The AAAI-17 Workshop on
Symbolic Inference and Optimization
WS-17-14

Conditional Term Equivalent Symmetry Breaking for SAT

Tim Kopp
University of Rochester
Rochester, NY

Parag Singla
Indian Institute of Technology
Hauz Khas, New Delhi

Henry Kautz
University of Rochester
Rochester, NY

Abstract

Symmetry-breaking is a technique for efficiently solving SAT instances that contain high degrees of symmetry among the variables of the instance. When satisfiability problems are represented as a relational schema, symmetries between objects in the domain can be detected directly from evidence, that is, variables known to have a particular setting prior to solving. These symmetries between domain objects are called term symmetries. In this work, we present two novel extensions to the technique of term equivalent symmetry breaking which allow the detection and exploitation of conditional or hidden symmetries, those relationships between domain objects that are obscured until the instance is partially solved. We give promising preliminary experimental results for this technique, and discuss how the techniques could be extended for use in probabilistic domains.

Introduction

Symmetry-breaking is an approach to speeding up satisfiability testing by adding constraints, called *symmetry-breaking predicates* (SBPs), to a theory (Crawford et al. 1996; Aloul, Markov, and Sakallah 2003; Katebi, Sakallah, and Markov 2010). Symmetries in the theory define a partitioning over the space of truth assignments, where the assignments in a partition either all satisfy or all fail to satisfy the theory. The added SBPs rule out some but not all of the truth assignments in the partitions, thus reducing the size of the search space while preserving satisfiability.

When a SAT instance is specified as a *relational theory*, a set of predicate logic formulas relating elements in a finite domain, symmetries between the domain objects (terms) can be detected and exploited in much the same way (Kopp, Singla, and Kautz 2015). These symmetries are called *term equivalent symmetries*, and represent a subset of symmetries that can be efficiently detected and broken. While these two approaches to symmetry-breaking differ in the level at which symmetries are detected, they are similar insofar as they are both analyze the given instance as is, no variable assignments are computed during the preprocessing procedure. This feature of term symmetry breaking prevents the system from detecting hidden symmetries, those symmetries which only hold when a subset of variables are assigned in a particular way.

We introduce two new extensions to term equivalent symmetry breaking which perform a dynamic analysis of the SAT instance, and therefore exploit hidden symmetries. The first extends the preprocessing procedure to perform analyses on versions of the instance with one or more variables assigned. The SBPs added to the formula are called *conditional* SBPs, since the symmetries they break depend upon the variables that have been assigned. The second extension detects symmetries online, rather than as a preprocessing procedure, and adds conditional SBPs to the formula in much the same way as conflict clauses are learned. We provide some preliminary experimental results that suggest that it may be worthwhile to exploit these symmetries in SAT solving.

Related Work

Our work has connections to research in in both the machine learning and constraint-satisfaction research communities. Developments include lifted versions of variable elimination (Poole 2003; de Salvo Braz, Amir, and Roth 2005), belief propagation (Singla and Domingos 2008; Singla, Nath, and Domingos 2014), and DPLL (Gogate and Domingos 2011). The approach of defining symmetries using group theory and detecting them by graph isomorphism is shared by Bui *et al.*'s work on lifted variational inference (Bui, Huynh, and Riedel 2013). Niepert gives a group-theoretic formalism of symmetries at the level of constants (Niepert 2012; 2013), applying them to MCMC methods. Kopp et. al. define the class of symmetries we exploit in this work (Kopp, Singla, and Kautz 2015). Bui notes that symmetry groups can be defined on the basis of *unobserved* constants in the domain, while the symmetries we exploit can be explicitly found in the evidence. Two lines of work in SRL make use of problem transformations. First-order knowledge compilation (Van den Broeck 2013; den Broeck, Meert, and Darwiche 2014) transforms a relational problem into a form for which MPE, marginal, and MAP inference is tractable. Recent work on MAP inference in Markov Logic has identified special cases where a relational formula can be transformed by replacing a quantified formula with a *single* grounding of the formula (Mittal et al. 2014).

