

Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Thomas J. Przybylinski

Date

New Insights Into the Similarity and Difference of Propositional Models Via Symmetry

By

Thomas J. Przybylinski
Doctor of Philosophy

Computer Science and Informatics

James Lu, Ph.D.
Advisor

Michelangelo Grigni, Ph.D.
Committee Member

Li Xiong, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the Graduate School

Date

New Insights Into the Similarity and Difference of Propositional Models Via Symmetry

By

Thomas J. Przybylinski
Ph.D., Emory University, 2015

Advisor: James Lu, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2015

Abstract

New Insights Into the Similarity and Difference of Propositional Models Via Symmetry

By Thomas J. Przybylinski

Recently, there has been an increased interest in finding diverse solutions to propositional formula. Modern SAT solvers are capable of producing a large number of models very efficiently. However, this can result in an overwhelming number of nearly indistinguishable results. Hence, finding a general way to discount most of these variants and leaving a semantically diverse set of solutions is an important problem. Historically, diversity techniques concentrated on distance, while uniform sampling and global symmetries have been put forward as diversity concepts.

No work in this area has closely examined diversity in general nor examined when their concepts return inferior results. In this dissertation we closely examine different notions of diversity, and show that existing criteria are lacking in different ways. We present a structure for unifying and combining previous notions of diversity, and study their relative power to discriminate solutions.

We then consider two new notions of diversity: local symmetry and constructive symmetry, that we show to be more intuitively satisfying. We analyze theoretical and computational properties, and present different algorithms. A comprehensive set of experiments are presented to explore different strategies.

New Insights Into the Similarity and Difference of Propositional Models Via Symmetry

By

Thomas J. Przybylinski
Ph.D., Emory University, 2015

Advisor: James Lu, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2015

Contents

1	Introduction	1
2	Technical Background and Related Work	3
2.1	Propositional Logic	3
2.2	Isomorphisms, Symmetry, and Automorphisms	4
2.3	Background Work	5
2.3.1	Diversity	5
2.3.2	Local Symmetry	6
2.4	Diverse Models and Data Mining	7
3	Augmenting Symmetry	9
3.1	Generalizing Diversity	9
3.1.1	Diversity Graphs	10
3.1.2	Symmetry-Aware Distance	12
3.2	Calculating Symmetry	14
3.2.1	JAUNTY	14
3.2.2	Internal Representation	16
3.2.3	Refinement	18
3.2.4	Pruning	19
4	Local Symmetry	20
4.1	Local Symmetry Diversity	20

4.1.1	Introduction	20
4.1.2	Computational Properties	24
4.2	Algorithms	30
4.2.1	Complete Offline Local Symmetries	30
4.2.2	Other Algorithms	39
4.3	Comparing Discrimination	53
5	Constructive Symmetry	64
5.1	Introduction	64
5.2	Constructive Symmetry Properties	66
5.3	A General Algorithm	69
5.4	Improving Constructive Symmetry	71
5.4.1	Restricting Choices	71
5.4.2	Shallow Approximation	73
5.5	Results	74
6	Discussion	80
6.1	Overview of Completed Work	80
6.2	Future Work	83
6.2.1	Link Analysis	83
6.2.2	Complete Symmetry	84
6.2.3	Over-estimating Symmetry	85
6.3	Conclusion	86
	Bibliography	87

List of Figures

3.1	The solutions for 3 Coloring a 3-Path	13
3.2	Diverse Sets using Global symmetry with and without distance. . .	13
4.1	12 Asymmetric 8 Queens Solutions	21
4.2	Similarity of Asymmetric 8-Queens. L_1 and R_1 are globally asymmetric but similar boards. L_2 and R_2 are 7x7 sub-boards of L_1 and R_1 that are 180 degree rotations of each other.	22
4.3	3 Locally Asymmetric 8 Queens Solutions	22
4.4	Maximal representative solutions to the 4×4 All-Squares problem: Set 1 is a set of global asymmetric solutions, set 2 is a set of local asymmetric solutions.	23
4.5	Diverse set ratios for 1000 100-var random 3-SAT problems.	24
4.6	Algorithm MILE*	33
4.7	Performance of MILE* and Deep Local Symmetry SCP on random sets of models	38
4.8	Algorithm Agree	39
4.9	Algorithm Online.	41
4.10	Percent Symmetries For Random 3-SAT problems After Removing Redundant Information on Various Clause/Variable ratios . . .	46
4.11	Comparing breaking and no breaking to obtain up to 100 models with 1000 second timeout. All use syntax breaking and non-deterministic phase selection, except as noted.	49

4.12	Speed in logscale milliseconds, and percentage of locally asymmetric pairs in maximal diverse sets over problems of increasing size.	50
4.13	A Diverse Set Created With DGS	60
4.14	A Diverse Set Created With Agreement Symmetry	61
5.1	The 4×4 All-Rectangles problem: Set 1 is a diverse set of solutions based on local symmetry, set 2 is based on constructive equivalence.	65
5.2	Computed diverse set for the 5×5 All-Filled-Squares problem using positive choices	72
6.1	Most unusual and characteristic ways to 3-coloring a path based on local symmetry. All other examples are globally symmetric to these.	84

Chapter 1

Introduction

Understanding similarities and differences between objects helps humans classify information and manage complexity. Science has created numerous taxonomies to describe the relationship between a host of entities through a number of measures. Similarity and diversity is often used for inference in qualitative logic. This inference tells us a set of similar objects should have many properties in common, while a set of dissimilar objects should have relatively few properties in common. Hence, if X is similar to Y, then any given property of X is more likely to also pertain to Y than if X and Y were dissimilar.

A set of objects that are the very diverse from each other exhibit the greatest number of distinct properties from the fewest possible number of objects. For example, when buying a car we would want to start by browsing a variety of cars. If we only looked at SUVs, we may never know that a compact car could meet our requirements. Moreover, similarity is useful when searching for good alternatives to some object. If we found a compact car we liked, but was too expensive, it would typically be a better use of time to look at similar cars (perhaps ones with fewer features) than to consider SUVs again.

A set of mutually dissimilar objects is called a *diverse set*, while a set of mutually similar objects is called a *similar set*. Similarity and dissimilarity are dual notions,

so it is typically easy to convert an algorithm that finds a diverse set to one that finds a similar set.

Our focus is on the diversity of objects encoded as equal-length binary data, equivalently solutions (or models) of propositional formula. Propositional logic is a reasonable basis for the study of diversity because of its simplicity and universality. There is no noise or missing data to handle, and it is expressive enough to represent all the problems in the class NP.

Solving a propositional formula is the SAT, or satisfiability problem. SAT is NP-Complete, which means there is no known efficient algorithm to solve it in general. However, most applications have structure that can be exploited for efficiency. One of the largest advances in the last 10-15 years is the tremendous progress in improving SAT solvers to utilize this structure effectively. However, most model finding programs generate an overwhelming number of nearly indistinguishable results; finding a general way to discount most of these variants and leaving a semantically diverse set of solutions is an important problem.

In this dissertation we closely examine different notions of diversity. While no one measure is definitively better than any other, we will argue that existing criteria are lacking and develop more substantial tools that are theoretically and intuitively satisfying.

Chapter 2

Technical Background and Related Work

2.1 Propositional Logic

Let V be the set of *propositional variables* (or variables). Without loss of generality, we assume V to be the first n natural numbers. The set of *literals*, L , consists of variables and their negations. A *clause* is a set of literals, and a *theory* (or a *formula*) is a set of clauses. While most work on symmetry focuses on theories in conjunctive normal form, where clauses represent disjunctions and theories are conjunctions of clauses, it is straightforward to apply our discussions to the dual representation: disjunctive normal form.

A *partial interpretation* (PI) I is a function in $V \rightarrow \{0, 1, \perp\}$, where \perp , 0 and 1 denote "undefined", "false" and "true", respectively. An interpretation (or assignment) is a PI with all variables defined. Two PI's are consistent if they agree on the truth values of all variables that are defined in both. We denote by \mathcal{I} the set of all 3^n PIs.

A PI I may be equivalently represented as a set of literals: $\{v | I(v) = 1\} \cup \{\neg v | I(v) = 0\}$. The set of all PIs \mathcal{I} forms a join-semilattice where the join of

$I_1, I_2 \in \mathcal{I}$ is $I_1 \cap I_2$. The meet may not exist since the union of two PIs that contain complementary literals is not a valid PI. PIs I_1, I_2 are *siblings* if $|I_1| = |I_2|$.

PIs are extended to clauses and theories in the usual ways. An assignment that assigns true to a theory is a *model* of the theory. A theory is *satisfiable* if it has at least one model, a *contradiction* otherwise.

2.2 Isomorphisms, Symmetry, and Automorphisms

An *isomorphism* θ of a set of models M is a permutation on L that satisfies $\theta(M) = M$ and $\theta(x) = -y \leftrightarrow \theta(-x) = y$. Suppose I is a PI. Then the set of models of M that are consistent with I is denoted M_I . Moreover, the *local symmetries associated with I* , Θ_I , is the set of all isomorphisms of M_I , which forms a permutation group. As a special case, a symmetry θ is *global* if $\theta \in \Theta_{\emptyset}$. Computing the end result of two or more permutations is called composition, denoted $\theta \circ \sigma$, so that $(\theta \circ \sigma)(x) = \theta(\sigma(x))$. We will also denote composition with \times so that $(\sigma \times \theta)(x) = \theta(\sigma(x))$.

Other than local and global symmetry, there are other important classes of symmetries [15, 7]. *Solution* or *semantic* symmetry is symmetry on the full set of models (of the underlying theory), while *constraint*, or *syntactic* symmetry is symmetry on the formula used to find the models. There is also a notion of *literal vs clause* symmetry, which act on literals and clauses, respectively.

Finding symmetries of logical theories is polytime reducible to finding the automorphisms of a graph [17]. An elegant approach is found in [2]. Although there can be many automorphisms, one can store far fewer by using a set of *generators* where every automorphism can be described as a composition of the generators.

Symmetry problems are interesting from a complexity standpoint. They are among the few problems that are known to be in *NP*, but they have also not been shown to be *NP-Complete* or in *P*. There are software packages that can find automorphisms and their generators very quickly in practice including nauty and

Traces[35, 36], saucy[32, 33], and bliss [30, 31].

2.3 Background Work

2.3.1 Diversity

The first study of diversity in SAT is by Bailleux and Marquis[6] in 1999. In dynamic or incompletely specified systems, a given solution can conflict with new observational data. When a conflict occurs, it is desirable to find a new solution that is as close as possible to the original solution. This problem is encoded as a new problem, Distance-SAT, which asks if there is a model that has at most d differences with some PI I .

In 2002, Crescenzi and Rossi[18] introduced the problem of finding two models of a boolean formula that have maximum hamming distance. This is equivalent to finding the most diverse set of size 2. In 2005 Angelsmark and Thapper[4] presented an algorithm that was faster in the worst case.

Hebrard et. al.[26] studied distance-based similarity and diversity in case-based reasoning and recommender systems. In this study they look at general well-behaved distance functions and find algorithms and complexity of many types of similarity and diversity problems. The important distinction between offline and online algorithms was introduced. Offline algorithms are computed on the set of solutions, and online algorithms computed diversity at the same time as the solutions. In 2007 Hebrard, O'Sullivan and Walsh [27] explore this further by creating and analyzing a distance-based algebra.

Eiter et. al studied similar problems in the related field of answer set programming [19, 20], especially in regard to phylogeny reconstruction.

Since then, the intersection of SAT and diversity has been concentrated on hardware applications. In particular it looks at SAT-based semi-formal hardware verification. Here, a propositional formula represents the workings of hardware subject

to constraints. For example, we may want a sequence of hardware states that will fill a queue, and so the models are all the ways the hardware could get to such a state. Optimally, all models would be tested for bugs but doing so is intractable. Therefore, a diverse set of models would maximize the coverage of the tests.

Up until now, diversity has measured using distance, typically hamming distance. This is the approach that Nadel[37] takes. Alternatives include uniform sampling, such as the work by Chakraborty et. al.[14, 13]. Jackson et. al.[29] use a symmetry-based approach that tries to enumerate the partitions formed from global symmetry in a uniform manner to enumerate different hardware architecture designs.

2.3.2 Local Symmetry

The intersection of SAT and symmetry has been primarily focused on the efficiency aspects of solving propositional formula. This idea was introduced by Krishnamurthy[34] by augmenting resolution with symmetries. The breakout paper for such ideas was from Crawford[17] who showed how to find the symmetries of first-order logical formula through a polynomial reduction to the graph automorphism problem, for which reasonable algorithms already existed. Using symmetry, Benhamou and Sais[10] were able to show the unsatisfiability of Ramsey's problem on 17 vertices and three colors. The application of symmetry to reduce computation time remained complicated until Crawford et. al[16] introduced the idea of symmetry breaking clauses. These clauses, if fully implemented, would remove all but the lexicographically largest model from each global symmetry partition. Aloul, with others, improved practicality of these techniques [2, 1, 3]. Among the improvements are: showing that breaking on generators is often effective, developing a representation that use only a linear number of symmetry-breaking clauses, and showing how to reduce the number of symmetry breaking clauses further by removing redundant clauses.

Relatively little work was done on local symmetries. Arai and Urquhart [5] showed that augmenting resolution using local symmetry can still lead to exponential proofs. On the other hand, Benhamou with others [9, 8] investigates several procedures to prune SAT branches using local symmetry.

There has also been symmetry work for CSPs in general, including work by Benhamou[7], Gent with others [22, 23, 24], and Walsh[39, 38, 40]

2.4 Diverse Models and Data Mining

Finding diverse models has some relation to the field of Data Mining (also known as knowledge discovery from data, or KDD) and its closely related field machine learning (ML). The goal of data mining is to find interesting patterns in (potentially large amounts of) data. With respect to KDD or ML, our work falls into the area of *unsupervised learning* as we are attempting to find properties of a set of models (diversity or similarity) without having any labeled examples of what such models look like.

But this work deviates from KDD and ML in some important ways. The use of proposition logic ensures full information: there is no noisy or missing data. Still, insights can be gained by comparing our approach with approaches from KDD and ML such as clustering.

Distance Data mining utilizes a range of similarity and dissimilarity measures when comparing data[25]. For binary data these include a normalized hamming distance, as well as an alternation that ignores matches where both data is false.

When comparing vectors of numeric data, the most common measures are the L_p norms, where $d(x, y) = \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + \dots + |x_k - y_k|^p}$. Famous examples are Manhattan distance ($p = 1$) and Euclidian distance ($p = 2$). Other measures violate the triangle inequality but are still useful, such as cosine dissimilarity, where the angle between the vectors is computed. None of these measures

are inherently superior to any other, the correct measure to use depends on the context. For example Euclidian distance is the most useful for determining the distance between objects in space, but is not as good as cosine dissimilarity when comparing documents using word frequency.

Occasionally more advanced methods are needed to correctly compare data, such as dynamic time warping[11] from the speech recognition community for time series data. Spellcheckers make use of specialized edit distance metrics to suggest likely corrections. These approaches take into account structural information to achieve better results for their particular problems. We will also be using the structural information encoded in propositional models and formula to implement more discerning measures of diversity.

Chapter 3

Augmenting Symmetry

Each measure of diversity considered in the literature has strengths and weaknesses, and it may be useful to consider their effects when combined. In this chapter, we present a general model of diversity that can capture all published concepts of diversity. This leads to a method of combining distance and symmetry into a diversity measure that surpasses a naive combination. We will then investigate computational concerns related to calculating symmetries for sets of models, and introduce the software system used to address these concerns.

3.1 Generalizing Diversity

Generic distance measures consider diversity syntactically; they are oblivious to the meaning of the models. More specialized distance functions may model the structure of problems more closely, but require specific analysis. Similarly, global symmetry can take into account semantic differences, but is unable to distinguish similarities between models that have a similar, but not identical, structure. Since the weakness of one is the strength of another, this suggests it can be useful to combine them to create an overall stronger notion of diversity.

To illustrate, take a simple example of coloring a path. Suppose m_1 , m_2 are

two colorings that differ on the color of exactly one node. Let θ denote a permutation on the set of colors that “rotates” the colors (i.e., maps color i to $(i + 1) \bmod p$ where p the number of colors). Clearly, $m_1, m_2, \theta(m_1)$ and $\theta(m_2)$ are all structurally similar.

One common distance metric for binary data is Hamming distance, calculated as the number of variables that disagree on their assignments. This is the distance that all previous work on SAT diversity used. However, its relation to the actual structural diversity of the problem can be quite limited. Event though m_1 and $\theta(m_1)$ are structurally identical, the difference being the colors we chose, according to Hamming distance these two models are the furthest possible since they disagree on the color of every node.

Since θ is a global symmetry, m_1 and $\theta(m_1)$ are similar, but m_1 and m_2 are not. Global symmetry is restricted to discovering structural identities, not similarities, so even a minor change can drastically change the similarity between two models. Even worse, sometimes there is no nontrivial global symmetry at all, just the identity.

Intuitively, any notion of similarity should consider $m_1, m_2, \theta(m_1)$ and $\theta(m_2)$ to be mutually similar. We can combine Hamming distance and global symmetry to do this, but it requires some post-processing.

If we just combine them naively, then the combination would determine that m_1 is similar to $\theta(m_1)$ and also m_2 , but it could not determine that m_1 is similar to $\theta(m_2)$ since they are not globally symmetric, and will agree on the assignment of at most one variable.

3.1.1 Diversity Graphs

To create a more sophisticated combination, we require a structure that allows us to incorporate symmetry and distance together. Let M be the set of all models. Given a weighted graph $G = (M, E)$, a function of weighted graphs to

unweighted graphs ρ s.t $\rho(G) = H$ where $H = (M, E')$. We call (G, H, ρ) a diversity family. Here, G represents some numerical evaluation of pairwise diversity, and ρ specifies which pairs should be considered diverse by evaluating G to create H , which consists of two nodes are connected by an edge if and only if they are considered diverse. Then a diverse set of this family is any independent set of H .

This provides a very flexible but complicated framework for diversity. To simplify, we observe that any (G, H, ρ) can be replaced with another diversity family (G', H, ρ') where the edges of G have real weights ≥ 0 , and ρ is simply a threshold function. Then m_1, m_2 is an edge in H if and only if the edge exists in G and it has weight $\geq k$ for some k determined by our choice of ρ' . This transformation could be implemented by enforcing G' to have the same edge structure as H , except we set every edge to a constant weight k and set ρ' to keep edges $\geq k$.

By allowing only non-negative weights we achieve greater salience in our graph. An edge of weight 0 represents two models that are identical. Larger weights represent increased dissimilarity, and the lack of an edge corresponds to an edge of infinite weight: two models that are completely dissimilar.

