The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial

Leonard Pitt*
University of Illinois
at Urbana-Champaign

Manfred K. Warmuth**
University of California
at Santa Cruz

Abstract

The minimum consistent DFA problem is that of finding a DFA with as few states as possible that is consistent with a given sample (a finite collection of words, each labeled as to whether the DFA found should accept or reject). Assuming that P $\neq$ NP, it is shown that for any constant k, no polynomial time algorithm can be guaranteed to find a consistent DFA of size opt^k, where opt is the size of a smallest DFA consistent with the sample. This result holds even if the alphabet is of constant size two, and if the algorithm is allowed to produce an NFA, a regular grammar, or a regular expression that is consistent with the sample. Similar hardness results are described for the problem of finding small consistent linear grammars.

1 Introduction

We consider the following problem. Given finite sets POS and NEG of words over a finite alphabet, can a small deterministic finite automaton (DFA) be constructed that is consistent with POS and NEG, i.e., accepts all words of POS and rejects all words of NEG? It is known that the problem of determining the smallest such consistent DFA for a given sample is NP-hard [8], and thus is unlikely to be solvable with a polynomial time algorithm [7]. It is natural to ask whether an *approximately small* DFA can be found: Is there an efficient algorithm, that for some reasonably slowly growing function f, can produce (or just determine the existence of) a consistent DFA of size $f(opt)$, where opt is the size of the smallest consistent DFA?

Our Main Theorem (Theorem 3.1) answers this question negatively by showing that assuming P$\neq$NP, there does not exist a polynomial time algorithm A and constant k such that on input of any finite sets of strings POS and NEG A outputs a *nondeterministic* finite automaton (NFA) that is consistent with POS and NEG, and has less than opt^k states, where opt is the minimum number of states of any consistent DFA. This improves the lower bound on approximability due to Li and Vazirani [14], which shows that a constant factor of $\frac{9}{8}$ cannot be achieved.

In the complete paper [17] we show that the Main Theorem still holds if POS and NEG are sets of strings over a two letter alphabet (The previous $\frac{9}{8}$ factor result of [14] holds for the two letter case). Also, as an extension of the Main Theorem, it is shown that unless P=NP, no element of any of the naturally used representations of the regular sets (DFAs, NFAs, regular expressions, or regular grammars) can always be found that is of size at most polynomially larger than the smallest DFA consistent with a sample over a two letter alphabet. In [17] we also show that the techniques introduced here can be used to show that the linear grammar consistency problem cannot be approximated within any polynomial factor unless P=NP. More specifically, given two finite sets POS and NEG consistent with some linear grammar G, it is NP-hard to find a linear grammar G' that generates all of the strings of POS, none of the strings of NEG, and has size bounded by some polynomial in the size of G. The extensions of the Main Theorem and the results on linear grammars are summarized in Section 5.

Other nonapproximability results

There seem to be few naturally arising optimization problems for which nonapproximability results have been shown. Indeed, the dearth of such results is one of the motivations given by Papadimitriou and Yannakakis [15] for their investigation of approximation pre-

*Supported in part by NSF grant IRI-8809570, and by the Department of Computer Science, University of Illinois at Urbana-Champaign.

**Supported by ONR grants N00014-86-K-0454 and N00014-85-K-0445. Currently on leave from UCSC at Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138.

serving reductions. The traveling salesperson problem (TSP) is perhaps the most notable optimization problem that cannot be approximated (in the absence of other constraints, e.g., triangle inequality) [7] assuming P≠NP. However, the reason that TSP is not approximable is that it is essentially the weighted version of the NP-complete Hamiltonian cycle problem. Although one may similarly define *optimization* problems based on other NP-complete *decision* problems in such a way that the optimization problem cannot be approximated at all (or at least not very well), such results are typically uninteresting for two reasons — the problems defined usually are not natural, and the resulting proofs are trivial. In contrast, the minimum consistent DFA problem discussed here is a natural problem, and the nonapproximability result is not obtained by simply adding weights to an NP-complete decision problem.

Besides TSP, among the seemingly few existing negative approximability results, two others are well known, but the bounds are much weaker than that shown for TSP and the result given here for DFAs. For minimum graph coloring [6], it was shown that (unless P=NP) no polynomial time approximation algorithm exists guaranteeing a constant factor approximation strictly smaller than twice optimal. Also, for maximum independent set (equivalently, maximum clique), it has been shown that if *some* constant factor approximation can be achieved, then *any* constant factor approximation can be achieved [7].

The minimum consistent DFA problem

Gold [8] proved that the problem of finding a smallest consistent DFA is NP-hard. Angluin [4] showed that it is NP-hard to determine whether there exists a two state DFA consistent with given data. Trakhtenbrot and Barzdin [19] gave a polynomial time algorithm for finding a smallest consistent DFA in the case where the sets POS and NEG together consist of all strings up to a given length. Angluin [3] extended Gold's result, and showed that if even some small fraction ϵ of strings up to a given length were missing from POS ∪ NEG, then the problem is again NP-hard, and also showed that the problem of finding the smallest regular expression consistent with a finite sample is NP-hard. In [1] it was left as an open question whether an approximately small DFA could be found.

In 1987, Li and Vazirani [14] gave the first nonapproximability result for the minimum consistent DFA problem, showing that if P≠NP, no polynomial time algorithm can find a consistent NFA of size smaller than $\frac{9}{8}$ times the size of a smallest consistent DFA. In this paper we replace the constant factor $\frac{9}{8}$ with any polynomial function of optimal.

Finally, concurrent with this research, Kearns and Valiant give even stronger nonapproximability results for the minimum consistent DFA problem than the one presented here [13]. However, their results rely on cryptographic assumptions (e.g., that factoring Blum integers is intractable), whereas our results assume only that P≠NP. We present a discussion of their work, and its relationship to ours, in Section 6.

Implications for learning DFAs

Although our results have main significance in the context of combinatorial optimization, our original motivation was in the study of the learnability of DFAs from randomly generated examples in the distribution independent model of learning (now called *pac-learning*) introduced by Valiant [20]. By results from [5], if in fact there *was* a polynomial time algorithm that could, given two finite sets POS and NEG, produce a consistent DFA of size at most polynomially larger than the smallest consistent DFA, then DFAs would be pac-learnable. This is only a sufficient condition for learnability; consequently our results *do not* show that DFAs are not pac-learnable[1]. However, our results *do* show that any efficient algorithm for learning DFAs would have to produce very large hypotheses (unless P=NP). Further discussion of the learnability of DFAs, the problem of finding a small consistent DFA, and the relationship between our work and recent work of Kearns and Valiant [13] (who show nonlearnability of DFAs based on cryptographic assumptions) is given in Section 6.

2 Definitions

2.1 DFAs and NFAs

For a finite alphabet Σ, the set Σ^* consists of all *words* (or *strings*) of finite length formed from the symbols of Σ. If $w \in \Sigma^*$ then $|w|$ denotes the number of symbols of w, and is called the *length* of w. The empty word λ is the unique word with length 0 in Σ^*.

A *deterministic finite automaton* (DFA) A is a 5-tuple $(Q, \Sigma, \delta, s_{init}, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $s_{init} \in Q$ is the *initial state*, δ is the *transition function* which maps $Q \times \Sigma$ to Q, and $F \subseteq Q$ is the set of *accepting* or *final* states. A *nondeterministic finite automaton* is a 5-tuple with the same parameters except that the transition function maps $Q \times (\Sigma \cup \{\lambda\})$ to 2^Q, the power set of Q.

We will use the standard graph representation of an NFA in which the vertices are the states of the NFA, and in which there is a directed edge (or *transition*) labeled

[1] Angluin [2] has recently shown that DFAs are not learnable if the learner may only ask *equivalence queries* instead of receiving randomly generated examples. This result has no bearing on the optimization problem considered in this paper.

with $\sigma \in \Sigma \cup \{\lambda\}$ from state s to state t if $t \in \delta(s, \sigma)$. Note that some edges may be labeled with λ. DFAs may be viewed as NFAs with the additional restriction that there are no λ-transitions, and for each state $s \in Q$ and letter $a \in \Sigma$, there is exactly one edge leaving s which is labeled with a.

