Research Statement

Shikha Singh

Overview of Interests and Philosophy

My research interests fall into the areas of algorithms, algorithmic game theory, and mechanism design.

As a theoretician, I like working on theory problems that are inspired by practice. The high-level goal of my research is to question the applicability and versatility of existing models used to solve an algorithmic problem. Does it capture how the algorithm is used in practice? Can the model be adapted, based on practical insights, to develop simpler and faster algorithms? Examples of this approach in my work are:

- In my work on adaptive Bloom filters [3], we question the guarantees of a Bloom filter [5], a ubiquitous data structure, and find that it leaves performance on the table for most applications where it is used.
- My work on rational proofs [7–9] follows the same line of inquiry. We look at the context in which interactive proofs are used—computation outsourcing—and show that updating the model to reflect the rational nature of these applications significantly improves the performance and simplicity of the protocols.
- We initiate the use of a resource-augmentation approach to solve offline optimization problems, in particular, the k-forest problem [1]. Resource-augmentation approach is useful in explaining why some optimization problems that are hard in the worst-case, admit simple and fast heuristics in practice.

Mechanism Design and Interactive Proofs

In mechanism design, the objective is to design the rules of a game, played by strategic self-interested agents, so as to achieve a particular desired outcome. It is sometimes called "reverse game theory".

Mechanism design has been mostly studied in the context of designing auctions. In my work on rational proofs, we use the principles of mechanism design to design interactive protocols where *rational* provers still find it in their best interest to report the answer truthfully to a verifier. As part of the work we design a solution concept suitable for extensive-form mechanisms with imperfect information.

Rational Proofs

Most computation today is not done locally by a "client" but rather outsourced to third-party "service providers" in exchange for money. Trading computation for money brings up two problems—(a) how to guarantee correctness of the outsourced computation (without the client redoing the computation), and (b) how to design the payment scheme. The two problems are closely related—that is, ideally, we want the payment scheme to be such that it incentivizes service providers to perform the computation correctly.

Interactive proofs are the most well-studied and widely used approach to verify correctness of outsourced computation. In an interactive proof, a weak client or *verifier* interacts with powerful servers, or *provers* to determine the correctness of their claim. At the end, the verifier probabilistically accepts or rejects the claim. Interactive proofs are extremely powerful because they guarantee correctness and soundness against arbitrary, even *malicious*, provers. That is, the verifier always accepts the claim of honest provers and no strategy of arbitrary provers can make the verifier accept a false claim with non-negligible probability. Furthermore, the model as-is does not allow for a non-trivial payment structure.[1]

While recent breakthroughs in IPs have lead to very efficient protocols [12–15], how payments can be leveraged to gain simplicity and speed ups in these outsourcing settings is not well understood.

My work focuses on designing mechanisms for interactive proofs, where the provers are not malicious but rather *rational*, that is, they want to maximize their payment. Thus, the goal is to design protocols and payment schemes such that the verifier learns the correct answer from the provers.

Cooperative Rational Proofs [7, 8]. As a first step towards general incentives, we studied *cooperative rational proofs*, in which provers act as a team to maximize their *total* payment from the verifier.[2] *Utility gap* in rational proof corresponds to soundness gap in IPs and is the amount lost by provers who report the

[1]One can always introduce a trivial payment scheme: pay \$1 to the provers if the verifier accepts, \$0 otherwise.

[2]Classical IPs also have cooperative provers—they act as a team to maximize the verifier's acceptance probability.

incorrect answer. We exactly characterize the power of cooperative rational proofs for different utility gaps. We show that cooperative rational proofs are not only more powerful than classical multi-prover IPs (even for constant utility gap) but that they lead are simple and more efficient protocols.

Non-cooperative Rational Proofs [9]. In non-cooperative rational proofs (ncRIP), the provers are neither totally cooperative, nor totally conflicting, they just want to maximize their *own* payment. In contrast, all other interactive-proof models have incentive structure that fall into the two extremes—provers are either completely cooperative (as in IPs) or are completely conflicting (as in refereed games).

Since the structure of interactive proofs is very different from games that are typically studied in game theory and mechanism design literature, we first define a new solution concept for extensive-form games with imperfect information, *strong sequential equilibrium* (SSE), one of the main conceptual contribution of our work. We also define the notion of utility gap for ncRIP protocols to capture possible deviations and how the provers reason in the extensive-form game induced by the protocol.

Finally, we give tight upper and lower bounds for classes of non-cooperative rational proofs with different utility gaps. We show that not only are these protocols simpler in structure than cooperative rational protocols, but are also more powerful.

Research Aims. The last decade has witnessed exceptional advances in the interactive-proof literature—from sublinear proofs [15] to practical delegation schemes [14], the progress has been very exciting. We can even build verification *systems* based on these proofs [16].

Since we now know that rational proofs lead to simpler and more efficient protocols compared to IPs (even for similar soundness gap guarantees), I want to address the following open problems that essentially ask—how far can we push the recent performance gains of IPs in a rational setting?

