Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 105 (2011)

Streaming Graph Computations with a Helpful Advisor

Graham Cormode* Michael Mitzenmacher† Justin Thaler‡

October 18, 2011

Abstract

Motivated by the trend to outsource work to commercial cloud computing services, we consider a variation of the streaming paradigm where a streaming algorithm can be assisted by a powerful helper that can provide annotations to the data stream. We extend previous work on such *annotation models* by considering a number of graph streaming problems. Without annotations, streaming algorithms for graph problems generally require significant memory; we show that for many standard problems, including all graph problems that can be expressed with totally unimodular integer programming formulations, only constant space (measured in words) is needed for single-pass algorithms given linear-sized annotations. We also obtain protocols achieving essentially *optimal* tradeoffs between annotation length and memory usage for several important problems, including integer matrix-vector multiplication, as well as shortest s-t path in small-diameter graphs. We also obtain non-trivial tradeoffs for minimum weight bipartite perfect matching and shortest s-t path in general graphs.

1 Introduction

The recent explosion in the number and scale of real-world structured data sets including the web, social networks, and other relational data has created a pressing need to efficiently process and analyze massive graphs. This has sparked the study of graph algorithms that meet the constraints of the standard streaming model: restricted memory and the ability to make only one pass (or few passes) over adversarially ordered data. However, many results for graph streams have been negative, as many foundational problems require either substantial working memory or a prohibitive number of passes over the data [1]. Apparently most graph algorithms fundamentally require flexibility in the way they query edges, and therefore the combination of adversarial order and limited memory makes many problems intractable.

To circumvent these negative results, variants and relaxations of the standard graph streaming model have been proposed, including the Semi-Streaming [2], W-Stream [3], Sort-Stream [4], Random-Order [1], and Best-Order [5] models. In Semi-Streaming, memory requirements are relaxed, allowing space proportional to the number of vertices in the stream but not the number of edges. The W-Stream model allows the algorithm to write temporary streams to aid in computation. And, as their names suggest, the Sort-Stream, Random-Order, and Best-Order models relax the assumption of adversarially ordered input. The Best-Order

*AT & T Labs – Research, graham@research.att.com

†Harvard University, School of Engineering and Applied Sciences, michaelm@eecs.harvard.edu. This work was supported in part by NSF grants CCF-0915922 and CNS-0721491, and in part by grants from Yahoo! Research, Google, and Cisco, Inc.

‡Harvard University, School of Engineering and Applied Sciences, jthaler@seas.harvard.edu. Supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program, and in part by NSF grants CNS-0721491 and CCF-0915922.

ISSN 1433-8092

model, for example, allows the input stream to be re-ordered arbitrarily to minimize the space required for the computation.

In this paper, our starting point is a relaxation of the standard model, closest to that put forth by Chakrabarti *et al.* [6], called the *annotation model*. Motivated by recent work on outsourcing of database processing, as well as commercial cloud computing services such as Amazon EC2, the annotation model allows access to a powerful advisor, or *helper* who observes the stream concurrently with the algorithm. Importantly, in many of our motivating applications, the helper is not a trusted entity: the commercial stream processing service may have executed a buggy algorithm, experienced a hardware fault or communication error, or may even be deliberately deceptive [5, 6]. As a result, we require our protocols to be *sound*: our *verifier* must detect any lies or deviations from the prescribed protocol with high probability.

The most general form of the annotation model allows the helper to provide additional annotations in the data stream *at any point* to assist the verifier, and one of the cost measures is the total length of the annotation. In this paper, however, we focus on the case where the helper's annotation arrives as a single message after both the helper and verifier have seen the stream. The helper's message is also processed as a stream, since it may be large; it often (but not always) includes a re-ordering of the stream into a convenient form, as well as additional information to guide the verifier. This is therefore stronger than the Best-Order model, which only allows the input to be reordered and no more; but it is weaker than the more general *online* model, because in our model the annotation appears only after the input stream has finished.

We argue that this model is of interest for several reasons. First, it requires minimal coordination between helper and verifier, since it is not necessary to ensure that annotation and stream data are synchronized. Second, it captures the case when the verifier uploads data to the cloud as it is collected, and later poses questions over the data to the helper. Under this paradigm, the annotation must come *after* the stream is observed. Third, we know of no non-trivial problems which separate the general online and our "at-the-end" versions of the model, and most prior results are effectively in this model. Indeed, for protocols with linear annotation, the two models are clearly equivalent, as the helper can afford to push the annotation to the end simply by replaying the stream with the required annotation.

Besides being practically motivated by outsourced computations, annotation models are closely related to Merlin-Arthur proofs with space-bounded verifiers, and studying what can (and cannot) be accomplished in these models is of independent interest.

Relationship to Other Work. Annotation models were first explicitly studied by Chakrabarti *et al.* in [6], and focused primarily on protocols for canonical problems in numerical streams, such as Selection, Frequency Moments, and Frequent Items. The authors also provided protocols for some graph problems: counting triangles, connectedness, and bipartite perfect matching. The Best-Order Stream Model was put forth by Das Sarma et al. in [5]. They present protocols requiring logarithmic or polylogarithmic space (in bits) for several problems, including perfect matching and connectivity. Historical antecedents for this work are due to Lipton [7], who used fingerprinting methods to verify polynomial-time computations in logarithmic space. Recent work verifies shortest-path computations using cryptographic primitives, using polynomial space for the verifier [8].

Our Contributions. We identify two qualitatively different approaches to producing protocols for problems on graphs with n nodes and m edges. In the first, the helper directly proves matching upper and lower bounds on a quantity. Usually, proving one of the two bounds is trivial: the helper provides a feasible solution to the problem. But proving *optimality* of the feasible solution can be more difficult, requiring the use of structural properties of the problem. In the second, we simulate the execution of a non-streaming algorithm, using the helper to maintain the algorithm's internal data structures to control the amount of memory used by the verifier. The helper must provide the contents of the data structures so as to limit the amount of annotation required.

Using the first approach (Section 3), we show that only constant space and annotation linear in the input size m is needed to determine whether a directed graph is a DAG and to compute the size of a maximum matching. We describe this as an $(m, 1)$ protocol, where the first entry refers to the annotation size (which we also call the hcost) and the second to the memory required for the verifier (which we also call the vcost). Our maximum matching result significantly extends the bipartite perfect matching protocol of [6], and is tight for dense graphs, in the sense that there is a lower bound on the *product* of hcost and vcost of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits for this problem. Second, we define a streaming version of the linear programming problem, and provide an $(m, 1)$ protocol. By exploiting duality, we hence obtain $(m, 1)$ protocols for many graph problems with totally unimodular integer programming formulations, including shortest s-t path, max-flow, min-cut, and minimum-weight bipartite perfect matching (MWBPM). We also show all are tight by proving lower bounds of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits for all four problems. A more involved protocol obtains *optimal* tradeoffs between annotation cost and working memory for dense LPs, matrix-vector multiplication, and eigenvalue verification; this complements recent results on approximate linear algebra in streaming models (see e.g. [9, 10]). Next, motivated by applications in which weak peripheral devices or sensors perform error correction on signals, we present extensions of our LP protocol to the more general class of quadratic programs. Finally, we present a novel extension of techniques from [6] to obtain non-trivial tradeoffs (optimal in some important regimes) for two of the totally modular integer programs: shortest s-t path and MWBPM (Section 4).

For the second approach (Section 5), we make use of the idea of "memory checking" due to Blum et al. [11], which allows a small-space verifier to outsource data storage to an untrusted server. We present a general simulation theorem based on this checker, and obtain as corollaries tight protocols for a variety of canonical graph problems. In particular, we give an $(m, 1)$ protocol for verifying a minimum spanning tree, an $(m + n \log n, 1)$ protocol for single-source shortest paths, and an $(n^3, 1)$ protocol for all-pairs shortest paths. We provide a lower bound of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits for the latter two problems, and an identical lower bound for MST when the edge weights can be given incrementally. While powerful, this technique has its limitations: there does not seem to be any generic way to obtain the same kind of tradeoffs between hcost and vcost observed above. Further, there are some instances where direct application of memory checking does not achieve the best bounds for a problem. We demonstrate this by presenting an $(n^2 \log n, 1)$ protocol to find the diameter of a graph; this protocol leverages the ability to use randomized methods to check computations more efficiently than via generating or checking a deterministic witness. In this case, we rely on techniques to verify matrix-multiplication in quadratic time, and show that this is tight via a nearly matching lower bound for diameter of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$.

In contrast to problems on numerical streams, where it is often trivial to obtain $(m, 1)$ protocols by replaying the stream in sorted order, achieving linear-sized annotations with logarithmic space is more challenging for many graph problems. Simply providing the solution (e.g. a graph matching or spanning tree) is insufficient, since we have the additional burden of demonstrating that this solution is *optimal*. A consequence is that we are able to provide solutions to several problems for which no solution is known in the best-order model (even though one can reorder the stream in the best-order model so that the "solution" edges arrive first).

2 Model and Definitions

Consider a data stream $\mathcal{A} = \langle a_1, a_2, \ldots, a_m \rangle$ with each a_i in some finite universe $\mathcal{U}$. Consider a probabilistic *verifier* $\mathcal{V}$ who observes $\mathcal{A}$ and a deterministic *helper* $\mathcal{H}$ who also observes $\mathcal{A}$ and can send a message $\hbar$ to

$\mathcal{V}$ after $\mathcal{A}$ has been observed by both parties. This message, also referred to as an *annotation*, should itself be interpreted as a data stream that is parsed by $\mathcal{V}$, which may permit $\mathcal{V}$ to use space sublinear in the size of the annotation itself. That is, $\mathcal{H}$ provides an annotation $\mathcal{h}(\mathcal{A}) = (\mathcal{h}_1(\mathcal{A}), \mathcal{h}_2(\mathcal{A}), \ldots \mathcal{h}_\ell(\mathcal{A}))$.

We study randomized streaming protocols for computing functions $f(\mathcal{A}) \to \mathbb{Z}$. Specifically, assume $\mathcal{V}$ has access to a private random string $\mathcal{R}$ and at most $w(m)$ machine words of working memory, and that $\mathcal{V}$ has one-way access to the input $\mathcal{A} \cdot \mathcal{h}$, where $\cdot$ represents concatenation. Denote the output of protocol $\mathcal{P}$ on input $\mathcal{A}$, given helper $\mathcal{h}$ and random string $\mathcal{R}$, by $out(\mathcal{P}, \mathcal{A}, \mathcal{R}, \mathcal{h})$. We allow $\mathcal{V}$ to output $\perp$ if $\mathcal{V}$ is not convinced that the annotation is valid.

If the function f is non-boolean, we sometimes consider a notion of approximation. We say f is computed correctly if the answer returned is in some set $C(f(\mathcal{A}))$, e.g., $\{a : |a - f(\mathcal{A})| \leq \epsilon |f(\mathcal{A})|\}$ for multiplicative approximations, and $C(f(\mathcal{A})) = \{f(\mathcal{A})\}$ for exact protocols.

Definition 2.1. *$\mathcal{h}$ is a valid helper with respect to $\mathcal{P}$ if for all streams $\mathcal{A}$,* $\Pr_{\mathcal{R}}(out(\mathcal{P}, \mathcal{A}, \mathcal{R}, \mathcal{h}) \in C(f(\mathcal{A}))) = 1$.

It would also be natural to weaken the requirement in the above definition to $\Pr_{\mathcal{R}}(out(\mathcal{P}, \mathcal{A}, \mathcal{R}, \mathcal{h}) \in C(f(\mathcal{A}))) \geq 1 - \delta$ for a small constant $\delta > 0$. However, all of our protocols satisfy the more stringent requirement above; this property is referred to as *perfect completeness* in the literature on interactive proofs (see, e.g., [12, Chapter 8]).

Definition 2.2. *$\mathcal{P}$ is a valid protocol for f if*

1. *There exists at least one valid helper $\mathcal{h}$ with respect to $\mathcal{P}$ and*
2. *For all helpers $\mathcal{h}'$ and all streams $\mathcal{A}$,*

$$\Pr_{\mathcal{R}}(out(\mathcal{P}, \mathcal{A}, \mathcal{R}, \mathcal{h}') \notin C(f(\mathcal{A})) \cup \{\perp\}) \leq 1/3.$$

Conceptually, $\mathcal{P}$ is a valid protocol for f if for each stream $\mathcal{A}$ there is *at least* one way to convince $\mathcal{V}$ of the true value of $f(\mathcal{A})$, and any false claims are detected by $\mathcal{V}$ with probability at least 1/3. The constant $\frac{1}{3}$ can be any constant less than $\frac{1}{2}$.

Let $\mathcal{h}$ be a valid helper chosen to minimize the length of $\mathcal{h}(\mathcal{A})$ for all $\mathcal{A}$. We define the help cost $\mathrm{hcost}(\mathcal{P})$ to be the maximum length of $\mathcal{h}$ over all $\mathcal{A}$ of length m, and the verification cost $\mathrm{vcost}(P) = w(m)$, the amount of working memory used by the protocol P. All costs are expressed in machine words of size $\Theta(\log m)$ bits, i.e. we assume any quantity polynomial in the input size can be stored in a constant number of words. In contrast, lower bounds are expressed in bits. We say that $\mathcal{P}$ is an (h, v) protocol for f if $\mathcal{P}$ is valid and $\mathrm{hcost}(\mathcal{A}) = O(h)$, $\mathrm{vcost}(\mathcal{A}) = O(v)$. While both hcost and vcost are natural costs for such protocols, we often aim to achieve a vcost of $O(1)$ and then minimize hcost. In other cases, we show that hcost can be decreased by increasing vcost, and study the tradeoff between these two quantities.

In this paper we primarily consider graph streams, which are streams whose elements are edges of a graph G. More formally, consider a stream $\mathcal{A} = \langle e_1, e_2, \ldots, e_m \rangle$ with each $e_i \in [n] \times [n]$. Such a stream defines a (multi)graph $G = (V, E)$ where $V = \{v_1, ..., v_n\}$ and E is the (multi)set of edges that naturally corresponds to $\mathcal{A}$. We use the notation $\{i : m(i)\}$ for the multiset in which i appears with multiplicity $m(i)$. Finally, we will sometimes consider graph streams with directed edges, and sometimes with weighted edges; in the latter case each edge $e_i \in [n] \times [n] \times \mathbb{Z}_+$, and we assume all edge weights are polynomial in n.

Note throughout that whenever we state a lower bound it is shown by generating a "worst-case" input instance. For the bounds we show, this corresponds to a dense graph, i.e. one with $m = \Omega(n^2)$ edges. Thus, we often show upper bounds proportional to m, and lower-bounds on dense graphs proportional to n^2.

2.1 Fingerprints

Our protocols make careful use of *fingerprints*, permutation-invariant hashes that can be efficiently computed in a streaming fashion. They determine in small space (with high probability) whether two streams have identical frequency distributions. They are the workhorse of algorithms proposed in earlier work on streaming models with an untrusted helper [5, 6, 7, 13]. We sometimes also need the fingerprint function to be linear.