The literature surrounding the use of symmetries in constraint satisfaction and model counting is quite extensive. Here we give just a few examples of key developments in the field. Symmetry detection has been based either on graph isomorphism on propositional theories as in the original work by by Crawford *et. al* (Crawford et al. 1996); by interchangeability of row and/or columns in CSPs specified in matrix form (Meseguer and Torras 2001); by checking for other special cases of geometric symmetries (Sellmann and Hentenryck 2005), or by determining that domain elements for a variable are exchangeable (Audemard, Benhamou, and Henocque 2006). Researchers have suggested symmetry-aware modifications to backtracking CSP solvers for variable selection, branch pruning, and no-good learning (Meseguer and Torras 2001; Flener et al. 2009). A recent survey of symmetry breaking for CSP (Walsh 2012) described alternatives to the lex-leader formulation of SBPs, including one based on Gray codes. The notion of a hidden or conditional symmetry is closely related to the contextual symmetries exploited in (Anand et al. 2016).

Symmetries over terms

In this section we provide the background necessary to understand the types of symmetries we will exploit in the subsequent sections. Symmetry-breaking for satisfiability testing, introduced by Crawford et. al.(Crawford et al. 1996), is based on concepts from group theory. A *permutation* θ is a mapping from a set L to itself. A permutation group is a set of permutations that is closed under composition and contains the identity and a unique inverse for every element. A literal is an atom or its negation. A clause is a disjunction over literals. A CNF theory $\mathcal{T}$ is a set (conjunction) of clauses. Let L be the set of literals of $\mathcal{T}$. We consider only permutations that respect negation, that is $\theta(\neg l) = \neg\theta(l)$ ($l \in L$). The *action* of a permutation on a theory, written $\theta(\mathcal{T})$, is the CNF formula created by applying θ to each literal in $\mathcal{T}$. We say θ is a *symmetry* of $\mathcal{T}$ if it results in the same theory i.e. $\theta(\mathcal{T}) = \mathcal{T}$. For example, the rotation permutation $\theta = \{x_1 \to x_2, x_2 \to x_3, x_3 \to x_1\}$ is a symmetry of the following theory:

$$\neg x_1 \vee x_3$$
$$\neg x_2 \vee x_1$$
$$\neg x_3 \vee x_2$$

A model M is a truth assignment to the atoms of a theory. The action of θ on M, written $\theta(M)$, is the model where $\theta(M)(\mathcal{T}) = M(\theta(\mathcal{T}))$. The key property of θ being a symmetry of $\mathcal{T}$ is that $M \models \mathcal{T}$ iff $\theta(M) \models \mathcal{T}$. The *orbit* of a model M under a symmetry group Θ is the set of models that can be obtained by applying any of the symmetries $\theta \in \theta$ to M. A symmetry group divides the space of models into disjoint sets, where the models in an orbit either all satisfy or all do not satisfy the theory.

In this work, we will consider symmetries over the terms (constants) of a relational theory. A *relational theory* is a tuple $\mathcal{T} = (F, \mathcal{E})$, where F is a set of predicate logic formulas and $\mathcal{E}$ is a set of evidence. We restrict the formulas in F to be built from predicates, variables, quantifiers, and logical connectives, but no constants or function symbols. $\mathcal{E}$ is a set of ground literals; that is, literals built from predicates and constant symbols. Universal and existential quantification is over the set of the theory's constants $\mathcal{D}$ (i.e. the constants that appear in its evidence). Quantification occurs over a finite domain, therefore universal quantifiers ground to a conjunction of clauses, and existential quantifiers ground to a disjunction of literals.

In (Kopp, Singla, and Kautz 2015), two classes of symmetries over terms of relational theories were formalized, and methods to exploit the symmetries were given. Both classes of symmetries can be detected over the evidence of a theory. The first was the set of *term symmetries*. A term symmetry is a permutation θ of the terms in the theory such that $\theta(\mathcal{T}) = \mathcal{T}$. Note that a permutation of terms over a ground theory induces a permutation of atoms of the theory. That is, given a literal $l = P(x_1, \ldots, x_k)$ and a permutation θ, the action of the permutation on the literal $\theta(l) = P(\theta(x_1), \ldots, \theta(x_k))$. A term symmetry can be broken with a *symmetry breaking predicate*, a set of clauses added to the theory which reduces the search space while preserving satisfiability. The SBP to break a single term symmetry is:

$$SBP(\theta) = \bigwedge_{1 \le i \le n} \Big(\bigwedge_{1 \le j < i} G_j \Leftrightarrow \theta(G_j) \Big) \Rightarrow G_i \Rightarrow \theta(G_i) \tag{1}$$

where the G_i are the ground atoms of the theory.

