On the possibility of faster SAT algorithms

Mihai Pătraşcu*
AT&T Labs
Florham Park, NJ

Ryan Williams†
IBM Almaden Research Center
San Jose, CA

Abstract

We describe reductions from the problem of determining the satisfiability of Boolean CNF formulas (CNF-SAT) to several natural algorithmic problems. We show that attaining any of the following bounds would improve the state of the art in algorithms for SAT:

- an $O(n^{k-\varepsilon})$ algorithm for k-Dominating Set, for any $k \geq 3$,
- a (computationally efficient) protocol for 3-party set disjointness with $o(m)$ bits of communication,
- an $n^{o(d)}$ algorithm for d-SUM,
- an $O(n^{2-\varepsilon})$ algorithm for 2-SAT formulas with $m = n^{1+o(1)}$ clauses, where *two* clauses may have unrestricted length, and
- an $O((n+m)^{k-\varepsilon})$ algorithm for HornSat with k unrestricted length clauses.

One may interpret our reductions as new attacks on the complexity of SAT, or sharp lower bounds conditional on exponential hardness of SAT.

1 Introduction

Do NP-hard problems require us to exhaustively search over all solutions? This basic question is the heart of the P versus NP problem. Over the last decade or so, the area of exact algorithms for NP-hard problems has seen vast development. Hundreds of papers have been written on how to solve hard problems exponentially faster than brute-force search. Most results report algorithms running in $O^\star(2^{\delta t})$ time for a constant $\delta < 1$, where $O^\star(2^t)$ would be the runtime of an exhaustive search.[1] We say that an *improved algorithm* for a problem to be one with a runtime bound of the above type. A surprising number of search problems have been shown to exhibit improved algorithms. To cite a representative list of papers at this point is not really possible, but the surveys by Woeginger [31, 32] are somewhat comprehensive on recent work.

For certain key problems, we still do not know improved algorithms. The most famous of these is the "original" NP-complete problem: the satisfiability problem for Boolean formulas in conjunctive normal form, which we abbreviate as CNF-SAT. Satisfiability is so ubiquitous that an entire conference on theory and applications of the problem is held annually [29]. A sequence of papers has given algorithms for CNF-SAT with $2^{n-o(n)} \cdot \mathrm{poly}(m)$ runtime [28, 30, 10, 11, 12, 13], where n is the number of variables and m is the number of clauses. The current best, implicit in Calabro, Paturi, and Impagliazzo [4] and observed by Dantsin and Hirsch [14], is a deterministic algorithm that runs in

$$2^{n\left(1-\frac{1}{O(\log(m/n))}\right)} \cdot \mathrm{poly}(m) \text{ time.}$$

One implication of this algorithm that for every constant $c > 0$, there is an $\delta(c) < 1$ such that SAT with n variables and cn clauses can be solved in $O(2^{\delta(c)n})$ time. An algorithm with such a guarantee was first given by Arvind and Schuler [2].

It does not appear that the current approaches will lead to a bound of $O^\star(2^{\delta n})$ for general CNF-SAT, with

*This work was done while the author was a Raviv Postdoctoral Fellow at IBM Almaden Research Center. Email: mip@alum.mit.edu.

†Parts of this work appeared in the author's PhD thesis [34] at Carnegie Mellon University, supported by the NSF ALADDIN Center (NSF CCR-0122581). Currently supported by a Raviv Postdoctoral Fellowship at IBM Almaden. Email: ryanwill@us.ibm.com.

[1] The $O^\star$ notation suppresses polynomial factors in the input size.

constant $\delta < 1$. Nor do they seem promising for a more modest goal, namely a bound of $O^\star(2^{\delta n})$ for k-SAT, where k can be any constant (hidden in the asymptotic notation), but $\delta < 1$ is a universal constant independent of k.

One positive result in this direction follows from work of Calabro, Paturi, and Impagliazzo [4]. Their duality between clause density and clause width shows that an improved algorithm for CNF-SAT with n variables and $f(n)$ clauses would follow from an algorithm for k-SAT with n variables and $f(n)$ clauses that runs in time $2^{\delta n}$, where $\delta < 1$ is any constant and $k \geq \frac{1}{\delta} \log f(n) + \Omega(1)$. Hence, it suffices to restrict attention to formulas with "logarithmic length" clauses.

In the opposite direction, Impagliazzo and Paturi [23] conjectured that there is no improved algorithm for CNF-SAT. They showed that an $O^\star(2^{\delta n})$ algorithm for CNF-SAT implies an $O^\star(2^{\delta n \cdot (1-1/(e \cdot k))})$ algorithm for k-SAT. Thus, an improvement to the state of the art for CNF-SAT will immediately imply algorithms for the family of k-SAT problems. This offers some evidence that any improvement for CNF-SAT is unlikely, though one must be careful in judging how compelling the evidence is. For example, if $\delta = .99$, the implied k-SAT algorithm is only an improvement over the known $O^\star((2 - \frac{2}{k+1})^n)$ algorithm for $k > 107$.

We have expended significant effort attempting to either find an improved algorithm, or give interesting evidence against its possibility. In this paper, we present several hypotheses which appear plausible, given the current state of knowledge. We prove that if any of the hypotheses are true, then CNF-SAT has an improved algorithm. One can either interpret our reductions as new attacks on the complexity of CNF-SAT, or lower bounds (ruling out all hypotheses) conditional on the hardness of CNF-SAT.

The proofs themselves exhibit strong connections between SAT and k-Dominating Set, 2-SAT, Horn-Sat, the 3-party set disjointness problem in communication complexity, and the d-SUM problem. A side result of our work is a further elucidation of the relationships between open problems in exact algorithms and celebrated open problems in other areas. (Weaker forms of these connections have been reported before in the literature, as we will discuss later.)

More precisely, we show that CNF-SAT can be solved in $2^{\delta n} \cdot \text{poly}(m)$ time for some $\delta < 1$, if any one of the following hypotheses are true:

1. (Section 2) Define k-Dominating Set to be the problem of finding a k-set S of nodes in graph, where all nodes are either adjacent to S are in S. The problem is a special case of the celebrated k-*junta* problem in learning theory. We know that for all $k \geq 7$, the k-Dominating Set problem on n node graphs can be solved in $n^{k+o(1)}$ time [17].

   Hypothesis: *For some $k \geq 3$ and $\varepsilon > 0$, k-Dominating Set is in $O(n^{k-\varepsilon})$ time.*

2. (Section 3) Define 2Sat+2Clauses to be the problem of satisfying a 2-CNF formula on n variables and m clauses, conjoined with two additional clauses of arbitrary length. We know the problem has an $O(mn + n^2)$ time algorithm, via a natural reachability algorithm on graphs.

