

Generating and Sampling Orbits for Lifted Probabilistic Inference

Steven Holtzen, Todd Millstein, Guy Van den Broeck

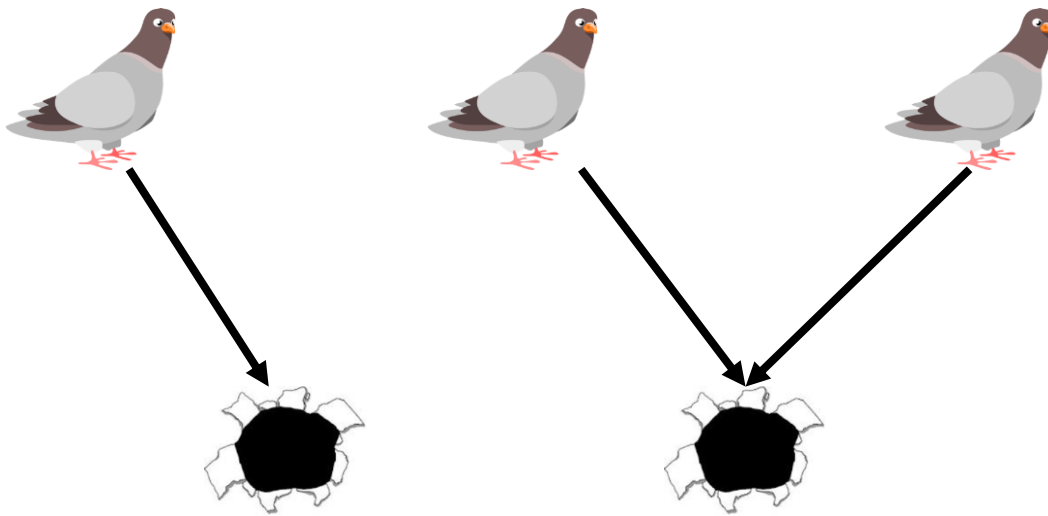
Computer Science Department, University of California, Los Angeles

`{sholtzen, todd, guyvdb}@cs.ucla.edu`



Motivation: The Pigeonhole Distribution

- Suppose there are 3 pigeons...



Each *dislikes* being placed into the same hole...

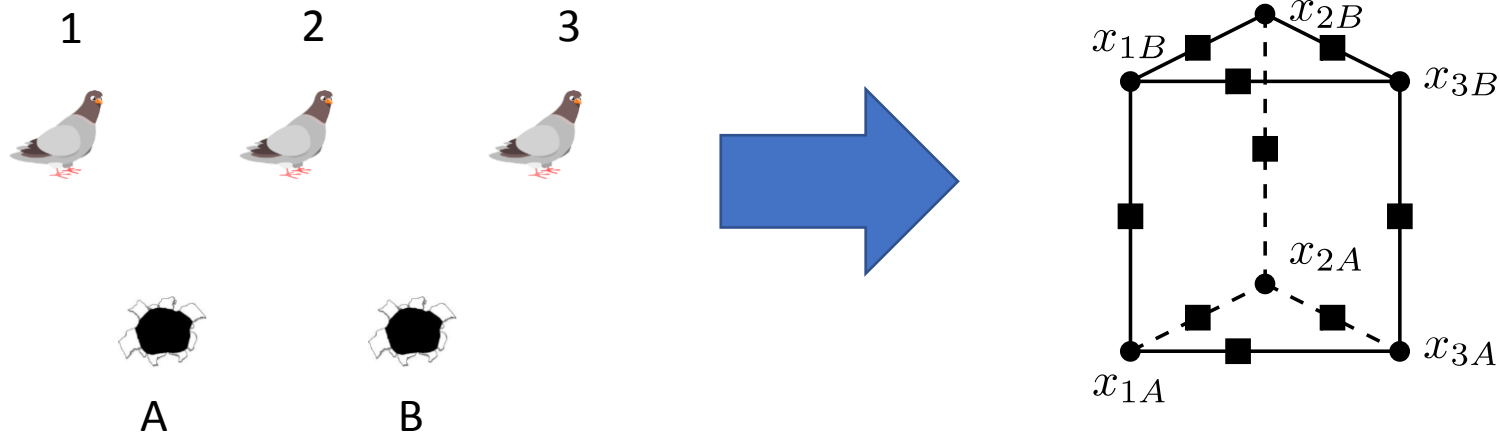
...no *quantum pigeons*, pigeons hiding in multiple holes simultaneously



What is the probability that k pigeons are placed into the same hole?
Requires computing partition (i.e., counting); does this seem *hard*?

Motivation: Encoding to Factor Graphs

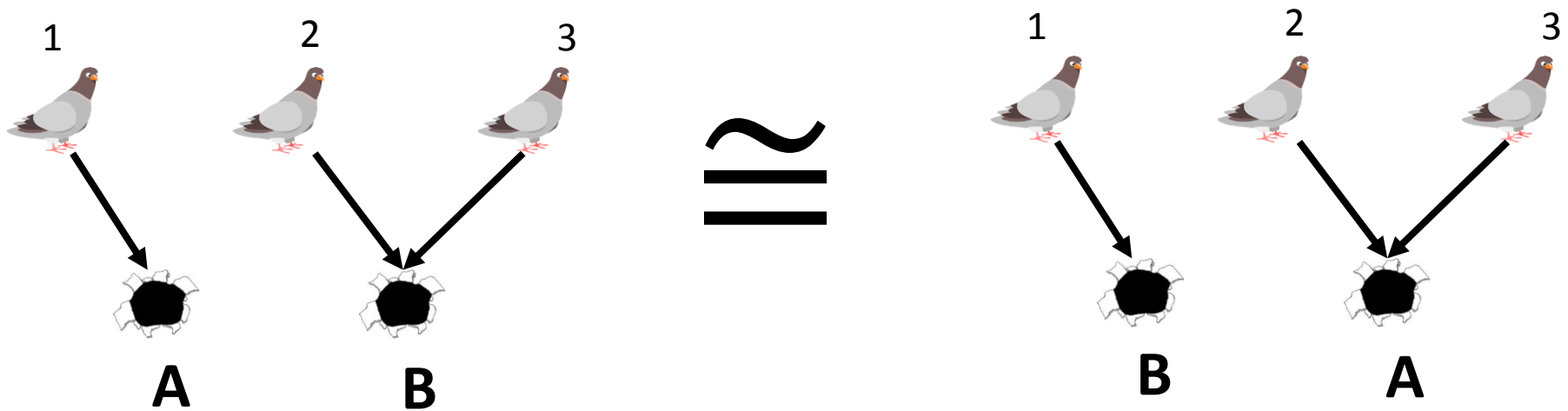
- One way to answer queries: convert to factor graph