Definition 2.3 (Fingerprints)**.** *A fingerprint of a multiset $M = \{i : m(i)\}$, where each $i \in [q]$ for some known upper bound q, is defined as an element over the finite field with p elements, $\mathbb{F}_p$, as $\mathfrak{f}_{p,\alpha}(M) = \sum_{i=1}^{q} m(i)\alpha^i$, where α is chosen uniformly at random from $\mathbb{F}_p$. We typically leave p, α implicit, and just write $\mathfrak{f}(M)$.*

Some properties of $\mathfrak{f}$ are immediate: it is linear in M, and can easily be computed incrementally as elements of $[q]$ are observed in a stream one by one. The main property of $\mathfrak{f}$ is that if $M \neq M'$ then $\Pr[\mathfrak{f}(M) = \mathfrak{f}(M')] \leq q/p$ over the random choice of α (due to standard properties of polynomials over a field). Therefore, if p is sufficiently large, say, polynomial in q and in an (assumed) upper bound on the multiplicities $m(i)$, then this event happens with only polynomially small probability. For cases when the domain of the multisets is not $[q]$, we either establish a bijection to $[q]$ for an appropriate value of q, or use a hash function to map the domain onto a large enough $[q]$ such that there are no collisions with high probability (whp). In all cases, p is chosen to be $O(1)$ words.

A common subroutine of many of our protocols forces $\mathcal{H}$ to provide a "label" $l(u)$ for each node upfront, and then replay the edges in E, with each edge (u, v) annotated with $l(u)$ and $l(v)$ so that each instance of each node v appears with the *same* label $l(v)$.

Definition 2.4. *We say a list of edges E' provided by $\mathcal{H}$ is* label-augmented *if (a) E' is preceded by a sorted list of all the nodes $v \in V$, each with a value $l(v)$ and $\deg(v)$, where $l(v)$ is the label of $\mathcal{V}$ and $\deg(v)$ is claimed to be the degree of v; and (b) each edge $e = (u, v)$ in E' is annotated with a pair of symbols $l(e, u)$ and $l(e, v)$. We say a list of label-augmented edges E' is valid if for all edges $e = (u, v)$, $l(e, u) = l(u)$ and $l(e, v) = l(v)$; and $E' = E$ as sets, where E is the set of edges observed in the stream.*

Lemma 2.5 (Consistent Labels)**.** *There is an $(m, 1)$ protocol that outputs the value 1 given any valid list of label-augmented edges,and outputs $\perp$ with probability at least $2/3$ if the list is not valid.*

Proof. Let E' denote the set of edges in the label-augmented list. $\mathcal{V}$ computes four separate fingerprints over the course of the protocol. First, while processing the stream, $\mathcal{V}$ computes a fingerprint, $\mathfrak{f}(E)$, of the set of edges in the stream. Second, $\mathcal{V}$ uses the annotation from Definition 2.4 (a) to make a fingerprint of the multiset $S_1 := \{(u, l(u)) : \deg(u)\}$. Third, while processing the label-augmented list of edges, $\mathcal{V}$ computes a fingerprint $\mathfrak{f}(E')$ of the set E'. Fourth, $\mathcal{V}$ computes a fingerprint of the multiset S_1' of all $(u, l(e, u))$ pairs seen while observing the label-augmented list. $\mathcal{V}$ outputs the value 1 if $\mathfrak{f}(S_1) = \mathfrak{f}(S_1')$ and $\mathfrak{f}(E) = \mathfrak{f}(E')$, otherwise $\mathcal{V}$ outputs $\perp$.

Clearly if the list of label-augmented edges is valid, $\mathcal{V}$ will output the value 1 with probability 1. If the list is not valid, then either $E' \neq E$ (as sets), or there exists an edge (u, v) such that $l(e, u) \neq l(u)$ or $l(e, v) \neq l(v)$. In the first case, $\mathfrak{f}(E) \neq \mathfrak{f}(E')$ whp, so $\mathcal{V}$ will output $\perp$ whp. In the second case, assume without loss of generality that $l(e, u) \neq l(u)$. The multiset S_1' does not equal S_1, because $(u, l(u))$ is the unique tuple in S_1 whose first coordinate equals u. Therefore, $\mathfrak{f}(S_1) \neq \mathfrak{f}(S_1')$ whp, and $\mathcal{V}$ will output $\perp$ whp in this case as well. □ □

3 Directly Proving Matching Upper and Lower Bounds

3.1 Warmup: Topological Ordering and DAGs

A (directed) graph G is a DAG if and only if G has a *topological ordering*, which is an ordering of $\mathcal{V}$ as $v_1, \dots v_n$ such that for every edge (v_i, v_j) we have $i < j$ [14, Section 3.6]. Hence, if G is a DAG, $\mathcal{H}$ can prove it by providing a topological ordering. If G is not a DAG, $\mathcal{H}$ can provide a directed cycle as witness.

Theorem 3.1. *There is an $(m, 1)$ protocol to determine if a graph is a DAG.*

Proof. If G is not a DAG, $\mathcal{H}$ provides a directed cycle C as $(v_1, v_2), (v_2, v_3) \dots (v_k, v_1)$. To ensure $C \subsetneq E$, $\mathcal{H}$ then provides a set C' claimed to equal $E \setminus C$, and $\mathcal{V}$ checks that $\mathfrak{f}(C \cup C') = \mathfrak{f}(E)$.

If G is a DAG, let $v_1, \dots v_n$ be a topological ordering of G. We require $\mathcal{H}$ to replay the edges of G, with edge (v_i, v_j) annotated with the ranks of v_i and v_j i.e. i and j. We ensure $\mathcal{H}$ provides consistent ranks via the Consistent Labels protocol of Lemma 2.5, with the ranks as "labels". If any edge (v_i, v_j) is presented with $j > i$, $\mathcal{V}$ immediately outputs $\perp$. □ □

3.2 Maximum Matching

We give an $(m, 1)$ protocol for maximum matching which leverages the combinatorial structure of the problem. This is tight up to logarithmic factors. Previously, matching was only studied in the bipartite case, where an $(m, 1)$ protocol and a lower bound of $\text{hcost} \cdot \text{vcost} = \Omega(n^2)$ bits for dense graphs were shown [6, Theorem 11].

Our protocol shows matching upper and lower bounds on the size of the maximum matching. Any feasible matching presents a lower bound. For the upper bound we appeal to the Tutte-Berge formula [15, Chapter 24]: the size of a maximum matching of a graph $G = (V, E)$ is equal to

$$\frac{1}{2} \min_{V_S \subseteq V} (|V_S| - \text{occ}(G - V_S) + |V|),$$

where $G - V_S$ is the subgraph of G obtained by deleting the vertices of V_S and all edges incident to them, and $\text{occ}(G - V_S)$ is the number of components in the graph $G - V_S$ that have an odd number of vertices. So for any set of nodes V_S, $\frac{1}{2}(|V_S| - \text{occ}(G - V_S) + |V|)$ is an upper bound on the size of the maximum matching, and there exists some V_S for which this quantity equals the size of a maximum matching M. Conceptually, providing both V_S and M, $\mathcal{H}$ proves that the maximum matching size is M. Additionally, $\mathcal{H}$ has to provide a proof of the value of $\text{occ}(G - V_S)$ to $\mathcal{V}$.

Theorem 3.2. *There is an $(m, 1)$ protocol for maximum matching. Moreover, any protocol for max-matching requires* $\text{hcost} \cdot \text{vcost} = \Omega(n^2)$ *bits.*

Proof. To prove a lower bound of k on the size of the maximum matching, $\mathcal{H}$ provides a matching $M = (V_M, E_M)$ of size $|E_M| = k$, and then proves that M is indeed a matching. It suffices to prove that $|V_M| = 2|E_M|$ and $M \subseteq E$. First, $\mathcal{H}$ lists E_M, and $\mathcal{V}$ fingerprints the nodes present as $f(V_M)$. $\mathcal{H}$ then presents V'_M which is claimed to be V_M in sorted order, allowing $\mathcal{V}$ to easily check no node appears more than once and that $\mathfrak{f}(V_M) = \mathfrak{f}(V'_M)$. Next, $\mathcal{H}$ provides $E \setminus M$, allowing $\mathcal{V}$ to check that $\mathfrak{f}(M \cup (E \setminus M)) = \mathfrak{f}(E)$. Hence M is a matching.

To prove an upper bound of k on the size of the maximum matching, $\mathcal{H}$ sends a (sorted) set $V_S \subseteq V$, where $\frac{1}{2}(|V_S| - \text{occ}(G - V_S) + |V|) = k$. Both $|V_S|$ and $|V|$ are computed directly; for $\text{occ}(G - V_S)$, $\mathcal{H}$ sends a sequence of (sub)graphs $C_i = (V_i, E_i) \subseteq V \times E$ claimed to be a partition of $G - V_S$ into connected components. $\mathcal{V}$ can easily compute c, the number of C_i's with an odd number of nodes. To ensure that the C_i's are indeed the connected components of $G - V_S$, it suffices to show that (a) each C_i is connected in $G - V_S$; (b) $V \setminus V_S$ is the disjoint union of the V_i's; and (c) there is no edge $(v, w) \in E$ s.t. $v \in V_i, w \in V_j, i \neq j$.

To prove Property (a), $\mathcal{H}$ presents the (sub)graph C_i as $V_i \subset V$ (in sorted order) where each $\mathcal{V}$ is paired with its degree $\deg(v)$; followed by $E_i \subset E$ (in arbitrary order). Fingerprints are used to ensure that the multiset of nodes present in E_i matches the claimed degrees of nodes in V_i. If these fingerprints agree, then (whp) $E_i \subseteq V_i \times V_i$. Then $\mathcal{H}$ uses the connectivity protocol from [6, Theorem 5.6] on the (sub)graph $C_i = (V_i, E_i)$ to prove that C_i is connected. Each of these checks on C_i has hcost $O(|E_i|)$. Note that $\mathcal{V}$ requires only a constant number of fingerprints for these checks, and can use the same working memory for each different C_i to check that $E_i \subseteq V_i \times V_i$ and that C_i is connected. The total vcost over all C_i is a constant number of fingerprints; the total hcost is $O(m)$.

Property (b) is checked by testing $\mathfrak{f}\big((\cup_i V_i) \cup V_S\big) = \mathfrak{f}(V)$, where the unions in the LHS count multiplicities; if the fingerprints match then whp $V \setminus V_S$ is the *disjoint* union of the V_i's. For (c), it suffices to ensure that each each edge in $E \setminus (\bigcup_i E_i)$ is incident to at least one node in V_S, as we have already checked that no edges in $\bigcup_i E_i$ cross between V_i and V_j for $i \neq j$. To this end, we use the "Consistent Labels" protocol of Lemma 2.5, with $l(u) = 1$ indicating $u \in V_S$ and $l(u) = 0$ indicating $u \notin V_S$, to force $\mathcal{H}$ to replay all of E with each edge (u, v) annotated with $l(u)$ and $l(v)$. This allows $\mathcal{V}$ to identify the set E_S of edges incident to at least one node in V_S. $\mathcal{V}$ checks that $\mathfrak{f}\big((\cup_i E_i) \cup E_S\big) = \mathfrak{f}(E)$, which ensures (whp) that Property (c) holds and that over the entire partition of G no edges are added or omitted. Finally, provided all the prior fingerprint tests pass, the protocol accepts if c, the number of C_i's with an odd number of nodes, satisfies $\frac{1}{2}(|S| - c + |V|) = k$.

For the lower bound, note that a lower bound of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits for dense graphs was shown in [6, Theorem 11] for bipartite perfect matching, and the same lower bound applies to the more general problem of maximum matching considered here. □ □

3.3 Linear Programming

We present protocols to verify solutions of large classes of linear programming problems in our model by leveraging the theory of LP duality. This leads to non-trivial protocols for a variety of graph problems.

Definition 3.3. *We are given a data stream $\mathcal{A}$ containing entries of vectors $\mathbf{b} \in \mathbb{Z}^b$, $\mathbf{c} \in \mathbb{Z}^c$, and non-zero entries of a $b \times c$ matrix A in some arbitrary order, possibly interleaved. We assume the absolute value of all entries is polynomial in b and c. Each item in the stream indicates the index of the object it pertains to. The LP streaming problem on $\mathcal{A}$ is to determine the value of the linear program* $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$.

Our definition makes the standard assumption that the constraints and optimization criteria are integral. We explain later when and how our results extend to rational valued programs.

We present our protocol as if each entry of each object appears at most once (if an entry does not appear, it is assumed to be zero). When this is not the case, the final value for that entry is interpreted as the *sum* of all corresponding values in the stream.

Numerical Issues. Our methods require $\mathcal{H}$ to specify an optimal solution to the linear program, and therefore our protocols are only efficient if there exists an optimal solution with a succinct representation. However, the fact that all entries of A, $\mathbf{b}$, and $\mathbf{c}$ are integral is not sufficient to guarantee this property, as the following example demonstrates:

Example 3.4. *Consider the following linear program:*

$$\text{maximize } \mathbf{1}^T\mathbf{x} \text{ subject to } \begin{bmatrix} A \\ -A \end{bmatrix} \mathbf{x} \le \begin{bmatrix} \mathbf{b} \\ -\mathbf{b} \end{bmatrix}.$$

Here $\mathbf{1}$ denotes the all-ones vector, $\mathbf{b} = 2\mathbf{e}_1$ has 2 in the first coordinate and 0 in all others, and A is the $b \times c$ matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & -2 & 1 & \dots & 0 & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \dots & -2 & 1 \end{bmatrix}$$

Note that A is sparse, with only $O(b)$ non-zero entries. The linear program has a unique feasible point: the vector $\mathbf{x} \in \mathbb{Z}^c$ with $\mathbf{x}_i = 2^i$. The value of the linear program is $\mathbf{1}^T\mathbf{x} = \Omega(2^c)$, and therefore specifying the value as an integer requires essentially linear space; $\mathcal{V}$ cannot manipulate values this large in sublinear space. Moreover, specifying the optimal solution $\mathbf{x}$ requires $\Omega(bc)$ bits of annotation, which is quadratic in the stream length if $b = c$. □

We therefore identify a simple and general condition that rules out unmanageable linear programs such as Example 3.4. Specifically, denote the maximum absolute value over all the subdeterminants of A by Δ. We observe that if $\Delta = \text{poly}(b, c)$, then the optimal solution has bounded size (i.e. each entry is bounded) as a function of Δ, and hence the optimal value is also bounded. This ensures that we can verify the computation using exact computations in a field of bounded size. As an important special case, which we elaborate upon below, we obtain protocols for all *totally unimodular* integer programs. More generally, linear programs with bounded subdeterminants are a common object of study in the optimization literature, with many algorithms running in time polynomial in b, c, and Δ or $\log \Delta$ depending on the algorithm and the setting (see e.g. [16], [17], and the references therein).

Theorem 3.5. *There is an $(|A|(1 + \frac{\log \Delta}{\log |A|}), 1 + \frac{\log \Delta}{\log |A|})$ protocol for the LP streaming problem, where $|A|$ is the number of non-zero entries in the constraint matrix A of $\mathcal{A}$. In particular, if $\Delta = \text{poly}(|A|)$ then there is an $(|A|, 1)$ protocol.*

Proof. First assume that the value of the linear program is finite. The protocol shows an upper bound by providing a primal-feasible solution $\mathbf{x}$, and a lower bound by providing a dual-feasible solution $\mathbf{y}$. When the value of both solutions match, $\mathcal{V}$ is convinced that the optimal value has been found.

Bounded Solution Size. We begin by arguing that there exists a primal-optimal solution $\mathbf{x}$ which requires $O(c(1 + \frac{\log \Delta}{\log |A|}))$ words to represent. This argument is standard in the theory of linear programming; we present it here for completeness.