   Hypothesis: *For $m = n^{1+o(1)}$ and some constant $\varepsilon > 0$, 2Sat+2Clauses is in $O(n^{2-\varepsilon})$ time.*

3. (Appendix A) Define HornSat+kClauses to be the problem of satisfying a Horn CNF formula conjoined with k additional clauses of arbitrary length. We know the problem is in $O(n^k \cdot (m+n))$ time, where n is the number of variables and m is the number of clauses.

   Hypothesis: *For some $\varepsilon > 0$ and $k \geq 2$, HornSat+kClauses is in $O((n+m)^{k-\varepsilon})$ time.*

A weaker problem than CNF-SAT is the k-SAT problem for arbitrary $k \geq 3$. All known algorithms for k-SAT have increasingly longer running times as k increases. In particular, the running times for k-SAT are all of the form $2^{(1-1/\Theta(k))n}$. Impagliazzo and Paturi [23] have shown that the running time must indeed increase with k, assuming the *Exponential Time Hypothesis* (ETH) which states that 3-SAT cannot be solved in $2^{o(n)}$ time.

The following hypothesis from commmunication complexity implies that the k-SAT problem can be solved in $O(1.74^n)$ time for all constants k. Note this does not have any known implications for general CNF-SAT. However, it does imply that CNF-SAT with constant clause density can also be solved in this running time, by results of Calabro, Impagliazzo, and Paturi [4].

- (Section 4) In 3-Party Set Disjointness, there are three parties and subsets $S_1, S_2, S_3 \subseteq [m]$, where the ith party has access to all sets except for S_i. (This problem is also called "set disjointness in the number on the forehead model".) The parties wish to determine if $S_1 \cap \cdots \cap S_k = \varnothing$, with minimum communication. An m-bit communication protocol is trivial, and a major open problem is to determine whether 3-party set disjointness has a sublinear communication protocol.

  Hypothesis: *There is a protocol for 3-party set disjointness where the parties communicate $o(m)$ bits and perform $2^{o(m)}$ time computations.*

Finally, we show a tight correspondence between the Exponential Time Hypothesis and the difficulty of the d-SUM problem. The d-SUM problem asks whether a set of N numbers contains a d tuple that sums to zero. The best known algorithm runs in $O(n^{\lceil d/2 \rceil}/\text{polylog}n)$ time. In computational geometry, d-SUM is a basis for a hardness theory for many problems. We show the following hypothesis implies k-SAT can be solved in $2^{o(n)}$ time for all constant k:

- (Section 5) **Hypothesis:** *There is a $d < N^{0.99}$ such that d-SUM on N numbers of $O(d \lg N)$ bits can be solved in $N^{o(d)}$ time.*

All our proofs use a special type of *divide and conquer*: we reduce CNF-SAT and k-SAT instances to mildly exponential-sized instances of the above problems, by enumerating short lists of partial assignments inside the instance and using the structure of the problem to encode satisfiability. This maneuver exponentially increases the problem size, but the task of combining subproblems to obtain a global solution becomes drastically easier.

While the above results can be seen as new attacks on the complexity of SAT, of course they can also be seen as *hardness results*. That is, if we assume that CNF-SAT and k-SAT for arbitrary constant k cannot be solved in less than 1.99^n time, then our work deduces a multitude of interesting lower bounds: strong lower bounds on dominating set, a nearly quadratic lower bound for finding a pair of nodes with no path between them (in a directed graph), $n^{\Omega(d)}$ lower bounds on d-SUM (which in turn imply other lower bounds in computational geometry), and an $\Omega(m)$ communication lower bound on computable protocols for 3-party set disjointness.

Remark on notation. All functions used in theorem statements are implicitly assumed to be efficiently computable.

2 SAT and Dominating Set

In Parameterized Complexity, k-Dominating Set is one of the canonical $W[2]$-complete problems [16]. Given an undirected graph on n nodes and m edges, the problem is to find a k-set S of nodes where every node of the graph is either in S, or is incident to a node in S. It is equivalent to finding a *set cover* of k sets. It is also a special case of the *k-junta problem* in learning theory, where we are given a set of examples from $\{0,1\}^n$, each labeled with a 0 or a 1, and wish to find a function on k variables that maps the examples to their corresponding labels.

For a long while, the best algorithm known for solving k-Dominating Set was the obvious $O(n^{k+1})$ brute-force algorithm. Fast matrix multiplication can improve this time bound slightly, as demonstrated by Eisenbrand and Grandoni [17]. Consider the special case of 2-dominating set. Take the Boolean adjacency matrix A of the graph G, complement it (flip 1's to 0's, and 0's to 1's) and multiply the resulting matrix with its transpose, *i.e.* compute $B = \overline{A} \cdot \overline{A}^T$. We have the following.

Proposition 2.1. *G has a 2-dominating set $\iff$ For some i and j, $B[i,j] = 0$.*

Proof. Let $M[i,:]$ denote the ith row of M and $M[:,j]$ denote the jth column of M. Let $V = [n]$ be the vertices of G. Then

$$\begin{aligned}
&\{i,j\} \text{ is a 2-dominating set} \\
\iff\ & (A \vee I)[i,:] \vee (A \vee I)[j,:] = \mathbf{1}, \text{the all-1's vector} \\
\iff\ & \langle \overline{(A \vee I)}[i,:], \overline{(A \vee I)}[j,:] \rangle = 0 \\
\iff\ & \langle \overline{(A \vee I)}[i,:], \overline{(A \vee I)}^T[:,j] \rangle = 0 \\
\iff\ & B[i,j] = 0.
\end{aligned}$$

□

Therefore 2-Dominating Set can be solved in $O(n^\omega)$ time, where $\omega < 2.376$ is the matrix multiplication exponent [8]. To generalize the algorithm to k-Dominating Set, let $v_1, \ldots, v_n$ be a list of the vertices, and $S_1, \ldots, S_{\binom{n}{k/2}}$ be a list of all $k/2$-sets of the vertices. Define an $\binom{n}{k/2} \times n$ Boolean matrix A_k, where

$$A_k[i,j] = 0 \iff v_j \text{ is dominated by } S_i.$$

Then, the product $B_k = A_k \times A_k^T$ is an $\binom{n}{k/2} \times \binom{n}{k/2}$ matrix, where $B_k[i,j] = 0$ iff $S_i \cup S_j$ is a dominating set.

Using Coppersmith's rectangular matrix multiplication [9], this algorithm can be implemented to run in $n^{k+o(1)}$ time for all $k > 7$.