- Problem: Factor graph is ***dense***; little conditional independence
 - Join-tree, variable elimination, etc. fail
- Is hope lost? What kind of ***structure*** is there to exploit?

Symmetry is Structure Too

- Pigeons and holes are **exchangeable**: relabeling them does not change the probability

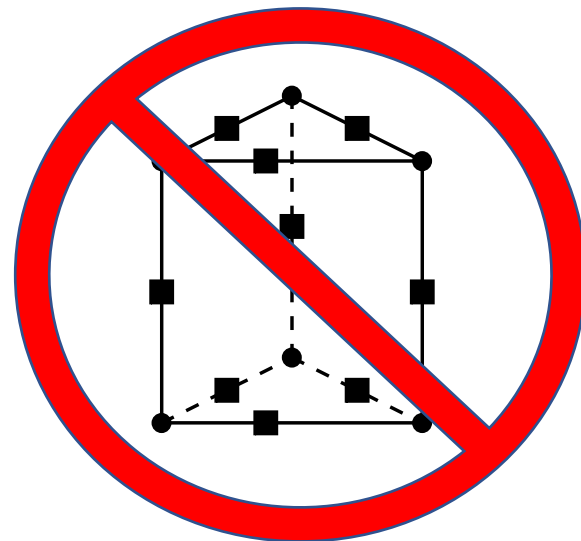
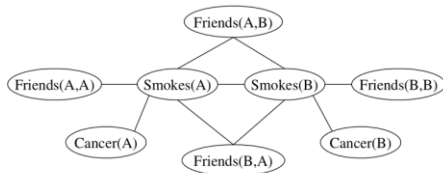


- These two states are in the **same orbit**
- Dramatically reduces state space of the problem

Related Work: Lifted Inference

- Lifted inference scales in degree of symmetry
 - Scales to large dense problems
 - Orthogonal to independence
- Problem:** Exact lifted inf. requires relational representation
 - Cannot handle factor graphs

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z \text{ Fr}(x, y) \wedge \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$	$\neg \text{Fr}(x, y) \vee \neg \text{Fr}(y, z) \vee \text{Fr}(x, z)$	0.7
Friendless people smoke.	$\forall x (\neg (\exists y \text{ Fr}(x, y)) \Rightarrow \text{Sm}(x))$	$\text{Fr}(x, g(x)) \vee \text{Sm}(x)$	2.3
Smoking causes cancer.	$\forall x \text{ Sm}(x) \Rightarrow \text{Ca}(x)$	$\neg \text{Sm}(x) \vee \text{Ca}(x)$	1.5
If two people are friends, either both smoke or neither does.	$\forall x \forall y \text{ Fr}(x, y) \Rightarrow (\text{Sm}(x) \Leftrightarrow \text{Sm}(y))$	$\neg \text{Fr}(x, y) \vee \text{Sm}(x) \vee \neg \text{Sm}(y),$ $\neg \text{Fr}(x, y) \vee \neg \text{Sm}(x) \vee \text{Sm}(y)$	1.1



[Richardson, Matthew, and Pedro Domingos. "Markov logic networks." Machine learning 62.1-2 (2006): 107-136.]

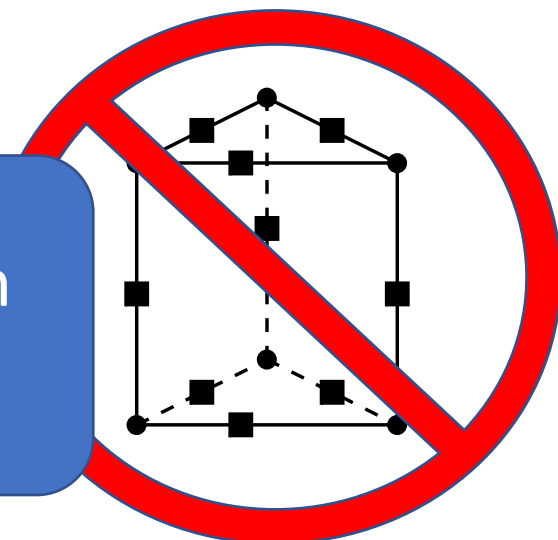
Related Work: Lifted Inference

- Lifted inference scales in degree of symmetry
 - Scales to large dense problems
 - Orthogonal to independence
- **Problem:** Exact lifted inf. requires relational representation
 - Cannot handle factor graphs

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z \text{ Fr}(x, y) \wedge \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$	$\neg \text{Fr}(x, y) \vee \neg \text{Fr}(y, z) \vee \text{Fr}(x, z)$	0.7
Friendless people smoke.	$\forall x (\neg (\exists y \text{ Fr}(x, y)) \Rightarrow \text{Sm}(x))$		
Smoking causes cancer.	$\forall x \text{ Sm}(x) \Rightarrow \text{Ca}(x)$		
If two people are friends, either both smoke or neither does.	$\forall x \forall y (\text{Fr}(x, y) \Rightarrow (\text{Sm}(x) \wedge \text{Sm}(y) \vee \neg \text{Sm}(x) \wedge \neg \text{Sm}(y)))$		