Let $P = \{\mathbf{x} | A\mathbf{x} \le \mathbf{b}\}$ denote the feasible region of the linear program. It is well known that for any linear program with a finite optimum, the set of optimal solutions contains a minimal face of P as a subset, where a set F is a minimal face of P if $\emptyset \ne F \subseteq P$ and $F = \{\mathbf{x} | A'\mathbf{x} = \mathbf{b}'\}$ for some subsystem $A'\mathbf{x} \le \mathbf{b}'$ of $A\mathbf{x} \le \mathbf{b}$ with A' having full row-rank [18, Theorem 8.4]. We may write $A' = \begin{bmatrix} U & V \end{bmatrix}$, possibly after permuting coordinates, for some integer matrix U whose determinant satisfies $0 < |\det(U)| \le \Delta$. Then $\mathbf{x} = \begin{bmatrix} U^{-1}\mathbf{b}' \\ 0 \end{bmatrix}$ is an optimal solution to the linear program. By Cramer's Rule, $\mathbf{x}_i = \frac{\det(\widetilde{U}_i)}{\det(U)}$, where $\widetilde{U}_i$

is obtained from U by replacing the i'th column with $\mathbf{b}'$. Both the numerator and denominator of this fraction are integers of magnitude $\mathrm{poly}(b, c, \Delta)$, and hence each entry of $\mathbf{x}$ can be represented exactly using $O(1 + \frac{\log \Delta}{\log |A|})$ words.

A nearly identical argument shows that there also exists a dual-optimal solution $\mathbf{y}$ requiring $O(b)$ words to represent. Indeed, the dual linear program is given by $\min\{\mathbf{b}^T\mathbf{y} | A^T\mathbf{y} = \mathbf{c}, \mathbf{y} \geq 0\}$ (see, e.g., [18, Corollary 7.1g]). Letting P denote the feasible region of the dual linear program, the set of optimal solutions contains a minimal face of P as a subset. We can therefore find a dual-optimal $\mathbf{y}$ such that $\mathbf{y}_i = \frac{\det(\widetilde{U}_i)}{\det(U)}$, where (possibly after permuting coordinates) U is a submatrix of $\begin{bmatrix} A & -A & I \end{bmatrix}^T$, and $\widetilde{U}_i$ is obtained from U by replacing the i'th column with $\begin{bmatrix} \mathbf{c}^T & -\mathbf{c}^T & \mathbf{0}^T \end{bmatrix}^T$. Since the subdeterminants of A are bounded in absolute value by Δ, the subdeterminants of $\begin{bmatrix} A & -A & I \end{bmatrix}^T$ are as well (this is an easy extension of [18, Corollary 19.1a]). Hence both the numerator and denominator of $\mathbf{y}_i$ are integers of magnitude $\mathrm{poly}(b, c, \Delta)$, and each entry of $\mathbf{y}$ can be represented in $O(1 + \frac{\log \Delta}{\log |A|})$ words.

Protocol Specification and Correctness. From the stream, $\mathcal{V}$ fingerprints the sets $S_A = \{(i, j, A_{i,j})\}$, $S_B = \{(i, \mathbf{b}_i)\}$ and $S_C = \{(i, \mathbf{c}_j)\}$. Then $\mathcal{H}$ provides all pairs of values $\mathbf{c}_j, \mathbf{x}_j, 1 \leq j \leq c$, with $\mathbf{x}_j$ represented as p_j/q_j for integers $p, q = \mathrm{poly}(b, c, \Delta)$, and with each $\mathbf{x}_j$ additionally annotated with $|A_{\cdot j}|$, the number of non-zero entries in column j of A. This allows $\mathcal{V}$ to fingerprint the multiset $S_X = \{(j, p_j, q_j) : |A_{\cdot j}|\}$ and calculate the solution cost $\sum_{j=1}^{b} \mathbf{c}_j \mathbf{x}_j$.

To prove feasibility, for each row $A_{i\cdot}$ of A, $\mathcal{H}$ sends $\mathbf{b}_i$, then (the non-zero entries of) $A_{i\cdot}$ so that A_{ij} is annotated with $\mathbf{x}_j$. This allows the ith constraint to be checked easily using $O(1+\frac{\log \Delta}{\log |A|})$ words of memory. Note that the denominator q_j in the expression $\mathbf{x}_j = \frac{\det(\widetilde{U}_j)}{\det(U)} = \frac{p_j}{q_j}$ is the same across *all* j, so $\mathcal{V}$ can exactly represent $\sum_j A_{ij}\mathbf{x}_j$ in $O(1 + \frac{\log \Delta}{\log |A|})$ words and compare to $\mathbf{b}_i$, without any need for rounding.

$\mathcal{V}$ fingerprints the values given by $\mathcal{H}$ for A, $\mathbf{b}$, and $\mathbf{c}$, and compares them to those for the stream. A single fingerprint of the multiset of values presented for $\mathbf{x}$ over all rows is compared to $\mathfrak{f}(S_X)$. The protocol accepts $\mathbf{x}$ as feasible if all constraints are met and all fingerprint tests pass.

Correctness follows by observing that the agreement with $\mathfrak{f}(A)$ guarantees (whp) that each entry of A is presented correctly and no value is omitted. Since $\mathcal{H}$ presents each entry of $\mathbf{b}$ and $\mathbf{c}$ once, in index order, the fingerprints $\mathfrak{f}(S_B)$ and $\mathfrak{f}(S_C)$ ensure that these values are presented correctly. The claimed $|A_{\cdot j}|$ values must be correct: if not, then the fingerprints of either S_X or S_A will not match those of the multisets provided by $\mathcal{H}$. $\mathfrak{f}(S_X)$ also ensures that each time $\mathbf{x}_j$ is presented, the same value is given (similar to Lemma 2.5).

To prove that $\mathbf{x}$ is primal-optimal, it suffices to show a feasible solution $\mathbf{y}$ to the dual linear program such that $\mathbf{c}^T\mathbf{x} = b^T\mathbf{y}$. That is, $\mathcal{V}$ checks that $\mathbf{y} \geq 0$, $A^T\mathbf{y} = \mathbf{c}$, and $\mathbf{c}^T\mathbf{x} = b^T\mathbf{y}$. The first check is trivial, while the latter two checks can be done as in the protocol above, with A replaced by A^T and $\mathbf{b}$ replaced by $\mathbf{c}$.

Infeasible Programs. If the value of the primal linear program is ∞, i.e. the primal is infeasible, then by Farkas' Lemma [18, Corollary 7.1e], $\mathcal{H}$ can prove this by specifying a vector $\mathbf{y} \geq 0$ such that $A^T\mathbf{y} = \mathbf{0}$, $\mathbf{b}^T\mathbf{y} = -1$. We may write this in matrix form as $\begin{bmatrix} A^T \\ \mathbf{b} \end{bmatrix} \mathbf{y} = \begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix}$. Let D be a matrix consisting of a maximal subset of linearly independent rows in the matrix $\begin{bmatrix} A^T \\ \mathbf{b} \end{bmatrix}$ and let $\mathbf{d}$ be the corresponding entries of $\begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix}$. Clearly the absolute value of all subdeterminants of D is $\mathrm{poly}(b, c, \Delta)$ because all subdeterminants of A^T are bounded in absolute value by Δ and all entries in $\mathbf{b}$ are $\mathrm{poly}(b, c)$ in absolute value. Since D has full row rank, we may write $D = \begin{bmatrix} U & V \end{bmatrix}$, possibly after permuting coordinates, for some integer matrix U with $0 < |\det(U)|$. Then $\mathbf{y} = \begin{bmatrix} U^{-1}\mathbf{d} \\ 0 \end{bmatrix}$ satisfies $A^T\mathbf{y} = \mathbf{0}$, $\mathbf{b}^T\mathbf{y} = -1$ as desired. By Cramer's Rule, $\mathbf{y}_i = \frac{\det(\widetilde{D})}{\det(D)}$, where $\widetilde{D}$ is obtained from D by replacing the i'th column with $\mathbf{d}$. Both the numerator and denominator of this fraction are integers of magnitude $\mathrm{poly}(b, c, \Delta)$, and hence each entry of $\mathbf{y}$ can be represented exactly in $O(1 + \frac{\log \Delta}{\log |A|})$ words.

So $\mathcal{H}$ can specify $\mathbf{y}$, and by essentially repeating the original protocol, $\mathcal{V}$ can check that indeed $A^T\mathbf{y} = \mathbf{0}$ and $\mathbf{b}^T\mathbf{y} = -1$.

Finally, if the value of the primal program is $-\infty$, then by Farkas' Lemma, $\mathcal{H}$ can prove this by specifying a vector $\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{0}$, $\mathbf{c}^T\mathbf{x} = 1$. Let $P = \{\mathbf{x} | A\mathbf{x} \leq \mathbf{0},\ \mathbf{c}^T\mathbf{x} = 1\}$. $\mathcal{H}$ can choose an $\mathbf{x}$ which is an element of a minimal face of P, as such an $\mathbf{x}$ can be specified succinctly by $\mathcal{H}$.

That is, let D be a matrix consisting of a maximal subset of linearly independent rows in the matrix $\begin{bmatrix} A \\ \mathbf{c} \end{bmatrix}$ and let $\mathbf{d}$ be the corresponding entries of $\begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$. Since D has full row rank, we may write $D = \begin{bmatrix} U & V \end{bmatrix}$, possibly after permuting coordinates, for some integer matrix U with $0 < |\det(U)|$. Then $\mathbf{x} = \begin{bmatrix} U^{-1}\mathbf{d} \\ 0 \end{bmatrix}$ satisfies $\mathbf{x} \in P$ as desired. By Cramer's Rule, $\mathbf{x}_i = \frac{\det(\widetilde{D})}{\det(D)}$, where $\widetilde{D}$ is obtained from D by replacing the i'th column with $\mathbf{d}$. Both the numerator and denominator of this fraction are integers of magnitude $\mathrm{poly}(b, c, \Delta)$, and hence each entry of $\mathbf{x}$ can be represented exactly in $O(1 + \frac{\log \Delta}{\log |A|})$ words. Thus, $\mathcal{H}$ can specify $\mathbf{x}$, and $\mathcal{V}$ can use the protocol above to check that indeed $A\mathbf{x} \leq \mathbf{0}$ and $\mathbf{c}^T\mathbf{x} = 1$.

$\square$ $\square$

Remark 3.6. *Notice that for each (multi)set fingerprinted in the above protocol, the fingerprint must be computed over a finite field large enough to represent the universe over which the (multi)set is defined. In particular, the size of the universe over which the multiset $S_X = \{(j, p_j, q_j) : |A_{\cdot j}|\}$ is defined depends on the magnitudes of the p_j and q_j values, which in turn depend on Δ. If Δ is not known to $\mathcal{V}$, it is sufficient for the prover to provide an upper bound on the p_j and q_j values before sending the multiset S_X, as this allows $\mathcal{V}$ to choose a sufficiently large finite field over which to fingerprint S_X.*

Rational Linear Programs. More generally, we are sometimes interested in solving linear programs with rational entries. Here, bounding subdeterminants is not sufficient to ensure the existence of an optimal solution which can be represented *exactly* with polynomial precision, or even the existence of an optimum of polynomial magnitude. Instead, we can have the prover send a rounded solution which is essentially optimal for a slightly perturbed linear program. We present full details in Appendix A.

3.4 TUM Integer Programs

For any graph problem that can be formulated as a linear program in which each entry of A, $\mathbf{b}$, and $\mathbf{c}$ can be derived as an linear function of the nodes and edges, we may view each edge in a graph stream $\mathcal{A}$ as providing an update to values of one or more entries of A, $\mathbf{b}$, and $\mathbf{c}$. Therefore, we immediately obtain a protocol for problems of this form via Theorem 3.5. More generally, we obtain protocols for problems formulated as *totally unimodular integer programs* (TUM IPs), which are integer programs for which $\Delta = 1$, since optimality of a feasible solution is shown by a matching feasible solution of the dual of its LP relaxation [18, Corollary 19.1a].

Corollary 3.7. *There is an $(|A|, 1)$ protocol for any graph problem that can be formulated as a TUM IP in which each entry of A, $\mathbf{b}$, and $\mathbf{c}$ is a linear function of the nodes and edges of graph.*

This follows immediately from Theorem 3.5 and the subsequent discussion: note that the linearity of the fingerprinting builds fingerprints of S_A, S_B and S_C, so $\mathcal{H}$ presents only their (aggregated) values, not information from the unaggregated graph stream.

Corollary 3.8. *Shortest s-t path, max-flow, min-cut, and minimum weight bipartite perfect matching (MWBPM) all have $(m, 1)$ protocols. For all four problems, a lower bound of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits holds for dense graphs.*

Proof. The upper bound follows from the previous corollary because all the problems listed possess formulations as TUM IPs and moreover the constraint matrix in each case has $O(m + n)$ non-zero entries. For example, for max-flow, $\mathbf{x}$ gives the flow on each edge, and the weight of each edge in the stream contributes (linearly) to constraints on the capacity of that edge, and the flow through incident nodes.

The lower bound for MWBPM, max-flow, and min-cut holds from [6, Theorem 11] which argues $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits for bipartite perfect matching, and straightforward reductions of bipartite perfect matching to all three problems, see e.g. [14, Theorem 7.37]. The lower bound for shortest s-t path follows from a straightforward reduction from INDEX, for which a lower bound linear in $\mathrm{hcost} \cdot \mathrm{vcost}$ was proven in [6, Theorem 3.1]. In the annotation setting, the INDEX problem is where the initial portion of the stream defines a binary string x. The final element of the stream is an index k, and the problem is to return the bit x_k.

Given an instance (x, k) of INDEX where $x \in \{0,1\}^{\binom{n}{2}}$, $k \in [\binom{n}{2}]$, we construct a graph G, with $V_G = [n + 2]$, and $E_G = E_A \cup E_B$. The edge set $E_A = \{(i, j) : x_{(i,j)} = 1\}$ is created from x alone, where, without loss of generality, we assume that x is indexed by edges (i, j) with $1 \le i < j \le n$. Then E_B is created from k alone, as $E_B = \{(n + 1, i), (j, n + 2)\}$ using $(i, j) = k$. Note that E_A and E_B can be created by $\mathcal{V}$ incrementally as the stream is seen, using $O(1)$ words of memory, to generate an implicit edge stream. The shortest path between nodes $n + 1$ and $n + 2$ is 3 if $x_k = 1$ and is 4 or more otherwise. Hence, solving the s-t path problem with $\mathrm{hcost} \cdot \mathrm{vcost} = o(n^2)$ would also solve the INDEX problem with this bound, contradicting the linear (in the length of x) bound from [6]. This also implies that any approximation within $\sqrt{4/3}$ requires $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ (better inapproximability constants may be possible). □ □

4 Trading off space and annotation size

In this section, we break the linear annotation barrier for a variety of problems. We first obtain optimal tradeoffs for dense integer matrix-vector multiplication, and use this result to obtain tradeoffs for dense linear and quadratic programs, as well as for eigenvalue computation. However, these results do not themselves yield efficient protocols for any of the totally unimodular graph problems considered earlier, as all four problems have sparse, rectangular constraint matrices. In Sections 4.5 and 4.6 we therefore present a different approach that allows us to obtain *optimal* tradeoffs between space and annotation length for shortest s-t path in graphs with small diameter, as well as non-trivial tradeoffs for the general shortest s-t path problem and MWBPM.