Proposition 2.2. *For $k \geq 7$, k-Dominating Set can be solved in $n^{k+o(1)}$ time.*

Proof. Coppersmith [9] gave an algorithm for multiplying a $n \times n^{.294}$ matrix with a $n^{.294} \times n$ matrix in $n^{2+o(1)}$ ring operations. The product $B_k = A_k \times A_k^T$ is essentially a product of an $N \times N^{2/k}$ matrix with a $N^{2/k} \times N$ matrix, for $N = \binom{n}{k/2}$. But $2/k \leq 0.294$ when $k \geq 7$, so Coppersmith's algorithm can be applied. □

This method is almost $\Omega(n)$ faster than the trivial algorithm, but still requires that one examine every possible k-set of vertices. A major open question in

parameterized algorithms is whether a time bound even slightly better than n^k is possible for k-DOMINATING SET.

HYPOTHESIS 1. *There exist $k \geq 3$ and $\varepsilon \in (0, k)$ such that k-DOMINATING SET is in $O(n^{k-\varepsilon})$ time.*

We know of no results suggesting that Hypothesis 1 may be false. Surprising algorithms have been found for hard parameterized problems in the past. For example, the $W[1]$-complete problem k-CLIQUE has an $O(n^{.793k})$ algorithm [27]. However, if one believes that there is even a slight improvement for k-DOMINATING SET, then one must believe there is an improved algorithm for CNF-SAT.

THEOREM 2.1. *Hypothesis 1 implies that CNF-SAT has an improved algorithm.*

A slightly weaker connection between k-DOMINATING SET and SAT has been established in the literature.

THEOREM 2.2. (CHEN *et al.* [6], THEOREM 5.4) *k-DOMINATING SET is not in $f(k)n^{o(k)}$ time, for any function f, unless $FPT = W[1]$.*

That is, it was known that if k-DOMINATING SET is in $n^{o(k)}$ time, then k-SAT has a $2^{o(n)}$ algorithm (*i.e.*, the Exponential Time Hypothesis is false [23]). However, this result does not say anything *a priori* about the complexity of CNF-SAT. As far as we know, it is consistent with current knowledge that k-SAT has a $2^{o(n)}$ algorithm, yet CNF-SAT still does not have an improved algorithm.

Theorem 2.1 is a special case of the following lemma.

LEMMA 2.1. *Suppose there is an integer $k \geq 3$ and function f such that k-DOMINATING SET is solvable in $O(n^{f(k)})$ time. Then CNF-SAT is in $O\left((m + k2^{\frac{n}{k}})^{f(k)}\right)$ time.*

Proof of Lemma 2.1. Fix $k \geq 3$. Let F be a CNF formula with n variables; we build a corresponding graph G_F. Without loss of generality, assume k divides n. Partition the set of its variables into k parts of n/k size each. For each part, make a list of all $2^{n/k}$ partial assignments to variables in that part. Each partial assignment shall correspond to a node in G_F.

Make each of the k parts a clique, so there are k disjoint $2^{n/k}$-cliques with $O(2^{2n/k})$ edges. Now add m more nodes, one for each clause, and place an edge from a partial assignment node to a clause node iff the partial assignment satisfies the clause. Finally, for each partial assignment clique, add a dummy node that has edges to all nodes in that clique, but no edges to clause nodes or any other clique.

Consider a k-dominating set S in G_F. Note that no clause node is in S, otherwise some dummy node would not be dominated. Suppose S has two (or more) partial assignment nodes from the same clique. Then there is some clique for which S chose no node; but then S does not dominate its dummy node. Therefore, the collection of partial assignments corresponding to the nodes of S is some satisfying variable assignment, since all clause nodes are dominated.

The total number of nodes is $k2^{n/k} + m + k$, so the lemma follows. □

The above result can be rephrased in terms of the k SET COVER problem. Here, one is given a collection $\mathcal{C}$ of n sets over a universe of size m, and the task is to find a $\mathcal{S} \subseteq \mathcal{C}$ so that $|\mathcal{S}| = k$ and every element of the universe is contained in some set of $\mathcal{S}$. By associating the set $S_v = N(v) \cup \{v\}$ with each vertex of a graph $G = (\{v_1, \ldots, v_n\}, E)$, and setting the universe to be $\{v_1, \ldots, v_n\}$, a k set cover for the collection $\{S_{v_1}, \ldots, S_{v_n}\}$ is a k-dominating set for G. Thus an immediate corollary of Theorem 2.1 is the following.

THEOREM 2.3. *If there is $k \geq 2$, k SET COVER can be solved in $O(n^{k-\varepsilon})$ time for a collection of n sets over a universe of size poly$(\log n)$, then SAT has an improved algorithm on instances with poly(n) clauses.*

2.1 A partial converse An intriguing question is whether or not a converse to Theorem 2.1 holds. That is, does the existence of a good CNF-SAT algorithm imply the existence of a good k-DOMINATING SET algorithm? One would like to encode a graph into a small CNF formula, whereby an assignment to $k \log n$ variables satisfies the formula iff the graph has a k dominating set.

We can show a partial converse of this kind. Define the problem CNF-SAT-S to have instances of the form (F, S), where F is a CNF formula and S is a subset of the variables of F. The problem is to find an assignment a to S such that $F[S = a]$ is a satisfiable Horn formula. (Recall a Horn formula is a CNF formula where each clause has at most one positive literal. Horn satisfiability is well-known to be in P.) In other words, the problem is to verify that S is a "backdoor set" [33] of variables for F, with respect to a subsolver for Horn formulas.

CNF-SAT-S is perhaps more difficult than CNF-SAT in terms of exact algorithms, in that we are only allowed to set variables within a certain subset (other variables are out of our control), and the assignment we find must not only extend to a satisfying assignment

for the formula, but also extend easily to a satisfying assignment. CNF-SAT-S is essentially equivalent to the CIRCUITSAT problem, where instead of a CNF F, we are given a circuit C and wish to set its n input variables so that the circuit is satisfied (here, the input variables play the role of S). Note that CNF-SAT-S can be solved in $O^\star(2^{|S|})$ and CIRCUITSAT can be solved in $O^\star(2^n)$. We show that a time improvement for CNF-SAT-S implies a better dominating set algorithm.

THEOREM 2.4. *If* CNF-SAT-S *is in* $O(f(m+n)\cdot 2^{\delta|S|})$ *time for some* $\delta \in (0,1)$ *and function* f*, then* k-DOMINATING SET *is in* $O(f(kn^2)\cdot n^{\delta k})$ *time.*

Notice that for any $\delta < 1$ and constant $c > 1$, $n^{c+\delta k} \in O(n^{k-\varepsilon})$ for sufficiently large k and sufficiently small $\varepsilon > 0$. So if f is a polynomial in the above, then the implication is indeed an improved dominating set algorithm for large enough k.