Friends(A,A)
Can

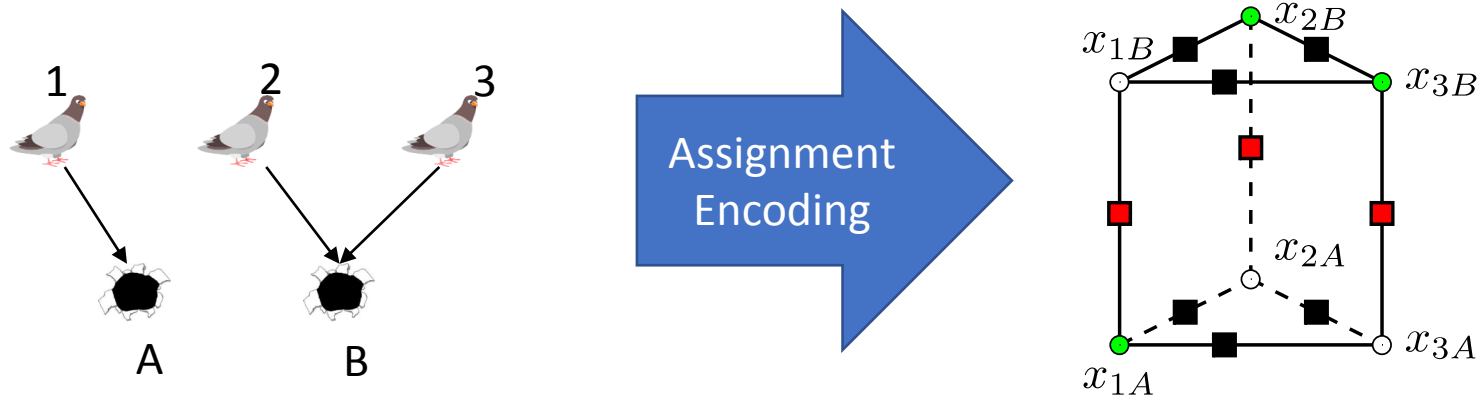
How can we exploit symmetry in exact factor graph inference?



[Richardson, Matthew, and Pedro Domingos. Markov logic networks. Machine learning 62.1-2 (2006): 107-136.]

Our Key Insight: Colored Assignment Encodings

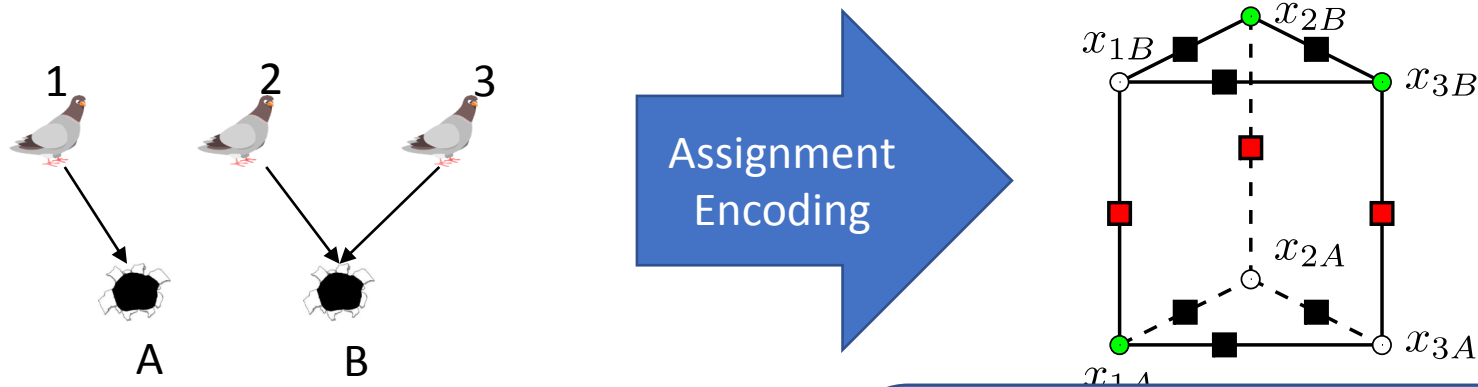
- Assignments have a natural colored encoding



- Black factors: Each pigeon dislikes being placed into the same hole
- Red factors: no quantum pigeons
- Green = true variable, red = false variable

Our Key Insight: Colored Assignment Encodings

- Assignments have a natural colored encoding



- Black factors: Each pigeon fits in the same hole
- Red factors: no quantum pigeon
- Green = true variable, red = false

Represent symmetries of distribution through isomorphisms of graph [Kersting et al., 2009, Niepert, 2012, 2013, Bui et al., 2013]

Contribution

- Two new algorithms:

Orbit Generation

First example of exact lifted inference for arbitrary discrete factor graphs

Orbit-Jump MCMC

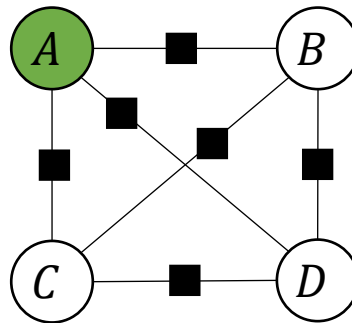
Approximate lifted inference that *mixes rapidly** in number of orbits

