Space-Time Tradeoffs for Graph Properties

Yevgeniy Dodis[1] and Sanjeev Khanna[2]

[1] Laboratory for Computer Science, MIT, USA. E-mail: yevgen@theory.lcs.mit.edu
[2] Department of Fundamental Mathematics Research, Bell Labs, USA. E-mail: sanjeev@research.bell-labs.com.

Abstract. We initiate a study of space-time tradeoffs in the cell-probe model under restricted preprocessing power. Classically, space-time tradeoffs have been studied in this model under the assumption that the preprocessing is unrestricted. In this setting, a large gap exists between the best known upper and lower bounds. Augmenting the model with a function family F that characterizes the preprocessing power, makes for a more realistic computational model and allows to obtain much tighter space-time tradeoffs for various natural settings of F. The extreme settings of our model reduce to the classical cell probe and generalized decision tree complexities.

We use graph properties for the purpose of illustrating various aspects of our model across this broad spectrum. In doing so, we develop new lower bound techniques and strengthen some existing results. In particular, we obtain near-optimal space-time tradeoffs for various natural choices of F; strengthen the Rivest-Vuillemin proof of the famous AKR conjecture to show that no non-trivial monotone graph property can be expressed as a polynomial of sub-quadratic degree; and obtain new results on the generalized decision tree complexity w.r.t. various families F.

1 Introduction

It is well known that preprocessing of static data often significantly speeds up the time it takes to answer dynamic queries about it. A data structure with appropriate auxiliary information about the static data can facilitate efficient answering of dynamic queries. Naturally, the more space is available for building such a data structure, the more auxiliary information one can precompute and the faster the queries can be answered. A natural and important question in this context is to characterize the tradeoff between the amount of available preprocessing space and the time it takes to answer the queries.

1.1 The Classical Cell Probe Model

A widely used computational model for studying such tradeoffs is the *cell probe model* introduced by Yao [17]. The *static data structure* problem in the cell probe model is as follows. We are given a function $f : Y \times Q \mapsto \{0,1\}$, where the first input $y \in Y$ ($|y| = m$) is static, and the second input $q \in Q$ ($|q| = n$) is the dynamic query (typically $n \ll m$, we assume so from now). We are also

given a parameter s which indicates the amount of space available for storing a data structure $D = \{c_1, c_2, \ldots, c_s\}$ containing information about y, where $c_i : Y \mapsto \{0,1\}$. Each c_i is called a *cell* and the process of generating D given y is called *preprocessing*. It is done only once, after which a given query q is answered by (adaptively) *probing* (the values of the) cells of D, and the time t spent in answering q is the number of probes made in order to compute $f(y,q)$. The objective is to build D so as to be able to compute $f(y,q)$ (for any y and q) by probing as few cells in D as possible, and the goal is to study this optimal worst-case time $t = \mathsf{T}_s(f)$ as a function of s, m and n. We emphasize that the time is not the running time of the "probing scheme" but only the *number of accesses* to D. This models the situation when the local computation is cheap/fast but the database access is expensive/slow. Moreover, this measure of time is indicative of a fundamental combinatorial limitation on how fast the queries can be answered for any natural notion of time.

The static data structure problem has been extensively studied in the literature ([1–4, 10–12, 16]). Yet, no explicitly defined function f is known for which $t = \omega(n)$ is proven when space $s = poly(n)$. Moreover, a simple argument demonstrates that showing such a super linear bound for an NP function f would *unconditionally* separate NP from the class of read-twice branching programs [13]; a long-standing question in complexity theory [15]. This situation is to be contrasted with an existential result of Miltersen [10] which states that for a *random* function $f : Y \times Q \mapsto \{0,1\}$ we have (w.h.p.) $\mathsf{T}_s(f) = \Omega(m) \gg n$ even when $s = 2^{n-1}$. In other words, one has to read essentially the whole static input even when exponential preprocessing space is given. The result is not surprising since one cannot hope to "efficiently share" limited information to answer queries about completely unrelated values. Still, constructing an *explicit* function that would close this large gap ($O(n)$ vs. $\Omega(m)$) is a major open question in the cell probe model.