The second class of symmetries over terms is a special case of the former called *term equivalent symmetries*. A term equivalent symmetry is a partitioning $\mathcal{Z} = \{Z_i\}$ of the terms in the theory such that if two terms C_1 and C_2 appear in the same term equivalent class Z_i, they can be permuted without changing the theory. In other words, they define equivalence classes of terms. Place an ordering on the terms of the theory, so that the members of the equivalence classes can be sorted. In order to break all of the symmetries that respect $\mathcal{Z}$, we need only explicitly break the symmetries that swap adjacent elements in term equivalent sets with respect to this ordering. Thus, the term equivalent SBP below, which explicitly breaks only a linear number of symmetries, implicitly breaks all symmetries that respect $\mathcal{Z}$. Let $\theta^i_{j,k}$ be the symmetry that swaps the jth and kth element in Z_i.

$$TESBP(Z) = \bigwedge_{i=1}^{|Z|} \bigwedge_{j=1}^{|Z_i|-1} SBP(\theta^i_{j,j+1}) \tag{2}$$

Consider a relational theory that models allocation of computational resources to distributed computing tasks that has types for CPUs in a cluster, cores on a CPU, and computational tasks, as well as predicates that describe properties of the computational resources and tasks. In this domain, all of the cores that belong to the same CPU belong to the same term equivalent symmetry group. Furthermore, if two CPUs have the same number and types of cores, then there are term symmetries that permute the CPUs while also permuting their respective cores.

In the next section, we formally define hidden or conditional symmetries, and give a detailed example of a domain with a high degree of hidden symmetry.

Symmetries over terms

In this section we provide the background necessary to understand the types of symmetries we will exploit in the subsequent sections. Symmetry-breaking for satisfiability testing, introduced by Crawford et. al.(Crawford et al. 1996), is based on concepts from group theory. A *permutation* θ is a mapping from a set L to itself. A permutation group is a set of permutations that is closed under composition and contains the identity and a unique inverse for every element. A literal is an atom or its negation. A clause is a disjunction over literals. A CNF theory $\mathcal{T}$ is a set (conjunction) of clauses. Let L be the set of literals of $\mathcal{T}$. We consider only permutations that respect negation, that is $\theta(\neg l) = \neg\theta(l)$ ($l \in L$). The *action* of a permutation on a theory, written $\theta(\mathcal{T})$, is the CNF formula created by applying θ to each literal in $\mathcal{T}$. We say θ is a *symmetry* of $\mathcal{T}$ if it results in the same theory i.e. $\theta(\mathcal{T}) = \mathcal{T}$. For example, the rotation permutation $\theta = \{x_1 \to x_2, x_2 \to x_3, x_3 \to x_1\}$ is a symmetry of the following theory:

$$\neg x_1 \vee x_3$$
$$\neg x_2 \vee x_1$$
$$\neg x_3 \vee x_2$$

A model M is a truth assignment to the atoms of a theory. The action of θ on M, written $\theta(M)$, is the model where $\theta(M)(\mathcal{T}) = M(\theta(\mathcal{T}))$. The key property of θ being a symmetry of $\mathcal{T}$ is that $M \models \mathcal{T}$ iff $\theta(M) \models \mathcal{T}$. The *orbit* of a model M under a symmetry group Θ is the set of models that can be obtained by applying any of the symmetries $\theta \in \theta$ to M. A symmetry group divides the space of models into disjoint sets, where the models in an orbit either all satisfy or all do not satisfy the theory.

In this work, we will consider symmetries over the terms (constants) of a relational theory. A *relational theory* is a tuple $\mathcal{T} = (F, \mathcal{E})$, where F is a set of predicate logic formulas and $\mathcal{E}$ is a set of evidence. We restrict the formulas in F to be built from predicates, variables, quantifiers, and logical connectives, but no constants or function symbols. $\mathcal{E}$ is a set of ground literals; that is, literals built from predicates and constant symbols. Universal and existential quantification is over the set of the theory's constants $\mathcal{D}$ (i.e. the constants that appear in its evidence). Quantification occurs over a finite domain, therefore universal quantifiers ground to a conjunction of clauses, and existential quantifiers ground to a disjunction of literals.

