A Perspective on Certain Polynomial Time Solvable Classes of Satisfiability

John Franco
Computer Science Division, ECECS
University of Cincinnati
Cincinnati, OH 45221-0030
E-mail: `franco@ececs.uc.edu`

Allen Van Gelder
Computer Science Dept.
University of California
Santa Cruz, CA 95064
E-mail: `avg@cs.ucsc.edu`

November 2, 2001

Keywords: Satisfiability, Random CNF formulas, Matching

Abstract

The scope of certain well-studied polynomial-time solvable classes of Satisfiability is investigated relative to a polynomial-time solvable class consisting of what we call *matched* formulas. The class of matched formulas has not been studied in the literature, probably because it seems not to contain many challenging formulas. Yet, we find that, in some sense, the matched formulas are more numerous than Horn, extended Horn, renamable Horn, q-Horn, CC-balanced, or Single Lookahead Unit Resolution (SLUR) formulas.

The behavior of random k-CNF formulas generated by the constant clause-width model is investigated as n and m, the numbers of variables and clauses, go to infinity. For $m/n > 2/k(k-1)$, the probability that a random formula is SLUR, q-Horn, extended Horn, CC-balanced, or renamable Horn tends to 0. For $m/n < .64$, random formulas are matched formulas with probability tending to 1. For $m/n^{k-1} \geq 2^k/k!$, random formulas are solved by a certain polynomial-time resolution procedure with probability tending to 1.

The propositional connection graph is introduced to represent clause structure for formulas with general-width clauses. Cyclic substructures are exhibited that occur with high probability and prevent formulas from being in the previously studied polynomial-time solvable classes, but do not prevent them from being in the matched class. We believe that part of the significance of this work lies in guiding the future development of polynomial-time solvable classes of Satisfiability.

1 Introduction

The Satisfiability problem (SAT) is to determine whether there exists a satisfying truth assignment for a given Boolean expression, usually in Conjunctive Normal Form (CNF). This problem is NP-complete; thus, there is no known polynomial-time algorithm for solving it. Because of the importance of SAT in logic, artificial intelligence, and operations research, considerable effort has been spent to determine how to cope with this disappointing reality. Two approaches are: (1) determine whether there exist algorithms for SAT which *usually* present a result in polynomial time; (2) identify special classes of SAT that can be solved in polynomial time. This paper is concerned with the second approach.

Several polynomial-time solvable classes of CNF formulas have been proposed. Some of the most notable (see Section 3 for definitions) are:

1. Horn [19, 31, 38],
2. renamable Horn [34, 5],
3. extended Horn [13],
4. q-Horn [8, 9].
5. CC-balanced [17],
6. SLUR (Single Lookahead Unit Resolution) solvable [37],

Below, we refer to these as the well known polynomial-time solvable classes. While this paper was under review, a new polynomial-time solvable class, called LinAut, was discovered [33, 35]. We give some preliminary analysis of this class also, in Section 8.3.

Schaefer proposed a scheme for defining classes of propositional formulas with a generalized notion of "clause" [36]. He proved the very interesting fact that every class definable within his scheme was either NP-complete or polynomial-time solvable, and he gave criteria to determine which. It is worth mentioning that not all classes can be defined within his scheme, and only the Horn class, among those listed above, can be so defined. The reason is that his scheme is limited to classes that can be recognized in log space.

We also introduce a new class, which we call the class of *matched* formulas, that is solved by a polynomial-time maximal matching algorithm. The matched formulas are formally presented in Definition 3.6. Our motivation for investigating this class is that (1) it seems to contain mostly unchallenging formulas, yet (2) matched formulas are not vulnerable to the presence of certain cyclic substructures of the type that prevent them from being members of other well known polynomial-time solvable classes.

We wish to determine whether any of the well known polynomial-time solvable classes is close to what might be considered the largest easily definable polynomial-time solvable class. A positive answer would suggest the use, in a general SAT solver, of a special preprocessor that would attempt to recognize whether a given input formula is a member of such a class and apply a fast algorithm, if it is. To answer the question we attempt to compare the class of matched formulas with the well known polynomial-time solvable classes. But, the largest of these classes are incomparable with each other and the class of matched formulas. Therefore, we use a probabilistic approach to determine whether one class subsumes another.

We find, by means of a reasonable probabilistic measure, the probability with which a random formula is in one of the classes listed above. We use the well known *random k-CNF* distribution, $\mathcal{M}^k_{m,n}$, described later in this section, which has a parameter $r = m/n$ that can be adjusted to change some of the structural characteristics of generated formulas. In the following discussion, "with high probability" means "with probability tending to 1 as n goes to ∞". In all cases k remains fixed, whereas m varies with n and also approaches ∞.

Our interest in the scope of the polynomial-time solvable classes is due primarily to results of Boros *et al.* [9], which suggest that the class of q-Horn formulas is close to what might be regarded as the largest easily expressible class of SAT that can be solved by a polynomial-time, uniform algorithm. They formulate a set of linear constraints, based on the input formula and a real parameter Z, and show that, for any fixed $c > 0$, the class of formulas that satisfies the constraints with $Z = 1 + c\frac{\log n}{n}$ can be solved in polynomial time. In addition, the class that satisfies the constraints with $Z = 1$ is precisely the q-Horn class (see Definition 3.3). On the other hand, for any $\beta < 1$, the class of formulas that satisfy these constraints with $Z = 1 + \frac{1}{n^\beta}$ is NP-complete.

To measure scope we use a well known probability distribution $\mathcal{M}^k_{m,n}$, also known as the constant clause-width model, defined over the sample space of k-CNF formulas constructed of m clauses taken uniformly and independently from n variables. Thus, the clause space for $\mathcal{M}^k_{m,n}$ consists of the $2^k\binom{n}{k}$ clauses with k literals such that no two literals are based on the same variable. (For actual clause sampling all permutations of the k literals are assumed equally likely.) The formula space consists of all sequences of m clauses; each having equal probability, which is $\left(2^k\binom{n}{k}\right)^{-m}$. In other words, clauses are generated by sampling *without* replacement, while formulas are generated by sampling *with* replacement. This paper primarily considers $k \geq 3$. Probability spaces will frequently be grouped according to the parameter $r \equiv m/n$. Asymptotic behavior is studied as m and n increase, while k remains fixed.

We determine those values of the parameter r for which a random formula has low probability of being in a certain class, such as q-Horn, etc., except in the case of matched formulas where we determine the values of r for which the probability of being a matched formula is high. We use this approach because

1. several of the classes considered are incomparable;
2. the ratio $r = m/n$ provides a scale which has been shown, both theoretically and experimentally, to measure the *hardness* of formulas;
3. many results already proven for $\mathcal{M}^k_{m,n}$ may be used to add dimension to the results presented here.

The following results for random formulas under $\mathcal{M}^k_{m,n}$, $k \geq 3$, are known:

1. For any $r \geq 2^k \ \ln 2$, a random formula is unsatisfiable with high probability [23].
2. For any fixed $r \geq 2^k \ \ln 2$, the shortest resolution refutation of a random formula has a super-polynomial number of steps with high probability [15].
3. For any fixed $r < 2^k/(4k)$, a random formula is satisfiable with high probability [16, 14]. For $k = 3$ a bound of $r \leq 3.003$ has been shown [25].
4. For any fixed $r < 2^k/(4k)$, with high probability, a random formula that is satisfiable can be solved in linear time by an iterative variable elimination algorithm that relies primarily on choosing variables for elimination from a shortest clause [14, 16, 25].
5. For any fixed $r < 1$, a random formula can be satisfied with high probability, by applying any algorithm for 2-SAT to the formula with all but 2 literals randomly removed from each clause [29].
6. The average number of occurrences of a variable in a random formula is less than 1 if $r < 1/k$.

The first two points above show where random formulas are "hard" and the remaining points show where random formulas are "easy." Notice the progression from hard formulas at $r \geq 2^k \ \ln 2$ to very easily solvable formulas at $r \leq 1$. For any fixed $r \geq 2^k \ \ln 2$, resolution is exponential and no known polynomial-time algorithm detects unsatisfiability with high probability. Almost all formulas are unsatisfiable in this range. For $r \leq 2^k/(4k)$ formulas are solvable in polynomial time with high probability by non-trivial methods. For $r \leq 1$, they are solvable by a 2-SAT algorithm with high probability. Also, for 3-CNF from a different distribution, the pure literal rule alone is sufficient with high probability, for $r < 1.63$ [10]. Thus, $\mathcal{M}^k_{m,n}$ may be thought of as a generator of formulas of hardness controlled by the ratio r. We wish to see where the well known polynomial-time solvable classes and the class of matched formulas fall on this scale. This paper presents the following results.

1. The propositional connection graph is introduced as a structure on sets of clauses (Section 5). This structure proves useful for the analysis of clauses of all widths, whereas the well-known implication graph is limited to binary clauses.
   (a) If a particular simple cyclic substructure exists in the propositional connection graph of a formula, then the formula cannot be q-Horn (Section 7).
   (b) If a particular simple cyclic substructure exists in the propositional connection graph of a formula, then the formula cannot be SLUR (Section 6).
2. For any $r > 2/(k(k-1))$, a random formula *is not* q-Horn or SLUR with high probability because the above-mentioned substructures occur with high probability. It follows that a random formula is not Horn, renamable Horn, extended Horn, or CC-

balanced with high probability since these classes are subsumed by q-Horn or SLUR or both [37].

3. For any fixed $r < .64$, a random formula *is* in the matched class with high probability (Section 8). Actually, the bound increases with k, in contrast with the other classes, in which the bound decreases with k; e.g., for $k = 4$, the range is $r < .84$. Computed values in Figure 4 suggest that the bound approaches 1 as k grows.
4. For any $r < 1.36/(k(k-1))$, the average number of cycles in the propositional connection graph associated with a random formula is less than one (Section 9). It is shown that a formula without cycles in its propositional connection graph must be satisfiable.

These results strengthen and simplify preliminary versions that were presented elsewhere [21, 24].

Related results concerning "easy" unsatisfiable formulas are also presented:

1. A minimally unsatisfiable set of clauses has fewer distinct variables than clauses (Section 8.1). This result was stated by Aharoni and Linial [2] without proof; we give a straightforward proof. Independently, Kleine Büning gave a different proof using entirely different techniques [12].
2. Consider classes $M^k_{n,m}$ where m grows superlinearly with n, while k is fixed. An algorithm is presented that searches for k-bounded resolution refutations; that is, each derived clause is required to contain at most k literals. If $m \geq (2^k/k!)n^{k-1}$, then a random formula can be certified as unsatisfiable in polynomial time by this algorithm, with high probability (Section 10). Fu independently observed that short resolution proofs exist with high probability in this range [26]. Recently, Beame *et al.* improved upon this by showing that certain polynomial-time resolution-based procedures succeed with high probability in the range $m = \Omega\left(n^{k-1}/\ln^{(k-2)} n\right)$ [6]. So after passing through an apparently hard region, random formulas again become tractable at very high clause densities.

Therefore, the well known polynomial-time solvable classes of SAT, by our measure, do not contain most of the "easy" formulas for a wide range of values of r and these classes are much smaller than the matched class for $2/k(k-1) < r < .64$. It is interesting to note that, with high probability, a random formula $\mathcal{F}$ fails to be in any of the well known polynomial-time solvable classes, unless the average number of occurrences of a variable in $\mathcal{F}$ is less than $2/(k-1)$, a very small number. In addition, our results show that most random unsatisfiable formulas are not members of one of the well known polynomial-time solvable classes. Finally, this probabilistic analysis has uncovered particular cyclic substructures such that just one such substructure in a formula prevents it from being in one of the well known polynomial-time solvable classes. For the matched class, there is no such "killer" substructure.

2 Notation and Terminology

We specify SAT for the purposes of this paper as follows.

Definition 2.1: Let $V_n = \{v_1, \ldots, v_n\}$ be a set of n Boolean variables. Let $L_n = \{v_1, \overline{v_1}, \ldots, v_n, \overline{v_n}\}$ be a set of n positive and n negative *literals* over variables in V_n. Our proofs consider arbitrary sequences of literals from L_n. In such cases, sequences will be represented as $u_1, u_2, \ldots, u_i, \ldots$ instead of specific sequences of L_n literals. Observe that the index of symbol u_i, for example, only reflects the order of that symbol in the representing sequence and does not require u_i to be the same as v_i or $\bar{v}_i$ or any other L_n literal. Each u_i is either a positive or negative literal and $\overline{u_i}$ denotes its complement.

There is no distinction between $\overline{\overline{u}}$ and u. The variable underlying literal u_i is denoted as $|u_i|$ Two literals u_i and u_j are said to *variable-distinct* if $|u_i| \neq |u_j|$. □

A subset of the literals in L_n is called a *clause*. Clauses are denoted as $[u_1, u_2, \ldots, u_k]$ for readability. Literals in a clause are joined disjunctively, and the *empty clause* is always false.

A sequence of clauses, $\{C_1, C_2, \ldots, C_m\}$, is a *formula in Conjunctive Normal Form* (CNF). Clauses in a formula are joined conjunctively, and the *empty formula* is always true. Throughout this paper, it is understood that *formula* means "formula in Conjunctive Normal Form". Although all permutations of a sequence of clauses yield the same logical formula, we formally define formulas as sequences, rather than multisets, to simplify some of the counting.

A (*partial*) *truth assignment* to the set of literals L_n is a (partial) mapping $\tau : L_n \to \{\mathsf{t}, \mathsf{f}\}$ such that $\tau(\overline{v}) = \mathsf{t}$ if and only if $\tau(v) = \mathsf{f}$. A clause C has truth value t under τ if and only if some literal in C is assigned value t. A formula $\mathcal{F}$ has truth value t under τ if and only if every clause in $\mathcal{F}$ is assigned value t. A formula $\mathcal{F}$ is *satisfiable* if and only if there exists a partial truth assignment τ that assigns t to $\mathcal{F}$. Such a τ is said to *satisfy* $\mathcal{F}$. The objective of an algorithm for SAT is to determine whether a given formula is satisfiable.