- **Rational proofs of proximity.** Interactive-proof systems with sublinear-time verifiers (that is, the verifier cannot even read the entire input) are based on property testing and are called proofs of proximity. These proofs are useful for computation delegation. They have a relaxed soundness guarantee (verifier rejects every input that is "far from the language") but are extremely succinct and efficient. Can rationality strengthen these proofs—in terms of verifications costs or in terms of soundness guarantees?
- **Correctness and soundness of rational vs. interactive proofs.** As part of an ongoing project, I prove the necessary and sufficient conditions under which any interactive protocol can be turned into a rational protocol and vice-versa. In the process, I discovered an interesting distinction between how correctness and soundness differs in the two models. In interactive proofs, correctness and soundness should hold for "for all inputs x". while in rational proofs the guarantees are *instance sensitive*, i.e., they hold "for a given input x." Furthermore, interactive proofs are only defined for proving membership (that is, $x \in L$), while in rational proofs the provers can prove $x \in L$ or $x \notin L$. I want to understand the implications of these differences in guarantees, especially in the context of outsourcing applications.
- **New scoring rules.** Scoring rules are an important tool to design efficient rational proofs. All known rational proofs use *Brier's scoring rule.* However, the use of Briers scoring rule results in protocols with weak utility gaps. Can we design stronger scoring rules that lead to constant-utility gap protocols?

Extensive-form mechanisms with imperfect information.

A few extensive-form mechanisms with *perfect information* have been studied in context of auctions and social choice theory [6, 10, 11]. However, perfect information may be hard to achieve and sometimes information asymmetry is crucial to designing the mechanism (as in interactive proofs).

Our new new solution concept—*strong sequential equilibrium* (SSE) [9]—is a strong refinement of sequential equilibrium. We believe it is a suitable solution concept to analyze general extensive-form mechanisms that have imperfect information.

Research Aims. I want to generalize the results on existing extensive-form mechanisms with perfect information to the imperfect-information setting and analyze them using SSE.

Algorithms and Data Structures

Conventionally, the efficiency of an algorithm is determined by the number of computations it performs. However, when the size of the input is too large to fit in in the memory of a device, computing on it requires transferring smaller chunks of data between an external disk and memory. These input/output operations then become a bottleneck, and *external memory algorithms* are developed to minimize the transfer cost.

Furthermore, if the data is large and is not available a priori but arrives over time, we design *streaming* algorithms that do well without knowing the future and use a limited amount of space.

Next, I describe my work and future research aims on external-memory and streaming algorithms. I conclude with my work on approximation algorithms based on a duality-based resource augmentation approach.

Dictionary Data Structures

A dictionary is a fundamental data structure that supports two main operations—lookup and updates. I am interested in designing I/O efficient and secure dictionary data structure.

Adaptive Bloom filters to speed up dictionaries [3]. Dictionary operations for the common case when the set S is too big to be stored locally (e.g., it does not fit in RAM), and so is stored remotely (e.g., on disk or across the network) involve expensive queries to the remote representation of S. In such cases, Bloom filters [5] and other approximate membership data structures (AMQs) are used to speed up the remote queries, as they filter out almost all *negative queries* (queries for $x \notin S$).

While Bloom filters are frequently used to speed up these dictionaries, they only provide performance guarantees for randomly-chosen workloads, and they do not give speed ups in general. To do well against arbitrary, even adversarial workloads, we need AMQs that adapt to the workload. We design adaptive AMQs that guarantee optimal cost for lookups and updates and use near-optimal space [3].

History-independent external-memory dictionaries. In collaboration with Sandia National Laboratories, we designed I/O-efficient history-independent dictionaries that have good worst-case bounds [2].

A data structure is *history independent* if it does not leak any information about how it was used in the past. In particular, if a malicious observer takes hold of the data structure he should only be able to see the contents and should not be able to glean any information about the *history* of its operations. For example, a data structure where recently-added elements are clustered together is not history independent.

History independence is extremely essential in applications where data must be shared without revealing sensitive redactions or other modifications. As history independent data structures have canonical structures, they also have important concurrency applications.

We design the first history-independent *cache-oblivious* dictionary [2].[3] We also give the first history-independent sequential file-maintenance data structure, which stores files in order in a linear array in which recently-added files cluster together, so it is surprising that it can be made history independent.

Research Aims. As part of our project on history independence [2], we realized that while most data structures used in a sensitive setting would like to have *strong* history independence (SHI), the definition of SHI in the literature is in fact too strong, and its security guarantees take a huge performance hit to maintain. Roughly speaking an SHI data structure must have a canonical representation at all times.

However, in the current definitions of history independence, time only proceeds when a user performs operations. In practice, system designers often implement background processes for necessary cleanups when the data structure is sitting idle. I want to define a more practical notion of strong history independence, which gives similar security guarantees but is more performance friendly.