This allows the removal of ρ and H from our structure. What is left is G , which we call a *diversity graph*. Diverse sets can then be found through direct algorithms.

Definition. A set $S \subseteq M$ of models is a *k-threshold independent set* if no pair in S has an edge with weight $\leq k$.

Definition. A *k-threshold independent set* of a diversity graph is called a *k-diverse set* or simply *k-diverse*. If the k is implicit, unimportant, or obvious, we will call such a set a *diverse set* or just *diverse*.

How we choose to construct this set can also have applications to diversity. Attempting to find candidates using uniform sampling on a graph with no edges corresponds to the diversity suggested by Chakraborty et. al. and Jackson et. al.'s diversity concept is fulfilled by a diversity graph whose edges come from

global symmetry. So a symmetry graph can encompass all published notions of diversity, requisite on a particular diversity graph and a particular algorithm to find a k -threshold independent set.

We will often label our diversity graphs based on what information is used to construct them. For example, a graph whose edges correspond to distance is called a *distance graph*, while a graph where there are only edges between symmetric models is known as a *symmetry graph*. Greater specificity can be used when needed, e.g. a distance graph that uses Hamming distance is known as a *Hamming distance graph*.

One interesting choice is the weight of the edges of a symmetry graph. Should they be 0, representing identity, or should they be some small ϵ since even though they have the same structure, they are not precisely equal? We take the latter approach., which is especially important for local symmetry diversity.

This graph has numerous applications, and many existing applications are easily understood in this context. This also suggests more sophisticated analyses, such as link analysis and graph centrality (e.g., PageRank), to find characteristic or unusual models.

3.1.2 Symmetry-Aware Distance

Now that we have an approach that deals with symmetry and distance equally, we can describe how to combine distance and symmetry to create a synergistic approach to diversity.

The naive combination is setting the edge weight to be the minimum of the edge weight in the distance and symmetry graphs. Recall our models for the path coloring problem. For all but the most lax thresholds, no diverse set will ever contain more than two of $\{m_1, m_2, \theta(m_1), \theta(m_2)\}$.

Recall that m_1 and $\theta(m_2)$ have no symmetry together and have a large distance, and so are among the most dissimilar models in our combined graph. The problem

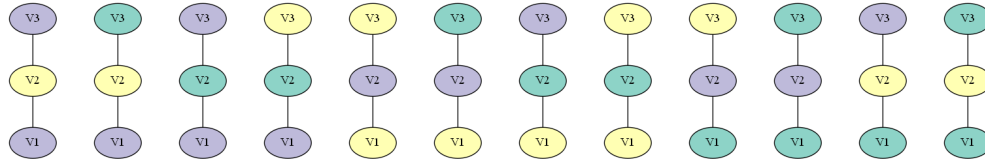


Figure 3.1: The solutions for 3 Coloring a 3-Path

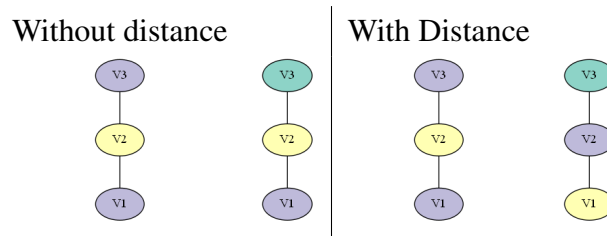


Figure 3.2: Diverse Sets using Global symmetry with and without distance.

is edge weights to not directly tell us that m_1 is very similar to m_2 , which is structurally identical to $\theta(m_2)$.

This suggests a general property for diversity: if objects a and b are very similar, and objects b and c are very similar, than objects a and c should not be too dissimilar. This idea applied to distance is the triangle inequality.

To be more concrete consider 3-coloring an n -path. The twelve solutions for 3-path are shown in Figure 3.1. The purple is color 1, yellow is color 2, and cyan is color 3.

Figure 3.2 shows diverse sets created with global symmetry only, and with global symmetry naively combined with distance. Adding distance has made the set more diverse by giving the second model a different appearance even though structurally they are exactly the same.

We are able to use the information already in the graph to perform a more sophisticated collusion. The logic to show that m_1 and $\theta(m_2)$ are similar is directly encoded as a path from one to the other. The path that goes from m_1 to m_2 to $\theta(m_2)$ traverses a total distance of $1 + \epsilon$, representing an extremely similar connection. A new diversity graph is created from the combined graph by making the

weight between two nodes be the *shortest path* distance between the nodes in the combined graph. No diverse set generated from this graph will ever contain more than a single model in $\{m_1, m_2, \theta(m_1), \theta(m_2)\}$, as desired.

Our shortest-path graph is a distance graph since it passes all the necessary properties. This new distance is symmetry-aware and uniquely computed for each set of models. In addition, finding globally symmetric models and shortest paths are relatively fast operations, making this procedure a good first-cut at diversity. Indeed, the merits of combining distance and symmetry will go beyond global symmetry.

3.2 Calculating Symmetry

An important procedure in our work is one that calculates the symmetries of clausal formula. We implemented several software packages that can solve this problem in the form of a graph automorphism but there are cases where this transformation is detrimental to the speed of the symmetry calculations. For example, when converting a formula to a graph there must be a node for every clause. In some cases there could be many more models than there are literals (in the worst case, exponentially more). This extra burden can be costly.

The size and structure of the formula will be significantly different for local symmetry, and there will be times when it is necessary to call the symmetry finding procedure over a million times within several minutes, so we need a tool that performs well under all circumstances and with minimal overhead.

3.2.1 JAUNTY

We introduce JAUNTY, a propositional logic toolkit. It was originally written to quickly iterate through prototype knowledge compilation techniques but has grown into a 45,000 line general-purpose propositional logic tool. This allows us

to use a single API and structure to perform all the procedures necessary to run all of our tests. It also minimizes unnecessary copying and reduces the latency required to run symmetry calculations.

Propositional Logic Functionality JAUNTY is capable of representing arbitrary propositional formula, and can transform them into equivalent NNF, DNF or CNF form. There are also simplified formula for fast CNF and DNF usage. Extending these are specialized classes to handle important functions, such as verifying symmetries and obtaining sub-formula when applying PIs. It is able to solve SAT problems within its own code, but it also integrates the SAT4J solver to iterate through models for us.

It is capable of loading and saving clausal formula in the DIMACS format, and has an extensive library of classes that can be used to generate formula of particular types. Many of these classes can also generate graphics that represent each solution. Graphviz is used to generate graph-based output.

Symmetry Functionality The most popular automorphism algorithm for SAT formula is saucy [32], which was built specifically to solve the kinds of sparse graphs that are typically created from real-world CNF formula. However, sets of models have different structure than most CNF formula. The derived graph is more dense since every model contains $|V|$ literals. The ratio of models to variables may also be larger.

When computing local symmetry, many PIs assign many variables, so relatively small subsets of M will typically be found. The local symmetries of less restrictive PIs are also related to the local symmetries of their supersets. Given $I \subset I'$, if θ is a local symmetry of I and $\theta(I') = I' - \theta$ is a *set stabilizer* of I' —then we know that θ is also a local symmetry of I' .

To take full advantage of these properties, JAUNTY implements a symmetry finding algorithm heavily based off of saucy. Among the key differences is that

JAUNTY finds the symmetries of clausal representations (e.g. CNF or DNF) without turning them into graphs as an intermediate step. JAUNTY can return all permutations, or just generators, and can be told to find only permutations that meet certain parameters or even be told to stop finding permutations once some user-created property is achieved. JAUNTY is programmed with flexibility in mind rather than pure speed. For example, being written in Java means that it is the only symmetry finding program we know of that should be able to run on multiple platforms without needing recompiling. Since optimization is a secondary concern, there are certainly cases, such as computing only global symmetry for the largest CNF instances, where SAUCY or BLISS would be a preferable choice.

When it finds a symmetry, it uses a callback mechanism similar to the one found in BLISS to allow the user to handle it arbitrarily. An important difference is that the callback function returns a boolean that tells the symmetry finding algorithm whether it should stop trying to find symmetries or not. So for example, if we wanted to compute if some CNF formula has a non-trivial isomorphism, the program can stop as soon as it finds one instead of waiting for it to find all of them.

Some necessary restrictions are hard coded, such as requiring $\theta(x) = -y \leftrightarrow \theta(y) = -x$. We also employ techniques from Butler[12] to use a set of known symmetries to avoid searching for them. Finding a set stabilizer group is at least as hard as finding an automorphism group, so for $I \subset I'$, we will only input the generators of the subgroup of symmetries of I that fix every literal of I' .

3.2.2 Internal Representation

The most notable feature of saucy is its use of ordered partition pairs (OPPs), which are indexed pairs of partitions that keep track of which nodes can be validly mapped to other nodes. The search occurs by choosing an element of a "top" partition and force it to map to a node in the "bottom." JAUNTY also represents

intermediate permutations using using an Ordered Partitions Pair.

An *ordered partition* $\pi = [W_1][W_2]\dots[W_m]$ of V is an ordered list of non-empty subsets that form a partition of V . The subsets W_i are called the *cells* of the partition. An Ordered partition is *unit* if it consists of exactly one cell, and *discrete* if every elements of v is in a distinct cell. An *ordered partition pair* Π can be denoted

$$\Pi = \begin{bmatrix} \pi_T \\ \pi_B \end{bmatrix} = \begin{bmatrix} T_1|T_2|\dots|T_m \\ B_1|B_2|\dots|B_k \end{bmatrix}$$

with π_T and π_B referring to the top and bottom, respectively. An OPP is *isomorphic* if the top and bottom have the same number of cells, and $|T_i| = |B_i|$ for all integers i , $1 \leq i \leq m$, otherwise it is *non-isomorphic*. An OPP is discrete (resp. unit) if its top and bottom partitions are discrete (resp. unit).

If an OPP is isomorphic, we can think of it as encoding a set of potential permutations, in particular those that send each element of T_i to some unique element of B_i for all integers i , $1 \leq i \leq m$. Hence, a unit OPP represents $Sym(V)$ while a discrete OPP represents a single permutation. Non-isomorphic OPPs represent the empty set, because it is not possible to find a permutation in such an OPP.

For a more in-depth survey of how OPPs are used, see Katebi et. al.[32]. A search tree is created by taking a single top element e_T of some cell T_i , and then for each element e_B of corresponding cell B_i , create a new branch with the new OPP being created by removing e_T and e_B and from their cells and placing them in new corresponding unit cells. So for example, if $\Pi = \begin{bmatrix} 0, 1, 2 \\ 0, 1, 2 \end{bmatrix}$, then if

$e_T = 0$ and $e_b = 1$, our resulting OPP is $\Pi = \begin{bmatrix} 0|1, 2 \\ 1|0, 2 \end{bmatrix}$, and the permutations it represents is $\{(0, 1), (0, 1, 2)\}$. The meaning of this is “element 0 is mapped to element 1.” This step is performed recursively until it results in a unit OPP. The only change between this and what JAUNTY does is that when we map literal x to literal y we also map $\neg x$ to $\neg y$ at the same time.

3.2.3 Refinement

An important technique that speeds up the search is called *refinement*. The formal idea for the graph automorphism problem is that for each pair of elements in a single cell of ordered partition π , they must be neighbors to the same number of elements in every cell of π . Otherwise, they need to be split into different cells, since there is no way they can map to the same set of nodes. Refinement can be done on the top and bottom separately and if the resulting OPP is non-isomorphic, we can stop the recursive search. A related complication is that the adjacency frequencies should be isomorphic for every pair of top and bottom cells. As such, it is often useful to refine the top and bottom simultaneously[33].

Refinement is essential for efficient algorithms, but it is a time-consuming process and is typically the bottleneck of automorphism algorithms. Models can vastly outnumber literals, which means much unnecessary time can be spent on performing refinement on the clausal nodes. Since representing clausal symmetries is unnecessary, given that they are side-effects of the variable symmetries, we choose to compute symmetries using the variable representation without transforming it first into a graph.

A key difference between JAUNTY and saucy is when comparing the frequencies of literals x and y , the frequencies of the literals $\neg x$ and $\neg y$ also have to be equal. Two literals are considered "neighbors" if they exist in the same clause. If they coexist in more than one clause, this frequency is used during refinement. The information can be easily stored in a $2|V| \times 2|V|$ matrix, so the only time we need to work with the formula is when verifying that a generated permutation is indeed a symmetry – an infrequent occurrence.

Since we will also need to find symmetries on typical CNF instances with many variables, that matrix can be much too large. So there is also a sparse version of refinement that works more nicely on typical CNF instances. Through effective use of data structures we are able to compute refinements on very large problems very quickly.

3.2.4 Pruning

Using OPPs, SAUCY incorporates a number of pruning techniques: coset pruning, Orbit pruning, Matching OPP pruning, and Non-isomorphic OPP pruning. We have already discussed non-isomorphic OPP pruning since it is so closely tied to refinement. Coset and Orbit pruning is important for finding generators, and JAUNTY uses a form of Orbit pruning. Coset and Orbit pruning are group-theoretic techniques, a full discussion is beyond the scope of this dissertation. However, they are important because they allow us to find a set of irreducible generators, so the space the output takes up is logarithmic to the total number of symmetries. Matching OPP pruning is a pruning technique where certain OPPs allow the identification of an automorphism without checking the rest of the sub-tree.

In our sparse version, these are augmented with some additional pruning. Given some clause C , it is *finished* if every literal in C belongs to a cell of size 1 in the current OPP. Every time a new unit cell is created, the formula is checked for finished clauses. If there are duplicate clauses we count the number of each type. If a finished clause, when permuted using our current partial permutation, does not exist in the original formula or does not exist with the proper frequency, then this branch of the search tree will never give us a valid permutation and can be pruned. We keep track of which clauses are finished using a watched literal strategy, but since all literals need to be permuted for a clause to be finished only a single watched literal per clause is required. So such a mechanism to be highly scalable. This procedure also guarantees that once we come to a unit OPP in our search, it is a valid symmetry.

With JAUNTY's flexibility and low latency, we are able to perform tens of thousands of symmetry calculations efficiently on formula with millions of clauses and tens of thousands of variables.

Chapter 4

Local Symmetry

4.1 Local Symmetry Diversity

4.1.1 Introduction

We present a novel notion of diversity based on local symmetry, which is able to discover deeper structural properties than global symmetry. Consider, for example, the 8-queens problem: place 8 queens on an 8x8 chess board in such a way that no two queens are positioned in the same row, column, or diagonal. The 92 distinct solutions to the 8-Queens problem can be represented by 12 asymmetric solutions. Every one of the 92 solutions maps to one of the 12 representatives through rotation or reflection. Twelve “representatives”, one from each set, are shown in figure 4.1.

But a closer inspection of the 12 solutions reveals that, despite their asymmetry, there are still pairs that share very similar patterns. Consider Figure 4.2. It shows two globally asymmetric solutions of the 8-Queens problem in L_1 and R_1 , their structures are very similar. They share a common queen in the lower-left corner (LLC). In its absence, the two 7x7 sub-boards, L_2 and R_2 , formed by removing the row and column that contain the LLC are simply 180 degree rotations of each

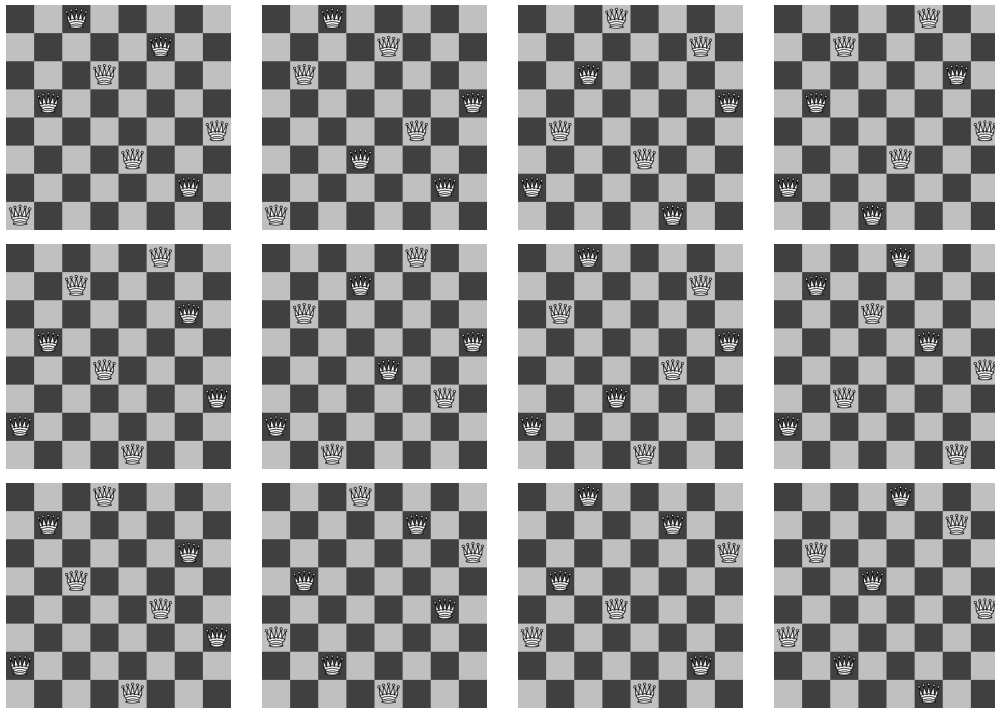


Figure 4.1: 12 Asymmetric 8 Queens Solutions

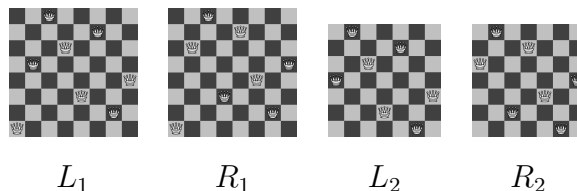


Figure 4.2: Similarity of Asymmetric 8-Queens. L_1 and R_1 are globally asymmetric but similar boards. L_2 and R_2 are 7×7 sub-boards of L_1 and R_1 that are 180 degree rotations of each other.

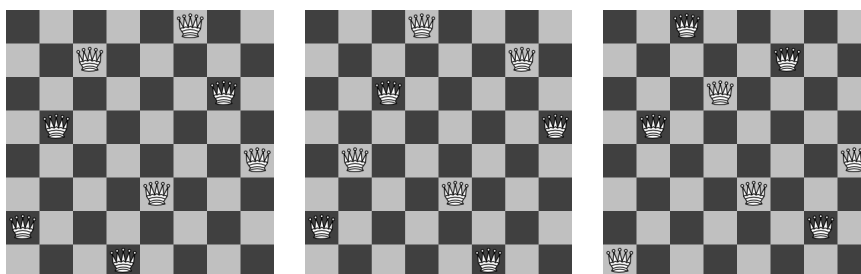


Figure 4.3: 3 Locally Asymmetric 8 Queens Solutions

other. There are two other globally symmetric solutions with a queen in the LLC: by a diagonal flip of L_1 and R_1 . These four boards, which are the only solutions where there is a queen in the LLC, illustrate conditional/local symmetry: Once we fix a queen in the LLC, all other solutions are “essentially” the same.

