Bernoulli Factories and Black-Box Reductions in Mechanism Design

SHADDIN DUGHMI
University of Southern California
and
JASON D. HARTLINE
Northwestern University
and
ROBERT KLEINBERG
Cornell University
and
RAD NIAZADEH
Stanford University

In this letter, we report on our work providing a polynomial time reduction from Bayesian incentive compatible mechanism design to Bayesian algorithm design for welfare maximization problems. Unlike prior results, our reduction achieves exact incentive compatibility for problems with multi-dimensional and continuous type spaces.

1. INTRODUCTION

Does running an incentive compatible mechanism require more computational resources than running an algorithm for the same problem? In other words, is there a computationally efficient reduction from mechanism design to algorithm design? Such a reduction, if it existed, would be an exceedingly powerful tool. System designers could ignore issues of private information and strategic behavior, and instead focus on coming up with algorithms that achieve their objectives in an environment where all information is public. Those algorithms could then be transformed by a general-purpose reduction into mechanisms that work as desired even when users of the system are strategic.

A reduction as envisioned can only exist under specific assumptions. If monetary transfers are not allowed, the Gibbard-Satterthwaite Impossibility Theorem [Gibbard 1973; Satterthwaite 1975] puts severe limitations on the social choice functions that can be implemented in dominant strategy equilibrium. Even for mechanisms with monetary transfers, there are important objectives, such as approximate makespan minimization in job scheduling, that are easy to achieve algorithmically but impossible to implement as a dominant-strategy equilibrium of a mechanism [Nisan and Ronen 2001; Ashlagi et al. 2012] no matter what computational resources one devotes to the mechanism. Finally, there are computa-

Authors' addresses: shaddin@usc.edu, hartline@eecs.northwestern.edu, rdk@cs.cornell.edu, rad@cs.stanford.edu

tional barriers that stand in the way of reducing mechanism design to algorithm design. A long line of work culminating in [Papadimitriou et al. 2008; Dughmi and Vondrák 2015; Dobzinski and Vondrák 2016] furnished examples of social welfare maximization problems for which there exist computationally efficient constant-factor approximation algorithms, yet no computationally efficient mechanism can implement a constant-factor approximation to social welfare in dominant-strategy equilibrium. In particular, this rules out the possibility of an efficient reduction from dominant-strategy mechanism design to algorithm design, even when the objective is social welfare maximization.

These negative results on dominant-strategy implementation motivate relaxing the solution concept to Bayesian incentive compatibility (BIC). The search for black-box reductions from BIC mechanism design to algorithm design was initiated by Hartline and Lucier [2010; 2015]. For single-parameter domains they presented a reduction, parameterized by an arbitrarily small $\epsilon > 0$, that incurs at most ϵ loss in expected social welfare and runs in time $\text{poly}(n, \epsilon^{-1})$, where n denotes the number of agents. The insights underlying this reduction were translated to environments with multi-dimensional type spaces in [Bei and Huang 2011; Hartline et al. 2011], resulting in a reduction that makes use of an oracle for computing *interim allocation functions*, a #P-hard problem. Alternatively, rather than exactly computing the interim allocation function one can estimate it by sampling. This results in a reduction that achieves ϵ-BIC rather than BIC, and whose running time is $\text{poly}(n, \epsilon^{-O(\Delta)})$ where Δ is a parameter representing the dimensionality of the type spaces. For discrete (i.e., zero-dimensional) type spaces, the incentive-compatibility of the reduction was improved from ϵ-BIC to BIC by [Hartline et al. 2015].

These positive results raised hope that there might be a polynomial time reduction from Bayesian incentive compatible mechanism design to algorithm design for welfare maximization problems in general. However, as explained in §2.4 below, it was actually quite unclear whether such a reduction could exist when type spaces are continuous and multi-dimensional. Our paper [Dughmi et al. 2017] resolves this question affirmatively: we present a polynomial time reduction that achieves exact BIC and places no restriction on agents' type spaces. The key novel ingredient in our reduction is a set of algorithms for mapping a tuple of input distributions to an output distribution, in a computational model where the input and output distributions are represented implicitly by sampling oracles rather than explicitly by listing the probability of each sample point of the distribution. This model, which we call *expectations from samples*, generalizes the literature on *Bernoulli factories* in probability theory [Keane and O'Brien 1994; Łatuszyński 2010].

