Defying Hardness With a Hybrid Approach

Ryan Williams
August 2004
CMU-CS-04-159

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A hybrid algorithm is a collection of *heuristics*, paired with a polynomial time procedure S (called a *selector*) that decides based on a preliminary scan of the input which heuristic should be executed. We investigate scenarios where the selector must decide between heuristics that are "good" with respect to different complexity measures, *e.g.* heuristic h_1 is efficient but approximately solves instances, whereas h_2 exactly solves instances but takes superpolynomial time. We present hybrid algorithms for several interesting problems Π with a "hardness-defying" property: there is a set of complexity measures $\{m_i\}$ whereby Π is conjectured or known to be hard (or unsolvable) for each m_i, but for each heuristic h_i of the hybrid algorithm, one can give a complexity guarantee for h_i *on the instances of* Π *that* S *selects for* h_i that is *strictly better* than m_i. For example, some NP-hard problems admit a hybrid algorithm that given an instance can either solve it exactly in "subexponential" time, or approximately solve it in polytime with a performance ratio exceeding that of the known inapproximability of the problem (under $\mathsf{P} \neq \mathsf{NP}$).

The author was supported in part by an NSF Graduate Research Fellowship, and the NSF ALADDIN Center under Grant No. 0122581.

Keywords: approximation, exact algorithms, hybrid algorithms, algorithm selection, complexity

"Plan for what is difficult while it is easy."
– Tao Te Ching

1 Introduction and Motivation

We propose a new avenue for confronting hard problems, inspired by 'hybrid algorithms' (also called 'algorithm portfolios'). Here we will define a hybrid algorithm as a collection of algorithms $\mathcal{H} = \{h_1, \ldots, h_k\}$ called *heuristics*, coupled with an efficient (polynomial time) procedure S called a *selector*. Given an instance x of a problem, $S(x)$ returns the index i of some heuristic in $h_i \in \mathcal{H}$ to be executed on the instance. The intention behind S is that it should somehow select the "best" h_i for solving or deciding x, according to some predefined criteria. The design of selectors given an existing collection of heuristics in practice is (quite naturally) known as the *algorithm selection problem* [27] and has been studied in numerous contexts within artificial intelligence and operations research (cf. [24, 23, 19, 9, 13, 5] for a sample).

With the exception of some studies in competitive analysis (*e.g.* [22]) where one selects from an unbounded number of heuristics, theoretical computer science has mostly resisted the selection problem and its relatives, out of apparent (asymptotic) triviality: given a constant number of heuristics, if runtime is the desired measure then one may simply interleave the runs of all heuristics, until one of them halts. However, when heuristics are good according to somewhat orthogonal measures of complexity, algorithm selection becomes a interesting and highly non-trivial process. For example, one hybrid algorithm we present is for satisfying a maximum number of linear equations over $GF(p)$ with $k \geq 3$ variables per equation, for constant k (formally, MAX-Ek-LIN-p). Also called *Bounded Gaussian elimination with errors*, this problem has been widely studied in learning, cryptography, and complexity theory. When k is odd, for all fixed $\varepsilon > 0$ our algorithm for this problem does *exactly one* of the following, after a polynomial time test of an n variable instance on m equations:

- Produces an optimal solution in $O(p^{\varepsilon n})$ time, or
- Approximates the optimum in polynomial time, with a $1/p + \varepsilon'$ performance ratio, for $\varepsilon' = O(\varepsilon/\Delta)$,

where $\Delta = m/n$. ($\Delta = O(1)$ is generally the "hard case", cf. Section 4. We can also formulate the tradeoff as $2^{\varepsilon m}$ time versus $1/p+\varepsilon'$ approximation, without the Δ-dependence.) This algorithm suggests an intriguing prospect–that a "hybrid" approach to hard problems could circumvent known hardness results, in a sense. To be precise: if MAX-E3-LIN-2 is in $O(2^{\varepsilon n})$ time for *all* ε, it is not hard to show that many other hard problems are also solvable within that time. Similarly, MAX-E3-LIN-p is inapproximable within $1/p + \varepsilon$ for *any* $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$ [14]. Therefore neither of these two measures seem universally achievable, but we *can* efficiently select exactly one of the measures on every instance. This is surprising on (at least) two accounts.

(1) It appears that, for many problems, the special cases admitting improved approximation and the cases with improved exact solution typically have great overlap. There are many examples of this in the literature. For one, PLANAR DOMINATING SET has a PTAS [1] and is fixed-parameter tractable [8], whereas DOMINATING SET is probably $(1-\varepsilon)\log n$-inapproximable [11], and not fixed-parameter tractable unless $W[2] = FPT$ [8]. Moreover, the prevailing intuition has been that problem classes inapproximable within some constant also do not admit subexponential

exact solution (*e.g.* MAX-SNP, cf. [6, 20]). Hence such a partitioning seems, *a priori*, to be either unlikely or undoable.

(2) Even if the partitioning is possible, the most naïve way of doing so requires a selector capable of solving NP-hard problems (which would defeat the point). Thus an efficient selector is in itself interesting.

While our algorithm heavily exploits properties of field arithmetic, the problem of Gaussian elimination with errors is fundamental to many hardness results in complexity and cryptography. We therefore feel that this hybrid algorithm (along with the several others described in this paper) suggests that many interesting hybrid algorithms exist for hard problems, and that our developments in this paper give some directions on how one might find more of these algorithms. Indeed, we will show that the above generalizes to certain hard-to-approximate constraint satisfaction problems, cf. Section 6. On the other hand, we will also show several limitations on hybrid algorithms of the exact/approximate variety, based on natural hardness assumptions (*e.g.* SAT is in $2^{\Omega(n)}$ time, $\mathsf{P} \neq \mathsf{NP}$).

1.1 A (worst-case) alternative to worst-case analysis

Traditional analysis of algorithms generally proves a single guarantee about the performance of an algorithm on its inputs. Our approach provides a way to formally describe the performance of a collection of heuristics in a *worst-case* setting, despite the fact that each single heuristic is only good on a small number of instances, and each one is good with respect to different measures. For example, the case of efficiently selecting either an exact algorithm or an approximate one may be seen in a very practical light. In several practical applications, one simply desires the best possible solution for an instance that can be found within the resources available. Allowing the parameter ε to vary, one may fine-tune a hybrid algorithm to fit these resource bounds. A researcher may be willing to devote a great number of cycles to solve a hard problem, if it results in an exact solution. However, if a solution will not be found, one will not want to spend a long time to discover that. In that case it is reasonable to think that one might settle for an approximately good solution. As a consolation prize for not getting the optimum, this solution is provided quickly.

1.2 Instance-based resource tradeoffs

Our work was first motivated by attempting to prove resource tradeoffs in complexity theory. Several quite surprising tradeoffs of the form "one of two interesting class equalities/simulations holds" have been established in complexity. The most of well-known of these are probably the continuously-growing family of "hardness versus randomness" tradeoffs, initiated with [3, 32]. Abstractly, our proposal is an instance-based version of resource tradeoffs. Rather than attempting to establish that one of two interesting class equivalences/separations hold, we wish to show that for every instance of some problems, one of two interesting procedures may be decided upon and executed accordingly. Moreover, we also try to characterize the degree to which such hybrid algorithms can be developed.

2 Outline

The following section gives some notation and definitions to be used throughout; note some of this notation is new and thus should not be skipped. Next we will present a few interesting hybrid algorithms for various hard problems, such as MAX-E3-LIN-2 and quantified Boolean formulas. Finally, we will discuss a few hardness results, concerning the impossibility of various hybrid algorithms and selectors.

3 Background

3.1 Preliminaries

For a set S and $k \in \mathbb{Z}^+$, define $[k] := \{1, \ldots, k\}$, and $\binom{S}{k}$ to be the collection of subsets of S with cardinality at most k. In MAX-E3-LIN-p, we are given a set of m linear equations (each equation having three variables) over n variables with values in $\mathbb{Z}_p$. That is, every equation is of the form $\sum_{j \in I} x_j \equiv b \mod p$, for some $I \in \binom{V}{3}$ and $b \in [p]$, where V is the set of variables $\{x_1, \ldots, x_n\}$. We wish to find a setting of the x_i's whereby a maximum number of equations hold. It will be convenient to translate the set of equations into a set of *constraints* to be set to zero, in which case an equation $(\sum_{j \in I} x_j \equiv b \mod p)$ becomes a constraint $(\sum_{j \in I} x_j - b)$. For a given constraint $c = (\sum_{j \in I} x_j - b)$, define $\mathsf{vars}(c) := \{x_j \in V \ : \ j \in I\}$. An i-constraint is defined as a constraint with exactly i variables.