4.1 Tradeoffs for Linear Programs

Conceptually, the above protocols for solving the LP streaming problem are straightforward: $\mathcal{H}$ provides a primal solution, potentially repeating it once for each row of A to prove feasibility, and repeats the protocol for the dual. There are efficient protocols for the problems listed in the corollary since the constraint matrices of their IP formulations are sparse. For dense constraint matrices, however, the bottleneck is proving feasibility. We observe that computing $A\mathbf{x}$ reduces to computing b inner-product computations of vectors of dimension c. For any real $0 \leq \alpha \leq 1$, there are $(c^{\alpha}, c^{1-\alpha})$ protocols to verify such inner-products [6]. But we can further improve on this since one of the vectors is held constant in each of the tests. This reduces the space needed by $\mathcal{V}$ to run these checks in parallel; moreover, we prove a lower bound of $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(\min(c, b)^2)$ bits, and so obtain an *optimal* tradeoff for square matrices, up to logarithmic factors.

Theorem 4.1. *We are given a $b \times c$ integer matrix A and a c-dimensional integer vector $\mathbf{x}$, with all entries having absolute value polynomial in b and c. For any real $0 \leq \alpha \leq 1$, the product $A\mathbf{x}$ can be verified with a $(bc^{\alpha}, c^{1-\alpha})$ protocol. Moreover, any such protocol requires $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(\min(c, b)^2)$ bits for dense matrices.*

Proof. We begin with the upper bound. The protocol for verifying inner-products which follows from [6] treats a c-dimensional vector as an $h \times v$ array F, where $hv \geq c$. We associate each entry of F, and each entry of $[h]$ and $[v]$, with the corresponding element of a suitably large prime field $\mathbb{F}_p$ in the natural manner. Through interpolation, this then defines a two-variate polynomial f over $\mathbb{F}_p$, such that $f(x, y) = F_{x,y}$ for all $(x, y) \in [h] \times [v]$; f has degree $h - 1$ in variable x and degree $v - 1$ in variable y. For an inner-product between two vectors, we wish to compute $\sum_{x \in [h], y \in [v]} F_{x,y} G_{x,y} = \sum_{x \in [h], y \in [v]} f(x, y) g(x, y)$ for the corresponding arrays F, G and polynomials f, g. These polynomials can then be evaluated at locations outside $[h] \times [v]$, so in the protocol $\mathcal{V}$ picks a random position $r \in \mathbb{F}_p$, and evaluates $f(r, y)$ and $g(r, y)$ for $1 \leq y \leq v$. $\mathcal{H}$ then presents a degree $2(h-1)$ polynomial $s(x)$ which is claimed to be $\sum_{y=1}^{v} f(x, y) g(x, y)$. $\mathcal{V}$ checks that $s(r) = \sum_{y=1}^{v} f(r, y) g(r, y)$, and if so accepts $\sum_{x=1}^{h} s(x)$ as the correct answer.

In [6] it is shown how $\mathcal{V}$ can compute $f(r, y)$ efficiently as F is defined incrementally in the stream: each addition of w to a particular index is mapped to $(x, y) \in [h] \times [v]$, which causes $f(r, y) \leftarrow f(r, y) + wp(r, x, y)$, where $p(r, x, y)$ depends only on x, y, and r. Equivalently, the final value of $f(r, y)$ over updates in the stream where the jth update is $t_j = (w_j, x_j, y_j)$ is $f(r, y) = \sum_{t_j : y_j = y} w_j p(r, x_j, y)$.

To run this protocol over multiple vectors in parallel naively would require keeping the $f(r, y)$ values implied by each different vector separately, which would be costly. Our observation is that rather than keep these values explicitly, it is sufficient to keep only a fingerprint of these values, and use the linearity of fingerprint functions to finally test whether the polynomials provided by $\mathcal{H}$ for each vector together agree with the stored values.

In our setting, the $b \times c$ matrix A implies b two-variate polynomials, each of degree h in the first variable and v in the second. We evaluate each polynomial at (r, y) for $1 \leq y \leq v$ for the same value of r: since each test is fooled by $\mathcal{H}$ with small probability, the chance that none of them is fooled can be kept high by choosing the field to evaluate the polynomials over to have size polynomial in $b + c$. Thus, conceptually, the parallel invocation of b instances of this protocol require us to store $f_i(r, y)$ for $1 \leq y \leq v$ and $1 \leq i \leq b$ (for the b rows of A), as well as $f_{\mathbf{x}}(r, y)$ for $1 \leq y \leq v$ (where $f_{\mathbf{x}}$ is the polynomial derived from $\mathbf{x}$). Rather than store this set of bv values explicitly, $\mathcal{V}$ instead stores only v fingerprints, one for each value of y, where each fingerprint captures the set of b values of $f_i(r, y)$.

From the definition of our fingerprints, this means over stream updates $t_j = (w_j, i_j, x_j, y_j)$ of weight

w_j to row i_j and column indexed by x_j and y_j, $\mathcal{V}$ computes a fingerprint

$$\mathfrak{f}(A, y) = \sum_{i=1}^{b} f_i(r, y)\alpha^i = \sum_{i=1}^{b} \sum_{t_j : y_j = y, i_j = i} w_j p(r_j, x_j, y)\alpha^i,$$

for each y, $1 \le y \le v$. Observe that for each y this can be computed incrementally in the stream by storing only r and the current value of $\mathfrak{f}(A, y)$.

To verify the correctness, $\mathcal{V}$ receives the b polynomials s_i, and builds a fingerprint of the multiset of $S = \{i : s_i(r)\}$ incrementally. $\mathcal{V}$ then tests whether

$$\sum_{y=1}^{v} \mathfrak{f}(A, y) f_{\mathbf{x}}(r, y) = \mathfrak{f}(S).$$

To see the correctness of this, we expand the left hand side, as

$$\begin{aligned}\sum_{y=1}^{v} \mathfrak{f}(A, y) f_{\mathbf{x}}(r, y) &= \sum_{y=1}^{v} \big(\sum_{i=1}^{b} f_i(r, y)\alpha^i \big) f_{\mathbf{x}}(r, y) \\ &= \sum_{y=1}^{v} \big(\sum_{i=1}^{b} f_{\mathbf{x}}(r, y) f_i(r, y)\alpha^i \big) \\ &= \sum_{i=1}^{b} \big(\sum_{y=1}^{v} f_{\mathbf{x}}(r, y) f_i(r, y) \big)\alpha^i.\end{aligned}$$

Likewise, if all s_i's are as claimed, then

$$\mathfrak{f}(S) = \sum_{i=1}^{b} s_i(r)\alpha^i = \sum_{i=1}^{b} \left(\sum_{y=1}^{v} f_{\mathbf{x}}(r, y) f_i(r, y) \right) \alpha^i.$$

Thus, if the s_i's are as claimed, then these two fingerprints should match. Moreover, by the Schwartz-Zippel lemma, and the fact that α and r are picked randomly by $\mathcal{V}$ and not known to $\mathcal{H}$, the fingerprints will not match with high probability if the s_i's are *not* as claimed, when the polynomials are evaluated over a field of size polynomial in $(b + c)$.

To analyze the vcost, we observe that $\mathcal{V}$ can compute all fingerprints in $O(v)$ space. As $\mathcal{H}$ provides each polynomial $s_i(x)$ in turn, $\mathcal{V}$ can incrementally compute $\mathfrak{f}(S)$ and check that this matches $\sum_{y=1}^{v} \mathfrak{f}(A, y) f_{\mathbf{x}}(r, y)$. At the same time, $\mathcal{V}$ also computes $\sum_{i=1}^{b} \sum_{x=1}^{h} s_i(x)$, as the value of $A\mathbf{x}$. Note that if each s_i is sent one after another, $\mathcal{V}$ can forget each previous s_i after the required fingerprints and evaluations have been made; and if h is larger than v, does not even need to keep s_i in memory, but can instead evaluate it term by term in parallel for each value of x. Thus the total space needed by $\mathcal{V}$ is dominated by the fingerprints and check values.

Setting $h = c^{\alpha}$ and $v = c^{1-\alpha}$, the total size of the information sent by $\mathcal{H}$ is dominated by the b polynomials of degree $h = c^{\alpha}$.

To prove the lower bound, following the outline of Corollary 3.8, we give a simple reduction of INDEX to matrix-vector multiplication. Suppose we have an instance (x, k) of INDEX where $x \in \{0, 1\}^{n^2}$, $k \in [n^2]$. Without loss of generality, we assume that x is indexed canonically by node pairs (i, j) with $1 \le i \le j \le j$.

An $n \times n$ matrix A is constructed incrementally from x alone, in which $A_{i,j} = 1$ if $x_{(i,j)} = 1$, and $A_{i,j} = 0$ otherwise. Then a vector $\mathbf{x} \in \mathbb{R}^n$ is contructed from $k = (i, j)$ such that $\mathbf{x}_i = 1$ and all other entries of $\mathbf{x}$ are 0. Then the j'th entry of $A\mathbf{x}$ is 1 if and only if $x_{(i,j)} = 1$, and therefore the value of $x_{(i,j)}$ can be extracted from the vector $A\mathbf{x}$. Therefore, if we had an (h, v) protocol for verifying matrix-vector multiplication given an $n \times n$ matrix A (even for a stream in which all entries of A come before all entries of $\mathbf{x}$), we would obtain a $(\sqrt{h}, \sqrt{v})$ protocol for INDEX. The lower bound for matrix-vector multiplication thus holds by the lower bound for INDEX given in [6, Theorem 3.1]. □ □

Corollary 4.2. *Denote the maximum absolute value of the subdeterminants of integer-valued A by Δ, with an upper bound on Δ known in advance. Suppose $c \geq b$. Then for any real $0 \leq \alpha \leq 1$ there is a $(c^{1+\alpha}(1 + \frac{\log \Delta}{\log c}), c^{1-\alpha}(1 + \frac{\log \Delta}{\log c}))$ protocol for the LP streaming problem.*

Proof. First assume the optimum is finite. We can use the protocol of Theorem 4.1 to verify $A\mathbf{x} \leq \mathbf{b}$ and $A^T\mathbf{y} = \mathbf{c}$ within the protocol of Theorem 3.5. Some changes are needed, since Theorem 4.1 only applies to integer matrices and vectors, while $\mathbf{x}$ and $\mathbf{y}$ may be rational vectors (for the special case of TUM IPs, $\mathbf{x}$ and $\mathbf{y}$ are indeed integers, and the following protocol can be simplified).

We focus on verifying $A\mathbf{x} \leq \mathbf{b}$ as the protocol for verifying $A^T\mathbf{y} = \mathbf{c}$ is identical. We show that there exists an integer q such that the optimal solution $\mathbf{x}$ can be written in the form $\mathbf{x} = (1/q)\hat{\mathbf{x}}$ with $\hat{\mathbf{x}}$ an integral vector. $\mathcal{V}$ requires $\mathcal{H}$ to send $\hat{\mathbf{x}}$ and q, and uses the protocol of Theorem 4.1 to compute $A\hat{\mathbf{x}}$. While observing coordinate i of $A\hat{\mathbf{x}}$, $\mathcal{V}$ can compute $(A\hat{\mathbf{x}})_i/q = (A\mathbf{x})_i$ and compare this to $\mathbf{b}_i$.

Examining the proof of Theorem 3.5, $\mathbf{x}$ can be chosen such that all entries are quotients of subdeterminants of A. Moreover, the denominator of each quotient is the same over all i, i.e. all $\mathbf{x}_i = \frac{\det(\widetilde{U_i})}{\det(U)}$ are integer multiples of $\det(U) \leq \Delta$. Therefore, $\mathcal{H}$ can let $q = \det(U)$.

Observe that all entries of $A\hat{\mathbf{x}}$ are integers of absolute value at most u, for $u = \text{poly}(b, c, \Delta)$. It is therefore safe for $\mathcal{V}$ to use the protocol of Theorem 4.1 working over the finite field $\mathbb{F}_p$ where $p > 2u$ can be determined in advance of observing the data stream. That is, the natural map from $\{-u, -u + 1, \ldots, u\}$ to $\mathbb{F}_p$ is an injection, and hence $\mathcal{V}$ can determine for each i which integer corresponds to $A\hat{\mathbf{x}}_i$ (viewed as an element of $\mathbb{F}_p$), and check whether that integer is at most $q\mathbf{b}_i$.

The argument is essentially identical if the value of the LP is ∞ or $-\infty$, as explained in Theorem 3.5. For example, if the value is ∞, the protocol of Theorem 3.5 checks that $\mathbf{A}^T\mathbf{y} = 0$, $\mathbf{b}^T\mathbf{y} = -1$. $\mathbf{x}$ and can be chosen so that all entries are quotients of subdeterminants of $\begin{bmatrix} A^T \\ \mathbf{b}^T \end{bmatrix}$, and all subdeterminants of this matrix are integers of absolute value polynomial in Δ. Moreover, the denominator of each quotient is the same, so the proof proceeds as in the case where the LP has finite value.

Each element of the field requires $O(1 + \frac{\log \Delta}{\log c})$ words to specify, and the cost thus becomes $((bc^\alpha + cb^\alpha)(1 + \frac{\log \Delta}{\log c}), (c^{1-\alpha} + b^{1-\alpha})(1 + \frac{\log \Delta}{\log c}))$. If $c \geq b$, this is dominated by $(c^{1+\alpha}(1 + \frac{\log \Delta}{\log c}), c^{1-\alpha}(1 + \frac{\log \Delta}{\log c}))$ □ □

Rational Linear Programs. Corollary 4.2 extends to rational data with no constraints on the subdeterminants in the same way as Theorem A.2 follows from Theorem 3.5.

Corollary 4.3. *Suppose all entries of A, $\mathbf{b}$, and $\mathbf{c}$ are rational numbers p/q for $p, q \in \mathbb{Z}$. Assume the value of the linear program is finite, and there is a known polynomial upper bound on the length of an optimum. For any $0 < \epsilon_1 < 1/c$, there is an $\epsilon_2 = g(\mathcal{A})\epsilon_1$, with $g(\mathcal{A}) = \text{poly}(b, c)$ independent of ϵ_1, such that the following is true. If $c \geq b$, then for any real $0 \leq \alpha \leq 1$ there is a $(c^{1+\alpha}\frac{\log 1/\epsilon_1}{\log c}, c^{1-\alpha}\frac{\log 1/\epsilon_1}{\log c})$ protocol for for* *obtaining an additive-ϵ_2 approximation to the value of the perturbed primal LP* $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}+\epsilon_1\mathbf{1}\}$. *In particular, if* $\epsilon_1 = 1/\operatorname{poly}(c)$*, then we obtain a* $(c^{1+\alpha}, c^{1-\alpha})$ *protocol.*

Proof. We use the protocol of Theorem 4.1 to compute all matrix products within the protocol of Theorem A.2. All numbers arising in the protocol of Theorem A.2 are integer multiples of ϵ (where ϵ was an internal parameter of the proof of Theorem A.2), so A/ϵ, $\mathbf{x}/\epsilon$, and $\mathbf{y}/\epsilon$ are all integral. Hence we can compute $A\mathbf{x}/\epsilon^2$ and $A^T\mathbf{y}/\epsilon^2$ using Theorem 4.1, working over the finite field $\mathbb{F}_p$ with $p = \operatorname{poly}(b, c, 1/\epsilon_1)$ known in advance. From these values, $\mathcal{V}$ can extract $A\mathbf{x}$ and $A^T\mathbf{y}$. □ □

In the next two subsections, we use the above results to obtain tradeoffs between hcost and vcost for eigenvalue computation and quadratic programming.