A string $w \in \Sigma^*$ is *accepted* by the NFA (DFA) A iff there is a directed path leading from the initial state to some accepting state such that the concatenation of the symbols of the edges of the path forms the string w. (We say that the path is "labeled with" w.)

For any states s, t in Q, and any string w, we say w *leads* from s to t if there is a path labeled with w from s to t (In the case of a DFA, such a path is always unique). We also write w *leads to* t iff w leads from s_{init} to t.

A *positive example* of A is a word accepted by A and a *negative example* is a word in Σ^* that is not accepted by A. The *language accepted by* A, denoted by $L(A)$, is the set of all words accepted by A. The class of languages accepted by DFAs is identical to the class of languages accepted by NFAs and is called the class of *regular languages* [12].

2.2 The Consistency Problem

A set of representations (encodings) of a class of languages $\mathcal{L}$ is a set $\mathcal{A}$ such that each $A \in \mathcal{A}$ denotes a language $L(A) \in \mathcal{L}$, and for each language $L \in \mathcal{L}$ there is at least one element of $\mathcal{A}$ that denotes L. Let $L(\mathcal{A}) = \{L(A) : A \in \mathcal{A}\}$ (thus if $\mathcal{A}$ is a set of representations for $\mathcal{L}$ then $L(\mathcal{A}) = \mathcal{L}$). For example, the set of deterministic finite automata (DFAs) is a set of representations for the regular languages, as are NFAs, regular grammars, and regular expressions. We associate a size measure with each set of representations. The size (a nonnegative integer) of any element $A \in \mathcal{A}$ is denoted by $|A|$. For the case of DFAs or NFAs, $|A|$ is defined as the number of states of the automaton.

Definition 2.1 *A representation A is* consistent *with two sets of finite strings* POS *and* NEG *if* POS *is contained in $L(A)$ and* NEG *is disjoint from $L(A)$.*

Definition 2.2 *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of representations of languages and let $L(\mathcal{A}) \subseteq L(\mathcal{B})$. The minimization problem* MIN-CON$(\mathcal{A}, \mathcal{B})$ *is defined as follows.*

Input instance: *An instance I of* MIN-CON$(\mathcal{A}, \mathcal{B})$ *consists of two finite sets of strings,* POS *and* NEG, *consistent with some element $A \in \mathcal{A}$.*

Feasible solution: *Any element $B \in \mathcal{B}$ that is consistent with I is a feasible solution. Note that there always exists a feasible solution.*

Cost: *The cost of a feasible solution B is the size $|B|$ of the representation B.*

Optimal solution: *For any instance I, the value $opt(I)$ is defined as the size of the smallest element of $\mathcal{A}$ that is consistent with I.*

Note that a feasible solution of the problem requires a representation from the class $\mathcal{B}$, and optimality is defined with respect to elements of the class $\mathcal{A}$. Thus for some choices of $\mathcal{A}$ and $\mathcal{B}$, there may be no feasible solutions with cost as small as an optimal solution. (For example, consider MIN-CON(NFA,DFA)). Although the general definition assumes nothing regarding the relationship between the size of the smallest consistent members of $\mathcal{A}$ and $\mathcal{B}$, in this paper the latter is usually equal to or smaller than the former.

MIN-CON(DFA,NFA) is the main optimization problem we consider. This problem is easier than MIN-CON(DFA,DFA), since (1) every DFA is an NFA, and (2) in some cases, the smallest consistent NFA for a language is significantly smaller than the smallest consistent DFA.

Definition 2.3 *Let $\mathcal{A}, \mathcal{B}$ be sets of language representations, and f be any function of the single variable opt. Then* MIN-CON$(\mathcal{A}, \mathcal{B})$ *is $f(opt)$-approximable iff there exists a constant c and a polynomial time algorithm* APPROX *such that on input of any instance I of* MIN-CON$(\mathcal{A}, \mathcal{B})$ *for which $opt(I) \geq c$,* APPROX *outputs a representation $B \in \mathcal{B}$ that is consistent with I and such that $|B| < f(opt(I))$.*

Note that the definition of $f(opt)$-approximable does not require the approximation algorithm to perform well on all instances, but only on those instances I with sufficiently large values of $opt(I)$. Consequently, a result showing nonapproximability must show, for all approximation algorithms, that for arbitrarily large values of opt there are instances I with $opt(I) = opt$, and for which the approximation algorithm fails to achieve the desired bound. Thus this is a stronger negative result than simply showing that the bound $f(opt)$ is not obtainable for particular values of opt.

The definition of MIN-CON$(\mathcal{A}, \mathcal{B})$ depends on the (implicit) size measures used for $\mathcal{A}$ and $\mathcal{B}$. However the results of this paper are in the following form: MIN-CON$(\mathcal{A}, \mathcal{B})$ is not $f(opt)$-approximable for any function f that is polynomially bounded. Such results are robust with respect to any size measure that is polynomially related to ours.

2.3 Using gaps to force nonapproximability

Our goal will be to show that, assuming P≠NP, MIN-CON(DFA,NFA) is not $f(opt)$-approximable for any function f bounded above by some polynomial. Non-approximability results may be obtained by exhibiting "gaps" in the cost measure for a minimization problem. Intuitively, if we can transform an instance of an NP-hard decision problem into a MIN-CON problem, such that if the answer to the NP-hard decision problem is "yes" then the optimal solution to the MIN-CON problem is some number p, whereas if the answer is "no", then there is no solution to the MIN-CON problem of size smaller than $f(p)$, then we can show that $f(opt)$-approximability of the MIN-CON problem implies that P=NP. More formally, we have the following sufficient condition.

Lemma 2.4 *Suppose there are infinitely many positive integers p such that there exists a polynomial time transformation R_p with the following properties:*

Property 1 *R_p takes as input some instance I of an* NP*-complete language S, and outputs an instance of* MIN-CON$(\mathcal{A}, \mathcal{B})$.

Property 2 *If instance $I \in S$ then $R_p(I)$ has an optimal solution with cost $opt(I) = p$.*

Property 3 *If $I \notin S$, then there does not exist a solution for $R_p(I)$ with cost less than $f(p)$.*

Then, under the assumption that P≠NP, MIN-CON$(\mathcal{A}, \mathcal{B})$ *is not $f(opt)$-approximable.*

2.4 1-in-3-SAT

The NP-hard problem we use in our reductions is a variant of 3-SAT, the "monotone 1-in-3-SAT problem" [7, 18]. An instance I of monotone 1-in-3-SAT consists of a set of variables $V = \{v_1, v_2, \ldots, v_n\}$ and a nonempty collection of clauses $\{c_i\}_{1 \le i \le s}$, each of size 3. (I.e., each c_i is a 3-element subset of V). For brevity, we henceforth omit the word "monotone", and refer to the problem as "1-in-3-SAT". $|I|$ denotes the *size* of the instance I according to some fixed encoding scheme. In particular, $|I|$ is always at least as large as the number of variables plus the number of clauses of instance I, and is not more than polynomially larger. A *(truth) assignment* is a function $\tau : V \to \{0, 1\}$. An assignment τ *is a solution to* I if for every clause (v_x, v_y, v_z) of I, the multiset $\{\tau(v_x), \tau(v_y), \tau(v_z)\} = \{0, 0, 1\}$. If we write that v_x is assigned true (respectively false) by τ, we mean $\tau(v_x) = 1$ (respectively, $\tau(v_x) = 0$). The decision problem for 1-in-3-SAT is to determine for any input instance I, whether or not there exists a solution to I.

