k-SAT Is No Harder Than Decision-Unique-k-SAT

Chris Calabro and Ramamohan Paturi*

University of California, San Diego
La Jolla, CA, USA
{ccalabro,paturi}@cs.ucsd.edu

Abstract. We resolve an open question by [3]: the exponential complexity of deciding whether a k-CNF has a solution is the same as that of deciding whether it has exactly one solution, both when it is promised and when it is not promised that the input formula has a solution. We also show that this has the same exponential complexity as deciding whether a given variable is backbone (i.e. forced to a particular value), given the promise that there is a solution. We show similar results for True Quantified Boolean Formulas in k-CNF, k-Hitting Set (and therefore Vertex Cover), k-Hypergraph Independent Set (and therefore Independent Set), Max-k-SAT, Min-k-SAT, and 0-1 Integer Programming with inequalities and k-wide constraints.

Keywords: k-SAT, unique satisfiability, exponential complexity, quantified Boolean formulas, hitting set, independent set.

1 Introduction

For a problem $L = \{x \mid \exists y\ R(x,y)\}$ where R is some relation, define the *solutions* of x as $\mathrm{sol}(x) = \{y \mid R(x,y)\}$. Define Decision-Unique-L (or DU-L) to be the problem of deciding whether the input x has $|\mathrm{sol}(x)| = 1$; Unique-L (or U-L) to be DU-L but with the promise that $|\mathrm{sol}(x)| \leq 1$; and Satisfiable-Unique-L (or SU-L) to be DU-L but with the promise that $|\mathrm{sol}(x)| \geq 1$.

Note that these definitions depend on R being understood from context, since alternative formulations of a problem could lead to alternative definitions of DU-L, U-L, SU-L, but this will rarely cause ambiguity. E.g. DU-k-SAT will be the problem of deciding whether a given k-CNF has a unique satisfying assignment and DU-IS will the the problem of deciding whether a given graph has a unique independent set of size at most a given integer.

For a problem P parameterized by parameter n and solvable in time poly $(|x|)2^{O(n)}$ on input x by a randomized algorithm with error $\leq \frac{1}{3}$, define the *exponential complexity* c_P of P to be the infimum of those c such that P is solvable in time poly$(|x|)2^{cn}$ by a randomized algorithm with error $\leq \frac{1}{3}$. For each problem considered here except for the weighted satisfiability variants, $|x| \leq \mathrm{poly}(n)$,

* Research supported by NSF Award CCF-0515332.

A. Frid et al. (Eds.): CSR 2009, LNCS 5675, pp. 59–70, 2009.

so that the poly($|x|$) can usually be dropped from the above definition. Also, for problems involving formulas (graphs), we will implicitly take the parameter to be the number of variables (nodes). We can similarly define the *deterministic* exponential complexity dc_P of a problem P by eliminating the word 'randomized' in the above definition. Clearly $c_P \leq dc_P$. The reverse inequality is not known, and even a strong hypothesis like P = BPP does not obviously imply it since, for all we know, derandomization may square running time, assuming it can be done at all.

A Boolean formula is in *conjunctive normal form* (CNF) iff it is a conjunction of disjunctions of literals - i.e. an AND of ORs of variables or their negations. A CNF is a k-CNF iff each disjunction contains $\leq k$ literals. SAT is the problem of deciding whether a given CNF has an assignment to the variables that makes the formula true. k-SAT further restricts the input to k-CNF.

While $c_{k\text{-SAT}} \leq c_{\text{DU-}k+1\text{-SAT}}$ is obvious (add a new variable z to each clause and n new clauses $(z \rightarrow x_i)$ for each i), the inequality is probably not tight since, assuming the exponential time hypothesis (that $c_{3\text{-SAT}} > 0$), $c_{k\text{-SAT}}$ increases infinitely often as a function of k [6], and it would be surprising if it did not increase with each k. [3] conjectured that the deterministic exponential complexity of k-SAT and DU-k-SAT are the same, i.e. that

$$dc_{k\text{-SAT}} = dc_{\text{DU-}k\text{-SAT}} \ . \tag{1}$$

The current authors, in [1], incorrectly cited [3], claiming that they had shown (1) rather than merely conjectured it. Here we remedy this - we prove (1) by giving a deterministic polytime Turing reduction from k-SAT to SU-k-SAT such that on an input with n variables, the oracle is called only on formulas with $\leq n+O(1)$ variables. Since $dc_{\text{SU-}k\text{-SAT}} \leq dc_{\text{DU-}k\text{-SAT}}$, and self-reducibility implies $dc_{\text{DU-}k\text{-SAT}} \leq dc_{k\text{-SAT}}$, we have

$$dc_{k\text{-SAT}} = dc_{\text{SU-}k\text{-SAT}} = dc_{\text{DU-}k\text{-SAT}} \ .$$

