Generating and Sampling Orbits for Lifted Probabilistic Inference

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.

1 INTRODUCTION

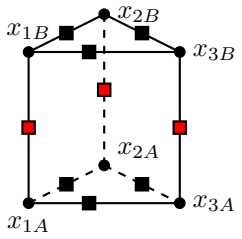
Probabilistic inference is fundamentally computationally hard in the worst case [Roth, 1996]. Thus, designers of probabilistic inference algorithms focus on identifying and exploiting sufficient conditions of the distribution that ensure tractable inference. For instance, many existing probabilistic 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 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



(a) The colored factor graph encoding of the soft pigeonhole problem.

000 000

100 000
010 000
001 000
000 100
000 010
000 001

110 000 011 000 101 000
000 110 000 011 000 101

100 100 010 010 001 001

100 010 100 001
010 100 001 100

110 100 110 010 011 001
100 110 010 110 001 011

111 000 000 111

110 001 101 010 011 100
001 110 010 101 100 011

(b) Orbits of the assignments to variables in the soft pigeonhole problem. An assignment is written as a binary string $x_{1A}x_{2A}x_{3A}x_{1B}x_{2B}x_{3B}$. Each orbit is boxed, and each canonical representative is bold. Cases where there are 4 or greater true variable assignments are omitted, as these are symmetric to previously listed cases where the true and false values are flipped.

Figure 1: A graphical model representation and orbit structure of the pigeonhole example with 3 pigeons and 2 holes.

order to quickly transition *within* orbits. A key advantage of lifted MCMC is that transitions do not each require a call to a graph isomorphism tool. However, lifted MCMC relies on Gibbs sampling to jump *between orbits*, and as a consequence has no guarantees about its mixing time in terms of the number of orbits. We will show that orbit-jump MCMC mixes rapidly in the number of distinct orbits, at the cost of requiring multiple graph isomorphism calls for each transition.

Note, however, that purely scaling in the number of orbits is not a panacea. Our methods are both limited: there are liftable probability models that still have too many orbits for our methods to be effective. The presented methods *only* exploit symmetry, which is in contrast to existing exact lifted inference algorithms that simultaneously exploit symmetry and independence. Therefore, our algorithms scale exponentially for certain well-known liftable distributions, such as the friends and smokers Markov logic network [Niepert and Van den Broeck, 2014]. Thus, we view this work as providing a foundation for future work on inference for factor graphs that exploits both symmetry and independence.

2 MOTIVATION

As a motivating example, consider performing exact lifted probabilistic inference on a probabilistic version of the pigeonhole problem. The pigeonhole problem is a well-studied problem from automated reasoning that exhibits nuanced symmetry. While seemingly simple, the pigeonhole problem is in fact extremely challenging to reason about, and is often used as a benchmark in automated reasoning tasks [Benhamou and Sais, 1994, Sabharwal, 2005, Raz, 2004]. A *weighted set of clauses* is a set of pairs $\Delta = \{(w, f)\}$ where f is a Boolean clause and $w \in \mathbb{R}$ is a weight. A weighted set of clauses defines a probability distribution over assignments $\mathbf{x}$ of variables in Δ according to the following:

$$\Pr(\mathbf{x}) = \frac{1}{Z} \exp\left[\sum_{\{(w,f)\in\Delta \mid \mathbf{x}\models f\}} w\right],$$

where Z is a normalizing constant, and $\mathbf{x} \models f$ denotes that clause f is satisfied in world $\mathbf{x}$. Our goal is to compute Z, a task that is #P-hard in general.

Consider a set of weighted clauses for a *soft pigeonhole problem*. There are n pigeons and m holes. Each pigeon can occupy at most one hole, and pigeons prefer to be solitary. To encode this situation as a weighted set of clauses, for each pigeon i and hole j, let x_{ij} be a Boolean variable that is true if and only if pigeon i occupies hole j.

The set Δ is a union of two sets of weighted clauses. For each pigeon, we introduce a clause that forces it to occupy at most a single hole:

$$(\infty,\ \overline{x_{ik}} \vee \overline{x_{il}}) \text{ for each pigeon } i \text{ and holes } k \neq l. \quad (1)$$

An infinite weight encodes a *hard clause* that must hold in the distribution [Richardson and Domingos, 2006]. Then, for each hole we introduce clauses that assign a positive weight to not having multiple pigeons:

$$(2,\ \overline{x_{kj}} \vee \overline{x_{lj}}) \text{ for each hole } j \text{ and pigeons } k \neq l. \quad (2)$$

Figure 1a depicts this probability distribution with 3 pigeons and 2 holes as a pairwise colored factor graph, where each weighted clause is a factor (box) and each distinct factor is given its own color [Niepert, 2012, Bui et al., 2013]. The factors in Equation 1 are colored red, and the factors in Equation 2 are colored black.

The symmetries of this probability distribution directly correspond to automorphisms of the colored graph in Figure 1a [Bui et al., 2013, Niepert, 2012]. In this paper, we consider only symmetries on assignments that

arise from symmetries on the variables. Any permutation of vertices that preserves the graph structure leaves the distribution unchanged.[1] Two assignments that are reachable from one another via a sequence of permutations are in the same *orbit*; all assignments in the same orbit thus have the same probability. Figure 1b shows the orbits of the 3-pigeon 2-hole scenario up to inversion of true and false assignments. Each orbit is boxed. There are few orbits relative to the number of states, which is the property that our lifted inference algorithms exploit.

We present both exact and approximate inference strategies that scale with the number of orbits of a probability distribution. Our exact inference algorithm is as follows. First, generate a single *canonical representative* from each orbit; in Figure 1b, canonical representatives are shown in bold. Then for each representative, compute the size of its orbit. If both of these steps are efficient, then this inference computation scales efficiently with the number of orbits, and we call it lifted. This *orbit generation* procedure is at the heart of many existing lifted inference algorithms that construct sufficient statistics of the distribution from a relational representation [Niepert and Van den Broeck, 2014]. We present an exact lifted inference algorithm in Section 4 that applies this methodology to arbitrary factor graphs by using graph isomorphism tools to generate canonical representatives and compute orbit sizes.

