SRC Technical Note
1998 - 011

May 26, 1998

Computing on data streams

Monika Rauch Henzinger, Prabhakar Raghavan, and Sridar Rajagopalan

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Prabhakar Raghavan and Sridar Rajagopalan are at the IBM Almaden Research Center, San Jose, CA. Their electronic mail addresses are: {pragh, sridhar}@almaden.ibm.com

Abstract

In this paper we study the space requirement of algorithms that make only one (or a small number of) pass(es) over the input data. We study such algorithms under a model of *data streams* that we introduce here. We give a number of upper and lower bounds for problems stemming from query-processing, invoking in the process tools from the area of *communication complexity*.

1 Overview

In this paper we study the space requirement of algorithms that make only one (or a small number of) pass(es) over the input data. We study such algorithms under a model of *data streams* that we introduce here. We develop an intimate connection between this setting and the classical theory of *communication complexity* [AMS96, NK, Yao79].

1.1 Motivation

A data stream, intuitively, is a sequence of data items that can be read once by an algorithm, in the prescribed sequence. A number of technological factors motivate our study of data streams.

The most economical way to store and move large volumes of data is on secondary and tertiary storage; these devices naturally produce data streams. Moreover, multiple passes are prohibitive due to the volumes of data on these media; for example internet archives [Ale] sell (a large fraction of) the web on tape. This problem is exacerbated by the growing disparity between the costs of secondary/tertiary storage and the cost of memory/processor speed. Thus to sustain performance for basic systems operations, core utilities are restricted to read the input only once. For example, storage managers (such as IBM's ADSM [ADSM]) use one-pass differential backup [ABFLS].

Networks are bringing to the desktop ever-increasing quantities of data in the form of data streams. For data in networked storage, each pass over the data results in an additional, expensive network access.

The SELECT/PROJECT model of data access common in database systems give a "one-pass like" access to data through system calls independent of the physical storage layout. The interface between the storage manager and the application layer in a modern database system is well-modeled as a data stream. This data could be a filtered version of the stored data, and thus might not be contiguous in physical storage.

In a large multithreaded database system, the available main memory is partitioned between various computational threads. Moreover, operators such as the hash-based GROUP BY operator compute multiple aggregation results concurrently using only a single scan over the data. Thus, the amount of memory used in each thread influences efficiency in two ways: (1) It limits the number of concurrent accesses to the database system. (2) It limits the number of different computations that can be performed simultaneously with one scan over the data. Thus effectively, even a 1Gb machine will have provide under 1Mb to each thread when supporting a thousand concurrent threads, especially after the operating system and the DBMS take their share of memory.

There is thus a need for studying algorithms that operate on data streams.

1.2 Scope of the present work

A *data stream* is a sequence of data items $x_1, \ldots, x_i, \ldots, x_n$ such that the items are read once in increasing order of the indices i. Our model of computation can be described by two parameters: The number P of *passes* over the data stream and the *workspace* S (in bits) required in main memory, measured as function of the input size n. We seek algorithms for various problems that use one or a small number of passes and require a workspace that is smaller than the size of the input. Our model does not require a bound on the computation time, though for all the algorithms we present, the time required is small. In query settings, the size of the output is typically much smaller than the size of the workspace.

For example, for graph problems (e.g., [MAGQW, MM]), we view the input as a sequence of m (possibly directed) edges between n nodes. Our goal is to find algorithms where space requirements can be bounded as a function of n (rather than m), or in establishing that the space must grow with m.

Our goal is to expose dichotomies in the space requirements along the different axis: (i) between one-pass and multi-pass algorithms, (ii) between deterministic and randomized algorithms, and (iii) between exact and approximation algorithms.

We first describe some classes of problems which can be described in this context.

(1) Systems such as LORE[MAGQW] and WEBSQL [AMM, MM, MMM] view a database as a graph/hypergraph. For instance, a directed edge might represent a hyperlink on the web, or a citation between scientific papers, or a pair of cities connected by a flight; in a database of airline passengers a hyperedge may relate a passenger, the airports he uses, and the airline he uses in a flight. Some typical queries might be: From which airport in Africa can one reach the most distinct airports within 3 hops? (The *MAXTOTAL* problem below.) Of the papers citing the most referenced graphics paper, which has the largest bibliography? (The *MAX*

problem below.)