Via standard error reduction techniques (lemma 1), such a reduction also shows

$$c_{k\text{-SAT}} = c_{\text{SU-}k\text{-SAT}} = c_{\text{DU-}k\text{-SAT}} \ . \tag{2}$$

It is not known whether $c_{k\text{-SAT}} = dc_{k\text{-SAT}}$, but it seems like the answer is 'no'. E.g. [7] gives a nice table contrasting the history of the development of deterministic vs. randomized k-SAT algorithms, with the more recent randomized algorithms with significantly less exponential complexity than their deterministic counterparts.

We also use the same technique to show similar results for True Quantified Boolean Formulas in k-CNF, k-Hitting Set, Vertex Cover, k-Hypergraph Independent Set, Independent Set, Max-k-SAT, Min-k-SAT, and 0-1 Integer Programming with inequalities and k-wide constraints.

Our main contribution is the following technique for efficiently reducing a problem P to SU-P: view an instance ϕ of P as a set of constraints $C_1, \ldots, C_m$ over a set of variables $x_1, \ldots, x_n$. Starting with the empty set of constraints, add

one constraint of ϕ at a time to a list and maintain the invariant that we know a solution to the list. To add a new constraint C_i to the list, construct some gadget that conditionally turns on C_i in the case that the variables are assigned differently than the solution already known for the $i-1$ case. Then, using this gadget, call the oracle repeatedly to learn a solution for the i case. Knowing a solution to the $i-1$ case allows us to encode such gadgets significantly more efficiently than we would otherwise know how to do. Although this idea is the basis for each reduction we consider, there are significant problem-specific details that prevent us from factoring out large common parts of the proofs.

The current work does not completely resolve the question of the exponential complexity of satisfiability vs. that of unique satisfiability since it still does not show $c_{k\text{-SAT}} = c_{\text{U-}k\text{-SAT}}$, though we strongly suspect it, as equality holds in the limit as $k \to \infty$ [1].

The reduction of Valiant and Vazirani [10] falls a bit shorter in proving this statement about k-SAT, but is quite general: (the intersection of a nonempty set system of size about 2^m on n variables with the solutions to a system of approximately m random linear equations over GF_2) has size exactly 1 with probability $\Omega(1)$. This idea shows the NP-hardness of U-k-SAT under randomized reductions but not that it has the same exponential complexity as k-SAT because expressing an $m \times n$ random linear system in k-CNF seems to require a quadratic increase in the number of variables. However this idea can be used to show that $c_P = c_{\text{U-}P}$ for any problem P where intersecting such a linear system could be expressed with only a $1 + o(1)$ factor increase in the complexity parameter, such as in CircuitSAT.

Paper Organization. §2 gives the previous work on the problem. In §3, we define *efficient* reductions, which will allow a cleaner presentation. §4–7 prove the main theorems, organized according to problem type: §4 covers constraint satisfaction problems, §5 covers quantified Boolean formulas, §6 covers optimization problems where the objective function is the solution size, and §7 covers optimization problems where the objective function is the number of satisfied constraints. §8 concludes.

2 Previous Work

[3] show that $dc_{k(r)\text{-SAT}} \leq dc_{\text{DU-}k\text{-SAT}}$ where $k(r)$-SAT is k-SAT but where the input is restricted to have $\leq r$ false clauses at one or more of the assignments $1^n, 0^n$. Since there is nothing special about these 2 assignments, we might as well think of this variant as requiring that the input include some assignment at which the formula has $\leq r$ false clauses - i.e. that not only does the formula have a small satisfiability gap, but that the input include a witness to such a small gap.

But it's not immediately clear whether $k(r)$-SAT is a very important restriction of SAT. On the one hand, the standard Cook reduction from an arbitrary problem in NP to SAT via computation tableaux generates a k-CNF with a gap of 1, and one can even generate a witness for this gap in polytime. This is because

the formula generated essentially encodes, "There is a y such that after computing the predicate $R(x, y)$, the result is true," and only a single clause actually encodes the part that says "the result is true". So $k(1)$-SAT is NP-complete, but the number of variables needed in a reduction to $k(r)$-SAT seems to be large, even when reducing from k-SAT, and so it doesn't seem to be very useful for upperbounding the exponential complexity of k-SAT.