In (Kopp, Singla, and Kautz 2015), two classes of symmetries over terms of relational theories were formalized, and methods to exploit the symmetries were given. Both classes of symmetries can be detected over the evidence of a theory. The first was the set of *term symmetries*. A term symmetry is a permutation θ of the terms in the theory such that $\theta(\mathcal{T}) = \mathcal{T}$. Note that a permutation of terms over a ground theory induces a permutation of atoms of the theory. That is, given a literal $l = P(x_1, \ldots, x_k)$ and a permutation θ, the action of the permutation on the literal $\theta(l) = P(\theta(x_1), \ldots, \theta(x_k))$. A term symmetry can be broken with a *symmetry breaking predicate*, a set of clauses added to the theory which reduces the search space while preserving satisfiability. The SBP to break a single term symmetry is:

$$SBP(\theta) = \bigwedge_{1 \le i \le n} \Big(\bigwedge_{1 \le j < i} G_j \Leftrightarrow \theta(G_j) \Big) \Rightarrow G_i \Rightarrow \theta(G_i) \tag{3}$$

where the G_i are the ground atoms of the theory.

The second class of symmetries over terms is a special case of the former called *term equivalent symmetries*. A term equivalent symmetry is a partitioning $\mathcal{Z} = \{Z_i\}$ of the terms in the theory such that if two terms C_1 and C_2 appear in the same term equivalent class Z_i, they can be permuted without changing the theory. In other words, they define equivalence classes of terms. Place an ordering on the terms of the theory, so that the members of the equivalence classes can be sorted. In order to break all of the symmetries that respect $\mathcal{Z}$, we need only explicitly break the symmetries that swap adjacent elements in term equivalent sets with respect to this ordering. Thus, the term equivalent SBP below, which explicitly breaks only a linear number of symmetries, implicitly breaks all symmetries that respect $\mathcal{Z}$. Let $\theta^i_{j,k}$ be the symmetry that swaps the jth and kth element in Z_i.

$$TESBP(Z) = \bigwedge_{i=1}^{|Z|} \bigwedge_{j=1}^{|Z_i|-1} SBP(\theta^i_{j,j+1}) \tag{4}$$

Consider a relational theory that models allocation of computational resources to distributed computing tasks that has types for CPUs in a cluster, cores on a CPU, and computational tasks, as well as predicates that describe properties of the computational resources and tasks. In this domain, all of the cores that belong to the same CPU belong to the same term equivalent symmetry group. Furthermore, if two CPUs have the same number and types of cores, then there are term symmetries that permute the CPUs while also permuting their respective cores.

In the next section, we formally define hidden or conditional symmetries, and give a detailed example of a domain with a high degree of hidden symmetry.

Conditional symmetries

The methods in the previous section analyze the formula as given, without assigning any variables, in order to detect symmetries. As a result, all of the symmetries that are detected are valid permutations for the formula irrespective of the (partial) assignment, and are therefore valid during the entirety of solving. However, there are *hidden* term equivalent symmetries that cannot be detected or broken directly by these methods. We formally define these symmetries below, and then demonstrate what they are by way of an example.

Definition 1. *A hidden or conditional term symmetry θ^L is a permutation of the constants of a theory $\mathcal{T}$ such that, for a set L of consistent literals, θ^L is a symmetry of all models of $\mathcal{T}$ that are consistent with the literals in L.*

Conditional term symmetries are related to the contextual symmetries exploited in (Anand et al. 2016). One can think of conditional term symmetries as the relational version of contextual symmetries. Whereas contextual symmetries operate on the level of logical atoms, conditional symmetries operate on the level of domain objects.

Consider a domain that expresses an unsatisfiable pigeonhole problem (PHP) with n pigeons and $n-1$ holes. There is an existential statement that states that every pigeon in the domain must be in a hole:

$$\forall p \exists h \quad In(p,h)$$

which grounds to n disjunctions each with $n-1$ disjuncts. There is also a constraint that states that for every pair of pigeons P_i and P_j, $i \neq j$, at least one of those pigeons is not in H, i.e. no two pigeons can be in the same hole:

$$\forall h, p_i, p_j, i \neq j, \quad \neg In(p_i,h) \vee \neg In(p_j,h)$$

This is a standard domain used to illustrate symmetries; all of the pigeon objects are in the same term equivalent class, all of the hole objects are in another, and these classes can be detected prior to solving and exploited with SBPs.