2.5 Counter DFAs

The smallest consistent DFA for the examples of the reductions presented will be of a very special form which is defined as follows. A *Counter* DFA (CDFA) over alphabet V is a deterministic finite automaton that counts the number of occurrences of characters in a subset V' of V mod p for some number p as follows: The labeled graph representing the CDFA consists of a simple cycle of p states, with the labeled edges of V' advancing one state around the cycle, and the labeled edges of $V - V'$ returning to the same state. Thus each character of V' read increases the "count" by 1 mod p, and a character of $V - V'$ leaves the count unchanged. Further, the start state is the same as the unique final state, thus CDFAs count from 0 to $p - 1$ mod p. CDFAs are a restricted subclass of DFAs that are contained in a class which is pac-learnable. Further discussion appears in Section 6.

Definition 2.5 *Let $\tau : V \to \{0, 1\}$ be a truth assignment to the variable set V. Then $C(p, \tau)$ is the* CDFA *that counts all true variables (i.e., counts the set $\tau^{-1}(1)$) mod p as described above.*

3 Outline of the Proof of the Main Theorem

Theorem 3.1 (Main Theorem)
For all positive integers k, MIN-CON(DFA,NFA) *is not opt^k-approximable unless* P=NP.

Proof: We provide a reduction from the 1-in-3-SAT problem to the MIN-CON(DFA,NFA) problem that introduces a gap between p and p^k. Let k be any constant, and let $m = 3 \cdot 2^{k-1}$. We show that for all sufficiently large primes p (in particular, for $p > 2^{k-1+m}$), there exists a reduction $R_{p,k}$ that is computable in polynomial time (Proposition 4.3) that takes as input an instance I of a 1-in-3-SAT problem, and produces two sets POS(p, k, I) and NEG(p, k, I) which satisfy Lemmas 3.2, 3.3, and 3.4 (stated immediately below). It follows that the hypothesis of Lemma 2.4 is satisfied with $f(opt) = opt^k$, and thus unless P=NP, MIN-CON(DFA,NFA) is not opt^k-approximable. □

The description of $R_{p,k}$, and of the sets POS(p, k, I) and NEG(p, k, I), are given in Section 4.2. The key lemmas used show that there is a p state counter machine $C(p, \tau)$ (Lemma 3.2), which is in fact optimal (Lemma 3.3), and that any NFA with fewer than p^k

states provides a solution to a 1-in-3-SAT problem (Lemma 3.4).

Lemma 3.2 *Let I be an instance of* 1-in-3-SAT. *If τ is a solution of I, then for all positive integers k and p, $C(p,\tau)$ is consistent with* $\mathrm{POS}(p,k,I)$ *and* $\mathrm{NEG}(p,k,I)$. *Thus if I has some solution, then there exists a consistent p state* DFA.

Lemma 3.3 *Let k and p be any positive integers, and let I be any instance of* 1-in-3-SAT. *Then any* NFA *that is consistent with* $\mathrm{POS}(p,k,I)$ *and* $\mathrm{NEG}(p,k,I)$ *has at least p states.*

Lemma 3.4 *Let k be any positive integer, and let p be a prime such that $p > 2^{k-1+m}$. If I is any instance of* 1-in-3-SAT, *and if $A = (Q, \Sigma, \delta, s_{init}, F)$ is an* NFA *such that $|Q| < p^k$ and such that A is consistent with* $\mathrm{POS}(p,k,I)$ *and* $\mathrm{NEG}(p,k,I)$, *then I has some solution.*

Lemmas 3.2 and 3.3 are proved in Section 4.3, and Lemma 3.4 is proved in Section 4.4. Before delving into the details of the proofs, we give a general overview of the techniques involved. Given an assignment τ, a string is accepted by the counter machine $C(p,\tau)$ iff the number of true variables in the string is congruent to 0 mod p. In fact, if a string γ leads from any state in $C(p,\tau)$ back to that state, then the number of true variables in γ is congruent to 0.

We may rewrite the above property of $C(p,\tau)$ as follows. For any string γ, let $\vec{\gamma} = \langle x_1, x_2, \ldots x_n \rangle$, such that for each i, $1 \leq i \leq n$, the number of occurrences of v_i in γ is congruent to x_i mod p. Then γ leads around a cycle in $C(p,\tau)$ iff $\vec{\gamma} \cdot \tau = 0$, where the assignment τ is interpreted as a Boolean vector of length n, "$\cdot$" is the dot product, and all arithmetic is mod p. Consequently, given a collection of strings $\{\gamma_i\}$ that lead around a cycle in some unknown counter machine $C(p,\tau)$, one way of determining τ would be to solve the simultaneous system of equations $\{\vec{\gamma_i} \cdot \vec{x} = 0\}$. This suggests a strategy for constructing $\mathrm{POS}(p,k,I)$, $\mathrm{NEG}(p,k,I)$: try to force the above property in *any* small consistent NFA, not simply a counter DFA. Thus we construct $\mathrm{POS}(p,k,I)$ and $\mathrm{NEG}(p,k,I)$ such that

- If τ is any solution for I, then $C(p,\tau)$ (which has p states) is consistent with $\mathrm{POS}(p,k,I)$ and $\mathrm{NEG}(p,k,I)$;
- A single carefully constructed positive example forces a cycle in any accepting NFA with *strictly less than p^k* states;
- From the cycle, a set of strings $\{\gamma_i\}$ may be extracted;
- If S is the matrix with rows $\{\vec{\gamma_i}\}$ representing the equations $\{\vec{\gamma_i} \cdot \vec{x} = 0\}$, then the set of solutions to the system S contains an element that is $\{0,1\}$ valued, and is a solution of I. Thus the existence of a consistent NFA with less than p^k states implies that I has some solution.

The last property is achieved by including in $\mathrm{NEG}(p,k,I)$ examples that rule out consistent automata with less than p^k states whose induced set of equations (those extracted from strings leading around the cycle) do not include a solution of I.

4 Technical Details

4.1 Definitions

Let k be any constant. Let $m = 3 \cdot 2^{k-1}$. Let p be a prime number such that $p > 2^{k-1+m}$. Let $P = \{0, 1, \ldots, p-1\}$. P^n denotes all vectors of length n with elements in P. Vectors $\vec{x}, \vec{y}, \vec{z}$ will always denote elements of P^n. All vectors are indexed starting at 1, thus $\vec{x} = \langle x_1, x_2, \ldots x_n \rangle$. We assume the standard lexicographic ordering on P^n. Thus $\vec{x} < \vec{y}$ indicates that $\vec{x}$ comes first in the lexicographic order. Matrices are sets of row vectors in P^n. All matrix operations are mod p. $COL(M)$ denotes the set $\{i$: the i-th column in M is nonzero$\}$. Note that $COL(M)$ is also defined if M consists of only one row. Let $K(M)$ denote the *kernel* of M, i.e., $K(M) = \{\vec{x} : M\vec{x} = \vec{0}\}$, and let $span(M)$ be the set of all linear combinations of rows of M. Recall from linear algebra that if M is a matrix and B is a basis of M (more precisely, a basis of the vector space $span(M)$), then $K(M) = K(B)$.

Any truth assignment τ may also be interpreted as a vector $\langle \tau(v_1), \tau(v_2), \ldots, \tau(v_n) \rangle \in \{0,1\}^n \subseteq P^n$. The symbol τ will be used to denote either the function, or the vector; the meaning will be clear from context. For example, in "$\tau(v)$", the function is denoted, whereas the vector is denoted in "$\vec{x} \cdot \tau$". Similarly, any vector $\vec{x} \in \{0,1\}^n$ may be interpreted as a truth assignment, in which v_i is assigned false (respectively, true) iff $x_i = 0$ (respectively $x_i = 1$).

Definition 4.1 *If I is an instance of* 1-in-3-SAT *over variable set V, and $V' \subseteq V$, then the instance I* restricted to *V' (written $I/_{V'}$) is the instance of* 1-in-3-SAT *over variable set V' that contains exactly the clauses c of I for which every element of c is in V'. If $\tilde{I}$ is an instance of* 1-in-3-SAT *over n variables (possibly an instance derived by restricting another instance as above) then $SOL(\tilde{I}) = \{\vec{x} \in \{0,1\}^n : \vec{x}$ is a solution to the instance $\tilde{I}\}$.*

4.2 The Reduction and Examples

Let I be an instance of 1-in-3-SAT, with variables $V = \{v_1, v_2, \ldots, v_n\}$. We let the alphabet Σ for the problem MIN-CON(DFA, NFA) be V.