We propose four problems that model queries such as those described here: Consider a directed multigraph with node set $V_1 \cup V_2 \ldots \cup V_k$, all of whose edges are directed from a node in V_i to a node in V_{i+1}. Let $n = \max_i |V_i|$. The degree of a vertex is its indegree unless specified otherwise.

The *MAX* problem. Let u_1 be the node of largest outdegree in V_1. Let $u_i \in V_i$ be a node of largest degree among those incident to u_{i-1}. Find u_k.

The *MAXNEIGHBOR* problem. Let u_1 have largest outdegree in V_1. Let $u_i \in V_i$ have the largest number of edges to u_{i-1}, determine u_k.

The *MAXTOTAL* problem. Find a node $u_1 \in V_1$ which is connected to the largest number of nodes of V_k.

The *MAXPATH* problem. Find nodes $u_1 \in V_1$, $u_k \in V_k$ such that they are connected by the largest possible number of paths.

(2) A second problem class is verifying consistency in databases. For instance, check if each customer in a database has a unique address, or if each employee has a unique manager/salary. We model these problems as consistency verification problems of relations. Let a k-ary relation R over $\{0, 1, \cdots n\}$ be given. Let $\phi = \forall u_1, u_2, \ldots !\exists(v_1, v_2, \ldots) : f(u_1, \ldots, v_1, \ldots)\text{for}(u_1, u_2, \ldots, v_1, v_2, \ldots) \in R$.

The Consistency Verification problem. Verify that R satisfies ϕ.

(3) More traditional graph problems like connectivity arise [BGMZ97], while analyzing various properties of the web. In database query optimization estimating the size of the transitive closure is important [LN89]. This motivates our study of study of various traditional graph properties.

(4) As pointed out in [SALP79, AMS96] estimates of the frequency moments of a data set can be used effectively for database query optimization. This motivates our study of approximate frequency estimation problems and approximate selection problems (e.g., find a product whose sales are within 10% of the most popular product).

1.3 Definitions

Las-Vegas and Monte-Carlo algorithms. A *randomized* algorithm is an algorithm that flips coins, i.e., uses random bits: no probabilistic assumption is made of the distribution of inputs. A randomized algorithm is called *Las-Vegas* if it gives the correct answer on all input sequences; its running time or workspace could be a random variable depending on the coin tosses. A randomized algorithm is called

Monte-Carlo with error probability ϵ if on every input sequence it gives the right answer with probability at least $1 - \epsilon$. If no ϵ is specified, it is assumed to be $2/3$.

Our principal tool for showing lower bounds on the workspace of limited-pass algorithms is drawn from the area of *communication complexity*.

Communication complexity. Let X, Y, and Z be finite sets and let $f : X \times Y \to Z$ be a function. The *(2-party) communication model* consist of two *players*, A and B such that A is given an $x \in X$ and B is given an $y \in Y$ and they want to compute $f(x, y)$. The problem is that A does not know y and B does not know x. Thus, they need to communicate, i.e., exchange bits according to an agreed-upon *protocol.* The *communication complexity of a function* f is the minimum over all communication protocols of the maximum over all $x \in X$ and all $y \in Y$ of the number of bits that need to be exchanged to compute $f(x, y)$. The protocol can be deterministic, Las Vegas or Monte Carlo. Finally, if the communication is restricted to one player transmitting and the other receiving, then this is termed *one-way communication complexity.* In a one-way protocol, it is critical to specify which player is the transmitter and which the receiver. Only the receiver needs to be able to compute f.

1.4 Related previous work

Estimation of order statistics and outliers [ARS97, AS95, JC85, RML97, Olk93] has received much attention in in the context of sorting [DNS91], selectivity estimation [PIHS96], query optimization [SALP79] and in providing online user feedback [Hel]. The survey by Yannakakis [Yan90] is a comprehensive account of graph-theoretic methods in database theory.