On the other hand, the expected fraction of clauses satisfied in a k-CNF under a random assignment is $1 - 2^{-k}$, and so an assignment satisfying at least that many clauses can be found in polytime with high probability. But Håstad showed, using a proof based on PCPs [4], that with no extra restriction on the input formula, no larger a fraction can be guaranteed, unless P = NP. This leaves unclear just how much smaller is the exponential complexity of $k(r)$-SAT than that of k-SAT. We will show that they are equal.

3 Efficient Reductions

Let us say that a parameterized problem A *efficiently reduces* to parameterized problem B, and write $A \preceq B$, iff $\exists$ a polytime Turing reduction f from A to B so that for each instance x of A of parameter n and oracle call y that $f^B(x)$ makes, the parameter of y is $\leq n+O(1)$. Obviously, $\preceq$ is reflexive and transitive. We will also write $A \simeq B$ iff $A \preceq B$ and $B \preceq A$.

Lemma 1. *If $A \preceq B$ then $dc_A \leq dc_B$ and $c_A \leq c_B$.*

Proof. The first inequality is obvious. For the second, let M_B be a randomized poly$(|x|)2^{cn}$-time Turing machine solving B with error $\leq \frac{1}{16} = p$ (this can be constructed by one with error $\leq \frac{1}{3}$ by taking the majority answer from 21 independent calls). Suppose $f^B(x)$ runs in time $\leq t$. Define M_A as f, but replacing each call to the oracle by $r = 2 \lg t$ calls to M_B and take the majority answer. From the union bound, the probability that M_A errs is $\leq t$ times the probability that a binomial random variable with parameters (r, p) is $\geq \frac{1}{2}r$, and this is $\leq \sum_{i=\lceil \frac{r}{2} \rceil}^{r} \binom{r}{i} p^i (1-p)^{r-i} \leq p^{\frac{r}{2}} 2^r = 2^{-r}$. So M_A solves A, takes time $\leq 2t \lg t 2^{c(n+O(1))}$, and errs with probability $\leq t2^{-r} = \frac{1}{t}$. □

Note that lemma 1 would hold even if we significantly loosened the notion of an efficient reduction by allowing subexponential time, randomness with one-sided error, and oracle calls to problems with parameter as much as $n(1+o(1))$, but we will actually be demonstrating this stronger notion here. Also it should be noted that this form of reduction is more strict than the similar SERF reductions of [5], which allow a linear increase in the complexity parameter since they want the loosest possible notion of reduction under which SUBEXP is closed.

4 Constraint Satisfaction Problems

4.1 *k*-SAT

In this section, we show the following.

Theorem 2

$$k\text{-SAT} \simeq \text{SU-}k\text{-SAT} \simeq \text{DU-}k\text{-SAT} .$$

(1) and (2) then follow from lemma 1. Note that all problems discussed in this paper with a parameter k are solvable in polytime for $k < 2$, so we will assume throughout that $k \geq 2$. For each problem P, SU-$P \preceq$ DU-P via the identity map, so we won't bother to state this in the proofs. Also, for each problem P that we consider in this paper, DU-$P \preceq P$ by using self-reducibility to find a solution and then making n more queries to decide whether there is another, so we won't bother to formally state this in the proofs to follow either.

For example, if $P = k$-SAT and the input formula is ϕ, we can set a variable x_i, then ask the oracle whether the formula is still satisfiable to discover a correct setting of x_i, then set x_i correctly and continue similarly with the remaining variables to get a complete solution a. Then for each variable x_i, ask the oracle whether $F|x_i \neq a_i$ is satisfiable.

Proof. (k-SAT $\preceq$ SU-k-SAT) Let A be an oracle for SU-k-SAT. Also, let any predicate on $\leq k$ Boolean variables represent the k-clauses logically equivalent to them; e.g. $(x \to y = 0)$ will stand for the clause $\{\bar{x}, \bar{y}\}$.

Let $\phi = \{C_1, \ldots, C_m\}$ with variables $x_1, \ldots, x_n$ be our input formula and let z be a new variable. For i going from 1 to m, we will find a solution a to the first i clauses $\phi_i = \{C_1, \ldots, C_i\}$, if there is one. Finding an a that satisfies 0 clauses is trivial. Suppose that we have an a that satisfies ϕ_{i-1}. For each literal l in C_i, we ask A whether $\phi_{i-1} \cup \{(z \to x_j = a_j) \mid j \in [n]\} \cup \{(\bar{z} \to l)\}$ (which is satisfiable: set $x = a, z = 1$) is uniquely satisfiable. It answers yes iff there is no solution to ϕ_i with $l = 1$. If each of the $|C_i| \leq k$ queries gives an answer of yes, then ϕ_i, and hence ϕ, is not satisfiable. If the jth query answers no, then we can safely set the first $j-1$ literals of C_i to 0 and the jth literal to 1 in a partial assignment b. At this point, we know that b can be extended to a solution to ϕ_i, and we want to use similar calls to the oracle to find assignments to the remaining variables to get a new assignment that satisfies ϕ_i.