4.2 Eigenvalue Verification.

Theorem 4.1 and its corollaries imply the existence of protocols for graph problems where *both* hcost and vcost are sublinear in the size of the input (for dense graphs). For example, we obtain an $(n^{1+\alpha\frac{\log 1/\epsilon_1}{\log n}}, n^{1-\alpha\frac{\log 1/\epsilon_1}{\log n}})$ protocol for verifying that λ is of an eigenvalue of the adjacency matrix A or the Laplacian L of G (within $\pm\epsilon_1$ for any desired precision $\epsilon_1 < 1/n$) as follows.

$\mathcal{H}$ provides a number $\hat{\lambda}$ as well as a vector $\hat{\mathbf{x}}$, normalized so that $1 \leq \|\hat{\mathbf{x}}\|_2^2 \leq 2$, and with all numbers integer multiples of $\epsilon = \frac{\epsilon_1}{n^{1/2}(u+2)}$. Here, $u \in \mathbb{Z}$ is a known upper bound on $\|A\|_\infty$ and $\|L\|_\infty$; note $u = n$ for A and $u = 2n$ for L suffice. $\mathcal{V}$ can check that $\epsilon_1/\epsilon \geq n^{1/2}(u+2)$, i.e., $\hat{\mathbf{x}}$ is provided at sufficiently high precision.

We now focus on A; the protocol for L is identical. Since all entries of $\mathbf{x}$ are integer multiples of ϵ, $\mathcal{V}$ can safely use the integer matrix-vector multiplication protocol of Theorem 4.1 to compute $\mathbf{y} = (1/\epsilon)A\hat{\mathbf{x}}$, working over a finite field $\mathbb{F}_p$ for a prime $p = \operatorname{poly}(n, 1/\epsilon)$. $\mathcal{H}$ annotates each coordinate of $\mathbf{y}$ with (a value claimed to be) $\hat{\mathbf{x}}_i$, which allows verifier to check that $|y_i - (1/\epsilon)\hat{\lambda}\hat{\mathbf{x}}_i| \leq u + 2$ for all i. $\mathcal{V}$ ensures $\hat{\mathbf{x}}_i$ is as claimed using fingerprints, and if all checks pass, $\mathcal{V}$ outputs $\hat{\lambda}$, and otherwise $\mathcal{V}$ outputs $\perp$.

We first argue that if, as claimed, $\hat{\lambda}$ and $\hat{\mathbf{x}}$ are actually an eigenvalue-eigenvector pair with all numbers rounded to the nearest integer multiple of ϵ, then $\mathcal{V}$ will output $\hat{\lambda}$ with probability 1. Indeed, $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \leq \epsilon$, and $|\hat{\lambda} - \lambda| \leq \epsilon$, where $\lambda\mathbf{x} = A\mathbf{x}$. Hence,

$$\begin{aligned}\|(1/\epsilon)A\hat{\mathbf{x}} - (1/\epsilon)\hat{\lambda}\hat{\mathbf{x}}\|_\infty &\leq (1/\epsilon)\big(\|A\hat{\mathbf{x}} - A\mathbf{x}\|_\infty + \|A\mathbf{x} - \lambda\mathbf{x}\|_\infty + \|\lambda\mathbf{x} - \hat{\lambda}\mathbf{x}\|_\infty\big)\\ &\leq \frac{1}{\epsilon}\big(u\epsilon + 0 + 2\epsilon\big) \leq u + 2.\end{aligned}$$

Now we argue that if $\mathcal{V}$'s checks pass, then with high probability $\hat{\lambda} = \lambda \pm \epsilon_1$ for some eigenvalue λ of A. Indeed, if $\mathcal{V}$'s checks pass, then with high probability $\|A\hat{\mathbf{x}} - \hat{\lambda}\hat{\mathbf{x}}\|_\infty \leq \epsilon(u+2) \leq \frac{\epsilon_1}{n^{1/2}}$. We show this implies there exists an eigenvalue λ of A such that $|\lambda - \hat{\lambda}| \leq \epsilon_1$. As A is symmetric, we may write $A = VDV^T$ for an orthogonal matrix V and matrix $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with the eigenvalues of A on the diagonal. Then

$$\begin{aligned}
\epsilon_1^2 &\geq n\|A\hat{\mathbf{x}} - \hat{\lambda}\hat{\mathbf{x}}\|_\infty^2 \\
&\geq \|A\hat{\mathbf{x}} - \hat{\lambda}\hat{\mathbf{x}}\|_2^2 \\
&= \|V(D - \hat{\lambda}I)V^T\hat{\mathbf{x}}\|_2^2 \\
&= \|(D - \hat{\lambda}I)\hat{\mathbf{x}}\|_2^2 \\
&= \sum_i \left((\lambda_i - \hat{\lambda})\hat{\mathbf{x}}_i\right)^2 \\
&\geq \left(\min_i |\lambda_i - \hat{\lambda}|\right)^2 \|\hat{\mathbf{x}}\|_2^2 \\
&\geq \left(\min_i |\lambda_i - \hat{\lambda}|\right)^2.
\end{aligned}$$

We conclude that there exists an i such that $|\lambda_i - \hat{\lambda}| \leq \epsilon_1$ as claimed.

4.3 Other Convex Programs

Our protocol for verifying the solution to a linear program relied on two main properties: strong duality, and the ability to compute the value of a solution $\mathbf{x}$ and check feasibility via matrix-vector multiplication. These properties also hold for more general convex optimization problems, such as quadratic programming and a large class of second-order cone programs [19], and we can therefore apply these techniques to large classes of mathematical programs. Such programs can, for example, be useful for applications in which weak peripheral devices or sensors must perform error correction of received communication, or in compressed sensing. See e.g. [20, 21].

However, unlike the linear programming case, there do not appear to be a reasonable set of sufficient conditions which ensure that we can *exactly* solve such general classes of programs. For example, the optimal solution of second order cone program $\min\{x^2 - 2 | x^2 \geq 2\}$ is irrational [22], and hence cannot be specified exactly in *any* finite amount of precision. As in Theorem A.2 and Corollary 4.3, our solution is to represent the optimum at some finite precision and observe that this does not introduce more than polynomial error in both the constraints and the objective function. For illustration, we describe in detail the protocol for Quadratic Programming.

Definition 4.4. *Consider a data stream $\mathcal{A}$ containing entries of vectors $\mathbf{b} \in \mathbb{Q}^b$, $\mathbf{c} \in \mathbb{Q}^c$, non-zero entries of a $c \times c$ positive-definite integer matrix Q, and a $b \times c$ integer matrix A in some arbitrary order, possibly interleaved. We assume the absolute value of all entries are polynomial in b and c. Each item in the stream indicates the index of the object it pertains to. The QP streaming problem on $\mathcal{A}$ is to determine the value of the quadratic program* $\min\{\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$.

In order to guarantee that the optimal solution is of polynomial length, we assume that the polyhedron $\{A\mathbf{x} \leq \mathbf{b}\}$ is bounded and all subdeterminants of A have absolute value polynomial in b and c; if this is the case then any feasible $\mathbf{x}$ satisfies $\|\mathbf{x}\|_\infty \leq b\|\mathbf{b}\|_\infty\Delta$ [23, p. 30]. Alternatively, we may assume there is a known upper bound on the length of the optimum. We obtain the following theorem, whose proof is similar to Theorem A.2 and is presented in Appendix B.

Theorem 4.5. *Assume all eigenvalues of Q are greater than $1/\operatorname{poly}(b, c)$. Also assume that the polyhedron $\{A\mathbf{x} \leq \mathbf{b}\}$ is bounded and all subdeterminants of A have absolute value polynomial in b and c, or alternatively that there is a known polynomial bound on the length of an optimum. For any $0 < \epsilon_1 < 1/c$, there* *is an $\epsilon_2 = g(\mathcal{A})\epsilon_1$, with $g(\mathcal{A}) = \mathrm{poly}(b, c)$ independent of ϵ_1, such that the following is true. If $c \geq b$, then for any real $0 \leq \alpha \leq 1$ there is a $(c^{1+\alpha \frac{\log 1/\epsilon_1}{\log c}}, c^{1-\alpha \frac{\log 1/\epsilon_1}{\log c}})$ protocol for obtaining an additive-ϵ_2 approximation to the value of the perturbed QP $\min\{\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} | A\mathbf{x} \leq \mathbf{b} + \epsilon_1\mathbf{1}\}$. In particular, if $\epsilon_1 = 1/\mathrm{poly}(c)$, then we obtain a $(c^{1+\alpha}, c^{1-\alpha})$ protocol. There is also a $((|A| + |Q|)^{\frac{\log 1/\epsilon_1}{\log c}}, \frac{\log 1/\epsilon_1}{\log c})$ protocol for this problem.*

4.4 Polynomial Agreement Protocol

Unfortunately, Theorem 4.1 and Corollary 4.2 are not sufficient to immediately result in improved protocols for any of the four totally unimodular graph problems we consider. The problem is that the constraint matrices of all four problems are sparse and rectangular, and the protocol of Theorem 4.1 is only efficient for dense (nearly) square matrices. Different ideas are required to break the linear annotation barrier for these problems. Specifically, we now describe an extension of the frequency moment protocol of [6] that is needed in order to obtain tradeoffs between annotation length and space usage for shortest s-t path and MWBPM.

The frequency moment protocol of [6] allows us to verify the function $\sum_i F_i^2$, where F_i is the number of times item i appeared in the stream. We will require the ability to compute $\sum_i g(X_i)$ for *arbitrary* functions $g : \mathbb{Z} \to \mathbb{Z}$ and vectors $X \in \mathbb{Z}^n$; the frequency moment protocol is clearly of this form, with $X_i = F_i$ and $g(X_i) = X_i^2$.

The protocol for verifying frequency moments from [6] treats a c-dimensional vector as an $h \times v$ array F, where $hv \geq c$. Through interpolation, this defines a two-variate polynomial f over a suitably large prime field $\mathbb{F}_p$, so that for all $x \in [h], y \in [v]$, $f(x, y) = F_{x,y}$. To compute the second frequency moment of a vector, we wish to compute $\sum_{x\in[h],y\in[v]} F_{x,y}^2 = \sum_{x\in[h],y\in[v]} f^2(x, y)$.

The polynomial f can then be evaluated at locations outside $[h] \times [v]$, so in the protocol $\mathcal{V}$ picks a random position r, and evaluates $f(r, y)$ for $1 \leq y \leq v$ ([6] shows how $\mathcal{V}$ can do this using v words of memory in a streaming manner). $\mathcal{H}$ then presents a degree $2(h - 1)$ polynomial $s(x)$ which is claimed to be $\sum_{y=1}^{v} f(x, y)^2$. $\mathcal{V}$ checks that $s(r) = \sum_{y=1}^{v} f(r, y)^2$, and if so accepts $\sum_{x=1}^{h} s(x)$ as the correct answer. The proof of validity follows from the Schwartz-Zippel lemma: if $s(x) \neq \sum_{y=1}^{v} f(x, y)^2$ as claimed by $\mathcal{H}$, then

$$\Pr[s(r) = \sum_{y=1}^{v} f(r, y)g(r, y)] \leq \frac{2(h-1)}{p}$$

where p is the size of the finite field.

We observe that there is nothing special about the function $f(x, y)^2$; given any degree-d polynomial g over $\mathbb{F}_p$, the protocol just described works for computing $\sum_{x\in[h],y\in[v]} g \circ f(x, y)$. $\mathcal{V}$ can evaluate $g \circ f(r, y)$ for $1 \leq y \leq v$ by computing each $f(r, y)$ while observing the stream as in [6], and then computing $g(f(r, y))$. Rather than send a degree $2(h - 1)$ polynomial $s(x)$ which is claimed to be $\sum_{y=1}^{v} f(x, y)^2$, $\mathcal{H}$ sends a degree $d(h - 1)$ polynomial $s(x)$ claimed to be $\sum_{y=1}^{v} g \circ f(x, y)$. $\mathcal{V}$ checks that $s(r) = \sum_{y=1}^{v} g \circ f(r, y)$, and if so accepts $\sum_{x=1}^{h} s(x)$ as the correct answer. By the Schwartz-Zippel lemma, if $s(x) \neq \sum_{y=1}^{v} g \circ f(x, y)$ as claimed by $\mathcal{H}$, then

$$\Pr[s(r) = \sum_{y=1}^{v} g \circ f(r, y)] \leq \frac{d(h-1)}{p}.$$

The result is a valid (dh, v) protocol for computing $\sum_{x\in[h],y\in[v]} g \circ f(x, y)$, since $\mathcal{V}$ requires v words of memory for storing each $f(r, y)$, and dh words of memory are required for $\mathcal{H}$ to send the degree $d(h-1)$ polynomial $s(x)$. To summarize:

Theorem 4.6 (Polynomial agreement protocol)**.** *Let $h, v \geq 0$ be such that $hv \geq n$. Let $\mathcal{A}$ be a data stream consisting of m updates, where the jth update is of the form (w_j, i_j), with $i_j \in [n]$ and w_j an integer of absolute value at most $u = \text{poly}(n, m)$ (w_j may be negative). Let F be the frequency vector of the stream. That is, $F_i = \sum_{j:i_j=i} w_j$ for all $i \in [n]$. We interpret each entry of F as an element of a finite field $\mathbb{F}_p$ in the obvious manner, where $p \geq m \cdot u$ is a prime.*

Then, for any degree-d polynomial g over $\mathbb{F}_p$, there is a (dh, v)-protocol for computing the function $\sum_{i=1}^{n} g(F_i)$.

4.5 Shortest s-t Path

We use the above polynomial agreement protocol in proving the following theorem. The key insight is that any dual solution is quite compact, requiring $O(n)$ words to specify, while a primal optimal solution can be succinctly specified by directly demonstrating a path which obtains the claimed length.

Theorem 4.7. *Given a graph G specified as a stream of weighted directed edges such that each edge appears at most once, let $d(s, w)$ denote the shortest-path distance from s to w in G, and let C_s be the set of nodes reachable from s. Let $d = \max_{w\in C_s} d(s, w)$ be the maximum distance from s to any node reachable from s. For any h, v such that $hv \geq dn^2$ and $h \geq dn$, there is a (h, v) protocol for shortest s-t path on directed graphs with non-negative integer edge weights.*

Proof. Our protocol handles graphs with non-negative integer edge weights; notice however that the lower bound of $hv = \Omega(n^2)$ from Corollary 3.8 applies even to unweighted graphs with constant diameter, so our protocol is optimal in this regime. We assume an upper bound on d is known in advance, and later show how to remove this assumption at the cost of a logarithmic factor in space, and no asymptotic increase in annotation.

To aid in the computation, $\mathcal{V}$ tracks properties of an (implicit) derived matrix X. Before observing the stream, $\mathcal{V}$ conceptually sets all entries of X to d, where d is the (assumed) upper bound on the distances. Then $\mathcal{V}$ sees the set of weights w_{ij} while observing the stream and treats each as an addition of $w_{ij} - d$ to entry (i, j) of X; this has the effect of setting $X_{ij} = w_{ij}$. This requires the assumption that each edge (i, j) appears at most once in the stream. At the end of the stream, $X_{ij} = w_{ij}$ if $(i, j) \in E$, and $X_{ij} = d$ otherwise. We note that it is straightforward for $\mathcal{V}$ to check in parallel that each edge appears at most once by tracking the matrix Y which counts the number of times each edge (i, j) is seen, and verifying that the squared Frobenius norm of Y, $\|Y\|_F^2 = \sum_{i,j} Y_{ij}^2$, satisfies $\|Y\|_F^2 = m$, using the F_2 protocol described above.

First we handle the case where an s-t path exists.