Recall that for any string $\gamma \in \Sigma^*$, $\vec{\gamma} = \langle x_1, x_2, \ldots x_n \rangle$, such that for each i, $1 \leq i \leq n$, the number of occurrences of v_i in γ is congruent to x_i mod p.

We now define a special word q, which is specified by a product (denoting concatenation) of all words of the following form: For any vector $\vec{x} \in P^n$, define

$$\vec{v}^{\vec{x}} = {v_1}^{x_1} {v_2}^{x_2} \cdots {v_n}^{x_n} {v_1}^{p-x_1} {v_2}^{p-x_2} \cdots {v_n}^{p-x_n}.$$

Note that $\vec{v}^{\vec{x}}$ has p occurrences of each letter v_i. Since the product sign below in the definition of q denotes concatenation, to be unambiguous, we must specify the order in which the terms (subwords) are concatenated: The choice in the product below is made in lexicographic order of the vectors $\vec{x}$.

$$q = \prod_{\substack{\vec{x} \in P^n \\ |COL(\vec{x})| \leq k}} \vec{v}^{\vec{x}}.$$

Note that $\vec{q} = \vec{0}$. Whenever they appear, α and β (and subscripted versions) will denote prefixes and suffixes, respectively, of q.

On input I, the transformation $R_{p,k}$ outputs $R_{p,k}(I)$ consisting of the two sets, POS(p, k, I) and NEG(p, k, I). We will show that these sets have the properties sketched above. Note that the transformation $R_{p,k}$ need only be computable in time polynomial in $|I|$, since p and k are constants. However, to prove the same results for DFAs and NFAs over the fixed two letter alphabet $\{0, 1\}$, we use the fact, proved below, that the transformation is also polynomial in the *value* p. (The reliance on k is doubly exponential however.) We now describe POS(p, k, I) and NEG(p, k, I).

POS(p, k, I) consists only of the word q^{p^k}.

NEG(p, k, I) is constructed as follows. Let $\{\alpha_i\}_{i=1}^m$ be any collection of m prefixes of q (recall $m = 3 \cdot 2^{k-1}$), let $\{\beta_i\}_{i=1}^m$, be any collection of m suffixes of q, and let $\langle p_1, p_2, \ldots, p_m \rangle$ be any element of P^m. Define $\gamma_i = \alpha_i \beta_i$. Let $\gamma = \prod_{i=1}^m (\gamma_i)^{p_i}$. Then if there exists a set D such that

1. $|D| \leq k - 1 + m$
2. $COL(\vec{\gamma}) \subseteq D \subseteq V$
3. $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$

then for any numbers a, b, c $(0 \leq a, b, c \leq p^k)$, include in the set NEG$(p, k, I)$ the string $\gamma_{a,b,c} = q^a (\prod_{i=1}^m (\gamma_i q^b)^{p_i}) q^c$.

Proposition 4.2 *The length of each example in* POS(p, k, I) *and* NEG(p, k, I), *and the number of examples in these sets, is polynomial in* $|I|$ *and in the value* p.

Proposition 4.3 *For any* k *and* p, $R_{p,k}(I)$ *is computable in time polynomial in* $|I|$ *and in the value* p.

Proof: By Proposition 4.2, clearly the only possible problem would be in determining for which strings γ, the strings $\{\gamma_{a,b,c}\}_{0 \leq a,b,c \leq p^k}$ should be included in NEG(p, k, I). In order to make this determination, it is sufficient to check all possible (at most $\mathcal{O}(n^{k-1+m})$) sets D of size $|D| \leq k - 1 + m$ and observe whether $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$. The determination of whether this intersection is empty can be achieved by enumerating every possible element in $SOL(I/_D)$ and checking whether any element is also an element of $K(\vec{\gamma})$. Note that $I/_D$ has at most $|D|^3$ clauses, hence $SOL(I/_D)$ can be enumerated in time $\mathcal{O}(2^{k-1+m}|D|^3)$ which is constant. Further, for each truth setting of $SOL(I/_D)$ it takes $\mathcal{O}(k-1+m)$ time to check whether it lies in $K(\vec{\gamma})$ and this too is a constant. □

4.3 A small consistent counter machine

We show that if I has some solution τ, then there is a small (p state) DFA consistent with POS(p, k, I) and NEG(p, k, I).

Proof of Lemma 3.2:
Recall that $C(p, \tau)$ accepts a string γ iff $\tau \in K(\vec{\gamma})$, i.e., iff the number of true variables occurring in γ is zero mod p. Since $\vec{q} = \vec{0}$, then for any power q^a of q, $\vec{q^a} = \vec{0}$. Clearly, $\tau \in K(\vec{q^a})$, and thus $C(p, \tau)$ accepts q^a. In particular, $C(p, \tau)$ accepts the only element q^{p^k} of POS(p, k, I).

To see that $C(p, \tau)$ rejects all elements of NEG(p, k, I), we show that if $C(p, \tau)$ accepts a string $\gamma_{a,b,c}$ (as described in the definition of NEG(p, k, I)), then $\gamma_{a,b,c}$ is not an element of NEG(p, k, I). Note that since q contains each variable occurring a number of times congruent to 0 mod p, if $C(p, \tau)$ accepts $\gamma_{a,b,c}$, then it also accepts any word formed from $\gamma_{a,b,c}$ by removing any number of copies of q. In particular, it also accepts γ, (described in the definition of NEG(p, k, I)). From the above comments it follows that $\tau \in K(\vec{\gamma})$. Further, by assumption $\tau \in SOL(I)$, hence for *any* $D \subseteq V$, $\tau \in SOL(I/_D)$. Thus certainly there is no set $D \subseteq V$ of size $|D| \leq k - 1 + m$, such that $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$, and therefore, for no choices of a, b, and c would the string $\gamma_{a,b,c}$ be placed in NEG(p, k, I). □

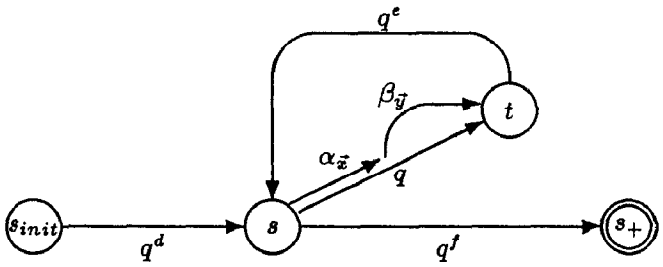

Figure 1: The loop of path ψ, and a bridge from s to t

To prove that p states are necessary in any NFA consistent with POS(p,k,I) and NEG(p,k,I) (Lemma 3.3), we first introduce some notation that will also be used in the proof of Lemma 3.4.

Let $A = (Q, \Sigma, \delta, s_{init}, F)$ be any NFA with fewer than p^k states that is consistent with POS(p,k,I) and NEG(p,k,I), and consider the graph representation of A. Since A is consistent, the positive example q^{p^k} defines a path ψ from s_{init} to some accepting state s_+. Since A has less than p^k states, there exist numbers d, e, and f such that $d+1+e+f = p^k$, and states s and t on path ψ such that: q^d leads from s_{init} to s; q leads from s to t; q^e leads from t to s; and q^f leads from s to s_+. Figure 1 shows the loop of ψ, together with states s and t, and the strings q^d, q^e, q^f.