More specifically, suppose we have a partial assignment b' that we know can be extended to a solution to ϕ_i. Let b be a partial assignment that extends b' by setting a new variable x_r to an arbitrary value b_r. Then we use A and the following lemma to discover whether b can also be extended to a solution of ϕ_i, and if not, we simply flip b_r. We continue in this way until we have a full solution to ϕ_i.

Lemma 3. *Let $a \in \text{sol}(\phi_{i-1})$, b be a partial assignment, and $\psi = \phi_{i-1} \cup \{(z \to x_r = a_r) \mid r \in [n]\} \cup \{(\bar{z} \to x_r = b_r) \mid r \in$ domain of $b\}$. Then ψ has a solution, and it is unique iff b cannot be extended to a solution to ϕ_{i-1}.*

Proof. a together with assigning z to 1 is a solution to ψ. There is no other solution to ψ with $z = 1$, and $\psi\,|_{z=0}$ forces the partial assignment b. □

The reduction uses poly(n) time and makes $\leq mn$ oracle calls, each with $\leq n+1$ variables, so it is efficient. □

4.2 Integer Programming

Let k-BIP be the problem of deciding whether there is a solution to a given system of m linear inequalities in n Boolean variables, where each inequality involves $\leq k$ variables.

Corollary 4

$$k\text{-BIP} \simeq \text{SU-}k\text{-BIP} \simeq \text{DU-}k\text{-BIP} \ .$$

Proof Since the construction is almost exactly the same as for k-SAT in theorem 2, we only give an outline. To express $\alpha \rightarrow \beta$ in the construction, use the inequality $\alpha \leq \beta$. To express a negated variable $\bar{x}$, use $1 - x$. When adding a new constraint C_i, ask the oracle $\leq 2^k$ questions, one for each setting c of the $\leq k$ variables in C_i that satisfies C_i: is it possible to extend $b \cup c$ to a solution of the first $i - 1$ constraints? This gives a polytime reduction that makes $\leq m(2^k - k + n)$ queries, each with $\leq n + 1$ variables. □

4.3 Backbone Variables

Backbone variables are a tool from statistical physics for understanding the nature of random k-SAT instances - see e.g. [8]. Dubois and Dequen [2] use such variables in a heuristic to refute large unsatisfiable hard random 3-SAT instances.

Given a nonempty set system $S \subseteq \mathcal{P}(U)$, $i \in U$ is a *backbone variable* iff $\forall a \in S \ i \in a \vee \forall a \in S \ i \notin a$. x_i is a backbone variable of formula ϕ iff x_i is a backbone variable of sol(ϕ). Define the k *backbone variable* promise problem (k-BB) to be to decide whether a given variable is backbone in a given k-CNF ϕ with the promise that ϕ is satisfiable.

Corollary 5

$$k\text{-SAT} \simeq k\text{-BB} \ .$$

Proof Immediate from theorem 2 since a satisfiable k-CNF has exactly 1 solution iff each of its variables is backbone. □

5 Extending the Result Up the PH

Define (d, k)-TQBF to be those true quantified Boolean formulas of the form $Q_1 \overrightarrow{w}_1 \ \cdots \ Q_d \overrightarrow{w}_d \ \phi$ where each $\overrightarrow{w}_i$ is a (possibly empty) tuple of Boolean variables, each quantifier $Q_i \in \{\exists, \forall\}$ is $\exists$ iff i is odd, each variable of $\phi \in k$-CNF is quantified, and the whole formula is true in the standard sense. The solutions are those assignments to $\overrightarrow{w}_1$ that make $Q_2 \overrightarrow{w}_2 \cdots Q_d \overrightarrow{w}_d \ \phi$ true. Then DU-(d, k)-TQBF is essentially those (d, k)-TQBFs where the first quantifier is $\exists!$ ("there is a unique") instead of $\exists$. If we parameterize on the total number of variables n, then we have the following.