Upper bound on path length. To prove an upper bound on the value of the shortest path, $\mathcal{H}$ lists the edges in a valid s-t path P.

To compute the cost of P, the inner-product protocol of [6] is used to compute $P \cdot X$, where we treat P as an indicator matrix, i.e. $P_{ij} = 1$ iff (i, j) is an edge in P, and 0 otherwise. If P includes any edges (i, j) not present in E, then $X_{ij} = d$ and so these are charged at a cost of d. That is, the cost of P is made higher than the bound on distances, so it is easily detected if P contains edges not in E. This protocol requires $O(h)$ annotation and $O(v)$ space for any $hv \geq n^2$.

Lower bound on path length. To prove a lower bound on the value of the shortest path, we leverage the total unimodularity of the integer program for the problem. The dual integer program for shortest s-t path has a variable y_i for every $v \in V$ and a constraint for every edge $(i,j) \in E$:

$$\text{maximize } \mathbf{y}_t - \mathbf{y}_s \text{ subject to } \mathbf{y}_j - \mathbf{y}_i \leq w_{ij} \text{ for all } (i,j) \in E.$$

At a high level, $\mathcal{H}$ will prove a lower bound on the value of the shortest s-t path by presenting a feasible solution $\mathbf{y}$ to the above linear program. Importantly, we note that the solution is *compact*: $\mathbf{y}$ has only n variables. We present a carefully posed protocol allowing $\mathcal{V}$ to check that $\mathbf{y}$ satisfies all m constraints using sublinear annotation.

First, we show that there exists an optimal solution to the dual such that $\mathbf{y}_s = 0$ and all $\mathbf{y}$ are non-negative integers with $\mathbf{y}_i \leq d$ for all i, where $d = \max_{v \in C_s} d(s,v)$ and C_s is the set of nodes reachable from s. Let $\mathbf{y}_v = d$ for all v not reachable from s, and let $\mathbf{y}_v = d(s,v)$ if v is reachable from s. All dual constraints are satisfied by $\mathbf{y}$: if not, suppose $\mathbf{y}_j - \mathbf{y}_i > w(i,j)$ for some edge (i,j). Then clearly $\mathbf{y}_i < d$ since edge weights are non-negative and $\mathbf{y}_j \leq d$, and hence i is reachable from s. But then j is reachable from s as well, and this contradicts that $\mathbf{y}_j = d(s,j)$, as there is a path from s to j of cost $\mathbf{y}_i + w(i,j) < \mathbf{y}_j$.

Specifying $\mathbf{y}$ requires $O(n)$ words of annotation since there are n dual variables (more precisely, it requires n' words where $n' = |C_s|$, since there are only n' variables not set to d). $\mathcal{V}$ immediately outputs $\perp$ if any variable $\mathbf{y}_i$ in the solution is non-integral, if $\mathbf{y}_s \neq 0$, or if $\mathbf{y}_v > d$ for any v.

Given the dual assignment $\mathbf{y}$, let $W \in \mathbb{Z}^{n^2}$ be the matrix defined by

$$W_{ij} = w_{ij} - \mathbf{y}_j + \mathbf{y}_i \text{ if } (i,j) \in E \text{ and } W_{ij} = d - \mathbf{y}_j + \mathbf{y}_i \text{ if } (i,j) \notin E.$$

It is clear that the $\mathbf{y}_i$'s constitute a feasible assignment to the dual if and only if $W_{ij} \geq 0$ for all $(i,j) \in E$: if $(i,j) \in E$, $W_{ij} \geq 0$ only if the constraint corresponding to edge (i,j) is satisfied, and if $(i,j) \notin E$, the addition of d to W_{ij} ensures $X_{ij} \geq 0$, which corresponds to "no constraint". We also observe that $W_{ij} = X_{ij} - y_j + y_i$ for X as described above.

Let w_{max} be the heaviest edge in G. We can assume $w_{\text{max}} \leq d$ since $\mathcal{V}$ can filter away any edges with $w_{\text{max}} > d$, as these edges will not effect the value of the shortest $s-t$ path.

We apply the polynomial agreement protocol of Theorem 4.6 to W, using the lowest-degree polynomial g over $\mathbb{F}_p$ such that $g(x) = 0$ for $x \in \{1, \ldots, d + w_{\text{max}}\}$ and $g(x) = 1$ for $x \in \{-d, \ldots, 0\}$. g has degree $w_{\text{max}} + 2d = O(d)$, and clearly $\sum_{i,j \in [n]} g(w_{i,j}) = 0$ if and only if $\mathbf{y}$ is feasible for the dual LP, as for all (i,j), $-d \leq W_{ij} \leq w_{\text{max}} + d$. The cost of the polynomial agreement protocol is thus $O(dh)$ annotation and $O(v)$ space for any $hv \geq n^2$. Lastly, note that $\mathcal{V}$ can apply the polynomial agreement protocol on the matrix X derived from the stream, and updated by performing the necessary additions and subtractions of y_i and y_j values to all affected coordinates.

We now remove the assumption that d is known in advance, at the cost of a logarithmic increase in space. At a high level, while observing the stream $\mathcal{V}$ can keep logarithmically many "guesses" for the value of d, and after the stream is seen, $\mathcal{H}$ can tell $\mathcal{V}$ which guess is the tightest upper bound on the value of all variables in the optimal solution to the dual LP. Then $\mathcal{V}$ can forget about the other guesses, and simply complete the execution of the protocol corresponding to the best guess.

More formally, it suffices for $\mathcal{V}$, while observing the stream, to run $O(\log n)$ instances of the above protocol in parallel, with the i'th instance run with parameter $d = 2^i$. This ensures that one instance will be run with parameter $d \leq \max_i \mathbf{y}_i < 2d$. We require $\mathcal{H}$ to prepend the annotation with the value $d^* = \min\{2^i : \mathbf{y}_j \leq 2^i \text{ for all } j\}$. $\mathcal{V}$ then only needs to continue the instance of the protocol run with parameter $d = d^*$. If d^* is not as claimed, $\mathcal{V}$ will detect this when a dual variable $\mathbf{y}_j$ is presented with $\mathbf{y}_j > d^*$, and output $\perp$. Thus, the protocol is valid. The space cost increases by a logarithmic factor compared to when the true value of d is known in advance, since $\mathcal{V}$ must run $O(\log n)$ instances of the protocol while observing the stream. The annotation cost does not increase asymptotically, since the only instance of the protocol $\mathcal{V}$ continues to run after the stream has been observed satisfies $2d \le \max_i \mathbf{y}_i$. That is, the instance is run with a "guess" for d that was within a factor of two of the true value of d.

No path from s to t. If the shortest s-t path is infinite (there is no s-t path), let $C_t \subseteq V$ be the connected component of t. Then the dual assignment with $\mathbf{y}_i = 1$ for $i \in C_t$ and $\mathbf{y}_i = 0$ for all other i satisfies $\mathbf{y}_i - \mathbf{y}_j = 0$ for all $(i,j) \in E$, and the value of the dual objective function $\mathbf{y}_t - \mathbf{y}_s$ is positive. By Farkas' Lemma, this serves as a witness to the fact that the primal is infeasible. $\mathcal{V}$ can check $\mathbf{y}$ is as claimed by running the polynomial agreement protocol on the vector Y with $Y_{ij} = \mathbf{y}_i - \mathbf{y}_j$ if $(i,j) \in E$, and $Y_{ij} = \mathbf{y}_i - \mathbf{y}_j - 3$ if $(i,j) \notin E$ and using the degree-5 polynomial g over $\mathbb{F}_p$ such that $g(0) = g(-4) = g(-3) = g(-2) = 0$ and $g(-1) = g(1) = 1$. We can construct the derived stream in the same manner as X and W above. It is clear that if $(i,j) \notin E$ then $Y_{ij} \in \{-4,-3,-2\}$, if $(i,j) \in E$ with $\mathbf{y}_i - \mathbf{y}_j = 0$ then $Y_{ij} = 0$, and otherwise $Y_{ij} \in \{-1, 1\}$. Thus, $\sum_{i,j} g(Y_{ij}) = 0$ if and only if $\mathbf{y}$ is as claimed; this instance of the polynomial agreement protocol requires annotation $O(h)$ and space $O(v)$ for any $hv \ge n^2$. □ □

Remark 4.8. *If d is not known in advance, then neither $\mathcal{H}$ nor $\mathcal{V}$ knows the annotation cost of the protocol of Theorem 4.7 until after observing the stream. Only the space usage v can be fixed in advance in this case, and the annotation cost will be $O(n^2 d/v)$.*

We note that the protocol of Theorem 4.7, as well as Theorem 4.9 below, does not handle edge weights which are specified incrementally. The reason is that $\mathcal{V}$ must be able to derive a stream specifying the matrices X and W, which we only know how to do in the absence of duplicate edges. This is in contrast to the earlier protocols of Theorems 3.5, A.2, and 4.1, as well as their corollaries, which work even when edge weights are specified incrementally.

4.6 Minimum Weighted Bipartite Perfect Matching (MWBPM)

Theorem 4.9. *Let w_{max} be the heaviest edge in G. For any h, v such that $hv \ge n^3 w_{max}$ and $h \ge n w_{max}$, there is a (h, v) protocol for MWBPM with non-negative integer edge weights. Both* hcost *and* vcost *can be chosen sublinear in the stream length if the number of edges m satisfies $m = \omega(n^{3/2} w_{max}^{1/2})$.*

Proof. Upper bound. To prove an upper bound on the value of the minimum weight perfect matching, $\mathcal{H}$ sends the edges in a valid perfect matching M. It is straightforward for $\mathcal{V}$ to store M and verify that it is perfect matching over n nodes in $O(n)$ space. As in the previous protocol, $\mathcal{V}$ can compute the cost of this matching as an inner product $M \cdot X$, where X_{ij} is set to w_{ij} if (i,j) is an edge, or $2nw_{\max}$ otherwise. Hence, if M includes edges not present in E, it will have excessively high cost, and can be rejected. $\mathcal{V}$ can check M is a perfect matching by comparing a fingerprint of the set $\{1,\ldots,n\}$ to that of the (multi)set of nodes incident to an edge in M. If the fingerprints match, then with high probability, each node in n is incident to exactly one edge in M.

Lower bound. A lower bound on the cost of the optimal matching is proven via a feasible solution to the dual linear program. The dual is given by:

$$\text{maximize} \sum_{i \in A} \mathbf{y}_i + \sum_{j \in B} \mathbf{y}_j$$

$$\text{subject to } \mathbf{y}_i + \mathbf{y}_j \le w_{ij} \text{ for all } (i,j) \in E,$$

where A and B are the two sides of the bipartition of G, and w_{ij} is the cost of edge (i, j). Given a dual solution $\mathbf{y} \in \mathbb{Z}^n$, let $X \in \{0, 1\}^{n^2}$ be the vector with $X_{ij} = \mathbf{y}_i + \mathbf{y}_j - w_{ij}$ if $(i, j) \in E$ and $X_{ij} = \mathbf{y}_i + \mathbf{y}_j - 2d'$ otherwise, where d' is an upper bound on the value of any variable $\mathbf{y}$. We show below $d' = nw_{\text{max}}$ is sufficient. $\mathbf{y}$ is a feasible solution to the dual if and only if all entries of X are less than or equal to zero.

The protocol now proceeds essentially identically to that of Theorem 4.7, although here we can only guarantee the existence of a a dual-optimal assignment $\mathbf{y}$ with $|\mathbf{y}_i| \leq nw_{\text{max}}$ for all i. This results in increased annotation requirements compared to those of Theorem 4.7. We remark that this bound is tight, in that there are graphs for which any dual-optimal solution $\mathbf{y}$ has $|\mathbf{y}_i| = \Omega(nw_{\text{max}})$ for some i; one such example is a simple path on n vertices, with $w_{i,i+1} = w_{\text{max}}$ if i is odd and $w_{i,i+1} = 0$ if i is even.

To argue that there always exists a dual-optimal $\mathbf{y}$ with $|\mathbf{y}_i| \leq nw_{\text{max}}$ for all i, notice it follows from the argument in Theorem 3.5 that there exists a dual optimum $\mathbf{y}$ with $\mathbf{y}_i = \frac{\det(\widetilde{U_i})}{\det(U)}$ for some submatrix U of the constraint matrix of the dual, where $\widetilde{U_i}$ obtained from U by replacing the i'th column with the vector $\mathbf{w}$ of edge weights. By total unimodularity of the dual program, $\det(U) = \pm 1$, and hence $|\mathbf{y}_i| \leq |\det(\widetilde{U_i})| \leq nw_{\text{max}}$, where the last inequality can be seen by performing cofactor expansion along the i'th column of $\widetilde{U_i}$.

To conclude, we apply the polynomial agreement protocol to the vector X with $X_{ij} = \mathbf{y}_i + \mathbf{y}_j - w_{ij}$ if $(i, j) \in E$ and $X_{ij} = \mathbf{y}_i + \mathbf{y}_j - 2nw_{\text{max}}$ otherwise. $\mathcal{V}$ can construct a *derived* stream defining X just as in Theorem 4.7, and the polynomial agreement protocol is applied using a polynomial g such that $g(x) = 1$ for $x \in \{1, \ldots, 2nw_{\text{max}}\}$ and $g(x) = 0$ for $x \in \{-4nw_{\text{max}}, \ldots, 0\}$. g has degree $O(nw_{\text{max}})$, and $\sum_{i,j \in [n]} g(w_{i,j}) = 0$ if and only if $\mathbf{y}$ is feasible for the dual LP. This protocol has $\text{hcost} = O(hnw_{\text{max}})$ and $\text{vcost} = O(v)$ for for any $hv = \Omega(n^2)$ and $h \geq nw_{\text{max}}$.

If no perfect matching exists, Farkas' Lemma implies this can be proven by demonstrating a dual solution $\mathbf{y}$ such that $\mathbf{y}_i + \mathbf{y}_j = 0$ for all $(i, j) \in E$, and $\sum_{i \in A} \mathbf{y}_i + \sum_{j \in B} \mathbf{y}_j > 0$. $\mathcal{V}$ can check this similarly to the protocol of Theorem 4.7 when no s-t path exists. $\square$

5 Simulating Non-Streaming Algorithms

Next, we give protocols by appealing to known non-streaming algorithms for graph problems. At a high level, we can imagine the helper running an algorithm on the graph, and presenting a "transcript" of operations carried out by the algorithm as the proof to $\mathcal{V}$ that the final result is correct. Equivalently, we can imagine that $\mathcal{V}$ runs the algorithm, but since the data structures are large, they are stored by $\mathcal{H}$, who provides the contents of memory needed for each step. There may be many choices of the algorithm to simulate and the implementation details of the algorithm: our aim is to choose ones that result in smaller annotations.

To make this concrete, consider the case of requiring the graph to be presented in a particular order, such as depth first order. Starting from a given node, the exploration retrieves nodes in order, based on the pattern of edges. Assuming an adjacency list representation, a natural implementation of the search in the traditional model of computation maintains a stack of edges representing the current path being explored. Edges incident on the current node being explored are pushed, and pops occur whenever all nodes connected to the current node have already been visited. $\mathcal{H}$ can allow $\mathcal{V}$ to recreate this exploration by providing at each step the next node to push, or the new head of the stack when a pop occurs, and so on. To ensure the correctness of the protocol, additional checking information can be provided, such as pointers to the location in the stack when a node is visited that has already been encountered.

