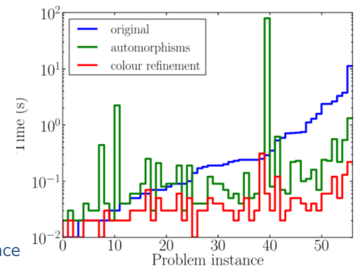
[Mladenov, Ahmadi, Kersting AISTATS 2012]

If you still do not care, what about **lifting linear programs,** working horse of AI, OR, …

$$\max_{[x,y,z]^T\in\mathbb{R}^3} \quad 0x+0y+1z$$
$$\text{s.t.} \quad \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$



Run WL on the „LP"-graph to reduce the dimension of the LP



Lessons learnt

- Loopy Belief Propagation and Linear Programming can be made aware of computational symmetries
- This can result in great speed-ups
- Computational symmetries can be detected using the Weisfeiler-Lehman (WL) algorithm
- WL computes fractional automorphisms in quasi-linear time; essentially no overhead!
- Few lines of Matlab code realize WL (with flooding) using sparse-matrix operations
- Strong connections to community detection, role discovery, graph kernels, clustering, …

St. Paul's Cathredal, London, UK