Now let us consider a slightly modified version of this problem which contains conditional symmetries. Let us call this domain the hidden pigeon hole problem (HPHP). We add another predicate, $Roost(P)$, that is true when pigeon P requires a hole in which to roost. We then modify the problem so that only those pigeons that roost must be in a hole:

$$\forall p \quad Roost(P) \Rightarrow \big(\exists h \quad In(P, H_i)\big)$$

Now we expand the domain so that there are n holes and $n+k$ pigeons. The predicate $Roost$ is partially observed, that is, there are some pigeons that are known to roost, some that are known not to roost, and some for which it is not known.

To humans, it is clear that if there are at least $n+1$ pigeons known to roost, then the instance is unsatisfiable, and otherwise it is satisfiable. The term equivalent symmetry detection procedure of (Kopp, Singla, and Kautz 2015) would not be able to exploit this knowledge. It would correctly group the pigeons into three classes, and therefore be able to recognize that is needs to solve pigeon hole problems with varying numbers of roosting pigeons, rather than solve pigeon hole problems for every combination of pigeon roosting patterns. This represents an exponential reduction in the number of individual pigeon hole problems to be solved. However, each individual pigeon hole problem would be solved somewhat naively.

When roosting patterns are assigned to pigeons, *new symmetries are revealed*. In particular, a pigeon who is assigned to roost is now in the same term equivalent class as all other roosting pigeons. If these symmetries could be detected and exploited, then it would be possible to solve each pigeon hole problem embedded in this modified problem as efficiently as current term equivalent symmetry breaking techniques solve the simpler version of the pigeon hole problem.

Conditional SBPs

We can potentially detect symmetries of this kind in a simple manner. Recall that symmetries are detected in evidence. We simply choose a set of literals $L = \{l_i\}$ to assert, add those literals to the evidence, and perform term symmetry analysis as normal. However, we cannot merely add SBPs to the formula as before. The detected symmetries are only valid when the asserted literals are valid. Therefore, we add a slightly modified SBP, which we call a *conditional symmetry breaking predicate*, which is valid only when the current assignment is consistent with the asserted literals:

$$CSBP(\theta^L) = \bigwedge_{1 \le i \le n} \left[\Big(\bigwedge_{l \in L} l \Big) \Rightarrow \left(\bigwedge_{1 \le j < i} G_j \Leftrightarrow \theta(G_j) \right) \Rightarrow G_i \Rightarrow \theta(G_i) \right] \tag{5}$$

In this formulation, the conjunction of the literals in L, called the *precondition*, must all be satisfied by the partial assignment before the constraint is enforced. The CSBP can be simplified where the precondition causes a conjunct to be trivial or redundant. For example, suppose $\neg G_1 \in L$. Then the first conjunct would be $\neg G_1 \Rightarrow G_1 \Rightarrow \theta(G_1)$, which is trivially satisfied.

Term equivalent symmetries are detected in a similar manner: add the set of literals to be asserted to the evidence, and run term equivalent symmetry analysis as normal. When adding TESBPs, use Equation 4, substituting calls to CSBP in Equation 5 for the calls to SBP in Equation 3. Note that this technique is a generalization of the work in (Kopp, Singla, and Kautz 2015). SBPs added using those techniques are equivalent to CSBPs with $L = \emptyset$.

In both cases, we can optimize by tracking the minimum set of literals that need to be asserted in order to create a symmetry. Suppose symmetry θ is detected when only the evidence, and no extra literals, are asserted. If we then assert some nonempty set of literals L and detect θ, we need not add $CSBP(\theta^L)$, as it is strictly weaker than an $CSBP(\theta^{\emptyset})$.

Conditional symmetry detection requires us to make assignments to (usually a small number of) the variables in the theory as part of preprocessing, allowing the detection of hidden symmetries at the cost of more expensive preprocessing. Choosing how many and which variables to assign is a topic for future exploration. An obvious heuristic for choosing variables to assign during processing is to choose those variables of partially-observed predicates, since hidden symmetries are revealed when those variables are assigned. However, if domains are large, the number of partial assignments that could be found with this heuristic may still be very large. In the next section, we give a further extension of conditional symmetry breaking that builds upon the work in this section and attempts to overcome this problem.