Our reduction builds heavily upon ideas from the prior work on black-box BIC reductions, and we devote §2 to sketching some of those ideas and highlighting the key barrier that our work overcomes. In §3-4 we explain some of the main novel ideas underlying our reduction. We conclude in §5 with some remarks about limitations of our approach and open questions.

2. BACKGROUND AND OVERVIEW OF THE REDUCTION

We assume each agent i has a private type t^i independently sampled from type space $\mathcal{T}^i$; the profile of all agents' types is denoted by $\mathbf{t}$. There is a set $\mathcal{O}$ of outcomes, and agent i's value for outcome o is given by a function $\mathrm{val}^i(t^i, o)$ taking values in $[0, 1]$. Agents are risk-neutral and have quasi-linear utility.

Our reduction is given black-box access to an algorithm computing an allocation function $\mathcal{A} : \prod_i \mathcal{T}^i \to \mathcal{O}$. It is also given a sampling oracle for each agent's type space. The objective is to use these oracles to implement, in polynomial time, a mechanism for which truthtelling is a Bayes-Nash equilibrium and whose expected social welfare loss, relative to $\mathcal{A}$, is bounded by ϵ:

$$\mathbf{E}\left[\sum_i \mathrm{val}^i(t^i, \mathcal{M}(\mathbf{t}))\right] > \mathbf{E}\left[\sum_i \mathrm{val}^i(t^i, \mathcal{A}(\mathbf{t}))\right] - \epsilon.$$

For an allocation function $\mathcal{A}$, the *interim allocation function* for agent i outputs the outcome distribution obtained by averaging over the randomness in other agents' reports:

$$\mathcal{A}^i(t^i) = \mathbf{E}\big[\mathcal{A}(t^i, \mathbf{t}^{-i})\big].$$

2.1 Ironing as type resampling

For linear single-parameter domains — in which types are scalars, outcomes are vectors indexed by agents, and $\mathrm{val}^i(t, o) = t \cdot o_i$ — an allocation function is implementable by a BIC mechanism if and only if each of its interim allocation functions is monotone. The key insight of Hartline and Lucier [2010; 2015] was that even if the interim allocation function $\mathcal{A}^i$ instituted by the black-box algorithm is non-monotone, we can make it monotone by "ironing". The ironing procedure identifies a set of intervals on which $\mathcal{A}^i$ is non-monotone and modifies the interim allocation function to be constant on those intervals. As observed by Hartline and Lucier, this can be accomplished the following resampling procedure: if the reported type t^i belongs to an ironing interval I, then we draw a fresh random sample s^i from I and substitute s^i in place of t^i when calling the allocation function. (For convenience, define $s^i = t^i$ in the event that t^i does not belong to one of agent i's ironing intervals.) Let $\bar{\mathcal{A}}^i(t^i)$ denote the interim allocation function of this modified mechanism, i.e. $\bar{\mathcal{A}}^i(t^i) = \mathbf{E}\big[\mathcal{A}^i(s^i) \mid t^i\big]$. The resampling procedure satisfies three crucial properties.

(1) **Distribution preservation:** s^i has the same distribution as t^i.
(2) **Monotonicity:** $\bar{\mathcal{A}}^i(t^i)$ is monotonic in t^i.
(3) **Welfare improvement:** $\mathbf{E}\big[\mathrm{val}^i(t^i, \bar{\mathcal{A}}^i(t^i))\big] \geq \mathbf{E}\big[\mathrm{val}^i(t^i, \mathcal{A}^i(t^i))\big]$.

The first property implies that, if all agents other than than i are truthful, then i is indifferent between bidding against profile $\mathbf{t}^{-i}$ or bidding against $\mathbf{s}^{-i}$. This fact, in conjunction with the second property, implies that the modified mechanism is BIC. The third property ensures that the mechanism's expected social welfare is at least as good as the original algorithm's.