Orbit Generation

Exact lifted inference for factor graphs

A Simpler Example

- Consider a complete factor graph



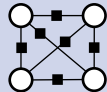
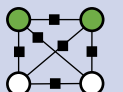
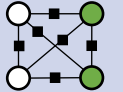
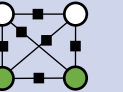
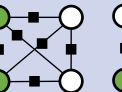
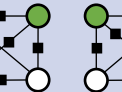

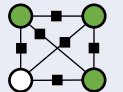
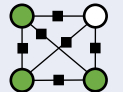
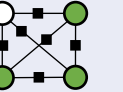
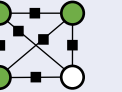
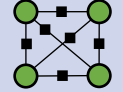
- If all factors *identical* and *symmetric*,

$$\text{then } \Pr(\text{graph with } A \text{ green}) = \Pr(\text{graph with } B \text{ green}) = \Pr(\text{graph with } C \text{ green}) = \Pr(\text{graph with } D \text{ green})$$

- Probability is determined by *number of true states*

Orbits of Factor Graphs

• $\Pr(\text{graph with top-left green}) = \Pr(\text{graph with top-right green}) = \Pr(\text{graph with bottom-left green}) = \Pr(\text{graph with bottom-right green})$

Orbit #	Elements of the Orbit
0	
1	
2	     
3	   
4	

Exact lifted inference algorithm

- If we can:

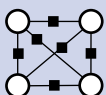
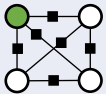
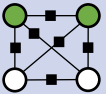
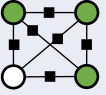
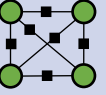
1. Efficiently generate one element of each orbit,
2. Efficiently compute the size of each orbit

- Then, the partition function can be computed efficiently in the number of orbits (Theorem 4.1)

Let's see an example...

Exact Lifted Inference

1. Efficiently find one *representative* of each orbit
2. Compute the size of the orbit

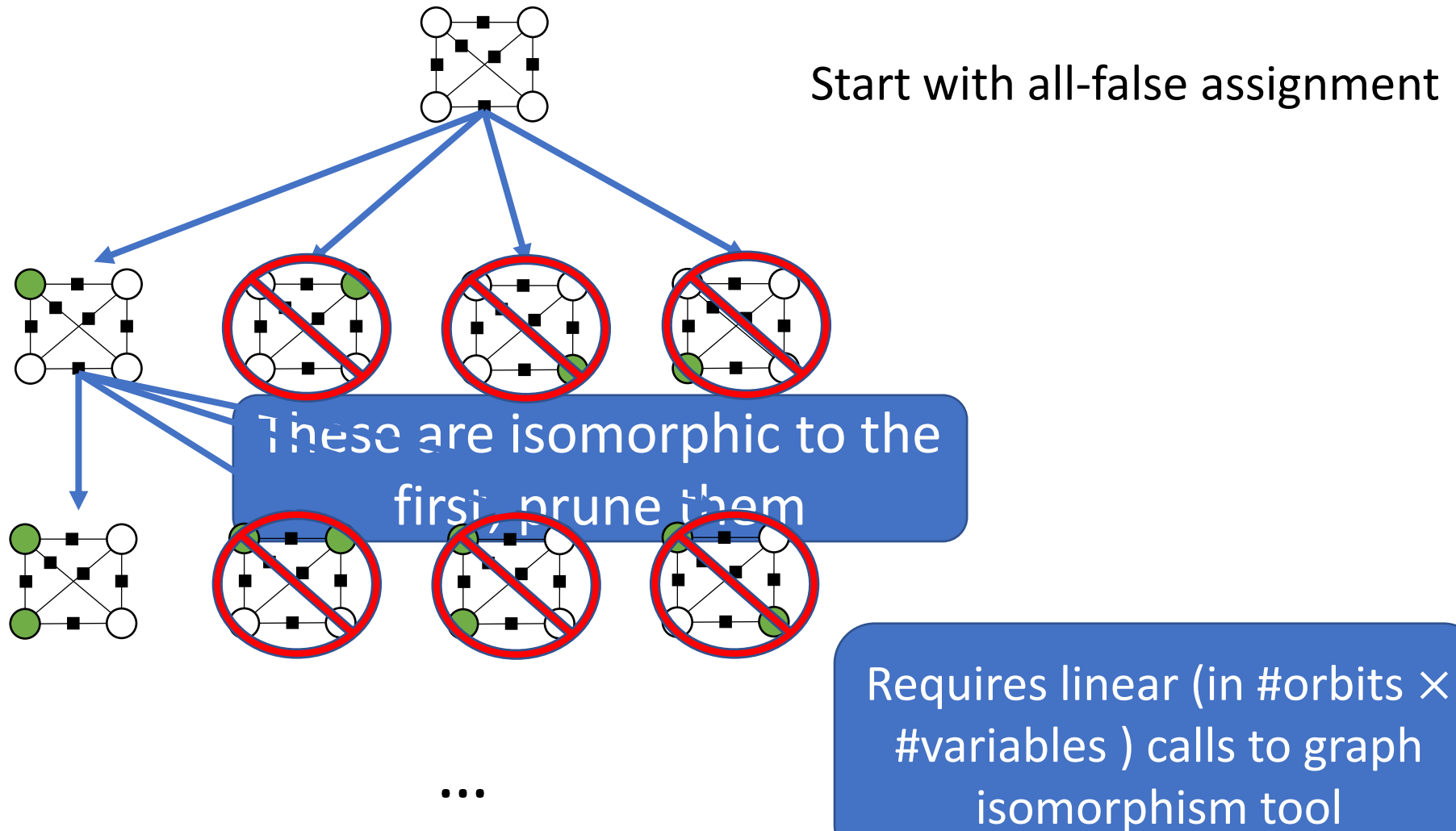
Orbit #	Orbit Repr.	Unnormalized State Probability	Orbit Size	Total Orbit Unnormalized
0		5	1	$5 \times 1 = 5$
1		13	4	$13 \times 4 = 52$
2		21	6	21×6
3		2	4	$= 126$
4		3	1	$2 \times 4 = 8$
				$3 \times 1 = 3$