Online conditional symmetry detection

In this section, we give a method for detecting conditional term equivalent symmetries online, that is, during solving, rather than as a preprocessing step. Naively, this can be done in a straightforward manner: after each variable is assigned, perform symmetry analysis, and add CSBPs as appropriate. However, this naive approach is simply too expensive. Term equivalent symmetry detection takes time $O(nM \log M)$ where n is the number of constants and M is the size of the partial assignment. To perform this computation at each point in an exhaustive combinatorial search is not practical, particular in the lower parts of the search tree where M is larger. In order to make online conditional term equivalent symmetry detection more practical, we give a method to perform full symmetry analysis once, and update that analysis inexpensively during solving.

Recall from (Kopp, Singla, and Kautz 2015) that term equivalent symmetry analysis places two constants in the same term equivalent class if and only if they have the same context. A context is a set of partial atoms that represents the atoms in which a constant appears in the partial assignment. For example, if the partial assignment is $\{P(A), Q(A, B)\}$, then the context of A is $\{P(*), Q(*, B)\}$ and the context of B is $\{Q(A, *)\}$. We introduce a hash table H whose keys are contexts, and whose values are terms, and we keep a lookup table L which maps terms to contexts (stored, e.g. as a balanced tree). Suppose that H and L contain the information to represent the partial assignment $\mathcal{E}$ at a particular point in the search, including evidence known prior to the search. Note that if we are at the beginning of the search, $\mathcal{E}$ is merely the evidence.

We have H and L which represent $\mathcal{E}$, and we want to assign an atom A to continue the search. We iterate over the arguments of the atom, and for each term C that appears as an argument, we:

1. Look up $context(C)$ in L. $O(1)$.
2. Remove C from $H[context(C)]$. $O(1)$.
3. Insert the new context from A into $context(C)$ in L. $O(\log(|atoms|))$.
4. Insert the updated $context(C)$ from L into H, using C as the value. $O(|atoms|)$.

The complexity of step 2 assumes we store the values of H as a set data structure. If a simple list, then it's $O(|D|)$, and in practice likely much smaller. The complexity of step 3 is because it takes logarithmic time to insert an item into a balanced tree, and the tree is bounded in size by the number of atoms. Similarly for step 4, we must iterate over the context list to hash it.

It is not always the case that we assign truth values to atoms one after the other. The inference procedure is run to take advantage of pure literals and unit propagation. It may be the case that after assigning a single truth value, we have to update using a *set* of atoms. The simplest way to handle this is to perform the updates serially. However, we can get an efficiency gain by copying the old values in L, updating L incrementally for each new atom, and then performing the hash table updates only once per term. Since this is the most expensive step, it is good to perform it once per chosen assignment, rather than once per atom assigned or inferred.

One problem with this method as stated is that each update can potentially take very long. Early in the search space, the size of the context lists will be small, and it should not matter. However, later in the search, when many or most of the atoms are assigned, the context lists will be very large. One way to deal with this problem is to provide a depth cutoff. The data structures would simply stop updating after k atoms have been assigned in the search. It is expected that this method will be impractical without this modification. Another strategy that can be used complementarily with the first is to only perform the update periodically, every j atom assignments for example. This lets us perform symmetry detection at lower levels of the tree less expensively.

In general, it may be the case that not every atom in the partial assignment needs to appear on the LHS of the CSBP in order for it to be valid. Suppose the partial assignment is $\{P(A), P(B), \neg P(C)\}$, which induces the term equivalent partition $\{\{A, B\}, \{C\}, \{D, E, \dots\}\}$. It should be clear that only the literals $P(A)$ and $P(B)$ contribute to differentiating A and B from the other term equivalent sets. Therefore, $\neg P(C)$ need not be included in the precondition in the CSBP that breaks the symmetry between A and B. In general, we need only include those atoms that contain at least one term that appears in the term equivalent set that we are breaking. This optimization can be done online while generating the SBP, each time the procedure would add a literal to the LHS, it performs a check and does so only if the literal contains one of the necessary terms.

By virtue of their preconditions, the clauses learned through this procedure are valid at every point in the search space. Furthermore, if unneeded literals are pruned from the precondition, then the clauses are potentially useful at every point in the search space, since that subset of the partial assignment could be asserted in the same manner on a different subtree of the search space, or after a restart. However, just as it is typically impractical to retain learned conflict clauses for the entire duration of the search, it would be impractical to retain the CSBPs for the whole search. Any of the traditional strategies for managing deletion of learned clauses in traditional CDCL SAT solvers can be used to delete SBPs.