For each $\vec{x} \in P^n$ for which $|COL(\vec{x})| \leq k$ we define a particular prefix $\alpha_{\vec{x}}$ and suffix $\beta_{\vec{x}}$ of q as follows

$$\alpha_{\vec{x}} = \left(\prod_{\substack{\vec{y} < \vec{x} \\ |COL(\vec{y})| \leq k}} \vec{v}^{\vec{y}} \right) {v_1}^{x_1} {v_2}^{x_2} \cdots {v_n}^{x_n}$$

and $\beta_{\vec{x}}$ is the unique suffix such that $\alpha_{\vec{x}}\beta_{\vec{x}} = q$. In other words,

$$\beta_{\vec{x}} = {v_1}^{p-x_1} {v_2}^{p-x_2} \cdots {v_n}^{p-x_n} \left(\prod_{\substack{\vec{y} > \vec{x} \\ |COL(\vec{y})| \leq k}} \vec{v}^{\vec{y}} \right).$$

Proof of Lemma 3.3:
It suffices to show that any consistent NFA with less than p^k states has at least p states. Let n be the number of variables of I and let $\{v_i, v_j, v_k\}$ be any clause of I. Let $\vec{x}_1$ be the n-component vector which contains all 0s except for components i, j and k, which are 1. Let $\vec{x}_r = r\,\vec{x}_1$, for $0 \leq r \leq p-1$.

For any prefix w of q^{p^k}, let $s_{init}(w)$ denote the state[2] that w leads to along ψ. It suffices to show that the states $\{s_{init}(q^d\alpha_{\vec{x}_r})\}_{0 \leq r \leq p-1}$ are distinct. Assume to the contrary, that for some r and r' such that $0 \leq r' < r \leq p-1$, we have $s_{init}(q^d\alpha_{\vec{x}_r}) = s_{init}(q^d\alpha_{\vec{x}_{r'}})$. Observe that $q^d\alpha_{\vec{x}_r}\beta_{\vec{x}_r}q^{e+f} = q^d\alpha_{\vec{x}_{r'}}\beta_{\vec{x}_{r'}}q^{e+f} = q^{p^k}$. Since $s_{init}(q^d\alpha_{\vec{x}_r}) = s_{init}(q^d\alpha_{\vec{x}_{r'}})$, we conclude that the word $\rho = q^d\,\alpha_{\vec{x}_r}\beta_{\vec{x}_{r'}}q^e\,q^f$ is accepted. We now obtain a contradiction by showing that $\rho \in$ NEG(p,k,I). Thus $s_{init}(q^d\alpha_{\vec{x}_r}) \neq s_{init}(q^d\alpha_{\vec{x}_{r'}})$, and A must have at least p states.

To see that ρ is a negative example, recall the definition of NEG(p,k,I). Set γ_1 to $\alpha_{\vec{x}_r}\beta_{\vec{x}_{r'}}$ and γ_l to $\alpha_{\vec{x}_r}\beta_{\vec{x}_r} = q$, for $2 \leq l \leq m$. Let $\langle p_1, p_2, \ldots, p_m \rangle$ be the vector $\langle 1, 0, \ldots, 0 \rangle$ of P^m and set D to be the clause $\{v_i, v_j, v_k\}$. The word ρ can be rewritten as $q^d(\prod_{i=1}^m(\gamma_i q^e)^{p_i})q^f = q^d\gamma_1 q^e q^f$. To show that ρ is a negative example we only need to show that $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$ for $\gamma = \prod_{i=1}^m(\gamma_i)^{p_i} = \gamma_1$. Note $\vec{\gamma}$ consists of all zeros except for the components i, j, and k, which have value $r - r'$. The restriction $I/_D$ consists of the clause D and thus any solution of $SOL(I/_D)$ must have exactly one of the three components corresponding to the variable of the clause set to 1 and the others set to 0. Since $1 \leq r - r' \leq p-1$, this implies that such a solution cannot lie in $K(\vec{\gamma})$ and $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$. □

4.4 Forcing polynomially larger NFAs

In this section we prove Lemma 3.4 by showing that if there exists an NFA consistent with POS(p,k,I) and NEG(p,k,I) with fewer than p^k states, then there exists a solution of instance I. Before sketching the proof of Lemma 3.4, we need a number of supporting propositions. The arguments to follow apply to any given NFA A that satisfies the hypothesis of Lemma 3.4. As described above, A must have a loop; in what follows we use the notation given in Figure 1.

For any set $X \subseteq \{1, 2, \ldots, n\}$ such that $|X| = k$, we define an X-bridge as follows. The string $\alpha_{\vec{x}}\beta_{\vec{y}}$ is an X-bridge iff $\vec{x} \neq \vec{y}$, $COL(\vec{x})$ and $COL(\vec{y})$ are both subsets of X, and the string $\alpha_{\vec{x}}\beta_{\vec{y}}$ leads from s to t. Figure 1 depicts a bridge.

Proposition 4.4 *For all $X \subseteq \{1, 2, \ldots, n\}$ such that $|X| = k$, there exists an X-bridge.*

Proof: Consider any X as in the hypothesis of the proposition. There are exactly p^k vectors $\vec{x}$ such that $COL(\vec{x}) \subseteq X$. Since $|Q| < p^k$, there must be two such

[2]Technically, if there are λ-transitions along ψ, then $s_{init}(w)$ is not necessarily well defined. However, λ-transitions can be eliminated without increasing the number of states [12].

vectors $\vec{x} \neq \vec{y}$ such that for some state r, the states $s_{init}(q^d\alpha_{\vec{x}}) = s_{init}(q^d\alpha_{\vec{y}}) = r$. The string $\alpha_{\vec{y}}\beta_{\vec{y}} = q$ leads from $s = s_{init}(q^d)$ to $t = s_{init}(q^{d+1})$, and thus $\beta_{\vec{y}}$ leads from r to t. Consequently, $\alpha_{\vec{x}}\beta_{\vec{y}}$ leads from s to t, and is therefore an X-bridge. □

A *bridge* is a string that for some X of size k, is an X-bridge. For each bridge $\alpha_{\vec{x}}\beta_{\vec{y}}$ let the string $\gamma_{\vec{x}\,\vec{y}} = \alpha_{\vec{x}}\beta_{\vec{y}}q^e$. By the definition of a bridge, and e, any $\gamma_{\vec{x}\,\vec{y}}$ leads from s to t and then back to s. Thus q^d (which leads from s_{init} to s), followed by any sequence of strings of the form $\gamma_{\vec{x}\,\vec{y}}$, followed by q^f (which leads from s to s_+) is accepted by A. (Refer to Figure 1.) We have just proved the following proposition.

Proposition 4.5 *Let $\{\alpha_i\}_{i=1}^m$ be any collection of prefixes of q, and $\{\beta_i\}_{i=1}^m$ any collection of suffixes of q, such that for each i, $1 \leq i \leq m$, the string $\gamma_i = \alpha_i\beta_i$ is a bridge. Then for any $\langle p_1, p_2, \ldots, p_m\rangle \in P^m$, and for any string $\gamma = \prod_{i=1}^m (\gamma_i q^e)^{p_i}$, the* NFA *$A$ accepts the string $q^d\gamma q^f$.*

Recall our earlier observation that in any counter machine $C(p, \tau)$, if γ is a word that leads from a state back to the same state, then τ satisfies $\vec{\gamma} \cdot \tau = 0$. Since every bridge $\gamma_{\vec{x}\,\vec{y}}$ is a word that forms a cycle in A, we will follow the approach discussed earlier, and from $\gamma_{\vec{x}\,\vec{y}}$ we derive the equation $\vec{\gamma}_{\vec{x}\,\vec{y}} \cdot \vec{z} = 0$. Our goal is to extract from A a small collection of bridges S such that at least one solution of the collection of associated equations is also a solution to the instance I. The examples $\mathrm{NEG}(p, k, I)$ will rule out collections of bridges whose corresponding sets of equations do not have this property.