An important obstacle to designing a black-box reduction based on this idea is that the reduction has only oracle access to $\mathcal{A}$, so it cannot directly manipulate

the interim allocation functions $\mathcal{A}^i$. Instead it must simulate the ironing procedure using only sample access to the type distribution of each agent, and oracle access to the allocation function. This inevitably leads to small sampling errors in estimating the ironed allocation function, which introduces the potential for small amounts of non-monotonicity. To avoid spoiling the mechanism's incentive compatibility, the reduction of [Hartline and Lucier 2010] mixes in a "blatantly monotone mechanism": with a small probability, instead of using the resampling procedure described above, the mechanism simply ignores the reports of all agents other than i and implements allocation function that is strictly monotone in i's reported type. The strict incentive for truthtelling instituted by this blatantly monotone mechanism overwhelms the small incentive for misreporting that might potentially be brought about by sampling error in estimating $\bar{\mathcal{A}}^i$.

2.2 Replicas and surrogates

In generalizing the approach of Hartline and Lucier from single-dimensional to multi-dimensional types, the first question that needs to be addressed is: what is the multi-dimensional counterpart of ironing? This question was answered in [Bei and Huang 2011; Hartline et al. 2011].

For general (as opposed to single-dimensional) type spaces, Rochet's [1987] well-known characterization of truthfulness says that an allocation function is implementable by a BIC mechanism if and only if each of its interim allocation functions satisfies *cyclic monotonicity*: for every finite sequence of types of agent i, $t^i_1, t^i_2, \ldots, t^i_m$, and every permutation π, we have

$$\sum_{j=1}^{m} \mathrm{val}^i(t^i_j, \mathcal{A}^i(t^i_j)) \geq \sum_{j=1}^{m} \mathrm{val}^i(t^i_{\pi(j)}, \mathcal{A}^i(t^i_j)).$$

(To verify this condition for all permutations, it suffices to verify it for cyclic permutations, which explains the term "cyclic monotonicity".) In light of the ideas presented in the preceding subsection, then, it makes sense to interpret multi-dimensional ironing as a type resampling procedure that satisfies distribution preservation, cyclic monotonicity, and welfare improvement. Bei and Huang [2011], and, independently, Hartline et al. [2011] discovered a type resampling procedure that satisfies all three properties. It couples the true type and the resampled "surrogate" into a joint distribution on pairs (t^i, s^i) that maximizes $\mathbf{E}\big[\mathrm{val}^i(t^i, \mathcal{A}^i(s^i))\big]$ subject to the constraint that the marginal distributions of t^i and of s^i are both equal to the type distribution of agent i.

When the type space $\mathcal{T}^i$ is finite and t^i is uniformly distributed on $\mathcal{T}^i$, the coupling can be computed by solving a maximum weight matching problem on a bipartite graph whose two vertex sets are both in one-to-one correspondence with $\mathcal{T}^i$; the edge joining t^i_0 to t^i_1 is defined to have weight $\mathrm{val}^i(t^i_0, \mathcal{A}^i(t^i_1))$. More generally, if $\mathcal{T}^i$ is finite and the distribution of t^i is non-uniform, the optimal coupling can be computed by reducing to a minimum-cost circulation problem.

However, for continuous type spaces the coupling is an infinite object and cannot be directly manipulated algorithmically. If one relaxes the welfare improvement property to approximate welfare improvement,

$$\mathbf{E}\big[\mathrm{val}^i(t^i, \bar{\mathcal{A}}^i(t^i))\big] \geq \mathbf{E}\big[\mathrm{val}^i(t^i, \mathcal{A}^i(t^i))\big] - \epsilon,$$

then there is a workaround that consists of approximating the continuous distribution on $\mathcal{T}^i$ by the empirical distribution of a finite random sample. In the *replica-surrogate matching reduction* of [Hartline et al. 2011], one draws two m-tuples of independent random samples from type space: a profile of *replicas* $r_1, \ldots, r_m$ and a profile of *surrogates* $s_1, \ldots, s_m$. A random one of the replicas r_j is replaced with the agent's reported type t^i (which, assuming truthful bidding, is drawn from the same distribution as r_j). Edge weights between replicas and surrogates are defined by $\mathbf{E}\big[\mathrm{val}^i(r_j, \mathcal{A}^i(s_k))\big]$ as above, and a maximum-weight perfect matching is computed. The type resampling procedure then finds the edge in the matching that contains t^i and outputs its opposite endpoint. This satisfies exact BIC, and it satisfies ϵ-approximate welfare improvement if the number of samples, m, is large enough for the empirical distribution of the replicas and surrogates to serve as a good approximation of the true type distribution. For this purpose, $m = \epsilon^{-O(\Delta)}$ samples suffice, where Δ roughly corresponds to the dimension of the type space; see [Hartline et al. 2011] for details.