Proof of Theorem 2.4. Let $G = (V, E)$ be given and let $n = |V|$. We will set up a formula F_G. Define a $(k \log n)$-set of variables $S = \{x_{1,1}, \ldots, x_{1,\log n}, x_{2,1}, \ldots, x_{2,\log n}, \ldots, x_{k,1}, \ldots, x_{k,\log n}\}$. These variables will represent the binary encoding of a dominating set: specifying an assignment to the variables of S will be equivalent to specifying a k-set of vertices in G. For $j \in [k]$ and $i \in [n]$, let $v_{j,i}$ be kn variables representing the n vertices in the graph G. Informally, $v_{j,i} = 1$ if and only if the jth vertex in the candidate dominating set does *not* dominate the ith vertex of G.

The clauses of F_G check that the k-set guessed by S is indeed dominating. Define $x^1_{i,j} := x_{i,j}$, and $x^0_{i,j} := \neg x_{i,j}$. Fix a vertex $u \in V$ in the following. Let $b_1 b_2 \cdots b_{\log n}$ be a binary encoding of u. Define the neighborhood $N(u) := \{v \mid \{u, v\} \in E\}$. Let us index the elements of $V - N(u)$ as

$$V - N(u) = \{u_{i_1}, \ldots, u_{i_{n-\deg(u)}}\}.$$

Then for all $j = [k]$ and $d = 1, \ldots, n - \deg(u)$, add the clause:

$$(x^{1-b_1}_{j,1} \vee \cdots \vee x^{1-b_{\log n}}_{j,\log n} \vee v_{j,i_d})$$

to F_G. Intuitively, this clause says that the i_dth vertex is not dominated by the jth vertex in the candidate dominating set. Note there are $O(kn^2)$ clauses of this kind, one for each possible setting of j, u, and d. For all vertices $i = 1, \ldots, n$, add the clause

$$(\neg v_{1,i} \vee \neg v_{2,i} \vee \cdots \vee \neg v_{k,i})$$

to F_G. These clauses stipulate that at least one of the k vertices in the candidate dominating set must dominate the ith vertex, for all i. This completes the description of F_G.

Observe that once all of the variables in S are set to values (say, an assignment a), all remaining clauses in F_G are either of the form (x) or $(\neg x \vee \neg y \vee \cdots \vee \neg z)$. That is, the remaining formula is Horn, and thus satisfiability for it can be determined in linear time.

We claim that the Horn formula $F_G[S = a]$ is satisfiable if and only if a denotes a dominating set of G. First, since every clause in $F_G[S = a]$ is either a positive literal or a collection of negative literals, observe that $F_G[S = a]$ is unsatisfiable if and only if the clauses $(v_{1,i})$,$(v_{2,i})$, …, $(v_{k,i})$ appear in $F_G[S = a]$, for some $i = 1, \ldots, n$. A clause $(v_{j,i})$ appears iff the literals $x^{1-b_1}_{j,1}$, …, $x^{1-b_{\log n}}_{j,\log n}$ are set false and $(x^{1-b_1}_{j,1} \vee \cdots \vee x^{1-b_{\log n}}_{j,\log n} \vee v_{j,i})$ is a clause in F_G. But $x^{1-b_1}_{j,1}$, …, $x^{1-b_{\log n}}_{j,\log n}$ are false iff the jth vertex in the set S has binary encoding $b_1 b_2 \cdots b_{\log n}$, and the above clause is in F_G iff the vertex with binary encoding $b_1 b_2 \cdots b_{\log n}$ does not have the ith vertex as a neighbor. Therefore the clauses $(v_{1,i})$,$(v_{2,i})$, …, $(v_{k,i})$ appear in $F_G[S = a]$ iff for all $j = 1, \ldots, n$, the jth vertex in S does not have the ith vertex as a neighbor, *i.e.* the set S is not dominating.

Hence the pair $(F, \{x_{i,j} \mid i \in [k],\ j \in [n]\})$ is an instance of CNF-SAT-S with $|S| = O(k \log n)$ and $|F| = O(kn^2)$. From the above discussion, it follows that a satisfying assignment to S is equivalent to a dominating set in the graph. □

3 SAT and 2-SAT

2-SAT is the well-known restriction of CNF-SAT to instances with at most two literals per clause. The problem has fast algorithms, being solvable in linear time [3]. One possible direction for finding an improved algorithm for CNF-SAT is to try reducing it to 2-SAT in some interesting way. As we do not believe $\mathsf{P} = \mathsf{NP}$, this reduction should be exponential, but not terribly exponential (say, of $2^{(1-\varepsilon)n}$ total size for some $\varepsilon > 0$). If such a reduction existed, the linear time algorithm for 2-SAT would imply an improved CNF-SAT algorithm.

The results of this section are inspired by this potential direction. We present a minor generalization of 2-SAT, which we call 2SAT+2CLAUSES. This problem admits a straightforward $O(mn + n^2)$ time algorithm. We prove that if it has a sub-quadratic time algorithm, then CNF-SAT has an improved algorithm, via a "mildly exponential" reduction.

Define an instance of 2SAT+2CLAUSES to be a 2-CNF formula that is conjoined with at most two

additional clauses of arbitrary length. For example,

$$(\neg x_1 \vee x_4) \wedge (x_2 \vee \neg x_3) \wedge (x_5 \vee x_6)$$
$$\wedge(x_1 \vee x_2 \vee \cdots \vee x_6) \wedge (\neg x_1 \vee \neg x_2 \vee \cdots \vee \neg x_6)$$

is a 2SAT+2CLAUSES instance. Similar "mixed" instances have been studied in the past, especially in the average-case setting (cf. [26]) where the satisfiability of random formulas has been analyzed. We start with a simple algorithm for solving this problem.

THEOREM 3.1. *2SAT+2CLAUSES is in $O(mn + n^2)$ time, where n is the number of variables and m is the total number of clauses.*

Proof. Let F be an instance and C_1, C_2 be its two arbitrary size clauses. Construct a directed graph G where each node is a literal of F (a variable or its negation) and there is an edge from ℓ_i to ℓ_j iff $(\neg\ell_i \vee \ell_j) \in F - C_1 - C_2$. Note the edge relation plays the role of implication.

We start by preprocessing G. Compute the transitive closure of G in $O(mn + n^2)$ time using standard techniques [7]. This results in a Boolean matrix M where $M[i,j] = 1 \iff \ell_i \to \ell_j$, for all literals ℓ_i and ℓ_j. If there is a variable x such that $x \to \neg x$ and $\neg x \to x$ then return *unsatisfiable*.