While the strength of the cell probe model lies in the clean framework that it provides for studying space-time tradeoffs, the model is unrealistic in at least two respects. Firstly, the probing scheme is all-powerful: it may compute any arbitrary function with the t bits that it reads. So for example, any (even undecidable) function f can be computed with m probes by storing y directly and reading it completely for any query q (so $s = t = m$). Secondly, the preprocessing stage is allowed to use arbitrarily complex functions c_i in the data structure D. For example, any function f can be computed with exponential space $s = 2^n$ and unit time $t = 1$ by simply storing the answer to every possible query. These two aspects of the model at least partly explain the difficulty in obtaining strong lower bounds in this model.

1.2 Our Model: Cell Probe Model with Restricted Preprocessing Power

The goal of this paper is to study space-time tradeoffs in the cell probe model when the preprocessing power may be restricted, making the model more realistic and fine-tuned (while it is an interesting direction to also limit the power of

the probing scheme, we focus on restricting the preprocessing power). We model limited preprocessing power by introducing a new parameter F which is used to denote a (typically infinite) family of (boolean) functions. Given a function family F, we require that the value of each cell c_i in D corresponds to an application of some $g \in \mathsf{F}$ to a subset of y's input bits. We call such data structure D F-*restricted*. Thus, while space constraint s limits the *amount* of precomputed information, the function family F limits the *complexity* of precomputed information. The time t in this new parameterized model is denoted by $t = \mathsf{T}_{\mathsf{F},s}(f)$, referred to as the *cell probe complexity of f w.r.t.* F. Since the extremal case of unrestricted F is simply the standard static data structure problem, our model is a proper extension of the classical cell probe model.

Another extreme of our model is when the space s is unrestricted but F is restricted. Then, the cell probe complexity of f w.r.t. F is simply the *generalized decision tree complexity* of evaluating f over a worst-case query; we denote this measure by $\mathsf{T}_{\mathsf{F}}(f)$. Indeed, unrestricted space implies that we have "precomputed" all possible applications of every $g \in \mathsf{F}$ to every appropriate subset of y's input bits. So the question is no longer the one of creating a space-efficient data structure but that of "adaptive expressibility" of a function f on a given query using functions from F. This is precisely the generalized decision tree complexity w.r.t. F. When F consists merely of the identity function (which we call *trivial* F), this is just the (simple) decision tree complexity measure. Unlike the classical data structure problem, it is often possible to tightly characterize the decision tree complexity of a given function (e.g. [5]). Thus, our general model elegantly *unifies the issue of space-efficient data structures with that of adaptive expressibility*, bridging together the cell probe complexity and the generalized decision tree complexity.

1.3 Our Function f: Graph Properties

The objective of our study is to illustrative different aspects of our new model across a broad spectrum of possible settings of F and s (in particular, to illustrate how much tighter space-time tradeoffs can be obtained once we put reasonable restrictions on F, but we do not limit ourselves to this). For this purpose, we use a problem related to verification of graph properties. Aside from combinatorial interest, an important motivation for this choice comes from the fact that graph properties form a rich class of functions having decision tree complexity. In fact, the famous Aandrea-Karp-Rosenberg (AKR) conjecture states that every non-trivial *monotone* graph property P on n-vertex graphs is *evasive*, i.e. its decision tree complexity $D(P) = \binom{n}{2}$[1]. The conjecture is proven up to a constant factor by Rivest and Vuillemin [14], i.e. $D(P) = \Omega(n^2) = \Omega(m)$. As we will see later, this result in fact forms a "base case" in our study (corresponding to trivial F).