2.3 Correcting sampling error using strictly IC mechanisms

The replica-surrogate matching reduction is not computationally efficient because computing the edge weights in the bipartite graph requires computing the interim allocation function $\mathcal{A}^i$, which in general is #P-hard. (It requires averaging over the product of $n-1$ other players' type spaces.) One can estimate the edge weights by sampling the allocation function's output on random profiles of the other players' types and averaging the results. This estimation inevitably involves some sampling error, but the error tends to zero with the number of samples, m. Consequently the replica-surrogate matching reduction using these estimated edge weights is ϵ-BIC where $\epsilon \to 0$ as $m \to \infty$, but is not necessarily BIC for any finite m. (See Lemma 4.1 of [Hartline et al. 2015] for an example.)

As noted earlier, for single-dimensional type spaces this difficulty is resolved in [Hartline and Lucier 2010] by running a "blatantly monotone" mechanism with some small probability, whose purpose is to correct the small incentives for misreporting caused by the error in estimating the edge weights $\mathrm{val}^i(r_j, \mathcal{A}^i(s_k))$. In the case of the replica-surrogate matching reduction, it is more difficult to correct for these errors. Ironically, the trouble is with small misreports, those in which the reported type assigns almost the same value, to every outcome, as the true type. A strictly incentive compatible mechanism has limited power to punish small misreports, and this power is further limited by the fact that the strictly IC mechanism is used only with small probability. On the other hand, since the function mapping the estimated edge weights to a maximum-weight matching is discontinuous, small misreports that cross such a discontinuity can greatly affect the mechanism's outcome distribution. The problem is analogous to the problem of overfitting in machine learning, and the solution is the same in both cases: *regularization*. Hartline et al. [2015] modify the objective function of the maximum-weight matching problem by adding a strongly concave function that promotes outcome distributions that balance probabilities among many alternatives. The result is that the outcome distribution varies slowly as a function of an agent's reported type, which limits the potential benefit of small misreports. When the type space is finite, it is possible to construct a strictly IC mechanism that penalizes misreports strongly

enough to correct these incentive constraint violations.

2.4 A barrier at dimension two

The story of progress up to this point may give the impression that a black-box reduction from BIC mechanism design to algorithm design would inevitably exist for multi-dimensional continuous type spaces as well. Actually, there was a good reason to doubt the existence of such a reduction. All of the previously described reductions tolerated small errors in estimating interim allocation functions, and corrected those errors by mixing in a strictly IC mechanism with small probability. The reason this works is that when the type space is finite or one-dimensional, any mechanism whose allocation function is sufficiently close to IC can be made perfectly IC by adding a strictly IC allocation function. This is no longer the case when the type space has two or more dimensions.[1]

To explain why, it is helpful to specialize to the case when there are n outcomes and type space is an open subset of $\mathbb{R}^n$, with the n components of a type vector representing the agent's valuations for the n outcomes. An interim allocation function, mapping types to distributions over outcomes, can thus be encoded as a function from $\mathbb{R}^n$ to itself. When this function is differentiable, its partial derivatives form a square matrix called the *Jacobian*, and a twice-differentiable allocation function is truthful if and only if its Jacobian matrix is symmetric and positive semidefinite (PSD) at every point of the type space [McAfee and McMillan 1988]. Randomizing between one mechanism and a second, strictly IC, mechanism has the effect of taking a convex combination of the first mechanism's Jacobian with a symmetric positive definite matrix. If the first mechanism's Jacobian matrix is symmetric but not PSD, this operation can make it PSD. But if the first mechanism's Jacobian is asymmetric, then combining it with a symmetric positive definite matrix will do nothing to fix the asymmetry. This issue does not arise unless the dimension of the type space is at least two, because an n-by-n matrix cannot be asymmetric unless $n \geq 2$.

