Research Statement

David P. Woodruff
MIT
dpwood@mit.edu

As a graduate student, I have worked in approximation algorithms, clustering, coding theory, combinatorics, communication complexity, computational complexity, cryptography, derandomization, graph theory, information retrieval, and streaming algorithms. Most of my work is centered around two common themes: distance approximation and efficient cryptographic protocols. I will describe my contributions to these two areas, and discuss my future research goals.

1 Distance approximation

1.1 Streaming algorithms

A decade ago the golden standard for an algorithm was that it runs in linear time and uses linear space, since most non-trivial problems cannot be solved with anything less. With today’s massive internet and data repositories, this standard is not well-suited to many problems which arise in practice. For instance, we cannot expect a resource-challenged internet router to store all traffic passing through it in an attempt to learn, say, the most common forwarding destination. However, it is clear that if the router were to forget even one piece of traffic, it could output the wrong answer. To cope with this, we relax the problem and only ask for an approximate solution.

This model was formalized in the seminal paper of Alon, Matias, and Szegedy [2]. The authors considered the space complexity of approximating the frequency moments $F_k$ in large data streams, where the $k$th frequency moment $F_k$ is defined as $\sum_{i=1}^{m} f_i^k$, where $f_i$ is the frequency, i.e., the number of occurrences, of the $i$th alphabet symbol in the data stream. Here $m$ denotes the alphabet size.

Thus, $F_k$ is just the $k$th power of the $\ell_k$-norm of the vector $(f_1, f_2, \ldots, f_m)$. The frequency moments capture many common tasks in data mining. For instance, interpreting 0 as 0, $F_0$ is the number of distinct elements in a data stream. This measure has numerous applications, e.g., in databases (where it is used to estimate the amount of storage needed to represent a multiset) and in networking (where it is used to detect Denial of Service attacks). $F_2$ measures the size of self-joins, which allows one to estimate the cost of merging tables in a database. $F_k$ for $k > 2$ is a statistic reflecting the skewness of a dataset.

My first contribution to this area was a resolution to the space complexity of approximating the number of distinct elements in a data stream. Before my work, it was known that any algorithm returning an estimate $\tilde{F}_0$ with $|\tilde{F}_0 - F_0| \leq \epsilon F_0$ with constant probability must use at least $\Omega(1/\epsilon)$ bits of space, while the best algorithms used space $\tilde{O}(1/\epsilon^2)$. This roughly quadratic gap can make the difference in practice between having a .01% approximation, say, or having to settle for a 1% approximation. In [18], Piotr Indyk and I showed that any algorithm must use $\Omega(1/\epsilon^2)$ space, and thus the inefficiency in current algorithms is unavoidable. Later, I generalized this [30] to show an
$\Omega(1/\epsilon^2)$ bound for approximating $F_2$, while the previous lower bound was only $\Omega(1/\epsilon)$. The proof techniques established a number of new results in combinatorics and communication complexity, including improved bounds on the number of bipartite graphs with a given minimum degree and the communication complexity of approximating the Hamming distance. Later, similar techniques were used by Andoni, Indyk, and Patrascu [3] to give tight lower bounds for the space complexity of the approximate nearest neighbor problem.

Next, Piotr and I turned to studying the space complexity of $F_k$, $k > 2$. A long sequence of work on lower bounds established an $\Omega(m^{1-2/k})$ bound for any algorithm approximating $F_k$ within a constant factor [2, 4, 10]. On the upper bound front, there was an $\tilde{O}(m^{1-1/k})$-space algorithm [2], and a recent $\tilde{O}(m^{1-1/(k-1)})$ upper bound of [13]. We resolved this final open problem of [2], giving a matching $O(m^{1-2/k})$ upper bound [19], up to logarithmic factors. Our techniques significantly departed from previous ones, and were later used by Bhuvanagiri et al [6] to improve the logarithmic factors. Using the connection between frequency moments and norms mentioned earlier, we also showed how to estimate the $\ell_k$-distance between two vectors whose coordinates are given in a data stream in arbitrary order, matching known lower bounds up to logarithmic factors.