For every pair of literals ℓ_i in C_1 and ℓ_j in C_2, we will determine if $\ell_i \wedge \ell_j$ can be extended to a satisfying assignment for all of F. Observe that

$$(\ell_i \wedge \ell_j) \equiv \neg(\neg\ell_i \vee \neg\ell_j) \equiv \neg(\ell_i \to \neg\ell_j).$$

We can determine in $O(1)$ time if $(\ell_i \to \neg\ell_j)$ is true, by looking up the corresponding entry in M. If that entry is 0, then we return *satisfiable*, since it means that $\ell_i =$ true and $\ell_j =$ true can be extended to a satisfying assignment for F. Otherwise we move to the next pair of literals. If all pairs of literals have been exhausted, we return *unsatisfiable*. □

Note the above proof shows that the following purely graph-theoretic problem is at least as difficult as the 2SAT+2CLAUSES problem: *given a directed graph $G = (V, E)$ and subsets $S, T \subseteq V$, determine if there is some $s \in S$ and $t \in T$ with no path from s to t.*[2]

Our second hypothesis is that there is a better algorithm for 2SAT+2CLAUSES:

HYPOTHESIS 2. *For some $\varepsilon > 0$, 2SAT+2CLAUSES is in $O((m+n)^{2-\varepsilon})$ time.*

THEOREM 3.2. *Hypothesis 2 implies that CNF-SAT has an improved algorithm.*

Proof. We show how to embed an CNF formula F into an (exponentially sized) 2SAT+2CLAUSES instance F'. In particular, if F has n variables and m clauses, then F' will have $O(2^{n/2} + m + n)$ variables and $O(m2^{n/2} + mn)$ clauses. This immediately implies the claim of the theorem.

The variables of F' will be of the form x_S, where S is a proposition that is either a conjunction of literals in F, or a disjunction of literals in F. Intuitively, we want x_S to be true if and only if S is true. We therefore want $\neg x_S \iff x_{(\neg S)}$, which we capture with the clauses

$$(x_S \vee x_{(\neg S)}) \wedge (\neg x_S \vee \neg x_{(\neg S)})$$

for every proposition S in the below.

We split the set of n variables into two parts of $n/2$ size. (WLOG we may assume n is even.) For both parts, list all the possible $2^{n/2}$ partial assignments P to the variables of that part. We interpret each P as a conjunction of $n/2$ literals in the natural way. (For example we would interpret the partial assignment $y_1 = 1, y_2 = 0, y_3 = 1$ as the conjunction $y_1 \wedge \neg y_2 \wedge y_3$.) For each P, make two variables x_P and $x_{(\neg P)}$ in F'. Let $P_1, \ldots, P_{2^{n/2}}$ and $Q_1, \ldots, Q_{2^{n/2}}$ be the partial assignments of the first and second part, respectively. The two arbitrary size clauses in F' will be:

$$(x_{P_1} \vee \cdots \vee x_{P_{2^{n/2}}}) \text{ and } (x_{Q_1} \vee \cdots \vee x_{Q_{2^{n/2}}}).$$

The remaining clause structure of F can be represented using a 2-CNF. For each clause C of F, let C_1 (C_2) be the disjunction of literals in C involving variables from the first (second) part, respectively. Make variables x_{C_1}, $x_{(\neg C_1)}, x_{C_2}$, and $x_{(\neg C_2)}$, and include the clause

$$x_{C_1} \vee x_{C_2}.$$

Finally, we relate the clause variables to the partial assignment variables. For each variable y_i of F, we have the variables x_{y_i} and $x_{\neg y_i}$ in F'. For every C_i of the form $(y_i \vee D)$, F' has the clause

$$x_{(\neg C_i)} \to x_{\neg y_i}.$$

For every partial assignment P that sets $y_i = 1$, F' has the clause $x_P \to x_{y_i}$, and for every partial assignment P that falsifies a clause C_i, F' also has $x_P \to x_{(\neg C_i)}$.

We now prove that F' is satisfiable iff F is satisfiable. It is not hard to see that, if F is satisfiable by

[2] Let G be the directed graph from the proof. In linear time we can rule out if G has a path from some x to $\neg x$ and back to x. Let S be the literals in C_1 and let T be the negations of those literals in C_2. There is an $s \in S$ and $t \in T$ with no path iff there are two literals, one in C_1 and one in C_2, that can be extended to a satisfying assignment to F.

assignment a, then F' is satisfied by setting $x_S = 1$ if and only if the proposition S is satisfied by a.

The other direction (F' is satisfiable implies F is satisfiable) is a little more involved. First, we claim that if $x_{P_j} = 1$ for an $n/2$-variable partial assignment P_j, then for all $j \neq i$, $x_{P_i} = 0$. Let y be a variable in which P_i and P_j differ in assignment. Without loss of generality, $x_{P_j} \to x_y$ and $x_{P_i} \to x_{\neg y} \to \neg x_y$, therefore only one of x_{P_i} and x_{P_j} can be true for all $i \neq j$.

Suppose there is a satisfying assignment to F' with $x_A = 1$ and $x_{A'} = 1$, where A (A') is a partial assignment for the variables in the first (respectively, second) part. We claim that the variable assignment denoted by A and A' satisfies F. For suppose this assignment falsified a clause C, and C_1 (C_2) is the disjunction of literals in C involving variables from the first (respectively, second) part. Then by definition, $x_A \to x_{(\neg C_1)}$ and $x_{A'} \to x_{(\neg C_2)}$. But the satisfying assignment to F' sets $x_A = 1$ and $x_{A'} = 1$, so $x_{(\neg C_1)} \wedge x_{(\neg C_2)}$ is satisfied by the assignment, and hence $(\neg x_{C_2}) \wedge (\neg x_{C_2})$ is also satisfied. However, the satisfying assignment to F' also satisfies the clause $(x_{C_1} \vee x_{C_2})$ in F'. This is a contradiction. □

4 SAT and the Communication Complexity of Disjointness

Next, we connect a major open problem in communication complexity to the feasibility of CNF-SAT. In the *k-party disjointness problem*, we have k computational parties, and given subsets $S_1, \ldots, S_k \subseteq [m]$ where the ith party has access to all sets except for S_i. (This problem is also called "set disjointness in the number on the forehead model".) The parties wish to determine if $S_1 \cap \cdots \cap S_k = \varnothing$, without communicating many bits.

The disjointness problem has received much attention in the complexity community, due to its fundamental nature and notorious difficulty. The best known upper bound on its communication complexity is $O(km/2^k)$, by a protocol of Grolmusz [22]. It has been only recently that progress has been made on lower bounds. For a long time, only an $\Omega(\log m)$ lower bound was known, but Lee and Shraibman [25] and Chattopadhyay and Ada [5] have shown $\Omega\left(m^{1/(k+1)}\right)$ lower bounds on communication, which hold even when the parties can use randomness.