Our Setup: Fix a graph property P. Given an n-vertex graph $G = (V, E)$ as a static input, our goal is to preprocess it to answer dynamic queries of the form:

[1] As we will see, n will be the size of our query, which is consistent with its usage before. Also, $m = \binom{n}{2}$, i.e. the size of the adjacency matrix of our graph.

"Given $X \subseteq V$, does the subgraph G_X of G induced by X satisfy P?"

We refer to this problem as the *induced subgraph problem*. Thus, $f_P(G, X) = P(G_X)$, $m = \binom{n}{2} = \Theta(n^2)$ and a good lower bound would be of the form $t = \Omega(n^2/polylog(s))$. For notational convenience, we will write $\mathsf{T}_{\mathsf{F},s}(P)$ in place of $\mathsf{T}_{\mathsf{F},s}(f_P)$. Note that every query evaluates the property P on some (sub)graph. The hope is that the distribution of P over induced subgraphs of G is very non-trivial, so that it is hard to reuse space when computing f_P.

2 Overview of Our Results

As we pointed, the induced subgraph problem will be an example problem demonstrating the richness of our model as well as different techniques to show space-time tradeoffs in various settings, ranging from the classical cell probe complexity to the generalized decision tree complexity. In what follows, we describe more precisely some of our results, leaving others out due to space limitations. In particular, we will not talk about oblivious and non-deterministic computation in our model. In the next section we will prove a representative result.

2.1 *Restricted* Space, *Unrestricted* Function Families

We start with the extremal case of unrestricted preprocessing. As remarked by Miltersen *et al.* [13], essentially all known lower bound results for the classical data structure problem can be viewed as applications of the following connection between the cell probe and the communication complexity model [10]. In this model, introduced by Yao [18], Alice is given $y \in Y$, Bob is given $q \in Q$ and they wish to compute $f(y, q)$ by exchanging the minimum number of bits. This number is the deterministic communication complexity of f. We describe the connection of [10] more generally using the notion of *asymmetric communication complexity* introduced in [13]. Here, instead of measuring simply the total number of bits exchanged, we measure the number of bits sent by Alice and Bob *individually*. An $[A, B]$-protocol is a protocol where Alice sends at most A bits and Bob sends at most B bits. A lower bound of $[A, B]$ means that in the worst case, either Alice must send $\Omega(A)$ bits or Bob must send $\Omega(B)$ bits in order to compute f. To obtain the most general result, we consider a variant of the cell probe model where a cell size is an additional parameter b. In other words, rather than storing boolean functions in the data structure, we store b-bit functions $c_i : Y \mapsto \{0,1\}^b$ and read all b bits per probe.

Lemma 1. [10] *Any cell probe scheme with space s, cell size b and time t for computing f yields a $[tb, t \log s]$-protocol for computing f. Hence, a lower bound of $[A, B]$ implies that $\mathsf{T}_s(f) \geq \Omega(\min(A/b, B/\log s))$.*

Since $B \leq n$ (Bob can always send his entire input), the best possible lower bound obtainable this way is $t = \Omega(\frac{n}{\log s})$ implying that the current proof techniques cannot cross the $\Omega(n/\log s)$ barrier (but they can possibly give this bound

for large values of b). We show that such a lower bound can indeed be established for the induced subgraph problem using Lemma 1, for any non-trivial monotone graph property as well as for the property PARITY: "Does G have an odd number of edges?". The first result is shown using the *fooling set method* (see [9]) and the second uses the *richness technique* of [13].

Theorem 1.
- $\mathsf{T}_s(P) \geq \Omega(\frac{n}{\log s})$, *for any non-trivial monotone property* P.
- $\mathsf{T}_s(\text{PARITY}) \geq \Omega(\frac{n}{\log s})$ *even when the cell size* $b = n$.

This is as far as we can get in the classical setting using current techniques.