An algorithm A solving an optimization problem Π exhibits *improved exponential time* when naïve brute force search for Π takes $O(d^n)$ time, and A runs in $O(c^n)$ time for some $c < d$. A runs in *subexponential time* if it is $2^{o(n)}$. We will use $\tilde{O}$ in such bounds to denote the subsumption of a polynomial multiplier.

For maximization problems (respectively, minimization problems), define the approximation ratio r of an algorithm to be the minimum (respectively, maximum) ratio of the algorithm's returned solution to the optimal solution, over all instances. (So for us, $r \leq 1$, and larger r imply better approximations.) It will be useful for us to define a natural complexity class involving both approximation complexity and time complexity.

Definition 3.1 *An optimization problem Π is in the class $\mathsf{APX\text{-}TIME}[r(n), t(n)]$ if there is an algorithm A that always returns a feasible solution y to Π in* time $t(n)$*, and the* approximation *ratio of A is $r(n)$.*

We also define a *polynomial select* of two complexity classes, as a formal version of algorithm selection.

Definition 3.2 *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes. A problem Π is in $\mathcal{C} \oplus_P \mathcal{D}$ if there is a polytime function f from instances of Π to $\{0,1\}$ such that $\{x \in \Pi : f(x) = 0\} \in \mathcal{C}$, $\{x \in \Pi : f(x) = 1\} \in \mathcal{D}$.*

As $C \cap D$ denotes the "intersection of C and D", we propose to call $C \oplus_P D$ the *p-selection of C and D*. To illustrate, Theorem 5.1 says MAX-E3-LIN-2 $\in \bigcap_{\varepsilon > 0}(\mathsf{TIME}[2^{\varepsilon n}] \ \oplus_p \ \mathsf{APX\text{-}TIME}[\frac{1}{2} + \frac{\varepsilon}{12\Delta}, n])$, where $\mathsf{TIME}[t(n)]$ is the class of optimization problems solvable in $t(n)$ time. (Unless otherwise stated, t always denotes a time constructible function.) The definition of p-selection is intended to be a complexity-theoretic classification of problems solvable by hybrid algorithms, in the following sense.

Definition 3.3 *A* hybrid algorithm *for optimization problem* Π *is a pair* $(\mathcal{H}, S)$*, where*

- $\mathcal{H} = \{h_1, \ldots, h_k\}$ *is a collection of algorithms, a.k.a.* heuristics*, and*
- S *is a polytime algorithm from instances of* Π *to* $\{1, \ldots, k\}$*, a.k.a. a selector, such that*

for all $i = 1, \ldots, k$ *and* $y \in \{x \in \Pi : S(x) = i\}$*,* $h_i(y)$ *returns a feasible solution to* y*.*

The feasible solution requirement in the definition simply ensures that, if S selects h_i to be run on y, then h_i returns something that makes sense. With this definition, we may put forth the following simple proposition connecting p-selection and hybrid algorithms, stated informally for the purposes of exposition.

Informal Proposition. *Let* $(\{h_1, \ldots, h_k\}, S)$ *be a hybrid algorithm for* Π*, and let* $\mathcal{C}_1, \ldots, \mathcal{C}_k$ *be complexity classes. Define* $L_i := \{x \in \Pi : S(x) = i\}$*. If for all* $i = 1, \ldots, k$ *it holds that* $L_i \in \mathcal{C}_i$ *and* h_i *solves* L_i *within the resources of* $\mathcal{C}_i$ *(i.e.* h_i *is a "witness" to* $L_i \in \mathcal{C}_i$*), then*

$$\Pi \in (\mathcal{C}_1 \oplus_P (\mathcal{C}_2 \oplus_P \cdots (\mathcal{C}_{k-1} \oplus_P \mathcal{C}_k))).$$

3.2 Prior Work

As mentioned above, the idea of having several algorithms with varying (time) performance on different cases is found in work on algorithm portfolios, hybrid algorithms, and the algorithm selection problem. The literature is vast and highly variable; it has recently seen the most notable level of study within machine learning. We have not encountered proposals explicitly similar to ours in the literature, though there are some superficially related ones. P-selective sets [28] are vaguely related to our notion of an efficient test. A set $S \subseteq \Sigma^*$ is P-selective if, for every pair (x, y), there is a polytime algorithm $A : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that (a) $A(x, y) \in \{x, y\}$ and (b) ($x \in S$ or $y \in L$) implies $A(x, y) \in L$. In our case we have one instance at a time and need only decide what algorithm to run, a potentially simpler choice. Some work has been devoted to (automatically) identifying cases for which certain approximation algorithms work well, with mostly negative results. For example, [4] show that identifying graphs where the canonical greedy algorithm for maximum independent set yields a ratio $r \geq 1$ is co-NP-hard, for any fixed r.

For all $k \geq 3$ and primes p, MAX-Ek-LIN-p (when there are exactly k variables per equation) is $1/p$-approximable by choosing random assignments, and this is optimal unless $P = NP$ [14]. We do not know of improved exptime algorithms for MAX-3-LIN-2, though some have been proposed for MAX-3-SAT (for example, [16]). Work on incorporating improved exptime and approximation in an interesting way has been extremely sparse, with the exception of Dantsin, Gavrilovich, Hirsch, and Konev [7] who show that MAX-3-SAT can be $(7/8 + \varepsilon)$-approximated in $O(2^{8\varepsilon m})$ time (later improved to be in terms of n [16]). Our proposal is of course much stronger, in that we only wish to commit to subexponential time for an exact solution.

4 Hardness of solving MAX-k-LIN-2 exactly

Before we give the MAX-k-LIN-2 algorithm, let us first (briefly) outline why it is unlikely for the problem to be exactly solvable in subexponential time. A standard reduction from MAX-k-LIN-2 to $(k+1)$-SAT gives the property that if there is an algorithm for MAX-k-LIN-2 running in $2^{\varepsilon v}$ time (where v is the number of variables) for all $\varepsilon > 0$, then $(k+1)$-SAT is solvable in $2^{\varepsilon(m+n)}$ time for all $\varepsilon > 0$ (here m is the number of clauses, n is the number of variables). The Sparsification Lemma of Impagliazzo, Paturi, Zane [20] (which we will abbreviate as *IPZ*) implies that this time bound can be improved to $O(2^{\varepsilon n})$ for all ε, by a series of careful backtracking reductions on an instance. These results imply a more general result.

Theorem 4.1 *If* MAX-k-LIN-2 *is in* $2^{\varepsilon n}$ *time for all* $\varepsilon > 0$*, then* $(k+1)$-SAT *on* n *variables,* Vertex Cover *(*Clique*, and* Independent Set*) on* n *vertices, and* Set Cover *on* k*-sets over a universe of size* n *are all solvable in* $2^{\varepsilon n}$ *time.*

In fact, the hypothesis can also be replaced with the assumption that MAX-k-LIN-2 *is in* $2^{\varepsilon m}$ *time*, as the reduction from $(k+1)$-SAT to MAX-k-LIN-2 only introduces $O(km)$ equations (where m is the number of clauses in the k-SAT instance). Moreover, the following reduction shows in yet another sense that the hard cases of approximation are the ones where $m/n = O(1)$. This is relevant to our cause, as the algorithm we will give works best on instances with this property.

Lemma 4.1 *Suppose* MAX-E3-LIN-2 *can be solved exactly in time* $t(m)$*, where* m *is the number of equations. Then for all* $\varepsilon > 0$*, there is a randomized* $(1-\varepsilon)$*-approximation algorithm (with high success probability) for* MAX-E3-LIN-2 *running in* $O(poly(n) \cdot t(n/\varepsilon^2))$ *time, where* n *is the number of variables.*

Proof 4.1 *See appendix.* □

The proof of the above lemma is not dependent on the number of variables per equation or equations mod 2, and holds for *any* constraint satisfaction problem where one has m constraints and n variables. It is interesting that, while IPZ has the same aim (reducing time bounds in terms of m to time bounds in terms of n), their lemma preserves exact solution, but their proof is more complicated and does not work for *arbitrary* constraint satisfaction.