- $Z = \sum \text{unnormalized} = 5 + 52 + 126 + 8 + 3 = 193$

Exact lifted inference algorithm

- 1. Efficiently generate one element of each orbit,**
2. Efficiently compute the size of each orbit

Orbit Generation: Breadth-First Search



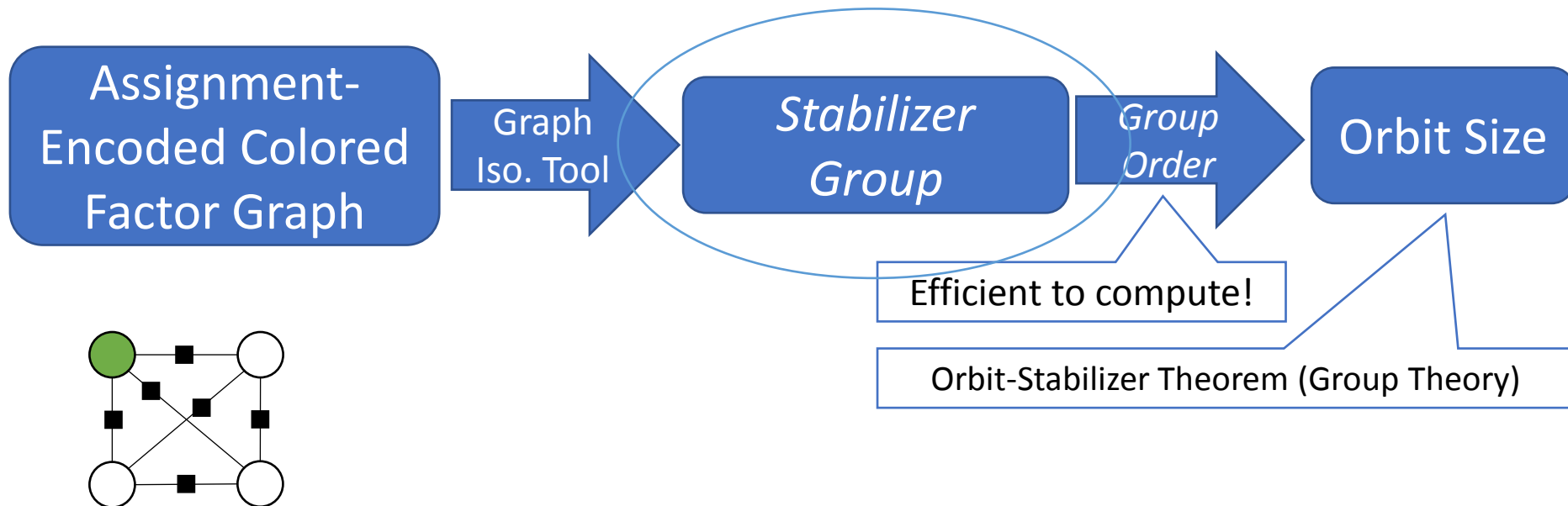
Exact lifted inference algorithm

1. Efficiently generate one element of each orbit, ✓
2. **Efficiently compute the size of each orbit**

- Seems #P-hard at first, but in fact is not
- Use graph isomorphism tools to *count things*

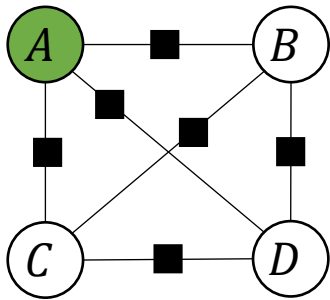
Orbit Size Pipeline

- Avoid enumerating the whole orbit



Stabilizer Group

- Question: Which isomorphisms *preserve* (***stabilize***) this coloring?



Answer: Any permutation of $\{B, C, D\}$

Assignment-
Encoded Colored
Factor Graph

Graph
Iso. Tool

Stabilizer Group:
Small set of
generators

Group
Order

Orbit Size

Orbit-Stabilizer Theorem

- Relates size of orbit to *order (size) of stabilizer*

Orbit size $\left(\begin{array}{c} A \\ \square \\ C \end{array} \begin{array}{c} B \\ \square \\ D \end{array} \right) = \frac{\text{\#ways of permuting } \{A, B, C, D\}}{\text{\#ways of permuting } \{B, C, D\}} = \frac{4!}{3!} = 4$

- Computing the order of a group is a *standard problem* in computational group theory
 - Efficient to compute (in size of graph)