Theorem 6

$$(d,k)\text{-TQBF} \simeq \text{SU-}(d,k)\text{-TQBF} \simeq \text{DU-}(d,k)\text{-TQBF} \ .$$

Proof. ((d,k)-TQBF $\preceq$ SU-(d,k)-TQBF) It was shown in [9] that TQBF restricted to quantified 2-CNF formulas is in linear time, so we may assume wlog that $k \geq 3$. Let A be an oracle for SU-(d,k)-TQBF and $F = Q_1 \overrightarrow{w}_1 \cdots Q_d \overrightarrow{w}_d \ \phi$ with variables $x_1, \ldots, x_n$ and clauses $\phi = \{C_1, \ldots, C_m\}$ be our input formula. Let $x_1, \ldots, x_q$ be the variables in $\overrightarrow{w}_1$, and y, z be variables not in F. If some clause has only $\forall$ quantified literals, then player $\forall$ can easily win and we reject. Otherwise, for i going from 1 to m, we will find a solution a, if there is one, to $F_i = Q_1 \overrightarrow{w}_1 \cdots Q_d \overrightarrow{w}_d \phi_i$ where $\phi_i = \{C_1, \ldots, C_i\}$. Finding an a that satisfies F_0 is trivial. Suppose we have an a that satisfies F_{i-1}. We use A and the following lemma (with $b = \emptyset$) to decide whether F_i is satisfiable.

Lemma 7. *Let $a \in \text{sol}(F_{i-1})$, b be a partial assignment to $\overrightarrow{w}_1$. Suppose $l_k \in C_i$ is $\exists$ quantified under Q_j. Let $\psi = \phi_{i-1} \cup \{(z \rightarrow x_r = a_r) \mid r \in [q]\} \cup \{(\bar{z} \rightarrow x_r = b_r) \mid r \in$ domain of $b\} \cup \{(z \rightarrow y), \{l_1, \ldots, l_{k-1}, y\}, \{\bar{y}, l_k, z\}\}$, and $G = Q_1 z, \overrightarrow{w}_1 \cdots Q_j y, \overrightarrow{w}_j \cdots Q_d \overrightarrow{w}_d \ \psi$. Then G has a solution, and it is unique iff b cannot be extended to a solution to F_i.*

Proof. a together with assigning z (and y, if $j = 1$) to 1 is a solution to G. There is no other solution to G with $z = 1$, and $G\,|_{z=0}$ forces the partial assignment b. A winning strategy for player $\exists$ for $G\,|_{z=0}$ is also a winning strategy for F_i: just ignore y. Conversely, a winning strategy for F_i is a winning strategy for $G\,|_{z=0}$ if, in addition, we set $y = l_k$, which is possible since they are both quantified under Q_j. □

If F_i is satisfiable, we find a solution b to F_i by starting with the empty partial assignment. Suppose we have a partial assignment b' that we know can be extended to a solution to F_i. Let b be a partial assignment that extends b' by setting a new variable x_r in $\overrightarrow{w}_1$ to an arbitrary value b_r. We use A and lemma 7 to discover whether b can be extended to a solution to F_i, and if not, we flip b_r. We continue in this way until we have a full solution to F_i. The reduction uses poly(n) time and makes $\leq m(q+1)$ oracle calls, each with $\leq n+2$ variables, so it is efficient. □

The same result also holds for (∞, k)-TQBF, i.e. without restricting the number of levels of alternation, but it is less exciting since it is obvious: just add 2 empty quantifiers at the beginning.

One could also ask whether the exponential complexity changes when !s are placed on some subset of existential quantifiers other than just the first one. But the above proof technique does not seem to generalize. One problem that arises when trying to prove that the exponential complexity of the unique case (where the ! is on a $\exists$ other than the first) is no more than that of the non-unique case is that, while before it was easy to store a solution (the a variable) in a polynomial amount of space, given a formula such as $\exists x \ \forall y \ \exists z \ \phi$, it now seems like we have

to store a whole function that maps each y to an appropriate z. It is not obvious how to store such a function in a subexponential amount of space. But another problem arises when trying to prove even the reverse inequality: it was easy to show that a problem of the form $\exists! x\ \phi$ Turing reduces to problems of the form $\exists x\ \phi$ with only $o(n)$ more variables and in a subexponential amount of time, but how can one even Turing reduce a problem of the form $\forall x\ \exists! y\ \phi$ to problems of the form $\forall x\ \exists y\ \phi$ with only $o(n)$ more variables and in a subexponential amount of time, let alone other, more complex formulas with more quantifiers?

6 Solution Optimization Problems

The following problems involve optimizing the size of the solution.

6.1 *k*-Hitting Set

Define the *k-Hitting Set* problem (k-HS) as given a k-set system and an integer l, decide whether there is a hitting set of size $\leq l$; i.e. given (U, S, l) such that $S \subseteq \mathcal{P}(U)$ and $\forall s \in S\ |s| \leq k$, if we define the hitting sets as $\{h \subseteq U \mid \forall s \in S\ h \cap s \neq \emptyset\}$, decide whether there is a hitting set of size $\leq l$.