Next, in Section 5 we describe an approximate inference algorithm called *orbit-jump MCMC* that provably mixes quickly in the number of distinct orbits of the distribution. This algorithm uses as its proposal the *uniform orbit distribution*: the distribution defined by choosing an orbit of the distribution uniformly at random, and then choosing an element within that orbit uniformly at random. We present a novel application of the *Burnside process* in order to draw samples from the uniform orbit distribution [Jerrum, 1993], and show how to implement the Burnside process on factor graphs by using graph isomorphism tools. Thus, this orbit-jump MCMC provides an alternative to lifted MCMC that trades computation time for provably good sample quality.

3 BACKGROUND

This section gives a brief description of important concepts from group theory and approximate lifted inference that will be used throughout the paper.[2]

[1] We assume here w.l.o.g. but for simplicity that the factors are individually fully symmetric. Asymmetric factors can either be made symmetric by duplicating variable nodes [Niepert, 2012] or encoded using colored edges [Bui et al., 2013].

[2] See Appendix A for a summary of the notation.

3.1 GROUP THEORY

We review some standard terminology and notation from group theory, following Artin [1998]. A *group* $\mathcal{G}$ is a pair $(S, \cdot)$ where S is a set and $\cdot : S \times S \to S$ is a binary associative function such that there is an identity element and every element in S has an inverse under $(\cdot)$. The *order* of a group is the number of elements of its underlying set, and is denoted $|\mathcal{G}|$. A *permutation group acting on a set* Ω is a set of bijections $g : \Omega \to \Omega$ that forms a group under function composition. For $\mathcal{G}$ acting on Ω, a function $f : \Omega \to \Omega'$ is *$\mathcal{G}$-invariant* if $f(g \cdot x) = f(x)$ for any $g \in \mathcal{G}, x \in \Omega$. Two elements $x, x' \in \Omega$ are in the same *orbit* under $\mathcal{G}$ if there exists $g \in \mathcal{G}$ such that $x = g \cdot x'$. Orbit membership is an equivalence relation, written $x \sim_{\mathcal{G}} x'$. The set of all elements in the same orbit is denoted $\mathrm{Orb}_{\mathcal{G}}(x)$. A *stabilizer* of x is an element $g \in \mathcal{G}$ such that $g \cdot x = x$; the set of all stabilizers of x is a group called the *stabilizer subgroup*, denoted $\mathrm{Stab}_{\mathcal{G}}(x)$. The subscript in the previous notation is elided when clear. A *cycle* $(x_1\ x_2\ \cdots\ x_n)$ is a permutation $x_1 \mapsto x_2, x_2 \mapsto x_3, \cdots, x_n \mapsto x_1$. A permutation can be written as a product of disjoint cycles.

3.2 LIFTED PROBABILISTIC INFERENCE & GRAPH AUTOMORPHISMS

Lifted inference relies on the ability to identify the symmetries of probability distributions. In existing exact lifted inference methods, the symmetries are evident from the relational structure of the probability model [Poole, 2003, De Salvo Braz et al., 2005, Gogate and Domingos, 2011, Van den Broeck, 2013]. In order to extend the insights of lifted inference to models where the symmetries are less accessible, many lifted approximation algorithms rely on graph isomorphism tools to identify the symmetries of probability distributions [Niepert, 2012, Bui et al., 2013, Mckay and Piperno, 2014].

A *colored graph* is a 3-tuple $\mathtt{G} = (V, E, C)$ where (V, E) are the vertices and edges of an undirected graph and $C = \{V_i\}_{i=1}^{k}$ is a partition of vertices into k sets. As notation, for a vertex v, let $\mathtt{color}(v, C) = i$ if $v \in V_i$. A colored graph automorphism is an edge and color-preserving vertex automorphism:

Definition 3.1. Let $\mathtt{G} = (V, E, C)$ and $\mathtt{G}' = (V, E', C')$ be colored graphs. Then $\mathtt{G}$ and $\mathtt{G}'$ are *color-automorphic* to one another, denoted $\cong$, if there exists a bijection $\phi : V \to V$ such that

1. Vertex neighborhoods are preserved, i.e. for any $v_1, v_2 \in V$, $(v_1, v_2) \in E \Leftrightarrow (\phi(v_1), \phi(v_2)) \in E'$;
2. Vertex colors are preserved, i.e. for all $v \in V$, $\mathtt{color}(v, C) = \mathtt{color}(\phi(v), C')$.

The *color automorphism group* of a colored graph $\mathtt{G}$, denoted $\mathbb{A}(\mathtt{G})$, is the group formed by the set of color automorphisms of $\mathtt{G}$ under composition. The group $\mathbb{A}(\mathtt{G})$ acts on the vertices of $\mathtt{G}$ by permuting them. Tools like `Nauty` can compute $\mathbb{A}(\mathtt{G})$ and are typically efficient in $|V|$ [Mckay and Piperno, 2014].

Colored graph automorphism groups are related to factor graphs via the following:

Definition 3.2. Let $\mathcal{F} = (\mathbf{X}, F)$ be a factor graph with variables $\mathbf{X}$, and factors F, where F are symmetric functions on assignments to variables $\mathbf{X}$, written $\mathbf{x}$. Then the *colored graph induced by* $\mathcal{F}$ is a tuple (V, E, C) where $V = \mathbf{X} \cup F$, the set of edges E connects variables and factors in $\mathcal{F}$, and C is a partition such that (1) factor nodes are given the same color iff they are identical factors, and (2) variables are colored with a single color that is distinct from the factor colors.