This difference goes beyond global symmetry into local symmetry. By considering local symmetry, a diverse set of solutions of the 8-Queens problem is shown in 4.3. As we would hope, there appears to be little in common among these boards.

A simple problem we call $N \times N$ All-Squares is especially useful for presenting the strengths of local symmetry. Given an $N \times N$ grid of monochrome pixels on a plane: the color of each pixel is associated with a propositional variable being false (white) or true (black). Models are grids where the black pixels form the outline of a single square. Global symmetry only considers rotation and reflection. Squares of the same size but positioned differently on the plane are thus

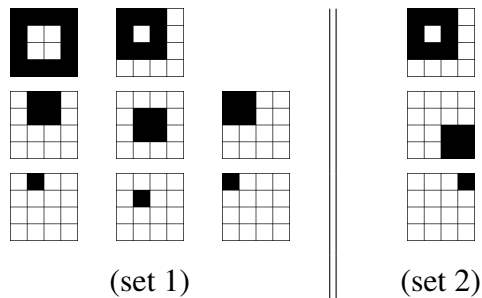


Figure 4.4: Maximal representative solutions to the 4×4 All-Squares problem: Set 1 is a set of global asymmetric solutions, set 2 is a set of local asymmetric solutions.

asymmetric. Under conditions that require certain pixels to be of a given color, same-sized squares would be locally symmetric. In other words, local symmetry can "discover" translational symmetry in this case. Figure 4.4 illustrates maximal representative sets of asymmetric solutions for the 4x4 All-Squares problem. It is hard to see how generic distance measures such as hamming distance could differentiate these cases in a similar way. Uniform random sampling is also not helpful since the number of squares of different sizes can be drastically different. There are N^2 1×1 squares, while there is only a single $N \times N$ square.

Although symmetry diversity aims to address diversity by accounting for the semantics of the problem, local symmetry diversity appears to capture diversity even for problems with no inherent semantics, such as random 3-SAT at the phase transition. For such unstructured problems, a natural way of modeling diversity is through clustering — we call *r-diversity*.

Given a set of models S and $r > 0$ (the radius), let $Clust(S, r)$ be a partitioning of S where two models are in the same partition if their Hamming distance is less than or equal to r .¹ This forms a partition of similar models much in the same way global symmetry partitions a given set of models. Then, if a subset of S consists

¹This is a simplification of clustering algorithm DBScan [21].

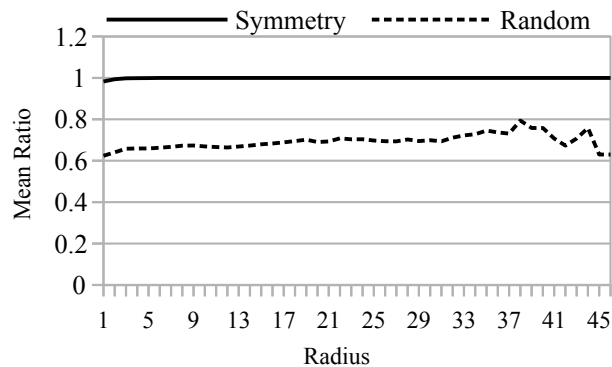


Figure 4.5: Diverse set ratios for 1000 100-var random 3-SAT problems.

of a random sample of models, one from each cluster, then it is r -diverse.

Local symmetry diversity mimics r -diversity for random 3-SAT well. The solid line of Figure 4.5 shows the percentage of local symmetry diverse sets (as computed through the online, approximate algorithm described in this paper) that contain a subset that is r -diverse. Randomly selecting the same number of points gives far less resemblance, as shown in the dashed line. The clusters in random 3-SAT have disparate sizes, making it unlikely that uniform sampling will find a representative of a small cluster. These results show the ability of local symmetry to discover structure in any set of models, even for random problems.

4.1.2 Computational Properties

The first step is to compute the local symmetry graph for a set of models M . This involves finding all pairs of models that are locally symmetric. The decision variant of the problem, we call *locally symmetric models* (LSM) is, given $m_1, m_2 \in M$, whether there is a local symmetry that maps m_1 to m_2 . Whether a local symmetry θ maps m_1 to m_2 is polynomial-time verifiable and hence LSM is in NP. It is at least as hard as the Graph Isomorphism problem (GI).

If we restrict the PIs of LSM by disallowing those that contain certain literals, then LSM is NP-Complete. We call this problem *LSM with Restrictions* or *LSM-R*, and show a reduction from the subgraph isomorphism problem: Given graphs G and H , return whether a subgraph of G is isomorphic to H .

A variable represents a node of a graph, and there is an additional variable g that we use to denote membership of edges in graph G . Our restriction allows only PIs that contains no literals belonging to H . For compactness, we will not portray the models' negative literals. For every edge $e = \{n_1, n_2\}$, we add a model $\{n_1, n_2, g\}$ if the edge is in G and $\{n_1, n_2\}$ if it is in H . Lastly, we add $m_1 = \{g\}$ $m_2 = \{\}$. For nontrivial graphs, this requires g to map to $\neg g$, which forces edges of G to edges of H . Since our restriction only allows us to remove nodes of G (by having a PI with literal $\neg n$), there is a local symmetry if and only if some induced subgraph of G is isomorphic to H . We thus derive the following.

Lemma 4.1. *LSM-R is NP-Complete.*

Agreement and Graph Structure

The *agreement* of two models m_1, m_2 is $m_1 \cap m_2$, denoted $\alpha(m_1, m_2)$. The models consistent with $\alpha(m_1, m_2)$, $M_{\alpha(m_1, m_2)}$, is called the *agreement set* of m_1, m_2 . An *agreement symmetry* (with respect to m_1 and m_2) is a symmetry of $M_{\alpha(m_1, m_2)}$. The diversity graph of agreement symmetry is called an *agreement graph*. In all, there are $O(|M|^2)$ agreements over M , which is exponentially smaller than the size of \mathcal{I} . Suppose $m_1, m_2 \in M$. If there is no other model m_3 for which $\alpha(m_1, m_2) \subseteq \alpha(m_1, m_3)$, then m_1 is said to have *maximal agreement* with m_2 in M .

Lemma 4.2. *Given M , if m_1 has maximal agreement with m_2 , then $M_{\alpha(m_1, m_2)} = \{m_1, m_2\}$.*

If there were more than 2 distinct models, then there is a literal that is contained

in two models but is not contained in at least one other model. Hence, the agreement of those two models is a superset of $\alpha(m_1, m_2)$.

Theorem 4.3. *For any nontrivial set of models M , the agreement graph is connected.*

For any pair of models m_1, m_2 , they either have maximal agreement and are trivially symmetric by mapping like literals to like literals, or we can find a model $m_3 \in M_{\alpha(m_1, m_2)}$ where m_2 and m_3 have maximal agreement and reduce the problem to showing m_1 and m_3 are connected.

As the agreement graph is connected there are at least $|M| - 1$ edges. Hence each model is similar to at least one other model. Since agreement symmetries form a subset of local symmetries, these results hold for them as well.

Corollary 4.4. *The diversity graph of full local symmetry is connected.*

We can actually make several stronger statements about the connectivity of an agreement symmetry graph. For example, not just the graph, but also every subgraph induced by a partial interpretation is connected.

Theorem 4.5. *A model m_1 , of a set of models M , $|M| \geq 3$ can have a single neighbor only if it has an assignment of variable v that is unique*

Proof. Let m_2 be a neighbor of m_1 . Let $D = m_1 - m_2$. If there exists some $D' \subseteq D$ that is non-empty, and there exists a distinct model m_3 that is consistent with D' , there must be a path from m_1 to m_3 of models consistent with D' . Since m_2 disagrees with every literal of D , m_2 cannot be part of that path. So the only way for m_1 to not have a different neighbor is if every other model disagrees with every individual literal of D , hence at least one assignment of some variable must be unique. \square

A naive reading of this would imply that there are at most $|V|$ models with a single neighbor, when we give every model a single variable with a unique

variable assignment. However, such construction actually has much symmetry, since every pair will agree on the assignment on all but two variables (let us call them a, b), we can then map the assignments from one model to the other either through a phase shift $((a \rightarrow \neg a)(b \rightarrow \neg b))$, or through simple swaps $((a \ b))$, which has the same effect.

This graph can be modified to obtain the minimum number of edges. First we will break the symmetry of a single pair, and then generalize it to the entire set. WLOG, say that every variable of every model is assigned false except for their single uniquely-valued variable, so m_1 assigns a true and m_2 assigns b true. If we add a model m_3 that assigns all variables to negative, then it is part of the $M_{\alpha(m_1, m_2)}$ as well. Therefore, we cannot permute m_1 to m_2 via phase shift symmetries since that would force m_3 to assign a and b true, an assignment that is not a model. Next, we split b into two equivalent variables, b_1 and b_2 . Both of these variables have a unique assignment on m_2 , and it also keeps us from permuting a and b through a swap. So m_1 and m_2 are not symmetric in this partial interpretation, and there is no other agreement set that contains both of them. However, they are both connected to m_3 . The introduction of m_3 will eliminate all agreement phase shift symmetries between the previously existing models, while splitting every variable into a unique number of equivalent variables will remove all existing swaps. So the only model that any (indeed every) model is symmetric to is m_3 . This is a tree, so there are exactly $|M| - 1$ edges, proving our bound is tight. However, there are only $\sqrt{|V|}$ models of degree one: the tightest bound which we are aware of.

This construction relies on the existence of equivalent variables. If we remove equivalent variables, the agreement graph will provably more dense. Equivalent variables can be easily found and removed when we have the set of models. This simplification can also be performed on a CNF formula, but this requires $O(|V|^2)$ NP-Complete operations.

Theorem 4.6. *On graphs with 3 or more nodes, when no literal is equivalent to*

any other literal there can only be a single vertex of degree 1

Proof. . Assume m_1, m_2 both have degree 1. We know they are not adjacent because the graph is connected. WLOG there are variables v_1 and v_2 that are positive in m_1 and m_2 , respectively and negative everywhere else. Let m_3 be the model adjacent to m_1 , we know that $m_1 - m_3$, is simply v_1 , since an additional variable would either be equivalent to v_1 or require m_1 to have degree > 1 due to Theorem 4.5. This is also true for m_2 and m_3 . This means that m_2 and m_3 must disagree on ONLY v_1 and v_2 . Then we know that $M_{\alpha(m_1, m_2)} = \{m_1, m_2, m_3\}$ and there is a local symmetry from m_1 to m_2 by mapping v_1 to v_2 and vice versa, making each have degree 2, contrary to our assumption. \square

Hence, when there are no equivalent literals we would expect to find a much denser agreement graph.

Pruning Properties

We identify several properties useful for reducing the number of partial interpretations for which symmetry needs to be computed.

Canonical Partial Interpretations. Given a PI I , the *fixed literals* of I , $\sigma(I)$, are literals, not in I , that appear in every model of M_I . The *fixed extension* of I , I^* , is the union $I \cup \sigma(I)$. We say that I is *canonical* if $I = I^*$.

Example 4.7. *Suppose*

$$M = \{\{1, 2, 3\}, \{1, -2, 3\}, \{-1, 2, 3\}, \{-1, -2, -3\}\}$$

Given I_2 and I_9 of Example 4.12, $M_{I_2} = \{\{1, 2, 3\}, \{1, -2, 3\}\} = M_{I_9}$, $\sigma(I_2) = \{3\}$ and $I_2^ = \{1, 3\} = I_9^*$. In other words, I_9 is canonical, I_2 is not, and I_9 is the fixed extension of I_2 .* \square

The following is immediate.

Lemma 4.8. $M_I = M_{I^*}$ for any $I \in \mathcal{I}$.

It follows that $\Theta_I = \Theta_{I^*}$; to compute the diversity graph, it suffices to compute only symmetries of canonical PIs.

Cardinality. A PI I has the *non-unit property* if $|M_I| > 1$. Any PI without the non-unit property does not contribute to the diversity graph since no interesting symmetry exists for unit (or the empty) sets. Hence, we may omit from consideration all such PIs. Note that all agreements are canonical and non-unit.

The Symmetry Composition Property. We may compute the symmetries of a PI J through a symmetry ϕ of a weaker condition (i.e., a subset of J). If the symmetries $\Theta_{\phi(J)}$ of the permuted PI $\phi(J)$ have already been computed, then we may compose them with ϕ to determine the symmetries Θ_J .

Example 4.9. Consider *8-Queens* and the problem of finding symmetries under the condition J where the *URC* (upper-right corner) holds a queen. Among the symmetries of the weaker condition $I = \{\}$, there is a symmetry ϕ that maps the *URC* to the *LLC*. Applying this symmetry to J produces a PI $\phi(J)$ that is consistent with boards L_1 and R_1 of Figure 4.3. Since L_1 and R_1 are locally symmetric in $\phi(J)$, we know that $\phi^{-1}(L_1)$ and $\phi^{-1}(R_1)$, their *URC* counterparts, are also symmetric. \square

In general, we simply need to permute each of these symmetries through the inverse of the above mapping to obtain the symmetries of the condition J . We formalize this property next.

Theorem 4.10. Given a PI J and $I \subset J$. If ϕ is a symmetry of M_I and θ is a symmetry of $M_{\phi(J)}$, then $\phi^{-1} \circ \theta \circ \phi$ is a symmetry of M_J .

The proof follows from noting that applying the permutation $\phi^{-1} \circ \theta \circ \phi$ to any model in J gives us another model in J , thus it is a local symmetry of J .

Corollary 4.11. If $e = (m_1, m_2)$ is an edge of the diversity graph created from the local symmetries of $\phi(J)$, then $\phi^{-1}(e) = (\phi^{-1}(m_1), \phi^{-1}(m_2))$ is an edge in J . In addition, if E is the set of all edges generated by $\Theta_{\phi(J)}$, then $\phi^{-1}(E) = \{\phi^{-1}(e) | e \in E\}$ is the set of all edges generated by the local symmetries of J .

We call a pair (J, J') of PIs *correspondent* if $J' = \phi(J)$ for some symmetry ϕ of a subset I of J . The symmetry ϕ is called the *link*.

4.2 Algorithms

We present three algorithms of increasing efficiency. The baseline is an offline, complete algorithm that computes the diversity graph for full local symmetry.

4.2.1 Complete Offline Local Symmetries

Recall that our goal is to compute the local symmetry graph. This involves finding all pairs of models that are locally symmetric in M . We abbreviate such pairs of models as LSPs. To compute, we traverse the semilattice \mathcal{I} and find the semantic symmetry group (specifically a generator set) associated with each PI. This symmetry group will encode both clause (model) symmetry and variable symmetry.

The simplest approach employs a depth-first traversal of \mathcal{I} . Each element of \mathcal{I} is a subset of L , and different algorithms for subset enumeration exist. Siblings are ordered relative to some lexicographic ordering, and repeats are discarded.

Specifically, we employ a depth-first traversal called MILE that starts with the empty PI. Sibling PIs are ordered in inverse lexicographic ordering, and repeats are discarded.²

Example 4.12. *Suppose $V = \{1, 2, 3\}$. Then MILE enumerates \mathcal{I} in the order shown below to the right (represented once as sets and once as strings). We number the PIs for easy reference later.* □

A quick inspection of the progression in the MILE sequence shows that, given a PI I where v is the largest absolute value of the literals in I , all combinations of

²MILE stands for the Mutilated Inverse Lexicographic Enumeration since the inverse lexicographic order is observed only on siblings.

literals from $\{v + 1, \dots, n\}$ are enumerated before the sibling immediately following I is visited. While this of course is just the definition of depth-first search, we especially name the value v as the *anchor* for two reasons. First, a naive depth-first traversal of the semilattice \mathcal{I} visits many PIs multiple times, and the anchor provides a simple check for preventing such repeated visits. Second, as we will see shortly, the anchor helps as a bookkeeping device that enables us to implement more complex pruning.

More generally, we associate the nodes of the MILE traversal with a pair (I, η) , where $I \in \mathcal{I}$, and $0 \leq \eta \leq n$ is the anchor. Note that the anchor is associated with the position of the node of the traversed tree, not the PI in that node. In the case of MILE, the anchor coincides with the largest absolute value of the literals in the PI, but that association does not hold in general.

$I1:$	$\{\}$	\perp	\perp	\perp	$I15:$	$\{-1,-2\}$	\perp	0	0
$I2:$	$\{1\}$	\perp	\perp	1	$I16:$	$\{-1,-2,3\}$	1	0	0
$I3:$	$\{1,2\}$	\perp	1	1	$I17:$	$\{-1,-2,-3\}$	0	0	0
$I4:$	$\{1,2,3\}$	1	1	1	$I18:$	$\{-1,3\}$	1	\perp	0
$I5:$	$\{1,2,-3\}$	0	1	1	$I19:$	$\{-1,-3\}$	0	\perp	0
$I6:$	$\{1,-2\}$	\perp	0	1	$I20:$	$\{2\}$	\perp	1	\perp
$I7:$	$\{1,-2,3\}$	1	0	1	$I21:$	$\{2,3\}$	1	1	\perp
$I8:$	$\{1,-2,-3\}$	0	0	1	$I22:$	$\{2,-3\}$	0	1	\perp
$I9:$	$\{1,3\}$	1	\perp	1	$I23:$	$\{-2\}$	\perp	0	\perp
$I10:$	$\{1,-3\}$	0	\perp	1	$I24:$	$\{-2,3\}$	1	0	\perp
$I11:$	$\{-1\}$	\perp	\perp	0	$I25:$	$\{-2,-3\}$	0	0	\perp
$I12:$	$\{-1,2\}$	\perp	1	0	$I26:$	$\{3\}$	1	\perp	\perp
$I13:$	$\{-1,2,3\}$	1	1	0	$I27:$	$\{-3\}$	0	\perp	\perp
$I14:$	$\{-1,2,-3\}$	0	1	0					

By default, MILE computes the symmetries of every PI in \mathcal{I} , and applying the pruning properties requires careful work. The non-unit property is checked explicitly.