5 Algorithm for MAX-E3-LIN-2

We will now establish the hybrid algorithm mentioned in the Introduction.

Theorem 5.1 *For all* $\varepsilon > 0$*, there is a triple of procedures* $(S_\varepsilon, E_\varepsilon, A_\varepsilon)$ *such that, for all instances* F *of* MAX-3-LIN-2 *with* n *variables:*

- $S_\varepsilon(F) \in \{0,1\}$ *and runs in* $|F|^{O(1)}$ *time.*
- $S_\varepsilon(F) =$ *"exact"* $\Longrightarrow E_\varepsilon(F)$ *returns an optimal solution to* F *in* $\tilde{O}(2^{\varepsilon n})$ *time.*
- $S_\varepsilon(F) =$ *"approximate"* $\Longrightarrow A_\varepsilon(F)$ *returns a* $(1/2 + \varepsilon/(12\Delta))$*-approximate solution to* F *in* $O(|F|^2)$ *time, where* Δ *is the clause-to-variable ratio in* F*.*

That is, MAX-3-LIN-2 *with* n *variables and* Δn *equations is in*

$\bigcap_{\varepsilon>0}(\mathsf{DTIME}[2^{\varepsilon n}] \oplus_P \mathsf{APX\text{-}TIME}[\frac{1}{2} + \frac{\varepsilon}{12\Delta}, n^{O(1)}])$.

Observe that Håstad's inapproximability result for MAX-E3-LIN-2 [14] still holds when $\Delta = O(1)$.

5.1 Proof of Theorem 5.1

Unless otherwise stated, all random choices made are both uniform and independent. Let F denote a collection of k-constraints, as defined in the Preliminaries.

5.1.1 The selector S_ε

We will find a certain subset $\mathcal{S}$ of the instance F that has both a "disjointness" property and a "covering" property, in that:

– If $\mathcal{S}$ is large, "disjointness" of constraints in $\mathcal{S}$ makes it possible to satisfy at least 3/4 of the constraints in $\mathcal{S}$, as well as roughly half of the optimal number of constraints in $F - \mathcal{S}$.

– If $\mathcal{S}$ is small, "covering" ensures that any assignment to the variables appearing in $\mathcal{S}$ results in an instance so simple that it is exactly solvable in polynomial time.

More precisely, we'll find a collection of constraints within F that not only hits all other constraints at least *twice* (the "covering" property), but also contains a large subcollection in which every constraint has two variables not appearing in any other constraint of the subcollection (the "disjointness" property). For visually-inspiring reasons we will call these "swiss army knife collections".

Recall a *sunflower* is a pair $(\mathcal{C}, H)$ where $\mathcal{C}$ is a collection of sets such that for all $S_i, S_j \in \mathcal{C}$, $S_i \cap S_j = H$ (H is called the *heart*). We define a *swiss army knife* $(\mathcal{K}, B)$ to be a collection of sets $\mathcal{K}$ along with a fixed set $B = \{b_1, \ldots, b_k\}$ (the *base*) such that there exist k disjoint set collections $K_1, \ldots, K_k$ (the *blades*) where:

- $\mathcal{K} = K_1 \cup \cdots \cup K_k$,
- for all $i \neq j$, $X \in K_i$, and $Y \in K_j$, $X \cap Y = \varnothing$, and
- for all $i = 1, \ldots, k$, K_i is either a sunflower with heart $\{b_i\}$ or the empty set.

Figure 1 illustrates a swiss army knife, giving some visual insight into its name. S_ε greedily constructs a maximal disjoint collection of swiss army knives over the constraints of F. (We construe 3-constraints as 3-sets in the standard way, considering a constraint c as merely its variable set $\mathsf{vars}(c)$. Note for linear equations mod 2, at most two constraints will map to the same 3-set; in the event of such a collision, arbitrarily pick one of the constraints in the following.) First the bases of are gotten by greedily choosing a maximal disjoint collection M of 3-sets from F, then the blades are gotten by greedily adding 3-sets to M that qualify, *i.e.* the set has an intersection less than two with every existing blade and base, and an intersection of one with some base. For more details, the reader can consult the appendix.

Let V_{bases} (and V_{blades}) be the number of variables appearing in bases (blades, respectively) of the collection. If $\max\{V_{\text{bases}}, V_{\text{blades}}\} \leq \varepsilon n/2$, S_ε returns "exact", otherwise S_ε returns "approximate".

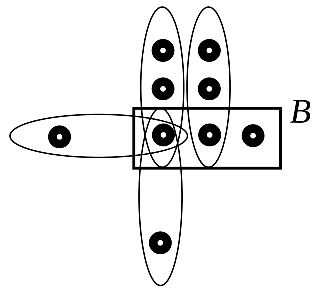

Figure 1: A swiss army knife with base B (the black donuts are elements, the ovals are blades).

5.1.2 Exact algorithm E_ε

The exact solution takes the swiss army knife collection built in S_ε, and tries all possible assignments to variables appearing in constraints of it.

Observation 5.1 *("Covering" property) The swiss army knife collection C from S_ε is such that for all $c \in F - C$, at least two variables of c appear in C.*

(Any $c \in F - C$ is hit at least once by a base of the collection, and a second time by either a blade or a base.)

For each assignment, by the observation, the remaining unassigned constraints have at most one variable each, and thus can be solved exactly in linear time. This procedure takes $\tilde{O}(2^{\varepsilon n})$ time, as there are at most εn variables in the collection (at most $\varepsilon n/2$ among the blades, and at most $\varepsilon n/2$ among the bases).

5.1.3 Approximation A_ε

The idea behind the approximation is to take a subset U of the swiss army knife collection (either all of the blades, or all of the bases) and show that a randomized algorithm can simultaneously satisfy all constraints represented in U *as well as* half the optimal number of constraints satisfiable in the rest of F. Since A_ε is only run if the knife collection is large, U is somewhat large, so this yields a non-trivial improvement in approximation.

Let U be either the set of bases in the knife collection or the set of blades, whichever is larger. Since $\max\{V_{\text{bases}}, V_{\text{blades}}\} > \varepsilon n/2$, it follows that $|U| > \varepsilon n/6$ (each set of U has at most three variables, and in the worst case the sets of U are disjoint).

Definition 5.1 *A constraint $c \in F$ is degenerate w.r.t. U if, when F is partitioned into U and $F - U$, there is c' in the partition that does not contain c such that $c' \equiv (c+1) \mod 2$.*

That is, if $c \in U$, there is $c' \in F - U$ with $c' + c \equiv 1$; if $c \in F - U$, there is $c' \in U$ with $c' + c \equiv 1$. Observe that the number of constraints degenerate w.r.t. U is even.

Claim 5.1 *Let F_{deg} be the set of constraints in F that are degenerate for U. Then every variable assignment satisfies exactly $|F_{deg}|/2$ constraints in F_{deg}.*

Proof 5.1 *Every assignment satisfies either c or $c+1$, but not both.* $\square$

Without loss of generality, we can therefore remove all degenerate-w.r.t.-U pairs of constraints from F, as $1/2$ of them will be satisfied regardless of the assignment chosen. After eliminating these degeneracies, A_ε runs a simple assignment procedure.

$\mathsf{Choose}(U, F)$: *For all $c \in U$, choose a satisfying assignment for c uniformly at random. Then set each unassigned variable in F to 1, with probability* $1/2$.

Let m^* be the maximum number of $c \in F$ that can be satisfied. Let $m_{non-deg}$ and $m^u_{non-deg}$ be the number of non-degenerate $c \in F - U$ and $c \in U$, respectively. Let m_{deg} and m^u_{deg} be the number of degenerate $c \in F - U$ and $c \in U$, respectively. Observe $m_{non-deg} + m_{deg} = m$, but

$$m_{non-deg} + \frac{m_{deg}}{2} \geq m^* \quad (1)$$

by Claim 5.1, and

$$m^u_{non-deg} + m^u_{deg} = |U| \geq \frac{\varepsilon n}{6} = \frac{\varepsilon m}{6\Delta}. \quad (2)$$

Clearly, the $m^u_{non-deg} + m^u_{deg}$ constraints in s are satisfied by Choose. The following shows that the other constraints are satisfied with probability $1/2$.