Experimental results

The extended preprocessing procedure for conditional term equivalent symmetry breaking was implemented as an extension to the work of (Kopp, Singla, and Kautz 2015). The system was evaluated on varying sizes of the HPHP domain outlined in above with evidence that forced unsatisfiability, i.e. with $n + 1$ pigeons p for which $Roost(p)$ appears in the evidence.

All SAT evaluation was performed with the latest version of Minisat (Eén and Sörensson 2003). The conditional term equivalent term symmetry preprocessing system (cond), was compared against a run of Minisat with no preprocessing, (vanilla), the term and term equivalent symmetry breaking systems from (Kopp, Singla, and Kautz 2015) (term and tequiv, respectively), and the Shatter (Aloul, Markov, and Sakallah 2003) SBP preprocessor (shatter), which works at the propositional rather than constant level.

Table 1 gives the results of running these systems on the HPHP domain for various sizes, where the size is the number of holes in the domain n, the number of pigeons in the domain is $\lceil 1.5n \rceil + 1$, and there are $n + 1$ pigeons P for which $Roost(P)$ appears in the evidence. The columns are given as $x + y$, where x is the runtime of the preprocessing procedure, and y is the runtime of Minisat on the formula with the SBPs. The `vanilla` column is omitted, as

Minisat with no preprocessing is unable to solve instances greater than size 10 within the 30-minute timeout. Shatter was unable to handle instances larger than size 40. All trials were given a 30-minute timeout. All runtimes are given in seconds.

Table 1: Conditional Symmetries on the HPHP Domain

Size	shatter	tequiv	term	cond
5	0.03 + 0.0	0.01 + 0.0	0.02 + 0.0	0.01 + 0.0
10	0.4 + 0.0	0.09 + 0.01	0.13 + 0.0	0.1 + 0.0
30	191 + 0.03	3.5 + 0.03	5.4 + 0.03	5.1 + 0.05
40	1026 + 0.05	9.8 + 0.07	16 + 0.07	14 + 0.13
60	N/A	45 + 0.24	79 + 0.3	66 + 0.5
70	N/A	79 + 0.9	139 + 0.8	113 + 0.7
80	N/A	129 + 1.4	231 + 1.9	170 + 1.0
90	N/A	207 + 5.7	372 + 6.8	247 + 1.6
100	N/A	310 + 12	563 + 10	374 + 2.0
110	N/A	440 + 10	811 + 12	540 + 6

We see that as the problem scales, the conditional symmetry breaking system performs strictly better than Shatter and the term symmetry breaking system. Furthermore, it generates SBPs that outperform those of the term equivalent symmetry breaking system in solving time, though the increased preprocessing time outweighs these benefits. This does, however, give empirical evidence that hidden symmetries exist and can be exploited in the manner described, which means that a well-engineered online conditional symmetry breaking system could potentially realize performance gains over other symmetry-breaking systems.

Future work

This is preliminary work shows that SAT solving performance can be improved by exploiting the class of symmetries defined. What remains to be shown is whether or not the performance gains in solving times can overcome the increased overhead required to exploit the symmetries.

Toward this end, there are two main directions in which to focus. The first is to study possible heuristics for determining which literals to assign in the preprocessing procedure, how many partial assignments on which the system should perform symmetry analysis, and to explore options for optimizing the performance of the preprocessing procedure on the whole. With better heuristics and more engineering work, it may be possible for a preprocessing system to exploit conditional symmetries in a manner which improves overall performance.

The second direction is to implement the online conditional symmetry breaking algorithm. The empirical results suggest that exploiting the class of conditional symmetries can increase solving performance, and the proposed online system has the potential to exploit these symmetries in a more efficient manner than the preprocessing system. In this work, as in the preprocessing system, there is work to be done on heuristics for variable choice, depth, and frequency.

While the empirical results show that conditional symmetries exist in theory, it is unknown to what extent they exist in real-world applications. Intuitively, they will exist in any situation in which some partially-observed property of a domain object determines how that object is matched to other objects. For example, partially-observed properties of computers in a networked cluster may influence how compute jobs are distributed across the nodes. Finding real-world applications that exhibit this class of symmetries in high degree would make the class more worthy of future study.