The Fixed Literal Propagation From lemma 4.8, $M_I = M_{I^*}$ for any PI I , hence $\Theta_I = \Theta_{I^*}$. It follows that to compute the LSPs, it suffices to compute only symmetries of canonical PIs. Given the set of models M of Example 4.7,

from I_1 , we may skip over PIs I_2, I_3, I_5, I_6, I_8 and I_{10} of Example 4.12. Indeed, together with the cardinality check, only the symmetries of PIs $I_1, I_9, I_{11}, I_{21}, I_{23}$, and I_{26} need to be computed. In general, fixed literal pruning can save us from calculating symmetry for an exponential number of PIs on the number of fixed literals. If we fix m literals, then $2^m - 1$ supersets may be eliminated from symmetry computation.

To incorporate fixed literal pruning into MILE, for each non-canonical PI I that MILE reaches, I is replaced by its fixed extension I^* . Call this the *context switch* from I to I^* . There are two ways in which I^* may relate to I : either the variable value of some literal in $\sigma(I)$ is smaller than or equal to the anchor of I , or all literals of $\sigma(I)$ have variable values greater than the anchor. In the first case, I^* has already been visited, or it is inconsistent (contains complementary literals). Hence we may prune the current branch of the depth-first traversal. For example, $\{-2\}$ is the set of fixed literals of PI $I_{19} = \{-1, -3\}$ in Example 4.12. As its fixed extension $I_{17} = \{-1, -2, -3\}$ appears earlier in the enumeration, all symmetries associated with I_{19} have already been computed when it is reached in the traversal.

In the second case, the context switch is what allows us to skip over segments of PIs in the MILE enumeration (as in the switch from I_2 to I_9 above). To continue the traversal at I^* , it is important that all non-fixed literals of I are still considered. Such literals are bookmarked by the anchor of I . Take again the context switch from I_2 to I_9 . While we correctly prune PIs I_2, I_3, I_5, I_6 and I_8 , in the process, we also skip PIs I_4 and I_7 whose symmetries should be computed. Through the anchor of I_1 , we may “recover” I_4 and I_7 from I_9 by adding 2 and -2 , respectively. The overall sequence traversed is shown to the below. These are all the canonical PIs. Again, only the symmetries of $I_1, I_9, I_{11}, I_{21}, I_{23}$ and I_{26} are computed after the cardinality check.

$$\begin{array}{llll}
 I_1 : & \{\} & I_7 : & \{1,-2,3\} & I_{17} : & \{-1,-2,-3\} & I_{26} : & \{3\} \\
 I_9 : & \{1,3\} & I_{11} : & \{-1\} & I_{21} : & \{2,3\} & & \\
 I_4 : & \{1,2,3\} & I_{13} : & \{-1,2,3\} & I_{23} : & \{-2\} & &
 \end{array}$$

The above discussion derives Algorithm MILE* of figure 4.6 (for MILE with

Input: A set of models M
Output: The LSPs of M

- 1: $S \leftarrow$ empty stack
- 2: $Res \leftarrow \{\}$
- 3: $S.push(\{\}^*, 0)$
- 4: **while** not $S.empty()$ **do**
- 5: $(I, \eta) \leftarrow S.pop()$
- 6: **if** $|M_I| > 1$ **then**
- 7: $Res \leftarrow Res \cup Sym(I)$
- 8: **for** $i \leftarrow \eta + 1$ to n **do**
- 9: $J \leftarrow I \cup \{i\}; J' \leftarrow I \cup \{-i\}$
- 10: **if** $\forall l \in \sigma(J'), |l| > \eta$ **then**
- 11: $S.push((J'^*, i))$
- 12: **if** $\forall l \in \sigma(J), |l| > \eta$ **then**
- 13: $S.push((J^*, i))$
- 14: **end for**
- 15: **end while**
- 16: Return Res

Figure 4.6: Algorithm MILE*

fixed literal pruning). Each node of the traversal is a pair of a PI and its anchor. The expression $Sym(I)$ in line 7 denotes a function that computes the LSPs from Θ_I .

MILE* dramatically reduces the number of PIs for which symmetries are computed. For the simple case of coloring a path with 6 nodes and 3 colors, there are 96 models. The number of PIs for which symmetries are computed decreases from 235,297 to 180,559 with just the cardinality size, and to 9,307 with the addition of the fixed literal pruning. This is largely because picking a positive literal

for a node will force several literals to be negative.

The Symmetry Composition Property Theorem 4.10 and its corollary tell us that we can use correspondences to find isomorphisms from one local symmetry group to another. This is generally faster than computing the symmetries of the node from scratch, through $Sym()$. Symmetry Composition Pruning (SCP) may thus be stated as follows. Given a sequence $S = I_1, \dots, I_m$ of PIs, suppose (I_i, I_j) is a correspondent with link ϕ . Then either $i < j$ or $i > j$.

1. $i > j$: Since Θ_{I_j} has already been computed, we may compute the LSPs associated with Θ_{I_i} by $\phi^{-1} \circ \Theta_{I_j} \circ \phi$.
2. $i < j$: Compute the LSPs associated with Θ_{I_i} , then propagate them to I_j by $\phi \circ \Theta_{I_i} \circ \phi^{-1}$.

We will qualify each correspondent as a case 1 or case 2.

Remarks:

- Unlike fixed literal pruning, the sequence of PIs is not perturbed, and no interpretations are removed. SCP, in this basic form, simply replaces many calls to $Sym()$ with symmetry composition.
- In general, I_i is the origin of multiple destinations with different links. If (I_i, I_j) is a case 1 correspondent, it suffices to find one link ϕ such that $I_j = \phi(I_i)$. The search for ϕ is thus a key to the success of SCP. We return to this shortly.

We may improve the effects of SCP by combining it with other strategies to remove nodes from the traversal. But care must be taken to ensure that if a node is pruned and its PI is the destination of a correspondent, that the symmetries associated with the PI are available for composition. We discuss two such strategies in the context of MILE.

First, observe that SCP is compatible with Fixed Literal Pruning. Given correspondent (J, J') , either J and J' are both canonical or they are not. Thus we may remove all non-canonical PIs from MILE (or any other enumeration), and the application of symmetry composition pruning remains valid.

A more involved enhancement employs memoization. For each PI I that the traversal visits, store the LSPs created from I and all of its descendants in memory denoted $\text{Pairs}[I]$. Now consider a case 1 correspondent (I_i, I_j) with link ϕ . Instead of computing the LSPs generated from $\phi^{-1} \circ \Theta_{I_j} \circ \phi$, we calculate $\phi(e) = \{\phi(m_1), \phi(m_2)\}$ for all $e = \{m_1, m_2\} \in \text{Pairs}[I_j]$. This effectively computes the LSPs for all descendants of I_i at once, without having to explicitly visit each one. Consequently, the node of I_i and all of its descendants in the traversal may be pruned.

For a case 2 correspondent where the destination follows the origin in MILE, memoization is less effective since the destination I_j may be pruned later in the traversal. Any composition calculation associated with I_j would be for naught. Moreover, for each base that links I_i to a different destination, the composition would need to be performed. This incurs a prohibitive space requirement on the implementation with a comparatively small time benefit. One exception is when the base of the correspondent is the empty set (i.e., the link is a global symmetry). As global symmetries are invariant to PIs: (I_i, I_j) is a correspondent iff (I_j, I_i) is a correspondent, it is not even necessary to store $\text{Pairs}[I_j]$. We simply add LSPs when the traversal reaches I_i , and prune I_j when it is reached knowing that its LSPs have already been computed.

As mentioned, a key to the success of the SCP (and the above enhancement) is in finding a good link. In the experiments, we will obtain links via several different strategies. **Fast:** The link is a symmetry of the parent node of I_i . **Global:** The link is a global symmetry. We also experiment with combinations of fast and global. **Deep:** From the parent of I_i , we compose in order symmetries of each of I_i 's ancestors to find a list of correspondents $\{(I_i, I_1), (I_1, I_2), \dots, (I_{n-1}, I_n)\}$ which

can be used to iteratively find $\text{Pairs}[I_i]$.

The **Deep** strategy is a novel approach to pruning. Local symmetry is typically used to prune in isolation, ignoring the relationships between related partial interpretations. This may be due in part to the fact that the local symmetries of related PIs do not form an obvious group. However, we have had success with a simple exhaustive-search method. This requires exponential space in the worst case. But such a case indicates the existence of a large number of symmetries with the potential for much pruning. Hence another strategy such as **Fast** would be an adequate substitute.

A related completeness complication occurs when a case 1 correspondent (I_i, I_j) is found, but I_j has been pruned previously. Thus $\text{Pairs}[I_j]$ does not exist. The straightforward solution is to default to compute $\text{Sym}()$ for I_i and its descendants. A more involved but worthwhile approach that we have adopted is to find the first unpruned ancestor $I_{j'}$ of I_j , and generate a link that could be used to find a second case 1 correspondent (I_j, I_k) . LSPs of $\text{Pairs}[I_i]$ may then be computed by calculating $\text{Pairs}[I_j]$ via $\text{Pairs}[I_k]$. The process may be repeated if I_k has also been pruned.

Experiments And Results

When choosing problems for experiments, we could not consider the usual SAT benchmarks since they are designed to be computationally challenging to model finding programs. We need the set of all models of the theory to study our algorithms. It follows that the problems discussed here are easy instances by normal SAT standards. They have, however, non-trivial numbers of symmetries.

The next table shows a selection of problem classes and their characteristics. Experimental results of these problems are representative of the wider set (of 20+ classes) of problems that we have studied. Problems 4x4 Sqs and 5x5 Sqs are instances of All-Squares. The 3-coloring problems are on paths of 7, 8 and 9 nodes (i.e., P_7, P_8 and P_9). Rand is representative of a 4096 models sampled

without replacement from all assignments with 13 variables. This was included as an adversary for our pruning strategies since there are few fixed literals and a small number of symmetries. 4x4 Recs (rectangles) are similar to All-Squares, but has very different traversal characteristics.

For each problem, the set V is the variable set from which models are formed, and the size n of V gives us the size of the space, \mathcal{I} , on which the traversal algorithms operate. The set \mathcal{I} ranges in size from 1.59×10^6 to 2.39×10^{23} . Loc Pairs shows the number of symmetric pairs of models (out of a possible $\binom{M}{2}$ number of pairs). For reference, Gl. Sym Pairs shows the number of pairs of models that are globally symmetric.

Problem	MILE		Characteristics			
	Sym Calls	Time	Models	$ V $	Loc Pairs	Gl. Pairs
7-Path	1124053	>1000	192	21	8298	912
8-Path	721066	>1000	384	24	28314	1968
9-Path	472008	>1000	768	27	98910	3936
4x4Sqs	95919	31.5	30	16	286	58
5x5Sqs	1804172	>1000	55	25	858	110
4x4Recs	360174	70	100	16	1233	241
Rand	1473837	533	4096	13	261173	0

Problem	MILE*			
	Sym Calls	% Prune	Time	Speedup
7-Path	40489	>96.4%	33.6	>29.8x
8-Path	175705	>75.6%	224	>4.46x
9-Path	514786	-	>1000	-
4x4 Sqs	66009	31.1%	22.2	1.42x
5x5 Sqs	1612904	-	>1000	-
4x4 Recs	72142	80.0%	11.7	6.00x
Rand	1375306	6.68%	494	1.08x

Problem	Global SCP				Problem	Fast SCP			
	Sym Calls	% Prune	Time	Speedup		Sym Calls	% Prune	Time	Speedup
7-Path	3557	99.4%	4.43	>226x	7-Path	6278	>99.4%	4.26	>235x
8-Path	14983	>97.9%	24.2	>41.2x	8-Path	26296	>96.4%	28.33	>35.3x
9-Path	64329	>86.4%	151.6	>6.60x	9-Path	108485	>77.0%	229.3	>4.36x
4x4 Sqs	8625	>91.0%	4.73	6.65x	4x4 Sqs	865	99.1%	1.01	31.2x
5x5 Sqs	1138458	-	>1000	-	5x5 Sqs	174336	>90.3%	176.8	>5.66x
4x4 Recs	9453	97.4%	3.22	21.7x	4x4 Recs	26388	92.7%	6.13	11.4x
Rand	1375306	6.68%	546	0.975x	Rand	1210131	17.8%	602.3	0.885x

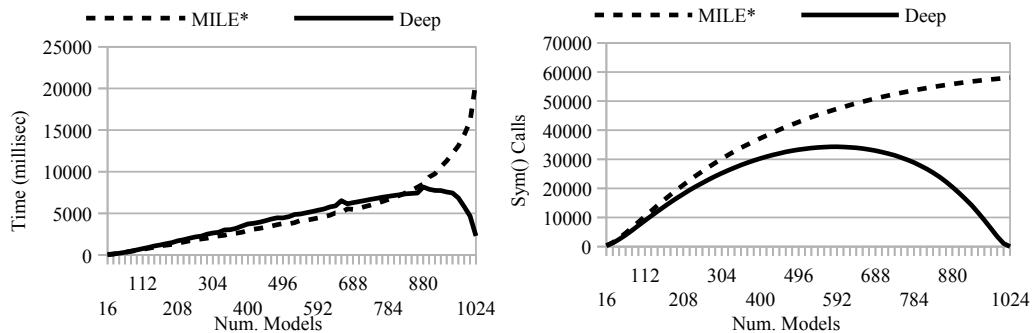


Figure 4.7: Performance of MILE* and Deep Local Symmetry SCP on random sets of models

	Fast + Global SCP					Deep SCP			
7-Path	2879	>99.7%	3.67	>272x	7-Path	2879	>99.7%	2.89	>346x
8-Path	11521	>98.4%	20.4	>49.0x	8-Path	11520	>98.4%	15.2	>65.7x
9-Path	47181	>90.0%	163.5	>6.11x	9-Path	47173	>90.0%	117.9	>8.48x
4x4 Sqs	499	99.5%	1.08	29.1x	4x4 Sqs	342	99.6%	0.85	37.0x
5x5 Sqs	69886	>96.1%	80.8	>12.4x	5x5 Sqs	38984	>97.8%	47.3	>21.2x
4x4 Recs	7187	98.0%	3.41	20.5x	4x4 Recs	6564	98.2%	2.5	28.0x
Rand	1210131	17.8%	631.2	0.844x	Rand	1207462	18.1%	596.3	0.894x

We compare the number of calls to $Sym()$ and the time that it takes for the algorithms to complete. MILE with cardinality check is the baseline against which various heuristics are tested. Results are shown in table above. Time-out is set to 1000 seconds. **Global** SCP employs case 2 correspondents exclusively, while for other SCP based methods, only case 1 correspondents are used.

MILE* is slower than all other complete methods that employ some form of SCP except on Rand problems, which have a small number of symmetries. Inspecting the performance of various SCP strategies, **Deep** is the fastest overall, only slower than MILE* on Rand. Of note is that **Global** performs particularly badly on the All-Squares problem.

Figure 4.7 shows how MILE* and Deep Local Symmetry SCP behave for random sets of models of different size. We ignore global symmetry since it acts similarly but slower than MILE* except at the very highest densities where there

Input: A set of models M
Output: An approximate diversity graph of M

- 1: $S \leftarrow \{\}$
- 2: $Res \leftarrow \{\}$
- 3: **for** each pair $m_1, m_2 \in M, m_1 \neq m_2$ **do**
- 4: **if** $\alpha(m_1, m_2) \notin S$ **then**
- 5: $S \leftarrow S \cup \{\alpha(m_1, m_2)\}$
- 6: $Res \leftarrow Res \cup Sym(M_{\alpha(m_1, m_2)})$
- 7: **end for**
- 8: Return Res

Figure 4.8: Algorithm Agree

are many global symmetries available for pruning. Fast Local Symmetry SCP follows Deep closely. For most of the experiments, the time increases linearly w.r.t the number of models. This shows that our literal-centric symmetry-finding approach effectively reduces the overhead of more models on an invariant set of variables. When the number of models increases enough, the local symmetry groups contain many more elements, allowing deep local symmetry to prune significantly earlier in the search, but it harms MILE* since it takes longer to find all of the generators.

4.2.2 Other Algorithms

Offline, Approximate.

Theorem 4.3 shows that an agreement graph has at least $|M| - 1$ edges. This is a linear approximation of the diversity graph for full local symmetry that has the potential for encoding a significant number of edges in the full graph. Algorithm Agree in Figure 4.8 presents an approximate algorithm. The algorithm requires $O(|M|^2)$ symmetry calculations, which is substantially better than the exponential

number of symmetry calculations for the complete algorithm. A linear filter to compute agreement sets will only require $O(|V||M|^3)$ literal comparisons in total. Hence we would expect the runtime to increase much more slowly compared to the complete algorithms.

An example of symmetries that aren't agreement symmetries is the 3-Coloring of a graph with no nontrivial automorphism. Then the only global symmetries are permutations of the colors. Let θ be a global symmetry that permutes all of the colors. Since every node has exactly one color, it only stabilizes PIs that have no solution consistent with them. Therefore, θ is only expressed as a global symmetry. Indeed, augmenting agreement symmetry with global symmetry can increase the accuracy of our approximation with a relatively low cost to time.

Online, Approximate.

A diversity graph represents all possible diverse sets, but finding a single diverse set may suffice and can be significantly easier to compute. Moreover, an offline algorithm assumes possession of all models, which may involve an unacceptably long wait for the solver.

Figure 4.9 gives an online, anytime algorithm that takes a theory F and directly builds an approximate diverse set Div without constructing the diversity graph. The set Div is found incrementally by repeated calls to a black-box solver. A solution m produced by the solver is added to Div if it is diverse with respect to models already in Div . Conversely, m modifies F by conjoining F with the negation $\neg m$ to eliminate m from the set of possible solutions in subsequent iterations.

To check that m is diverse with respect to every model $m_i \in Div$ involves an exponential number of symmetry calculations. To make the process tractable, we approximate by only considering the symmetries of $\alpha(m, m_i)$. As we don't have the full set of models a priori, an alternate representation is needed to calculate symmetries. This is captured in the procedure *getRep()* where several representation choices exist. The straightforward approach is to use the formula $F \wedge \alpha(m, m_i)$

```

Input: Theory  $F$ 
Output: An approximate distinctive set  $Div$ 
1:  $Div \leftarrow \{\}$ 
2: while  $F$  is satisfiable do
3:    $m \leftarrow solve(F)$  // from a black-box solver
4:    $F \leftarrow F \wedge \neg m$ 
5:    $add \leftarrow true$ 
6:   for  $m_i \in Div$  do
7:      $F' \leftarrow getRep(\alpha(m, m_i))$ 
8:      $\Theta = Sym(F')$ 
9:      $F \leftarrow F \wedge brkCl(\Theta)$ 
10:    if  $\exists \theta \in \Theta$  s.t.  $\theta(m) = m_i$  then
11:       $add = false$ ;
12:      break;
13:    end for
14:    if  $add$  then
15:       $Div \leftarrow Div \cup \{m\}$ 
16:    end while
17: Return  $Div$ 

```

Figure 4.9: Algorithm Online.