Theorem 8

$$k\text{-HS} \simeq \text{SU-}k\text{-HS} \simeq \text{DU-}k\text{-HS}\ .$$

In particular, taking $k = 2$ gives the result for vertex cover:

$$\text{VC} \simeq \text{SU-VC} \simeq \text{DU-VC}\ .$$

Proof. (k-HS $\preceq$ SU-k-HS) Let A be an oracle for SU-k-HS and let (U, S, l) with nodes $U = \{x_1, \ldots, x_n\}$ and sets $S = \{C_1, \ldots, C_m\}$ be our input instance. Let $z, \bar{z}$ be 2 nodes not in U. For i going from 1 to m, we will find a smallest hitting set a of $S_i = \{C_1, \ldots, C_i\}$. Initially, $a = \emptyset$ is a smallest hitting set of S_0. Suppose that we have a smallest hitting set a of S_{i-1}.

Let $L = (l_1, \ldots, l_n)$ be an ordering of U with the nodes of C_i first. Let sets b, c be initially empty. For j going from 1 to n, we will maintain the invariant that at step j, if S_i has a hitting set of size $\leq |a|$, then the lexicographically largest such (according to the order L, where l_1 is the most significant) contains all of b and none of c, and $b \cup c = \{l_1, \ldots, l_j\}$. Suppose the invariant holds for $j - 1$. We use A and the following lemma (with $b' = b \cup \{l_j\}$) to discover whether $b \cup \{l_j\}$ is contained in a hitting set of S_i of size $\leq |a|$.

Lemma 9. *Let a be a smallest hitting set of S_{i-1}, $b' \subseteq U$, and $T = S_{i-1} \cup \{\{z, \bar{z}\}\} \cup \{\{\bar{z}, x\} \mid x \in a\} \cup \{\{z, x\} \mid x \in b'\}$. Then T has a hitting set of size $\leq |a| + 1$, and it is unique iff S_i does not have a hitting set of size $\leq |a|$ that contains b'.*

Proof. $a \cup \{z\}$ is a hitting set of T of size $\leq |a| + 1$. There is no other containing z, and any without z contains b'. □

If the answer is yes (which corresponds to an oracle answer of no), we add l_j to b, otherwise to c. If c ever contains all of C_i then S_i has no hitting set of size $\leq |a|$, in which case, letting l be an arbitrary element of C_i, $a \cup \{l\}$ is a smallest hitting set of S_i. Otherwise, we continue applying the lemma, adding elements to either b or c, until we have a hitting set of S_i of size $\leq |a|$. Once we have a smallest hitting set a for S_m, we simply compare $|a|$ to l. The reduction uses poly(n) time and makes $\leq mn$ oracle calls, each with $\leq n + 2$ nodes, so it is efficient. □

6.2 k-Hypergraph Independent Set

The *k-Hypergraph Independent Set* problem (k-HIS) is, given a k-hypergraph (i.e. where each edge contains $\leq k$ vertices) and an integer l, decide whether there is a set I of vertices of size $\geq l$ such that no edge is contained in I.

Corollary 10

$$k\text{-HIS} \simeq \text{SU-}k\text{-HIS} \simeq \text{DU-}k\text{-HIS} \ .$$

In particular, taking $k = 2$ gives the result for independent set:

$$\text{IS} \simeq \text{SU-IS} \simeq \text{DU-IS} \ .$$

Proof. Follows immediately from theorem 8 by observing that a set of vertices is independent and of size $\geq l$ iff its complement is a hitting set of the edges and of size $\leq n - l$.

6.3 Limitations

Although Hitting-Set (HS) and Set-Cover (SC) are duals of one another, it is not as obvious how the exponential complexities of k-HS and k-SC are related. The technique used to show theorem 8 does not seem to work for SC since constraints are represented by the universe elements and not the sets, so adding linearly many constraints to construct the oracle queries increases the parameter from n to cn for some $c > 1$, causing the reduction to be inefficient.

The situation for k-Coloring is similar.[1] Though we can easily construct an oracle query with the right logical properties, using only the techniques here will cause it to have linearly many more vertices, and thus only show that $c_{k\text{-Coloring}} \leq O(c_{\text{DU-}k\text{-Coloring}})$, where the constant factor in the big-Oh does not depend on k.

7 Constraint Optimization Problems

The following problems involve optimizing the number of constraints that are satisfied.

Max-k-SAT (Min-k-SAT) is the problem of deciding whether a given k-CNF with an integer weight for each clause has an assignment that satisfies at least (at most) some given weight l.