We show that reasonably computable nondeterministic protocols for 3-party set disjointness with $o(m)$ communication complexity would imply a breakthrough in satisfiability algorithms. In particular, CNF SAT would have an algorithm running in $2^{\omega n/3} \cdot 2^{o(m)}$ time, where ω is the matrix multiplication exponent, n is the number of variables, and m is the number of clauses. By the Sparsification Lemma of Impagliazzo, Paturi, and Zane [24], such an algorithm can be used to solve the k-SAT problem in $O(1.74^n)$ time *for all* k. As a consequence of the clause density and clause width duality of Calabro, Impagliazzo, and Paturi [4], this k-SAT algorithm can be used to solve CNF SAT for formulas with cn clauses and n variables in the same running time, for any fixed $c > 0$. For those researchers who believe that an algorithm with such a running time is not possible, they may interpret our theorem as compelling evidence that 3-party disjointness does require $\Omega(m)$ bits of communication.

Let $\omega < 2.376$ be the matrix multiplication exponent [8].

THEOREM 4.1. *If 3-party disjointness has a nondeterministic protocol with communication complexity $o(m)$ such that all parties run in deterministic $2^{o(m)}$ time, then for* every *$k \geq 3$, the k-SAT problem is in $O(2^{\omega n/3 + o(n)}) \leq O(1.74^n)$ time, where n is the number of variables.*

All known algorithms for k-SAT run in $O(2^{c_k n})$ time for some sequence $c_k \to 1$. Assuming k-SAT cannot be solved in $2^{o(n)}$ time, the sequence $\{c_k\}$ is non-decreasing [23].

Proof of Theorem 4.1. Let F be an instance of k-SAT with n variables and m clauses. By the Sparsification Lemma of [24], for every $\varepsilon > 0$ we can reduce any k-CNF F to a $2^{\varepsilon n}$ collection of k-CNF formulas $\mathcal{F}$, where each formula in $\mathcal{F}$ has $c_{k,\varepsilon} n$ clauses, for a constant $c_{k,\varepsilon}$ depending on k and ε. Thus we may assume without loss of generality that $m \leq cn$ for some constant. We also suppose that n is divisible by 3.

Conceptually split the set of variables of F into three equal parts V_1, V_2, V_3. Let a_i be an assignment on the variables from part V_i. Define $S_{a_i} \subseteq [m]$, where $j \in S_{a_i}$ if and only if the jth clause is not satisfied after plugging a_i into the jth clause. (That is, $j \in S_{a_i}$ iff a_i does not assign *true* to any variable in the jth clause.) For partial assignments a_1, a_2, a_3 from parts V_1, V_2, V_3 (respectively), it is easy to see that the assignment a_1, a_2, a_3 satisfies F if and only if $S_{a_1} \cap S_{a_2} \cap S_{a_3} = \varnothing$.

The SAT algorithm first cycles over every possible sequence Q of pairs $(b_1, i_1), (b_2, i_2), \ldots, (b_k, i_k)$, where $b_j \in \{0,1\}^*$, $\sum_j |b_j| \leq o(m)$, and $i_j \in \{1,2,3\}$. These sequences represent all possible $o(m)$-bit communications that could take place in a (nondeterministic) communication protocol for 3-party disjointness. Note the number of such sequences is $2^{o(m)}$.

Given F and communication sequence Q, we construct an $O(2^{n/3})$ node graph G that contains a triangle if and only if there is a variable assignment a_1, a_2, a_3

such that Q represents a communication sequence between the parties in which they *accept*, concluding that $S_{a_1} \cap S_{a_2} \cap S_{a_3} = \varnothing$.

Let $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$ be three disjoint sets of vertices on $O(2^{n/3})$ nodes, where each node v_i of some $\mathcal{V}_i$ is indexed by an $n/3$-bit string $a(v_i)$. For $i = 1, 2, 3$, relabel each node v_i in $\mathcal{V}_i$ with the set $S_{a(v)}$.

Put an edge $\{v_2, v_3\} \in \mathcal{V}_2 \times \mathcal{V}_3$ iff it is consistent for the first party to *accept* and speak according to communication sequence Q, while holding the sets $S_{a(v_2)}$ and $S_{a(v_3)}$, and viewing the communications (stated in Q) from the other two parties. Similarly put an edge $\{v_1, v_3\} \in \mathcal{V}_1 \times \mathcal{V}_3$ iff Q is consistent with the second party, and an edge $\{v_1, v_2\} \in \mathcal{V}_1 \times \mathcal{V}_2$ iff Q is consistent with the third party. Note the construction of G can be done in polynomial time. Then, a triangle v_1, v_2, v_3 in the graph corresponds to a situation where the parties determine that $S_{a_1} \cap S_{a_2} \cap S_{a_3} = \varnothing$ and each party holds two of the three sets.

Finally, we can determine whether or not G is triangle-free in $O(2^{\omega n/3})$ time, by taking the cube of the adjacency matrix of G, and determining if the trace of the resulting matrix is nonzero. (In general, one can test if an N-node graph is triangle-free or not, in $O(N^\omega)$ time.) □

The above proof also implies:

Corollary 4.1. *If 3-party disjointness has a nondeterministic protocol with communication complexity $o(m)$ such that all parties run in deterministic $2^{o(m)}$ time, then CNF-SAT can be solved in $O^\star(2^{\omega n/3 + o(m)})$ time, where n is the number of variables and m is the number of clauses in the instance.*

Therefore, the hypothesis implies that CNF-SAT with cn clauses and n variables can always be solved in $O(1.74^n)$ time for all constants c.

In their work on algebrization, Aaronson and Wigderson [1] gave a Merlin-Arthur protocol for two-party set disjointness that uses only $\tilde{O}(\sqrt{m})$ bits of communication, which can be generalized to three parties. Is it possible to adapt their protocol to get a better satisfiability algorithm? This does not seem possible, and the main problem is the use of randomness. In order for their protocol to work in our setting, the number of random bits must be $\Omega(n)$, as we need to rule out false positives over all possible variable assignments.

5 SAT and d-SUM

The d-SUM problem asks whether a set of N numbers contains a d tuple that sums to zero. It is conjectured that this problem requires $\Omega(n^{\lceil d/2 \rceil}/\text{polylog}n)$ time, and this conjecture is instrumental in understanding the complexity of many problems in computational geometry (via reductions).

The d-SUM problem hardly needs an introduction. The seminal work of Gajentaan and Overmars [19] introduced 3SUM for the purpose of arguing that problems in planar geometry "should" take $\Omega(n^2)$ time. Showing 3SUM-hardness for a problem is considered routine today.