This definition is due to Bui et al. [2013], where the following theorem is proved (with different terminology):

Theorem 3.1 (Bui et al. [2013], Theorem 2)**.** *Let $\mathcal{F}$ be a factor graph and $\mathtt{G}$ be its induced colored graph. Then, the distribution of $\mathcal{F}$ is $\mathbb{A}(\mathtt{G})$-invariant.*

4 EXACT LIFTED INFERENCE

In this section we describe our exact lifted inference procedure. First we discuss the group-theoretic properties of orbit generation that enable efficient exact lifted inference. Then, we describe our algorithm for implementing orbit generation on colored factor graphs. Finally, we present some case studies demonstrating the performance of our algorithm.

4.1 $\mathcal{G}$-INVARIANCE AND TRACTABILITY

In this section we describe the group-theoretic underpinnings of our orbit-generation procedure and describe its relationship with previous work on tractability through exchangeability. We will capture the behavior of a $\mathcal{G}$-invariant probability distribution on a set of *canonical representatives* of each orbit:

Definition 4.1. Let $\mathcal{G}$ be a group that acts on a set Ω. Then, there exists a set of *canonical representatives* $\Omega/\mathcal{G} \subseteq \Omega$ and surjective *canonization function* $\sigma : \Omega \to \Omega/\mathcal{G}$ such that for any $x, y \in \Omega$, (1) $\mathrm{Orb}(x) = \mathrm{Orb}(\sigma(x))$; and (2) $\mathrm{Orb}(x) = \mathrm{Orb}(y)$ if and only if $\sigma(x) = \sigma(y)$.

In statistics, σ is often called a *sufficient statistic* of a partially exchangeable distribution [Niepert and Van den Broeck, 2014, Diaconis and Freedman, 1980]. The motivating example hinted at a general-purpose solution for exact inference that proceeds in two phases. First, one constructs a representative of each orbit; then, one efficiently computes the size of that orbit. We can formalize this using group theory:

Theorem 4.1. *Let $\Pr$ be a $\mathcal{G}$-invariant distribution on Ω, and evidence $\mathbf{e} : \Omega \to \mathbb{B}$ be a $\mathcal{G}$-invariant function. Then, the complexity of computing the most probable explanation (MPE) is $\mathrm{poly}(|\Omega/\mathcal{G}|)$ if the following can be computed in $\mathrm{poly}(|\Omega/\mathcal{G}|)$:*

1. *Evaluate $\Pr(x)$ for $x \in \Omega$;*
2. *(Canonical generation) Generate a set of canonical representatives $\Omega/\mathcal{G}$,*

Moreover, if $|\mathrm{Orb}(x)|$ can be computed in $\mathrm{poly}(|\Omega/\mathcal{G}|)$, then $\Pr(\mathbf{e})$ can be computed in $\mathrm{poly}(|\Omega/\mathcal{G}|)$.

Proof. To compute the MPE, choose:

$$\arg\max_{\{x \in \Omega/\mathcal{G} \mid \mathbf{e}(x) = \mathtt{T}\}} \Pr(x). \tag{3}$$

The $\mathcal{G}$-invariance of $\mathbf{e}$ allows us to evaluate $\mathbf{e}$ on only x without considering other elements of $\mathrm{Orb}(x)$. To compute $\Pr(\mathbf{e})$, compute

$$\sum_{\{x \in \Omega/\mathcal{G} \mid \mathbf{e}(x) = \mathtt{T}\}} |\mathrm{Orb}(x)| \times \Pr(x). \tag{4}$$

Both of these can be accomplished in $\mathrm{poly}(|\Omega/\mathcal{G}|)$. □

Niepert and Van den Broeck [2014] identified a connection between bounded-width *exchangeable decompositions* and tractable (i.e., domain-lifted) exact probabilistic inference using the above approach. Exchangeable decompositions are a particular kind of $\mathcal{G}$-invariance. Let $\Pr(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n)$ be a distribution on sets of variables $\mathbf{X}_i$. Let S_n be a group of all permutations on a set of size n. Then, this distribution has an exchangeable decomposition along $\{\mathbf{X}_i\}$ if, for any $g \in S_n$:

$$\Pr(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n) = \Pr(\mathbf{X}_{g \cdot 1}, \mathbf{X}_{g \cdot 2}, \cdots, \mathbf{X}_{g \cdot n})$$

Niepert and Van den Broeck [2014] showed how to perform exact lifted probabilistic inference on any distribution with a fixed-width exchangeable decomposition by directly constructing canonical representatives. However, this construction does not generalize to other kinds of symmetries, and thus cannot be applied to factor graphs which may have arbitrarily complex symmetric structure. In the next section, we show how to apply Theorem 4.1 to factor graphs.

4.2 ORBIT GENERATION

The previous section shows that inference can be efficient if we can (1) construct representatives of each orbit

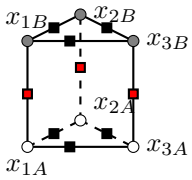


Figure 2: A colored graph of the 3-pigeon 2-hole problem that encodes the assignment $\mathbf{x} = 000\ 111$. True variable nodes are gray and false variable nodes are white.

class, (2) compute how large each orbit is. In this section, we give an algorithm for performing these two operations for colored factor graphs. First, we describe how to encode variable assignments directly into the colored factor graph, allowing us to leverage graph isomorphism tools to compute canonical representatives and orbit sizes for assignments to variables in factor graphs. This colored assignment encoding is our key technical contribution, and forms a foundation for our exact and approximate inference algorithms. Then, we will give a breadth-first search procedure for generating all canonical representatives of a colored factor graph.