(simplified through rules such as subsumption checking and unit propagation). We call this the syntax-only representation. It is simple and fast, but it may not represent all symmetries. Alternatively, it is sometimes feasible to find all models of the simplified formula $F \wedge \alpha(m, m_i)$ even if it isn't feasible to find them for the original formula. This guarantees finding all semantic symmetries with respect to $\alpha(m, m_1)$. This is the strategy that we employ, which we call the semantic-symmetry-first representation. But if a timeout occurs, $getRep()$ defaults to using the formula itself.

This algorithm performs at most $O(|M||Div|)$ automorphism calculations, gen-

erally faster than Algorithm 4.8 since $|Div| \ll |M|$. And as it is an anytime algorithm, it could be halted to return a smaller diverse set at any time. To improve the accuracy of the output diverse set, we check global symmetries as well. To improve its efficiency, we may constrain F by adding symmetry breaking clauses using the procedure $brkCl()$. This can improve solver speed and reduces the number of models for the formula. The latter decreases the number of times through the **while** loop.

The simplest symmetry breaking clauses are for global symmetries, but they may be easily adapted for local symmetries as well. If C is a symmetry breaking clause for a symmetry on PI I , then the clause $(I \rightarrow C) = (\neg I \vee C)$ is the local symmetry breaking clause for I . While the use of symmetry breaking clauses can introduce a bias (e.g., preferring lex-leaders), in practice they increase the quality of the diverse sets since any model must pass more stringent constraints before it is added to Div .

Alternate Representations The choice of representation is very important for the online algorithm. A more thorough analysis of what makes certain representations better or worse could lead to great improvements to online algorithm's performance.

The most naive representation is a DNF where each model is fully represented by a single clause. The symmetries of this representation are obviously the semantic symmetries of the underlying theory. Similarly we could use *prime implicants*. A prime implicant is a minimal conjunctive representation of some set of models. If a prime implicant M' is implied by model M , we say that M' covers M . The set of prime implicants could be much smaller than the full set of models and so may be a better representation in some cases.

Theorem 4.13. *Let θ be a semantic symmetry, and M, N models where $\theta(M) = N$. Let $M' \subseteq M$ be a prime implicant, then there exists a prime implicant $N' \subseteq N$ such that $\theta(M') = N'$.*

Proof. Since M' is a prime implicant, it covers the models M, M_0, M_1, \dots, M_n , so $\theta(M')$ covers models $\theta(M) = N, \theta(M_0), \dots, \theta(M_n)$. So $\theta(M') = N'$ is an implicant. Assume it is not prime. Then there exists $N'' \subset N'$ that is prime, so $\theta^{-1}(N'') \subset M'$ and is an implicant, and smaller than M' , contrary to the assertion that M' is prime. \square

If we wish to find a CNF representation, the set of *prime implicates* can also be used.

Theorem 4.14. *Given a CNF formula F , if there exists a semantic symmetry θ then for every clause $C \in F$ at least one of the following must be true:*

1. $\theta(C) \in F$
2. $F \implies \theta(C)$

Proof. If θ is a syntactic symmetry, then (1) is always true for all clauses. Otherwise, there is some clause where (1) is not true. However, since θ is a semantic symmetry, that means that $\theta(F) \equiv F$. Since F is a CNF formula, it means that $\theta(C) \wedge F \equiv F$, which means that $\theta(C)$ is never false when F is true, so $F \implies \theta(C)$. \square

In particular that means that $\theta(C)$ is either a resolvent of F or is subsumed by a resolvent of F . So we can find all of the resolvents of F , remove all subsumed clauses and let that resulting formula be our new representation.

The memory requirements of the output is fixed-parameter tractable if the only output clauses are of size less than or equal to the size of the largest clause in F . This is because for any clause C , we only need to add every $\theta(C) \notin F$. Given that the size of the largest clause is some constant n , and the number of variables is v our representation will be no larger than $\binom{v}{n}$, which is bounded above by the number of permutations v^n . This is not a tight bound because the

only time there would be all combinations of clauses of a certain size is when the CNF is a tautology or a contradiction which can both be represented in linear or constant space (depending if you want to record which variables were used in the original formula or not). The downside is we may have an exponential intermediate representation.

This does not imply a cubic bound on the number of variables by converting a CNF to 3-SAT because the 3-SAT conversion process does not preserve symmetries. For example, the formula $\{\{a, b, c, d\}\}$ would be converted to the 3-CNF $\{\{a, b, e\}, \{\neg e, c, d\}\}$ which would exclude such symmetries as (a, d) which is valid on the former but not the latter.

This does lead to an approximate intermediate representation that is no worse, and sometimes better than the original representation at representing semantic symmetries. In particular, instead of finding all prime implicants, we find a set prime implicants whose conjunction is equivalent to the original formula. We find this cover by removing redundant literals from clauses.

Definition 4.15. *For CNF formula F , a clause C has redundant information if for some $l \in C$, $F \models (C - \{l\})$. Otherwise, a clause C where $F \vdash C$ is irredundant, or contains no redundant information.*

This property basically means that v can be removed from the clause without effecting equivalence. This is tested by seeing if $F \bigwedge_{u \in C, u \neq v} \neg u$ is a contradiction.

This means the speed of modern SAT solvers can be leveraged to help us find a formula that contains no redundant information. We can iteratively test the variable property until there is no more redundant information there. Since F implies no clauses that are subsets of the current clauses, they are all prime implicants. If two or more clauses are equal, we keep one, and remove the rest.

Since a clause can have several redundant variables but removing each single redundant variable can create an redundant clause, one clause may turn into many clauses. This potentially means the intermediate form is exponential, but it is

never larger than the full implicate form and is usually smaller. Hence, it is still fixed-parameter tractable.

A simple example of when this intermediate representation helps find new symmetries is $\{\{a, c\}, \{a, \neg c\}, \{b\}\}$. Here, (ab) is a semantic symmetry, but not a syntax symmetry since $\{a, c\}, \{a, \neg c\} \models \{a\}$. So c and $\neg c$ are redundant in their respective clauses. So our new representation is $\{\{a\}, \{b\}\}$, which contains (ab) as a syntax symmetry.

It is possible that this new representation misses some semantic symmetries. For example $\{\{a, b\}, \{a, k\}, \{\neg k, \neg c\}, \{\neg a, j\}, \{\neg j, c\}\}$ does not have any syntactic symmetry and cannot be simplified any further. However, this representation does help in cases where a formula has many more clauses than necessary, as we can see from the following figure, when random 3-Sat problems become over-constrained, removing redundant information helps us find additional symmetries that would have not been found otherwise. In our tests, we generated random 3-Sat problems with 100 variables, and we only tested symmetries on satisfiable instances, since all unsatisfiable instances have a symmetry. It is also a good intermediate representation because it can reduce the number of intermediate resolution steps we have to take to get the full prime implicate form.

To prove this intermediate representation preserves symmetries, we first prove the following lemma:

Lemma 4.16. *If clause $C \in F$ contains redundant information, and $C' \subset C$ contains no redundant information then given semantic symmetry θ , $\theta(C)$ contains redundant information and $\theta(C')$ contains no redundant information.*

Proof. We know $\theta(C')$ is irredundant w.r.t $\theta(F)$. Since θ is a semantic symmetry, $\theta(F) \equiv F$, and $F \models \theta(C')$. So $\theta(C)$ contains redundant information and $\theta(C')$ contains no redundant information. \square

Theorem 4.17. *If F is a non-tautologous CNF formula with syntactic symmetry θ , and F' is F where all redundant clauses of F are replaced by their irredundant*

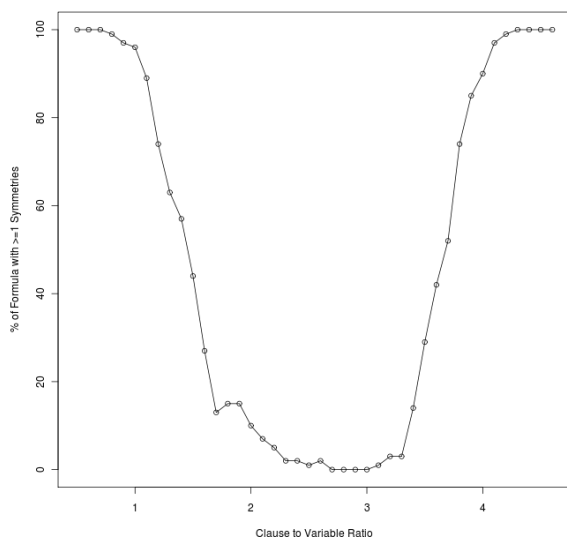


Figure 4.10: Percent Symmetries For Random 3-SAT problems After Removing Redundant Information on Various Clause/Variable ratios

subsets, then θ is also a syntactic symmetry on F' .

Proof. If F is a contradiction, F' derives the empty set, which obviously contains all syntactic symmetries.

So F is satisfiable, but not a tautology. This means that every clause of F is a non-empty clause in F' . Let C' be some clause in F' , we want to prove that $\theta(C') \in F'$.

C' is either a clause in F , or is a subset of a clause of F , C . If C' is a clause of F , then $\theta(C')$ is a clause of F , since θ is a syntax symmetry. Otherwise, C contained redundant information, and have replaced it with at least C' . From 4.16 we know that we have replaced $\theta(C)$ with at least $\theta(C')$ as well. Since $\theta(C') \in F$, θ is a valid syntax symmetry for C' in F' . Since this is true for all $C' \in F'$, we know θ is a valid syntax symmetry on F' . \square

A potential improvement that was not used in the original algorithm because it's

possible to eliminate syntactic symmetries is removing *redundant clauses*, where $F - \{C\} \models C$. This means that the information contained in C is redundant, so C can be removed. This is tested by checking if $((F - \{C\}) \bigwedge_{u \in C} \neg u)$ is a contradiction. Applying this to the formula $\{\{a, b\}, \{a, k\}, \{\neg k, c\}, \{a, j\}, \{\neg j, b\}\}$ can open up new symmetries.

The only known intermediate form that preserves existing syntactic symmetries but also does not increase the number of clauses can be computed as follows: if there exists two clauses whose resolvent subsumes both clauses, then we can replace those clauses with their resolvent. However, it turns out this requires two clauses that are the same size and differ on a single literal, where one is the positive literal of a variable and the other is the negative literal. This is not expected to be a frequent case.

Experimental Results

We performed experiments on over 40 problems taken from hardware verification [37][13]³, SATLIB [28], and other problems such as N-Queens (with the most typical representation). Note that challenging SAT problems do not necessarily equate to challenging LSD problems. Indeed, they are usually at odds: a difficult SAT instance with very few solutions would likely be an easy diversity problem, while easy SAT instances with large number of models challenge our algorithms. Most of the problems that we tested on have model counts of 10^6 to 10^{300} and above. Our goal was to find diverse sets of size 50. We report a representative subset of the experiments.

The tests are on the Oracle Java 7 server JVM using the SAT4J solver, modified to select a random phase for each variable that it chooses. Each time we compare a candidate to an existing member of a diverse set, we attempt to calculate a semantic representation by solving the constrained formula within 750 ms. If there

³www.cs.rice.edu/CS/Verification/Projects/UniGen/Benchmarks

Name	Semantic-Symmetry-First Representation							Syntax-Only Representation		
	# Cl	V	Time	Cand. Time	Div	# Cands	Sem. Timeout	Time	Div	# Cands
25 Queens	24850	625	1000	425	2	7188	0	5	50	50
50 Queens	203450	2500	1000	978	1	337	0	133	50	50
15x15 AFS	138251	225	1000	<1	14	160	0.14	1000	37	101
mo_prop_9_12912_2	8367	3665	937	3	50	50	1	5	50	50
squaring7	5837	1628	1000	6	37	2677	0.01	4	50	50
squaring8	3642	1101	522	2	50	1317	0.01	2	50	51
squaring10	3632	1099	823	3	50	2098	0.01	2	50	50
enqueueSeqSK	58515	16466	599	18	50	52	0.23	18	50	50
karatsuba	82417	19594	1000	991	5	93	0	154	50	50
LoginService2	41411	11511	1000	<1	50	50	0.91	10	50	50
s953a_3_2	1297	515	20	<1	50	192	0	1	50	81
s1196a_7_4	1881	708	107	<1	50	84	0	1	50	52
scenarios_llreverse	257657	63797	1000	958	12	55	0	726	50	62
sort	49611	12125	1000	109	45	78	0.1	20	50	50
tutorial3_4_31	2598178	486193	1000	511	25	26	0.64	791	50	50
uf250-01	1065	250	93	45	50	50	0	29	50	54
uf250-02	1065	250	1000	38	23	2092	0	1000	26	142997
flat200-1	2237	600	1000	1	46	88	0.3	2	50	53
flat200-2	2237	600	1000	10	35	134	0.23	3	50	142
logistics.a	6718	828	1000	1	47	54	0.9	1	50	52
logistics.b	7301	843	937	1	50	50	1	1	50	50
logistics.c	10719	1141	937	1	50	50	1	2	50	50
logistics.d	21991	4713	941	1	50	50	0.96	4	50	50
bw_large.d	131973	6325	159	154	1	106	0	12	43	106
cl_schedule	3556393	18656	1000	<1	35	57	1	192	50	50
bmc-ibm-1	55870	9685	943	13	50	50	1	9	50	50
bmc-ibm-2	11683	2810	828	<1	50	51	0.17	2	50	50
bmc-ibm-3	72106	14930	1000	1	35	86	1	17	50	123
bmc-ibm-4	139716	28161	946	1	50	50	1	23	50	50
bmc-galileo-8	294821	58074	≈1000	4	50	52	1	67	50	57
bmc-galileo-9	326999	63624	100	7	49	51	1	54	50	52

Table 4.1: Results of running online algorithm with no symmetry breaking using semantic-symmetry-first representation for *getRep()*, compared with relative success using syntactic-only representation. All times are in seconds.

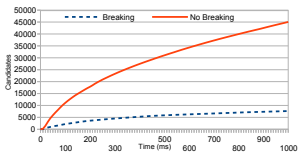
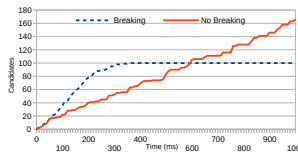
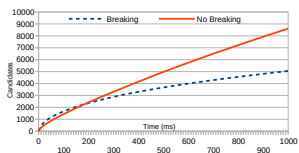
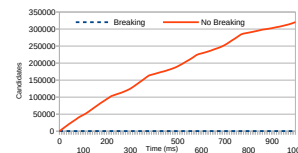
Name	UF250 (100 instances)	15x15 AFS (Semantic)
# Cands.		
Median Time without breaking	285s	1000s
$ Div $ without breaking	71.2	14
Median Time with breaking	158s	211s
$ Div $ with breaking	71.1	13.6
# Breaking Clauses	202589	5991
Name	Flat200 (100 instances)	logistics.a (Deterministic)
# Cands.		
Median Time without breaking	10s	1000s
$ Div $ without breaking	91.8	13
Median Time with breaking	533s	4s
$ Div $ with breaking	66.4	100
# Breaking Clauses	581994	18167

Figure 4.11: Comparing breaking and no breaking to obtain up to 100 models with 1000 second timeout. All use syntax breaking and non-deterministic phase selection, except as noted.

is a timeout rate of 90% or above, then syntactic symmetry is used. The overall timeout time is 1000 seconds.

For most problems, we begin with a theory. But since the offline algorithms assume that all models are available a priori, to compare offline to online algorithms, we need instances that are easy to solve by normal SAT standards, but that have non-trivial numbers of symmetries nevertheless.

Offline algorithms work with a models representation, while online algorithms work on a CNF representation. So when comparing offline and online algorithms

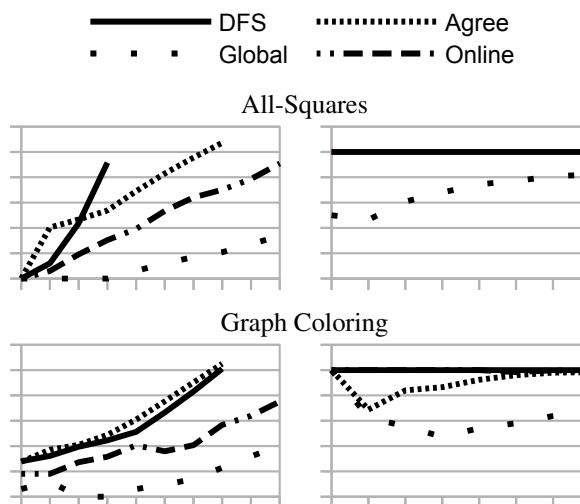


Figure 4.12: Speed in logscale milliseconds, and percentage of locally asymmetric pairs in maximal diverse sets over problems of increasing size.

in Experiment 1, we use a slightly modified version of the online algorithm that works on models instead of a CNF formula.

Offline Algorithm Comparisons. We have conducted an extensive set of experiments on the speed of various algorithms, the effectiveness of pruning strategies, and on the accuracy for determining diversity by approximation algorithms. Figure 4.12 shows two representative results of the offline, agreement based approximation (Agree) and the online, symmetry breaking based approximation (Online). We also include the results of global symmetry diversity for reference (Global), and DFS is the full local symmetry computation, serving as the baseline.

Besides pruning by the non-unit property and fixed extensions shown in Figure 4.6, we also employ the symmetry composition property in the experiments. For Algorithm Online, the time reported includes the time spent in the solver. Symmetry breaking clauses are used, but as fully breaking symmetry can require an exponential number of clauses, we only break the generators. To fill in the gaps, we also break any symmetry we find to reject models that have been determined

not diverse relative to *Div*.

The top two graphs compare AFS data for grid sizes 2×2 to 11×11 . The bottom two graphs are for Path-Coloring between 2-11 nodes. For each pair, the left graph shows speed in logscale milliseconds, and the right graph shows accuracy, defined as the percentage of distinct pairs of models in a maximal diverse set that also diverse under full local symmetry.

Differences in efficiency greatly depend on the structure of the problem. The time to compute agreement symmetry diversity grows much slower than computing full local symmetry diversity for AFS, but they are similar for coloring paths. In path coloring, much stronger symmetries exist with smaller PIs, thereby allowing for significant pruning. The online algorithm is an order of magnitude or more faster than either DFS or Agree. Global symmetry is the fastest of all the algorithms since it requires a single automorphism calculation, but it achieves poor accuracy. Indeed, the accuracy of global symmetry is not much better than picking models randomly. On the other hand, the online algorithm gives diverse sets that are nearly identical to local symmetry diverse sets. Agreement symmetry is accurate for AFS, but drops off for some smaller Path-Coloring problems due in part to the absence of global symmetry. Combining agreement and global symmetry yields results similar to the online algorithm.