In d dimensions, the best algorithms for many natural problems run in only $n^{O(d)}$ time, a phenomenon labelled the "curse of dimensionality" because the problems quickly become intractable for high dimensions. The presumed hardness of the d-SUM problem is used to argue that these running times are likely optimal. Here, we show a relationship between the difficulty of d-SUM and the difficulty of k-SAT:

Theorem 5.1. *Let $d < N^{0.99}$ and $\delta < 1$. If d-SUM over N numbers of $O(d \lg N)$ bits can be solved in $O(N^{\delta d})$ time, then for every $k \geq 3$, the k-SAT problem over n variables can be solved in $2^{\delta^{1/O(k)} \cdot O(kn)}$ time.*

In particular, our result implies:

Corollary 5.1. *Let $d < N^{0.99}$. If d-SUM over N numbers of $O(d \lg N)$ bits can be solved in $N^{o(d)}$ time, then 3-SAT on n variables can be solved in $2^{o(n)}$ time.*

The unparameterized version of d-SUM is, of course, the Subset-Sum problem, which is well-known to be NP-complete. For the parameterized version, the standard hardness reduction (see, for example, the book of Downey and Fellows [16]) embeds an instance of the k-clique problem into an instance of $O(k^2)$-SUM. While this is enough to rule out a fixed parameter solution with running time $f(k) \cdot poly(n)$, the "lower bound" is suboptimal: as k-clique can be solved in $O(n^k)$ time, this reduction at most implies an $n^{\Omega(\sqrt{d})}$ hardness for d-SUM.

Proof of Theorem 5.1. Let F be an instance of k-SAT with n variables and m clauses. By the improved Sparsification Lemma of Calabro, Impagliazzo, and Paturi, [4], for any $\varepsilon > 0$, we can reduce F to a collection of $2^{\varepsilon n}$ k-CNF formulas $\mathcal{F}$, all of which have n variables and $n \cdot (k/\varepsilon)^{O(k)}$ clauses. With hindsight, choose $\varepsilon = k \cdot \delta^{1/(\gamma k)}$ for an appropriate constant γ. Then, the number of clauses of each formula will be $m' = n/\sqrt{\delta}$.

Now convert each formula to a 3-SAT formula with $O(m'k)$ variables and clauses. Secondly, convert each 3-SAT formula to a 1-in-3 SAT formula, by the classic reduction of Schaefer (see [20]). For every clause $(x \vee y \vee z)$, this reduction introduces 6 new variables a, b, c, d, e, f and clauses $R(x, a, d) \wedge R(y, b, d) \wedge$

$R(a,b,e) \wedge R(c,d,f) \wedge R(z,c,False)$, where R denotes the 1-in-3 relation. The final formula also has $O(m'k)$ variables and clauses.

Now we reduce 1-in-3 SAT to the d-SUM problem. Conceptually split the variables of each 1-in-3 SAT formula into d blocks of equal size. In each block, try all $2^{O(m'k/d)}$ assignments to the variables. For each assignment, we generate another number in the list. Our numbers are generated as strings of digits in base $d+1$. The first d digits represent the block of variables to which the number belongs: a number from block k will have a zero in all positions except the k-th, where it has a one. The next digits, one per clause, mark the number of variables in this assignment that are satisfying each particular clause.

Let X be the number with all digits equal to one. We can ask for a sum of d values equal to the number X. (This is a mild generalization of d-SUM, which can be reduced to the original problem by adding $-X/d$ to all numbers.) To obtain X as a sum of d values, it must be that one value comes from each block. Furthermore, it must be that each clause is satisfied exactly once. Thus, a d-sum exists if and only if the 1-in-3 SAT formula is satisfiable.

The size of our d-SUM problem is $N = d \cdot 2^{O(m'k/d)}$. If the problem can be solved in $O(N^{\delta d})$ time, a 1-in-3 SAT problem can be solved in $2^{O(\delta m'k)}$ time. This is time $2^{O(\delta nk/\sqrt{\delta})}) = 2^{O(kn\sqrt{\delta})}$. We must multiply this by the number of sparse k-CNF formulas that we have to solve, which is $2^{\varepsilon n} = 2^{kn \cdot \delta^{1/O(k)}}$. Thus, the overall time to solve an arbitrary k-CNF is $2^{O(kn) \cdot \delta^{1/O(k)}}$. □

6 Conclusion

We have demonstrated new connections between algorithms for satisfiability and other diverse problems from different research threads. The fact that there are some relationships is probably not too surprising, given the widespread phenomenon of NP-completeness in general. The surprising property of our reductions are their striking tightness: seemingly minor improvements on any of several well-studied problems would imply faster satisfiability algorithms.

To complement these results, it would be interesting if one could find good evidence for the *impossibility* of an improved CNF-SAT algorithm, beyond our mere intuition.

Acknowledgements

We thank Edward Hirsch for pointing us to the reference [14], and the SODA program committee for their helpful comments. R.W. thanks the organizers of Dagstuhl Seminar 05301 on Exact Algorithms and Fixed-Parameter Tractability for their invitation which initiated this work, and his thesis committee for their feedback on some early results.

References

[1] Scott Aaronson and Avi Wigderson. Algebrization: A New Barrier in Complexity Theory. *Proc. of ACM STOC*, 731–740, 2008.

[2] Vikraman Arvind and Rainer Schuler. The quantum query complexity of 0-1 knapsack and associated claw problems. In *Proc. of International Symposium on Algorithms and Computation* (ISAAC), Springer-Verlag LNCS:168–177, 2003.

[3] Bengt Aspvall, Michael F. Plass, and Robert E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Proc. Letters* 8(3):121–123, 1979.

[4] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A Duality between Clause Width and Clause Density for SAT. In *Proc. IEEE Conference on Computational Complexity*, 252–260, 2006.

[5] Arkadev Chattopadhyay and Anil Ada. Multiparty Communication Complexity of Disjointness. *ECCC Report TR08-002*, 2008.

[6] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Computer and System Sciences* 72:1346–1367, 2006.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms, 2nd Edition*, MIT Press, 2001.

[8] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation* 9:251–280, 1990.

[9] Don Coppersmith. Rectangular Matrix Multiplication Revisited. *J. Complexity* 13:42–49.

[10] Evgeny Dantsin, Edward A. Hirsch, and Alexander Wolpert. Algorithms for SAT based on search in Hamming balls. In *Proc. Symposium on Theor. Aspects of Comput. Sci.* (STACS), Springer-Verlag LNCS 2996, 141–151, 2004.