Definition 2.2: The probability space $\mathcal{M}^k_{m,n}$ is defined as follows. Let $\mathcal{C}^k_n$ be the set of all clauses with exactly k variable-distinct literals from L_n. (Note that $|\mathcal{C}^k_n| = 2^k\binom{n}{k}$.) A random formula in $\mathcal{M}^k_{m,n}$ is a sequence of m clauses from $\mathcal{C}^k_n$, selected uniformly, independently, and with replacement. □

We are primarily interested in the cases $k \geq 3$ since random formulas generated from $\mathcal{M}^k_{m,n}$, $k \leq 2$, are solved in linear time by existing 2-SAT algorithms [4, 20]. In all cases k remains constant as n and m grow.

Definition 2.3: Given a formula $\mathcal{F}$, its *clause-variable matrix*, denoted as $\mathbf{F}$, is the $m \times n$ matrix in which element (i, j) has the value $+1$ if clause C_i has positive literal v_j, has the value -1 if clause C_i has negative literal $\overline{v_j}$, and has the value 0 otherwise. □

The clause-variable matrix representation of $\mathcal{F}$ is used in Section 3 for characterizing and/or processing several classes of CNF formulas. For example, satisfiability may be cast as an integer linear programming problem, using this matrix.

The analysis makes extensive use of the following notation:

$$n^{\underline{q}} \;=\; \prod_{i=0}^{q-1}(n-i) = n(n-1)(n-2)\cdots(n-q+1) = \frac{n!}{(n-q)!} \tag{1}$$

We read $n^{\underline{q}}$ as "n to the q falling". Some texts use the notation $(n)_q$. A useful bound is given by

$$n^q\, e^{-q^2/2n} \;<\; n^{\underline{q}} \leq n^q \quad \text{for } q < \frac{n}{2} \tag{2}$$

3 Polynomial-Time Solvable Classes of SAT

This section discusses certain classes of SAT that are solvable in polynomial time. Except for the "matched" and "LinAut" classes, these are the classes that we have referred to as "well known". Use of the clause-variable matrix, **F** (Definition 2.3) serves to clarify the relationships among several of the classes.

Definition 3.1: A formula $\mathcal{F}$ is *Horn* if and only if every row of **F** has at most one $+1$ value; that is, every clause has at most one positive literal. □

Definition 3.2: (Lewis [34]) A formula $\mathcal{F}$ is *renamable Horn* if and only if multiplying some subset of columns of **F** by -1 yields a Horn formula. Multiplying column j by -1 is called *renaming* v_j, and corresponds to mapping v_j into $\overline{v_j}$ and *vice versa*. □

Horn formulas can be solved in linear time by unit resolution [19, 31, 38]. Renamable Horn formulas can also be solved in linear time [5].

Definition 3.3: Let a formula $\mathcal{F}$ with m clauses and n variables be given, and let **F** be its clause-variable matrix. Let $|C_i|$ denote the number of literals in clause C_i, and let w be the m-vector whose components are $|C_i|$. Let x be a real n-vector. Let **1** be the vector of 1's whose dimension is implied by the context. Construct the following system of real inequalities:

$$\mathbf{F}\,\mathbf{x} \;\leq\; 2\,Z\,\mathbf{1} - w, \qquad -\mathbf{1} \leq x \leq \mathbf{1}. \tag{3}$$

If all these constraints can be satisfied for $Z = 1$, then the formula is *q-Horn*. □

The class q-Horn was developed by Boros *et al.* [8, 9]. If (3) can be satisfied with $Z = 1$ and all the x_j's are 1, then it is easy to see that the formula is Horn. If $Z = 1$ and all the x_j's are ± 1, then it is renamable Horn. The q-Horn class is interesting for having the following property.

Theorem 3.1: [9] For any fixed $c > 0$, the class of formulas that satisfy the constraints (3) with

$$Z \;=\; 1 + \frac{c \log n}{n} \tag{4}$$

can be solved in polynomial time.

For any fixed $\beta < 1$, the class of formulas that satisfy (3) with

$$Z \;=\; 1 + \frac{1}{n^{\beta}} \tag{5}$$

is NP-complete. □

Very recently, a new polynomial-time solvable class named *LinAut* has been introduced [33]. Its definition involves the notion of linear autarky. We simplify the definition by restricting it to formulas with at least three literals per clause.

Definition 3.4: Let a formula $\mathcal{F}$ be given and use the same notation as in Definition 3.3 for m, n, and $\mathbf{F}$. Let $\mathbf{0}$ be the vector of 0's whose dimension is implied by the context.

A *linear autarky* is a nonzero real n-vector x such that

$$\mathbf{F}\,\mathbf{x} \;\geq\; \mathbf{0} \tag{6}$$

The class *LinAut* for formulas with at least three literals per clause is defined inductively:

1. The empty formula is in LinAut.
2. Suppose $\mathcal{F}$ has a linear autarky x. Define *linSat*$(\mathcal{F}, x)$ to be the set of clauses C_i in $\mathcal{F}$ such that some literal v_j or $\overline{v_j}$ in C_i corresponds to a nonzero component x_j of x; that is, the clauses that are "touched" by x. Let $\mathcal{F}' = \mathcal{F} - $ *linSat*$(\mathcal{F}, x)$; that is, the clauses that are *not* "touched" by x. If $\mathcal{F}'$ is in LinAut then so is $\mathcal{F}$.

□

The significance of the definition is that *linSat*$(\mathcal{F}, x)$ can be satisfied by a partial assignment A that does not affect $\mathcal{F}'$, so that $\mathcal{F}$ is satisfiable if and only $\mathcal{F}'$ is satisfiable. The partial assignment A is constructed simply by assigning v_j to $\mathtt{t}$ if $x_j > 0$ and to $\mathtt{f}$ if $x_j < 0$. If C_i is in *linSat*$(\mathcal{F}, x)$, then at least one of its literals is assigned to $\mathtt{t}$ because $\mathbf{F_{ij}x_j}$ cannot be negative in all of the cases that it is nonzero [33]. If C_i is in $\mathcal{F}'$, then none of its literals are assigned a value by A.

A well-known special case of the linear autarky is the pure literal rule: let $x_j = 1$ if v_j occurs only positively and let $x_j = -1$ if v_j occurs only negatively, otherwise let $x_j = 0$.

The class *SLUR* (Single Lookahead Unit Resolution solvable) is peculiar in that it is defined using a nondeterministic algorithm, and not by structural properties of formulas. Since the definition refers to all possible runs of the nondeterministic algorithm, the class is not apparently recognizable in polynomial time; however, the algorithm may be used without deciding whether the formula is in the class SLUR. The nondeterministic algorithm named SLUR is given in Figure 1. In it, the function *unitprop*$(\mathcal{F})$ returns a *pair* of values $(\mathcal{F}', t)$, where $\mathcal{F}'$ is the formula that results from repeatedly performing *unit resolution* until no unit clauses remain in the formula and t is the set of unit clauses found and derived. It is known that *unitprop* can be implemented in time linear in $|\mathcal{F}|$ [18].

Definition 3.5: A formula is in the class SLUR if, for all possible sequences of selected variables, algorithm SLUR does not give up. Algorithm SLUR eventually gives up if it starts with, or creates, an unsatisfiable formula with no unit clauses. □

Algorithm $\mathbf{SLUR}(\mathcal{F})$

Input: A CNF formula $\mathcal{F}$ with no empty clause

Output: A satisfying partial truth assignment for the variables in $\mathcal{F}$, or "unsatisfiable", or "give up"

$(\mathcal{F}, t) := \textit{unitprop}(\mathcal{F})$.
If $\emptyset \in \mathcal{F}$, then
  Output "unsatisfiable" and halt.
While $\mathcal{F}$ is not empty do the following:
  Select a variable v appearing as a literal of $\mathcal{F}$.
  $(\mathcal{F}_1, t_1) := \textit{unitprop}(\mathcal{F} \cup \{[\overline{v}]\})$.
  $(\mathcal{F}_2, t_2) := \textit{unitprop}(\mathcal{F} \cup \{[v]\})$.
  If $\emptyset \in \mathcal{F}_1$ and $\emptyset \in \mathcal{F}_2$, then
    Output "give up" and halt.
  Otherwise, if $\emptyset \in \mathcal{F}_1$, then
    $(\mathcal{F}, t) := (\mathcal{F}_2,\, t \cup t_2)$.
  Otherwise, if $\emptyset \in \mathcal{F}_2$, then
    $(\mathcal{F}, t) := (\mathcal{F}_1,\, t \cup t_1)$.
  Otherwise, arbitrarily do one of the following.
   1. $(\mathcal{F}, t) := (\mathcal{F}_1,\, t \cup t_1)$.
   2. $(\mathcal{F}, t) := (\mathcal{F}_2,\, t \cup t_2)$.
  (Continue the loop.)
Output t and halt.

Figure 1: The **SLUR** algorithm. Notation $(\mathcal{F}, t)$ is explained in the text.

Algorithm SLUR takes linear time with the modification, due to Trümper [40], that unit resolution (in *unitprop*) be applied simultaneously to both branches of a selected variable, abandoning one branch if the other finishes first without deriving an empty clause. Note that due to the definition of this class, the question of class recognition is unresolved. If one run of SLUR returns "unsatisfiable" we know the formula is in SLUR; if one run returns "give up" we know the formula is not in SLUR; otherwise, the formula is satisfied, but we have no information about membership in SLUR.

The class SLUR was developed as a generalization of other classes including Horn, renamable Horn, extended Horn, and CC-balanced formulas [37]. We refer the interested reader to the literature for definitions of extended Horn [13] and CC-balanced [17].

The last class we consider is based on matchings in a particular bipartite graph developed from a CNF formula $\mathcal{F}$. This class can be recognized in polynomial time.

Definition 3.6: Given a CNF formula $\mathcal{F}$ with clause-variable matrix $\mathbf{F}$, we define $\mathcal{G}(\mathcal{F})$ to be an undirected bipartite graph $\mathcal{G}(\mathcal{F})$ with vertex sets $\mathcal{C}$ and V, where the elements of $\mathcal{C}$ are the clauses of formula $\mathcal{F}$, and the elements of V are the variables of $\mathcal{F}$. An edge (v_j, C_i) exists if and only if the variable v_j is in clause C_i either as a positive literal or a negative literal; that is, $\mathbf{F_{ij}}$ is nonzero.

A *total matching for clauses* in $\mathcal{G}(\mathcal{F})$ is a subset M of its edges such that every vertex in $\mathcal{C}$ is incident upon exactly one edge of M, and every vertex in V is incident upon *at most* one edge of M. Formula $\mathcal{F}$ is said to be *matched* if and only if $\mathcal{G}(\mathcal{F})$ has a total matching for clauses. □

A total matching for clauses can be computed for $\mathcal{G}(\mathcal{F})$ in polynomial time, if one exists, by an augmenting path algorithm. If a total matching for clauses exists, then $\mathcal{F}$ is satisfiable (see Lemma 8.1). This observation was credited to Adam Rosenberg by Tovey [39].

In the analysis below we will not directly consider some of the classes defined above because they are subclasses of either SLUR or q-Horn. However, the classes of matched formulas, SLUR, and q-Horn formulas are incomparable as the following examples show.

Example 1: The formulas below show that none of the three classes contains another class, among matched, SLUR, and q-Horn. These formulas can be embellished to make the same point even if the SLUR algorithm is modified to exclude such trivial counter-examples (e.g., by checking for pure literals and/or by checking for the empty formula).

1. Any Horn formula with more clauses than distinct variables is not in the matched class, but is in both SLUR and q-Horn classes.
2. The following formula is in the matched and q-Horn classes.

$$[v_1, \overline{v_2}, v_4], [v_1, v_2, v_5], [\overline{v_1}, \overline{v_3}, v_6], [\overline{v_1}, v_3, v_7]$$

   However, it is not SLUR (see Lemma 6.2). In particular, initially choosing v_4, v_5, v_6, v_7, and choosing false values at arbitrary choices, leaves an unsatisfiable formula with no unit clauses. (To verify q-Horn membership, set $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{2}$ and the remaining α's to 0 in (3).)

3. The following formula is in the matched and SLUR classes.

$$[\overline{v_1}, v_2, \overline{v_4}], [\overline{v_0}, \overline{v_2}, v_3], [\overline{v_3}, v_0, v_1]$$

However, it is not q-Horn (see Lemma 7.1). In particular, $Z \geq 4/3$ is needed to make the inequalities (3) satisfiable. (To verify SLUR membership, note that in the SLUR algorithm, no choice sequence leads to an unsatisfiable formula without unit clauses.)

The above ideas can be combined to show that no class is contained in the union of the other two classes. □

Van Maaren further showed that LinAut properly contains the q-Horn class, whereas LinAut and SLUR are incomparable [35]. Although various classes are incomparable, we shall show that matched formulas are more common than SLUR or q-Horn formulas under $\mathcal{M}^k_{m,n}$. We shall also show that LinAut properly contains the matched class.

4 Overview of the Second Moment Method

The *second moment method* is (among other applications [3]) a tool for proving that a specified property P holds on a class of random structures with high probability as the size parameter of the class tends to ∞. Its early applications in combinatorial analysis were on random graphs [7], and more recently it has been applied to random formulas [29, and elsewhere]. We shall use it several times in this paper.

For a given random structure (in our case, a CNF formula $\mathcal{F}$ in the class $\mathcal{M}^k_{m,n}$), a *witness* w is a substructure (in our case, a set of clauses, $w \subseteq \mathcal{F}$) whose presence implies that the structure has property P. Let W be the set of all possible witnesses for the class. The technique is to prove that the probability that a randomly chosen $\mathcal{F}$ *fails to contain any witness* approaches zero.