Recently, Xiaoming Sun and I established a new suite of bounds on the space complexity of computing and approximating the longest common and increasing subsequences [29] of a data stream. These problems occur in computational biology, especially when measuring the similarity of long genome sequences or computing the edit distance between them. We resolved the main open question in a paper of Liben-Nowell, Vee, and Zhu [23].

### 1.2 Private approximations

Another problem I considered in this area was that of two parties privately approximating the distance (Euclidean or Hamming) between their private input vectors. In practice, telephone companies, ISPs, or web search engines may want to compute joint statistics on their data, without revealing any unnecessary information. It is known, even without privacy, that computing the exact distance between two vectors has randomized communication complexity linear in the dimension of the vectors. However, if one is content with an approximation, then polylogarithmic communication suffices. Unfortunately, one cannot easily turn any protocol computing an approximation to one that does so privately. This is because the approximation itself may reveal additional information about the other party’s input, i.e., more than what the exact answer reveals. For instance, consider a protocol which changes the least significant bit of the distance to agree with a certain bit in one of the party’s inputs. Then the output is a $\pm 1$ approximation, but it is not private.

Before I began working on this problem, it was known that one could privately approximate the Hamming distance with $\tilde{O}(\sqrt{n})$ communication, where $n$ is the dimension of the vectors [16]. It was entirely conceivable that any private protocol for the Hamming distance requires $\Omega(\sqrt{n})$ communication, thus showing that privacy is inherently harder to achieve. Piotr and I showed that this is not the case, by giving an upper bound with only polylogarithmic communication [20], almost matching the setting without privacy. Moreover, the upper bound even holds for the Euclidean distance. Later, our private approximation protocol for the Euclidean distance was used as a subroutine by Strauss and Zheng [28] for privately approximating heavy hitters.

In the future I would like to develop efficient private approximation protocols for other primitives that are useful in practice.
1.3 Graphs spanners

Consider the problem of finding a sparse subgraph $H$ of an arbitrary unweighted, undirected graph $G$ so that for all pairs of vertices $u, v \in G$, $\delta_H(u, v) \leq \delta_G(u, v) + k$, where $\delta_H(u, v)$ denotes the shortest path distance from $u$ to $v$ in $H$, and $k$ is a small positive integer known as the distortion. Such an $H$ is called an additive spanner of $G$. In practice, one can replace a complicated network by a much simpler (sparser) one while still roughly preserving all pairs of distances. As the time complexity of many graph algorithms is proportional to the number of edges, replacing the input graph $G$ with $H$ can result in a large speedup. In the streaming model, by storing the edges of $H$ rather than those of $G$, one can significantly reduce the space complexity.

I improved the known lower bound on the number of edges required of such subgraphs. In particular, for distortion $k = 2^r - 1$ spanners, I showed there exist graphs for which any such $H$ must have $\Omega(n^{1+1/r})$ edges [32], improving the previous lower bound of $\Omega(n^{1+2/(3r)})$ for any $r > 5$.

Unfortunately, for constant $k$, the best upper bound [5] is only $O(n^{4/3})$ - that is, it is even independent of $k$! One of my main goals in the near future is to understand this problem better, and bring the bounds closer to the truth. It should not be very difficult to resolve this question for certain, specific families of graphs, which hopefully will shed light on the general case.

1.4 Other related research

I designed an algorithm [31] for the minimum common integer partition problem, which is a measure of the distance between two strings that occurs in computational biology [11] when estimating the similarity between gene sequences. My algorithm improved the known approximation ratio by a constant factor, and even has a faster running time.