2.2 *Restricted* Space, *Restricted* Function Families

We now turn our attention to the core of our study where we restrict the preprocessing power by means of a function family F. Our goal here is to develop techniques to beat $\Omega(n/\log s)$ illustrating how restricted preprocessing allows to obtain tighter bounds on query time. In fact, for many natural families F we obtain nearly optimal lower bounds of the form $\Omega(n^2/polylog(s))$. The extreme case where F is trivial (i.e. we are allowed to store just the edges of the graph) reduces to the AKR set-up. An $\Omega(n^2)$ lower bound can thus be obtained for any non-trivial monotone graph property (even on a fixed query set). However, once we allow F to contain more general function families, many evasive properties can now be decided by reading very few cells for any *fixed* query. Yet, as we will show, this efficiency in answering specific queries can provably not be translated into a space-efficient data structure that allows to efficiently answer *all* the queries. As in the classical model, the main difficulty is in efficient sharing of information contained in the cells across different queries. However, unlike the classical model, explicit knowledge about the family F enables us to *reason about the behavior of the precomputed information.*

As an elementary example, consider the following evasive property P: Is G an empty graph? Let F be simply the family of OR functions. Clearly, a single OR can express the property on a given G. In fact, this seems to be a very natural function family for computing P. However, a cell storing an OR of all the edges in the graph is of no use in determining whether an induced subgraph G_X satisfies P. Setting any edge outside of G_X to true fixes this OR to 1 without affecting $P(G_X)$. Intuitively speaking, the cells that are sensitive to "many" edges are useful for answering only very few queries, while "short" cells contain little information forcing us to read many of them to answer a "large" query. Indeed, we prove that if F is restricted to only AND and OR functions, $\mathsf{T}_{\mathsf{F},s}(P) = \Omega(n^2/\log^2 s)$ for any evasive property P.

We obtain similar results for more general α-CNF, α-DNF (for constant α) and symmetric function families. We also study the following curious question: What is the time complexity of induced subgraph problem for a property P when the data structure can only contain answers about whether an induced subgraph of the input graph has property P? While any single query can now be answered in one probe, we show $\Omega(n^2/\log^2 s)$ bound for any non-trivial "uniform" monotone property.

Basic Idea: The central technique used for the induced subgraph problem is a probabilistic argument which shows that for any data structuring strategy, there exists an input graph G s.t. (a) it "stabilizes" the value of any cell that is sensitive to many variables (where "many" will depend on space s), and (b) still leaves a large subset X "untouched" such that one can reveal the edges of G_X via an evasive strategy[2]. Since the evasive game on X is now only sensitive to cells with small number of edge variables, we get our desired bounds (same results will apply to monotone graph properties as well by the AKR). We illustarate this "stabilization technique" on α-CNF/α-DNF formulas in Section 3.

2.3 *Unrestricted* Space, *Restricted* Function Families

We now turn our attention to the extreme where the space s is unrestricted and only the family F is restricted. In case of the induced subgraph problem, $\mathsf{T}_{\mathsf{F}}(P)$ reduces to the decision tree complexity w.r.t. F of computing P on the *entire vertex set* V. In other words, how many functions from F one needs to evaluate (adaptively in the worst-case) in order to verify if a given graph G satisfies P? When F contains merely the identity function, it is the setting of the AKR and $\Omega(n^2)$ lower bound is known for any non-trivial monotone P [14, 8]. We examine what happens when we allow more powerful functions in F such as AND, OR (more generally, threshold functions), and XOR (more generally, small degree polynomials). Since space is not an issue, even these seemingly simple function families efficiently capture many evasive properties, e.g., $\mathsf{T}_{\text{AND}}(\text{CLIQUE}) = 1$, $\mathsf{T}_{\text{OR}}(\text{CONNECTIVITY}) = \Theta(n \log n)$ [5], $\mathsf{T}_{\text{XOR}}(\text{PARITY}) = 1$. Yet we will show that for large classes of evasive properties, these families are no more powerful than the trivial identity function.