Classical work on time-space tradeoffs [Cob66, Tom80] may be interpreted as lower bounds on workspace for problems such as verifying palindromes, perfect squares and undirected st connectivity. Paterson and Munro [MP80] studied the space required in selecting the kth largest out of n elements using at most P passes over the data. They showed an upper bound of $n^{1/P} \log n$ and an almost matching lower bound of $n^{1/P}$ for large enough k. Alon, Matias and Szegedy [AMS96] studied the space complexity of estimating the *frequency moments* of a sequence of elements in one-pass. In this context, they show (almost) tight upper and lower bounds for a large number of frequency moments and show how communication complexity techniques can be used to prove lower bounds on the space requirements.

Our model appears at first sight to be closely related to papers on I/O complexity [HK81], hierarchical memory [AACS87], paging [ST85] and competitive analysis [KMRS88], as well as external memory algorithms [VV96]. However, our model is considerably more stringent: whereas in these papers on memory management one can bring back (into fast memory) a data item that was previously evicted (and is required again), in our model we cannot retrieve items that are discarded.

1.5 Our main results

We expose the following dichotomies in our model. (i) Some problems require large space in one pass but small space in two. (ii) We show that there can be an exponential gap in space bounds between Monte Carlo and Las Vegas algorithms. (iii) We show that if we settle for an approximate solution, we can reduce the space requirement substantially. Our tight lower bounds for the approximate solution apply communication complexity techniques to approximation algorithms.

Theorem 1 *In one pass, the* MAX *problem requires* $\Omega(kn^2)$ *space and has an* $O(kn^2 \log n)$ *space solution. In* $P > 1$ *passes it requires* $\Omega(kn/P)$ *space and can be solved in* $O((kn \log n)/P)$ *space.*

Theorem 2 *In one pass, the* MAXNEIGHBOR*,* MAXTOTAL*, and* MAXPATH *problem require* $\Omega(kn^2)$ *space and have* $O(kn^2 \log n)$ *space solutions.*

Notice however, that unlike the *MAX* problem, the other three do not seem to admit efficient two pass solutions. Resolving this remains an open issue. We believe that no constant number of passes will result in substantial savings.

Let R be a $k = (k_1 + k_2)$-ary relation over $\{1, \ldots n\}$. Consider the formula

$$\phi = \forall u_1 \ldots u_{k_1}, !\exists (v_1 \ldots v_{k_2}) : f(u_1 \ldots u_{k_1}, v_1 \ldots v_{k_2}) \text{ for } (u_1, \ldots v_1 \ldots) \in R$$

where f is a function assumed to be provided via an oracle. Also suppose that we are presented the relation R one tuple at a time. Then, we have the following:

Theorem 3 *Verifying that* R *satisfies* ϕ *can be done by an* $O(\log \frac{1}{\delta} \log n)$ *space Monte Carlo algorithm that outputs the correct answer with probability* $1 - \delta$*. Any Las Vegas algorithm that verifies that* R *satisfies* ϕ *requires at least* $\Omega(n^2)$ *space.*

Theorem 3 shows an exponential gap between Las Vegas and Monte Carlo algorithms. In Section 4, we describe an algorithm and its analysis; these are easily modified to yield Theorem 3 through a *completeness* property described further in Section 4. We also have the following open problem: Let R be a binary relation. Let $\phi = \forall x \exists u : (x, u) \in R$. Is there any sub-linear space Monte Carlo algorithm that verifies that R satisfies ϕ?

Theorem 4 *Given a sequence of* m *numbers in* $\{1, \ldots, n\}$ *with multiple occurrences finding the* k *most frequent items requires* $\Omega(n/k)$ *space. Random sampling yields an upper bound of* $O(n(\log m + \log n)/k)$.

The proof of Theorem 4 is in Section 5.

The *approximate median problem* requires finding a number whose rank is in the interval $[m/2 - \epsilon m, m/2 + \epsilon m]$. It can be solved by a one-pass Monte Carlo algorithm with error probability $1/10$ and $O(\log n (\log 1/\epsilon)^2/\epsilon)$ space [RML97]. We give a corresponding lower bound in Section 5.