Let w denote a witness, and also represent the *event* that $w \subseteq \mathcal{F}$; which meaning is intended will be clear from context. Usually the set of witnesses W is chosen so its elements are *symmetric*; i.e., for any pair $w, z \in W$, there is an automorphism of the probability space that maps w to z. We shall assume this is the case in this section. Thus $p = Pr(w)$ is independent of w. Let I_w be the *indicator random variable* that is 1 if event w occurs and 0 otherwise.

$$E(I_w) = p \quad \text{and} \quad \mathrm{var}(I_w) = E((I_w - p)^2) = p(1-p).$$

Now define the random variable $I = \sum_{w \in W} I_w$, and let $\mu = E(I) = |W|p$ and $\sigma^2 = \mathrm{var}(I) = E\left((I-\mu)^2\right)$. A special case of the Chebyshev inequality states that

$$Pr(I = 0) \quad \leq \quad \frac{\sigma^2}{\mu^2} \tag{7}$$

Thus it suffices to show that this ratio approaches zero as the size parameter goes to infinity (in our case, the number of variables, $n \to \infty$).

Notice that, if the events w were independent, then $\sigma^2 = |W|p(1-p) = O(\mu)$, and it would be sufficient to show that $\mu \to \infty$ as $n \to \infty$. The crux of the second moment method is to show that, although the events w are not independent, the dependencies are weak enough that $\sigma^2 = o(\mu^2)$.

To analyze the expression for σ^2, we introduce the notation:

$A(w)$ is the set of witnesses having some clause in common with w, other than w itself;
$D(w)$ is the set of witnesses having no clause in common with w.

Let w be an arbitrary witness; by symmetry, the choice is immaterial. It follows that:

$$\sigma^2 = \mu(1-p) + \mu \left(\sum_{z \in A(w)} (Pr(z|w) - p) + \sum_{z \in D(w)} (Pr(z|w) - p) \right) \tag{8}$$

We can now state a basic lemma of the second moment method, which pinpoints what we need to prove when $|A(w)| \ll |W|$.

Lemma 4.1: (Alon and Spencer [3, Ch. 4.3, Cor. 3.5]) With the above notation, if:

1. elements of W are symmetric,
2. $\mu \to \infty$ as $n \to \infty$,
3. $\sum_{z \in A(w)} Pr(z|w) = o(\mu)$ for an arbitrary $w \in W$,

then $Pr(P) \to 1$ as $n \to \infty$. □

5 Propositional Connection Graphs

Propositional connection graphs turn out to be useful for analysis of several classes of random formulas. They are a specialization of the more general first-order connection graphs developed by Kowalski [32].

Definition 5.1: Let A and B be clauses and let x be a literal such that x is in A and $\overline{x}$ is in B. Then the *resolvent on x* of A and B is the clause $C = (A - [x]) \cup (B - [\overline{x}])$. The *clashing literals* are x and $\overline{x}$. If C does not contain a complementary pair of literals, the resolvent is said to be *non-tautologous*.

Definition 5.2: In the *propositional connection graph* for a CNF formula, each clause is a node. Two nodes are connected by an undirected edge if and only if they have a non-tautologous resolvent. Thus, two nodes are adjacent if their clauses contain exactly one pair of complementary literals, say u in one clause and $\overline{u}$ in the other. Technically, this edge is labeled by the pair $(u, \overline{u})$, although only one member of the pair may be mentioned, for brevity. We shall abbreviate "propositional connection graph" to "connection graph" in this paper. □

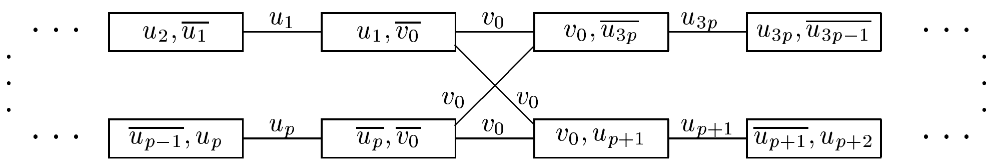

Figure 2: Connection graph for a "criss-cross loop" of $t = 3p + 2$ clause nodes. Only "cycle literals" are shown in the nodes; "padding literals" are present but not shown for $k \geq 3$. See text for discussion.

In the case of 2-CNF, the connection graph has a close relationship to the *implication graph* [4], but is not identical. Recall that in an implication graph, the literals are nodes and each binary clause $[u_i, u_j]$ is a pair of directed edges that interpret the clause as two implications: $\overline{u_i} \to u_j$ and $\overline{u_j} \to u_i$. However, in the connection graph each node is a clause and the edges are undirected. Consequently, the *connection graph* is easily generalized to k-CNF, whereas the *implication graph* is not so easily generalized.

We now discuss the criss-cross loop, which is a structure in k-CNF that will be used for analysis of the SLUR class (Section 6). A slight variation will be used for the q-Horn class (Section 7).

Definition 5.3: For any integer $t = 3p + 2 \geq 5$, a set $\mathcal{S}$ of t clauses is called a *criss-cross loop* if the connection graph of $\mathcal{S}$ is isomorphic to that shown in Figure 2 and its clauses have the following form, where the R_i denote sets of $k - 2$ literals, v_0 is a variable and the u_j are literals:

$$\begin{aligned}&[\overline{v_0}, u_1, R_0]\,,\ [\overline{u_1}, u_2, R_1]\,,\ \ldots,\ [\overline{u_{p-1}}, u_p, R_{p-1}]\,,\ [\overline{u_p}, \overline{v_0}, R_p]\,,\\ &[v_0, u_{p+1}, R_{p+1}]\,,\ [\overline{u_{p+1}}, u_{p+2}, R_{p+2}]\,,\ \ldots,\ [\overline{u_{3p-1}}, u_{3p} R_{3p}]\,,\ [\overline{u_{3p}}, v_0, R_{3p+1}]\end{aligned} \quad (9)$$

The literals $u_1, \ldots, u_{3p}$ are variable-distinct (Section 2) from each other and from v_0, which is called the *criss-cross variable*. The literals v_0, u_1, $\ldots$, u_{3p} and their complements are called the *cycle literals*; they appear explicitly in Figure 2.

Furthermore, the literals in the R_i, called the *padding literals*, are variable-distinct from each other and from the cycle literals. The set of padding literals is denoted as $R(\mathcal{S})$ and it contains $(k - 2)t$ literals, recalling that $t = 3p + 2$. Figure 2 does not show the padding literals. □

Observe that the graph in Figure 2 has two Hamiltonian cycles. If each node represents a binary clause, we have an unsatisfiable set of clauses. This formula contains $t - 1$ distinct variables and t clauses, and it is minimally unsatisfiable. It is closely related to the *simple cycle*, as defined by Goerdt [29].

One automorphism of the formula $\mathcal{S}$, other than the identity, corresponds to Figure 2:

$$\{u_j \leftrightarrow \overline{u_{4p+1-j}} \mid p + 1 \leq j \leq 3p\}\,.$$

(I.e., vertically flip all nodes to the right of the criss-cross edges.) Requiring v_0 to be a variable reduces the number of automorphisms. Also, note that v_0 has a distinguished role, being the only literal to label more than one edge.

6 SLUR Analysis

This section shows that the probability of a SLUR formula occurring at random approaches zero for fairly small ratios $r = m/n$. The critical substructure is a refinement of that introduced in Definition 5.3.

Definition 6.1: A criss-cross loop $\mathcal{S}$ is said to be *activated* in a formula $\mathcal{F}$ if every clause $C \in (\mathcal{F} - \mathcal{S})$ contains at least two literals that are variable-distinct from those in $R(\mathcal{S})$, the padding literals of $\mathcal{S}$, as defined in Definition 5.3. □

The motivation for the definition of "activated criss-cross loop" is that we can do unit resolution on each literal in $R(\mathcal{S})$ without creating any unit clause or empty clause.

Definition 6.2: If a set of clauses $\mathcal{S}$ defines a criss-cross loop with v_0 as the criss-cross variable, then a *representative sequence* for $\mathcal{S}$ is a Hamiltonian path (i.e., a simple path that includes all nodes in the connection graph) that begins at the clause $[\overline{v_0}, u_1, \ldots]$ (upper left center) and ends at the clause $[\overline{u_{3p}}, v_0, \ldots]$ (upper right center), in Figure 2. The Hamiltonian path can be extended to a Hamiltonian cycle by returning to the initial node.

Associated with each representative sequence is a *literal sequence*, from which the representative sequence can be constructed by a fixed mapping. In Figure 2,

1. (v_0, u_1, …, u_p, u_{p+1}, …, u_{t-2}) maps into the representative sequence that traverses all the nodes with an exterior loop;
2. (v_0, u_1, …, u_p, $\overline{u_{t-2}}$, …, $\overline{u_{p+1}}$) maps into the representative sequence that traverses all the nodes with a "figure eight".

Thus, a given literal sequence uniquely defines the placements of all cycle literals (including the criss-cross variable) in a criss-cross loop. Every criss-cross loop is defined (except for padding literals) by exactly two literal sequences. □

The connection graph of a criss-cross loop has exactly two Hamiltonian cycles, two representative sequences, and two literal sequences. In Figure 2 one may traverse all the nodes with an exterior loop, or with a "figure eight".

Theorem 6.1: In the class $\mathcal{M}^k_{m,n}$, with $m/n = r > 2/(k(k-1))$, and $k \geq 3$, the probability that a random formula $\mathcal{F}$ is in the class SLUR tends to 0 as n tends to ∞.

Proof: Immediate by Lemmas 6.2 and 6.3, which follow. □

Lemma 6.2: If a formula $\mathcal{F}$ in k-CNF has an activated criss-cross loop $\mathcal{S}$, then $\mathcal{F}$ is not in the class SLUR.

Proof: In algorithm SLUR (see Fig. 1), first select the literals from $R(\mathcal{S})$ for unit resolution. For each such literal, neither truth assignment can produce $\emptyset$, and no additional unit clauses are produced, so make the arbitrary choice that the literal from $R(\mathcal{S})$ is false (i.e., $\mathcal{F}_1$ and t_1), and continue down the search tree. When $R(\mathcal{S})$ is exhausted, the binary clauses derived from $\mathcal{S}$ remain and are unsatisfiable, so algorithm SLUR must eventually "give up". □

Lemma 6.3: In the class $\mathcal{M}_{m,n}^k$, with $m/n = r > 2/(k(k-1))$, and $k \geq 2$, the probability that a random formula $\mathcal{F}$ contains an activated criss-cross loop tends to 1 as n tends to ∞.

Proof: We apply the second moment method. Let $p = \lceil \ln^2 n \rceil$, and $t = 3p + 2$, assuming $n \geq 2$. Let B_t denote the number of activated criss-cross loops with t clauses in a random formula $\mathcal{F}$. Note that such a subformula has $t-1$ cycle variables, including the *criss-cross variable.* We show that $E(B_t) > \alpha^t$, for some $\alpha > 1$ when $r > 2/(k(k-1))$. Then we show that Lemma 4.1 applies under the same conditions, where P is the property that $B_t > 0$.

First, we find a lower bound on $E(B_t)$. Fix a literal sequence (see Definition 6.2 and Figure 2) consisting of $t-1$ cycle literals, $v_0, u_1, u_2, \ldots, u_{t-2}$. Now choose a sequence of t clauses independently, with replacement, from $\mathcal{C}_n^k$, and call it $\mathcal{S}$. The probability that this clause sequence is a representative sequence for some criss-cross loop (Definition 6.2), and is associated with the fixed literal sequence is:

$$p_1(n,k,t) \quad = \quad \prod_{i=0}^{t-1} \left(\frac{2^{k-2}\binom{n-t+1-(k-2)i}{k-2}}{2^k \binom{n}{k}} \right) = \left(\frac{k(k-1)}{4n^{\underline{k}}} \right)^t (n-t+1)^{\underline{(k-2)t}} \qquad (10)$$

where $n^{\underline{k}}$, etc., use the notation of (1). To derive (10), observe that the numerator of the i-th product term is the number of ways to choose $(k-2)$ padding literals that are variable-distinct from the cycle literals and the padding literals occurring in previous clauses. The two cycle literals for the i-th clause are fixed by the literal sequence.

The number of ways to select a sequence of t clauses (without replacement) from the m clauses of $\mathcal{F}$ is $m^{\underline{t}}$. The number of ways to choose a literal sequence of length $t-1$ is $2^{t-2}\, n^{\underline{t-1}}$, since the first literal must be positive. The total ways for $\mathcal{F}$ to contain a representative sequence of length t is therefore

$$m^{\underline{t}}\ 2^{t-2}\ n^{\underline{t-1}} \qquad (11)$$

Assume that a specific criss-cross loop $\mathcal{S}$ is present. We now estimate the probability that it is *activated.* Using the fact that

$$\frac{b^{\underline{j}}}{c^{\underline{j}}} < \frac{b^j}{c^j} \quad \text{when b < c,}$$

we find that the probability that a specific clause of $\mathcal{F} - \mathcal{S}$ has fewer than two literals that are variable-distinct from the padding literals $R(\mathcal{S})$ is at most $k\left(\frac{t(k-2)}{n}\right)^{k-1}$. Therefore,

$$Pr(\mathcal{S} \text{ is activated} \mid \mathcal{S} \text{ is present}) \quad > \quad \left(1 - O\left(\frac{(k-2)t}{n}\right)^{k-1}\right)^{m-t} = 1 - O\left(\frac{(k-2)t^{k-1}}{n^{k-2}}\right) \qquad (12)$$

Combining (10), (11), (12), and recalling that each criss-cross loop has two representative sequences:

$$E(B_t) \quad > \quad \left(\frac{(k(k-1))^t\ (n-t+1)^{\underline{(k-2)t}}}{4^t\ (n^{\underline{k}})^t} \right) \left(\frac{m^{\underline{t}}\ 2^{t-2}\ n^{\underline{t-1}}}{2} \right) \left(1 - O\left(\frac{t^{k-1}}{n^{k-2}} \right) \right)$$

$$> \quad \frac{n^{\underline{t}}\ n^{\underline{(k-1)t-1}}}{8n^t\ n^{(k-1)t}} \left(\frac{rk(k-1)n^k}{2n^{\underline{k}}} \right)^t \left(1 - O\left(\frac{t^{k-1}}{n^{k-2}} \right) \right)$$

$$> \frac{1}{8n}\left(\frac{rk(k-1)}{2}\right)^t\left(1-O\left(\frac{t^2}{n}\right)\right) \tag{13}$$

where (2) was used for the last estimate.