Observation 5.2 *("Disjointness" property) Every $c \in U$ has at least two variables not appearing in any other $c' \in U$.*

(Either U is the collection of bases, where the sets are disjoint, or it is the collection of blades, where each set has two variables not appearing in any other set.)

Lemma 5.1 *Suppose U has the disjointness property. Then for any non-degenerate 3-constraint $c'' \in F - U$, $\mathsf{Choose}(U, F)$ satisfies c'' with probability 1/2.*

Informally, this means that the distribution of assignments by Choose "looks random" to the constraints of $F - U$.

Proof 5.2 *Consider the following (equivalent) algorithm. Set each variable not appearing in any constraint of U randomly. For each 3-constraint $c \in U$, suppose c is $x + y + z = K$ in the following, with x and y being two variables not appearing in any other $c' \in U$. We will say that x and y is a correlated pair. Choose a random assignment for z (if one has not already been chosen), then randomly choose one of the two possible assignments to x and y that will satisfy c.*

Clearly, two correlated pairs obtained from two different constraints in U are disjoint. Thus if $\{x, y\}$ is a correlated pair, then for any variable $v \neq y$ and $v \neq x$, $Pr[x = v] = 1/2$. Moreover, for any variable $v \neq z$, $v \neq x$, $v \neq y$, the assignment to v is independent of the assignment to x and y.

Now suppose $c'' \in F - U$. The lemma is certainly true if there is no correlated pair of variables in c'', as all other variable assignments are independent. As the correlated pairs are disjoint, the 3-constraint c'' contains at most one correlated pair $\{x, y\}$ from some $c \in U$. (Note the other variable of c'' may be in some correlated pair, just not one with x or y.) Since c'' is non-degenerate, the other variable (say, v) of c'' is assigned independently of x and y (c does not contain v), hence $Pr[c'' = 1] = 1/2$ and the lemma holds. $\square$

Thus the expected number of constraints satisfied by Choose returns is $|U|$, plus *half* of the non-degenerate constraints in $F-U$ (by Lemma 5.1), plus *half* of the degenerate constraints in $F-U$ (by Claim 5.1). This quantity is $(m^u_{non-deg}+m^u_{deg})+(m_{non-deg}-m^u_{non-deg})/2+(m_{deg}-m^u_{deg})/2 = (m_{non-deg}+m_{deg})/2+(m^u_{non-deg}+m^u_{deg})/2 \geq m^*/2+\varepsilon m/12$, due to (1) and (2). The proof of Theorem 5.1 is complete. □

5.2 A more general case

Extending the algorithm to MAX-Ek-Lin-p for odd $k \geq 3$ and prime p is relatively straightforward. First, observe the degeneracy notion here still means that at most one of the constraints among a group of at most p are satisfied by any assignment, and in our case, we will satisfy at least one of them. The notion of correlated pairs is analogous.

The selector now picks sets of constraints S_k, S_{k-1}, …, S_2. Each S_i is a maximal disjoint set of k-constraints, chosen after the variables of the sets $S_k,\ldots,S_{i+1}$ were removed from consideration. (For the Swiss army knife collection, we only chose an S_3, then an S_2.) If $|\cup_{i=2}^k S_i| \leq \varepsilon n/k$, then do an exact solution as before in time $(p^k)^{\varepsilon n/k} = p^{\varepsilon n}$. Otherwise, some S_i is of size at least $\varepsilon n/(k(k-1))$. Choose now selects a random satisfying assignment over all constraints in S_i, with independent random assignments for all variables appearing in multiple constraints of S_i and variables not in S_i. Now there are $i \geq 2$ variables remaining in each equation of S_i, and a random satisfying assignment is chosen from them. We claim now that every non-degenerate k-constraint c not in S_i is satisfied with probability $1/p$. Consider just the case $i=2$; the other cases follow similarly. If c contains no correlated pairs, then trivially it is satisfied with probability $1/p$. Otherwise, because k is odd, there is at least one variable x in c whose correlated counterpart (if there is one) does not appear in c; x is thus set 0-1 uniformly and independently from the $k-1$ other variables of c. The claim follows.

Theorem 5.2 *Let $k \geq 3$ be odd.* MAX-k-LIN-p *instances with n variables and Δn equations is in* $\bigcap_{\varepsilon>0}(\mathsf{DTIME}[p^{\varepsilon n}] \oplus_P \mathsf{APX\text{-}TIME}[\frac{1}{p}+\frac{\varepsilon}{pk(k-1)\Delta}, n^{O(1)}])$.

5.3 An application: "hardness-defying" algorithm selection

Note $1/\varepsilon$ in the above may be replaced by any polytime computable $f(n,m)$ such that $f(n,m) \geq 1/6$ (so that $1/2+1/[12f(n,m)] \leq 1$ and $|S| \geq \frac{n}{6f(n,m)}$ makes sense). Håstad and Venkatesh [15] proved strong inapproximability bounds for MAX-3-LIN-2. While they showed that MAX-3-LIN-2 is in $\mathsf{APX\text{-}TIME}[\frac{1}{2}+\frac{O(1)}{\sqrt{m}}, m]$, they also showed (rewritten in our notation and notion of approximation):

Theorem 5.3 *[15] For $\varepsilon > 0$,*

$$\mathsf{NP} \not\subseteq \mathsf{TIME}[2^{(\log m)^{O(1)}}] \implies \text{MAX-3-LIN-2} \notin \mathsf{APX\text{-}TIME}[\tfrac{1}{2}+\tfrac{1}{2^{(\log m)^{1-\varepsilon}}}, 2^{(\log m)^{O(1)}}].$$

However, by Theorem 5.1 we may (for example) either $(\frac{1}{2}+\frac{1}{12\cdot 2^{(\log m)^{1/2}}})$-approximate in polynomial time, or solve exactly in $2^{m/2^{(\log m)^{1/2}}}$ time. Thus their result can, in a sense, be (subexponentially) side-stepped.

6 Optimization with three-variable Boolean constraints

The previous algorithm can actually be extended to a variety of optimization problems, which we outline here. Let f be a Boolean function on k variables, and X be the set of all $2n$ literals on n Boolean variables. An *f-formula* $\mathcal{C}$ is a collection of k-tuples from X^k. Each k-tuple c is called a *constraint*, and a constraint c is *satisfied* by an assignment to X if the assignment makes $f(c)$ true. The MAX-k-f problem gives an f-formula, and the goal is to find a variable assignment that satisfies a maximum number of constraints.

Definition 6.1 *Let $\mathcal{C}$ be an f-formula. $\mathcal{C}$ has overlap if two clauses in $\mathcal{C}$ have exactly the same variables.*

E.g., the CNF formula $(x \vee y) \wedge (\overline{x} \vee y)$ has overlap, but $(x \vee y) \wedge (\overline{x} \vee z)$ does not.

Similar to [12], let t be the number of assignments satisfying the 3-variable Boolean function f, and b be the number of satisfying assignments for f with odd parity. Say f is *biased* if $2b \neq t$. Notice that OR, AND, MAJORITY, XOR, *etc.* are all biased. We will give a general hybrid strategy for optimization problems defined with respect to some biased function f, with three variables per constraint and no overlap. The main tool employed is an easy-to-verify lemma.