Theorem 5 *Any 1-pass Las Vegas algorithm for the approximate median problem requires $\Omega(1/\epsilon)$ space.*

Easy one-pass reductions from the communication complexity of the DISJOINTNESS function [NK] yields:

Theorem 6 *In P passes, the following graph problems on an n-node graph all require $\Omega(n/P)$ space: computing the connected components, k-edge connected components with $1 < k < n$, k-vertex connected components with $1 < k < n$, testing graph planarity. Finding the sinks in a directed graph requires $\Theta(n/P)$ space.*

Incremental graph algorithms give one-pass algorithms for all the problems of Theorem 6. Thus, there are one-pass algorithms for connected components, k-edge and k-vertex connectivity with $k \leq 3$, and planarity testing that use $O(n \log n)$ space.

Theorem 7 *For any $1 > \epsilon > 0$, estimating in one pass the size of the transitive closure to within a factor of ϵ requires space $\Omega(m)$.*

We prove this theorem in Section 5. Computing the exact size of the transitive closure requires $O(m \log n)$ space.

The lower bounds of Theorems 1, 2, 6 and 7 hold even for Monte Carlo algorithms that are correct with error probability ϵ for a sufficiently small ϵ.

All our lower bounds are information-theoretic, placing no bounds on the computational power of the algorithms. The upper bounds, on the other hand, are all "efficient": in all cases, the running time is about the same as the space usage.

2 Three lower bounds from communication complexity

Many of the lower bounds in our model build on three lower bounds in communication complexity. We review these lower bounds in this section.

Bit-Vector Probing. Let A have a bit-vector x of length m. Let B have an index $0 < i \leq m$. B needs to know x_i, the ith input bit. The only communication allowed is from A to B.

There is no better method for A to communicate x_i to B than to send the entire string x. More precisely, any algorithm that succeeds in B guessing x_i correctly with probability better than $(1+\epsilon)/2$, requires at least ϵm bits of communication [NK].

Bit-Vector Comparison. Let A and B both have bit-vectors x y respectively, each of length m. B wishes to verify that $x = y$.

Any deterministic or Las Vegas algorithm that successfully solves this problem must essentially send the entire string x from A to B, or vice versa. More precisely, any algorithm that outputs the correct answer with probability at least ϵ and never outputs the wrong answer must communicate ϵm bits [NK].

Bit-Vector Disjointness. Let A and B both have bit-vectors x y respectively, each of length m. B wishes to find an index i such that $x_i = 1$ and $y_i = 1$.

There is no better protocol than to essentially send the entire string x from A to B, or vice versa. More precisely, any algorithm that outputs the correct answer with probability at least $1-\epsilon$ (for some small enough ϵ) must communicate $\Omega(m)$ bits [NK].

Notice that the second theorem is weaker than the first and the third in some respects: it does not apply to Monte Carlo algorithms. There is a good reason: there is a Monte Carlo algorithm that does much better, i.e. communicates only $O(\log n)$ bits. On the other hand, the first theorem is weaker than the second and third in some respects: it insists that there be no communication in one of the two directions. This too is for good reason: B could send A the index, and then A could respond with the bit. For a description of these and other issues in this area, see [NK].

3 One pass versus many passes

Our goal in this section is to outline the proof of Theorem 1 showing that some problems require large space in one pass but small space in two. We give here a lower bound of $\Omega(n^2)$ on the space used by any Monte Carlo one-pass algorithm for the 2-layer *MAX* problem; a somewhat more elaborate construction (omitted here) yields a lower bound of $\Omega(kn^2)$ for the k-layer version.

Proof of Theorem 1. We provide a reduction from the bit-vector probing problem. Denote the node-set on the left of the bipartite graph by U, and the node-set on the right by V, where $|U| = |V| = n$. We further partition U into U_1, U_2 where

$|U_1| = n/3 = \sqrt{m}$. Likewise we partition V into V_1, V_2 where $|V_1| = n/3$. The bit-string x is interpreted as specifying the edges of a bipartite graph on $U_1 \times V_1$ in the natural way, i.e. the edge (u, v) corresponds to index $u\sqrt{m} + v$. On getting the query i, we translate it to an edge $(u, v), u \in U_1, v \in V_1$ and augment the graph with edges (u, v') and (u', v) for each $u' \in U_2$ and $v' \in V_2$. The answer to the *MAX* problem on this graph is v if and only if the edge (u, v) is in the bipartite graph, i.e. the ith input bit is set.