All in all, the full local symmetry algorithm (Algorithm 4.6) can solve 5x5 AFS within the timeout, but not any of the tested problems. Algorithm 4.8 performs significantly better, and can solve upto 12x12 AFS as well as `bw_large.d` with 106 models.

Online Algorithm Comparisons. Table 4.1 records results of computing diverse sets of the online algorithm using the semantic-symmetry-first representation strategy that we discussed, as well as some comparisons using syntax-only representation. All times are in seconds, the candidate time is the time the solver takes to find candidate diverse models, and the semantic timeout is the percentage of times we had to resort to syntax symmetries only.

As one would expect, results vary depending on the structure of the problem. Different elements are at play in the overall efficiency: the speed of the SAT solver for finding candidates and computing representations, and calculating the symmetries on those representations. Since there can be many models, and each model as a DNF clause is much larger than average clause in the original formula, symmetries using semantic representations is often much slower than calculating symmetries on the syntax.

Even though computing semantic symmetries may be time consuming, there is a striking increase in selectivity (the number of candidate models accepted as a member of the diverse set). For 25 Queens, every candidate is accepted with syntax symmetry (50 out of 50), but less than .1% of the candidates when using semantic symmetry (2 out of 7188). This suggests an important topic of future work, namely to find syntax of formulas that more closely reflect its underlying semantics.

The results of Table 4.1 shows cases where the number of candidates tested far outstrips the number of diverse models found. Thus a large amount of computing is performed on finding and testing candidates that can be easily dismissed. Figure 4.11 shows that symmetry breaking clauses can help. It reduces the time that it takes to find 100 diverse models while testing far fewer candidates. Employing a deterministic phase selection for logistics.a causes the solver to find many similar models without symmetry breaking clauses, but with symmetry breaking clauses the solver is much more likely to return candidates that are diverse. This implies that symmetry breaking clauses can help to expand the range of solver strategies practical for LSD.

But symmetry-breaking clauses are not without issues. Their size can dominate the size of the original formula, increasing memory usage and possibly slowing down the SAT solver. Occasionally, they make it harder to find diverse models, for example the Flat200 graph coloring problems. Since the clauses force candidates into certain configurations, candidates can end up being more similar than

if we did not add symmetry-breaking clauses. In some sense it forces a notion of similarity that is stronger than local symmetry.

4.3 Comparing Discrimination

We can now assess how combinations of diversity discriminate by evaluating which methods are better or worse at mimicking the structure of the other methods. The first evaluation uses two diversity methods: a creation method and rejection method. The *rejection percentage* is defined as the percentage of diverse sets created by the creation method that are also diverse sets of the rejection method. We create random diverse sets by uniformly sampling valid diverse models and adding them to the diverse set until no such model exists. Our experiments create 10,000 different diverse sets for this measure.

Rejection percentage is biased towards diversity measures that lead to small diverse sets since they reject larger sets. While this is fine for pure discrimination, it's also helpful to see how hard it is to create such a set. For example, a method that always selects a single model will reject most diverse sets, but we could get the same effect by just removing models from larger diverse sets.

To account for this bias, we will also compute the *rejection ratio* or *rr*. To calculate *rr*, we take modify the original rejection percentage with a new rejection percentage. The rejection methods remain the same, but there is a new creation method creates a new diverse set that is the same size as the one the original method created. The models to add are chosen by uniformly sampling from the set of models. We apply Laplacian Correction to ensure that a sensible ratio is obtained. Thus, a method with a $rr < 1$ does better than random at creating sets that are accepted by the second method, while a method with $rr > 1$ does worse than random. The *rejection ratio* is similar to a correlation measure used in data mining and pattern analysis known as the *lift*.

Since diversity is highly context-sensitive, a high rejection percentage or rejec-

	D	DG	DGS	A	DGA	DGSA
Mean Size:	7.71	6.87	4.93	1.61	1.35	1.35

Table 4.2: 8-Queens Set Size

		Rejection Methods					
		D	DG	DGS	A	DGA	DGSA
Creation Methods	D	-	0.92	1.00	1.00	1.00	1.00
	DG	0.00	-	< 1.00	1.00	1.00	1.00
	DGS	0.93	0.93	-	1.00	1.00	1.00
	A	0.067	0.067	0.10	-	0.33	0.37
	DGA	0.00	0.00	0.18	0.00	-	0.35
	DGSA	0.12	0.12	0.00	0.00	0.35	-

Table 4.3: 8-Queens Rejection Percentage

tion ratio does not mean a method is worse than another method, but that the structure of its diverse sets is significantly different.

The tested methods are: distance only (**D**), distance and global symmetry (**DG**), distance and global symmetry with shortest-path correction (**DGS**), agreement symmetry (**A**), distance and global and agreement symmetry (**DGA**), and finally distance, global symmetry, and agreement symmetry with shortest-path correction (**DGSA**). For the final method, we apply shortest-path correction *before* adding agreement symmetry information. If we apply it after adding all the information, our method will almost always return a trivial diverse set.

8-Queens

The pure distance method is soundly rejected by all the other methods tested, while it rejects methods that use shortest-path correction. **DGSA** does worse than random in this regard. The distance plus global symmetry method rejects the

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.92	1.00	1.00	1.00	1.00
	DG	0.00	-	1.00	1.00	1.00	1.00
	DGS	0.95	0.94	-	1.00	1.00	1.00
	A	0.34	0.29	0.43	-	0.53	0.61
	DGA	0.00	0.00	1.26	0.00	-	1.01
	DGSA	1.09	0.96	0.00	0.00	1.00	-

Table 4.4: 8-Queens Rejection Ratio

other methods in a very similar manner to the distance method, but since the global symmetry increases discriminativeness, the rejection ratios are somewhat lower even though the rejection percentages are about the same. This shows that agreement symmetry covers global symmetry in this case.

One pattern we will see throughout the tests is that distance plus global symmetry with shortest-path correction rejects almost all regular distance plus global symmetry methods. We also see this behavior in the interplay between **DGA** and **DGSA**. **DGA** does very well against all methods except those that use shortest path correction, where it does worse than random. On the other hand, **DGSA** does poorly against all methods that do not use shortest path correction. From this we can conclude that shortest-path correction significantly changes the structure of the dissimilarity graph.

Lastly, we point out that agreement symmetry works significantly better than random against all other methods, even when we adjust for the small size of its sets.

	D	DG	DGS	A	DGA	DGSA
Mean Size:	3.22	3.23	3.87	4.46	3.00	3.32

Table 4.5: Path Coloring Set Size

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.26	0.89	0.60	0.67	0.94
	DG	0.00	-	0.87	0.56	0.56	0.93
	DGS	0.98	0.98	-	0.92	0.98	0.92
	A	0.96	0.97	0.98	-	0.97	0.98
	DGA	0.00	0.00	0.85	0.00	-	0.85
	DGSA	0.91	0.91	0	0	0.91	-

Table 4.6: Path Coloring Rejection Percentage

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.27	0.95	0.66	0.68	0.96
	DG	0.00	-	0.93	0.61	0.56	0.94
	DGS	1.01	1.01	-	0.97	1.07	0.94
	A	0.96	0.97	0.99	-	0.97	0.98
	DGA	0.00	0.00	0.93	0.00	-	0.86
	DGSA	0.97	0.96	0.00	0.00	0.94	-

Table 4.7: Path Coloring Rejection Ratio

	D	DG	DGS	A	DGA	DGSA
Mean Size:	7.57	7.34	5.46	14.67	4.65	4.47

Table 4.8: Space Touching Cycles Set Size

Path Coloring

A unique aspect of the path coloring problem is the relative similarity of the sizes of the sets created by each method. Unlike 8-Queens, pure distance works well against methods that don't use shortest-path correction while agreement symmetry does uniformly poorly unless it is augmented by distance. We also note that unlike 8-Queens, agreement symmetry also obtains larger sets than the distance-based methods.

Space Touching Cycles

Space Touching Cycles is a problem that works on an $n \times n$ grid. Each grid cell is either off or on, which corresponds to a single propositional variable being false or true, respectively. If a grid cell is on, exactly two of its horizontal or vertical (not diagonal) neighbors must be on, and if a grid cell is off at least one of its horizontal or vertical neighbors must be on. The first rule means that the “on” portions must represent cycles, while the second rule means that a cycle must touch every space in the grid. It was created as a problem that was easy to visualize, but whose full structure is difficult to determine. We tested it on an 8×8 grid. Unlike many problems, where agreement symmetry seems to subsume global symmetry, on this problem there are problems that are globally symmetric but not agreement symmetric.

Again, we point out that agreement symmetry creates sets that are significantly larger than distance-based methods. Every method does very badly against every other method that is not its subset. Of all of them, DGSA performs the best overall.

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.45	<1.00	<1.00	1.00	1.00
	DG	0.00	-	1.00	<1.00	1.00	1.00
	DGS	0.96	0.96	-	0.98	<1.00	0.98
	A	1.00	1.00	1.00	-	1.00	1.00
	DGA	0.00	0.00	0.96	0.00	-	0.96
	DGSA	0.86	0.86	0.00	0.00	0.96	-

Table 4.9: Space Touching Cycles Rejection Percentage

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.00	0.45	1.00	1.00	1.00
	DG	0.00	-	1.00	1.00	1.00	1.00
	DGS	0.96	0.96	-	1.02	1.00	0.98
	A	1.00	1.00	1.00	-	1.00	1.00
	DGA	0.00	0.00	0.97	0.00	-	0.96
	DGSA	0.88	0.88	0.00	0.00	0.96	-

Table 4.10: Space Touching Cycles Rejection Ratio

	D	DG	DGS	A	DGA	DGSA
Mean Size:	4.97	4.73	6.11	3.61	2.65	2.25

Table 4.11: Cycle Matching Set Size

		Rejection Methods					
		D	DG	DGS	A	DGA	DGSA
Creation Methods	D	-	0.54	0.96	<1.00	1.00	1.00
	DG	0.00	-	0.96	<1.00	1.00	1.00
	DGS	<1.00	<1.00	-	1.00	1.00	1.00
	A	0.80	0.81	0.74	-	0.97	0.97
	DGA	0.00	0.00	0.32	0.00	-	0.90
	DGSA	0.20	0.20	0	0	0.80	-

Table 4.12: Cycle Matching Rejection Percentage

Cycle Matching

A graph *matching* is a set of edges such that no two edges share a vertex. Each edge has a propositional variable that is true if it is in the set, and false otherwise. This test does matching on a cycle of size 11.

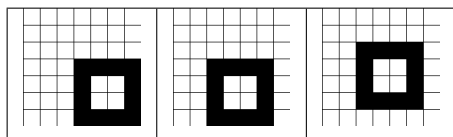
The **DGSA** method does much better here than for all the other problems we've seen so far.

All Squares

By inspection, the diverse sets created by non-agreement methods, such as **DGS**, are worse than those created by agreement methods. A subset of an diverse set created by **DGS** can be seen at Figure 4.13 on page 60. Not only do the squares have the same size, they are all in the same part of the grid, and also overlap with each other. On the other hand, if we look at Figure 4.14 on page 61, we see a variety of square sizes in a variety of places with almost no overlap. These are

		Rejection Methods					
		D	DG	DGS	A	DGA	DGSA
Creation Methods	D	-	0.54	0.99	1.00	1.00	1.00
	DG	0.00	-	0.97	1.00	1.00	1.00
	DGS	1.00	1.00	-	1.00	1.00	1.00
	A	0.84	0.85	0.84	-	0.98	0.98
	DGA	0.00	0.00	0.46	0.00	-	0.93
	DGSA	0.32	0.30	0.00	0.00	0.89	-

Table 4.13: Cycle Matching Rejection Ratio

Figure 4.13: A Diverse Set Created With **DGS**

both good representatives of the diverse sets returned as a whole.

It may be that **DGS** could be improved if we used a different distance measure, since two models that have many “off” spaces in common would be close when calculated with Hamming distance. For consistency we did not do this, and would also make the observation that agreement symmetry was able to find very diverse models without any additional human intervention.

Agreement symmetry certainly encodes a different pattern of diverse sets than non-agreement measures. Agreement symmetry on its own does worse than random compared to most other methods. Of particular note is how well **DGSA** performs compared to all other methods. It typically performs rather poorly on the methods it is not a superset of, especially **DGA**.

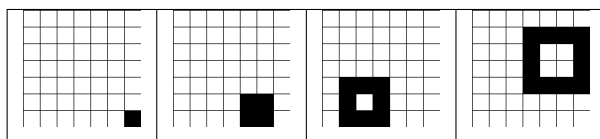


Figure 4.14: A Diverse Set Created With Agreement Symmetry

	D	DG	DGS	A	DGA	DGSA
Mean Size:	28.43	10.73	9.72	4.03	3.03	2.92

Table 4.14: All Squares Set Size

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	1.00	1.00	1.00	1.00	1.00
	DG	0.00	-	0.54	1.00	1.00	1.00
	DGS	0.46	0.46	-	1.00	1.00	1.00
	A	<1.00	<1.00	<1.00	-	<1.00	<1.00
	DGA	0.00	0.00	0.40	0.00	-	0.40
	DGSA	0.05	0.05	0	0	0.05	-

Table 4.15: All Squares Rejection Percentage

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	1.00	1.00	1.00	1.00	1.00
	DG	0.00	-	0.54	1.00	1.00	1.00
	DGS	0.46	0.46	-	1.00	1.00	1.00
	A	1.05	1.04	1.03	-	1.00	1.00
	DGA	0.00	0.00	0.46	0.00	-	0.41
	DGSA	0.06	0.06	0.00	0.00	0.05	-

Table 4.16: All Squares Rejection Ratio

	D	DG	DGS	A	DGA	DGSA
Mean Size:	4.85	4.00	3.00	1.00	1.00	1.00

Table 4.17: Reduced Latin Squares Set Size

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.98	1.00	1.00	1.00	1.00
	DG	0.95	-	1.00	1.00	1.00	1.00
	DGS	0.89	0.71	-	1.00	1.00	1.00
	A	0.00	0.00	0.00	-	0.00	0.00
	DGA	0.00	0.00	0.00	0.00	-	0.00
	DGSA	0.00	0.00	0.00	0.00	0.00	-

Table 4.18: Reduced Latin Squares Rejection Percentage

Reduced Latin Squares

In the Latin Squares problem is creating an $n \times n$ grid, where each cell can be given a number 1 through n and that number occurs only once for every row and column. In the reduced version, the first row and column are set. The most important takeaway from this problem is that this is a case where agreement symmetry is a bit too powerful.

Here, we see that sometimes agreement symmetry can be too powerful and only give us trivial diverse sets.

Conclusions

It is clear that there is no single method that is more discriminatory than the others over all problems, though the DGA and DGSA methods do the best overall. This is not surprising since they use the most information. We also see that using shortest-path correction creates dissimilarity graphs that tend to create diverse sets that are

		Rejection Methods					
Creation Methods		D	DG	DGS	A	DGA	DGSA
	D	-	0.98	1.00	1.00	1.00	1.00
	DG	0.96	-	1.00	1.00	1.00	1.00
	DGS	1.01	0.76	-	0.00	1.00	1.00
	A	1.00	1.00	1.00	-	1.00	1.00
	DGA	1.00	1.00	1.00	1.00	-	1.00
	DGSA	1.00	1.00	1.00	1.00	1.00	-

Table 4.19: Reduced Latin Squares Rejection Ratio

significantly different than those created without this correction.

No method uniformly creates larger diverse sets than the other methods, though methods that take into account more information tend to create smaller sets.

Chapter 5

Constructive Symmetry

5.1 Introduction

Even though local symmetry can discover similarities that other diversity measures cannot and we can find approximations that can be efficiently computed, there is a weakness to local symmetry that makes it less appealing from a theoretical perspective.

Recall the 8-Queens problem and the two globally asymmetric solutions as shown in Figure 4.2. Once we fix a queen in the LLC, all other solutions are “essentially” the same. However, if we rotated R_1 180 degrees then we have not changed the solution in a meaningful way, but we could not longer tell they are equivalent by fixing the LLC, even though they are both putting a queen on some corner.

To illustrate further, consider the $N \times N$ All-Rectangles problem. One set of mutually locally asymmetric models for the 4 All-Rectangles problem can be seen in Figure 5.1. While no two rectangles of the same size are present, there are two rectangles of size 3×1 and 1×3 . Intuitively, these are similar rectangles because we can do a rotation and translation to transform one to the other.

In general, the similarity between m_1 and m_2 should not be modified if we ap-

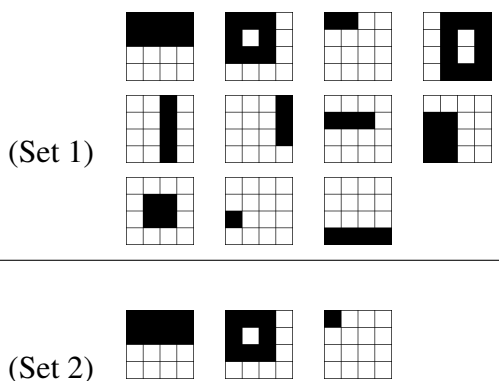


Figure 5.1: The 4×4 All-Rectangles problem: Set 1 is a diverse set of solutions based on local symmetry, set 2 is based on constructive equivalence.

ply a symmetry $\theta \in \text{Sym}(I), I \subseteq \alpha(m_1, m_2)$ to one of them. Our approach is to reconcile the two models using compositions of local symmetry. For example, we had two 8-Queens models with a queen on different corners, we would take advantage of global symmetry to rotate one of the boards so that the queens align and then proceed to use local symmetry to show their similarity. For the All-Rectangles problem, we would use global symmetry to rotate one of the rectangles so that they have the same orientation, and then use local symmetry to show that they are equivalent through translation.

We call this form of symmetry *Constructive Symmetry*. Each literal represents a "choice" and if two models can be built using equivalent choices, then they are considered similar to each other. Whether the choices are equivalent depends on local symmetry. This notion is the same as the **deep** pruning strategy for complete local symmetry; it attempts to find constructively equivalent PIs.

This new notion of similarity can be much stronger than local symmetry. In some cases it will be advantageous to weaken it by restricting which literals represent actual "choices" and which only represent side-effects.

5.2 Constructive Symmetry Properties

Constructive symmetry finds equivalences between ordered lists of literals we call *constructions*. M_c is the set of all such lists for a model M . For construction x we denote each element with index i as x_i . The indices are 0-based, and our last index will be n as a shorthand for $|V| - 1$.