[1] Here we take 'uniqueness' of a coloring solution to mean 'unique up to permutations of the colors'.

Theorem 11

$$\text{Max-}k\text{-SAT} \simeq \text{SU-Max-}k\text{-SAT} \simeq \text{DU-Max-}k\text{-SAT} \ .$$

Proof. (Max-k-SAT $\preceq$ SU-Max-k-SAT) Let A be an oracle for SU-Max-k-SAT and $\phi \in k$-CNF, $l \in \mathbb{Z}$ be our input where ϕ has variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ with (wlog) nonzero weights $w_1, \ldots, w_m$. Let z be a variable not in ϕ. For i going from 1 to m, we will find the maximum weight l' that can be satisfied in $\phi_i = \{C_1, \ldots, C_m\}$ and an assignment a that satisfies that weight. Initially, $l' = 0$ and a is arbitrary. Suppose we know the maximum weight l' that can be satisfied in ϕ_{i-1} and a satisfies it. We want to find the maximum weight l'' that can be satisfied in ϕ_i. We consider 2 cases: $w_i > 0$ and $w_i < 0$.

If $w_i > 0$, then $l'' \in [l', l' + w_i]$. For each literal $l \in C_i$ and $t' \in (l', l' + w_i]$, we can use A and the following lemma (with b the partial assignment that sets $l = 1$, and $t = t' - w_i$) to discover whether some assignment with $l = 1$ satisfies weight $\geq t'$ in ϕ_i.

Lemma 12. *Let l' be the largest weight that can be satisfied in ϕ_{i-1} and let a satisfy that weight. Let $|w_i| > 0$ and $l' - |w_i| < t \leq l'$. Let b be a partial assignment and $\psi = \phi_{i-1} \cup \{|w_i| \cdot (z \to x_r = a_r) \mid r \in [n]\} \cup \{|w_i| \cdot (\bar{z} \to x_r = b_r) \mid r \in$ domain of $b\}$. Then ψ has an assignment that satisfies weight $\geq t + (n + |b|)|w_i|$, and it is unique iff b cannot be extended to an assignment that satisfies weight $\geq t$ in ϕ_{i-1}.*

Proof. a together with $z = 1$ satisfies weight $\geq l' + (n + |b|)|w_i| \geq t + (n + |b|)|w_i|$ in ψ. There is no other such assignment with $z = 1$, and any with $z = 0$ agrees with b. □

So we can use binary search to discover l'' using $\leq |C_i| \lg(w_i + 1)$ queries. If $l'' = l'$, then a satisfies weight l'' in ϕ_i and we are done. Otherwise, as a slight optimization, if the last query set the jth literal of C_i to 1, then we can safely set literal j to 1 and literals 1 through $j - 1$ to 0 in the partial assignment b, and at this point, we know that b can be extended to a full assignment satisfying weight l'' in ϕ_i *and* that b satisfies C_i (since $l'' > l'$). We can continue using the lemma to extend b to such an assignment using $n - j$ more queries to A.

If $w_i < 0$, then $l'' \in [l' + w_i, l']$. Let b be the partial assignment that makes C_i false. For each $t \in (l' + w_i, l']$ we can use 1 query to A and the lemma to discover whether there is an assignment satisfying weight $\geq t$ in ϕ_i. So we can use binary search to discover l'' using $\leq \lg(|w_i| + 1)$ queries. If $l'' = l' + w_i$, then a satisfies weight l''. Otherwise, we can continue using the lemma to extend b to a full assignment satisfying weight l'' in ϕ_i using $n - |C_i|$ more queries.

At the end, we simply compare l' to l. The reduction uses time polynomial in the size of the input and makes $\leq \sum_i (|C_i| \lg(|w_i| + 1) + n - |C_i|)$ oracle calls, each with $\leq n + 1$ variables, so it is efficient. □

Theorem 11 applies to integer weights, but the proof is robust and can easily be modified to accommodate rational weights, e.g. by first multiplying by the

LCM of the denominators. Since the weights used in the proof have the same size as those of the input, the same proof works for the problem restricted to unit weights. The same proof also works for nonnegative weights, or any combination of the above.

Negating the input weights together with the algorithm in the proof also gives an efficient reduction from Min-k-SAT to SU-Min-k-SAT. The only rub is that this reduction is not correct if we restrict to nonnegative weights. To handle this case, we use a different reduction below.