4.2.1 Encoding Assignments

Our objective in this section is to leverage graph isomorphism tools to compute the key quantities necessary for applying the procedure described in Theorem 4.1 to factor graphs. Let $\mathtt{G}$ be the induced colored graph of $\mathcal{F}$. As terminology, an element $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$ is an assignment to variables $\mathbf{X}$. We will use graph isomorphism tools to construct (1) a canonization function for variable assignments, $\sigma : \mathbb{B}^{\mathbf{X}} \to \mathbb{B}^{\mathbf{X}}/\mathbb{A}(\mathtt{G})$; and (2) the size of the orbit of $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$ under $\mathbb{A}(\mathtt{G})$. To do this, we encode assignments directly into the colored factor graph, which to our knowledge is a novel construction in this context:

Definition 4.2. Let $\mathcal{F} = (\mathbf{X}, F)$ be a factor graph, let $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$, and let $\mathtt{G} = (V, E, C)$ be the colored graph induced by $\mathcal{F}$. Then the *assignment-encoded colored graph*, denoted $\mathtt{G}(\mathcal{F}, \mathbf{x})$, is the colored graph that colors the variable nodes that are true and false in $\mathbf{x}$ with distinct colors in $\mathtt{G}$.

An example is shown in Figure 2, which shows an encoding of the assignment $000\ 111$. The assignment $000\ 111$ is isomorphic to the assignment $111\ 000$ under the action of $\mathbb{A}(\mathtt{G})$, specifically flipping holes. Then, assignments that are in the same orbit under $\mathbb{A}(\mathtt{G})$ have isomorphic colored graph encodings:

Theorem 4.2. *Let $\mathcal{F} = (\mathbf{X}, F)$ be a factor graph, $\mathtt{G}$ be its colored graph encoding, and $\mathbf{x}, \mathbf{x}' \in \mathbb{B}^{\mathbf{X}}$. Then, $\mathbf{x} \sim \mathbf{x}'$ under the action of $\mathbb{A}(\mathtt{G})$ iff $\mathtt{G}(\mathcal{F}, \mathbf{x}) \cong \mathtt{G}(\mathcal{F}, \mathbf{x}')$.*

Proof. See Appendix B.1. □

Canonization Our goal now is to use graph isomorphism tools to construct a canonization function for variable assignments. In particular, it maps all isomorphic assignments to exactly one member of their orbit. We will rely on *colored graph canonization*, a well-studied problem in graph theory for which there exist many implementations [Mckay and Piperno, 2014]:

Definition 4.3. Let $\mathtt{G} = (V, E, C)$ be a colored graph. Then a *colored graph canonization* is a canonization function $\sigma : V \to V/\mathbb{A}(\mathtt{G})$.

A colored graph canonization function applied to Figure 2 will select exactly one color-isomorphic vertex configuration as the canonical one, for example putting all pigeons in hole A. Then, the canonization of the assignment-encoded colored graph is a canonization of variable assignments:

Definition 4.4. Let $\mathcal{F} = (\mathbf{X}, F)$ and $\mathbf{x} = \{(x, v)\}$ be a variable assignment, where $x \in \mathbf{X}$ and $v \in \mathbb{B}$. Let $\sigma_{\mathtt{G}(\mathcal{F},\mathbf{x})}$ be a canonization of $\mathtt{G}(\mathcal{F}, \mathbf{x})$. Then, let $\sigma' : \mathbb{B}^{\mathbf{X}} \to \mathbb{B}^{\mathbf{X}}$ be defined $\sigma'(\mathbf{x}) = \{(\sigma_{\mathtt{G}(\mathcal{F},\mathbf{x})}(x), v) \mid (x, v) \in \mathbf{x}\}$. Then σ' is called the *induced variable canonization* of $\mathbb{B}^{\mathbf{X}}$.

Intuitively, an induced variable canonization computes the canonization of the assignment-encoded colored graph, and then applies that canonization function to variables. Then,

Proposition 4.1. *For a factor graph $\mathcal{F}$ with colored graph $\mathtt{G}$, the induced variable canonization is a canonization function $\mathbb{B}^{\mathbf{X}} \to \mathbb{B}^{\mathbf{X}}/\mathbb{A}(\mathtt{G})$.*

Computing the size of an orbit Theorem 4.1 requires efficiently computing the size of the orbit of an assignment. To accomplish this, we will apply the orbit-stabilizer theorem in a manner similar to Niepert [2013]. The size of a stabilizer is related to the size of an orbit with the following well-known theorem:

Theorem 4.3 (Orbit-stabilizer)**.** *Let $\mathcal{G}$ act on Ω. Then for any $x \in \Omega$, $|\mathcal{G}| = |\mathrm{Stab}(x)| \times |\mathrm{Orb}(x)|$.*

Thus, to compute orbit size of assignments $\mathbf{x}$, we will compute (1) the order of the $\mathrm{Stab}(\mathbf{x})$ under $\mathbb{A}(\mathtt{G})$; and (2) the order of $\mathbb{A}(\mathtt{G})$. Now we can again use graph isomorphism tools. The stabilizer of $\mathbf{x}$ corresponds with the automorphism group of the colored graph encoding of $\mathbf{x}$. To see this, observe that a permutation that relabels pigeons but leaves holes fixed is a stabilizer of the assignment in Figure 2; this permutation is also a member of the color-automorphism group of the graph. Formally:

Theorem 4.4. *Let $\mathcal{F} = (\mathbf{X}, F)$ be a factor graph with colored graph encoding $\mathtt{G}$. Then for any $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$, $\mathrm{Stab}_{\mathbb{A}(\mathtt{G})}(\mathbf{x}) = \mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}))$.*

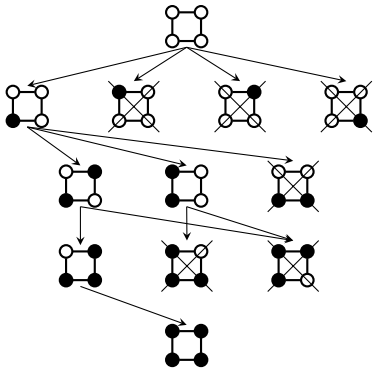


Figure 3: Example breadth-first search tree, read top-down. White nodes encode false assignments, and black nodes encode true assignments.

The proof can be found in Appendix B.2. Thus we have reduced computing orbit sizes to computing group orders, which can be computed efficiently using computational group theory tools such as `GAP` [GAP, Seress, 2003]. Thus if we can exhaustively generate canonical representatives, then we can perform lifted exact inference. The next section shows how to do this.