If $r > 2/k(k-1)$, then let $c = \ln(rk(k-1)/2) > 0$, and we get

$$\ln(E(B_t)) > c(3\ln^2(n)+2) - \ln(n) - 3,$$

which goes to infinity with n.

The next task is to apply Lemma 4.1. We need to obtain an upper bound on $Pr(z|w)$, the conditional probability that a new witness, z, is in $\mathcal{F}$ given that a reference witness, w, is present and they overlap on at least one clause. In this context a "witness" is actually an activated criss-cross loop, but since an upper bound is sufficient, we will ignore the restriction that it be activated, and we will also count each of the two *representative sequences* that represent the criss-cross loop as separate witnesses. Thus w is an arbitrary fixed representative sequence of t clauses, and z ranges over representative sequences of t clauses.

The set $A(w)$ denotes the set of possible witnesses that share at least one clause with w. This set is partitioned into $A_q(w)$, according to the number of shared clauses, q, $1 \le q \le t$.

Suppose the new witness z shares q clauses with the reference witness w, i.e., $z \in A_q(w)$. Once the nonshared clauses of z are fixed, the probability that the nonshared clauses all occur in $\mathcal{F}$ is bounded by the expectation of the sum of appropriate indicator random variables:

$$\begin{aligned} Pr(z|w) &\le (m-t)^{\underline{t-q}}\left(\frac{1}{2^k\binom{n}{k}}\right)^{t-q} \\ &< \left(\frac{mk!}{2^kn^k}\right)^{t-q} = \left(\frac{rk!}{2^kn^{k-1}}\right)^{t-q} \end{aligned} \tag{14}$$

This is about a factor of $n^{(k-1)q}$ larger than the unconditional probability that witness z occurs.

We now obtain an upper bound on how many representative sequences share q nodes with w. Let S denote the set of shared clauses of w, or the corresponding connection graph of shared clauses, depending on context. Let z denote a new witness that shares S with w. All edge labels of S are preserved in z; their number depends on the structure of S. In all cases $(k-2)(t-q)$ padding literals for the non-shared clauses can be chosen among unused variables, in fewer than

$$\left(2^{(k-2)}\binom{n}{k-2}\right)^{t-q} \tag{15}$$

ways. To count the ways in which cycle literals may be chosen, we classify the cycle literals of z as *fixed*, *limited*, or *free*:

1. A cycle literal is *fixed* if it labels an edge of S; there is one choice.

2. A cycle literal is *limited* if any edge that it labels in the connection graph of z joins a shared clause with a non-shared clause; there are at most k choices for each *limited* cycle literal.
3. A cycle literal is *free* if all edges that it labels in the connection graph of z join two non-shared clauses; there are about $2n$ choices for each *free* cycle literal.

Clearly, the number of free cycle literals is the major factor in counting the ways to choose cycle literals in z. The case in which the unshared clauses of w (i.e., $w - S$) consist of one connected component is considered separately from the case of multiple connected components, then the upper bound over all cases is used.

Suppose $(w - S)$ has a single connected component. If at least three of the four clauses of w that contain v_0 or $\overline{v_0}$ are in S, then the role of v_0 must be the same in the new witness z. (There are up to four ways to map the clauses with v_0 or $\overline{v_0}$ into z.) In Figure 2 either all nodes to the left of the criss-cross edges are in S or all nodes to the right are in S; otherwise, $w - S$ has multiple components. But this means $|S| = q > p + 2$. There are only 4 ways to map edges from S into the connection graph of z. If $q = t$, there is no other flexibility in z. For $p + 1 < q \le (t - 1)$, there are at least $(q - 2)$ cycle literals, so there is a sequence of at most $(t - q + 1)$ cycle literals that are not edge labels in S. In the new witness z, two of these are limited and at most $(t - q - 1)$ are free (as defined above).

Still supposing $(w - S)$ has a single connected component, if at most two of the four clauses of w that contain v_0 or $\overline{v_0}$ are in S, then the shared clauses of w can be mapped into fewer than t different subsequences of the representative sequence z, with two choices of direction. (Other mappings would not preserve the edge labels of S; in particular, "doubling back" would use two edges with the same label, which are not present in S.) However, S now fixes $(q - 1)$ cycle literals (because all edges must be labeled with distinct variables). That leaves two limited and only $(t - q - 2)$ free. The resulting bound is smaller, but for $q \le p + 2$, the case of the previous paragraph cannot occur. Therefore, the upper bounds for the case in which $w - S$ has one connected component, can be summarized as:

$$|A_q(w)| \le 4t\ k^2 (2n)^{(t-q-2)} \left(2^{(k-2)} \binom{n}{k-2}\right)^{t-q} < \frac{t\ k^2\ (2n)^{(k-1)(t-q)}}{n^2\ ((k-2)!)^{t-q}} \quad \text{for } q \le p+3 \tag{16}$$

$$|A_q(w)| \le 4\ k^2 (2n)^{(t-q-1)} \left(2^{(k-2)} \binom{n}{k-2}\right)^{t-q} < \frac{k^2\ (2n)^{(k-1)(t-q)}}{n\ ((k-2)!)^{t-q}} \quad \text{for } q > p+3 \tag{17}$$

Now suppose $(w - S)$ has $h \ge 2$ connected components. Again consider two cases. If at least three of the four clauses of w that contain v_0 or $\overline{v_0}$ are in S, then the role of v_0 must be the same in the new witness z, and S has $(h - 1)$ connected components. Recall that edge labels of S must be preserved in z. There are $(q - h)$ or $(q - h - 1)$ distinct edge labels in S due to the multiplicity of v_0, the criss-cross variable. These fix the associated cycle literals. In the connection graph of z there are at least $2h$ limited cycle literals. Therefore, at most $(t - q - h)$ cycle literals of z are free, in this case. There are four ways to map the first connected component of S, which contains the v_0 edges, into z, and at most $2^{h-2}\ t^{2(h-2)}$ ways to map the remaining connected components of S into z. Alternatively, if at most two of the four clauses of w that contain v_0 or $\overline{v_0}$ are in S, then S also has h connected components, and $(q - h)$ distinct edge labels, thereby fixing $(q - h)$ cycle literals of z. There are still $2h$ limited literals, which can be chosen in at most k^{2h} ways, leaving $(t-q-h-1)$ free. In this case $2^h\ t^{2h}$ is an upper bound on the number of ways to map the connected components of S into z *without* using any criss-cross edges in z. Because all edge labels of S are distinct, using criss-cross edges of z only doubles this number. That is, at most one edge labeled with the criss-cross variable of z can be mapped into from S. The upper bound for all of the cases with $h \geq 2$ occurs when $h = 2$, and is $16\ k^4\ t^4\ (2n)^{t-q-3}$.

In summary, for $h \geq 2$ connected components in $(w - S)$, we have

$$|A_q(w)| \quad \leq \quad 16\ k^4\ t^4\ (2n)^{(t-q-3)} \ \left(2^{(k-2)}\binom{n}{k-2}\right)^{t-q} < \frac{(2n)^{(k-1)(t-q)}}{n^2\ ((k-2)!)^{t-q}} \qquad (18)$$

which is smaller than the case $h = 1$.

Multiply the upper bounds for $|A_q(w)|$ and $Pr(z|w)$:

$$|A_q(w)|Pr(z|w) \quad < \quad \frac{t\ k^2}{n^2}\left(\frac{rk(k-1)}{2}\right)^{t-q} \qquad \text{for } q \leq p+3 \qquad (19)$$

$$|A_q(w)|Pr(z|w) \quad < \quad \frac{k^2}{n}\left(\frac{rk(k-1)}{2}\right)^{t-q} \qquad \text{for } q > p+3 \qquad (20)$$

Combining (13) with the above, and using the assumption that $r > 2/k(k-1)$, we obtain the result:

$$\begin{aligned}\sum_{z \in A(w)} Pr(z|w) &= \sum_{q=1}^{p+3} |A_q(w)|Pr(z|w) + \sum_{q=p+4}^{t} |A_q(w)|Pr(z|w) \\ &< O\left(\frac{t}{n^2}\left(\frac{rk(k-1)}{2}\right)^{t} + \frac{1}{n}\left(\frac{rk(k-1)}{2}\right)^{t-p-3}\right) \qquad (21) \\ &= O\left(\frac{t}{n} + \left(\frac{rk(k-1)}{2}\right)^{-p-3}\right)\ E(B_t) = o(E(B_t))\end{aligned}$$

The theorem follows by Lemma 4.1. □

Corollary 6.4: (Goerdt) In $\mathcal{M}^2_{m,n}$, with $m/n = r > 1$, the probability that a random formula $\mathcal{F}$ is unsatisfiable tends to 1 as n tends to ∞.

Proof: A criss-cross loop in 2-CNF is unsatisfiable. □

7 Q-Horn Analysis

This section shows that the probability of a Q-Horn formula occurring at random approaches zero for the same fairly small ratios as SLUR, namely $r = m/n > 2/(k(k-1))$. The analysis supports a slightly stronger conclusion, which is discussed at the end of the section. First, the critical substructure is introduced.

Definition 7.1: For any integer $t = 3p \geq 6$, call a set of t clauses a *chorded loop* if its connection graph is isomorphic to that shown in Figure 3. This graph corresponds to a

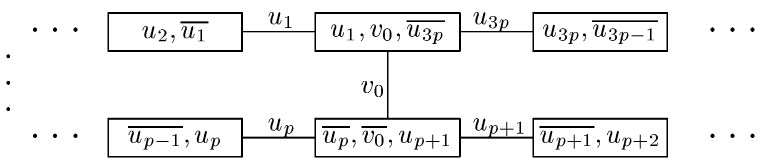

Figure 3: Connection graph for a "chorded loop" of $t = 3p$ clause nodes. Only "cycle literals", including the "chord variable" v_0, are shown in the nodes; "padding literals" are also present but not shown. See text for discussion.

formula in which all but two literals can be removed from each of $t-2$ clauses, all but three literals can be removed from two clauses, the variables can be renumbered, and the clauses can be ordered in the following sequence

$$\begin{aligned} &[\overline{u_1}, u_2]\,,\ \ [\overline{u_2}, u_3]\,,\ \ \ldots,\ \ [\overline{u_{p-1}}, u_p]\,,\ \ [\overline{v_0}, \overline{u_p}, u_{p+1}]\,, \\ &[\overline{u_{p+1}}, u_{p+2}]\,, \ldots, [\overline{u_{3p-1}}, u_{3p}]\,, [\overline{u_{3p}}, v_0, u_1] \end{aligned} \tag{22}$$

where v_0 and the u_i are all variable-distinct from each other (Section 2). Variable v_0 is called the the *chord variable*. As before with criss-cross loops, literals v_0 and u_i are called *cycle literals*, and those not shown are called *padding literals*. Padding literals must not produce any additional edges in the connection graph. Requiring v_0 to be a variable reduces the number of automorphisms, and it has the distinguished role as the label of the only edge connecting two nodes of degree three in the connection graph.

Given a chorded loop $\mathcal{S} \subset \mathcal{F}$, if none of the padding literals are the same or complementary, then $\mathcal{S}$ is called a *q-blocked chorded loop*. $\square$

It is easily determined, by inspecting both renamings of v_0 (Definition 3.2), that a chorded loop is not Horn renamable. The q-Horn class is a generalization of the Horn renamable class, but a chorded loop is not q-Horn, either, as shown next.

Lemma 7.1: If a formula $\mathcal{F}$ has a chorded loop then it is not q-Horn.

Proof: Let $\mathcal{S}$ be a chorded loop in $\mathcal{F}$ with $t = 3p$ clauses. Letting real variables α_i correspond to chord variable v_0 and cycle literals u_i. develop inequalities (3) for $\mathcal{S}$. Without loss of generality, assume all padding literals are negative, and their real variables are set equal to 1. (If this is impossible, the constraint on Z derived below is just stronger.) We get the following:

$$\begin{aligned} 1 - \alpha_1 + \alpha_2 &\leq Z \\ 1 - \alpha_2 + \alpha_3 &\leq Z \\ &\ldots \\ 1 - \alpha_{p-1} + \alpha_p &\leq Z \\ 2 - \alpha_0 - \alpha_p + \alpha_{p+1} &\leq Z \\ 1 - \alpha_{p+1} + \alpha_{p+2} &\leq Z \\ &\ldots \\ 1 - \alpha_{t-1} + \alpha_t &\leq Z \\ 1 - \alpha_t + \alpha_0 + \alpha_1 &\leq Z \end{aligned}$$

Adding these t inequalities, all α's cancel, forcing $tZ \geq t+1$. However, it is known that $\mathcal{F}$ is q-Horn only if the set of inequalities has a solution with $Z \leq 1$ [9]. □

Theorem 7.2: Under $\mathcal{M}^k_{m,n}$, with $m/n = r > 2/(k(k-1))$, and $k \geq 3$, the probability that a random formula $\mathcal{F}$ is q-Horn tends to 0 as n goes to ∞.

Proof: Immediate from Lemma 7.1, and Lemma 7.3, which follows. □

Lemma 7.3: Under $\mathcal{M}^k_{m,n}$, with $m/n = r > 2/(k(k-1))$, and $k \geq 3$, the probability that a random formula $\mathcal{F}$ contains a chorded loop tends to 1 as n goes to ∞.