Stabilizer Group:
Small set of
generators

GAP

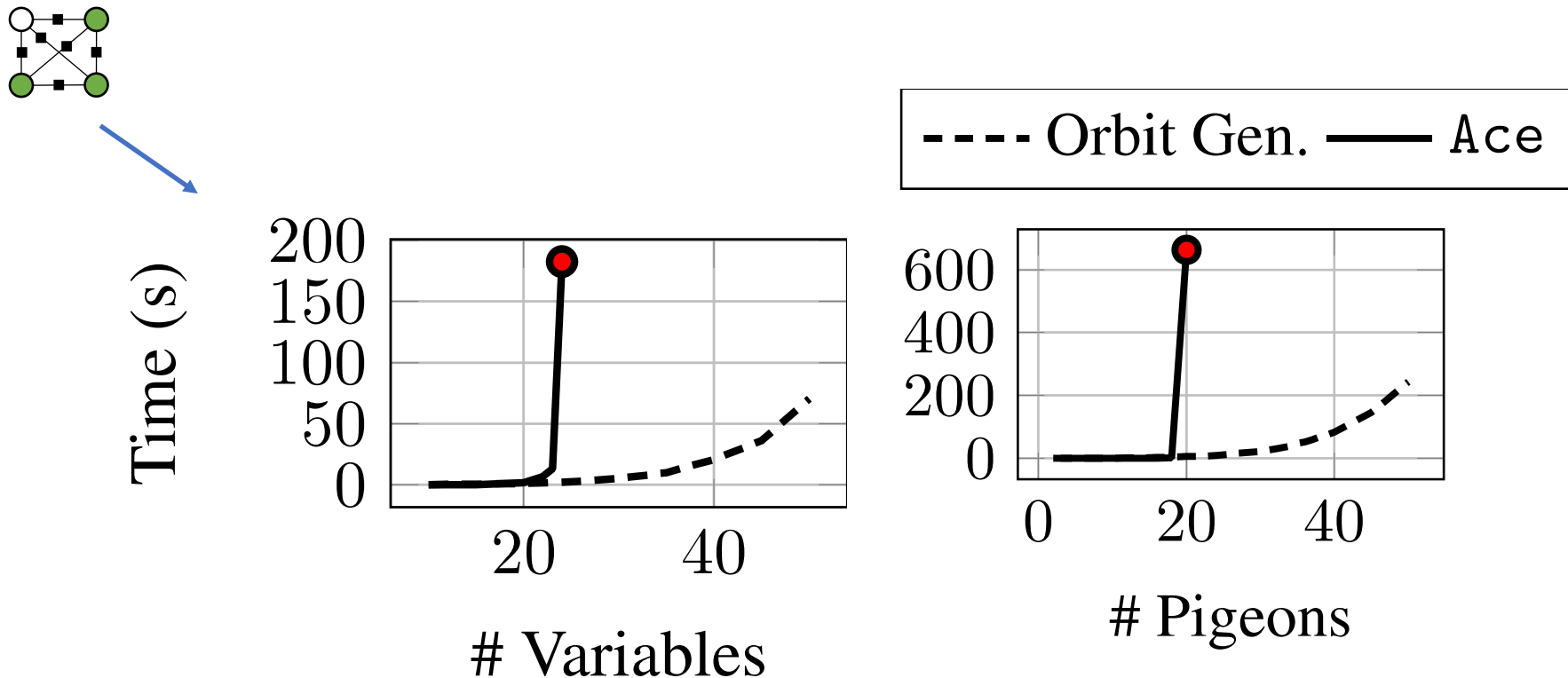
Order of group



4×10^{84} states

Exact Inference Experiments

- Proof of concept: Compared against existing exact inference tool, ACE

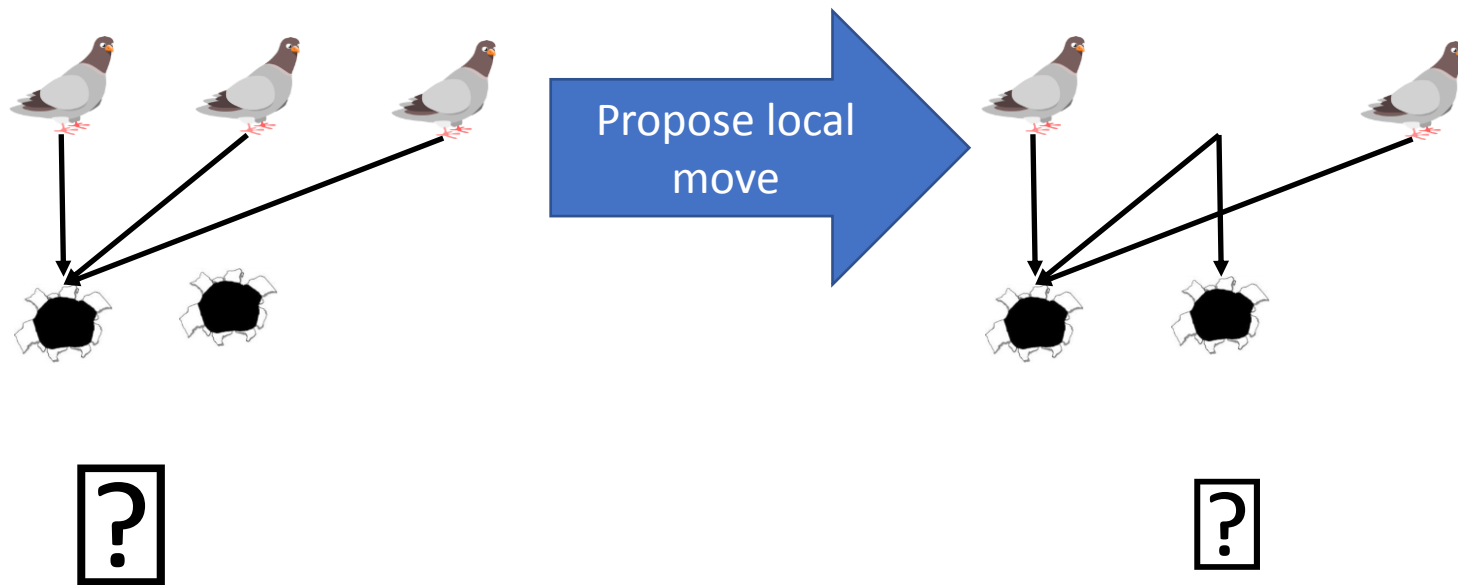


Orbit-Jump MCMC

Approximate lifted inference with mixing time guarantees

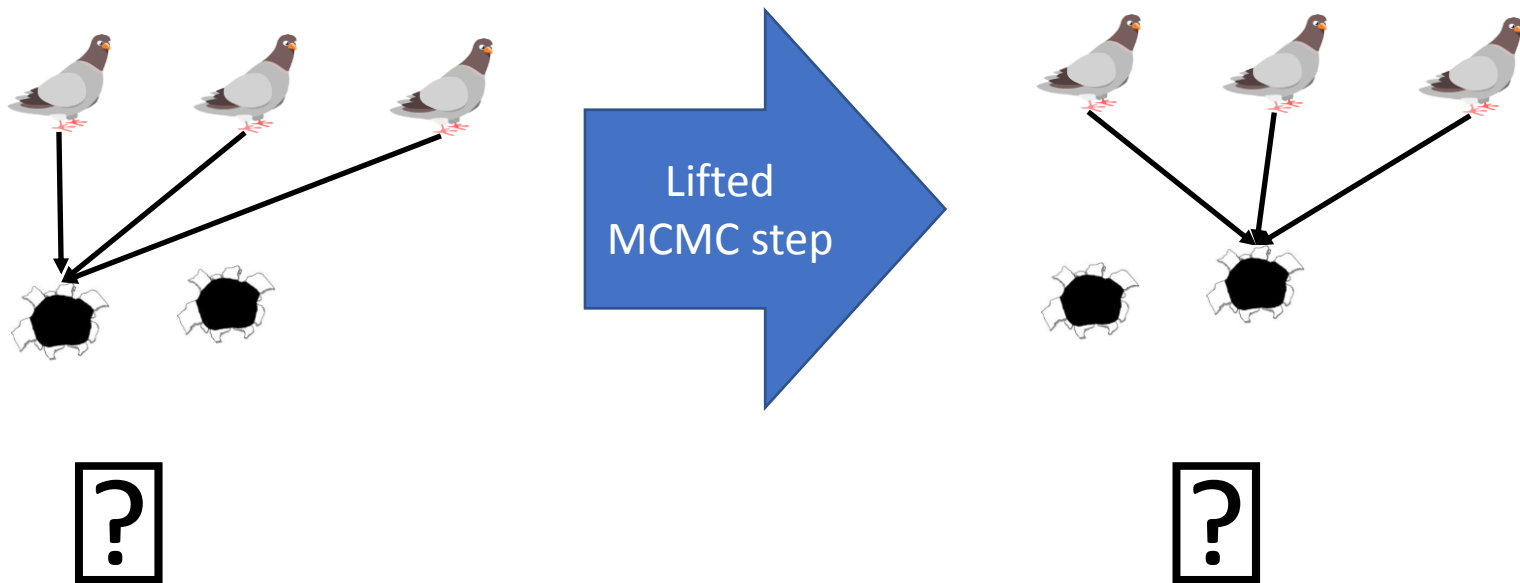