The *MAX* problem is solved in 2 passes with space $O(kn \log n)$, even on k layered graphs: In the first pass find the degree of each vertex. In the second pass determine the highest degree neighbor in V_i for each vertex in V_{i-1}. Then compute u_1 and repeatedly find the highest degree neighbor of the current node until u_k is determined. This algorithm can be modified to use only space $O(kn \log n/P)$ in P passes.

Note that the lower bound proof that we provide above applies to approximate versions of the *MAX* problem as well, namely, it requires $\Omega(n^2)$ space to compute a near max degree neighbor of a vertex with near max degree in U.

Proof of Theorem 2. The proofs for all three problems are reductions from the bit-vector probing problem similar to the proof of Theorem 1.

To show the bound for the *MAXNEIGHBOR* problem we construct the same initial graph as in the proof of Theorem 1, but double each edge from V_1 to U_1. On getting the query i, we translate it into an edge (u, v), $u \in U_1, v \in V_1$, and add this edge to the graph. Additionally we augment the graph with two edges (u, v') for each $v' \in V_2$. Then there are three edges between u and v iff the ith input bit is set; otherwise there is only one edge between u and v. Thus, v is returned iff the ith input bit is set.

For the *MAXTOTAL* problem construct a tripartite graph with node set $U \cup V \cup W$, where $|U| = |V| = |W| = \sqrt{m} + 1$. The nodes in set U are numbered from 1 to $\sqrt{m} + 1$. The same holds for set V and set W. As in the proof of Theorem 1, the bit-string x is translated into edges from U to V as follows: edge (u, v) exists in the graph iff index $u\sqrt{m} + v$ of x is set. Additionally there is an edge from node $\sqrt{m} + 1$ in U to node $\sqrt{m} + 1$ in V and from the latter node to node $\sqrt{m} + 1$ in W. On getting a query i we translate it into an edge (u, v) with $u \in U$ and $v \in V$ and augment the graph by edges (v, w) for each $w \in W$. Then u reaches the most nodes in W iff the ith input bit was set; otherwise node $\sqrt{m} + 1$ of U reaches the most nodes of W.

For the *MAXPATH* problem augment the graph for the *MAXTOTAL* problem with a fourth node set X and connect every node of W with an edge to the same node x of X. Using the same reduction for a query as for problem *MAXTOTAL* shows that u and x are connected by the largest number of paths iff the ith input bit was set; otherwise node $\sqrt{m}+1$ and x are connected by the largest number of paths.

4 Las Vegas versus Monte Carlo

In this section we present an exponential gap between Las Vegas and Monte Carlo one-pass algorithms. The *symmetry* property given a sequence of ordered pairs over $\{1, \ldots, n\}$, is that (u, v) is in the sequence if and only if there is a unique (v, u) in it. In this section, we show a $O(\log n)$ space Monte Carlo algorithm to verify the symmetry property. By contrast, any one pass Las Vegas algorithm requires $\Omega(m)$ space, where m is the size of the relation.

Algorithm. Choose p, a random prime smaller than n^3. Let $l_{u,v} = n^{2((nu)+v)}$ if $u < v$ and $l_{u,v} = -(n^{2((nu)+v)})$ if $u > v$ and 0 otherwise. Compute the sum $s = \sum_{(u,v)\in R} l_{u,v}$ modulo p. Check if $s = 0 \bmod p$ at the end. Storing s mod p requires only $\log p < 3\log n$ space. Also check that there are no more than n^2 edges in all.

Theorem 8 *The above algorithm will output a correct response with probability at least* $1-(2\log^2 n/n)$. *Moreover, any one pass Las Vegas algorithm that outputs the correct response with probability* $2/3$ *or more uses* $\Omega(n^2)$ *space.*

Proof. It is easily seen that $\sum_{(u,v)\in R} l_{u,v}$ is 0 if and only if R is symmetric. On the other hand, it follows from the Chinese Remainder Theorem and the fact that there are at least $n^3/\log n$ primes smaller than n^3, that the probability that a non zero sum evaluates to 0 modulo a random prime is smaller than $2\log^2 n/n$. This follows, since s could be 0 modulo p for at most $2n^2\log n$ of them since $s < 2^{2n^2\log n}$.