Thus, extending the existing black-box reduction methods to accommodate multi-dimensional continuous type spaces requires coming up with a type resampling procedure that, when composed with the black-box allocation function, has a symmetric Jacobian matrix. Given that we can only evaluate the allocation function by sampling, and that the set of symmetric n-by-n matrices has measure zero when $n \geq 2$, it was not clear *a priori* whether any such procedure would exist. The key to overcoming this barrier was to incorporate, and extend, a family of exact sampling methods developed in the applied probability literature on *Bernoulli factories*. The next section details these ideas.

3. EXPECTATIONS FROM SAMPLES AND BERNOULLI FACTORIES

One way of looking at the replica-surrogate matching reduction of [Hartline et al. 2011] for a fixed agent i is by considering a different auction where there exists a buyer for each replica and an item for each surrogate. If the buyer j has type r_j, define her value for the item k to be $\mathbf{E}\big[\mathrm{val}^i(r_j, \mathcal{A}^i(s_k))\big]$. Note that as before the true

[1]Here, and for the remainder of this subsection, we abuse terminology by calling an allocation function IC if it is equal to the interim allocation function of a BIC mechanism.

agent i is indeed a random one of these buyers. Now, the reduction in [Hartline et al. 2011] simply runs an incentive compatible social welfare maximizing mechanism for this auction problem, also known as the VCG mechanism, and re-samples the true agent's type according to the matching selected by the VCG allocation. As long as this mechanism is incentive compatible the final reduction is exact BIC, and as long it preserves the maximum social welfare (either exactly or by an additive loss ϵ per-replica) the final reduction satisfies ϵ-approximate welfare improvement per-agent for large enough m, as stated earlier.

3.1 Sampling a cycle monotone perfect matching

As a workaround to not calculate the interim allocation rule, our paper [Dughmi et al. 2017] looks at a relaxed problem in which we only need to *sample* a perfect matching, rather than calculating the expected values for replica-surrogate edges and computing the maximum-weight perfect matching. Moreover, we need our sampling procedure to output a randomized matching satisfying three properties:

(1) its social welfare loss per-replica is bounded by an additive ϵ in expectation;
(2) it is *exactly* cycle monotone (truthfully implementable with some payments);
(3) given sampling oracles $\{\text{val}^i(r_j, \mathcal{A}^i(s_k))\}_{j,k}$, it can be sampled efficiently.

Given such a sampling procedure, we can also compute the truthful payments implicitly in polynomial time by using the result of [Babaioff et al. 2015]. This then gives us a polynomial time reduction that is exact BIC and satisfies ϵ-approximate welfare improvement relative to the ideal reduction of [Hartline et al. 2011] per-agent (which already has a welfare loss bounded by ϵ relative to the allocation algorithm $\mathcal{A}$ for each agent).

3.2 Expectations from Samples

The problem of sampling an exactly cycle monotone perfect matching is an example of our *expectations from samples* computational model. This model calls for sampling a distribution that is a precise function of the expectations of some random inputs, which themselves are given by sample access. Formally, let $f : [0,1]^m \to \Delta(X)$, where X is an abstract set of outcomes (matchings, in our example). We say an algorithm implements f from samples if, when given sample access to random variables $v_1, \ldots, v_m$ with unknown expectations $\mu_1, \ldots, \mu_m$, its output is distributed precisely as $f(\mu_1, \ldots, \mu_m)$. In our matching problem, we seek a function f which is approximately optimal for the max-weight matching problem, is exactly cycle monotone, and is implementable from samples.

3.3 Single-agent multiple-urns

Our algorithm for sampling a cycle-monotone perfect matching is built on a solution to a simpler problem, also in the expectations from samples model. In this problem, which we call *single-agent multiple-urns*, there is a single agent who needs to be assigned to one of the m different urns. For a given agent type t, every urn i has an associated distribution $D_i(t)$ over $[0,1]$. Assigning the agent to urn i generates a value $V_i(t)$, where $V_i(t) \triangleq \mathbf{E}_{v \sim D_i(t)}[v]$. The objective is to design a polynomial time incentive compatible (or in other words *truthful*) mechanism with a welfare loss bounded by ϵ relative to the maximum welfare, i.e. $\max_i V_i(t)$.