Formally, two models mx, my are constructively equivalent if there exists constructions $cx \in mx_c, cy \in my_c$ where for every index i , cx_i is equivalent to cy_i . Literal equivalence is defined recursively. At index 0, cx_0 and cy_0 are equivalent if there is a global symmetry θ_0 such that $\theta_0(cx_0) = cy_0$. At index i , cx_i and cy_i are equivalent if there exists a local symmetry σ_i of partial interpretation $\{cy_k | k < i\}$ so that $\theta_i = \theta_{i-1} \times \sigma_i$ and $\theta_i(cx_i) = cy_i$. Thus, the index represents the size of the PI we compute local symmetries on. Since all the next symmetries are local symmetries on a larger partial interpretation, the index also tells us which literal is being stabilized.

A problem that immediately presents itself is that the number of constructions of a single model is the factorial of the number of variables. We will show that we do not need to explicitly create every possible construction, removing the factorial runtime.

At each index the combined symmetries transform mx into an intermediate model that is consistent with my 's next partial interpretation. We can represent a constructive symmetry as a *valid descending symmetry sequence*, a composition $\theta_0 \times \theta_1 \times \dots \times \theta_n$ from PIs $I_0 \subset I_1 \dots \subset I_n = my$. We call this a *descending sequence* when the meaning is obvious. For convenience, we will consider any subsequence of a valid descending sequence to be valid as well, under the presumption that the skipped local symmetries were identity permutations.

Each sequence will be denoted using an upper-case letter, each intermediate sequence will be denoted using the same upper-case letter and indexed by the size of the partial interpretation the last symmetry works on, and each symmetry will

be the denoted using the same letter but lower-case and indexed similarly. So for example $\Theta = \Theta_n = \Theta_{n-1} \times \theta_n = \theta_0 \times \theta_1 \times \dots \times \theta_n$.

Constructive symmetry can be pruned using SCP.

Theorem 5.1. *Given descending sequence Θ that maps construction cx to cy , if ϕ is a local symmetry on some $\{cy_0, cy_1 \dots cy_i\}$, then $\Theta^\phi = \theta_0 \times \theta_1 \times \dots \times \theta_i \times (\theta_{i+1} \times \phi) \times (\phi^{-1} \times \theta_{i+2} \circ \phi) \times \dots \times (\phi^{-1} \times \theta_n \times \phi)$ is a valid descending sequence.*

Proof. Apply Θ_{i+1} to cx to get intermediate construction

$$(cy_0, cy_1, \dots, cy_i, cy_{i+1}, \Theta_{i+1}(cx_{i+2}) \dots \Theta_{i+1}(cx_n))$$

Applying ϕ gives us:

$$(cy_0 \dots, cy_i, \phi(cy_{i+1}), (\Theta_{i+1} \times \phi)(cx_{i+2}) \dots (\Theta_{i+1} \times \phi)(cx_n))$$

Using SCP, $\phi^{-1} \times \theta_{i+2} \circ \phi$, is a local symmetry, so we have $(\Theta_{i+1} \times \phi \circ \phi^{-1} \times \theta_{i+2} \times \phi)(cx_j)$ for all $j > i + 1$, which is $(\Theta_{i+1} \times \theta_{i+2} \times \phi)(cx_j) = (\Theta_{i+2} \times \phi)(cx_j)$. So in particular $\Theta_{i+1}^\phi(cx_{i+2}) = \phi(cy_{i+2})$. Each new application of each new local symmetry in Θ^ϕ applies similarly. \square

We can interpret this theorem as sending cx to some new construction, or in a way that is more in line with our previous pruning theorem where we found a way to map

$$(cx_0, \dots, cx_i, \phi(cx_{i+1}), \phi(cx_{i+2}), \dots \phi(cx_n))$$

to

$$(cy_0, \dots, cy_i, \phi(cy_{i+1}), \phi(cy_{i+2}), \dots \phi(cy_n))$$

Constructive symmetry is a true equivalence relation on the set of constructions. If two constructions, cx and cy are constructively symmetric, we will denote it $cx \Delta cy$ as a relation. If cx and cy are related via some descending sequence Θ , we can also denote it $cx \Delta_\Theta cy$. The relation is obviously reflexive.

Theorem 5.2. *If $cx \Delta_{\Theta} cy$ and $cy \Delta_{\Lambda} cz$, then there is a descending sequence Φ where $cx \Delta_{\Phi} cz$.*

Proof. We will build Φ inductively. So $\phi_0 = \theta_0 \times \lambda_0$. We will make $\phi_k = \Lambda_{k-1}^{-1} \times \theta_k \times \Lambda_k$. What remains is to show that ϕ_k is a local symmetry.

There are two parts to being a local symmetry. The first is that it maps models to models within the partial interpretation, and the other is that it is the identity map for each literal in the partial interpretation.

Our next construction will be $(\Lambda_{k-1}^{-1} \times \theta_k \times \Lambda_k)(\Phi_{k-1}(cx)) = (\Theta_k \times \Lambda_k)(cx)$. Now $\Theta_k(cx) = (cy_0, cy_1, \dots, cy_k, \Theta_k(cx_{k+1}), \dots, \Theta_k(cx_n))$, which is a construction of some model that will work at every step of Λ_k , so $(\Theta_k \times \Lambda_k)(cx)$ is a construction of a model as desired. Indeed, the first k elements are identical to $\Phi_{k-1}(cx)$, meaning that ϕ_k is the identity map for each of those. \square

We can also prove this as a corollary to Theorem 5.1 by taking $(\theta_0 \times \theta_1 \times \dots \times \theta_n) \times (\lambda_0 \times \lambda_1 \times \dots \times \lambda_n)$ and applying SCP repeatedly. The first iteration will give us $((\theta_0 \times \lambda_0) \times (\lambda_0^{-1} \times \theta_1 \times \lambda_0) \times \dots \times (\lambda_0^{-1} \times \theta_n \times \lambda_0)) \times (\mathbf{id} \times \lambda_1 \times \dots \times \lambda_n)$. Distributing out λ_1 to λ_n in a similar manner gives us the same Φ .

In the end $\Phi = \Theta \times \Lambda$. This implies that composition of compatible descending sequences does not require storing every local symmetry in the sequence.

There are some cases where the transitive nature of the descending sequences is a more flexible. For example, if we had a descending sequence where every symmetry after index k is the identity, then the order of the construction after k doesn't matter since the identity exists in all partial interpretations and commutes with everything. In that case Θ , could map cx to some cy_1 and then Λ could map some compatible cy_2 to cz and the transitive property would still hold.

Theorem 5.3. *If $cx \Delta_{\Theta} cy$, then there is a descending sequence Φ so that $cy \Delta_{\Phi} cx$.*

Proof. The idea behind the proof is similar to that of Theorem 5.2. We let $\phi_0 = \theta_0^{-1}$ and $\phi_k = \Phi_{k-1}^{-1} \times \Theta_k^{-1}$. Our base case obviously holds. Inductively, we

know Φ_{k-1} is a valid descending sequence so far, so we have model $\Phi_{k-1}(cy)$, so $\phi_k(\Phi_{k-1}(cy)) = \Theta_k^{-1}(cy) = (\theta_k^{-1} \times \theta_{k-1}^{-1} \dots \times \theta_0^{-1})(cy)$. Since each θ_j^{-1} only changes the values of literals with indices between j and n , and each inverse is a valid local symmetry at each point it is applied, we end up with a model $(cx_0, cx_1, \dots, cx_k, \Phi_k(cy_{k+1}), \dots, \Phi_k(cy_n))$. Since it preserves the partial interpretation $\{cx_0, cx_1, \dots, cx_{k-1}\}$, ϕ_k is indeed a local symmetry. \square

Even though \triangle is an equivalence relation on constructions, it is not the case when we project the relation into the space of models. Since there are $|M||V|!$ different constructions, we need to have an algorithm that works on models to have a chance of having an efficient algorithm on even mild examples.

This also gives us an alternate way of expressing constructions. If we have descending sequence Θ , then we know that $\Theta^{-1} = \phi_0 \times \phi_1 \times \dots \times \phi_n$ so $\Theta = \phi_n^{-1} \times \phi_{n-1}^{-1} \times \dots \times \phi_0^{-1}$: An *ascending* sequence that works on partial interpretations that are decreasing in size at each step. In fact, ascending and descending sequences are equivalent and are in 1-1 correspondence. While descending sequences are more intuitive, ascending sequences will allow us to find constructively symmetric models more easily.

5.3 A General Algorithm

With these properties, we can show that computing complete constructive symmetry is not much harder than computing complete local symmetry in general since they can be generated from the same search tree MILE and its variants use to generate local symmetries.

The key is the memoization approach we used for SCP. Recall that for local symmetry the algorithm stores every edge generated by every visited PI I and its children, denoted $\text{Pairs}[I]$. So $\text{Pairs}[I] = \text{Sym}(I) \cup (\bigcup_{l \in L} \text{Pairs}[I \cup \{l\}])$. For constructive symmetry, $\text{Pairs}[I]$ should be all the constructive sym-

metric pairs. This is where ascending sequences become important; since they are built up from the most restrictive PIs first, we want to implicitly compose $\bigcup_{l \in L} \text{Pairs}[I \cup \{l\}]$ with every symmetry of I, ϕ . Composition applied to edges is simply applying ϕ to an individual vertex.

If the edges are directed, then we can only apply a symmetry to the endpoint vertex. Since constructive symmetry can move from one PI to another, we cannot say a priori that the edges of $\text{Pairs}[I]$ are undirected. If there is an ascending sequence Θ that goes through partial interpretation I , and all symmetries after I are the identity, then there is an ascending sequence equivalent to Θ^{-1} that also goes through I and uses only identity symmetries after I . Hence, at every node in our propagation we can consider each edge as an undirected edge. This means that the algorithm doesn't in general take up more space to compute constructive symmetry compared to local symmetry using case 1 correspondents. In practice, more storage is required since constructive symmetry generates more edges, but we don't increase the theoretical maximum number of edges we generate at each node.

The only difference to MILE* is how $\text{Pairs}[I]$ is generated hence, all of the weaknesses of the local symmetry algorithm are weaknesses for this algorithm as well. Even worse is that we cannot do any time/space trade offs by pruning using global symmetry alone.

There is an agreement symmetry version of constructive symmetry, but it can require far more time and space than the local symmetry version. We store the symmetries of every agreement and then organize them into a lattice. Using this lattice, we compute the constructively symmetric pairs in the same manner as the complete algorithm. While this is still sometimes fast, other times it is slower than even the complete algorithm. If we require a faster approximation, agreement local symmetry is still an $O(n)$ approximation of complete constructive symmetry because the number of edges has an upper bound but is often a poor substitute in practice.

5.4 Improving Constructive Symmetry

So constructive symmetry is more theoretically satisfying, but it is more difficult to approximate quickly. Another weakness is that it can often be too powerful, finding so many similarities that most diverse sets are trivial. However, this power gives room to investigate other approximations that can help with both weaknesses.

We investigated two incomplete offline methods that are better able to take advantage of the unique methodology that constructive symmetry uses. In the first, we restrict which literals represent a valid choice, as opposed to a side-effect of a choice. In the other, we only perform a shallow search on the PIs. Both of these methods can decrease the the number of similarities found in a principled way, and as a side effect can vastly decrease the runtime.

5.4.1 Restricting Choices

We defined every literal to represent a distinct "choice" for constructive symmetry, but there are other reasonable definitions that may depend on the problem. For example, the intuition behind building up solutions to the N-Queens problem is adding queens to the board (adding positive literals), not by planning ensuring that some space will never contain a queen (adding negative literals). In this case it makes sense to restrict our definition of a choice to only correspond to positive literals.

In general, we wish to only build PIs using a restricted set of choices. This set could be computed, or be chosen by a human. As long as each model is uniquely determined by a distinct set of the remaining literals, we will be able to modify our complete offline algorithm to compute constructive symmetries with these restrictions.

We test two general restrictions. The first assumes that positive literals represent constructive actions, while negative literals represent the lack of some action.

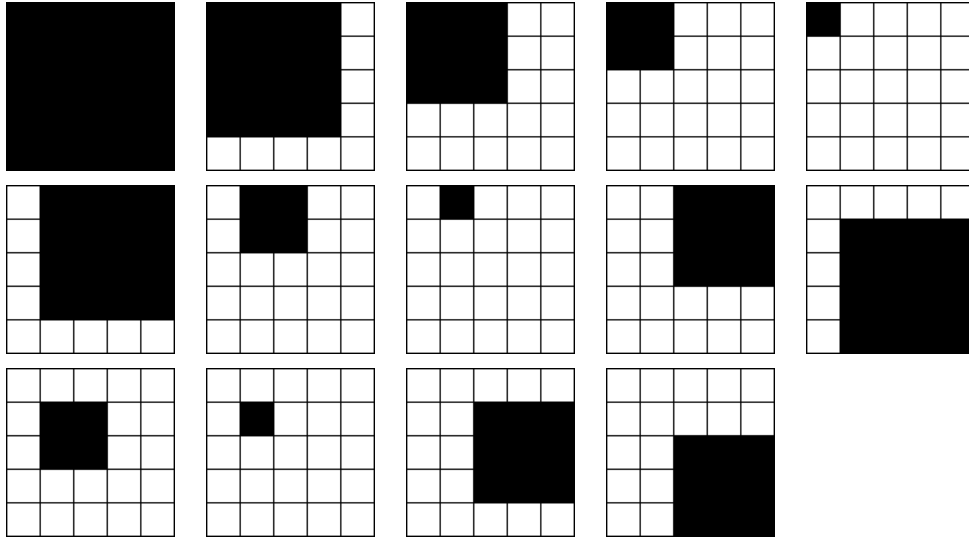


Figure 5.2: Computed diverse set for the 5×5 All-Filled-Squares problem using positive choices

This sort of interpretation is common in applications like planning. The main advantages of this restriction is that it is intuitive and simple. The set of positive literals of a model clearly represents that model. However, there are cases where negative literals are important for finding adequate similarity.

For the next restriction, we will only accept a literal l if $\forall l' \in L ((l' \implies l) \implies (l \implies l' \wedge l \leq l')) \vee (\forall m \in M (l' \notin m))$, where literals are compared using MILE ordering. Essentially, we wish to remove all literals implied by other literals under the interpretation that these literals represent only side-effects of choices. In regards to N-Queens, this leads to a more refined intuition as to why negative literals are nonconstructive: they are simply side-effects of our positive literals. This restriction also correctly allows both positive and negative literals for the $N \times N$ All-Squares problems. This flexibility has a cost; we are not guaranteed that every model will have a unique representation, though this can easily be checked.

Curtailing choices can lead to problems applying pruning, since we may not

traverse our tree in MILE order. A naive implementation may prune branches that need to be traversed, but this can be easily fixed. Since two models are only equivalent if we make equivalent choices, we can simply replace each model with its set of valid choices without losing any meaningful symmetry. Then every canonical interpretation will consist only of choices and every symmetry will only map between those choices. With this change, we can curtail choices and leave the complete offline algorithm almost unaltered.

5.4.2 Shallow Approximation

The second method is restricting the depth of the search tree. There are some similarities that may not be immediately intuitive even for local symmetry. For example in the $N \times N$ All-Filled-Squares problem squares of size N and $N - 1$ are considered similar, while we are able to find nonsymmetric examples of all other sizes. We have observed that most unintuitive similarities exist in the larger PIs. Therefore, restricting the number of choices we make can increase the efficiency, decrease the power to a more acceptable level, and perhaps increase the salience.

We made the search tree shallow using two approaches. First, we may restrict the minimum number of models that are consistent with any PI. Since we already remove PIs with fewer than 2 models, this change can be made to the complete offline algorithm by increasing that number and making no additional changes to the algorithm. The potential efficiency problem is if some set of literals are disproportionately represented in the models, then some branches may be much longer than others.

The other restriction is by restricting the total number of choices that are made. This is not a restriction on the size of the canonical PI, so some nodes that are pruned in the general case could not be pruned here, since we aren't guaranteed to have visited that particular interpretations. However, the runtime is more predictable. For example, if we are only allowed to make $O(\log(v))$ choices, then we

are guaranteed to only examine a polynomial number of PIs.

5.5 Results

We used the same tests we ran for complete local symmetry and added a test that use 8-Queens, which complete algorithms cannot finish. The complete constructive symmetry algorithm used **deep** SCP pruning and we compared it to its local symmetry counterpart.

Algorithms	8-Queens			
	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	4187	7.94	125.94	4186
Constr	504007	-1	0.00	0
Pos Lits	112	0.43	2325.58	2810
NoImpl	112	0.45	2222.22	2810
Depth 2	4368	5.61	178.25	2394
Depth 3	162934	169.42	5.90	3930
32 Mods	983757	-1	0.00	0
16 Mods	863439	-1	0.00	0
LocalSym	528264	-1	0.00	0

	7-Path			
Algorithms	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	12559	9.95	0.42	18084
Constr	2879	3.35	1.23	18336
Pos	95	0.46	8.98	18336
NoImpl	95	0.49	8.43	18336
Depth 2	104	0.52	7.94	13260
Depth 3	1205	1.42	2.91	17868
32 Mods	511	1.04	3.97	15132
16 Mods	1538	2.02	2.04	17724
LocalSym	2879	4.13	1.00	8298
	8-Path			
Algorithms	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	45742	70.37	0.22	72492
Constr	11520	19.27	0.81	73536
Pos	201	0.82	18.99	73536
NoImpl	201	0.86	18.10	73536
Depth 2	129	0.75	20.76	42900
Depth 3	1870	2.79	5.58	70188
32 Mods	4022	6.56	2.37	68784
16 Mods	8186	12.53	1.24	72744
LocalSym	11520	15.57	1.00	28314

Algorithms	9-Path			
	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	166108	-1	0.00	97458
Constr	47173	156.13	0.76	294528
Pos	437	2.21	53.47	294528
NoImpl	437	2.27	52.05	294528
Depth 2	172	1.36	86.88	123492
Depth 3	2855	7.94	14.88	268464
32 Mods	25329	71.52	1.65	287436
16 Mods	39444	120.59	0.98	293340
LocalSym	47173	118.16	1.00	98910

Algorithms	4×4 All-Squares			
	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	436	0.74	1.26	430
Constr	342	1.01	0.92	430
Pos	13	0.17	5.47	58
NoImpl	342	1.04	0.89	430
Depth 2	98	0.4	2.33	242
Depth 3	870	0.92	1.01	414
32 Mods	1	0.13	7.15	58
16 Mods	87	0.46	2.02	162
LocalSym	342	0.93	1.00	286

5×5 All-Squares				
Algorithms	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	1486	1.43	26.87	1467
Constr	38984	55.03	0.70	1467
Pos	36	0.33	116.45	130
NoImpl	38984	72.25	0.53	1467
Depth 2	314	0.87	44.17	522
Depth 3	5630	3.92	9.80	994
32 Mods	905	1.28	30.02	450
16 Mods	36775	44.87	0.86	462
LocalSym	38984	38.43	1.00	858
4×4 All-Rectangles				
Algorithms	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	4402	2.5	1.34	3549
Constr	6564	3.28	1.02	3549
Pos	36	0.25	13.36	241
NoImpl	6564	4.01	0.83	3549
Depth 2	115	0.41	8.15	479
Depth 3	1427	1.06	3.15	1647
32 Mods	611	0.69	4.84	673
16 Mods	3875	1.75	1.91	673
LocalSym	6564	3.34	1.00	1233

Algorithms	Random Models			
	#Sym	Time	%Sym Speedup	#Edges
ConstrAgree	546497	-1	0.00	120681
Constr	1207462	881.4	0.81	732717
Pos	8009	6.67	106.52	285
NoImpl	1187002	-1	0.00	0
Depth 2	339	1.97	360.66	0
Depth 3	2627	6.64	107.00	0
32 Mods	286467	156.41	4.54	339
16 Mods	573239	272.63	2.61	48910
LocalSym	1207462	710.5	1.00	261173

Constructive symmetry is slower but within the same order of magnitude as local symmetry. The slowdown is attributed to the cost of propagating many more edges using SCP. We also find that the agreement version of constructive symmetry can be much slower than the complete algorithm in some cases. For graph coloring, this is because it computes the local symmetries of many more PIs. These local symmetries tend to be on much smaller formula, which is why local symmetry agreement is still fast in most cases. However, since we have to compute a lattice and propagate through it, adding more PIs extracts an additional computational cost. There are still cases where an agreement algorithm is still faster than a complete algorithm, and it still computes close to the complete set of edges so it is still a useful tool.