[11] Evgeny Dantsin and Alexander Wolpert. Derandomization of Schuler's Algorithm for SAT. In *Proc. Int. Conf. on Theory and Applications of Satisfiability Testing* (SAT), Springer-Verlag LNCS 3542, 80–88, 2005.

[12] Evgeny Dantsin and Alexander Wolpert. An improved upper bound for SAT. In *Proc. Int. Conf. on Theory and Applications on Satisfiability Testing* (SAT), Springer-Verlag LNCS 3569, 400–407, 2005.

[13] Evgeny Dantsin, Edward A. Hirsch, and Alexander Wolpert. Clause Shortening Combined with Pruning Yields a New Upper Bound for Deterministic SAT Algorithms. *Electronic Colloquium on Computational Complexity*, Report 102, 2005.

[14] Evgeny Dantsin and Edward A. Hirsch. Worst-Case Upper Bounds. In *Handbook of Satisfiability*, Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsch (eds.), 341–362, 2008.

[15] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Programming* 1(3):267–284, 1984.

[16] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.

[17] Friedrich Eisenbrand and Fabrizio Grandoni. On the Complexity of Fixed Parameter Clique and Dominating Set. *Theor. Comput. Sci.* 326(1-3):57–67, 2004.

[18] Uriel Feige and Joe Kilian. On Limited versus Polynomial Nondeterminism. *Chicago J. Theor. Comput. Sci.*, 1997.

[19] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications* 5(3): 165–185, 1995.

[20] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[21] Martin Grohe. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. In *Proc. IEEE Symposium on Foundations of Computer Science* (FOCS), 552–561, 2003.

[22] Vince Grolmusz. The BNS lower bound for multi-party protocols is nearly optimal. *Information and computation* 112(1):51–54, 1994.

[23] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62(2):367–375, 2001.

[24] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63(4): 512–530, 2001.

[25] Troy Lee and Adi Shraibman. Disjointness is hard in the multi-party number-on-the-forehead model. In *Proc. IEEE Conference on Computational Complexity*, 309–336, 2009.

[26] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. 2+p-SAT: Relation of Typical-Case Complexity to the Nature of the Phase Transition. *Random Structures and Algorithms* 15:414–440, 1999.

[27] Jaroslav Nesetril and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2): 415–419, 1985.

[28] Pavel Pudlak. Satisfiability – algorithms and logic. In *Proc. Int. Symp. on Mathematical Foundations of Computer Science* (MFCS'98), Springer-Verlag LNCS 1450, 129–141, 1998.

[29] International Conference on Theory and Applications of Satisfiability Testing. See `http://www.satisfiability.org/`

[30] Rainer Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms* 54(1):40–44, 2005.

[31] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization - Eureka! You shrink!*, Springer-Verlag LNCS 2570, 185–207, 2003.

[32] Gerhard J. Woeginger. Space and time complexity of exact algorithms: some open problems. In *Proc. Int. Workshop on Parameterized and Exact Computation* (IWPEC), Springer-Verlag LNCS 3162, 281–290, 2004.

[33] Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proc. International Joint Conference on Artificial Intelligence* (IJCAI), 2003.

[34] R. Ryan Williams. Algorithms and resource requirements for fundamental problems. PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, 2007.

A Horn SAT and CNF-SAT

Similar to 2-SAT, the HornSat problem is another restriction of CNF-SAT that is known to be solvable in linear time [15]. An instance of HornSat is a CNF formula with at most one non-negative literal per clause. HornSat is considered to be a more powerful restriction than 2-SAT, since HornSat is P-complete and 2-SAT is NL-complete. Somewhat analogous to the previous section, we show that better algorithms for an extension of HornSat imply better algorithms for CNF-SAT. Owing to the power of HornSat, the result we prove here will be more general than our result for 2-SAT.

Let $k \geq 2$. We define HornSat+kClauses to be a Horn CNF formula conjoined with k additional clauses of arbitrary size having only positive literals. Clearly, an instance of this problem can be solved in $O(n^k \cdot (m+n))$ time, where n is the number of variables and m is the number of clauses.

Theorem A.1. *If there is a k and $\varepsilon > 0$ such that* HornSat+kClauses *is in $O((n+m)^{k-\varepsilon})$ time, then CNF-SAT has an improved algorithm.*

We must admit that we are less certain of a better algorithm for this problem, since the "gap" between the runtime of the best algorithm we know and the time bound we would like is larger. The proof is similar to the one for 2Sat+2Clauses, but with a few modifications.

Proof. (Sketch) Let F be a CNF of n variables and m clauses. Split the set of n variables into k equal-sized parts, and form all possible $2^{n/k}$ assignments for the variables in each part. For each of the $k2^{n/k}$ partial assignments A to some n/k variables, we make an "assignment variable" x_A in the new formula. For each part $i = 1, \ldots, k$, we make an "assignment clause" in the new formula, which is just the disjunction of all

$2^{n/k}$ assignment variables from part i. These are the k clauses of arbitrary length.

Now we describe the rest of the formula, which is Horn. For each literal ℓ in F, we make a "literal variable" x_ℓ in the new formula, and add clauses

$$(\neg x_A \vee x_\ell)$$

for each literal ℓ that is directly implied by a partial assignment A. Note an assignment variable x_A implies exactly n/k literal variables. We also add n clauses of the form

$$(\neg x_\ell \vee \neg x_{\neg \ell})$$

for all pairs of literal variables, forbidding two opposing literal variables to both be true. (For this reason, at most one assignment variable from an assignment clause can possibly be true.)

For each clause C from F, define C_i to be the restriction of C to just those variables from part i. Each clause C from F will have $2k$ "clause variables" in the new formula. In particular, for each part, there are two clause variables x_{C_i} and $x_{\neg C_i}$, representing C_i. A clause C in F is represented by the Horn clause

$$(\neg x_{\neg C_1} \vee \cdots \vee \neg x_{\neg C_{k-1}} \vee x_{C_k}).$$

Suppose $C_i = (\ell_1 \vee \cdots \vee \ell_j)$. Then we make the Horn clauses

$$(\neg x_{\neg \ell_1} \vee \cdots \vee \neg x_{\neg \ell_j} \vee x_{\neg C_i}), (\neg x_{\ell_1} \vee x_{C_i}), \ldots, (\neg x_{\ell_j} \vee x_{C_i}).$$

We add clauses forbidding both x_{C_i} and $x_{\neg C_i}$, for all clause variables C_i:

$$(\neg x_{C_i} \vee \neg x_{\neg C_i}).$$

Finally, we claim that this new formula is satisfiable iff F is satisfiable. Namely, the chosen partial assignment variables from each of the k assignment clauses correspond to a satisfying assignment for F. □