Proof: This is another application of the second moment method closely following that of Lemma 6.3. Let $p = \lceil \ln^2 n \rceil$, and $t = 3p$, assuming $n \geq 2$. This time we seek the expected value of B_t, the number of q-blocked chorded loops containing t clauses in $\mathcal{F}$. We can show that this expectation is large and that Lemma 4.1 also applies, over the indicated range of r.

Taking advantage of the close similarities between q-blocked chorded loops and criss-cross loops, we need modify the proof of Lemma 6.3 only by the small changes. In this case each chorded loop has one Hamiltonian cycle (see Figure 3). Starting a *representative sequence* at the clause containing $\overline{u_1}$, where v_0 is the chord variable, each chorded loop has one representative sequence. Clauses containing $\overline{v_0}$ or v_0 require $(k-3)$ padding literals, rather than $(k-2)$. There are $t+1$ cycle literals, rather than $t-1$. "Activation" does not apply. Using methods similar to the proof of Lemma 6.3, we obtain

$$E(B_t) \quad > \quad \frac{(k-2)^2}{8n}\left(\frac{rk(k-1)}{2}\right)^t \left(1 - O\left(\frac{t^2}{n}\right)\right) \tag{23}$$

The details are omitted.

The application of Lemma 4.1 is also similar to Lemma 6.3. Let S be the set of shared clauses of witness w. Recall the definitions of *fixed*, *limited*, and *free* cycle literals in the new witness z, from the proof of Lemma 6.3. In addition we define an *unshared padding literal* as a padding literal in an unshared clause of z. As before h denotes the number of connected components of $(w - S)$, the connection graph of the unshared clauses of w.

If S contains both clauses that contain the chord variable v_0, then either $h \geq 2$, or $h = 1$ and S contains at least $p+1$ clauses, making $q \geq p+1$. For $h \geq 2$, there are $2h$ limited cycle literals, $(t-q-h)$ free cycle literals, and $(k-2)(t-q)$ unshared padding literals. For $h = 1$, there are q fixed cycle literals because S necessarily has a cycle, 2 limited cycle literals, $(t-q-1)$ free cycle literals, and $(k-2)(t-q)$ unshared padding literals, but necessarily $q \geq p+1$.

If S contains exactly one clause that contains the chord variable v_0, then there are $(q-h)$ fixed, $2h+1$ limited, $(t-q-h)$ free cycle literals, and $(k-2)(t-q)-1$ unshared padding literals.

If S contains no clause that contains the chord variable v_0, then there are $(q-h)$ fixed, $2h$ limited, $(t-q-h+1)$ free cycle literals, and $(k-2)(t-q)-2$ unshared padding literals.

In summary, the combined number of free cycle literals and unshared padding literals is at most $(k-1)(t-q)-h-1$ whenever $q \leq p+1$, and is at most $(k-1)(t-q)-h$ when $q > p+1$. The number of ways to map S into z is $O(t^{2h})$, whereas the number of ways to choose literals falls off as $(2n)^{-h}$, so as in Lemma 6.3, the maximum occurs when h is minimum, in this case $h = 1$. The remainder of the argument parallels (19) through (21) (with t^2 instead of t in the numerator, which makes no difference). □

The above proof showed that a chorded loop of length $t = 3\ln^2(n)$ occurs with high probability, for $r \geq 2/(k(k-1))$. In these cases, that structure forces $Z \geq 1 + 1/t$ in (3) by Lemma 7.1. It follows that all classes specified by (4), which guarantees polynomial-time solvability, occur with probability approaching 0 for this range of r.

8 Matched Formula Analysis

This section analyzes clause-variable matchings in formulas. First we show a matching property of minimally unsatisfiable formulas. Then we study when random formulas are in the matched class, according to Definition 3.6, with high probability under $\mathcal{M}^k_{m,n}$.

Definition 8.1: Given a formula $\mathcal{F}$, let Q be any subset of its clauses. Define the *neighborhood* of Q, denoted as $\mathbf{V(Q)}$, to be the set of variables that occur in Q. Recalling Definition 3.6, the bipartite graph $\mathcal{G}(\mathcal{F})$ contains two vertex sets $\mathcal{C}$ and V, where the elements of $\mathcal{C}$ are the clauses of formula $\mathcal{F}$, and the elements of V are the variables of $\mathcal{F}$. The edges of $\mathcal{G}(\mathcal{F})$ connect $v \in V$ to $C \in \mathcal{C}$ just when variable v appears (positively or negatively) in clause C. Thus $\mathbf{V(Q)}$ is also the set of vertices in $\mathcal{G}(\mathcal{F})$ adjacent to the vertices corresponding to Q.

Define the *deficiency* of Q, denoted $\delta(Q)$, as $\delta(Q) = |Q| - |\mathbf{V(Q)}|$, that is, the excess of clauses over distinct variables in those clauses. A subset $Q \subseteq \mathcal{C}$ is said to be *deficient* if $\delta(Q) > 0$. □

Lemma 8.1: Any formula that has a total matching for clauses (Definition 3.6) is satisfiable.

Proof: If M is such a matching for $\mathcal{G}(\mathcal{F})$, then its edges produce a satisfying partial assignment for $\mathcal{F}$: for each edge (v, C) in M, if $\overline{v}$ occurs in clause C, set variable v to f, otherwise to t. □

The idea of Lemma 8.1 was credited to Adam Rosenberg by Tovey [39]. The following classical theorem, restated in terms of deficiency, gives a necessary and sufficient condition for a total matching for clauses.

Theorem 8.2: (Hall's Theorem [30])
Given a bipartite graph with vertex sets V and $\mathcal{C}$, a matching that includes every vertex of $\mathcal{C}$ exists if and only if no subset of $\mathcal{C}$ is deficient. □

8.1 *Minimal Unsatisfiability and Deficiency*

We now prove a relationship between deficiency and minimal unsatisfiability. As a corollary, any minimally unsatisfiable formula has more clauses than variables. Aharoni and Linial stated the corollary without proof [2], attributing it to an unpublished manuscript of M. Tarsi. The corollary was also obtained independently by Kleine Büning, using entirely different techniques [12]. Recall that a set of clauses is *minimally unsatisfiable* if it is unsatisfiable, and every proper subset is satisfiable.

Theorem 8.3: If S is a minimally unsatisfiable set of clauses, then S has maximum deficiency among all subsets of S, including S itself.

Proof: Let δ^* be the maximum deficiency among all subsets of S, including S itself. Choose some subset $S_1 \subseteq S$ that is maximal (w.r.t. set inclusion) among all subsets whose deficiency is δ^*. It remains to show that S_1 must be S.

Let V be the set of variables occurring in S, i.e., $V = \mathbf{V(S)}$. Let $V_1 = \mathbf{V(S_1)}$.

Let $S_2 = S - S_1$. Each clause in S_2 has at least one literal whose variable does *not* occur in V_1, by the maximum deficiency of S_1. Let S_2' consist of the clauses of S_2 with all literals whose variables occur in V_1 being discarded.

First, we show that S_2' is satisfiable. Let S_3' be any subset of S_2', including S_2' itself. Let $V_3' = \mathbf{V(S_3')}$. Define S_3 to be the set of clauses in S that generated the clauses of S_3' (by removing literals whose variables occurred in V_1). Note that $\mathbf{V(S_3)} \subseteq \mathbf{V_3'} \cup \mathbf{V_1}$ and $|S_3| = |S_3'|$. Also, S_3 and S_1 are disjoint, and V_3' and V_1 are disjoint, so

$$\delta(S_1 \cup S_3) = |S_1| + |S_3'| - |V_1| - |V_3'| = \delta^* + \delta(S_3')$$

But δ^* is the maximum deficiency among *all* subsets of S, so $\delta(S_3') \leq 0$. Since S_3' was arbitrary, by Theorem 8.2, S_2' has a total matching for clauses, and then by Lemma 8.1, S_2' is satisfiable.

It follows that S_1 must be unsatisfiable, or else any model of S_1, combined with any model of S_2', would satisfy S. Since S is minimally unsatisfiable, $S_1 = S$. □

Corollary 8.4: If $\mathcal{F}$ is a minimally unsatisfiable CNF formula, then $\mathcal{F}$ has more clauses than variables.

Proof: Immediate from Theorem 8.3, Theorem 8.2, and Lemma 8.1. □

8.2 *Random Matched Formulas*

The probability that a random clause has all k of its variables in a specified subset of V of cardinality $x-1$ is $\binom{x-1}{k}/\binom{n}{k}$. The probability that a set of x random clauses has variables taken from a specified subset of no more than $x-1$ variables (i.e., the set is deficient) is

$\left(\binom{x-1}{k}/\binom{n}{k}\right)^x$. The number of clause sets of size x is $\binom{m}{x}$ and the number of ways to specify $x-1$ variables is $\binom{n}{x-1}$. Let D_x^* denote the number of subsets of clause vertices of cardinality x that are deficient and D^* the total number of subsets that are deficient. Then

$$E(D_x^*) \leq \binom{m}{x}\binom{n}{x-1}\left(\frac{\binom{x-1}{k}}{\binom{n}{k}}\right)^x \tag{24}$$

$$E(D^*) = \sum_{x=k+1}^{m} E(D_x^*) \tag{25}$$

We wish to obtain an upper bound on the right hand side of (24) that is simple enough to analyze. Noting that binomial coefficients have a recursive representation as products, we would like our bounding expression to have a similar representation. Recall that the empty product is 1, just as the empty sum is 0. Thus motivated, we use the relations, valid for integers $x \geq 1$:

$$\left(\frac{\binom{x-1}{k}}{\binom{n}{k}}\right)^x \leq \left(\frac{x}{n}\right)^{kx} \leq \frac{1}{n^k}\prod_{y=1}^{x-1}\left(\frac{y(y+1)^{k-1}e^k}{n^k}\right) \qquad \text{for } x \geq 1, k \geq 3 \tag{26}$$

These relations can be verified by taking logs. The latter inequality is a bit delicate, but follows from the next inequality, which can be justified by noting that the second derivative of $\ln(y)$ is negative.

$$\sum_{y=1}^{x} \ln y = \sum_{y=2}^{x} \ln y > \int_{\frac{3}{2}}^{x+\frac{1}{2}} \ln(y)\ dy$$

Now we define $D_x(m,n,k)$ and $D(m,n,k)$ to serve as bounding expressions:

$$D_x(m,n,k) = \binom{m}{x}\binom{n}{x-1}\left(\frac{1}{n^k}\right)\prod_{y=1}^{x-1}\left(\frac{y(y+1)^{k-1}e^k}{n^k}\right) \tag{27}$$

$$D(m,n,k) = \sum_{x=k+1}^{m} D_x \tag{28}$$

$$E(D_x^*) \leq D_x(m,n,k) \tag{29}$$

$$E(D^*) \leq D(m,n,k) \tag{30}$$

That is, $E(D_x^*)$ and $E(D^*)$ are actual expectations, while $D_x(m,n,k)$ and $D(m,n,k)$ are upper bounds.

The function g, defined as the ratio D_x/D_{x-1}, has important structural properties for the analysis, which are collected in the following lemma. The parameters of $g(m,n,k,x)$ may be considered as reals.

$$g(m,n,k,x) = \frac{D_x}{D_{x-1}} = e^k\left(\frac{m+1-x}{n}\right)\left(\frac{n+2-x}{n}\right)\left(\frac{x}{n}\right)^{k-2} \tag{31}$$

$$D_1(m,n,k) = \frac{m}{n^k} \tag{32}$$

$$D_x(m,n,k) = g(m,n,k,x)D_{x-1}(m,n,k) \tag{33}$$

The subsequent analysis will use Stirling's formula [1]:

$$y! = \sqrt{2\pi}y^{y+\frac{1}{2}}e^{-y+\theta(y)/12y} \qquad \text{for some } 0 < \theta(y) < 1 \text{ and } y \geq 1. \tag{34}$$

In the following technical lemma the proofs are indicated, where necessary, as each part is stated.

Lemma 8.5: With g and D_x as defined in (31)–(33),

A.

$$g(m,n,k,0) = g(m,n,k,m+1) = 0, \qquad g(m,n,k,x) > 0 \qquad \text{for } 0 < x < m+1 \tag{35}$$

B. For fixed (m,n,k), $\partial g/\partial x$ has $k-3$ roots at 0, and has two positive roots that satisfy the quadratic equation:

$$(k-2)\left((m+1-x)(n+2-x)\right) \quad = \quad x(n+m+3-2x) \tag{36}$$

One root is greater than m and the other (see (37)) is less than m. Combining this fact with (35) implies that g has a unique maximum w.r.t. x in the range $(0,m)$ at

$$x^* \quad = \quad \frac{(k-1)(n+m+3) - \sqrt{k(k-2)(n-m+1)^2 + (n+m+3)^2}}{2k} \tag{37}$$

C. For fixed (m,n,k) and for $k+2 \leq x \leq m$, if the maximum value of g is at most 1, then D_x is monotonically nonincreasing over $k+1 \leq x \leq m$.

D. For fixed (m,n,k) and for $k+2 \leq x \leq m$, if the maximum value of g is greater than 1, then the maximum D_x occurs where g crosses 1 from above; that is, $g(m,n,k,x) \geq 1$ and $g(m,n,k,x+1) < 1$. The x that maximizes D_x is greater than x^* in (37).

E. For $n > e^k$, $g(m,n,k,m) < 1$, so D_m cannot be the maximum of D_x over the range $k+1 \leq x \leq m$.

F. For fixed (n,k,x) and $1 \leq x \leq m < n$, $g(m,n,k,x)$ is an increasing function of m. Also, $D_1(m,n,k)$ is an increasing function of m. Therefore (by monotonicity of multiplication on positive arguments),

$$D_x(m',n,k) \quad < \quad D_x(m,n,k) \qquad \text{for } k+1 \leq x \leq m' < m < n \tag{38}$$

Thus a bound found for m immediately applies for all $m' < m$.