4.2.2 Generating All Canonical Representatives

Our algorithm for generating canonical representatives is a simple breadth-first search that relies on assignment canonization. This procedure is a kind of *isomorph-free exhaustive generation*, and there exist more sophisticated procedures than the one we present here [McKay, 1998].

Let $\mathbf{x}$ be some variable assignment. Then, an *augmentation* of $\mathbf{x}$ is a copy of $\mathbf{x}$ with one variable that was previously false assigned to true. We denote the set of all augmentations as $\mathcal{A}(\mathbf{x})$. Our breadth-first search tree will be defined by a series of augmentations as follows:

1. Nodes of the search tree are assignments $\mathbf{x}$.
2. The root of the tree is the all false assignment.
3. Each level L of the search tree has exactly L true assignments to variables.
4. Nodes are expanded until level $|\mathbf{X}|$.
5. Before expanding a node, check if it is not isomorphic to one that has already been expanded by computing its canonical form.
6. Then, expand a node $\mathbf{x}$ by adding $\mathcal{A}(\mathbf{x})$ to the frontier.

An example of this breadth-first search procedure is visualized in Figure 3. The search is performed on a 4-variable factor graph that has one factor on each edge, and all factors are symmetric. The factors are elided in the figure for visual clarity. Each arrow represents an augmentation. Crossed out graphs are pruned due to being isomorphic with a previously expanded node.

Now we bound the number of required graph isomorphism calls for this search procedure:

Theorem 4.5. *For a factor graph $\mathcal{F} = (\mathbf{X}, F)$ with $|\mathbb{B}^{\mathbf{X}}/\mathbb{A}(\mathtt{G})|$ canonical representatives, the above breadth-first search requires at most $|\mathbf{X}| \times |\mathbb{B}^{\mathbf{X}}/\mathbb{A}(\mathtt{G})|$ calls to a graph isomorphism tool.*

Proof. There are at most $|\mathbb{B}^{\mathbf{X}}/\mathbb{A}(\mathtt{G})|$ expansions, and each expansion adds at most $|\mathbf{X}|$ nodes to the frontier. A canonical form must be computed for each node that is added to the frontier. □

Pruning expansions This expansion process can be further optimized by preemptively reducing the number of nodes that are added to the frontier in Step 6, using the following lemma:

Lemma 4.1 (Expansion Pruning)**.** *Let $\mathcal{F}$ be a factor graph, $\mathbf{x}$ be a variable assignment, and $\mathbf{x}_1, \mathbf{x}_2$ be augmentations of $\mathbf{x}$ that update variables x and y respectively. Then, $\mathbf{x}_1 \sim \mathbf{x}_2$ under $\mathbb{A}(\mathtt{G})$ if x and y are in the same variable orbit under $\mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}))$.*

The proof is in Appendix B.3. Using this lemma we can update Step 6 to only include a single element of each variable orbit of $\mathbf{X}$ under $\mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}))$.

4.3 EXACT LIFTED INFERENCE ALGORITHM

Now we combine the theory of the previous two sections to perform exact lifted inference on factor graphs. Algorithm 1 performs exact lifted inference via a breadth-first search over canonical assignments. Variable r holds a set of canonical representatives, q holds the frontier, p accumulates the unnormalized probability of the evidence, and Z accumulates the normalizing constant. A graph isomorphism tool is used to compute σ on Line 5. Each time the algorithm finds a new representative, it computes the size of the orbit using the orbit stabilizer theorem on Line 9; `GAP` is used to compute the order of these permutation groups. Lemma 4.1 is used on Line 13 to avoid adding augmentations to the frontier that are known a-priori to be isomorphic to prior ones. This algorithm can be easily modified to produce the MPE by simply returning the canonical representative from r with the highest probability.

Experimental Evaluation To validate our method we implemented Algorithm 1 using the `Sage` math library, which wraps `GAP` and a graph isomorphism tool [The Sage Developers, 2018].[3] We compared our lifted inference procedure against `Ace`, an exact inference tool for

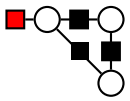
[3]The source code for our exact and approximate inference algorithms can be found at `https://github.com/SHoltzen/orbitgen`.

Algorithm 1: `ExactLiftedInference`$(\mathcal{F}, \mathbf{e})$

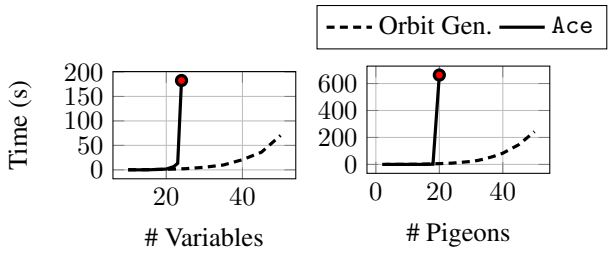
Data: A factor graph $\mathcal{F} = (\mathbf{X}, F)$ with color encoding $\mathtt{G}$; $\mathbb{A}(\mathtt{G})$-invariant evidence $\mathbf{e}$
Result: The probability of evidence $\Pr(\mathbf{e})$