Let R be the matrix with a row $\vec{\gamma}_{\vec{x}\,\vec{y}}$ for each bridge $\gamma_{\vec{x}\,\vec{y}}$ of A. Since for each X of size at most k, there is an X-bridge (Proposition 4.4), and each X-bridge is a string $\gamma_{\vec{x}\,\vec{y}}$ such that $COL(\vec{x})$ and $COL(\vec{y})$ are both contained in X, it follows that the vector $\vec{\gamma}_{\vec{x}\,\vec{y}}$ also is such that $COL(\vec{\gamma}_{\vec{x}\,\vec{y}}) \subseteq X$, since it consists only of $\alpha_{\vec{x}}$, $\beta_{\vec{y}}$, and some copies of q. Further note that $\vec{\gamma}_{\vec{x}\,\vec{y}} \neq \vec{0}$ because in any bridge $\gamma_{\vec{x}\,\vec{y}}$, $\vec{x} \neq \vec{y}$. Thus for each $X \subseteq \{1, 2, \ldots n\}$ of size at most k, there is a nonzero row in R with all nonzero entries appearing in columns indexed by a value of X. Such matrices have special properties that we exploit. In particular, we prove the following

Proposition 4.6 *There exists a subset $C \subseteq \{1, \ldots, n\}$ (called a "core" of R) such that $|C| \leq k - 1$, and for each number $i \in \{1, \ldots, n\} - C$, there is a row $\vec{x}$ of R (called a* determining row *of index i) such that $COL(\vec{x}) \subseteq C \cup \{i\}$ and such that x_i is nonzero.*

In other words, there are at most 2^{k-1} solutions to the system R; once the variables indexed by the core C are set, the values of all other variables are forced.

Now let S be any matrix of $n - |C|$ rows of R containing exactly one determining row for each index of $\{1, 2, \ldots, n\} - C$. Let V_C denote the set of variables indexed by the core, i.e., $\{v_i : i \in C\}$. For any set T of rows of S, let $V(T) = V_{COL(T)}$, the variables that occur with nonzero coefficient in some row of T. Given a setting of values for V_C, i.e. $\sigma : V_C \to \{0, \ldots, p-1\}$ then the determining rows in S for all indices outside of the core ensure that there is exactly one extension of σ to a setting of all variables $\sigma' : V \to \{0, \ldots, p-1\}$ that lies in $K(S)$.

Proposition 4.7 *Consider the numbered statements below.*

(i) $SOL(I) \cap K(S) = \emptyset$.

(ii) *There exists a subset of rows T of S such that $|T| \leq m = 3 \cdot 2^{k-1}$, and such that $SOL(I/_{C \cup V(T)}) \cap K(T) = \emptyset$.*

(iii) *There exists an element $\vec{\omega}$ of $span(T)$ such that $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega}) = \emptyset$.*

Then (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii).

Proof (sketch) that (i) implies (ii): This portion of the proposition asserts that if the system of equations S has no solution in common with the set of satisfying assignments of I, then a constant sized subset T of S has no solutions in common with the set of satisfying assignments of I *restricted* (see Definition 4.1) to those variables V_C of the core, and those variables $V(T)$ with nonzero entries in some row of T.

Since S consists of $n - |C|$ determining rows for variables in $V - V_C$, there are exactly $2^{|C|} \leq 2^{k-1}$ solutions of S. Since every solution τ is not a satisfying assignment of I, either τ is not $\{0, 1\}$ valued, or τ violates some clause of I. In the first case there is a row of S that, once the variables V_C are assigned, forces some variable of $V - V_C$ to have nonzero value. In the second case there are at most three rows of S witnessing that some clause of I is violated (one determining row for each variable of the clause). Thus we may include in T at most three rows for each of at most 2^{k-1} satisfying assignments of S. It follows that if S had no solution that was a satisfying assignment for I, then T also has this property, for the version of I restricted to the finite set of variables consisting of V_C together with the variables $V(T)$.

Proof that (ii) implies (iii): Suppose to the contrary, that no such element $\vec{\omega}$ existed. Then for each element $\vec{\omega}$ of $span(T)$, $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega}) \neq \emptyset$. Since $V(\vec{\omega}) \subseteq V(T)$, if $\tau : C \cup V(T) \to \{0, 1\}$, then either all extensions $\tau' : V \to \{0, 1\}$ of τ are such that

$\tau' \in SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega})$, or no extensions τ' are such that $\tau' \in SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega})$. In the first case, we say that τ is a witness to the nonemptiness of $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega})$. Since for each $\vec{\omega}$, we have $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega}) \neq \emptyset$, there is a witness for each $\vec{\omega}$. There are at most 2^{k-1+m} witnesses. Thus there must be some assignment $\tau_0 : C \cup V(T) \rightarrow \{0,1\}$ that is a witness for at least

$$\frac{|span(T)|}{2^{k-1+m}} > \frac{|span(T)|}{p}$$

elements of $span(T)$ (because $p > 2^{k-1+m}$). Let B be the subset of elements of $span(T)$ for which τ_0 is a witness, i.e., B is the set of elements of $span(T)$ such that for every $\vec{\omega} \in B$, every extension of τ_0 is an element of $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega})$. Immediately we have that every extension of τ_0 is an element of $SOL(I/_{C \cup V(T)}) \cap K(B)$.

The above bound on the size of B may be used to prove that B contains a basis of $span(T)$ and thus $K(T) = K(B)$. (The general lemma states that for any set of vectors M, if $U \subseteq span(M)$ has size larger than $\frac{|span(M)|}{p}$, then U contains a basis of $span(M)$.) Thus $SOL(I/_{C \cup V(T)}) \cap K(T) = SOL(I/_{C \cup V(T)}) \cap K(B)$. But since every extension of τ_0 is in $SOL(I/_{C \cup V(T)}) \cap K(B)$, this implies that $SOL(I/_{C \cup V(T)}) \cap K(T)$ is nonempty, contradicting the hypothesis of the proposition. □

Proof of Lemma 3.4: Assume the hypothesis of the lemma is true, we need only show there exists a solution of instance I. Thus Lemma 3.4 follows immediately from:

Proposition 4.8 $SOL(I) \cap K(S) \neq \emptyset$.

Proof: Suppose to the contrary that $SOL(I) \cap K(S) = \emptyset$, then by Proposition 4.7 there exists $T \subseteq S$ of size at most m such that $SOL(I/_{C \cup V(T)}) \cap K(T) = \emptyset$. This implies, again by Proposition 4.7, that there exists a vector $\vec{\omega} \in span(T)$ such that $SOL(I/_{C \cup V(T)}) \cap K(\vec{\omega}) = \emptyset$.

Recall that T consists of $l \leq m$ vectors $\vec{\gamma}_1, \vec{\gamma}_2, \ldots, \vec{\gamma}_l$, corresponding to bridges $\gamma_1, \gamma_2, \ldots, \gamma_l$, where each γ_i is formed from the concatenation of some prefix α_i and some suffix β_i of q. For syntactic convenience, define $\gamma_j = \gamma_1$ for $l < j \leq m$. By the definition of $span(T)$, there exists $\vec{p} \in P^m$ such that $\vec{\omega} = \sum_{i=1}^{m} p_i \vec{\gamma}_i$. Let the string $\gamma = \prod_{i=1}^{m} (\gamma_i)^{p_i}$. Then clearly $\vec{\gamma} = \vec{\omega}$, and thus $SOL(I/_{C \cup V(T)}) \cap K(\vec{\gamma}) = \emptyset$. Now define $D = C \cup V(T)$, and we have that $SOL(I/_D) \cap K(\vec{\gamma}) = \emptyset$, $|D| \leq k - 1 + m$, and $COL(\vec{\gamma}) \subseteq D$. Then by the definition of $NEG(p,k,I)$, for any a, b, c, $\gamma_{a,b,c} = q^a (\prod_{i=1}^{m} (\gamma_i q^b)^{p_i}) q^c$ is an element of $NEG(p,k,I)$. But when $a = d, b = e$, and $c = f$, by Proposition 4.5, this string is accepted by A, contradicting the consistency of A. We conclude that $SOL(I) \cap K(S) \neq \emptyset$, completing the proof of Proposition 4.8 and Lemma 3.4. □

5 Extensions

In the complete paper [17] we show that our Main Theorem (Theorem 3.1) remains true for the two letter case. Let $\text{DFA}^{\{0,1\}}$ denote the class of DFAs over the two letter alphabet $\{0,1\}$ and similarly for $\text{NFA}^{\{0,1\}}$.