G. Recall that $r = m/n$. Rewrite

$$g(m,n,k,x) \quad = \quad \left(\frac{m+1-x}{x}\right)\left(\frac{n+2-x}{x}\right)\left(\frac{erx}{m}\right)^k \tag{39}$$

Now we have

$$\frac{D_x(m,n,k+1)}{D_x(m,n,k)} \quad = \quad \frac{x!}{e}\left(\frac{e}{n}\right)^x \tag{40}$$

which is independent of k. Taking the log of this ratio, then taking the first two partial derivatives w.r.t. x shows that there is no local maximum. Therefore, in the range $k+1 \leq x \leq m$, the maximum of the ratio $D_x(m,n,k+1)/D_x(m,n,k)$ occurs at either $x = k+1$ or at $x = m$. At $x = k+1$ the ratio is $O\left(n^{-k-1}\right)$. At $x = m = rn$, using Stirling's formula (34), the ratio is $O\left(\sqrt{m}r^m\right)$. Therefore, for $r < 1$ and sufficiently large n and $k+1 \leq x \leq m$, $D_x(m,n,k)$ is a decreasing function of k.

Proof: All observations are established by standard calculus and the theory of roots of polynomials.

We are now ready to prove the main result of this section.

Theorem 8.6: Under $\mathcal{M}^k_{m,n}$, $k \geq 3$, the probability that a random formula $\mathcal{F}$ is matched (per Definition 3.6) tends to 1 if $r \equiv \frac{m}{n} < 0.64$.

Proof: We show, for $r < 0.64$, that the probability that $\mathcal{G}(\mathcal{F})$ has a total matching for clauses tends to 1 as $n \to \infty$ (holding the ratio r fixed). We show that the expected number of deficient subsets (see Definition 8.1) in $\mathcal{G}(\mathcal{F})$ tends to 0 for fixed $r < 0.64$. This is an upper bound on the probability that a deficient subset exists. Therefore, by Theorem 8.2, random formulas will be matched with probability tending to 1 for the same range of r. Specifically, we show that $D(m,n,k) \to 0$ if $r < 0.64$.

First, observe that $D_{k+1} = O(n^{-5})$ for $k \geq 3$, because $D_{k+1} = O(m^{k+1}/n^{k^2})$ and m/n is bounded by a constant. Therefore, if D_x is monotonically nonincreasing on $k+1 \leq x \leq m$, $D(m,n,k) = O(n^{-4})$, and we are done. By Lemma 8.5C, we need to find a range of r such that $g \leq 1$. We now show that $g \leq 1$ for $r \leq e^{-1}$.

To this end, bound g separately in the ranges $x < m(k-1)/k$ and $x \geq m(k-1)/k$.

$$g(m,n,k,x) \;\leq\; e^k \left(\frac{m+1-x}{n}\right)\left(\frac{x}{n}\right)^{k-2} \qquad \text{for } k+1 \leq x < m(k-1)/k \tag{41}$$

$$g(m,n,k,x) \;\leq\; e^k \left(\frac{m+1-x}{n}\right)\left(\frac{n+2-x}{n}\right)\left(\frac{m}{n}\right)^{k-2} \qquad \text{for } m(k-1)/k \leq x \leq m \tag{42}$$

In the first case, the maximum occurs at $x = m(k-2)/(k-1) + O(1)$. Using the relation $\left(\frac{k-2}{k-1}\right)^{k-1} < e^{-1}$, we get $g < \left(\frac{1}{k-2}\right)(re)^{k-1} + O(1/n)$ in the lower range. In the upper range the maximum occurs at $x = m(k-1)/k$, giving $g < \left(\frac{e}{k}\right)\left(1 - r\left(\frac{k-1}{k}\right)\right)(re)^{k-1} + O(1/n)$. In both cases, for $k \geq 3$ and for $r \leq e^{-1}$, we have $g < 1$ over the whole range of x. Therefore, D_x is monotonically nonincreasing under these conditions, as was to be proved.

For the remainder of the proof, we assume $r > e^{-1}$. We define $q = (n/m) = r^{-1}$. Throughout this discussion the ranges of interest are

$$k \geq 3, \qquad 1 < q = r^{-1} < e, \qquad n > 1{,}000{,}000.$$

By Lemma 8.5D, the maximum of D_x (if it is not monotonically nonincreasing) occurs when g crosses 1 from above; this point is some $x > x^*$ of (37). By Lemma 8.5G, D_x is a decreasing function of k for all $k+1 \leq x \leq m$, so any bound for a particular k applies to all $k' > k$.

To obtain a good estimate of the maximum of D_x, it is useful to bound the value of x at which $g = 1$ away from m. To this end, we introduce $h(\beta)$:

$$\beta \;=\; 1 - \frac{x}{m} \tag{43}$$

$$h(\beta) \;=\; g(m,n,k,(1-\beta)m) = \left(\frac{e}{q}\right)^k \left(\frac{1}{m} + \beta\right)\left(q - 1 + \frac{2}{m} + \beta\right)(1-\beta)^{k-2} \tag{44}$$

We need a lower bound on a positive solution of $h(\beta) = 1$. It is easy to verify that, for sufficiently large n,

$$h(\beta) \;<\; 1 \qquad \text{for } 0 \leq \beta \leq \frac{q^{k-1}}{e^k} \quad \text{or} \quad 1 - \left(\frac{q^{k-1}}{e^k}\right) \leq \beta \leq 1 \tag{45}$$

That is, the x that maximizes D_x (when that maximum is greater than 1) is bounded away from both 0 and m by at least $\left(\frac{q^{k-1}}{e^k}\right)m$. In this region, both $1/x$ and $1/(m-x)$ are $O(1/n)$.

In the remaining discussion we use approximations that ignore $O(1/n)$ terms. Let β^* (not directly related to x^*) denote that β which maximizes h. Define

$$\begin{aligned}\phi(\beta) \equiv \ln(h(\beta)) &= k - k\ln(q) + \ln(\beta) + \ln(q-1+\beta) + (k-2)\ln(1-\beta) \qquad (46)\\ \phi(\beta^*) &= 0 \qquad (47)\end{aligned}$$

In addition, β^* must be the smallest positive solution of $\phi(\beta) = 0$. Observe that

$$D_x(m,n,k) = \binom{m}{x}\binom{n}{x-1}(x-1)!\ (x!)^{k-1}\left(\frac{e^{k(x-1)}}{n^{kx}}\right) \qquad (48)$$

By Stirling's formula (34), using $n = qm$ and $x = (1-\beta)m$, we find that all terms of order $m\ln m$ cancel in the expression for $\ln(D_x)$, giving:

$$\begin{aligned}\ln(D_x) &= m\big((k-2)(1-\beta)\ln(1-\beta) - \beta\ln\beta\\ &\quad -(q-1+\beta)\ln(q-1+\beta) + (q-(1-\beta)k)\ln q\big) + O(\log m) \qquad (49)\end{aligned}$$

Using (46)–(47) to eliminate $k\ln q$, the above simplifies further at β^*. Noting that β^* is a function of k and $r = 1/q$:

$$\begin{aligned}\ln(c_k(r)) &\equiv \frac{-q\ln\left(1 - \frac{(1-\beta^*)}{q}\right) - k(1-\beta^*) - \ln\beta^*}{q} \qquad (50)\\ \ln(D_x) &= n\ln(c_k(r)) + O(\log n) \qquad (51)\end{aligned}$$

To conclude the proof, we sketch a numerical procedure to compute a safe upper bound on $c_k(r)$. Given r and k, setting $q = 1/r$, (46)–(47) can be solved for β^* by a monotonically convergent Newton-Raphson iteration, starting at $\beta_0 = q^{k-1}/e^k$, because $d\phi^2(\beta)/d\beta^2 < 0$ and $\phi(\beta_0) < 0$. (For $k = 3$, Cardano's procedure for cubics can also be used.) Using the approximate solution, find close upper and lower bounds, $\beta^- < \beta^* < \beta^+$. Now verify by direct evaluation that $\phi(\beta^-) > 0$ and $\phi(\beta^+) < 0$. Finally, evaluate an upper bound on $\ln(c_k(r))$ by using β^- in terms that are decreasing in β and using β^+ in terms that are increasing in β. If this upper bound is negative, $D_x = O(c_k^n)$, which implies that $D(rn,n,k) \to 0$ as $n \to \infty$.

For $k = 3$ and $r = .64$, the above procedure shows that $\beta^- = .3160743$, $\beta^+ = .3160744$, and $c_3 < .9998$. By Lemmas 8.5D and 8.5G, the same bound applies for smaller r and larger k. □

Figure 4 shows computed safe bounds on r for k up to 10. While it is apparent from this table that r is approaching 1 as k gets large, we do not have a proof.

8.3 Matched Formulas are in LinAut

We now show that any matched formula is in LinAut (see Definition 3.4). This implies that formulas of $\mathcal{M}^k_{m,n}$ are in LinAut with high probability for $r = m/n$ in the ranges shown in

k	r	β^-	β^+	c
3	.64	.3160743	.3160744	.9998
4	.84	.1495843	.1495844	.9916
5	.92	.0788789	.0788790	.9799
6	.96	.0441916	.0441917	.9850
7	.98	.0258741	.0258742	.9922
8	.990	.0155634	.0155635	.9972
9	.995	.0095016	.0095017	.9999
10	.997	.0056522	.0056523	.9685

Figure 4: Ratios r for which a total clause matching exists with probability 1 as $n \to \infty$. See proof of Theorem 8.6 for discussion.

Figure 4. Because of the inductive nature of the definition for LinAut we anticipate that such results will be difficult to show directly. We use the matrix and vector notations that were used in the discussion of q-Horn and LinAut in Section 3.

Theorem 8.7: If $\mathcal{F}$ is matched, then $\mathcal{F}$ is in LinAut.

Proof: The proof is by induction on m, the number of clauses. The base case, $m = 0$ is immediate. For $m > 0$, let $\mathbf{F}$ be the $m \times n$ clause-variable matrix of $\mathcal{F}$. Necessarily $n \geq m$. First we show that $\mathbf{F}$ has a linear autarky. Consider the rows of $\mathbf{F}$ as real n-vectors. The first $m-1$ rows of $\mathbf{F}$ cannot span R^n so there is some nonzero n-vector p that is orthogonal to all of these rows. If the inner product of the m-th row of $\mathbf{F}$ and p is nonnegative, let $x = p$, otherwise, let $x = -p$. Then x is nonzero and $\mathbf{Fx} \geq \mathbf{0}$.

If x has only nonzero components, we are done. Otherwise, define $linSat(\mathcal{F}, x)$ and $\mathcal{F}' = \mathcal{F} - linSat(\mathcal{F}, x)$ as in Definition 3.4. No subset of $\mathcal{F}'$, including itself, is deficient (in the sense of Definition 8.1) because all such subsets are also subsets of $\mathcal{F}$, which is matched. (In particular, $\mathcal{F}'$ has at most as many clauses as variables.) So $\mathcal{F}'$ is also matched and has fewer clauses than $\mathcal{F}$. By the inductive hypothesis $\mathcal{F}'$ is in LinAut. By the definition of LinAut, so is $\mathcal{F}$. $\square$

LinAut also contains the class of formulas that are solvable by application of the pure literal rule alone; Broder *et al.* have shown that 3-CNF formulas generated from a distribution that is slightly different from $\mathcal{M}^3_{m,n}$ are in the latter class with high probability, for ratios $m/n < 1.63$ [10]. However, corresponding results for larger k seem not to be known.

9 Cycles in Connection Graphs

A *cycle* in a formula is an undirected cycle in the propositional connection graph of that formula. Let us say a pair of clauses has a *double clash* if there are (at least) two literals in one clause that are complements of literals in the second clause: e.g., $[a, b, c]$ and $[\overline{a}, \overline{b}, d]$. There is no edge in the connection graph between clauses that have a double clash, because their resolvent is tautologous. In this section we study formulas without cycles, but possibly with clauses that have double clashes.

A formula without unit clauses and without a cycle can be satisfied efficiently, even if it has double clashes. A formula with no cycles and no double clashes is a member of all the well known polynomial-time solvable classes except for Horn (however, it is renamable Horn). The results above for SLUR and q-Horn show that cycles in a random formula are abundant if $r > 2/(k(k-1))$. The next theorem shows that random formulas have few cycles if $r < 1.36/(k(k-1))$.

Theorem 9.1: Under $\mathcal{M}^k_{m,n}$, $k \geq 3$, the average number of cycles in a random formula $\mathcal{F}$ is less than 1 when $r \equiv \frac{m}{n} < \frac{1.36}{k(k-1)}$.

Proof: Index the clauses of $\mathcal{F}$ from 1 to m somehow. A *cycle witness* is a sequence of pairs (i_j, v_j), $0 \leq j < x$, that corresponds to a possible cycle in the propositional connection graph. The correspondence is that the j-th vertex in the possible cycle is the clause whose index is i_j and the j-th and $(j+1)$-th vertices (mod x) are connected by an edge labeled with the variable v_j. We find the expected number of cycle witnesses in $\mathcal{F}$ which is an overestimate of the expected number of cycles in $\mathcal{F}$. The number of cycle witnesses of length x is the number of ways to select x clause indexes in sequence without replacement times the number of ways to select x variables in sequence with replacement, but such that no two adjacent variables (including v_0 and v_{x-1}) are identical. This number is less than $m^{\underline{x}} n(n-1)^{x-1}$. The probability that a sequence of x clauses in $\mathcal{F}$ matches a specified cycle witness of length x is

$$\left(\frac{2^{k-1}\binom{n-2}{k-2}}{2^k \binom{n}{k}} \right)^x = \left(\frac{k(k-1)}{2n(n-1)} \right)^x \tag{52}$$

The factor of 2^{k-1} reflects the fact that literals labeling an edge must occur with opposite signs in the two clauses: only two out of four possible sign combinations result in the edge being present.