Also, Hanson Zhou and I designed an algorithm for recovering a planted clustering in a dataset [38] by looking at very short random walks in an appropriate graph used to estimate the distance between two sets of features. The algorithm is more efficient than previous ones, and is possibly even practical, though the class of datasets it can correctly cluster is narrower.

2 Efficient cryptographic protocols

2.1 Key exchange

One of the oldest cryptographic protocols is Diffie-Hellman key-exchange, which appeared in the 1970s. If Alice has a secret exponent $a$, Bob a secret exponent $b$, and $g$ is a generator of $\mathbb{F}_p^*$ for some prime $p$, then they can agree on the shared key $g^{ab}$ over an insecure channel. A natural question is what happens if the parties work in an extension field $\mathbb{F}_{p^r}$ for some $r > 1$? It was argued by Brouwer, Pellikan, and Verheul [8], as well as Rubin and Silverberg [27], that one can substantially compress elements in $\mathbb{F}_{p^r}$ by working in certain small subgroups of $\mathbb{F}_{p^r}$, while preserving the security of working in the large group. This results in more efficient protocols since Alice and Bob can transmit and compute in the small group, while enjoying the security of the large group.

Before working on this problem, it was known how to work in a subgroup of $\mathbb{F}_{p^6}$, resulting in protocols which were about 3 times more efficient than working in the full group $\mathbb{F}_{p^6}$ [8, 27]. It was, however, unknown if one could work in small subgroups of $\mathbb{F}_{p^n}$, for $n > 6$, that would lead to even more efficient protocols. Marten van Dijk and I partially answered this question, showing this is indeed the case for many cryptographic applications. For instance, for digital signatures,
encryption, voting protocols, and generating a sequence of session keys, we showed how to achieve an \(\frac{n}{\phi(n)}\) efficiency improvement for any positive integer \(n\) [15], where \(\phi(\cdot)\) is the Euler-totient function. In practice, one can take \(n = 30\), and this results in a \(5/4\) efficiency gain over all known protocols for a variety of applications. For this case, myself and others developed additional optimizations and provided an implementation, demonstrating its utility in practice [14].

There are still applications, such as agreeing on a single session key, for which we do not know how to save by working in small subgroups of \(\mathbb{F}_{p^n}^*\) for \(n > 6\). This is a fascinating avenue for future research, and is related to the problem of showing that certain algebraic tori are rational.

### 2.2 Broadcast encryption

Imagine there is a single server sending broadcasts along a channel to clients listening to the channel. In many applications, such as PayPerView TV and music on demand, one only wants certain privileged users, e.g., those who pay their monthly bills, to decipher the broadcast. Therefore, the server needs to distribute keys to certain groups of users so that for any possible set of privileged users, the server can use these keys to efficiently communicate the desired content. Naturally, there is a tradeoff between the server storage, user storage, communication complexity, computational complexity, types of privileged sets the scheme can handle, and the security guarantee.

My main result in this area is a complete characterization of the tradeoff between these parameters for the arguably most popular model of information-theoretic broadcast encryption known as the subset-cover framework [24]. Craig Gentry, Zulfikar Ramzan, and I devised a new protocol [17] within this framework realizing any possible tradeoff between these parameters, and matching (up to small factors) an essentially trivial lower bound. Previously, there was a non-constructive proof [22] of existence of such protocols, but we were the first to provide an explicit construction. The explicit construction has many advantages over a probabilistic construction, resulting in exponentially faster broadcast time and smaller storage. The techniques are algebraic, associating users with points in finite fields, and considering multivariate polynomials and varieties over these fields.

Another problem I contributed to was the free rider problem. Here the main question is if we allow a small number of non-privileged users to decrypt each broadcast, called free riders, can we significantly save resources? This may have commercial impact, since the small cost of allowing a few free riders may be dwarfed by the communication and computational savings of the company. Prior to my work it was known [1] that there exist somewhat artificial schemes for which it is hard, even to approximate, a communication-optimal assignment of free riders, unless \(P = NP\).