Lemma 6.1 *Let x, y, z be Boolean variables, and $\varepsilon \in [0, 1/4]$. Consider the procedure that picks either equation $x + y + z = 1$ with probability $1 - \varepsilon$, or equation $x + y + z = 0$ with probability ε, then a random satisfying assignment for the equation picked. The resulting distribution is pairwise independent over $\{x, y, z\}$.*

Theorem 6.1 *Let k be odd, and f be a k-variable biased Boolean function, with random assignment threshold α. MAX-k-f instances on n variables and Δn non-overlapping constraints are solvable in $\bigcap_{\varepsilon>0}(\mathsf{DTIME}[2^{\varepsilon n}] \oplus_P \mathsf{APX\text{-}TIME}[\alpha + \frac{\varepsilon}{pk(k-1)\Delta}, n^{O(1)}])$.*

Proof 6.1 *(Sketch) Similar to the proof of Theorem 5.2, choose maximal disjoint sets S_k, S_{k-1}, $\ldots$, S_2. If all of the sets are small, exactly solve, otherwise let S_i be the large set. Let $c \in S_i$ be a constraint on variables x, y, and z. Since c is biased, one of the equations $x + y + z = 1$ or $x + y + z = 0$ has the property that the number of its satisfying assignments that also satisfy c is strictly greater than the corresponding number for the other equation. Set $\varepsilon > 0$ such that the following procedure satisfies at least $1 - \varepsilon/2$ constraints in S_i:* For each $c \in S_i$, let x, y, z be its variables. With probability ε, pick a random satisfying assignment of $x + y + z = 0$; otherwise pick a random falsifying assignment. *Such an $\varepsilon > 0$ exists, by Lemma 6.1. A similar analysis to Theorem 5.1 yields the result.* □

It may appear at first that forbidding overlap is a severe restriction, and therefore the hybrid algorithm is not as surprising in this case. Nevertheless, one can show that approximating instances without overlap is still difficult. We focus on the case of MAX-E3-SAT; in our opinion it is most convenient to formulate.

Theorem 6.2 *If* MAX-E3-SAT *instances without overlap can be approximated within a $7/8 + \varepsilon$ factor in polynomial time, then* $\mathsf{RP} = \mathsf{NP}$.

The proof uses two applications of a Chernoff bound and is deferred to the appendix.

7 Counting the fraction of solutions to a 2-CNF formula

It is not known if there is a polytime approximation with additive error $1/2^{f(n)}$ for counting the fraction of 2-SAT solutions for any $f(n) \in o(n)$, though it is possible to do so in subexponential time (that is, $2^{O(f(n))}$) [17]. A hybrid approach gives a quick partial result in this direction.

Theorem 7.1 *For any $\varepsilon > 0$ and $f(n) \in o(n)$, there is a hybrid algorithm for the fraction of satisfying assignments of 2-SAT, that gives either the* exact *fraction in $\tilde{O}(2^{\varepsilon n})$ time, or counts within additive error at most $1/2^{f(n)}$ in polynomial time.*

Proof 7.1 *Choose a maximal independent set M over the 2-CNF clauses (all clauses in M are disjoint). If $|M| \leq \log_2(3)\varepsilon n$, then try all $3^{\log_2(3)\varepsilon n} = 2^{\varepsilon n}$ satisfying assignments of M for the exact fraction. Otherwise at most a $(3/4)^{\log_2(3)\varepsilon n}$ fraction of the assignments satisfy the formula. If $(3/4)^{\log_2(3)\varepsilon n} > 1/2^{f(n)}$, then n is bounded from above by a constant, and exact solution takes $O(1)$ time. Otherwise, output $(3/4)^{\log_2(3)\varepsilon n}$ as an approximate fraction within error $1/2^{f(n)}$.* $\square$

8 Solving certain hard quantified Boolean formulas (QBF)

We do not intend hybrid algorithms to be restricted solely to exact/approximate tradeoffs, but this pair of measures has certainly helped us develop our ideas. We now briefly turn to a pair more motivated by complexity theoretic interests than practical considerations. In terms of quantified Boolean formulas, the notion of approximation is a little less coherent. Rather than pitting efficient approximation versus exact solution, our aim is to show that QBFs either admit *faster-than-*2^n algorithms, or they are solvable in alternating linear time with a relatively *small* number of alternations. (For background on alternation, cf. [26].) Let EQBF be the set of true quantified Boolean formulas in prefix-normal form over arbitrary propositional predicates, satisfying the following regularity condition on quantification:

- If the formula has n variables and has a alternations, then every quantifier block contains exactly $\lfloor n/a \rfloor$ variables, except for the last block which contains (n mod a) variables.

Proposition 8.1 EQBF *is* PSPACE*-complete.*

We will show that for all $\varepsilon > 0$, every instance of EQBF can be solved either in (essentially) $O(2^{(1-\varepsilon/2)n})$ expected worst case time, or in alternating linear time with few (εn) alternations. The test deciding which case happens simply checks the number of alternations. Below, $\mathsf{ZPTIME}[t]$ is the class of decision problems solvable in worst-case *expected* time t, and $\mathsf{\Sigma_k} - \mathsf{TIME}[t]$ is the class solvable by alternating TMs using at most k alternations (starting in an existential state).

Theorem 8.1 *For all $\varepsilon > 0$,*

$$\text{EQBF } \textit{for } n\textit{-variable instances is in } \mathsf{ZPTIME}[2^{\left(1-\varepsilon/2+O(1/2^{1/\varepsilon})\right)n}] \oplus_P \Sigma_{\varepsilon n}\text{--}\mathsf{TIME}[n^{O(1)}].$$

Observe that *no better algorithm than the trivial 2^n one* is currently known for EQBF, or any interesting variant of it. Similarly, it not known (or believed) that one can quickly reduce a QBF F on an arbitrary predicate down to an F' where the number of alternations in F' is a small fraction of the number of alternations in F. However, one can neatly partition EQBF into "lower alternation" cases and "faster runtime" cases. The construction not only holds for constant $\varepsilon > 0$ but for any decreasing function f with values in the interval $(0, 1]$, so *e.g.* one gets either $2^{n-n/\log n}$ expected time or $(n/\log n)$-alternating linear time for EQBF (these particular values are interesting, as the best known randomized algorithm for CNF satisfiability has such runtime [29]). Theorem 8.1 holds due to a generalization of probabilistic game-tree search [30]. We defer the algorithm and proof to the appendix.

Theorem 8.2 *EQBF with k alternations are solvable in expected $O\left(\binom{2^k+1}{2}^{\frac{n}{2k}}\right)$ worst-case time.*

Remark 8.1 *For $k = 1$, one obtains $\binom{3}{2}^{n/2} = 3^{n/2}$ runtime. Observe $\binom{2^k+1}{2}^{\frac{1}{2k}}$ increases as k increases. Therefore: the* fewer *the alternations in the formula, the* greater *the runtime bound.*

Proof of Theorem 8.1: (Sketch) Initially, $\mathcal{B}$ chooses two uniform random permutations of $[2k]$. Call the two sequences $u_1, \ldots, u_{2^k}$, $e_1, \ldots, e_{2^k}$. Note each u_i and e_i may be seen as both k-bit strings and as integers from $[2^k]$; we will use both interpretations in what follows.

Suppose the first variable x_1 is universal (the case where x_1 is existential is analogous). The k bits in u_1 are substituted for the k variables in the quantifier block containing x_i, then the bit string e_1 is substituted for the k variables in the subsequent existential block.

(*) $\mathcal{B}$ is then recursively called on the remaining formula.

• If the call returns false, e_2 is substituted instead of e_1, and $\mathcal{B}$ is called again as in (*). If that fails, e_3 is substituted for e_2 and $\mathcal{B}$ is called, etc., until either (1) an e_i is found that yields true, or (2) e_{2^k} failed.

– If (1), the next k values for the universal variables (u_2) is substituted for u_1, a new random permutation $e_1, \ldots, e_{2^k}$ is chosen, e_1 is substituted for the existential block following the universal block of x_i, and B is recursively called as in (*).

– If (2), $\mathcal{B}$ concludes the formula is false.

• If the call returns true and u_i is the current set of k values for the universal variables, then u_{i+1} is substituted in place of u_i, a new random permutation $e_1, \ldots, e_{2^k}$ is chosen, e_1 is substituted for the existentials, and the process restarts from (*). If $i = 2^k$, $\mathcal{B}$ concludes the formula is true. This concludes the description of $\mathcal{B}$.

$\mathcal{B}$ is indeed a backtracking algorithm simply making randomized assignment choices in a certain manner, thus it always returns the correct answer. It remains to show its expected runtime is the claimed quantity. Clearly, two quantifier blocks are assigned before each recursive call, so the recursion depth is $n/2k$.