The main restriction in the single-agent multiple-urns that makes it challenging is that we assume the auctioneer does not know the urn valuation functions $V_i(t)$, and only has access to sampling oracles $D_i(s)$ for every urn i and type s. From the above description, it is clear that single-agent multiple-urns is essentially the matching problem when there is only one replica. As we show in §4, the solution to this problem is also the main building block for designing the perfect matching sampling procedure that satisfies (1),(2) and (3).

It is tempting to try the first candidate solution that comes to mind; assigning the *exact* maximum value urn (and charging no money). This is clearly truthful. However, it is not hard to see we cannot implement this allocation rule with only samples from $\{D_i(s)\}$, no matter how many samples we take. In other words, the arg-max function is not exactly implementable in the expectations from samples model. The good news is that this is not the only truthful allocation. In particular, it is well-known that one can penalize deterministic allocations such as the arg-max function by regularizing the welfare objective, and one still gets a truthful allocation. In the mechanism design literature, such allocation functions are often called *affine maximizers*. If the regularizer takes values between 0 and ϵ, the welfare is approximately preserved with at most ϵ loss. Then, we ask the following question: is there a member of the family of maximum regularized-welfare allocations that can be sampled by only efficiently sampling from oracles $\{D_i(t)\}$, where t is the truthful type?

The key technical idea we use to answer this question is generalizing and incorporating Bernoulli factories [Keane and O'Brien 1994; Łatuszyński 2010]. We next introduce this tool and explain its applications in solving relevant questions in the expectations from samples computational model. We will return to our mechanism design problem afterward, to see how to apply these solutions.

3.4 Bernoulli factories and races

The Bernoulli factory problem, introduced in [Keane and O'Brien 1994], is the following. Suppose you have access to a coin with unknown bias p. Your task is to sample from a new coin with bias $f(p)$ for a given function $f : [0, 1] \to [0, 1]$, given access to as many samples as desired (hopefully not many) from the original coin. As some clarifying examples, for $f(p) = p^2$ one can flip the coin twice, and outputs a heads if both coin flips are heads, and tails otherwise. Another example is the Bernoulli factory for $f(p) = \mathbf{E}[p^X]$, when X is a random variable over $\mathbb{N}$. To do this, realize X first, then flip the coin X times, and finally output heads if all coin flips are heads, and output tails otherwise.

The question of existence of Bernoulli factories has been the main subject of interest in this literature. In particular, [Keane and O'Brien 1994] provide necessary and sufficient conditions for f under which a Bernoulli factory exists and [Nacu and Peres 2005] suggest an algorithm for simulating an $f(p)$-coin based on polynomial envelopes of f. The canonical challenging problem of Bernoulli factories – and a primitive in the construction of more general Bernoulli factories – is the *Bernoulli doubling* problem: $f(p) = 2p$ for $p \in (0, 1/2)$. See [Łatuszyński 2010] for a survey on this topic. The *multivariate* version of Bernoulli factory is also studied. One example is linearly *mixing* the coins [Huber 2015], e.g. sampling a coin with bias $f(p_1, p_2, p_3) = p_1 + p_2 + 0.5 \cdot p_3$ given p_i-coins for $i = 1, 2, 3$. In particular, by

the Bernoulli doubling result of [Nacu and Peres 2005], it is not hard to see if $p_1 + p_2 \in [0, 1-\delta]$ then summing two coins can be done with only $O(1/\delta)$ samples.

To apply the idea of Bernoulli factory to the problem of sampling cycle monotone allocations, we generalize Bernoulli factories to the *random selection from samples problems* in our paper [Dughmi et al. 2017]. In a random selection from samples problem, a set of elements is given. Moreover, a coin with unknown bias p_i is given for each element i. The objective is to sample an element from a distribution $\mathbf{f}(p_1, \ldots, p_m)$ over elements, given only access to the p_i-coins as sampling oracles.