My contribution was to show that actually for a few very popular schemes in the subset cover framework, one can find the optimal assignment of free riders extremely efficiently [26]. Moreover, Zulfikar Ramzan and I designed an algorithm, which, if one relaxes the problem to only require a good approximation to the optimal solution, runs in roughly \(O(rf^{1/3})\) time, where \(r\) denotes the number of non-privileged users, and \(f \leq r\) denotes the number of free riders one can tolerate. The algorithm and the explicit protocol above have both been patented.

In the future, I plan to study the recent broadcast encryption schemes achieving computational security [7], and see if my work can have an impact there.

### 2.3 Private information retrieval and locally decodable codes

This is an old favorite problem of mine, with many applications. Suppose there is a database held by a server that one would like to query. The catch is that this needs to be done privately, so that
the server does not learn which parts of the database are queried. This problem arises in practice when querying patent or medical databases. One trivial, impractical solution is for the server just to send the entire database, and then the user can privately make whatever queries he/she wants. Perhaps surprisingly, under certain number-theoretic assumptions, and if we assume the server is polynomial-time, it is possible to privately make individual queries to the database with only polylogarithmic (in the database size) communication [9].

My first contribution was to consider what happens if we extend the model to require server privacy as well. By server privacy, I’m supposing that certain information in the database is sensitive, and users should not be able to retrieve it. A trivial solution is for the server just to omit this sensitive information from the database, and then run the cryptographic protocol described above. However, what if individual pieces of information are not sensitive, but after combining several non-sensitive pieces together, a user can infer something sensitive? This has been well-studied when we don’t require that the user’s queries be kept private, and is called inference control. Jessica Staddon and I proposed and patented a new primitive, called private inference control, integrating user privacy into inference control [35]. It is intuitively difficult to achieve such privacy. How can the server block inferences if it doesn’t even know what the user is querying? We showed how to achieve very efficient protocols, optimal up to logarithmic factors.

Next, I revisited the problem of private information retrieval itself. What happens if the server is very powerful, and one is not willing to believe the unproven assumptions necessary for the construction mentioned above? One solution is to consider $k$ non-communicating servers, each holding a copy of the database. Surprisingly, under no additional assumptions, the user can privately query the database with substantially sublinear communication [12]. Sergey Yekhanin and I provided several new algebraic constructions of such protocols based on partial derivatives [36]. Some of these protocols have been recently improved [37]. Nevertheless, in the setting where up to $k/2$ of the $k$ servers can collude, our protocol achieves the best known communication. Also, for 2 servers our protocol achieves the smallest amount of computation known in the preprocessing model.

Private information retrieval is intimately related to locally decodable error-correcting codes, that is, codes for which it is possible to recover individual bits of an $n$-bit message by only probing a few (constant) positions in the possibly corrupt codeword. Lower bounds on the length of these codes are notoriously difficult to prove, and the previous best such bounds have come from quantum information theory [21]. I improved the previous lower bound for any odd number of queries without using quantum information theory. For instance, for 3 queries I improved the lower bound from $\Omega(n^2/\log^2 n)$ to $\Omega(n^2/\log n)$, and even further to $\Omega(n^2/\log \log n)$ if the code is linear [34].

This is still far from known upper bounds, and I want to help close this gap in the future.

2.4 Secure function evaluation

I devised a new protocol for computing any efficiently computable function, whose inputs are shared between two parties, even if one of the parties is malicious. It was known how to do this with polynomial communication in the size of a circuit computing the function, but the actual communication is large in practice. Using cut-and-choose techniques and Ramanujan graphs, I recently improved at least one measure of efficiency of all known protocols for this problem [33].

Very recently I have had some ideas on developing more efficient protocols in the malicious model which have communication even less than the circuit size for computing the function. The hope is to make the theoretical result of Naor and Nissim [25] in this setting more practical by using cut-and-choose techniques rather than PCPs. I am actively investigating this.
References