The lower bound follows from a reduction from the bit-vector comparison problem. complexity. Assume that one player has a graph with each edge directed from the smaller numbered vertex to the larger numbered one. The second player has a graph with each edge directed the other way. The union of the two inputs is symmetric if and only if the two graphs are the same. Consider a truth table whose rows are indexed by all the possible inputs of the first player, and the columns are all possible inputs of the second player; each entry is the output corresponding to the players' inputs for that row and column. The truth table corresponding to the symmetry relation under our class of inputs has $2^{\binom{n}{2}}$ distinct rows, one corresponding to each possible graph. By a technique from communication complexity [NK], we require at least $c\binom{n}{2}$ communication in any algorithm that purports to solve this problem with probability $(1+c)/2$. Consider the following input data stream: we present the first player's ordered pairs followed by the second player's. The state of the data stream algorithm after the first half represents the communication between the players. ∎

More interesting, however, is a completeness property that arises here. Namely, that every problem of the form in Theorem 3 can be reduced to the symmetry problem.

Proof of Theorem 3. The reduction works as follows: we encode each tuple $(u_1, \ldots u_{k_1})$ as an index i using a standard Gödel encoding, g, let this encoding range from 1 through m. Then we output the ordered pair $(i, 0)$ for each $1 \leq i \leq m$. Then for each tuple $(u_1 \ldots, v_1, \ldots)$ we output $(0, g(u_1, \ldots u_{k_1}))$ if and only if $f(u_1, \ldots, v_1 \ldots)$. The resultant graph is symmetric if and only if the relation R satisfies ϕ.

5 Exact versus approximate computation

In this section, we show that if we settle for an approximate solution, we can reduce the space requirement substantially. Our matching lower bounds for the approximate solution require a generalization of communication complexity techniques to approximation algorithms.

Proof of Theorem 4. Alon *et al.* show that finding the mode (i.e., the most frequently-occurring number) of a sequence of m numbers in the range $\{1, \ldots, n\}$ requires space $\Omega(n)$. By a simple reduction (replace each number i in the original sequence by a sequence of k numbers $ki + 1, ki + 2, \ldots ki + k$) it follows that that finding one of the k most frequent items in one pass requires space $\Omega(n/k)$

The almost matching upper bound is given by the following Monte-Carlo algorithm that succeeds with constant probability: before the start of the sequence sample each number in the range with probability $1/k$ and then only keep a counter for the successfully sampled numbers. Output the successfully sampled number with largest count. With constant probability one of the k-th most frequent numbers has been sampled successfully. This needs $O(n(\log m + \log n)/k)$ space.

Proof of Theorem 5. We show that any algorithm that solves the ϵ-approximate median problem requires $\Omega(1/\epsilon)$ space. The proof follows from a reduction from the bit-vector probing problem. Let $b_1, b_2 \cdots b_n$ be a bit vector followed by a query index i. This is translated to a sequence of numbers as follows: First output $2j + b_j$ for each j. Then on getting the query, output $n - i - 1$ copies of 0 and $i + 1$ copies of $2(n + 1)$. It is easily verified that the least significant bit of the exact median of this sequence is the value of b_i. Choose $\epsilon = \frac{1}{2n}$. Thus, the ϵ approximate median is the exact median. Thus, any one pass algorithm that requires fewer than $\frac{1}{2\epsilon} = n$ bits of memory can be used to derive a communication protocol that requires fewer than n bits to be communicated from A to B in solving bit vector probing. Since every protocol that solves the bit-vector probing problem must communicate n bits, this is a contradiction.

We next prove Theorem 6.

Proof of Theorem 6. We reduce the bit-vector disjointness problem to the graph connectivity problem. Construct a graph whose node-set is $\{a, b, 1, 2, \ldots, n\}$. Insert an edge (a, i) if bit i is set in A's vector, and insert an edge (b, i) if bit i is set in B's vector. Now, a and b are connected in the graph if and only if there exists a bit that is set in both A's vector and B's vector. By the lower bound for the bit-vector disjointness problem, every protocol must exchange $\Omega(n)$ bits between A and B. Thus, if there are P passes over the data, one of the passes must use at least $\Omega(n/P)$ space. The reduction for k-edge or k-vertex connectivity follows by adding $k - 1$ nodes $c_1, \ldots, c_{k-1}$ and an edge from each c_j, $1 \leq j \leq k - 1$ to both a and b.