$r \leftarrow$ empty set, $p \leftarrow 0$, $Z \leftarrow 0$;
$q \leftarrow$ queue containing the all-false assignment;
while *q is not empty* **do**
  $\mathbf{x} \leftarrow q$.pop();
  $\mathtt{Canon} \leftarrow \sigma(\mathtt{G}(\mathcal{F}, \mathbf{x}))$; // Invoke graph iso. tool
  if $\mathtt{Canon} \in r$ **then**
    continue;
  Insert $\mathtt{Canon}$ into r;
  $|\mathrm{Orb}(\mathbf{x})| \leftarrow |\mathbb{A}(\mathtt{G})|/|\mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}))|$; // Invoke GAP
  if $\mathbf{e}(\mathbf{x}) = \mathtt{T}$ **then**
    $p \leftarrow p + |\mathrm{Orb}(\mathbf{x})| \times F(\mathbf{x})$;
  $Z \leftarrow Z + |\mathrm{Orb}(\mathbf{x})| \times F(\mathbf{x})$;
  for *o from each variable orbit of* $\mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}))$ **do**
    if *o is a false variable* **then**
      $\mathbf{x}' \leftarrow \mathbf{x}$ with o true;
      Append $\mathbf{x}'$ to q;
return p/Z

discrete Bayesian networks that is unaware of the symmetry of the model [Chavira and Darwiche, 2005]. Figure 4 shows experimental results for performing exact lifted inference on two families of factor graphs. The first is a class of pairwise factor graphs that have an identical symmetric potential between all nodes, with one factor (in red) designated as an evidence factor:



We also evaluated our method on the pigeonhole problem from Section 2 with two holes and increasing number of pigeons. In both experiments, the number of orbits grows linearly, even though there is little independence. Thus, `Ace` scales exponentially, since the treewidth grows quickly, while our method scales subexponentially. To our knowledge, this is the first example of performing exact inference on this family of models.



(a) Inference for pairwise factor graph.

(b) Inference for 2-hole pigeonhole problem.

Figure 4: Evaluation of Algorithm 1. A red circle indicates that `Ace` ran out of memory at that time.

5 ORBIT-JUMP MCMC

In this section we introduce *orbit-jump MCMC*, an MCMC algorithm that mixes quickly when the distribution has few orbits, at the cost of requiring multiple graph isomorphism calls for each transition. The algorithm is summarized in Algorithm 2. Orbit-jump MCMC is an alternative to Lifted MCMC [Niepert, 2012, 2013] that generates provably high-quality samples at the expense of more costly transitions. Lifted MCMC exploits symmetric structure to quickly transition *within* orbits. Lifted MCMC is efficient to implement: it requires only a single call to a graph isomorphism tool. However, lifted MCMC relies on Gibbs sampling to jump *between orbits*, and therefore has no guarantees about its mixing time for distributions with few orbits. Orbit-jump MCMC is a Metropolis-Hastings MCMC algorithm that uses the following distribution as its proposal:

Definition 5.1. Let $\mathcal{G}$ act on Ω. Then for $x \in \Omega$, the *uniform orbit distribution* is:

$$\Pr_{\Omega/\mathcal{G}}(x) \triangleq \frac{1}{|\Omega/\mathcal{G}| \times |\mathrm{Orb}(x)|} \tag{5}$$

This is the probability of uniformly choosing an orbit $o \in \Omega/\mathcal{G}$, and then sampling uniformly from $\sigma^{-1}(o)$.

The *orbit-jump MCMC chain* for a $\mathcal{G}$-invariant distribution $\Pr$ is defined as follows, initialized to $x \in \Omega$:

1. Sample $x' \sim \Pr_{\Omega/\mathcal{G}}$;
2. Accept x' with probability $\min\left(1, \frac{\Pr(x') \times |\mathrm{Orb}(x')|}{\Pr(x) \times |\mathrm{Orb}(x)|}\right)$

This Markov chain has $\Pr$ as its stationary distribution. Orbit-jump MCMC has a high probability of proposing transitions *between* orbits, which is an alternative to the within-orbit exploration of lifted MCMC.[4]

Next we will describe how to sample from $\Pr_{\Omega/\mathcal{G}}$ using an MCMC method known as the *Burnside process*. Then, we will discuss the mixing time of this proposal, and prove that it mixes in the number of orbits of the distribution.

[4] This proposal is independent of the previous state, a scheme that is sometimes called *Metropolized independent sampling* (MIS) [Liu, 1996]. Importance sampling is an alternative to MIS. We use MIS rather than importance sampling in order to make the connection with lifted MCMC more explicit.

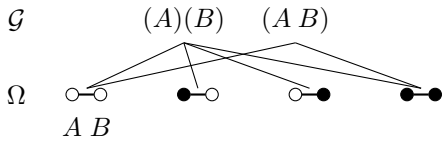


Figure 5: Illustration of the Burnside process on a colored graph with two nodes and two colors.

5.1 SAMPLING FROM $\Pr_{\Omega/\mathcal{G}}$

Jerrum [1993] gave an MCMC technique known as the *Burnside process* for drawing samples from $\Pr_{\Omega/\mathcal{G}}$. The Burnside process is a Markov Chain Monte Carlo method defined as follows, beginning from some $x \in \Omega$:

1. Sample $g \sim \mathrm{Stab}(x)$ uniformly;
2. Sample $x \sim \mathrm{Fix}(g)$ uniformly, where $\mathrm{Fix}(g) = \{x \in \Omega \mid g \cdot x = x\}$. We call elements of $\mathrm{Fix}(g)$ *fixers*.

Theorem 5.1 (Jerrum [1993])**.** *The stationary distribution of the Burnside process is equal to* $\Pr_{\Omega/\mathcal{G}}$.

This process can be visualized as a random walk on a bipartite graph. One set of nodes are elements of Ω, and the other set are elements of $\mathcal{G}$. There is an edge between $x \in \Omega$ and $g \in \mathcal{G}$ iff $g \cdot x = x$.

An example of this bipartite graph is shown in Figure 5. The set Ω is the set of 2-node colored graphs, and the group $\mathcal{G} = S_2$ permutes the vertices of the graph. The identity element $(A)(B)$ stabilizes all elements of Ω, and so has an edge to every element in x; $(A\ B)$ only stabilizes graphs whose vertices have the same color.

Jerrum [1993] proved that the Burnside process mixes rapidly for several important groups, but it does not always mix quickly [Goldberg and Jerrum, 2002]. In such cases, it is important to draw sufficient samples from the Burnside process in order to guarantee that the orbit-jump proposal is unbiased. Next we will describe how to implement the Burnside process on factor graphs using the machinery from Section 4.2.1.