Small Degree Polynomials: We study the family $\mathcal{F}_{\deg \leq k}$ of all multivariate polynomials over $\mathbb{Z}_2$ of degree at most k; the case $k = 1$ gives the XOR family. Let $\deg(f)$ denote the degree of the (unique) multi-linear polynomial q computing a boolean function f over $\mathbb{Z}_2$. Extending the ideas from Rivest and Vuillemin [14], we establish the following theorem:

Theorem 2. *For any non-trivial monotone graph property P,* $\deg(P) = \Omega(n^2)$.

Since $\deg(P) \leq D(P)$ (any decision tree of depth d yields a polynomial of degree at most d), the theorem implies the AKR conjecture (up to a constant factor) and shows that the degree of a monotone graph property over $\mathbb{Z}_2$ essentially matches its decision tree complexity. We note that for a general (even evasive and monotone!) function f, much larger gaps are possible. Moreover, *any* multi-linear polynomial over $\mathbb{Z}_2$, invariant under relabeling of vertices, computes some valid graph property (e.g. PARITY has degree 1), but this property can never be monotone unless the degree is large. Using the easy observation that $\deg(P) \leq k\mathsf{T}_{\mathcal{F}_{\deg \leq k}}(P)$, we get a tight (e.g., achieved by CLIQUE) general bound:

Corollary 1. *For any non-trivial monotone P, $\mathsf{T}_{\mathcal{F}_{deg \leq k}}(P) = \Omega(\frac{n^2}{k})$. In particular, $\mathsf{T}_{\text{XOR}}(P) = \Omega(n^2)$.*

[2] A strategy that forces one to probe all edges of the graph.

This corollary still implies the AKR result as the identity is a trivial XOR function. Thus, having access to 2^m possible XORs is no more powerful than being able to query only an edge at a time!

AND/OR **Families:** On the other hand, this approach does not work for two most natural extensions of the AKR setup, namely the AND and OR function families, since the degree of these functions can be as large as $\Omega(n^2)$ (in fact, a general bound of $\Omega(n^2)$ does not hold for these families). We develop general techniques for studying these families by essentially reducing their decision tree complexity to a certain measure on simple decision trees. Intuitively, if (simple) decision tree complexity of a property P corresponds to looking at as few edges of G as possible, a good bound on $\mathsf{T}_{\text{AND}}(P)$ ($\mathsf{T}_{\text{OR}}(P)$) corresponds to looking at as few *missing* (*present*) edges of G as possible. We then develop several techniques to lower bound this measure and obtain $\Omega(n^2)$ bound for many properties. Our techniques are based on examining the combinatorial structure of graph certificates and design of general "edge-revealing" strategies for monotone graph properties. One of the strategies we examined in detail is to answer "no" ("yes") unless forced to say "yes" ("no"). Our study of these strategies might be of independent interest. We remark that our techiniques for AND/OR families apply to arbitrary functions and not only to graph properties. We defer the details to the full version.

3 Stabilization Technique

In order to explain the technique, we need two definitions.

Definition 1 (Gadget Graph). An $\langle n, q(n)\rangle$-*gadget graph* $H(V, E)$ is a labeled clique on n vertices such that: (a) each edge is labeled 0 (missing), 1 (present), or $*$ (unspecified), and (b) there exists a subset $Q \subset V$ with $|Q| = q(n)$, such that Q induces a clique where each edge of the clique is labeled $*$. We refer to Q as the *query set* of H.

Definition 2 (Stabilizing Graph). Given an F-restricted data structure D of size s, a graph H is called an $\langle n, q(n), g(s)\rangle$-*stabilizing graph* for D if: (a) H is a $\langle n, q(n)\rangle$-gadget graph, and (b) every cell in D reduces to being a function of at most $g(s)$ edge variables on the partial assignment specified by H.