Finally, the scope of this work was limited to satisfiability problems. However, the principles should be readily applicable to maxSAT, the optimization version of satisfiability, with and without weights. The problem of MPE inference in statistical relation AI systems such as Markov logic networks (Richardson and Domingos 2006) reduces to weighted maxSAT, meaning these techniques could potentially be used with inference problems in those models.

Conclusion

In this work, we have formalized the notion of conditional term and term equivalent symmetries in the context of SAT problems specified by relational theories. We have given both a preprocessing and online method for exploiting these symmetries for satisfiability, and given empirical evidence that exploiting these symmetries can improve the runtime of SAT solving.

References

Aloul, F. A.; Markov, I. L.; and Sakallah, K. A. 2003. Shatter: efficient symmetry-breaking for boolean satisfiability. In *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*, 836–839. ACM.

Anand, A.; Grover, A.; Mausam; and Singla, P. 2016. Contextual symmetries in probabilistic graphical models. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 3560–3568. IJCAI/AAAI Press.

Audemard, G.; Benhamou, B.; and Henocque, L. 2006. Predicting and detecting symmetries in FOL finite model search. *J. Autom. Reasoning* 36(3):177–212.

Bui, H. H.; Huynh, T. N.; and Riedel, S. 2013. Automorphism groups of graphical models and lifted variational inference. In Nicholson, A., and Smyth, P., eds., *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013.* AUAI Press.

Crawford, J. M.; Ginsberg, M. L.; Luks, E. M.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996.*, 148–159. Morgan Kaufmann.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In Kaelbling and Saffiotti (2005), 1319–1325.

den Broeck, G. V.; Meert, W.; and Darwiche, A. 2014. Skolemization for weighted first-order model counting. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Eén, N., and Sörensson, N. 2003. An extensible sat-solver. In Giunchiglia, E., and Tacchella, A., eds., *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, 502–518. Springer.

Flener, P.; Pearson, J.; Sellmann, M.; Hentenryck, P. V.; and Ågren, M. 2009. Dynamic structural symmetry breaking for constraint satisfaction problems. *Constraints* 14(4):506–538.

Gogate, V., and Domingos, P. M. 2011. Probabilistic theorem proving. In Cozman, F. G., and Pfeffer, A., eds., *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, 256–265. AUAI Press.

Kaelbling, L. P., and Saffiotti, A., eds. 2005. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*. Professional Book Center.

Katebi, H.; Sakallah, K. A.; and Markov, I. L. 2010. Symmetry and satisfiability: An update. In Strichman, O., and Szeider, S., eds., *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, 113–127. Springer.

Kopp, T.; Singla, P.; and Kautz, H. A. 2015. Lifted symmetry detection and breaking for MAP inference. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1315–1323.

Meseguer, P., and Torras, C. 2001. Exploiting symmetries within constraint satisfaction search. *Artif. Intell.* 129(1-2):133–163.

Mittal, H.; Goyal, P.; Gogate, V. G.; and Singla, P. 2014. New rules for domain independent lifted MAP inference. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 649–657.

Niepert, M. 2012. Lifted probabilistic inference: An mcmc perspective. In *Proc. of 2nd Intl. Workshop on Statistical Relational AI.*

Niepert, M. 2013. Symmetry-aware marginal density estimation. In desJardins, M., and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.* AAAI Press.

Poole, D. 2003. First-order probabilistic inference. In Gottlob, G., and Walsh, T., eds., *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, 985–991. Morgan Kaufmann.

Richardson, M., and Domingos, P. M. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Sellmann, M., and Hentenryck, P. V. 2005. Structural symmetry breaking. In Kaelbling and Saffiotti (2005), 298–303.

Singla, P., and Domingos, P. M. 2008. Lifted first-order belief propagation. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 1094–1099. AAAI Press.

Singla, P.; Nath, A.; and Domingos, P. M. 2014. Approximate lifting techniques for belief propagation. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2497–2504. AAAI Press.

Van den Broeck, G. 2013. *Lifted Inference and Learning in Statistical Relational Models*. Ph.D. Dissertation, KU Leuven.

Walsh, T. 2012. Symmetry breaking constraints: Recent results. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press.