Two interesting examples of random selection from samples problems are distributions $f_i = p_i / \sum_j p_j$, i.e. *linear weights*, and $f_i = e^{\lambda p_i} / \sum_j e^{\lambda p_j}$, i.e. *exponential weights*. The algorithm that solves the former is simple and natural: pick a coin at random and flip it, if the coin flips heads pick it, and otherwise repeat the process. We call this algorithm the *Bernoulli race*. For the latter problem, note that for X drawn from a Poisson distribution with parameter λ we have $\mathbf{E}\left[p^X\right] = e^{\lambda \cdot (p-1)}$. So, one solution would be pre-processing the coins using the Bernoulli factory for $e^{\lambda(p-1)}$ first (as described earlier), and then running a Bernoulli race for these exponential coins. We call this algorithm the *exponential Bernoulli race*.

3.5 Entropy regularization and fast exponential Bernoulli race

Back to the single-agent multiple-urns problem, remember that our goal was sampling from an allocation, i.e. a distribution $\mathbf{q} = (q_1, \ldots, q_m)$ over the urns, that is also a regularized welfare maximizer. Given that we know how to do an exponential Bernoulli race, one appealing idea would be to penalize the welfare with a negative entropy regularizer $(-1/\lambda) \cdot H(\mathbf{q}) = 1/\lambda \cdot \sum_i q_i \log(q_i)$, for $\lambda = \log(m)/\epsilon$. In this way, the optimal allocation $\mathbf{q}^*$ will have a loss bounded by ϵ as desired, because the entropy is bounded by $\log(m)$, and $q_i^* = e^{\lambda V_i(t)} / \sum_j e^{\lambda V_j(t)}$, which can be implemented by the exponential Bernoulli race given access to coins with biases $\{V_i(t)\}_{i \in [m]}$. Recall that we have sample access to distributions $\{D_i(t)\}_{i \in [m]}$ whose expectations are $\{V_i(t)\}_{i \in [m]}$, and we can convert those to Bernoulli distributions with the same expectations.[2]

There is only a caveat regarding the running time. It is not hard to see that the running time of exponential Bernoulli race is proportional to the inverse of sum of terms of the form $e^{\lambda(V_i - 1)}$. Now if all the expected values have a constant gap from 1, the running time will be exponential in λ. To make this race fast, a key observation is that the distribution $\mathbf{q}^*$ is invariant to a uniform additive shift to all of the V_i's. Therefore, here is a technical trick that can make this algorithm fast: by sampling from the coins and performing empirical estimation, simulate a *boosting* coin with bias L such that $\max_i\{V_i + L\} \in [1 - O(\delta), 1 - \delta]$ for $\delta = 1/\lambda$. Now, pre-process all the input coins by mixing them with the boosting coin, which guarantees there always exists a coin with boosted bias ≈ 1. This forces the race to terminate quickly, causing the fast Bernoulli race to solve the problem in time polynomial in $1/\epsilon$ and m.

[2]Given sample access to a distribution D supported on $[0, 1]$, we sample a Bernoulli distribution with the same expectation as follows: draw $X \sim D$, then output a draw from $\text{Bern}[X]$.

4. PUTTING THE PIECES TOGETHER

We can now describe how our paper [Dughmi et al. 2017] applied the solution of the single-agent multiple urns problem to sample a welfare preserving cycle monotone perfect matching. The first tempting idea to try is simple: what if we go over the replicas one by one, run the fast exponential Bernoulli race for the current replica over remaining unmatched surrogates to sample one surrogate, and then assign the replica to this surrogate? This perfect matching is truthful per-replica and is therefore a cycle monotone matching. However, this algorithm is *almost* the greedy algorithm (i.e. going over replicas one by one, and adding the maximum-weight unmatched edge incident to every replica), which is only a constant factor approximation to the maximum-weight perfect matching.