Suppose the formula given to $\mathcal{B}$ is false. Then $\mathcal{B}$ guesses a k-bit string v for the universal variables (such that every setting of the existentials yield false) with probability $1/2^k$. (That is, $u_1 = v$ with probability $1/2^k$.) If this fails, $u_2 = v$ with probability $1/2^k$, and so on. Every time that the wrong u_i is chosen, $\mathcal{B}$ must consider all 2^k settings of the existential variables. Hence the expected number of recursive calls at any point (of the recursion tree) is at most

$$2^k \cdot \left(\frac{1}{2^k} + 2 \cdot \frac{2^k-1}{2^k}\frac{1}{2^k-1} + 3 \cdot \frac{2^k-1}{2^k}\frac{2^k-2}{2^k-1}\frac{1}{2^k-2} + \quad \cdots \quad + 2^k \cdot \frac{2^k-1}{2^k}\frac{2^k-2}{2^k-1}\cdots\frac{1}{2}\right)$$

$$= 2^k \cdot \left(\frac{1}{2^k} + \frac{2}{2^k} + \frac{3}{2^k} + \frac{2^k}{2^k}\right) = \sum_{i=1}^{2^k} i = \binom{2^k+1}{2}.$$

Similarly, if the formula is true, all u_i settings are considered. For each one, the guessed e_1 is the correct existential setting with probability $1/2^k$; if this fails, e_2 is the correct one with probability $1/2^k$, etc. The expected number of recursive calls is therefore represented by the same expression above. $\square$

Letting $\varepsilon = 1/k$, the above runtime bound for EQBF with greater than εn alternations is at most $\binom{2^{1/\varepsilon}+1}{2}^{\frac{\varepsilon n}{2}} = (2^{1/\varepsilon}+1)^{\varepsilon n/2} 2^{\frac{n}{2}-\frac{\varepsilon n}{2}} \le 2^{n/2(1+\frac{\varepsilon}{2^{1/\varepsilon}})} 2^{n/2} 2^{-\frac{\varepsilon n}{2}} = 2^{n\left(1-\frac{\varepsilon}{2}+O(\frac{1}{2^{1/\varepsilon}})\right)}$, where the inequality holds by a small lemma.

Lemma 8.1 *For all* $\varepsilon \in (0,1]$, $(2^{1/\varepsilon}+1)^{\varepsilon n/2} \le 2^{n/2(1+\frac{\varepsilon}{2^{1/\varepsilon}})}$.

Proof 8.1 $\frac{(2^{1/\varepsilon}+1)^{\varepsilon n/2}}{2^{n/2}} = (\frac{2^{1/\varepsilon}+1}{2^{1/\varepsilon}})^{\varepsilon n/2} = (1+\frac{1}{2^{1/\varepsilon}})^{\varepsilon n/2}$, *thus* $(2^{1/\varepsilon}+1)^{\varepsilon n/2} = 2^{n/2}(1+\frac{1}{2^{1/\varepsilon}})^{\varepsilon n/2}$.

By the well-known inequality $\log_2(1+x) \le x$,

$2^{n/2}(1+\frac{1}{2^{1/\varepsilon}})^{\varepsilon n/2} = 2^{n/2} 2^{\log_2\left(1+\frac{1}{2^{1/\varepsilon}}\right)\varepsilon n/2} \le 2^{n/2} 2^{\varepsilon n/2^{1+1/\varepsilon}} = 2^{n/2(1+\frac{\varepsilon}{2^{1/\varepsilon}})}$. $\square$

9 A few hardness results

Several dimensions are available for exploring the possibility of these hybrid algorithms. One can inquire about the complexity of the two classes required given a polynomial time selector, or one can consider hardness in terms of the complexity of the algorithm selector. The former kind we will call "class-based hardness", and the latter kind we will call "selector-based hardness". Here we will focus on algorithms of the exact/approximate variety.

In terms of class-based hardness, we unfortunately have little to say at the moment (that has not already been said in some other way). It would be nice to have reasonable conditions that imply something like $\mathsf{MAXSNP} \subseteq \mathsf{SUBEXP} \oplus_P \mathsf{PTAS}$ is unlikely; *i.e.* one cannot efficiently select between $(1+\varepsilon)$-approximation or subexponential exact solution. Let the Exponential Time Hypothesis for SAT, a.k.a. *ETH for SAT* be the assumption that *SAT* on inputs of size L requires $2^{\delta L}$ time for some $\delta > 0$, almost everywhere.

Proposition 9.1 ETH for SAT *implies* MAX-Ek-LIN-2 *is* not *in* $\mathsf{APX\text{-}TIME}[1/2+\varepsilon, 2^{n^{o(1)}}]$, *for some* $\varepsilon > 0$.

Proof 9.1 *Follows from Håstad's well-known inapproximability results [14]. There, a reduction from solving SAT on n variables to approximating* MAX-Ek-LIN-2 *on* $n^{O(f(\varepsilon))}$ *variables within* $1/2+\varepsilon$ *is given (for some large function* f*).* $\square$

Corollary 9.1 ETH for SAT *implies* MAX-Ek-LIN-2 *is* not *in* $\mathsf{APX\text{-}TIME}[1/2+\varepsilon, n^{O(1)}] \oplus_P \mathsf{TIME}[2^{n^{o(1)}}]$, *for some* $\varepsilon > 0$.

Thus we cannot improve the MAX-Ek-LIN-2 algorithm significantly (from exact time $2^{\varepsilon n}$ to $2^{n^{o(1)}}$), assuming the exponential-time hypothesis. Note the same results extend to MAX-Ek-SAT without overlap, MAX-Ek-AND without overlap, *etc.*, with appropriate substitutions for the fraction 1/2 in the above.

The MAX-E3-LIN-2 algorithm itself immediately implies that a "linear" many-one *subexponential time* reduction from SAT to MAX-E3-LIN-2 $(1/2+\varepsilon)$-approximation in fact *does not exist*, if ETH for SAT holds. Contrast this with the facts:

1) there is a many-one *polynomial time* reduction from SAT to MAX-E3-LIN-2 $(1/2+\varepsilon)$-approximation (but the reduction blows up the number of clauses by a large polynomial, whereas we assume only a linear increase), and

2) there is a subexponential time *Turing* reduction from k-SAT to MAX-3-LIN-2 (but the Turing reduction needs $2^{\varepsilon n}$ oracle calls).

This result may have some relevance to the question of whether linear length PCPs with perfect completeness exist, as PCPs of linear length along with the ETH for SAT would imply for some constant c that c-approximating MAX-SAT requires $2^{\Omega(n)}$ time [2]. (Of course, here it is important to note that our result only holds for SAT where the number of clauses is linearly related to the number of variables, so we have not shown completely that some consequence of 'perfect' linear length PCPs indeed holds.)

Theorem 9.1 *For all $k' \geq 3$,* ETH for SAT *implies that for all constants $c > 1$ and $\Delta > 1$,* any *many-one reduction from* SAT *on $m = \Delta n$ clauses and n variables to* MAX-E3-LIN-2 $(1/2+\varepsilon)$-*approximation on (at most) cm clauses and (at most) cn variables* requires $2^{\varepsilon m}$ *time for some $\varepsilon > 0$.*

Proof 9.2 *If such a reduction existed running in time $2^{\varepsilon m}$ for all $\varepsilon > 0$, then choose $\varepsilon < \min\{\frac{\delta}{12}, \frac{c\cdot\delta}{12\Delta}\}$, where δ is such that SAT-solving requires $O(2^{\delta n})$ steps. Reduce a given SAT formula into a MAX-E3-LIN-2 instance, and letting $\varepsilon' = 12\varepsilon$, run the selector $C_{\varepsilon'}$ from Theorem 5.1. If $S_{\varepsilon'}$ says "exact", then solving the SAT instance takes less than $2^{\varepsilon' n} < 2^{\delta n}$ time. If $S_{\varepsilon'}$ says "approximate", the approximation algorithm gives a fast solution within $1/2 + \varepsilon'/12 = 1/2 + \varepsilon$ of the optimum, which is enough to decide the SAT instance by assumption.* □

9.1 Selector-based hardness

We can also show some simple requirements on the complexity of selectors for certain hard problems under certain measures. Intuitively, our objective here is to prove that in the exact/approximation case, an efficient selector cannot be heavily biased towards one type of solution over the other. We have a couple of results along these lines, using the following assumption.