Now suppose for a function family F we want to show that $\mathsf{T}_{\mathsf{F},s}(P) = \Omega(q^2(n)/g(s))$ for every evasive property P. We start by showing existence of a $\langle n, q(n), g(s)\rangle$-stabilizing graph G_D for every F-restricted data structure D. Thus when G_D is presented as the static input, every cell in D reduces to be a function of at most $g(s)$ edge variables. At the same time, we have access to a query set Q whose every edge is unspecified as yet. We present this set Q as the dynamic input to the scheme and play the evasive game for property P on the subgraph induced by Q. Since each cell probe can reveal at most $g(s)$ edge variables, we obtain the desired $\Omega(q^2(n)/g(s))$ lower bound. The following theorem summarizes this argument.

Theorem 3. *If every* F*-restricted data structure of size* s *has a* $\langle n, q(n), g(s)\rangle$*-stabilizing graph, then for any evasive property* P, $\mathsf{T}_{\mathsf{F},s}(P) = \Omega(q^2(n)/g(s))$.

Thus the heart of our approach is to show the existence of a $\langle n, q(n), g(s)\rangle$-stabilizing graph with suitable parameters. We show existence of such graph using the *probabilistic method*. Typically, we pick a random $\langle n, q(n)\rangle$-gadget graph s.t. for any $h \in \mathsf{F}$, the probability that $h|H$ does not reduce to a function of at most $g(s)$ variables is less than $1/s$. Applying the union bound to the s cells of D, we conclude that $\langle n, q(n), g(s)\rangle$-stabilizing graph exists for *any* D.

As a simple example, let us construct an $\langle n, n/2, \log^2 s\rangle$-stabilizing graph for the family of OR functions, implying that $\mathsf{T}_{\text{OR},s}(P) = \Omega(n^2/\log^2 s)$ for any evasive P. Create H by picking a random subset $Q \subset V$ of size $n/2$ and setting all edges outside Q's induced subgraph to true. Take any OR function $c = e_1 \vee \ldots \vee e_p$ and assume that edges e_i touch k vertices of V. The only way that c is not stabilized to 1 by H, is when all k vertices fall inside Q, i.e. with probability $\binom{n-k}{n/2-k}/\binom{n}{n/2} \leq 1/2^k < 1/s$ if $k > \log s$. So any c that has a reasonable ($> 1/s$) chance of "surviving" must have $k \leq \log s$, i.e. depends on at most $\log^2 s$ edges.

In the remainder of this section, we construct stabilizing graphs for α-CNF and α-DNF formulas (for constant α), deferring other results to the full version.

Theorem 4. *For any evasive property* P *and constant* α,
$$\mathsf{T}_{\alpha-\mathsf{CNF/DNF},s}(P) = \Omega(n^2/(\log^{\alpha-1} n \log^{2\alpha} s)).$$

First, it suffices to show the claimed lower bound for α-DNF formulas (store any α-CNF by storing its complement α-DNF formula). By Theorem 3, it is enough to show the existence of a $\langle n, n/2, O(\log^{\alpha-1} n \log^{2\alpha} s)\rangle$-stabilizing graph for every α-DNF - restricted data structure D. We will proceed similarly to the case of OR formulas above, but slightly change our random experiment to make the analysis simpler. Let us say that a formula f is *stabilized* by a partial assignment if it fixes the value of f. We will show that, if S is a random subset, constructed by choosing each vertex of V with probability $1/2$, then setting each edge variable in $S \times V$ to 0/1 uniformly at random (we refer to this experiment as $\mathbf{A}$) either stabilizes any α-DNF formula or reduces it to be a function of $O(\log^{\alpha-1} n \log^{2\alpha} s)$ edge variables, with probability $1 - o(1/s)$. Setting then $Q = V \setminus S$, and noticing that $|Q| \geq n/2$ with probability at least $1/2$, we get that the claimed stabilizing graph exists.