With care, this idea of "augmenting a transcript" of a traditional algorithm can be made to work on an algorithm-by-algorithm basis. However, while the resulting protocols are lightweight, it rapidly becomes tedious to provide appropriate protocols for other computations based on this idea. Instead, we introduce a more general approach which argues that any (deterministic) algorithm to solve a given problem can be converted into a protocol in our model. The running time of the algorithm in the RAM model becomes the size of the proof in our setting.

Our main technical tool is the off-line memory checker of Blum et al. [11], which we use to efficiently verify a sequence of accesses to a large memory. Consider a *memory transcript* of a sequence of read and write operations to this memory (initialized to all zeros). Such a transcript is *valid* if each read of address i returns the last value written to that address. The protocol of Blum et al. requires each read to be accompanied by the timestamp of the last write to that address; and to treat each operation (read or write) as a read of the old value followed by the write of a new value. Then to ensure validity of the transcript, it suffices to check that a fingerprint of all write operations (augmented with timestamps) matches a fingerprint of all read operations (using the provided timestamps), along with some simple local checks on timestamps. Consequently, any valid (timestamp-augmented) transcript is accepted by $\mathcal{V}$, while any invalid transcript is rejected by $\mathcal{V}$ with high probability.

We use this memory checker to obtain the following general simulation result.

Theorem 5.1. *Suppose P is a graph problem possessing a non-randomized algorithm $\mathcal{M}$ in the random-access memory model that, when given $G = (V, E)$ in adjacency list or adjacency matrix form, outputs $P(G)$ in time $t(m, n)$, where $m = |E|$ and $n = |V|$. Then there is an $(m + t(m, n), 1)$ protocol for P.*

Proof. $\mathcal{H}$ first repeats (the non-zero locations of) a valid adjacency list or matrix representation G, as writes to the memory (which is checked by $\mathcal{V}$); $\mathcal{V}$ uses fingerprints to ensure the edges included in the representation precisely correspond to those that appeared in the stream, and can use local checks to ensure the representation is otherwise valid. This requires $O(m)$ annotation and effectively initializes memory for the subsequent simulation. Thereafter, $\mathcal{H}$ provides a valid augmented transcript T' of the read and write operations performed by algorithm $\mathcal{M}$, which is checked by $\mathcal{V}$ using the memory checking protocol of [11]. $\mathcal{V}$ rejects if the memory checking protocol rejects T', or if any read or write operation executed in T' does not agree with the prescribed action of $\mathcal{M}$. As only one read or write operation is performed by $\mathcal{M}$ in each timestep, the length of T' is $O(t(m, n))$, resulting in an $(m + t(m, n), 1)$ protocol for P. □ □

Although Theorem 5.1 only allows the simulation of deterministic algorithms, $\mathcal{H}$ can non-deterministically "guess" an optimal solution S and prove optimality by invoking Theorem 5.1 on a (deterministic) algorithm that merely checks whether S is optimal. Unsurprisingly, it is often the case that the best-known algorithms for verifying optimality are more efficient than those finding a solution from scratch (see e.g. the MST protocol below), and this gives the simulation theorem considerable power.

Theorem 5.2. *There is a valid $(m, 1)$ protocol to find a minimum cost spanning tree; a valid $(m+n \log n, 1)$ protocol to verify single-source shortest paths; and a valid $(n^3, 1)$ protocol to verify all-pairs shortest paths.*

Proof. We first prove the bound for MST. Given a spanning tree T, there exists a linear-time algorithm $\mathcal{M}$ for verifying that T is minimum (see e.g. [24]). Let $\mathcal{M}'$ be the linear-time algorithm that, given G and a subset of edges T in adjacency matrix form, first checks that T is a spanning tree by ensuring $|T| = n - 1$ and T is connected (by using e.g. breadth-first search), and then executes $\mathcal{M}$ to ensure T is minimum. We obtain an $(m, 1)$ protocol for MST by having $\mathcal{H}$ provide a minimum spanning tree T and using Theorem 5.1 to simulate algorithm $\mathcal{M}'$.

The upper bound for single-source shortest path follows from Theorem 5.1 and the fact that there exist implementations of Djikstra's algorithm that run in time $m + n \log n$. The upper bound for all-pairs shortest

paths also follows from Theorem 5.1 and the fact that the Floyd-Warshall algorithm runs in time $O(n^3)$. $\square$
$\square$

We now provide near-matching lower bounds for all three problems.

Theorem 5.3. *Any protocol for verifying single-source or all pairs shortest paths requires* $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ *bits. Additionally, if edge weights may be specified incrementally, then an identical lower bound holds for MST.*

Proof. The lower bounds for single-source and all-pairs shortest paths are inherited from shortest s-t path (Corollary 3.8).

To prove the lower bound for MST, we present a straightforward reduction from an instance of the problem INDEX, (x, k), where $x \in \{0,1\}^{\binom{n}{2}}$, $k \in [\binom{n}{2}]$. As in Corollary 3.8, we show how to construct a graph from x and k so that solving MST on this graph reveals the answer to the instance of INDEX. We construct graph G incrementally from x and k, with $V_G = [n]$. Without loss of generality, we assume that x is indexed canonically by node pairs (i, j) with $i \neq j$, so $k = (k_1, k_2)$. We write (i, j, w) to denote an edge between nodes i and j with weight w. The edge set $E_A = \{(i, j, 1) : x_{(i,j)} = 1\}$ is created from x alone, using $O(1)$ words of memory. Likewise, $E_B = \{(u, v, n) : (u, v) \neq k\}$ is created from k alone; intuitively this increments the weight of all edges by n, except for the "k'th" edge of G. We consider $E_G = E_A \cup E_B$; if edge (u, v) is in $E_A \cap E_B$ then we interpret the weight of (u, v) in E_G to be the *sum* of its weights in E_A and E_B.

Studying the combination of these two edgesets, we observe that all edges in E_G except edge k have weight at least n and at most $n + 1$, while the k'th edge has weight 1 if $x_k = 1$ and does not exist in E_G if $x_k = 0$. Thus, if $x_k = 0$, the cost of the minimum spanning tree of G is at least $n(n-1) = n^2 - n$, while if $x_k = 1$, the cost of the minimum spanning tree is at most $(n+1)(n-2) + 1 = n^2 - n - 1$.

Thus, by determining the cost of the minimum spanning tree, we can determine whether $x_k = 0$. Therefore, from the hardness of INDEX, we must have the claimed bound, linear in the length of x. $\square$ $\square$

Diameter. The diameter of G can be verified by simulating any known algorithm for the problem via Theorem 5.1, but the next protocol improves over the memory checking approach. Indeed, the best known algorithm for computing diameter even in unweighted *undirected* graphs requires time $O(n^\omega)$ where ω is the exponent of matrix multiplication [25, 26]. As the best known bounds on n^ω are currently polynomially larger than $O(n^2 \log n)$, the following theorem cannot be obtained by direct application of the memory checking approach to any known algorithm.

Theorem 5.4. *There is a valid* $(n^2 \log n, 1)$ *protocol for computing graph diameter in unweighted directed graphs. Further, any protocol for diameter requires* $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ *bits.*

Proof. [6, Theorem 5.2] gives an $(n^2 \log l, 1)$ protocol for verifying that $A^l = B$ for a matrix A presented in a data stream and for any positive integer l. Note that if A is the adjacency matrix of G; then $(I + A)^l_{ij} \neq 0$ if and only if there is a path of length at most l from i to j. Therefore, the diameter of G is equal to the unique $l > 0$ such that $(I + A)^l_{ij} \neq 0$ for all (i, j), while $(I + A)^{l-1}_{ij} = 0$ for some (i, j). Our protocol requires $\mathcal{H}$ to send l to $\mathcal{V}$, and then run the protocol of [6, Theorem 5.2] twice to verify that l is as claimed. Since the diameter is at most $n - 1$, this gives an $(n^2 \log n, 1)$ protocol.

We prove the lower bound via a reduction from an instance of INDEX, (x, k), where $x \in \{0,1\}^{n^2/4}$, $k \in [n^2/4]$. A bipartite graph $G = (V, E)$ is created from x alone: it includes edge (i, j) in E if and only if $x_{(i,j)} = 1$, again using the convention that x is indexed by edges. We also add to G two nodes L and R, with edges from L to each node in the left side of the bipartition, edges from R to each node in the right side of the bipartition, and an edge between L and R. This ensures that the graph is connected, with diameter at most 3. Finally, given $k = (k_1, k_2)$, we append a path of length 2 to node k_1, and a path of length 2 to node k_2. If $x_k = 0$, then the diameter is now 7, while if $x_k = 1$, the diameter is 5. The lower bound follows from the hardness of INDEX [6, Theorem 3.1] (this also shows that any protocol to approximate diameter better than $\sqrt{1.4}$ requires $\mathrm{hcost} \cdot \mathrm{vcost} = \Omega(n^2)$ bits). □ □

6 Conclusion and Future Directions

In this paper, we showed that a host of graph problems possess streaming protocols requiring only constant space and linear-sized annotations. For many applications of the annotation model, the priority is to minimize vcost, and these protocols achieve this goal. However, these results are qualitatively different from those involving numerical streams in the earlier work [6]: for the canonical problems of heavy hitters, frequency moments, and selection, it is trivial to achieve an $(m, 1)$ protocol by having $\mathcal{H}$ replay the stream in sorted ("best") order. The contribution of [6] is in presenting protocols obtaining optimal tradeoffs between hcost and vcost in which both quantities are sublinear in the size of the input. There are good reasons to seek these tradeoffs. For example, consider a verifier with access to a few megabytes or gigabytes of working memory. If an $(m, 1)$ protocol requires only a few kilobytes of space, it would be desirable to use more of the available memory to significantly reduce the running time and communication cost of the verification protocol. Achieving a $(\sqrt{n}, \sqrt{n})$ protocol would potentially allow both space and annotation costs to be under a megabyte, even when the stream contains terabytes of data.

In contrast to [6], it is non-trivial to obtain $(m, 1)$ protocols for the graph problems we consider. Nonetheless, we obtain tradeoffs involving sublinear values of hcost and vcost for several important problems: matrix-vector multiplication, computing eigenvalues of the Laplacian, shortest s-t path and MWBPM. We thus leave as an open question whether it is possible to obtain such tradeoffs for a wider class of graph problems, and in particular if the use of memory checking can be adapted to provide tradeoffs. Showing such tradeoffs are *impossible* seems to require fundamentally new techniques in communication complexity. Particularly important problems left to characterize are flow problems and minimum spanning tree.

A final open problem is to ensure that the work of $\mathcal{H}$ is scalable. In motivating settings such as cloud computing environments, the data is very large, and $\mathcal{H}$ may represent a *distributed* cluster of machines. We leave open the question of demonstrating that these protocols can be executed in a model such as the MapReduce framework.

Acknowledgements. We thank Moni Naor for suggesting the use of memory checking as a general approach to generating protocols. We also thank Dorit Hochbaum, Dimitri Bertsekas, Thomas Steinke, and Varun Kanade for helpful discussions, and the anonymous reviewers for providing helpful suggestions for improving the clarity and completeness.

References

[1] A. McGregor, "Graph mining on streams," in *Encyc. of Database Systems*. Springer, 2009.

[2] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, "On graph problems in a semi-streaming model," *Theor. Comput. Sci.*, vol. 348, no. 2, pp. 207–216, 2005.

[3] C. Demetrescu, I. Finocchi, and A. Ribichini, "Trading off space for passes in graph streaming problems," in *SODA*, 2006, pp. 714–723.

[4] G. Aggarwal, M. Datar, S. Rajagopalan, and M. Ruhl, "On the streaming model augmented with a sorting primitive," in *FOCS*, 2004, pp. 540–549.

[5] A. Das Sarma, R. J. Lipton, and D. Nanongkai, "Best-order streaming model," in *Theory and Applications of Models of Computation*, 2009, pp. 178–191.

[6] A. Chakrabarti, G. Cormode, and A. Mcgregor, "Annotations in data streams," in *ICALP*, 2009, pp. 222–234.

[7] R. J. Lipton, "Efficient checking of computations," in *STACS*, 1990, pp. 207–215.

[8] M. Yiu, Y. Lin, and K. Mouratidis, "Efficient verification of shortest path search via authenticated hints," in *ICDE*, 2010.

[9] K. L. Clarkson and D. P. Woodruff, "Numerical linear algebra in the streaming model," in *STOC*, 2009, pp. 205–214.

[10] T. Sarlos, "Improved approximation algorithms for large matrices via random projections," in *IEEE FOCS*, 2006.

[11] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor, "Checking the correctness of memories," *Algorithmica*, pp. 90–99, 1995.

[12] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[13] R. J. Lipton, "Fingerprinting sets," Princeton University, Tech. Rep. Cs-tr-212-89, 1989.

[14] J. M. Kleinberg and É. Tardos, *Algorithm design*. Addison-Wesley, 2006.

[15] A. Schrijver, *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, Jul. 2004.

[16] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Oper. Res.*, vol. 34, pp. 250–256, March 1986.

[17] D. S. Hochbaum and J. G. Shanthikumar, "Convex separable optimization is not much harder than linear optimization," *J. ACM*, vol. 37, pp. 843–862, October 1990.

[18] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley, Jun. 1998.

[19] D. Bertsekas, *Convex Optimization Theory*. Athena Scientific.

[20] E. J. Candès and P. A. Randall, "Highly robust error correction by convex programming," *CoRR*, vol. abs/cs/0612124, 2006.

[21] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[22] D. S. Hochbaum, "Personal communication," 2011.

[23] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982.

[24] V. King, "A simpler minimum spanning tree verification algorithm," *Algorithmica*, vol. 18, no. 2, pp. 263–270, 1997.

[25] R. Seidel, "On the all-pairs-shortest-path problem in unweighted undirected graphs," *J. Comput. Syst. Sci.*, vol. 51, pp. 400–403, December 1995.

[26] R. Yuster, "Computing the diameter polynomially faster than apsp," *CoRR*, vol. abs/1011.6181, 2010.

A Linear Programs with Rational Entries

One might hope that Theorem 3.5 extends naturally to linear programs with rational entries. The following simple variant of Example 3.4 demonstrates that with rational data it is no longer sufficient to assume all subdeterminants of the constraint matrix are bounded in absolute value.

Example A.1. *Consider the linear program of Example 3.4, with A replaced by the matrix*

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1/2 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1/2 & \dots & 0 & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \dots & -1 & 1/2 \end{bmatrix}.$$

Observe that all subdeterminants of this matrix are bounded (above) in absolute value by 1. Exactly as in Example 3.4, the unique feasible point is the vector $\mathbf{x} \in \mathbb{Z}^c$ with i'th coordinate equal to 2^i, and the value of the linear program is $\Omega(2^c)$. $\mathcal{V}$ cannot manipulate quantities of such magnitude exactly with less than linear space. □

However, context often provides an *a priori* bound on the value of the program and the length of the optima, especially in applications to combinatorial optimization. Still, without sufficiently strong assumptions we cannot guarantee that there is an optimal solution that can be exactly specified in small precision, and $\mathcal{H}$ must therefore send an approximate representation of the optimum. This rounded optimum is not guaranteed to be feasible for the exact LP, but will be essentially optimal for a very slightly perturbed LP.