Theorem 13. *If we restrict to nonnegative integer weights,*

$$\text{Min-}k\text{-SAT} \simeq \text{SU-Min-}k\text{-SAT} \simeq \text{DU-Min-}k\text{-SAT} .$$

Two clauses are in conflict with each other, i.e. cannot be simultaneously unsatisfied, iff they share a variable, but positively in the one and negatively in the other. We want to find a maximum weight independent set in the graph with the clauses as vertices and edges between conflicting clauses. But even if we restricted to unit weights, we cannot simply invoke corollary 10 since the parameter here is variables, not clauses. Thus it seems like we need another proof.

It should be pointed out that the proof below does not work if we restrict to unit weights since the construction of ψ uses non-unit weights. There may be some more convoluted construction using the same ideas but that avoids this technical point.

Proof. (Min-k-SAT $\preceq$ SU-Min-k-SAT) Let A be an oracle for SU-Min-k-SAT and $\phi \in k$-CNF, $l \in \mathbb{Z}$ be our input where ϕ has variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ with (wlog) positive weights $w_1, \ldots, w_m$. Let z be a variable not in ϕ. For i going from 1 to m, we will find the minimum weight l' that can be satisfied in $\phi_i = \{C_1, \ldots, C_i\}$ and an assignment a that satisfies that weight. Initially, $l' = 0$ and a is arbitrary. Suppose we know the minimum weight l' that can be satisfied in ϕ_{i-1} and a satisfies it. The minimum weight l'' that can be satisfied in ϕ_i is in the interval $[l', l' + w_i]$. Letting b be the partial assignment that makes C_i false, we can use the following lemma and binary search to discover l'' using $\lg(w_i + 1)$ calls to A.

Lemma 14. *Let l' be the smallest weight that can be satisfied in ϕ_{i-1} and let a satisfy that weight. Let $w_i > 0$ and $l' \leq t < l' + w_i$. Let b be a partial assignment and $\psi = \phi_{i-1} \cup \{w_i \cdot (z \to x_r \neq a_r) \mid r \in [n]\} \cup \{w_i \cdot (\bar{z} \to x_r \neq b_r) \mid r \in$ domain of $b\} \cup \{(n - |b|)w_i \cdot (z)\}$. Then ψ has an assignment that satisfies weight $\leq t + nw_i$, and it is unique iff b cannot be extended to an assignment that satisfies weight $\leq t$ in ϕ_{i-1}.*

Proof. a together with $z = 1$ satisfies weight $\leq l' + nw_i \leq t + nw_i$ in ψ. There is no other such assignment with $z = 1$, and any with $z = 0$ agrees with b. □

If $l'' = l' + w_i$, then a satisfies weight l'' in ϕ_i. Otherwise, we use the lemma (with $t = l''$) $n - |C_i|$ times to extend b to a full assignment achieving weight l''

in ϕ_i. At the end, we simply compare l' to l. The reduction uses time polynomial in the size of the input and makes $\leq \sum_i (\lg(w_i + 1) + n - |C_i|)$ oracle calls, each with $\leq n + 1$ variables, so it is efficient. □

8 Conclusions

We show a simple technique to settle conjecture (1) as well as many questions relating the exponential complexity of similar parameterized constraint satisfaction problems to their unique-solution counterparts. Our problem list here is not intended to be exhaustive but demonstrative. Theorem 11 shows the robustness of the technique, allowing considerable variation in problem specification without disturbing the proof.

Relating the exponential complexities of such problems to their unique-solution counterparts under the *promise* of at most 1 solution appears to be harder. Current techniques [10,1] for that problem use oblivious hashing (i.e. not looking at the input but only its size), but fall short of such strong results as are here. It seems like new, non-oblivious techniques are needed.

References

1. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of Unique k-SAT: An isolation lemma for k-CNFs. Journal of Computer and System Sciences 74(3), 386–393 (2008)
2. Dubois, O., Dequen, G.: A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 248–253 (2001)
3. Grandjean, E., Büning, H.: SAT-problems and reductions with respect to the number of variables. Journal of Logic and Computation 7(4), 457–471 (1997)
4. Håstad, J.: Some optimal inapproximability results. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM Press, New York (1997)
5. Impagliazzo, R., Paturi, R., Zane, F.: Which Problems Have Strongly Exponential Complexity? Journal of Computer and System Sciences 63, 512–530 (2001)
6. Impagliazzo, R., Paturi, M.: On the complexity of k-SAT. Journal of Systems Sciences 62(2), 367–375 (2001)
7. Iwama, K., Tamaki, S.: Improved upper bounds for 3-SAT. ECCC Report TR03-053 (2003)
8. Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L.: Determining computational complexity from characteristic phase transitions. Nature 400(8), 133–137 (1999)
9. Plass, M., Aspval, B., Tarjan, R.: A linear time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters 8, 121–123 (1979)
10. Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47, 85–93 (1986)