Theorem 5.1 *If* P≠NP, *then for all positive integers* k, MIN-CON($\text{DFA}^{\{0,1\}}, \text{NFA}^{\{0,1\}}$) *is not* opt^k*-approximable.*

The proof of this theorem uses a modification of the reduction of the previous section. In the new examples each of the n variables of the instance of 1-in-3-SAT is encoded as a bit string of length (roughly) $\log n$.

Let $\text{REGEXPR}^{\{0,1\}}$ and $\text{REGGRAM}^{\{0,1\}}$ be the class of regular expressions and regular grammars [12], respectively, over the two letter alphabet $\{0,1\}$. Let the size of a regular expression be the number of symbols in the expression and let the size of a regular grammar be the total number of symbols in all of the productions of the grammar. Since for each regular expression (and similarly for each regular grammar) there is an NFA of size at most a constant larger that accepts the same language [12], the following is implied by the previous theorem.

Theorem 5.2 *If* P≠NP, *then for all positive integers* k, MIN-CON($\text{DFA}^{\{0,1\}}, \text{REGGRAM}^{\{0,1\}}$) *and* MIN-CON($\text{DFA}^{\{0,1\}}, \text{REGEXPR}^{\{0,1\}}$) *are not* opt^k*-approximable.*

Because the above theorem shows polynomial nonapproximability, the theorem holds with respect to other natural size measures than the ones introduced above. By an involved extension of the reduction of the previous section we prove a nonapproximability result for linear grammars in [17]. Let LIN be the class of linear grammars.

Theorem 5.3 *If* P≠NP, *then for all positive integers* k, MIN-CON(LIN,LIN) *is not* opt^k*-approximable.*

6 Approximability and Learnability

Following [7], let Π be a minimization problem, let I be any instance of Π, and let A be an approximation algorithm. There are a number of ways we might

wish to measure the performance of A. One measure is to simply express the size of the approximate solution as a function of the size of the smallest feasible solution. Consider the minimization problem MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) of Theorem 5.1. In this case, an instance I consists of a collection of positive and negative examples, $opt(I)$ is the number of states in the smallest consistent DFA, and for any approximation algorithm A, $A(I)$ is the consistent NFA produced by A and $|A(I)|$ is the number of states of the NFA produced. Theorem 5.1 states that if P$\neq$NP, then no polynomial time algorithm A for MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) can achieve $|A(I)| < opt(I)^k$ for any constant k.

Another reasonable measure is to express the ratio $\frac{|A(I)|}{opt(I)}$ as a function of the size of the input (denoted by $|I|$). In the case of MIN-CON, $|I|$ is the total number of letters in all of the examples. In the reductions $R_{p,k}$ of this paper, $|I|$ grows quickly relative to $opt(I) = p$ and k. (If $p \geq n$, the number of variables, then $|I|$ may be as large as $p^{O(k2^k)}$.) Consequently, the best non-approximability result with respect to this measure is obtained for the special case $k = 2$. In this case, in [17], we present a very different reduction (the techniques of which are of interest in their own right) introducing a quadratic gap for which we show the bound of Theorem 6.1 below. These bounds are stronger than those obtained by using the reductions $R_{p,2}$ of this paper.

Note that if P=NP, then $\frac{|A(I)|}{opt(I)} = 1$ for MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$). For an arbitrary DFA or NFA M, let $||M||$ be the number of bits needed to encode M according to some standard encoding. Recall that $|M|$ is the number of states of M. We assume that $||M|| \geq |M|$.

Theorem 6.1 *If* P$\neq$NP, *then for any* $\epsilon > 0$, *for any polynomial time approximation algorithm* A *for* MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$), *and for infinitely many numbers* c, *there are instances* I *of* MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) *such that* $opt(I) \geq c$, *and for which the performance ratio of* A *satisfies* $\frac{||A(I)||}{opt(I)} \geq \frac{|A(I)|}{opt(I)} > |I|^{1/(14+\epsilon)}$.

A standard measure of performance of an approximation algorithm A is the *asymptotic performance ratio* (denoted R_A^∞) [7], defined by $R_A^\infty = \inf\{r \geq 1 :$ for some positive integer c, $\frac{|A(I)|}{opt(I)} \leq r$ for all instances I such that $opt(I) \geq c\}$. Theorem 6.1 above implies that, unless P=NP, R_A^∞ is infinite for MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$).

We next introduce another measure of approximation performance which is motivated by recent work in computational learning theory. Pac-learning of a class of objects (e.g., of DFAs) requires that from randomly generated examples of some unknown member of the class (the target DFA), a (possibly different) member of the *same class* (called the hypothesis) is produced that is likely to agree (in a precisely quantified way) with future examples generated from the same distribution [20]. A relaxation of this definition allows pac-learning of a class *in terms of* some other class [16]. For example, to pac-learn DFAs in terms of NFAs, a learning algorithm may choose its hypothesis from the class of NFAs. Thus pac-learning of DFAs in terms of NFAs is easier than pac-learning of DFAs (in terms of DFAs). It follows from [5] that the pac-learnability of DFAs in terms of NFAs would be established from the existence of a polynomial time algorithm A, and any constants $\alpha \geq 0$ and $\beta < 1$, with the following properties: A, on input of any instance I of the MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) problem (i.e., finite sets POS and NEG), will produce a consistent NFA $A(I)$ such that

$$||A(I)|| \leq opt(I)^\alpha card(I)^\beta, \tag{1}$$

where $opt(I)$ is the size of the smallest DFA consistent with the examples of I, and $card(I)$ is the number of examples of I.

If we restrict our attention to pac-learning DFAs in terms of NFAs from *polynomially length bounded* examples (all examples with nonzero probability are at most polynomially larger than the size of the DFA to be learned), then the results of [5] also give that pac-learnability is implied by the existence of a polynomial time algorithm A, and any constants $\alpha \geq 0$ and $\beta < 1$, such that on input of any instance I of MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$), A outputs a consistent NFA $A(I)$ such that

$$||A(I)|| \leq opt(I)^\alpha |I|^\beta, \tag{2}$$

where $|I|$ is the total size of all examples of I.

Consequently, an approach toward pac-learning of DFAs in terms of NFAs from polynomially length bounded examples is to produce a polynomial time algorithm that satisfies the guarantee of inequality (2) for instances of MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) for which the elements of POS and NEG are polynomially length bounded in the size of the smallest consistent DFA.

All of the nonapproximability results of this paper in fact apply to a restricted version of MIN-CON in which the size of the elements of POS and NEG are polynomially length bounded. We complete this section by investigating the implications that these (and other) reductions have with respect to the performance criterion given by inequality (2).

Recently, Kearns and Valiant [13] have shown that DFAs are not *polynomially predictable* based on any of several well established cryptographic assumptions:

that deciding quadratic residuosity is hard, that the RSA public key cryptosystem is secure, or that factoring Blum integers is hard. Their results hold even if the examples are polynomially length bounded.

Polynomial predictability is equivalent to pac-learnability in terms of the class of polynomially sized programs (i.e., the hypothesis may be any polymomial time algorithm for classification of examples which is representable with polynomially many bits) [10]. In [13], it is shown that the nonpredictability of DFAs, together with the results in [5], imply that there is no polynomial time algorithm A for MIN-CON($\mathrm{DFA}^{\{0,1\}}$, $\mathrm{NFA}^{\{0,1\}}$) producing a consistent NFA $A(I)$ such that inequality (2) holds for any constants $\alpha \geq 0$ and $\beta < 1$, unless the previously mentioned cryptographic assumptions are false.