To reduce to planarity testing we add four nodes c_1, c_2, c_3, c_4 and connect them pairwise. Additionally we add the edges (c_1, a), (c_2, a), (c_3, a), and (c_4, b). Then the graph contains K_5 as a minor if and only if a and b are connected.

We also reduce the bit-vector disjointness problem to the problem of deciding whether the graph contains a sink. Construct a graph whose node-set is $\{a, b, 1, 2, \ldots, n\}$. Insert edges (a, b) and (b, a) to guarantee that neither of them is a sink. If bit i is set in A's vector, insert an edge (a, i), otherwise insert an edge (i, a). Similarly, if bit i is set in B's vector, insert an edge (b, i), otherwise insert an edge (i, b). Now node i is a sink if and only if bit i is set in both A's and B's vector. It follows that the graph contains a sink if and only if there exists a bit that is set in both A's vector and B's vector. By the lower bound for the bit-vector disjointness problem, every protocol must exchange $\Omega(n)$ bits between A and B. Thus, if there are P passes over the data, one of the passes must use at least $\Omega(n/P)$ space.

A P-pass algorithm that keeps a bit for node $(i - 1)n/P$, $(i - 1)n/P + 1$, $\ldots, in/P - 1$ in pass i indicating whether an edge leaving the node was read gives the desired upper bound.

We also provide a lower bound here for the transitive closure problem.

Proof of Theorem 7. We reduce the bit-vector probing problem to the transitive closure estimation problem. Let $d \geq 1$ be a constant. Given a bit-vector of length m, we construct a graph G on $2(dm + \sqrt{m})$ vertices, V_i, $1 \leq i \leq 4$, with $|V_2| =$

$|V_3| = \sqrt{m}$ and $|V_1| = |V_4| = dm$, such that edge (i, j) with $i \in V_2$ and $j \in V_3$ exists iff entry $i\sqrt{m} + j$ is set in the vector. To test whether entry $i\sqrt{m} + j$ is set, add edges from each vertex in V_1 to $i \in V_2$ and from $j \in V_3$ to each vertex in V_4. The size of the transitive closure is larger than m if and only if the edge (i, j) is in the graph. Furthermore, for $\epsilon < 1 - 2/(d^2 + 1)$, any ϵ-approximation algorithm for the transitive closure can answer a query correctly. Thus, any ϵ-approximation algorithm must use $\Omega(m)$ space.

6 Further work

Our work raises a number of directions for further work; we list some here:

1. We need more general techniques for both lower and upper bounds when multiple passes can be performed over the data. They might also imply interesting new results about communication complexity. From a practical perspective, algorithms are needed for a wider class of problems than the selection problem that has been extensively studied [ARS97, AS95, JC85, RML97, Olk93].

2. Can we design algorithms that minimize the number of passes performed over the data given the amount of memory available? This would be useful when, for instance, the number of active concurrent threads governs the memory available at runtime

3. How can we arrange the data physically in a linear order with the express goal of optimizing the memory required to process some set of queries? Recall that the results of a query may not necessarily be physically contiguous (e.g., in the database of airports, the subset from Africa may not be together; more generally, we will have to cope with the results of some class of SELECT and GROUPBY operations). Can we model the class of "likely" queries and use it to drive the data layout?

4. From a theoretical perspective, we have highlighted the importance of studying the communication complexity of approximation problems (as in our bounds for the approximate solutions of selection and transitive closure); existing work only treats computations that yield exact answers.

References

[ADSM] http://www.storage.ibm.com/software/adsm/addoc.htm

[AACS87] A. Aggarwal, B. Alpern, A.K. Chandra and M. Snir. A model for hierarchical memory. *Proc. ACM STOC*, 305–314, 1987.

[ABFLS] M. Ajtai, R. Burns, R. Fagin, D. Long, and L. Stockmeyer, Compactly encoding arbitrary files with differential compression. To appear.

[Ale] `http://www.alexa.com`

[AMS96] N. Alon, Y. Matias and M. Szegedy. The space complexity of approximating frequency moments. *Proc. ACM STOC*, 20–29, 1996.