Therefore, the expected number of cycle witnesses in $\mathcal{F}$ is less than

$$\sum_{x \geq 3} m^{\underline{x}} n(n-1)^{x-1} \left(\frac{k(k-1)}{2n(n-1)} \right)^x < \sum_{x \geq 3} \left(\frac{mk(k-1)}{2(n-1)} \right)^x \tag{53}$$

$$< \frac{(mk(k-1))^3}{(2(n-1) - mk(k-1))\ (2(n-1))^2}. \tag{54}$$

This is less than 1 if $mk(k-1)/(n-1) < 1.36$. $\square$

This result shows how closely SLUR, q-Horn, and other classes are tied to cycles in formulas: it seems that they are defeated rapidly by the presence of cycles. That is, as r rises, formulas are not SLUR, and are not q-Horn, etc., soon after they begin to contain a significant number of cycles.

We conclude this section by briefly describing how to satisfy a formula efficiently when its connection graph is acyclic, and it has no unit clauses, even though there may be double clashes. The algorithm is recursive. Choose any leaf node of the connection graph, having clause C. At most one edge is incident on C, and say it corresponds to literal u_0. Due to

U-Solve($\mathcal{F}, k$)
Repeat the following
If there are two clauses in $\mathcal{F}$ that resolve to $C \notin \mathcal{F}$ with $|C| \leq k$
Then add C to $\mathcal{F}$.
Until $\emptyset \in \mathcal{F}$ or no more clauses can be added to $\mathcal{F}$.
If $\emptyset \in \mathcal{F}$, Output ("unsatisfiable")
Otherwise Output ("give up")

Figure 5: An algorithm for certifying unsatisfiability

lack of unit clauses, there is at least one more literal in C, say u_1. If this is a pure literal, assign it t, delete clauses with u_1, and continue recursively.

The less obvious case is when u_1 is a not a pure literal. Then all clauses containing $\overline{u_1}$ must have a double clash with C; that is, all clauses containing $\overline{u_1}$ also contain the complement of some other literal in C. Let C_1 be the set of literals in C that cause double clashes involving u_1. Now delete C from the connection graph and recursively satisfy the remaining clauses. Inspect the resulting assignment. If u_1 or any literal of C_1 was assigned t, we are done. But if all literals in C_1 were assigned f, then all clauses containing $\overline{u_1}$ are satisfied, regardless of the assignment to $\overline{u_1}$. So simply change the assignment to $u_1 = \mathsf{t}$, satisfying C as well. This idea is due to Oliver Kullmann, who designates C as a *blocked clause*.

10 Easy Unsatisfiable Families of Formulas

This section describes an algorithm, **U-Solve**, that either demonstrates unsatisfiability in polynomial time, or gives up. We then show that certain families of $\mathcal{M}^k_{m,n}$ can be solved by **U-Solve** with high probability, when m sufficiently rapidly in relation to n. As usual, k remains fixed.

Algorithm **U-Solve** implements a simplified form of *k-closure* [41]. It repeatedly applies the resolution rule to a formula with the restriction that all created resolvents are of size no greater than k. In other words, **U-Solve** finds what are sometimes called "k-bounded" refutations.

The complexity of **U-Solve** is $O(n^k m^2)$ since the number of different resolvents possible (therefore the maximum number of iterations of **U-Solve**) is $O(n^k)$, and a resolving pair can be found in $O(m^2)$ time. If it returns "unsatisfiable" then the input formula is unsatisfiable. However, it may "give up." To study the behavior of **U-Solve** on random k-CNF formulas, we define some terminology.

Definition 10.1: A *minimal k-group* for a binary clause, $[x, y]$, where x and y are literals, is a set of k-clauses, $\mathcal{C}$, that logically implies $[x, y]$, such that:

1. $\mathcal{C}$ logically implies $[x, y]$,
2. no proper subset of $\mathcal{C}$ logically implies $[x, y]$, and

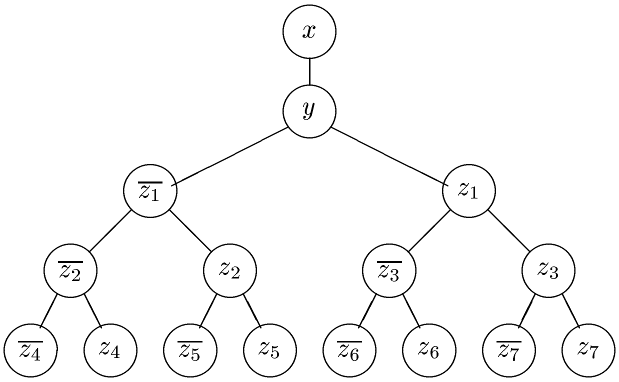

Figure 6: Each path from root to leaf corresponds to a different k-clause of the k-module for $[x, y]$. This diagram illustrates $k = 5$. For $k = 3$ the tree would stop with $\overline{z_1}$ and z_1.

3. $\mathcal{C}$ does not logically imply any clause that is a proper subset of $[x, y]$, namely, $[x]$, $[y]$, or $[]$.

□

Clearly, minimal k-groups can be arbitrarily large, but for them to occur frequently at random, it is desirable that they be as short as possible. However, $|\mathcal{C}| \geq 2^{k-2}$, or else there are not enough clauses to eliminate all models of $[\overline{x}]$ and $[\overline{y}]$. For a pattern with a given number of clauses to occur frequently at random, it is desirable that they have as many distinct variables as possible, because the re-occurrence of the same variable is only about $1/n$ times as likely the occurrence of some new variable. Recall that $\mathbf{V}(\mathcal{C})$ denotes the number of distinct variables in $\mathcal{C}$. By Corollary 8.4, $|\mathcal{C}| + 2 > \mathbf{V}(\mathcal{C})$, since adding $[\overline{x}]$ and $[\overline{y}]$ to $\mathcal{C}$ gives a minimally unsatisfiable set of clauses. That is, the number of distinct variables in $\mathcal{C}$, *other than* $|x|$ and $|y|$, is at most $|\mathcal{C}| - 1$. (Recall that $|u|$ denotes the propositional variable that occurs in literal u.) Therefore, an *optimal* pattern for a minimal k-group for $[x, y]$, in the sense of being most likely to occur at random, should have 2^{k-2} clauses and $2^{k-2} - 1$ *local variables*, distinct from each other and variable-distinct from x and y. We next define a special kind of minimal k-group that is optimal in this sense.

Definition 10.2: Let a formula $\mathcal{F}$ be given and let $\mathcal{C} \subseteq \mathcal{F}$ be a set of clauses. Set $\mathcal{C}$ is called a *k-module* for the binary clause $[x, y]$ if:

1. $\mathcal{C}$ is a minimal k-group for $[x, y]$,
2. $|\mathcal{C}| = 2^{k-2}$,
3. Besides $|x|$ and $|y|$, $\mathcal{C}$ contains exactly $2^{k-2} - 1$ other distinct variables. We shall call these the "local variables" of the k-module, and denote them as z_j, for $1 \leq j \leq 2^{k-2}-1$.
4. The clauses of $\mathcal{C}$ have a 1-1 correspondence to the sequences of k literals encountered on all directed paths in the binary tree of height k, whose nodes are labeled with x, y, and the local variables, as shown in Figure 6.

Actually, the first two properties follow from the last two. □

Now consider again the *criss-cross loop* shown in Figure 2, which is associated with the set of clauses, (9), as well as the literal sequence, v_0, u_1, u_2, ..., u_{t-2}, as in Definition 5.3 and Definition 6.2. Recall that $t = 3p + 2$. If a formula $\mathcal{F}$ contains a k-module for each node of this graph (treating nodes as binary clauses), then it is unsatisfiable, and **U-Solve** finds a refutation. This provides a scheme for showing that **U-Solve** succeeds with high probability under certain conditions.

Definition 10.3: Let formula $\mathcal{F} \in \mathcal{M}^k_{m,n}$ be given and let $t = 3p + 2$, where p is a parameter that is a function of m, n, and k, as yet unspecified (but $2 \leq p \leq n/3$). Let a *cyclic literal sequence* be specified as v_0, u_1, u_2, ..., u_{t-2}. As usual, v_0 is restricted to be a positive literal, while the u_i may be positive or negative. Let $\mathcal{S}$ be the set of binary clauses in (9), corresponding to nodes in the criss-cross loop of Figure 2. For definiteness, denote these binary clauses as S_i, $i = 0, \ldots, t-1$, starting from the lower left and proceeding counter-clockwise around the loop.

An associated *t-k-module* for this cyclic literal sequence is the union of k-modules for all of the S_i, such that all local variables in different k-modules are distinct from each other and are variable-distinct from the cyclic literals. To reduce automorphisms, the binary tree associated with each k-module is required to have as its root the cyclic literal that is earlier in the sequence.

In this context, define:

$$\begin{aligned} L &= 2^{k-2} - 1, \quad \text{which is the number of local variables in a } k\text{-module,} \\ T &= (L+1)t, \quad \text{which is the number of clauses in the } t\text{-}k\text{-module.} \end{aligned}$$

Note that a t-k-module has $T - 1$ distinct variables, of which $t - 1$ are cyclic variables, v_0, $|u_1|$, $|u_2|$, ..., $|u_{t-2}|$. There are $T - t$ local variables among all t or the k-modules. □

A t-k-module is minimally unsatisfiable; that is, removal of any clause produces a satisfiable set. It is also "optimal" in the sense that it has only one fewer variable than the number of clauses. **U-Solve** succeeds on any formula containing a t-k-module by resolving on variables at the bottom level of the tree structure, then the next higher level and so on.

The motivation for considering *minimally* unsatisfiable patterns is that other unsatisfiable clause patterns with a much higher ratio of variables to clauses exist, but the probability that such a padded pattern exists is not greater than the probability that one of its underlying minimally unsatisfiable sets exists. However, restricting attention just to t-k-modules may mean the the strongest results are not obtained for **U-Solve**.

Theorem 10.1: Let m and n approach ∞ in such a way that

$$\frac{m}{n^{k-1}} = \rho \geq \frac{2^k}{k!}$$

while $k \geq 3$ remains fixed. Then Algorithm **U-Solve** succeeds, with probability tending to 1, on a random formula $\mathcal{F} \in \mathcal{M}^k_{m,n}$.

Proof: It suffices to show that $\mathcal{F}$ contains a t-k-module with high probability. The second moment method is used, with many of the same ideas as in Theorem 6.3. However, the details are more complicated, because padding literals occur only once in a formula whereas local variables of a k-module occur multiple times.

Let $\mathcal{R}_{t,k}$ be a t-k-module (see Definition 10.3), where $t = 3p + 2$ and $p = \lceil \ln^2 n \rceil$, assuming $n \geq 2$. It is straightforward to obtain a lower bound on the expected number of t-k-modules of this size. Then we find conditions under which this bound tends to ∞. As mentioned in Definition 10.3, $\mathcal{R}_{t,k}$ contains T clauses and $T-1$ distinct variables, of which $(T-t)$ are local and $(t-1)$ are cyclic. Recall that $L = 2^{k-2} - 1$ is the number of local variables in a k-module:

Fix a sequence for the clauses of $\mathcal{F}$, and a sequence for the clauses of the pattern $\mathcal{R}_{t,k}$. The probability that a particular subsequence of T clauses matches $\mathcal{R}_{t,k}$ for a particular choice of cyclic literals and local variables is

$$\frac{1}{\left(2^k \binom{n}{k}\right)^T}.$$

There are $m^{\underline{T}}$ ways to choose a subsequence.

There are $2^{t-2} n^{\underline{t-1}}$ ways to choose the cyclic literal sequence. The t sequences of L local variables for the k-modules (see Definition 10.3) can be chosen from $(n-t+1)$ variables that have not been chosen for cyclic variables in

$$(n-t+1)^{\underline{T-t}}$$

ways. Due to the symmetries of k-modules, there is no difference between choosing x and $\overline{x}$ in a specific local position.

Therefore, the average number of $\mathcal{R}_{t,k}$ patterns in $\mathcal{F}$ is the product of the above four terms:

$$\begin{aligned} E(B_t) &= \frac{m^{\underline{T}}\ 2^{t-2} n^{\underline{t-1}}\ (n-t+1)^{\underline{T-t}}}{\left(2^k \binom{n}{k}\right)^T} \\ &= \frac{m^{\underline{T}}\ 2^{t-2} n^{\underline{t-1}} (n-t+1)^{\underline{T-t}}\ (k!)^T}{2^{kT}\ \left(n^{\underline{k}}\right)^T} \\ &= \frac{1}{4n} \left(\frac{2^{1/(L+1)}\ k!\ m}{2^k\ n^{k-1}} \right)^T \left(1 - O\left(\frac{t^2}{n}\right)\right) \end{aligned} \tag{55}$$

where (2) was used for the last estimate. Therefore, $E(B_t) \to \infty$ as $n \to \infty$ whenever

$$\frac{m}{n^{k-1}} = \rho \geq \frac{2^k}{k!}$$

The next task is to apply Lemma 4.1. We need to obtain an upper bound on $Pr(z|w)$, the conditional probability that a new witness, z, is in $\mathcal{F}$ given that a reference witness, w, is present and they overlap on at least one clause. The development parallels the proof of Lemma 6.3, and uses its terminology for *fixed*, *limited*, and *free* cycle literals in the new witness z, as well as Definition 6.2. In this context a "witness" is actually a t-k-module for some criss-cross loop, but since an upper bound is sufficient, we will count each of the two *representative sequences* separately. Thus w is an arbitrary fixed t-k-module, which is specified by a cyclic literal sequence of length $(t-1)$ and t sequences of local variables, of length L each. The new witness z ranges over all such sequences.

The set $A(w)$ denotes the set of possible witnesses that share at least one clause with w. This set is partitioned into $A_Q(w)$, $1 \leq Q \leq T$, according to the number of shared clauses of w.

Let q denote the number of k-modules in w that contain at least one shared clause, q, $1 \leq q \leq t$. In any t-k-module clauses in the same k-module must have at least two literals in common, while clauses in different k-modules may have at most one *variable* in common, and that variable occurs as complementary literals. Therefore, the partition of shared clauses according to k-module is preserved in the mapping of shared clauses from w to any new witness z. We shall define a *shared k-module* as one that contains at least one shared clause.