All the graph coloring problems have complete constructive symmetry graphs, and the All-Squares problems have around 99% of the edges of a complete graph. This number of edges makes finding nontrivial diverse sets difficult, which limits its usefulness. By utilizing restrictions, we are not only able to compute diversity graphs more quickly, but we are also able to reduce the number of edges to a more reasonable degree. However, it is clear that the restrictions need to be tuned to the problem. For 8-Queens, restricting the minimum number of models in a

PI still leads to timeouts since adding negative literals remove very few models. For graph coloring, restricting choices doesn't decrease the number of edges at all. There are many cases where no literal is implied by any other literal, which makes the NoImpl strategy useless. Lastly, for random models most of the symmetries only exists in the very smallest PIs. So restricting based on choices and depth give useless diversity graphs.

Overall, constructive symmetry is a more theoretically satisfying, but there are more computational challenges to address.

Chapter 6

Discussion

6.1 Overview of Completed Work

Results of the previous chapters reaffirms the context sensitivity of diversity. Our work deepens the understanding of the tradeoffs between diversity measures, and we have added new notions of diversity.

There are numerous generic distance variations that are fast to compute, have clear interpretations, and can work well even in the presences of incomplete data and noise. Utilizing shortest-path correction we can generalize any pairwise notion of diversity D into a D -aware distance measure, so all of our work is compatible with distance measures.

Generic distance measures such as Hamming distance are widely applicable, but are not the most precise. Large distances do not necessarily correspond to objects that have very dissimilar structure but there it seems that small distances will often correspond to similar objects. Even for local symmetry, a small Hamming distance between two objects implies their agreement set is small, which is more likely to contain symmetries that map one object to the other.

Uniform sampling's advantages are that it is unbiased and unpredictable. Every object has an equal chance of being sampled and it is relatively likely that

if we build two diverse sets they will contain different models. But there are no guarantees that a set picked by uniform sampling is actually diverse. This is especially clear in cases where a large percentage of the objects are structurally similar with each other, the chosen set will likely contain many such objects. Uniform sampling should not be seen as a diversity measure itself, but rather as part of a strategy for choosing a diverse set given a diversity graph.

Global symmetry represents objects that are structurally identical. It is fast to compute in practice and partitions objects, which can be useful in some contexts. Unfortunately, it is not fit as a general-purpose notion of diversity. Most sets of objects will have no global symmetry at all. Even when there is global symmetry, there are other structural similarities that will often be missed.

Local symmetry is at least as powerful as global symmetry in all cases, and is fit to be a general notion of diversity. The difficulty is the computational cost. To apply to large application-driven problems, local symmetry must be approximated and even then there are tradeoffs between speed and accuracy which can drastically effect the quality of the results.

Still, it is an important diversity concept since it can find similarities that no other measure can easily encompass. Early on in our work, it wasn't clear that local symmetry could be approximated in an efficient manner on even problems with $|V|$ in the teens and the idea of approximating it for problems with over 100,000 variables and $|M|$ orders of magnitude beyond a billion was unthinkable. This dissertation has progressed not only the idea of using local symmetry for diversity, but its applicability as well.

While less applicable, constructive symmetry is able to fix some theoretical problems of local symmetry. While local symmetry is a known concept that has not yet been applied to this specific problem, constructive symmetry is a completely novel notion. Its advantages go beyond diversity, as it is actually the idea behind **deep** symmetry pruning, the most effective of all our pruning algorithms. We are confident that there are search problems that could benefit from this prun-

ing technique.

Choosing Diversity Given the wide variety of diversity techniques available, it may be confusing to determine which to use for a given problem. This is a hard problem in general but there are some heuristics that can be used to accelerate the process.

Much depends on whether an offline algorithm is feasible or not. If it is not, then the first concept attempted should be online semantic local symmetry. If the performance is unacceptable, then syntax local symmetry should be attempted. The precise application of symmetry breaking clauses will likely have to be tuned to the problem. It may be the case that online local symmetry accepts the vast majority of candidates, which would suggest it isn't so useful for that problem. One may choose to either augment or completely replace local symmetry with some distance concept using techniques from other's work. Choosing to sample deterministically, by random phase selection, random sampling or some other technique depends on individual needs.

If offline algorithms are feasible, there are many more options to choose from. We suggest starting by applying diversity measures that don't require any parameter tuning. This means the complete and agreement versions of local/constructive symmetry and global symmetry. Constructive symmetry is recommended if it is fast enough and gives good results. Otherwise, local symmetry should be used. If even agreement local symmetry is too slow, then our suggestion is global symmetry plus distance using shortest path correction. In the rare cases that global symmetry is too powerful, then distance is the best we can do.

If local or constructive symmetry is not discriminating enough, then distance and shortest path correction can apply more leverage. When combining them, the weight given to symmetric models needs to be tuned. Unlike global symmetry, local symmetry graphs often have very small diameter, so using small edge weights would completely overshadow the distance information. We recommend a weight

close to the smallest measured distance; we find .9 when using Hamming distance tends to work well.

6.2 Future Work

6.2.1 Link Analysis

The diversity graph is a powerful representation of commonality between models whose applications go beyond diverse models. One we believe may be particularly interesting is link analysis.

Link analysis uses the edge of structure of the graph to find relationships between nodes. One such relationship is graph centrality which is often used to find the most "important" or "influential" nodes in a graph. Our interpretation of centrality on diversity graphs is that highly central nodes are "characteristic" models that exhibit a typical structure, while nodes that have low centrality describe more unusual models that have few similarities with the other models.

For even simple link analysis, where the most characteristic model is the one with the most incident edges, local symmetry can yield interesting results. Consider 3-coloring a path of 6 nodes. With global symmetry, there is a 72-way tie for the most characteristic solution, while the remaining 24 solutions are tied for the least characteristic (or the most unusual). The two sets are split based on whether the sequence of colors forms a palindrome (i.e., the coloring is symmetric about the middle node(s) of the graph). But this is not at all descriptive of the overall problem structure.

Local symmetry judges the unusual models as the six that are 2-colored, since they are models that are different from all other ways to 3-color the path. The characteristic models consist of 12 models that contain every color with equal frequency in some irregular way. This reflects the high likelihood that such a model shares some substructure with every other model. So again local symmetry

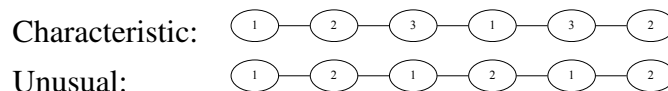


Figure 6.1: Most unusual and characteristic ways to 3-coloring a path based on local symmetry. All other examples are globally symmetric to these.

analysis can discover more fine-grained similarities than global symmetry.

6.2.2 Complete Symmetry

The biggest problem with the fastest complete algorithms is that they require a large amount of space to efficiently prune. A memory-sensitive cache could reduce memory overhead. If a PI that could have been used to prune was removed from memory, the subtree simply needs to be re-computed. We do not currently know if this would be effective, as this would require certain PIs being used far more than others. If it is, there is also a question of what strategy we should use. One potentially powerful heuristic would be to remove that largest PIs first, since a cache miss would require recreating smaller subtrees.

We also did not investigate modifying the order in which we add literals to PIs. For some search problems this can have large effect on the runtime. Doing a global reordering of the variables would not effect the algorithm, but a dynamic reordering depending on the current sub-formula may present an insurmountable obstacle to most of the benefits of fixed-literals and SCP pruning. It is not clear if there is an effective heuristic for variable ordering since performing an exhaustive search and pruning is based on properties of sub-formula that are currently unpredictable.

Instead of a depth-first approach, we could instead use a breadth-first strategy. The appeal of this strategy is that it allows for even greater pruning. Since we have all of the subsets of a given PI, we can use the symmetries of every one of its subsets to try and prune it. In our early experiments, the cost of testing all the

symmetries was not worthwhile. This was before deep pruning was developed, so it's possible that strategy could work better in this case. The use of breadth-first search may still be helpful in some particular areas.

A future approach could instead solve a number of LSM decision problems for a more focused search. Since any PI that maps m_1 to m_2 is a subset of their agreement, the search space for this problem is at most $O(2^{|\alpha(m_1, m_2)|})$, which will be much smaller than an $O(3^{|V|})$ for an exhaustive search

As a basic optimization would be to add all the edges we find while attempting to find if there is an edge from m_1 to m_2 . A unique optimization is that this method can tell us if two models are definitely not symmetric, so the lack of an edge could be propagated using SCP.

This approach could lead to drastically better performance, since many of the symmetries we compute in exhaustive search are redundant. To fully realize this performance, we believe that there needs to be a sound heuristic that can identify situations where two models are definitely not locally symmetric on any PI. Otherwise, a complete search could have significant overlap. Another possibility is some way to mark PIs has previously visited in a space-efficient manner. Even without a way to ensure there is not overlap, this algorithm could still have some applications to the online algorithm since we only compare a small subset of the total number of models.

6.2.3 Over-estimating Symmetry

In this paper we concentrated on approximation methods that underestimate the amount of symmetry, but there are reasons why overestimation may be appropriate. For instance, the typical application of the online algorithm is to find diverse sets that are much smaller than the typical size of a maximal diverse set. Since it will ignore many models that are diverse, it makes sense to strengthen our notion of diversity to ensure our small set is as diverse as possible.

Overestimation allows for local symmetry pruning without the memory overhead that is typically required. Edges are immediately set in the diversity graph as soon as they are found. When finding a case 1 correspondent (I, I') with link ϕ we calculate the edges of the subtree I' by taking the diversity subgraph induced by models consistent with I and propagating the edges of that graph using ϕ . This eliminates the need to store individual edges for each PI. It may end up propagating edges that were not found in the subtree rooted at I , but in our preliminary tests on this method the amount of overestimation was typically very small.

To over-estimate symmetry for the online algorithm, we may replace local-symmetry breaking clauses with global symmetry breaking clauses. This causes far more pruning to occur at a far lower memory cost. In preliminary tests this seemed to produce a reasonable number of highly diverse models.

6.3 Conclusion

This dissertation should be a valuable resource to those working on propositional diversity problems to bring insight into their choice of diversity concepts. It is important to consider what each diversity measure is gauging to ensure finding objects that are truly structural different instead of being only superficially distinct. The symmetry-based approaches we have presented here represent the state-of-the-art for finding these true differences that work for a wide variety of problems.

Bibliography

- [1] Fadi A. Aloul, Igor L. Markov, and Karem A. Sakallah. Shatter: efficient symmetry-breaking for boolean satisfiability. In *Proceedings of DAC'03*, pages 836–839. ACM, 2003.
- [2] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Solving difficult sat instances in the presence of symmetry. In *Proceedings of DAC'02*, pages 731–736. ACM, 2002.
- [3] Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov. Efficient symmetry breaking for boolean satisfiability. *IEEE Trans. Comput.*, 55(5):549–558, May 2006.
- [4] Ola Angelsmark and Johan Thapper. Algorithms for the maximum hamming distance problem. In BoiV. Faltings, Adrian Petcu, Francois Fages, and Francesca Rossi, editors, *Recent Advances in Constraints*, volume 3419 of *Lecture Notes in Computer Science*, pages 128–141. Springer Berlin Heidelberg, 2005.
- [5] Noriko H. Arai and Alasdair Urquhart. Local symmetries in propositional logic. In *Proceedings of TABLEAUX'00*, pages 40–51. Springer-Verlag, 2000.

- [6] Olivier Bailleux and Pierre Marquis. Distance-sat: Complexity and algorithms. In *Proceedings of the 16th AAI and the 11th IAAI*, pages 642–647. AAI, 1999.
- [7] Belaid Benhamou. Study of symmetry in constraint satisfaction problems. In *Proceedings of CP'94*, pages 246–254, 1994.
- [8] Belaid Benhamou, Tarek Nabhani, Richard Ostrowski, and Mohamed Réda Saidi. Dynamic symmetry breaking in the satisfiability problem. In *Proceedings of LPAR'10*, 2010.
- [9] Belaid Benhamou and Mohamed Réda Saidi. Dynamic detection and elimination of local symmetry in cps.
- [10] Belaid Benhamou and Lakhdar Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1):89–102, 1994.
- [11] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- [12] Gregory Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *LNCS*. Springer-Verlag, 1991.
- [13] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Balancing scalability and uniformity in sat witness generator. *DAC '14*, pages 60:1–60:6, New York, NY, USA, 2014. ACM.
- [14] Supratik Chakraborty, KuldeepS. Meel, and MosheY. Vardi. A scalable and nearly uniform generator of sat witnesses. In *Computer Aided Verification*, volume 8044 of *LNCS*, pages 608–623. Springer Berlin Heidelberg, 2013.

- [15] David Cohen, Peter Jeavons, Christopher Jefferson, Karen E Petrie, and Barbara M Smith. Symmetry definitions for constraint satisfaction problems. In *Proceedings of CP'05*, pages 17–31. Springer, 2005.
- [16] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of KR'96*, volume 96, pages 148–159. Morgan Kaufmann, 1996.
- [17] James M. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic (extended abstract). In *AAAI Workshop on Tractable Reasoning*, pages 17–22, 1992.
- [18] P. Crescenzi and G. Rossi. On the hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85 – 100, 2002. Complexity and Logic.
- [19] Thomas Eiter, Esra Erdem, Halit Erdogan, and Michael Fink. Finding similar or diverse solutions in answer set programming. In *Logic Programming*, volume 5649 of *LNCS*, pages 342–356. Springer Berlin Heidelberg, 2009.
- [20] Thomas Eiter, Esra Erdem, Halit Erdogan, and Michael Fink. Finding similar/diverse solutions in answer set programming. *Theory and Practice of Logic Programming*, 13:303–359, 5 2013.
- [21] Martin Ester, Hans P. Kriegel, Jorg Sander, and Xiaowei Xu. A Density-Based algorithm for discovering clusters in large spatial databases with noise. In *2nd KDD*, pages 226–231. AAAI Press, 1996.
- [22] Ian P Gent, Iain McDonald, Ian Miguel, and Barbara M Smith. Approaches to conditional symmetry breaking. In *Proceedings of the satellite workshop of CP*, 2004.
- [23] Ian P Gent, Iain McDonald, and Barbara M Smith. Conditional symmetry in the all-interval series problem. In *Proc. SymCon. Vol. 3.*, 2003.

- [24] IanP. Gent, Tom Kelsey, SteveA. Linton, Iain McDonald, Ian Miguel, and BarbaraM. Smith. Conditional symmetry breaking. In Peter Beek, editor, *CP'05*, volume 3709 of *Lecture Notes in Computer Science*, pages 256–270. Springer Berlin Heidelberg, 2005.
- [25] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2012.
- [26] Emmanuel Hebrard, Brahim Hnich, Barry O’Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *Proceedings of the 20th National Conference on AI - Vol 1*, AAI’05, pages 372–377. AAI Press, 2005.
- [27] Emmanuel Hebrard, Barry O’Sullivan, and Toby Walsh. Distance constraints in constraint satisfaction. In *Proceedings of the 20th IJCAI*, pages 106–111. Morgan Kaufmann Publishers Inc., 2007.
- [28] Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In T.Walsh I.P.Gent, H.v.Maaren, editor, *SAT 2000*, pages 283–292. IOS Press, 2000. SATLIB is available online at www.satlib.org.
- [29] Ethan K. Jackson, Gabor Simko, and Janos Sztipanovits. Diversely enumerating system-level architectures. In *Proceedings of the 11th ACM EMSOFT*, pages 11:1–11:10. IEEE Press, 2013.
- [30] Tommi Junttila and Petteri Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of ALENEX'07*. SIAM, 2007.
- [31] Tommi Junttila and Petteri Kaski. Conflict propagation and component recursion for canonical labeling. In *Theory and Practice of Algorithms in (Computer) Systems*, volume 6595 of *LNCS*, pages 151–162. Springer-Verlag, 2011.

- [32] Hadi Katebi, Karem A. Sakallah, and Igor L. Markov. Symmetry and satisfiability: an update. In *Proceedings of SAT'10*, pages 113–127. Springer-Verlag, 2010.
- [33] Hadi Katebi, Karem A. Sakallah, and Igor L. Markov. Conflict anticipation in the search for graph automorphisms. In *Proceedings of LPAR'12*, pages 243–257. Springer-Verlag, 2012.
- [34] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22(3):253–275, 1985.
- [35] Brendan D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.
- [36] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.
- [37] Alexander Nadel. Generating diverse solutions in sat. In Karem A. Sakallah and Laurent Simon, editors, *SAT 2011*, volume 6695 of *LNCS*, pages 287–301. Springer Berlin Heidelberg, 2011.
- [38] Toby Walsh. General symmetry breaking constraints. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006*, volume 4204 of *Lecture Notes in Computer Science*, pages 650–664. Springer Berlin Heidelberg, 2006.
- [39] Toby Walsh. Breaking value symmetry. In Christian Bessière, editor, *Principles and Practice of Constraint Programming CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, pages 880–887. Springer Berlin Heidelberg, 2007.
- [40] Toby Walsh. Symmetry breaking constraints: Recent results, 2012.