Motivation

- Local-search (e.g. Gibbs sampling) can get stuck



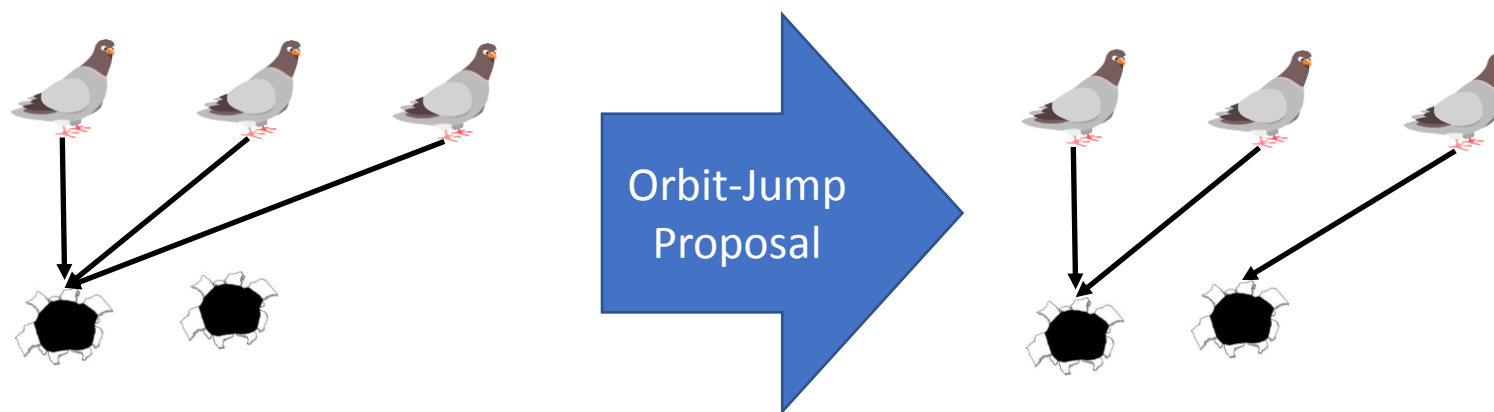
Related Work: Within-Orbit Jumps

- Lifted MCMC [Niepert, 2012, 2013] jumps *within* orbits, unfortunately doesn't help here



Jumping Between Orbits

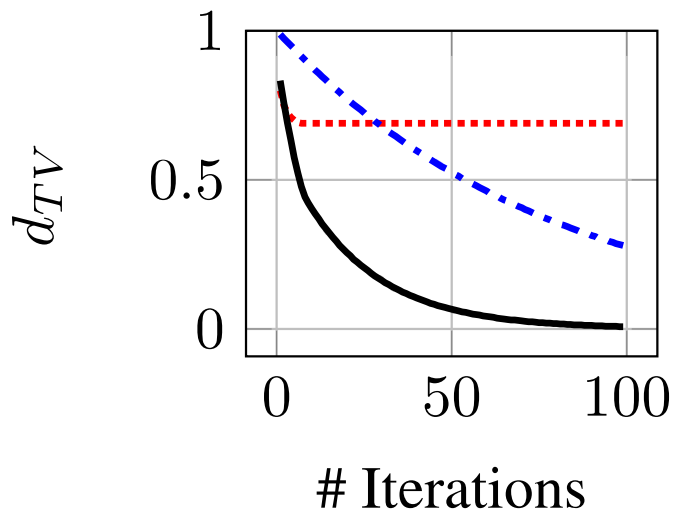
- Orbit-Jump MCMC proposes jumps *between orbits*