Theorem A.2. *Suppose all entries of A, $\mathbf{b}$, and $\mathbf{c}$ are rational numbers p/q for $p, q \in \mathbb{Z}$. Assume the value of the linear program is finite, and there is a known polynomial upper bound on the length of an optimum. For any $0 < \epsilon_1 < 1/|A|$, there is an $\epsilon_2 = g(\mathcal{A})\epsilon_1$, with $g(\mathcal{A}) = \operatorname{poly}(b, c)$ independent of ϵ_1, such that the following is true. There is a valid $(|A|\frac{\log 1/\epsilon_1}{\log |A|}, \frac{\log 1/\epsilon_1}{\log |A|})$ protocol for obtaining an additive-ϵ_2 approximation to the value of the perturbed primal LP $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b} + \epsilon_1 \mathbf{1}\}$. In particular, if $\epsilon_1 = 1/\operatorname{poly}(|A|)$, then we obtain an $(|A|, 1)$ protocol.*

Proof. First, suppose all entries of A, $\mathbf{b}$, and $\mathbf{c}$ are integer multiples of ϵ for $\epsilon = 1/\operatorname{poly}(b, c, \epsilon_1^{-1})$ to be determined; we remove this assumption later.

Protocol Specification. Notice that all entries of A, $\mathbf{b}$ and $\mathbf{c}$ are elements of a universe of size $\text{poly}(b, c, \epsilon_1^{-1})$, and $\mathcal{V}$ can fingerprint all multisets as in Theorem 3.5 using a finite field $\mathbb{F}_p$ where $p = \text{poly}(b, c, \epsilon_1^{-1})$. These fingerprints require $O(\frac{\log 1/\epsilon_1}{\log |A|})$ words of space.

Since the value of the LP is finite, there exist primal and dual optimal solutions $\mathbf{x}^*$ and $\mathbf{y}^*$, and by assumption all entries of $\mathbf{x}^*$ and $\mathbf{y}^*$ have absolute value polynomial in b and c. Let $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{y}}^*$ denote the vectors $\mathbf{x}^*$ and $\mathbf{y}^*$, with all entries rounded up to integer multiples of ϵ where $\epsilon = \frac{\epsilon_1}{2u}$. Here, $u \in \mathbb{Z}$ is a strict upper bound on $\|A\|_\infty$, $\|\mathbf{c}\|_1$, $\|\mathbf{b}\|_1$, $\|\mathbf{x}\|_1$, and $\|\mathbf{y}\|_1$, where throughout, if A is a matrix then $\|A\|_p$ denotes the operator norm of A induced by the p norm on vectors, and if $\mathbf{x}$ is a vector then $\|\mathbf{x}\|_p$ denotes the p-norm on vectors. Notice u is polynomial in b and c, does not depend on ϵ_1, and can be determined by V while observing the stream.

The protocol is exactly as in Theorem 3.5, except $\mathcal{H}$ sends vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ claimed to be $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{y}}^*$. $\mathcal{V}$ checks that

1. $\epsilon_1/\epsilon \geq 2u$. That is, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are represented at sufficiently high precision.
2. $\hat{\mathbf{x}}$ is feasible for the perturbed primal.
3. $|\mathbf{b}^T\hat{\mathbf{y}} - \mathbf{c}^T\hat{\mathbf{x}}| \leq \epsilon_1$.
4. $\hat{\mathbf{y}}$ is feasible for the LP
$$\text{minimize } \mathbf{b}^T\mathbf{y}$$
$$\text{subject to } \begin{bmatrix} A^T \\ -A^T \end{bmatrix} \mathbf{y} \geq \begin{bmatrix} \mathbf{c} \\ -\mathbf{c} \end{bmatrix} - \epsilon_1 \mathbf{1},$$
$$\mathbf{y} \geq 0.$$
This LP can be thought of as the perturbed dual, although it is not the dual of the perturbed primal.

If all checks pass, $\mathcal{V}$ outputs $\mathbf{c}^T\hat{\mathbf{x}}$.

Protocol Validity. Write $\mathbf{e}(\mathbf{x}^*) = \hat{\mathbf{x}}^* - \mathbf{x}^*$ and $\mathbf{e}(\mathbf{y}) = \hat{\mathbf{y}}^* - \mathbf{y}^*$. Notice that $\|\mathbf{e}(\mathbf{x}^*)\|_\infty \leq \epsilon$ and $\|\mathbf{e}(\mathbf{y}^*)\|_\infty \leq \epsilon$ for all i. Since $\mathbf{x}^*$ and $\mathbf{y}^*$ are primal and dual feasible, it is easy to see $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{y}}^*$ are feasible for the perturbed LPs. Indeed,

$$\|A\mathbf{x}^* - A\hat{\mathbf{x}}^*\|_\infty = \|A\mathbf{e}(\mathbf{x}^*)\|_\infty \leq \|A\|_\infty \|\mathbf{e}(\mathbf{x}^*)\|_\infty \leq u\epsilon \leq \epsilon_1,$$

and a similar argument shows $\hat{\mathbf{y}}^*$ is feasible for the perturbed dual.

Since $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$, it follows that

$$|\mathbf{b}^T\hat{\mathbf{y}}^* - \mathbf{c}^T\hat{\mathbf{x}}^*| \leq |\mathbf{c}^T\mathbf{e}(\mathbf{x})| + |\mathbf{b}^T\mathbf{e}(\mathbf{y})| \leq 2\epsilon u \leq \epsilon_1.$$

Therefore, if $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are as claimed, all of $\mathcal{V}$'s checks will pass.

To prove the protocol is valid, it therefore suffices to show that if vectors $\mathbf{x}$ and $\mathbf{y}$ pass $\mathcal{V}$'s checks, then $\mathbf{c}^T\mathbf{x}$ is within an additive $\epsilon_2 = (u+2)\epsilon_1$ of the true value of the perturbed primal LP. $\mathbf{c}^T\mathbf{x}$ is clearly a lower bound on the value of the perturbed primal, since $\mathbf{x}$ is feasible for the perturbed primal. We claim

$\mathbf{b}^T\mathbf{y} + (u+1)\epsilon_1$ is an upper bound on the value of the perturbed primal. Indeed, since $\mathbf{y}$ is feasible for the perturbed dual, $\|A^T\mathbf{y} - \mathbf{c}\|_\infty \leq \epsilon$ for all i, and thus for all feasible $\mathbf{x}$ of the perturbed primal,

$$\begin{aligned}\mathbf{c}^T\mathbf{x} &\leq \mathbf{y}^T A\mathbf{x} + |(A^T\mathbf{y} - \mathbf{c})^T\mathbf{x}| \\ &\leq \mathbf{y}^T A\mathbf{x} + \epsilon u \\ &\leq \mathbf{y}^T\mathbf{b} + \epsilon_1|\mathbf{y}^T\mathbf{1}| + u\epsilon \\ &\leq \mathbf{b}^T\mathbf{y} + (u+1)\epsilon_1,\end{aligned}$$

where the third inequality holds because $\mathbf{x}$ is feasible for the perturbed primal and $\mathbf{y} \geq 0$. Therefore, $\mathbf{c}^T\mathbf{x}$ is a lower bound on the value of the perturbed primal and $\mathbf{b}^T\mathbf{y} + (u+1)\epsilon_1 \leq \mathbf{c}^T\mathbf{x} + (u+2)\epsilon_1$ is an upper bound, which completes the proof of validity.

If the entries of A, $\mathbf{b}$, and $\mathbf{c}$ are not integer multiples of ϵ, but are instead arbitrary rationals p/q for $p, q \in \mathbb{Z}$, we run the above protocol on the *derived* stream in which each stream element is rounded to the nearest integer multiple of ϵ. (If $\mathcal{V}$ cannot determine the necessary precision ϵ in advance, we can afford for $\mathcal{V}$ to calculate it while observing the stream, and then have $\mathcal{H}$ replay the entire stream). This introduces $\text{poly}(b, c)\epsilon$ error into all of the above calculations, but we can obtain the same approximation guarantees by setting ϵ a polynomial factor smaller than above. This does not affect the asymptotic costs of the protocol. □ □

B Proof of Theorem 4.5

Proof. The dual function of any quadratic programming problem is given by

$$q(\mu) = -\frac{1}{2}\mu^T AQ^{-1}A^T\mu - \mu^T(\mathbf{b} + AQ^{-1}\mathbf{c}) - \frac{1}{2}\mathbf{c}^T Q^{-1}\mathbf{c} \text{ if } \mu \geq 0$$

and $q(\mu) = -\infty$ otherwise. Strong duality always holds for quadratic programs. That is, for all μ, $q(\mu)$ is a lower bound on the value f^* of the primal quadratic program, and moreover there exists a $\mu^* \geq 0$ such that $q(\mu^*) = f^*$ [19, Example 5.3.1]. Consequently, we can use a protocol akin to that of Theorem A.2 and Corollary 4.3: essentially $\mathcal{H}$ proves optimality of a primal solution $\mathbf{x}$ by providing a dual-optimal solution μ, and proving to $\mathcal{V}$ that the values of $\mathbf{x}$ and μ are equal. But since $\mathcal{H}$ cannot afford to send an exact representation of $\mathbf{x}$ or μ, $\mathcal{H}$ instead sends vectors $\hat{\mathbf{x}}$ and $\hat{\mu}$ with all entries integer multiples of $\epsilon = \frac{\epsilon_1}{h(\mathcal{A})}$. Here h is some function polynomial in b and c and independent of ϵ_1, to be specified later. These vectors are claimed to be rounded versions of the true optima.

Let $f(\mathbf{x})$ denote the primal objective function evaluated at $\mathbf{x}$. $\mathcal{V}$ checks that

1. $\epsilon_1/\epsilon \geq h(\mathcal{A})$. That is, $\mathbf{x}$ and μ are represented at sufficiently high precision.
2. $A\hat{\mathbf{x}} \leq \mathbf{b} + \epsilon_1\mathbf{1}$.
3. $\mu \geq \mathbf{0}$.
4. $q(\hat{\mu}) - f(\hat{\mathbf{x}}) \leq \epsilon_1$.

   If all checks pass, $\mathcal{V}$ outputs $\mathbf{c}^T\hat{\mathbf{x}}$, otherwise $\mathcal{V}$ outputs $\perp$.

The first and third checks are trivial to perform, and $\mathcal{V}$ can perform the second check with a single matrix-vector multiplication operation. To perform the fourth check, $\mathcal{V}$ first computes$f(\hat{\mathbf{x}})$ with a single matrix-vector multiplication and two inner product computations. $\mathcal{V}$ then computes $q(\mu)$ by verifying a constant number of matrix-vector multiplications and inner product computations as follows. First $\mathcal{V}$ computes $\mathbf{z}_1 := -\frac{1}{2}\mu^T AQ^{-1}A^T\mu$ in a sequence of four matrix-vector multiplications: (1) $\mathbf{z}_{11} := A^T\mu$; (2) $\mathbf{z}_{12} := Q^{-1}\mathbf{z}_{11}$ (computed by having $\mathcal{H}$ specify $\mathbf{z}_{12}$ and verifying that $Q\mathbf{z}_{12} = \mathbf{z}_{11}$); (3) $\mathbf{z}_{13} := A\mathbf{z}_{12}$; and (4) $\mathbf{z}_1 = -\frac{1}{2}\mu^T\mathbf{z}_{13}$. We stress that $\mathcal{V}$ need not explicitly invert Q or be provided with Q^{-1} to compute $\mathbf{z}_{12} := Q^{-1}\mathbf{z}_{11}$; instead, it suffices to verify a single matrix-vector multiplication. Next, $\mathcal{V}$ computes $\mathbf{z}_2 := \mu^T\mathbf{b}$ with a single inner product computation, and computes $\mathbf{z}_3 := \mu^T AQ^{-1}\mathbf{c}$ and $\mathbf{z}_4 := \frac{1}{2}\mathbf{c}^T Q^{-1}\mathbf{c}$ similarly to $\mathbf{z}_1$. Given $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$, and $\mathbf{z}_4$, $\mathcal{V}$ may compute the value $q(\mu) = \mathbf{z}_1 - \mathbf{z}_2 - \mathbf{z}_3 - \mathbf{z}_4$.

It remains to argue that we can set $\epsilon = \epsilon_1/h(\mathcal{A})$ such that: if $\hat{\mathbf{x}}$ and $\hat{\mu}$ are as claimed, then all checks will pass with probability 1, and if all checks pass then with high probability the value of the perturbed QP is $\mathbf{c}^T\hat{\mathbf{x}} \pm \epsilon$.

Suppose $\hat{\mathbf{x}}$ and μ are as claimed. Then first and third checks will pass, and the second check will pass by exactly the same argument as in Theorem A.2. For the final check, let $\mathbf{e} = \mu - \hat{\mu}$; $\|\mathbf{e}\|_\infty \leq \epsilon$. Then

$$q(\mu) - q(\hat{\mu}) = \mu^T AQ^{-1}A^T\mathbf{e} + \frac{1}{2}\mathbf{e}^T AQ^{-1}A^T\mathbf{e} - \mathbf{e}^T(\mathbf{b} + AQ^{-1}\mathbf{c})$$

By repeated application of the triangle inequality to each term in the above sum, it can be seen that the above expression is at most $\text{poly}(u)\epsilon$, where u is a known upper bound on $\|A\|_\infty$, $\|Q\|_\infty$, $\|Q^{-1}\|_\infty$, $\|\mathbf{b}\|_1$, $\|\mathbf{c}\|_1$, $\|\mathbf{x}\|_1$ and $\|\mu\|_1$. Notice that $\|Q^{-1}\|_\infty$ is polynomially bounded as long as Q is sufficiently positive definite, in the sense that all eigenvalues of Q are at least $1/\text{poly}(b, c)$. All other quantities for which u is an upper bound are polynomial in b and c by assumption, and the above expression is therefore at most $\text{poly}(b, c)\epsilon$ for some known polynomial in b and c that is independent of ϵ_1.

Similarly, $f(\hat{x}) - f(\mathbf{x}) = \text{poly}(b, c)\epsilon$ for some polynomial in b and c that is independent of ϵ_1. Thus, $f(\hat{x}) - q(\hat{\mu}) \leq |f(\mathbf{x}) - q(\mu)| + |f(\hat{x}) - f(\mathbf{x})| + |q(\mu) - q(\mu)|$ is also $\text{poly}(b, c)\epsilon$ for some polynomial in b and c that only depends on u and is independent of ϵ_1. We can therefore choose $h(\mathcal{A}) = \text{poly}(b, c)\epsilon$ such that the final check will pass if $\hat{x}$ and $\hat{\mu}$ are as claimed.

Finally, we argue that if all four checks pass, then $f(\mathbf{x})$ is a $g(\mathcal{A})\epsilon_1$-approximation to the value of the perturbed primal for some $g(\mathcal{A}) = \text{poly}(b, c)$ independent of ϵ_1. The argument in this case is simpler than that in Theorem A.2, because if the third check passes, then μ is actually feasible for the dual of the perturbed primal, as the only constraint of the dual is that $\mu \geq 0$. Indeed, let q_P denote the dual of the perturbed primal; it is easy to see that $q_P(\mu) = q(\mu) - \mu^T(\epsilon_1\mathbf{1})$. By weak duality, for any $\mathbf{x}$ that is feasible for the perturbed primal, and any μ, $f(\mathbf{x}) \geq q_P(\mu) \geq q(\mu) - u\epsilon_1$. Since the fourth check passed, $f(\mathbf{x}) - q(\mu) \leq \epsilon_1$, and therefore $f(\mathbf{x}) - q_P(\mu) \leq (u+1)\epsilon_1$. We can let $g(\mathcal{A}) = u + 1$.

By invoking the matrix-vector multiplication protocols of Theorem 3.5 and Theorem 4.1 (handling the fact that the vectors are not integral as in Corollary 4.3) to compute all matrix-vector multiplications in the above description, we obtain the theorem. □ □