Assumption 1 *Π is a MAXSNP-complete problem in $\mathsf{TIME}[t_1] \oplus_P \mathsf{APX}-\mathsf{TIME}[\alpha+\varepsilon, t_2]$ for some time constructible t_1 and t_2, where α is an inapproximability ratio for Π. (That is, $\Pi \notin \mathsf{APX}-\mathsf{TIME}[\alpha+\varepsilon, n^{O(1)}]$ unless $\mathsf{P} = \mathsf{NP}$.) Define $\mathcal{A} \subseteq \Pi$ to be the subset of instances $(\alpha+\varepsilon)$-approximated in t_2, and $\mathcal{E} \subseteq \Pi$ be the set solved exactly in t_1 (so $\{\mathcal{A}, \mathcal{E}\}$ is a partition of Π).*

Let $m \in \mathbb{N}$, and Π_m be the class of instances on inputs of size m; more precisely, we measure size by the *number of constraints*. Say that a set $S \subseteq \Pi$ is $f(m)$-*sparse* if there is a $k \in \mathbb{N}$ such that $|S \cap \Pi_m| \leq f(m)$, and $f(m)$-*dense* if $|S \cap \Pi_m| \geq f(m)$. Assuming the p-dimension of NP is greater than zero (a working conjecture with numerous plausible consequences, [25]) and the exact solution runs in only $2^{n^{o(1)}}$ time, there must be a dense set of instances being approximated.

Theorem 9.2 *Given Assumption 1, if $t_1 \in 2^{n^{o(1)}}$ then $\mathcal{A}$ is 2^{n^δ}-dense for some $\delta > 0$, unless $dim_p(NP) = 0$. (cf. [25] for definition).*

Proof 9.3 *The following is proved in Hitchcock [18]:*

If $dim_p(NP) > 0$, then for all $\varepsilon > 0$ there exists a $\delta, \delta' > 0$ such that any 2^{n^δ}-time approximation algorithm for MAX-3-SAT has performance ratio less than $7/8 + \varepsilon$ on a $2^{n^{\delta'}}$-dense set of satisfiable instances.

One can, in a straightforward way, adapt his proof to any Π *in* MAX-SNP, *where if* α *is the inapproximability ratio for* Π, *then* α *takes the place of* $7/8$ *in the above.* $\Box$

Also, if a selector only employs exact solution on a sufficiently sparse set, then we can effectively remove the selector entirely.

Definition 9.1 Π admits arbitrarily large constraints of constant size *iff there exists a* $K \geq 1$ *such that for all* $k \geq K$ *and instances* x *of* Π, *adding a constraint* c *on* k *variables to* x *still results in an instance of* Π.

Theorem 9.3 *Given Assumption 1, if* Π *admits arbitrarily large constraints of constant size,* $\mathcal{E}$ *is* $m^{O(1)}$*-sparse, and* $t_2 \in O(n^{O(1)})$, *then* $\mathsf{P} = \mathsf{NP}$.

Proof 9.4 *We will approximate* Π *within the ratio* $\alpha + \varepsilon$. *Let* S_ε *be the selector and* A_ε *be the approximation algorithm existing due to Assumption 1. For any* $x \in \Pi$, *if* $S_\varepsilon(x)$ *says to exactly solve, we will local search for an approximately good solution. Suppose* $\mathcal{E}$ *is* m^k*-sparse. Construe* $x \in \Pi$ *(recall* Π *is* MAXSNP*-complete) as a set of* constraints, *and the optimization problem is to satisfy a maximum number. Let* $\mathbf{S}_{k+1}$ *be the collection of all* $(k+1)$*-sets of constraints not in* x, *over the variables of* x. *For all* k*-sets* $y \subseteq x$, *and* $y' \in \mathbf{S}_{k+1}$, *consider* $x' = (x - y) \cup y'$. x' *is still an instance of* Π *since it admits arbitrarily large constraints of constant size. Since* $\mathcal{E}$ *is sparse, there must be an* y *and* y' *such that* $S_\varepsilon(x')$ *says to approximate. Suppose the optimal value for* x *is* m^*, *and thus on* $x \Delta y'$ *it is at least* $(m^* - k)$. *But then* $7/8(m^* - k) + \varepsilon(m^* - k)$ *clauses are satisfied, i.e.* $(7/8 + \varepsilon)m^* - O(1)$ *clauses, so* $\mathsf{P} = \mathsf{NP}$. $\Box$

In general, if $\mathcal{E}$ is $m^{f(m)}$-sparse then Π is in APX-TIME$[m^{f(m)} + t_2,\ \alpha + \varepsilon - \frac{f(m)}{m}]$.

10 Conclusions

In a very real sense, the areas within theoretical computer science today are characterized by the different methodologies and measures that researchers use to analyze and attack problems. We have introduced a theory of algorithm selection as a possible means of unifying such (worst-case) measures, and gaining a better understanding of how these measures relate. To reflect our poor intuitions concerning this relationship, this study revealed several counter-intuitive results, some of which are not only theoretically interesting but also practically so. There are myriad directions for further work; here are some that we consider most promising.

- Improve the tradeoffs for the given algorithms, if possible. One problem with our exact/approximate algorithms is that the exact cases are somewhat trivial. They probably are improvable by more careful analysis of (for example) the Swiss army knife construction; our goal was to merely demonstrate and motivate these algorithms' possibility.

- Prove structural properties of the p-selection operator. Can interesting characterizations of common complexity classes be found using this kind of operator? It seems likely, given our preliminary findings.
- Extend the ideas here to other problems (*e.g.* MAX-3-SAT). The current techniques of the paper might be enough to get an algorithm for MAX-E3-SAT without the "no overlap" requirement; however, our prolonged efforts have failed thus far.
- The hardness results given are at a preliminary stage. It might be productive to focus on problems that are *very* hard to approximate, such as Max Independent Set. It seems unlikely to us, for example, that Max Independent Set is in the p-selection of $2^{\varepsilon n}$ time and efficient $(1+\varepsilon)$-approximation.

11 Acknowledgements

This paper greatly benefited from the input of many individuals. Special thanks go to the author's advisor Manuel Blum for encouragement, and to Maverick Woo, Hubie Chen, and especially Virginia Vassilevska, for helpful discussions and comments.

References

[1] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM* 41: 153–180, 1994.

[2] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of Proximity, Shorter PCPs and Applications to Coding. To appear in *STOC*, 2004.

[3] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.* 13:850–864, 1984.

[4] H. L. Bodlaender, D. M. Thilikos, and K. Yamazaki. It is Hard to Know when Greedy is Good for Finding Independent Sets. *Inf. Process. Lett.* 61(2):101–111, 1997.

[5] C. E. Brodley. Addressing the Selective Superiority Problem: Automatic Algorithm / Model Class Selection. *Proc. International Conference on Machine Learning*, 17–24, 1993.

[6] Liming Cai and David W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.* 67(4):789–807, 2003.

[7] E. Dantsin, M. Gavrilovich, E. A. Hirsch, B. Konev. MAX SAT approximation beyond the limits of polynomial-time approximation. *Annals of Pure and Appl. Logic* 113(1-3), 81–94, 2001.

[8] R. Downey and M. Fellows. *Parameterized Complexity.* Springer-Verlag NY, 1999.

[9] Wayne R. Dyksen and Carl R. Gritter, Scientific computing and the algorithm selection problem. *Expert Systems for Scientific Computing*, North-Holland, 19–31, 1992.

[10] P. Erdös and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.* 35:85–90, 1960.

[11] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *STOC*, 1996.

[12] Uriel Feige. Relations between average case complexity and approximation complexity. *STOC*, 534–543, 2002.

[13] Carla Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence* 126(1-2): 43–62, 2001.

[14] Johan Håstad. Some optimal inapproximability results. *J. ACM* 48:798–859, 2001.

[15] Johan Håstad and S. Venkatesh. On the advantage over a random assignment. *STOC*, 43–52, 2002.

[16] E. A. Hirsch. Worst-case study of local search for MAX-k-SAT. *Discrete Applied Mathematics* 130(2):173–184, 2003.