The claim is shown by induction on α. The base case of $\alpha = 1$ is just slighly more technical than the case of OR functions. Picking appropriate constants, if 1-DNF formula f has more than $\Omega(log^2 s)$ edges, its edges touch $\Omega(\log s)$ vertices of V, so w.h.p. S contains $\Omega(\log s)$ vertices touched by f. Thus, at least $\Omega(\log s)$ edges will be set to 0/1 at random in $\mathbf{A}$. Since each such setting stabilizes 1-DNF w/pr. $1/2$, w.h.p. f will be stabilized during experiment $\mathbf{A}$.

Our inductive step relies on two technical Lemmas, whose proofs we omit due to the space limitations. The first Lemma says that any α-DNF f either has a certain "compact" decomposition into $(\alpha-1)$-DNF's or it has a "large" number of pairwise disjoint terms. The next Lemma shows that experiment $\mathbf{A}$ stabilizes α-DNF formulas with large number of pairwise disjoint terms.

Lemma 2. *Let f be an α-DNF formula on N variables and let $0 < r < 1$ be a positive real. Then either*

- *f has a decomposition of the form $l_0 f_0 + l_1 f_1 + ... + l_{p-1} f_{p-1}$ where l_i's are literals, f_i's are $(\alpha-1)$-DNF formulas, and $p \leq \ln(\alpha 2^\alpha N^\alpha)/r$, or*
- *f has at least $(1/2\alpha r)$ pairwise disjoint (i.e. no common variables) terms.*

Lemma 3. *Let f be an α-DNF formula with $\alpha^2 2^{4\alpha+2} \log^2 s$ pairwise disjoint terms. Then experiment* **A** *stabilizes f with probability at least $1 - 1/s^2$.*

We can now complete the proof. Consider any α-DNF formula f. Let $r = 1/(\alpha^3 2^{4\alpha+3} \log^2 s)$. By Lemma 2, either f has $\alpha^2 2^{4\alpha+2} \log^2 s$ pairwise disjoint terms or it has a representation of the form $l_0 f_0 + l_1 f_1 + \ldots + l_{p-1} f_{p-1}$, where $p \leq p_{\max} = \ln(\alpha n^{2\alpha})/r = O(\log n \log^2 s)$. In case of the first scenario, we know by Lemma 3 that f will be stabilized "almost certainly". Otherwise, we argue that almost certainly f will have no more than $h(\alpha) = O(\log^{\alpha-1} n \log^{2\alpha} s)$ variables. We sketched that $h(1) = O(\log^2 s)$. Using the compact decomposition of f and applying induction to each f_i in the decomposition, we must satisfy the recurrence: $h(\alpha) \leq p_{\max} h(\alpha-1) + p_{\max} \leq (p_{\max})^{\alpha-1} h(1) + \sum_{i=1}^{\alpha-1} (p_{\max})^i$. Using $h(1) = O(\log^2 s)$, we get $h(\alpha) = O(\log^{\alpha-1} n \log^{2\alpha} s)$.

To analyze the probability of failure, denote by $R(\alpha)$ the probability that a given α-DNF formula does not reduce to a function of at most $h(\alpha)$ distinct variables. Using Lemma 3, we have $R(\alpha) \leq p_{\max} R(\alpha-1) + \frac{1}{s^2}$. Scaling $h(1)$ by a suitably large constant, it is easy to see that $R(\alpha)$ can be bounded by $o(1/s)$ for any constant α. This completes the proof of Theorem 4.

Remark: The stabilization technique also works for the *randomized complexity*. Using the best known bound of $\Omega(n^{4/3})$ [6] for randomized decision tree complexity of monotone graph properties, we get a bound $\Omega(n^{4/3}/polylog(s))$ for each of the families we considered.