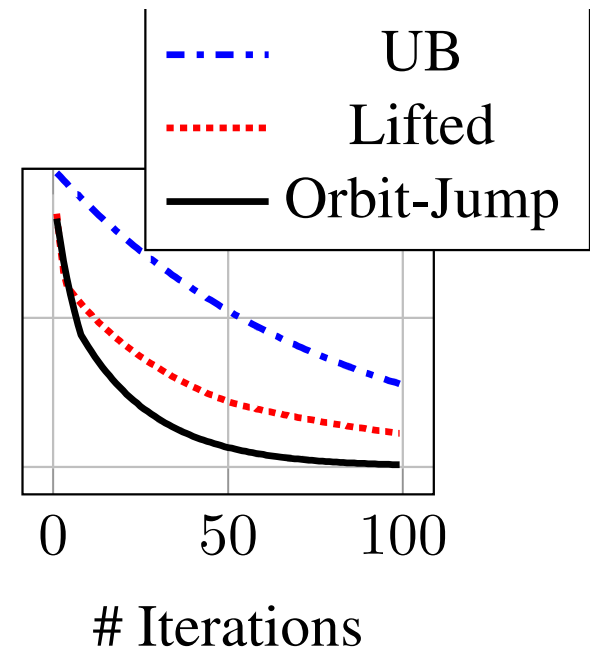
- We show how to jump between orbits using the *Burnside process*
- Requires multiple graph isomorphism + group order computations
- More expensive than lifted MCMC, with mixing rate guarantees

Orbit-Jump MCMC Mixing Time

- Empirical mixing time, 5 pigeons 2 holes
 - Total variation distance from stationary dist.



(a) Hard pigeonhole.



(b) Quantum pigeonhole.

Conclusion

- Some distributions have little independence, but inference remains tractable
 - *Symmetry* complements independence
- This work develops symmetry as a source of tractability for factor graph inference
 - First *exact lifted inference* for factor graphs
 - Orbit-Jump MCMC algorithm, mixes rapidly in #orbits (with some caveats)

Grand challenge: Integrating independence and symmetry into a single algorithm for factor graphs

Steven Holtzen and Todd Millstein and Guy Van den Broeck

Computer Science Department
University of California, Los Angeles
{sholtzen,todd,guyvdb}@cs.ucla.edu

Abstract

A key goal in the design of probabilistic inference algorithms is identifying and exploiting properties of the distribution that make inference tractable. Lifted inference algorithms identify *symmetry* as a property that enables efficient inference and seek to scale with the degree of symmetry of a probability model. A limitation of existing exact lifted inference techniques is that they do not apply to non-relational representations like factor graphs. In this work we provide the first example of an exact lifted inference algorithm for arbitrary discrete factor graphs. In addition we describe a lifted Markov-Chain Monte-Carlo algorithm that provably mixes rapidly in the degree of symmetry of the distribution.

that scale in the number of distinct orbits. Highly symmetric distributions have few orbits relative to the size of their state space, allowing lifted inference algorithms to scale to large probability distributions with scant independence. Thus, lifted inference algorithms identify symmetry as a complement to independence in the search for efficient inference algorithms.

An important challenge in designing lifted inference algorithms is identifying symmetries of a probability distribution from its high-level description. Existing exact lifted inference algorithms rely on relational structure to extract symmetries, and thus cannot be directly applied to propositional probability models like factor graphs [Getoor and Taskar, 2007]. Several approximate lifted inference algorithms ease this requirement by extracting symmetries of the probability distribution by computing an automorphism group of a graph, and can thus be applied directly to factor graphs [Kersting et al., 2009, Niepert, 2012, 2013, Bui et al., 2013]. However, existing lifted MCMC algorithms are not guaranteed to mix rapidly in the number of orbits.

This paper presents exact and approximate lifted inference algorithms for arbitrary factor graphs that provably scale with the number of orbits of the probability distribution. Inspired by the success of existing approximate lifted inference techniques on graphical models, we apply graph isomorphism tools to extract the necessary symmetries. First, we present a motivating example that highlights the strengths and weaknesses of our approach. Then, we describe our exact inference procedure. Computationally, our method combines efficient group theory libraries like GAP [GAP] with graph isomorphism tools.

Next, we describe an approximate inference algorithm called *orbit-jump MCMC* that provably mixes quickly in the number of orbits of the distribution. Orbit-jump MCMC provides an alternative to lifted MCMC [Niepert, 2012, 2013], a family of approximate lifted inference algorithms that compute a single graph automorphism in

1 INTRODUCTION

Probabilistic inference is fundamentally computationally hard in the worst case [Roth, 1996]. Thus, designers of probabilistic inference algorithms focus on identifying and exploiting sufficient conditions of the distribution that ensure tractable inference. For instance, many existing probabilistic inference strategies for graphical models exploit independence in order to scale efficiently [Koller and Friedman, 2009, Darwiche, 2009]. The performance of these algorithms is worst-case exponential in a graph metric known as the *treewidth* that quantifies the degree of independence in the graph.

Lifted inference algorithms identify *symmetry* as a key property that enables efficient inference [Poole, 2003, Kersting, 2012, Niepert and Van den Broeck, 2014]. These methods identify *orbits* of the distribution: sets of points in the probability space that are guaranteed to have the same probability. This enables inference strategies

Thank you!

Questions? Comments?

sholtzen@cs.ucla.edu