[17] E. A. Hirsch. A Fast Deterministic Algorithm for Formulas That Have Many Satisfying Assignments. *Logic Journal of the IGPL* 6(1):59–71, 1998.

[18] J. M. Hitchcock. MAX3SAT is exponentially hard to approximate if NP has positive dimension. *Theor. Comput. Sci.* 289(1):861–869, 2002.

[19] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and M. Chickering. A Bayesian approach to tackling hard computational problems. *Proc. Uncertainty in Artificial Intelligence*, 2001.

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

[21] Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? *FOCS*, 406–415, 1997.

[22] Ming-Yang Kao, Yuan Ma, Michael Sipser, and Yiqun Yin. Optimal Constructions of Hybrid Algorithms. *J. Algorithms* 29:142–164, 1998.

[23] M. G. Lagoudakis and M. L. Littman. Reinforcement Learning for Algorithm Selection. In *International Conference on Machine Learning*, 2000.

[24] Kevin Leyton-Brown and Eugene Nudelman and Galen Andrew and Jim McFadden and Yoav Shoham. Boosting as a Metaphor for Algorithm Design. *Proc. Principles and Practice of Constraint Programming*, 899-903, 2003.

[25] J. H. Lutz and E. Mayordomo. Twelve problems in resource-bounded measure. *Bulletin of the European Assoc. for Theor. Comp. Sci.* 68:64–80, 1999.

[26] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[27] J. R. Rice. The algorithm selection problem. Advances in Computers, 15:65–118, 1976.

[28] Alan Selman. P-selective Sets, Tally Languages, and the Behavior of Polynomial Time Reducibilities on NP. *Math. Sys. Theory* 13:55–65, 1979.

[29] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. Accepted in *J. Algorithms*, 2003.

[30] Marc Snir. Lower Bounds on Probabilistic Linear Decision Trees. *Theor. Comput. Sci.* 38:69–82, 1985.

[31] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. *STOC*, 453–461, 2001.

[32] A. C. Yao. Theory and applications of trapdoor functions. *FOCS*, 80–91, 1982.

12 Appendix: Proof of Lemma 4.1

We will employ the following Chernoff bound in our argument.

Fact 1 *Let $X_1, \ldots, X_n$ be independent Boolean-valued random variables, with $Pr[X_i = 1] = p$. Then for $\varepsilon \in (0, 1/4)$, $Pr[\sum_i X_i \geq (p+\varepsilon)n] \leq e^{-2\varepsilon^2 n}$.*

We will give an algorithm $\mathcal{A}$ that, on instance F, runs in $t(n/\varepsilon^2)$ time and returns $f \in [0,1]$ that is within ε of the maximum fraction of equations satisfiable in F. Querying $\mathcal{A}$ via substitution of variables for values in F allows one to produce an assignment satisfying this fraction with $O(poly(n))$ calls to $\mathcal{A}$.

Given an instance F, $\mathcal{A}$ chooses a random sample S (independent and uniform) of equations in F, with $|S| = n/\varepsilon^2$, and runs the exact algorithm on S. Let $a \in \{0,1\}^n$ be an assignment to F, and suppose a satisfies a fraction f_a of equations in F. The probability that a satisfies more than $f_a + \varepsilon$ equations in S is, by Fact 1, at most $e^{-2\varepsilon^2(n/\varepsilon^2)} = e^{-2n}$; by symmetry, less than $f_a - \varepsilon$ equations in S are satisfied with this probability. A union bound over all 2^n assignments shows that S has a maximum satisfiable fraction of clauses within ε of the optimum for F with probability at least $1 - 1/c^n$ for some $c > 1$. □

13 Appendix: Greedily choosing a maximal disjoint collection of Swiss army knifes

Let F be the collection of k-constraints. Lexicographically order the constraints of F by the indices appearing in each one, making a list L. Pick a maximal disjoint set S of constraints in L, described as follows. Initially, $S = \varnothing$. Consider each constraint c in order, starting with the first; if for all $c' \in S$ we have that $\mathsf{vars}(c) \cap \mathsf{vars}(c') = \varnothing$, then set $S := S \cup \{c\}$.

Now remove the constraints in S from L. For each constraint c remaining in L, remove any variable x in $\mathsf{vars}(S) \cap \mathsf{vars}(c)$ from c. What is left is a list L' of constraints with at most two variables each, with some constraints possibly repeated. Pick a maximal disjoint set T of constraints from L', in a manner analogous to the above. The collection of Swiss army knifes is given with S as set of the bases, and the blades of each $B \in S$ are those $C \in T$ that intersect B.

14 Appendix: Proof of Theorem 6.2

Proof 14.1 *Fix $\varepsilon > 0$. Let F be a E3-CNF formula on n variables $x_1, \ldots, x_n$, with m clauses. For all $i = 1, \ldots, n$, a literal on the variable x_i will be denoted by the variable $l_i \in \{x_i, \overline{x_i}\}$. Put*

$c = (n^2 m)/9$ and $L = n^{2/3}$. We will randomly build a new formula F' with $L \cdot n = n^{5/3}$ variables and c clauses, such that: F' has no clauses sharing exactly the same three variables, if F is satisfiable then F' is satisfiable, and if no more than $(7/8+\varepsilon)m$ clauses of F can be satisfied at once, then no more than $(7/8+\varepsilon)c$ clauses of F' can be satisfied.

Our reduction bears resemblance to inapproximability results of Trevisan [31]. For each variable x_i in F, we have L variables $x_i^1, x_i^2, \ldots, x_i^l$ in a formula F''. For each clause $\{l_i \vee l_j \vee l_k\}$ in F, add the $[L]^3$ clauses $\{l_i^r \vee l_j^s \vee l_k^t\}$ in F'', for all $(r,s,t) \in [L]^3$. Now, randomly sample c clauses from F''. If any two clauses have exactly the same variables, remove them from F''. Output the remaining collection as F'.

It is clear that, if F is satisfiable then F' is satisfiable. By construction, F' has no clauses sharing exactly the same variables. We will first show that the number of clauses removed from F'' due to this vanishes to zero asymptotically, in which case the removal of these clauses does not affect the ratio of satisfied clauses. Let $(r,s,t) \in [L]^3$, and the indicator variable $X_I^{r,i,s,j,t,k}$ to be 1 iff the Ith clause chosen in the sample ($I = 1,...,c$) has variables x_i^r,x_j^s, and x_k^t. The number of clauses in F' with the variables x_i^r,x_j^s, and x_k^t is at most 8 (the same number as F), whereas the total number of clauses is $L^3 m$. Thus $Pr[X_I^{r,i,s,j,t,k}] \leq \frac{8}{mL^3}$. By a standard Chernoff bound (Fact 1), the probability that a clause with x_i^r,x_j^s, and x_k^t is chosen more than once in c trials is $Pr[\sum_{i=1}^{c} X_I^{r,i,s,j,t,k} \geq 2] \leq \exp(-2\varepsilon^2 c)$, assuming $2 \geq (1+\varepsilon)8c/(mL^3) = (1+\varepsilon)8/9$ (note $\varepsilon \leq 1/8$). Thus the expected number of clauses occurring more than once in the sample of c is, by a union bound, at most $L^3 m/\exp(\varepsilon^2 c) = 9c/\exp(\varepsilon^2 c) \in o(1)$.

Now we show that if no assignment to variables of F satisfies $(7/8+\varepsilon)m$ clauses of it, then no assignment to variables of F' satisfies $(7/8+\varepsilon+\varepsilon')c$ clauses of it for all constant $\varepsilon' > 0$, with high probability. Suppose αm clauses are satisfied by the original F. Then an α fraction of the clauses in F'' (i.e. F' prior to clause sampling) are satisfied by a. Let a be one of the $2^{n \cdot L}$ variable assignments to F''. When a clause is picked from F' at random, it has probability α of being satisfied by a. By the same Chernoff bound, the probability that more than $(\alpha+\varepsilon)c$ clauses of F' are satisfied by a is at most $\exp(-2\varepsilon^2 c)$. Assuming at most αm clauses are satisfied by any assignment to F, a union bound implies that the probability that any assignment satisfies more than $(\alpha+\varepsilon)c$ clauses of F' is $2^{n \cdot L}/e^{-2\varepsilon^2 c} < 1/d^n$ for some $d > 1$. □