Suppose the shared clauses of one k-module of w have c literals in common. Then these c literals must map into a path beginning at the root in the binary tree of the corresponding k-module in z. Moreover, there are at most 2^{k-c} such clauses. There are at most $c!$ possibilities if both cyclic literals are *limited*, $(c-1)!$ possibilities if one cyclic literal is *limited* and one is *fixed*, and $(c-2)!$ possibilities if both are *fixed*. (Recall definitions from proof of Lemma 6.3.) Notice that the path is determined down to level c for all clauses by the signs of the literals that map into local variables. Then at level $c+1$ all clauses must have the same variable, although with different signs.

As in Lemma 6.3, connected components of shared k-modules in w map into connected components of k-modules in z. Also following that lemma, for any q, the number of ways to map shared clauses distributed among q k-modules is maximized when the number of free literals in z is maximized; however, in this case local variables take the role of padding literals. The number of free literals is the sum of the number of free cyclic literals and the number of free local variables. It is easy to see that the number of free local variables is maximized when the Q shared clauses are packed into as few k-modules as possible, namely $\lceil Q/2^{k-2} \rceil$. Consequently, the number of free local variables is at most $(T-t-Q(1-2^{-(k-2)}))$. The shared clauses within one shared k-module can only be mapped in one way to the new k-module, when the entire k-module is shared; however, it is easy to see that the added flexibility of mappings in partially shared k-modules is more than offset by the reduction in the overall number of free local variables.

Suppose the new witness z shares Q clauses with the reference witness w. Once the nonshared clauses of z are fixed, the probability that the nonshared clauses all occur in $\mathcal{F}$ is bounded by the expectation of the sum of appropriate indicator random variables:

$$\begin{aligned} Pr(z|w) &\leq (m-T)^{\underline{T-Q}} \left(\frac{1}{2^k \binom{n}{k}} \right)^{T-Q} \\ &< \left(\frac{mk!}{2^k n^k} \right)^{T-Q} = \left(\frac{\rho k!}{2^k n} \right)^{T-Q} \end{aligned} \tag{56}$$

Now, regarding each k-module of w as a node in the criss-cross loop, we split into cases according to whether the unshared $k-$modules, as a subgraph, form one connected component (1-CC) or $h \geq 2$ connected components (h-CC); we also split according to whether at least three shared k-modules contain v_0, the criss-cross variable (3+), or at most two do (2$-$).

(1-CC) & 3+) At most $(t-q-1)$ free cyclic literals, and $q > p+2$; $O(1)$ ways to map shared connected component.

(1-CC) & 2−) At most $(t-q-2)$ free cyclic literals; $O(t)$ ways to map shared connected component.

(h-CC) & 3+) The total is maximized when $h = 2$, as before. At most $(t-q-2)$ free cyclic literals; $O(1)$ ways to map shared connected components.

(h-CC) & 2−) At most $(t-q-h-1)$ free cyclic literals. $O(2^h\ t^{2h})$ ways to map shared connected components.

The counting arguments parallel Lemma 6.3. The first two cases dominate, so the number of ways to share Q clauses has these upper bounds, where $L = 2^{k-2}-1$ is the number of local variables in a k-module:

$$\begin{aligned} |A_Q(w)| &\leq O(t)\ (2n)^{(t-q-2)}\ \left(n^L\right)^{t-q} \\ &< \frac{O(t)\ 2^{t-q}\ n^{(L+1)(t-q)}}{n^2} \quad \text{for } \left\lceil \frac{Q}{L+1} \right\rceil = q \leq p+3 \end{aligned} \tag{57}$$

$$\begin{aligned} |A_Q(w)| &\leq O(1)\ (2n)^{(t-q-1)}\ \left(n^L\right)^{t-q} \\ &< \frac{O(1)\ 2^{t-q}\ n^{(L+1)(t-q)}}{n} \quad \text{for } \left\lceil \frac{Q}{L+1} \right\rceil = q > p+3 \end{aligned} \tag{58}$$

Multiply the upper bounds for $|A_Q(w)|$ and $Pr(z|w)$:

$$\begin{aligned} |A_Q(w)|Pr(z|w) &< \frac{O(t)}{n^2} \left(\frac{\rho k!\ 2^{1/(L+1)}}{2^k}\right)^{T-Q} \\ &\leq E(B_t) \left(\frac{O(t)}{n}\right) \left(\frac{\rho k!\ 2^{1/(L+1)}}{2^k}\right)^{-Q} \quad \text{for } \left\lceil \frac{Q}{L+1} \right\rceil = q \leq p+3 \end{aligned} \tag{59}$$

$$\begin{aligned} |A_Q(w)|Pr(z|w) &< \frac{O(1)}{n} \left(\frac{\rho k!\ 2^{1/(L+1)}}{2^k}\right)^{T-Q} \\ &\leq E(B_t)\ O(1) \left(\frac{\rho k!\ 2^{1/(L+1)}}{2^k}\right)^{-Q} \quad \text{for } \left\lceil \frac{Q}{L+1} \right\rceil = q > p+3 \end{aligned} \tag{60}$$

It follows that

$$\sum_{Q=1}^{T} |A_Q(w)|Pr(z|w) \ < \ E(B_t) \left(O\left(\frac{1}{n}\right) + O\left(\left(\frac{\rho k!\ 2^{1/(L+1)}}{2^k}\right)^{-\ln^2 n}\right)\right) \tag{61}$$

and Lemma 4.1 applies, concluding the proof. □

As already stated, $\mathcal{R}_{t,k}$ subsets have T clauses and $T-1$ distinct variables. This relationship between clauses and distinct variables cannot be improved upon, among minimally unsatisfiable clause sets, in view of Corollary 8.4. However, $\mathcal{R}_{t,k}$ has 2^{T-t} automorphisms (rename any subset of local variables to their complements). A construction without automorphisms would improve the constant ρ in Theorem 10.1 by a factor of two, but would not affect the exponent of n in that theorem. Fu has given a construction with fewer automorphisms upon which **U-solve** succeeds with high probability, but his proof requires results from random hypergraph theory [26].

A different polynomial-time algorithm for detecting unsatisfiability was presented by Beame *et al.* [6], and they showed that it succeeds with high probability for $m = \Omega\left(n^{k-1-((k-2)\ln\ln n/\ln n)}\right)$.

Their proof also requires results from random graph theory. Reducing this exponent to a constant below $(k-1)$ seems to require a new idea, and is an interesting open problem.

11 Discussion and Conclusions

The aim of this paper is to determine the scope of some well known polynomial-time solvable classes of Satisfiability. Since the classes we studied are incomparable, we used a probabilistic approach to determine scope. The popular distribution $\mathcal{M}^k_{m,n}$ along with the parameter $r = m/n$ was chosen to provide a scale of formula "hardness." We determined where, on that scale, random formulas are members of the classes with high probability.

We found that random formulas are not SLUR and not q-Horn about where formula cycles begin to appear. Thus, neither class dominates in any range of values of r except where formulas are extremely "easy." The results reveal the vulnerability of SLUR, q-Horn and other polynomial-time solvable classes to the presence of particular cyclic substructures. Both SLUR and q-Horn are about equally handicapped by this: that is, they are defeated by cycles that are somewhat different, but become abundant at the same ratio, $r = m/n = 2/(k(k-1))$. This was surprising since SLUR is nearly useless on unsatisfiable formulas, while q-Horn can solve non-trivial unsatisfiable formulas. Because of this, we had expected that q-Horn might dominate in some range of values of r where formulas are unsatisfiable with high probability. But, this turned out not to be the case.

We introduced another class of formulas consisting of matched formulas. The matched class is incomparable with SLUR and q-Horn but we showed that its prevalence, with respect to r, subsumes the others. This class is interesting because it is not necessarily defeated by the presence of cycles. However, it is handicapped by the fact that no matching is possible if $m > n$.

We also showed that the new polynomial-time solvable class LinAut contains the matched class, so its formulas are also more prevalent than SLUR and q-Horn in the same sense. We leave a more complete investigation of LinAut by means of the techniques given here to future work.

Acknowledgements

We thank Alasdair Urquhart for pointing out an error in the proof of an earlier version of Theorem 8.3. We thank the anonymous referees for noticing several other errors, for numerous suggestions to improve the presentation, and for calling our attention to the work on LinAut. The work of Franco was supported in part by the Office of Naval Research grant N00014-94-1-0382. The work of Van Gelder was supported in part by NSF grants CCR-9503830 and CCR-9505036.

References

[1] M. Abramowitz, and I. A. Stegun. Handbook of Mathematical Functions. Dover Publications, New York, 1972.

[2] R. Aharoni and N. Linial. Minimal Non-Two-Colorable Hypergraphs and Minimal Unsatisfiable Formulas. *Journal of Combinatorial Theory, Series A*, 43(2):196–204, 1986.

[3] N. Alon and J. H. Spencer. The Probabilistic Method. Wiley, New York, 1992.

[4] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–132, 1979.

[5] B. Aspvall. Recognizing disguised NR(1) instances of the satisfiability problem. *Journal of Algorithms* 1:97–103, 1980.

[6] P. Beame, R. Karp, T. Pitassi and M. Saks. On the Complexity of Unsatisfiability Proofs for Random Formulas. In *Proceedings of 30th Annual ACM Symposium on Theory of Computing*, 1998.

[7] B. Bollobas. Random Graphs. Academic Press, London, 1985.

[8] E. Boros, P. L. Hammer, and X. Sun. Recognition of q-Horn formulae in linear time. *Discrete Applied Mathematics* 55:1–13, 1994.

[9] E. Boros, Y. Crama, P. L. Hammer, and M. Saks. A complexity index for satisfiability problems. *SIAM Journal on Computing* 23:45–49, 1994.

[10] A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 322–330, 1993.

[11] H. Kleine Büning. On generalized Horn formulas and k resolution. *Theoretical Computer Science* 116:405–413, 1993.

[12] H. Kleine Büning. On the minimal unsatisfiability problem for some subclasses of CNF. In *Abstracts of 16th Int'l Symposium on Mathematical Programming.* Lausanne, 1997.

[13] V. Chandru and J. N. Hooker. Extended Horn sets in propositional logic. *J. ACM* 38:205–221, 1991.

[14] M. T. Chao and J. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiable problem. *Information Sciences*, 51:289–314, 1990.

[15] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *J. of ACM*, 35:759–770, 1988.

[16] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *Proc. 32nd Symposium on the Foundations of Computer Science*, 1992.

[17] M. Conforti, G. Cornuéjols, A. Kapoor, K. Vušković, and M. R. Rao. Balanced Matrices. Mathematical Programming: State of the Art. J. R. Birge and K. G. Murty, eds. Braun-Brumfield, United States. Produced in association with the 15th Int'l Symposium on Mathematical Programming, University of Michigan, 1994.

[18] M. Dalal, and D. W. Etherington. A hierarchy of tractable satisfiability problems. *Information Processing letters* 44:173–180, 1992.

[19] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1:267–284, 1984.

[20] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multi-commodity flow problems. *SIAM J. on Computing*, 5(4):691–703, 1976.

[21] J. Franco. Relative size of certain polynomial time solvable subclasses of satisfiability. In *Satisfiability Problem: Theory and Applications. DIMACS Workshop, March 1996.*, D. Du, J. Gu, and P. M. Pardalos, eds. DIMACS Series on Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, pages 211–223, 1997.

[22] J. Franco, J. Goldsmith, J. S. Schlipf, E. Speckenmeyer, R. P. Swaminathan. An algorithm for the class of pure implicational formulas. *Proc. of the Workshop on Satisfiability*, Siena, Italy, 1996.

[23] J. Franco and M. Paull. Probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[24] J. Franco and A. Van Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. In *Abstracts of 16th Int'l Symposium on Mathematical Programming.* Lausanne, 1997.

[25] A. M. Frieze, and S. Suen. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, 20:312–355, 1996.

[26] X. Fu. On the Complexity of Proof Systems. PhD thesis, U. of Toronto, 1995.

[27] G. Gallo, and M. G. Scutella. Polynomially solvable satisfiability problems. *Information Processing Letters* 29:221–227, 1988.

[28] G. Gallo and D. Pretolani. Hierarchies of polynomially solvable SAT problems. Presented in the *3rd International Symposium on AI & Mathematics*, January, 1994.

[29] A. Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences* 53(3):469–486, Dec. 1996. (Preliminary version in *Proc. MFCS*, Prague, 1992, LNCS 692.)

[30] P. Hall. On representatives of subsets. *Journal of London Mathematical Society* 10:26–30, 1935.

[31] A. Itai and J. Makowsky. On the complexity of Herbrand's theorem. Working paper 243, Department of Computer Science, Israel Institute of Technology, 1982.

[32] R. A. Kowalski. A proof procedure using connection graphs. *Journal of the ACM* 22:572–595, 1974.

[33] O. Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, to appear.

[34] H. R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the Association for Computing Machinery* 25:134–135, 1978.

[35] H. van Maaren. A short note on some tractable cases of the Satisfiability problem. *Information and Control* 158:125–130, 2000.

[36] T. S. Schaefer. The complexity of satisfiability problems. In *Proceedings of Tenth Annual ACM Symposium on Theory of Computing* pages 216–226, 1978.

[37] J. S. Schlipf, F. Annexstein, J. Franco, and R. P. Swaminathan. On finding solutions for extended Horn formulas. *Information Processing Letters* 54:133–137, 1995.

[38] M. G. Scutella. A note on Dowling and Gallier's top-down algorithm for propositional Horn satisfiability. *Journal of Logic Programming* 8:265–273, 1990.

[39] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

[40] K. Trümper. Effective Logic Computation. Wiley, New York, 1998.

[41] A. Van Gelder and Y. K. Tsuji. Satisfiability Testing with More Reasoning and Less Guessing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, D. S. Johnson and M. Trick, eds. American Mathematical Society, 1996.