Streaming Algorithms and Optimization

Light hitters problem. In an ongoing collaboration with Sandia National Laboratories, we solve a streaming problem that requires exact element counts and immediate event triggers. Specifically, given a stream of N keys and a threshold T, we must report a key immediately upon its Tth occurrence. The threshold T is allowed to be small with respect to N.

The light-hitter problem models real-time monitoring applications which need to give timely reports when an anomalous event occurs. We give an I/O-efficient data structure, the *popcorn filter*, for solving the light-hitters problem when the key-counts are bounded above by a power-law distribution.

Run generation [4]. Sorting big data is one of the most fundamental computational tasks. External merge sort is a well-known and widely used external-memory sorting algorithm. In a collaborative paper [4], we improve the first phase of external merge sort: we give an optimal online algorithm for the *run generation*

[3]A cache-oblivious data structure is memory-hierarchy universal, in that it has no memory-hierarchy-specific parameter.

problem [4]. Interestingly, a similar heuristic was proposed previously and shown to work well in practice.

Primal-dual with resource augmentation. A major direction in algorithmic research is to explain the gap between the observed practical performance and the provable worst-case guarantees of an algorithm. This gap often arises due to pathological worst-cases that do not arise in practice. The model of *resource augmentation* has been successfully used in the area of online scheduling to rule out such worst cases.

In the resource-augmentation model, an algorithm is given some additional power and its performance is compared against that of an optimal algorithm without the additional power.

As part of my *Chateaubriand Fellowship*, which was offered to me by the Embassy of France in the U.S. to conduct research in France, I worked with researchers at University of Évry Val d'Essonne, to initiate the study of using resource augmentation for offline optimization problems that admit no reasonable approximations in the worst-case. In particular, we used a primal-dual approach to design the first polynomial-time constant-factor algorithm for the k-forest problem in the resource-augmentation model. In the k-forest problem, given an edge-weighted graph $G(V, E)$, a parameter k and a set of m demand vertex pairs, we need to find a minimum-cost subgraph that connects at least k demand pairs.

Research Aims. While the resource-augmentation model has been widely and successfully used for many online scheduling problems, it has surprisingly not been used for (offline) optimization problems. I want to extend the primal-dual approach we used on the k-forest problem to design algorithms in the resource augmentation model for other hard problems which currently admit no meaningful approximation guarantees.

References

[1] E. Angel, N. K. Thang, and S. Singh. Approximating k-forest with resource augmentation: A primal-dual approach. *International Conference on Combinatorial Optimization and Applications*, 2017. To appear.

[2] M. Bender, J. Berry, R. Johnson, T. M. Kroeger, S. McCauley, C. A. Phillips, B. Simon, S. Singh, and D. Zage. Anti-persistence on persistent storage: History-independent sparse arrays and dictionaries. In *Proc. 35th Symposium on Principles of Database Systems*, pages 289–302. ACM, 2016.

[3] M. A. Bender, M. Farach-Colton, M. Goswami, R. Johnson, S. McCauley, and S. Singh. Bloom filters, adaptivity, and the dictionary problem. *arXiv preprint arXiv:1711.01616*, 2017.

[4] M. A. Bender, S. McCauley, A. McGregor, S. Singh, and H. Vu. Run generation revisited. In *Proc. 26th Int'l Symposium on Algorithms & Computation*, pages 703–714, 2015.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] J. Chen, A. Hassidim, and S. Micali. Robust perfect revenue from perfectly informed players. In *Innovations in Theoretical Computer Science*, pages 94–105, 2010.

[7] J. Chen, S. McCauley, and S. Singh. Rational proofs with multiple provers. In *Proc. 7th Innovations in Theoretical Computer Science*, pages 237–248, 2016.

[8] J. Chen, S. McCauley, and S. Singh. Rational proofs with multiple provers (full version). *arXiv preprint arXiv:1504.08361*, 2017.

[9] J. Chen, S. McCauley, and S. Singh. Rational proofs with non-cooperative provers. *arXiv preprint arXiv:1708.00521*, 2017.

[10] J. Chen, S. Micali, and P. Valiant. Robustly leveraging collusion in combinatorial auctions. In *Innovations in Theoretical Computer Science*, pages 81–93, 2010.

[11] J. Glazer and M. Perry. Virtual implementation in backwards induction. *Games and Economic Behavior*, 15(1):27–32, 1996.

[12] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proc. 14th Annual Symposium on Theory of Computing*, pages 113–122, 2008.

[13] Y. T. Kalai and R. D. Rothblum. Arguments of proximity. In *Advances in Cryptology*, pages 422–442. 2015.

[14] O. Reingold, G. N. Rothblum, and R. D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proc. 48th Annual Symposium on Theory of Computing*, pages 49–62. ACM, 2016.

[15] G. N. Rothblum, S. Vadhan, and A. Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proc. 45th Annual Symposium on Theory of Computing*, pages 793–802, 2013.

[16] M. Walfish and A. J. Blumberg. Verifying computations without reexecuting them. *Communications of the ACM*, 58(2):74–84, 2015.