5.1.1 Burnside Process on Factor Graphs

For $\mathcal{G}$ acting on a set of variables $\mathbf{X}$, the Burnside process requires the ability to (1) draw samples uniformly from the stabilizer subgroup of an assignment to variables, and (2) sample a random fixer for any group element in $\mathcal{G}$. Here we describe how to perform these two computations for a colored factor graph $\mathcal{F} = (\mathbf{X}, F)$.[5] This procedure is summarized in lines 3–7 in Algorithm 2.

[5]This process is conceptually similar to the procedure for randomly sampling orbits in the Pólya-theory setting described by Goldberg [2001], but this is the first time that this procedure is applied directly to factor graphs

Algorithm 2: A step of Orbit-jump MCMC

Data: A factor graph $\mathcal{F} = (\mathbf{X}, F)$, a point $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$, number of Burnside process steps k

1 $\mathbf{x}' \leftarrow \mathbf{x}$;
2 **for** $i \in \{1, 2, \cdots, k\}$ **do**
3 $\quad \mathcal{G}_{\mathrm{Stab}} \leftarrow \mathbb{A}(\mathtt{G}(\mathcal{F}, \mathbf{x}'))$; // Invoke graph iso. tool
4 $\quad$ Sample $s \sim \mathcal{G}_{\mathrm{Stab}}$ using product replacement;
5 $\quad$ **for** *Each variable cycle* c *of* s **do**
6 $\quad\quad v \sim$ Bernoulli$(1/2)$;
7 $\quad\quad$ Assign all variables c in $\mathbf{x}'$ to v;
8 Accept $\mathbf{x}'$ with probability $\min\left(1, \frac{F(\mathbf{x}') \times |\mathrm{Orb}(\mathbf{x}')|}{F(\mathbf{x}) \times |\mathrm{Orb}(\mathbf{x})|}\right)$

Stabilizer Sampling Section 4.2.1 showed how to compute the stabilizer group of $\mathbf{x} \in \mathbb{B}^{\mathbf{X}}$ using graph isomorphism tools. To sample uniformly from this stabilizer group, we rely on the *product replacement algorithm*, which is an efficient procedure for uniformly sampling group elements [Pak, 2000].

Fixer Sampling Let $g \in \mathcal{G}$ be a permutation that acts on the vertices of a colored factor graph. Then we uniformly sample an assignment-encoded colored factor graph that is fixed by g in the following way. First, decompose g into a product of disjoint cycles. Then, for each cycle that contains variable nodes, choose a truth assignment uniformly randomly, and then color the vertices in that cycle with that color. This colored graph is fixed by g and is uniformly random by the independence of coloring each cycle and the fact that all colorings fixed by g can be obtained in this manner.

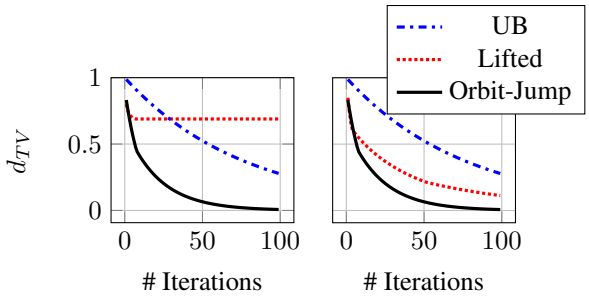
5.2 MIXING TIME OF ORBIT-JUMP MCMC

The *total variation distance* between two discrete probability measures μ and ν on Ω, denoted $d_{TV}(\mu, \nu)$, is:

$$d_{TV}(\mu, \nu) = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)|. \tag{6}$$

The *mixing time* of a Markov chain is the minimum number of iterations that the chain must be run starting in any state until the total variation distance between the chain and its stationary distribution is less than some parameter $\varepsilon > 0$. The mixing time of orbit-jump MCMC can be bounded in terms of the number of orbits, which is a property not enjoyed by lifted MCMC:

Theorem 5.2. *Let* $\Pr$ *be a* $\mathcal{G}$*-invariant distribution on* Ω *and let* P *be the transition matrix of orbit-jump MCMC. Then, for any* $x \in \Omega$, $d_{TV}(P^t x, \Pr) \leq \left(\frac{|\Omega/\mathcal{G}|-1}{|\Omega/\mathcal{G}|}\right)^t$. *It follows that for any* $\varepsilon > 0$, $d_{TV}(P^t x, \Pr) \leq \varepsilon$ *if* $t \geq \log(\varepsilon^{-1}) \times |\Omega/\mathcal{G}|$.



(a) Hard pigeonhole. (b) Quantum pigeonhole.

Figure 6: Total variation distance between Markov chains and their stationary distributions for a pigeonhole problem with 5 pigeons and 2 holes. "Lifted" is lifted MCMC [Niepert, 2012] and "UB" is the upper bound predicted by Theorem 5.2.

For a detailed proof see Appendix B.4. Note that the bound on this mixing time does not take into account the cost of drawing samples from $\Pr_{\Omega/\mathcal{G}}$, which involves multiple graph isomorphism calls.

Pigeonhole case study We implemented the orbit-jump MCMC procedure on factor graphs using `Sage`. In order to evaluate the performance of orbit-jump MCMC, we will compare the total variation distance of various MCMC procedures. We experimentally compare the mixing time of lifted MCMC [Niepert, 2012, 2013] and our orbit-jump MCMC in Figure 6, which computes the total variation distance of these two MCMC methods from their stationary distribution as a function of the number of iterations on two versions of the pigeonhole problem.[6] The first version in Figure 6a is the motivating example with hard constraints from Section 2. The second version in Figure 6b shows a *"quantum" pigeonhole problem*, where the constraint in Equation 1 is relaxed so that pigeons are allowed to be placed into multiple holes.