[ARS97] K. Alsabti, S. Ranka, and V. Singh. A One-Pass Algorithm for Accurately Estimating Quantiles for Disk-Resident Data. In *Proc. 23rd VLDB Conference*, Athens, Greece, 1997.

[AS95] R. Agrawal and A. Swami. A One-Pass Space-Efficient Algorithm for Finding Quantiles. In *Proc. 7th Intl. Conf. Management of Data (COMAD-95)*, Pune, India, 1995.

[AMM] G. Arocena, A. Mendelzon, G. Mihaila. Applications of a Web Query language. *Proceedings of the 6th International WWW Conference,* 1997.

[AAFL96] B. Awerbuch, Y. Azar, A. Fiat, T. Leighton Making commitments in the face of uncertainty: how to pick a winner almost every time In *Proc. of 28th STOC*, 519-530, 1996.

[BGMZ97] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proc. of the 6th International WWW Conference*, pages 391-404, April 1997.

[Cob66] A. Cobham. The recognition problem for the set of perfect squares. IBM Research Report RC 1704, T.J. Watson Research Center, 1966.

[DNS91] D. DeWitt, J. Naughton, and D. Schneider. Parallel Sorting on a Shard-Nothing Architecture using Probabilistic Splitting. In *Proc. Intl. Conf. on Parallel and Distributed Inf. Sys.*, pages 280–291, Miami Beach, 1991.

[Hel] J. M. Hellerstein. Online Processing Redux. *IEEE Data Engineering Bulletin,* 1997.

[HK81] J-W Hong and H.T. Kung. I/O Complexity: the red-blue pebble game. *Proc. ACM STOC*, 326–333, 1981.

[JC85] R. Jain and I. Chlamtac. The P^2 Algorithm for Dynamic Calculation for Quantiles and Histograms without Storing Observations. *CACM*, 28(10):1076–1085, 1985.

[KMRS88] A. R. Karlin, M. S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):70–119, 1988.

[MAGQW] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. SIGMOD Record, 26(3), 54–66, 1997.

[MP80] J. I. Munro and M. S. Paterson. Selection and Sorting with Limited Storage. *Theoretical Computer Science*, 12:315–323, 1980.

[LN89] R.J. Lipton and J.F. Naughton. Estimating the size of generalized transitive closure. *Proc. 15th VLDB*, 165–172, 1989.

[MM] A. Mendelzon and T. Milo. Formal Models of the Web. *Proceedings of ACM PODS Conference,* 1997.

[MMM] A. Mendelzon, G. Mihaila, T. Milo. Querying the World Wide Web. *Journal of Digital Libraries* 1(1), 68-88, 1997.

[NK] N. Nisan and E. Kushelevitz. Communication Complexity.

[Olk93] F. Olken. *Random Sampling from Databases*. PhD thesis, University of California Berkeley, 1993.

[PIHS96] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *ACM SIGMOD 96*, pages 294–305, Montreal, June 1996.

[RML97] G. Manku, S. Rajagopalan, and B. Lindsay. Approximate medians and other order statistics in one pass and with limited memory: Theory and Database applications. *Proceedings of SIGMOD 98, to appear.*

[SD77] B. W. Schmeiser and S. J. Deutsch. Quantile Estimation from Grouped Data: The Cell MidPoint. *Communications in Statistics: Simulation and Computation*, B6(3):221–234, 1977.

[SALP79] P. G. Selinger, M. M. Astrahan, R. A. Lories, and T. G. Price. Access Path Selection in a Relational Database Management System. In *ACM SIGMOD 79*, June 1979.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, February 1985.

[VV96] D.E. Vengroff and J.S. Vitter. I/O-efficient algorithms and environments. *Computing Surveys* 28, 212, 1996.

[Tom80] M. Tompa. Time-Space Tradeoffs for Computing Functions, Using Connectivity Properties of their Circuits. *JCSS* 20, 118–132, 1980.

[Yan90] M. Yannakakis. Graph-theoretic methods in database theory. *Proc. PODS*, 230–242, 1990.

[Yao79] A. C-C. Yao. Some complexity questions related to distributed computing. *Proc. 11th ACM STOC*, 209–213, 1979.