Our results, given in Corollary 6.2, complement those presented in [13]; we obtain analogous nonapproximability results for restricted choices of α and β, but using the (ostensibly weaker) assumption that P≠NP.

Corollary 6.2 *If* P≠NP, *then for any polynomial time approximation algorithm A for* MIN-CON($\mathrm{DFA}^{\{0,1\}}$, $\mathrm{NFA}^{\{0,1\}}$), *and for infinitely many numbers c, there are instances I of* MIN-CON($\mathrm{DFA}^{\{0,1\}}$, $\mathrm{NFA}^{\{0,1\}}$) *such that $opt(I) \geq c$, and for which the performance of A satisfies $\|A(I)\| \geq |A(I)| > opt(I)^{\alpha}|I|^{\beta}$ for any of the following choices of α and β:*

1. *for any $\alpha \geq 0$, when $\beta = 0$;*
2. *for $\alpha = 1$ and any $\beta < \frac{1}{14}$;*
3. *for any $0 \leq \alpha = 1 + \alpha' < 2$ and any $\beta < \frac{1-\alpha'}{14}$.*

Besides the use of different assumptions (i.e., cryptographic versus P≠NP), another difference between our work and that appearing in [13], is that while the cryptographic based results of [13] rely on the inability to predict DFAs, the subfamily of DFAs for which we show nonapproximability results is actually easy to predict. The class of CDFAs accept *permutation invariant* languages (w is accepted iff any word formed by permuting the characters in w is accepted), and for each CDFA the start state equals the unique final state. DFAs with these properties have been shown to be predictable [11], thus the techniques of [13] cannot apply to show that the related MIN-CON problem for this restricted class of DFAs is not polynomially approximable.

7 Conclusion

The problem of finding an approximately small DFA consistent with a finite sample was investigated. It was shown that unless P=NP, no polynomial time algorithm can be guaranteed to produce a DFA, NFA, regular expression, or regular grammar of size at most polynomially larger than the smallest consistent DFA. The minimum consistent linear grammar problem was also shown to have the same nonapproximability properties. Our results are summarized by the statements of Theorems 3.1, 5.1, 5.2, 5.3, and 6.1. A complete proof was given for the Main Theorem (Theorem 3.1), the nonapproximability result for MIN-CON(DFA,NFA) (variable alphabet size). For detailed remaining proofs we refer the reader to the complete paper [17].

It should be noted that the proofs of each of these theorems was nonconstructive in the sense that from an approximately small NFA (regular grammar, regular expression, linear grammar, respectively), it was shown that a solution to an instance of 1-in-3-SAT must have existed. The proofs can be easily modified so as to be constructive, i.e., so that from an approximately small NFA (for example), a solution to the relevant 1-in-3-SAT instance can be found in polynomial time. Of course, since the problem of finding a solution to a 1-in-3-SAT problem reduces in polynomial time to the decision problem, our observation concerning constructiveness is of dubious interest.

In our definition of approximability (Definition 2.3) we required that the approximation algorithm must output a representation of size less than the upper bound. It should be noted that all nonapproximability results of this paper still hold when the approximation algorithm only *decides* whether there exists a consistent representation of size less than the upper bound.

Because the DFAs used in the reductions of this paper were of a very special form (CDFAs or counter-like DFAs), the proof of Theorem 3.1 (for example) actually shows the stronger result that for any constant k, MIN-CON(CDFA,NFA) (and thus MIN-CON(DFA,NFA)) is not opt^k-approximable unless P=NP. As discussed at the end of the previous section, it has been shown that CDFAs are polynomially predictable [11].

The research presented here suggests a large number of open problems. The investigation of the approximability of versions of the MIN-CON problem other than the ones considered here seems appropriate. Can the nonapproximability results (assuming P≠NP) for MIN-CON(LIN,LIN) be extended to MIN-CON(CFG,CFG)? (At present, it is not even known whether it is NP-hard to find the *smallest* consistent CFG). In the problem MIN-CON(DFA,NFA), the approximation algorithm has the "freedom" to output a consistent NFA instead of a consistent DFA. Further generalizing along these lines, it would be of interest to know whether similar nonapproximability results may be shown for MIN-CON(DFA,2DFA), MIN-CON(DFA,2NFA), MIN-CON(DFA,LIN), and MIN-

CON(DFA,CFG), etc., where 2DFAs and 2NFAs are the two-way versions of DFAs and NFAs, respectively. It has been shown that MIN-CON(DNF,DNF) is not $(2-\epsilon)opt$-approximable [16], (where DNF denotes the set of Boolean formulas in disjunctive normal form). Can this result be strengthened using the techniques of this paper?

Another set of Boolean functions other than DNF that is of interest in computational learning theory is the set of Boolean decision trees (DT). It has been recently shown in [9] that MIN-CON(DT,DT) is not $opt + opt^{\beta}$ approximable for any constant $\beta < 1$. The decision trees used in the reduction are very unbalanced. Let a *balanced decision tree* (BDT) have the additional property that all leaves are on the same level. Can it be decided in polynomial time whether there is a BDT that is consistent with given examples POS and NEG?

In Section 6, the work of [13] was discussed. These results show nonapproximability for DFAs based on cryptographic assumptions. By relying instead on the assumption that P≠NP, our results strengthen theirs, but only for a subrange of the parameters α and β. Can the entire range of results presented in [13] be proved using only the assumption that P≠NP?

This paper presented a number of very strong nonapproximability proofs for certain types of NP-hard optimization problems. A final open problem is whether similar proof techniques can be used to obtain nonapproximability results for other classical NP-hard problems.

Acknowledgments

We thank Dana Angluin, Ming Li, and Umesh Vazirani for getting us interested in this problem, Don Hatch for helping to clarify the group theoretic structure in some of our early reductions, and Michael Kearns for discussions regarding the material of Section 6. Additional thanks to Dana for carefully plowing through an early draft containing the technical arguments (and little else).

References

[1] D. Angluin. An application of the theory of computational complexity to the study of inductive inference. 1976. Ph.D. Thesis, Electrical Engineering and Computer Science Department, University of California, Berkeley.

[2] D. Angluin. *Negative Results for Equivalence Queries.* Technical Report YALEU/DCS/RR-648, Department of Computer Science, Yale University, September 1988.

[3] D. Angluin. On the complexity of minimum inference of regular sets. *Inform. Contr.*, 39(3):337–350, 1978.

[4] D. Angluin. Private communication. 1988.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Inform. Process. Letters*, 24:377–380, 1987.

[6] M. Garey and D. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976.

[7] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, California, 1979.

[8] E. M. Gold. Complexity of automaton identification from given data. *Inform. Contr.*, 37:302–320, 1978.

[9] T. Hancock. Finding the smallest consistent decision tree is NP-hard. 1989. Unpublished manuscript, Harvard University.

[10] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. *Equivalence of Models for Polynomial Learnability.* In *Proceedings of the 1st Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1988.

[11] D. Helmbold, R. Sloan, and Manfred K. Warmuth. *Bootstrapping One-sided Learning.* 1988. Extended abstract, Dept. of Computer and Information Sciences, University of California at Santa Cruz.

[12] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachussetts, 1979.

[13] M. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Assoc. Comp. Mach., New York, May 1989. Begins next page of these proceedings.

[14] M. Li and U. Vazirani. On the learnability of finite automata. In *Proceedings of the 1st Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, California, August 1988.

[15] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 229–234, Assoc. Comp. Mach., New York, May 1988.

[16] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

[17] L. Pitt and M. K. Warmuth. *The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial.* Technical Report UIUCDCS-R-89-1499, University of Illinois at Urbana-Champaign, February 1989.

[18] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, Assoc. Comp. Mach., New York, 1978.

[19] B. A. Trakhtenbrot and Ya. M. Barzdin. *Finite Automata.* North-Holland, Amsterdam, 1973. pp. 98-99.

[20] L. G. Valiant. A theory of the learnable. *Comm. Assoc. Comp. Mach.*, 27(11):1134–1142, 1984.