Lifted MCMC fails to converge in Figure 6a because it cannot transition due to the hard constraint from Equation 1; this illustrates that lifted MCMC can fail even for distributions with few orbits. In addition to comparing against lifted MCMC, we also compare the theoretical upper bound from Theorem 5.2 against the two mixing times. This upper bound only depends on the number of orbits, and does not depend on the parameterization of the distribution.[7] Orbit-jump MCMC converges to the true distribution in both cases faster than lifted MCMC, and the upper bound ensures that orbit-jump MCMC cannot get stuck in low-probability orbits. Note however that lifted MCMC transitions are less expensive to compute than orbit-jump MCMC transitions. We hope to explore this practical tradeoff between sample quality and the cost of drawing a sample in future work.

[6] In these experiments, for each step of orbit-jump MCMC, we use 7 steps of the Burnside process.

[7] For this example, there are 78 orbits.

6 RELATED WORK

Lifted inference Existing exact lifted inference algorithms apply to relational models [Getoor and Taskar, 2007]. The tractability of exact lifted inference was studied by Niepert and Van den Broeck [2014], but their approach cannot be directly applied to factor graphs. Approximate lifted inference can be applied to factor graphs, but existing approaches do not provably mix quickly in the number of orbits [Niepert, 2012, 2013, Bui et al., 2013, Van den Broeck and Niepert, 2015, Madan et al., 2018, Kersting et al., 2009, Gogate et al., 2012].

Symmetry in constraint satisfaction and logic Some techniques for satisfiability and constraint satisfaction also exploit symmetry. The goal in that context is to quickly select one of many symmetric candidate solutions, so a key difference is that in our setting we must exhaustively explore the search space. Sabharwal [2005] augments a SAT-solver with symmetry-aware branching capabilities. Symmetry has also been exploited in integer-linear programming [Margot, 2010, Ostrowski et al., 2007, Margot, 2003].

7 CONCLUSION & FUTURE WORK

In this paper we provided the first exact and approximate lifted inference algorithms for factor graphs that provably scale in the number of orbits. However, our methods are limited: there are tractable highly symmetric distributions that still have too many orbits for our methods to be effective. Existing lifted inference algorithms utilize independence to extract highly symmetric sub-problems, which is an avenue that we can see for integrating independence into this current approach. A further limitation of our approach is that we exploit only symmetries on variables; additional forms of symmetries, such as block symmetries, are beyond the scope of our current algorithms [Madan et al., 2018].

ACKNOWLEDGMENTS

This work is partially supported by NSF grants #IIS-1657613, #IIS-1633857, #CCF-1837129, DARPA XAI grant #N66001-17-2-4032, NEC Research, a gift from Intel, and a gift from Facebook Research. The authors would like to thank Tal Friedman, Pasha Khosravi, Jon Aytac, Philip Johnson-Freyd, Mathias Niepert, and Anton Lykov for helpful discussions and feedback on drafts.

References

M. Artin. *Algebra*. Birkhäuser, 1998.

B. Benhamou and L. Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1):89–102, Feb 1994.

H. H. Bui, T. N. Huynh, and S. Riedel. Automorphism groups of graphical models and lifted variational inference. In *UAI*, pages 132–141, 2013.

M. Chavira and A. Darwiche. Compiling bayesian networks with local structure. In *IJCAI*, pages 1306–1312, 2005.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

R. De Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *IJCAI*, pages 1319–1325, 2005.

P. Diaconis and D. Freedman. De Finetti's generalizations of exchangeability. In R. C. Jeffrey, editor, *Studies in Inductive Logic and Probability*, pages 2–233. Berkeley: University of California Press, 1980.

GAP. *GAP – Groups, Algorithms, and Programming, Version 4.10.0*. The GAP Group, 2018.

L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

V. Gogate and P. Domingos. Probabilistic theorem proving. In *UAI*, pages 256–265, 2011.

V. Gogate, A. K. Jha, and D. Venugopal. Advances in lifted importance sampling. In *AAAI*, 2012.

L. Goldberg. Computation in permutation groups: Counting and randomly sampling orbits. *Surveys in Combinatorics*, pages 109–143, 2001.

L. A. Goldberg and M. Jerrum. The Burnside process converges slowly. *Combinatorics, Probability and Computing*, 11(1):21–34, 2002.

M. Jerrum. Uniform sampling modulo a group of symmetries using markov chain simulation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 37–47, 1993.

K. Kersting. Lifted probabilistic inference. In *ECAI*, pages 33–38, 2012.

K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *UAI*, pages 277–284, 2009.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

D. A. Levin and Y. Peres. *Markov chains and mixing times*. American Mathematical Society, 2017.

J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, 1996.

G. Madan, A. Anand, Mausam, and P. Singla. Block-value symmetries in probabilistic graphical models. In *UAI*, pages 886–895, 2018.

F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming*, 98:3–21, 2003.

F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming*, 2010.

B. D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26(2):306 – 324, 1998.

B. D. Mckay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

M. Niepert. Markov chains on orbits of permutation groups. In *UAI*, pages 624–633, 2012.

M. Niepert. Symmetry-aware marginal density estimation. *AAAI*, 2013.

M. Niepert and G. Van den Broeck. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *AAAI*, 2014.

J. Ostrowski, J. T. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. In *IPCO*, 2007.

I. Pak. What do we know about the product replacement algorithm? In *Groups and Computation III*, pages 301–347, 2000.

D. Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.

R. Raz. Resolution lower bounds for the weak pigeonhole principle. *J. ACM*, 51(2):115–138, 2004.

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273 – 302, 1996.

A. Sabharwal. Symchaff: A structure-aware satisfiability solver. In *AAAI*, volume 5, pages 467–474, 2005.

A. Seress. *Permutation Group Algorithms*. Cambridge Tracts in Mathematics. Cambridge University Press, 2003.

The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.5.0)*, 2018. `https://www.sagemath.org`.

G. Van den Broeck. *Lifted inference and learning in statistical relational models*. PhD thesis, 2013.

G. Van den Broeck and M. Niepert. Lifted probabilistic inference for asymmetric graphical models. In *AAAI*, 2015.