4 Conclusions and Open Problems

We showed that our model provides a uniform framework to study lower bounds across a spectrum of computational models. Using as an example the induced subgraph problem, we showed some techniques for breaking the $\Omega(n/\log s)$ barrier for various natural settings of F. In the process, we also obtained a strengthening of the Rivest-Vuillemin result, showing that monotone graph properties cannot be expressed by polynomials of sub-quadratic degree. Finally, we obtained new results and techniques on the generalized decision tree complexity with respect to some natural families like the AND/OR family.

We introduced a parameterized cell probe model in an effort to examine space-time tradeoffs in computationally realistic preprocessing scenarios. However, for the sake of getting unconditional results and in order to illustrate our new model more cleanly, we only considered *syntactic* restrictions on the preprocessing function family. It is perhaps more interesting to examine *computational* restrictions. For example, to examine the measure $\mathsf{T}_{\mathsf{poly}}$ (where POLY is the set

of polynomial time computable functions), with a hope of obtaining the first super linear lower bound. A complimentary direction is to place some restriction on the power of the probing scheme, that is, to restrict how the information from the data structure is accessed and/or processed. An example of that is *oblivious computation*, where, given a query q, the probing scheme has to decide right away which t cells to access. Again, it is perhaps a feasible intermediate goal to obtain a super linear bound in this setting. Another interesting direction is to examine the effects of randomization, i.e. when the probing scheme might be probabilistic and have a small propbability of error. Finally, it will be extremely interesting to apply our model to some other important problems, like the *nearest neighbor search* problem (see [2, 3]).

References

1. M. Ajtai. A lower bound for finding predecessors in Yao's cell probe model. In *Combinatorica*, 8:235–247, 1988.
2. A. Borodin, R. Ostrovsky, Y. Rabani. Lower Bounds for High Dimensional Nearest Neighbor Search and Related Problems. In *Proc. of STOC*, 1999.
3. A. Chakrabarti, B. Chazelle, B. Gum, A. Lvov. A good neighbor is hard to find. In *Proc. of STOC*, 1999.
4. P. Elias, R.A. Flower. The complexity of some simple retrieval problems. In *J. ACM*, 22:367–379, 1975.
5. A. Hajnal, W. Maass and G. Turan. On the communication complexity of graph properties. In *Proc. of STOC*, pp. 186–191, 1988.
6. P. Hajnal. An $n^{4/3}$ lower bound on the randomized complexity of graph properties. In *Combinatorica*, 11:131–143, 1991.
7. L. Hellerstein, P. Klein, R. Wilber. On the Time-Space Complexity of Reachability Queries for Preprocessed Graphs. In *Information Processing Letters*, 27:261–267, 1990.
8. J. Kahn, M. Saks, D. Sturtevant. A topological approach to evasiveness. In *Proc. of FOCS*, pp. 31–39, 1983.
9. E. Kushilevitz, N. Nisan. Communication Complexity. *Cambridge University Press*, 1997.
10. P. Miltersen. The bit probe complexity measure revisited. In *Proc. of STACS*, pp. 662–671, 1993.
11. P. Miltersen. Lower bounds for union-split-find related problems on random access machines. In *Proc. STOC*, pp. 625–634, 1994.
12. P. Miltersen. On cell probe complexity of polynomial evaluation. In *Theoretical Computer Science*, 143:167–174, 1995.
13. P. Miltersen, N. Nisan, S. Safra, A. Wigderson. On Data Structures and Asymmetric Communication Comlexity. In *Proc. of STOC*, pp. 103–111, 1995.
14. R. Rivest, J. Vuillemin. On recognizing graph properties from adjecency matrices. In *Theoretical Computer Science*, 3:371–384, 1976.
15. I. Wegener. The complexity of Boolean functions. In *Wiley-Teubner series in Computer Science*, 1987.
16. B. Xiao. New bounds in cell probe model. Ph.D. thesis, UC San Diego, 1992.
17. A. Yao. Should tables be sorted. In *J. ACM*, 28:615–628, 1981.
18. A. Yao. Some complexity questions related to distributed computing. In *Proc. of STOC*, pp. 209–213, 1979.