On the positive side, if we navigate the greedy algorithm using the *optimal dual* solution of the linear program for the maximum-weight perfect matching as a compass, greedy will find the optimal solution. This navigation can be done by shifting the value of edges incident to a replica by the optimal dual variables of the corresponding surrogate vertices and then running greedy. Yet, we do not have access to the optimal dual solution and we do not have access to an exact arg-max function. We need to calculate the optimal dual solution (in a truthful way) and we need to only use our efficient sampling oracle for the exponential weights (i.e., the fast exponential Bernoulli race), which is essentially a randomized smooth approximation to the arg-max function.

4.1 Entropy regularized perfect matching

The right way of combining the above primal-dual ideas is by considering a replica-surrogate matching mechanism that maximizes the welfare when penalized with the negative entropy regularizer. Let $V_{j,k} \triangleq \mathbf{E}\big[\mathrm{val}^i(r_j, \mathcal{A}^i(s_k))\big]$, let $x_{j,k}$ be the probability that replica j gets assigned to surrogate k, and let $\mathcal{P}_B$ be the perfect matching polytope. We then consider this convex program:

$$\begin{aligned} &\max_{\{x_{j,k}\}} \sum\nolimits_{j,k} x_{j,k}\, V_{j,k} - \tfrac{1}{\lambda} \sum\nolimits_{j,k} x_{j,k} \log x_{j,k}, \\ &\text{s.t.} \quad \mathbf{x} \in \mathcal{P}_B. \end{aligned}$$

It is not hard to see that the optimal solution to this program would have the form of exponential weights, i.e. $x^*_{j,k} = e^{\lambda\cdot(V_{j,k}-\alpha^*_k)} / \sum_{k'} e^{\lambda\cdot\left(V_{j,k'}-\alpha^*_{k'}\right)}$, where $\boldsymbol{\alpha}^*$ is the optimal Lagrangian dual vector of the above concave program. If we know this optimal dual solution up front, we can efficiently sample from distributions $\mathbf{x}^*_j$ for each replica independently. To do so, roughly speaking, we first mix each coin (j,k) with a coin $1-\alpha^*_k$, and we then sample an edge using the fast exponential Bernoulli race. This sampling procedure will output a cycle monotone perfect matching (or can be converted into a perfect matching by small technical tweaks in the case of conflicts at the surrogate side), and for large enough λ will satisfy ϵ-approximate welfare improvement per-replica. The main caveat is that we do not have access to $\boldsymbol{\alpha}^*$.

4.2 Many-to-one perfect matchings

The first technical ingredient we add in [Dughmi et al. 2017] to overcome the problem of not knowing $\boldsymbol{\alpha}^*$ is considering many-to-one perfect matchings instead. In this case, we have $B \cdot m$ replicas that need to be assigned to m surrogates, where each surrogate has capacity B. The ideal model reduction of [Hartline et al. 2011] still holds under this modification, because per-replica welfare can only increase for the maximum-weight perfect many-to-one matching. However, this will help us to *estimate* the optimal dual solution by sacrificing ϵ fraction of the replicas, and calculating the dual for the sub problem restricted to these replicas (with budgets scaled to $\epsilon \cdot B$). These replicas need to be sampled uniformly, so that we have an unbiased estimator.

Here is how we may want to implement this. We shuffle the replicas randomly, we look at them one by one in an online fashion (basically, we *simulate* an online problem), and consider the first ϵ fraction for dual estimation. It is pretty well-known in the online matching literature, e.g. [Agrawal et al. 2009; Devanur et al. 2011; Badanidiyuru et al. 2013], that this approach combined with the greedy algorithm will produce a near-optimal B-to-1 perfect matching for large enough B (a polynomial in $1/\epsilon$ and m). This is almost what we want, with the small missing piece that the fast exponential Bernoulli race cannot implement the greedy allocation exactly, as it samples from the exponential weights distribution rather than outputting the maximum weight edge.

4.3 Online stochastic convex optimization

The last technical ingredient we use in [Dughmi et al. 2017] to fill the missing gap in §4.2 is to incorporate primal-dual techniques that solve online stochastic convex optimization, e.g. in [Chen and Wang 2013; Agrawal and Devanur 2015], rather than the approach suggested in §4.2. The main idea is basically *learning* the optimal Lagrangian dual $\boldsymbol{\alpha}^*$ by applying a black-box no-regret online learning algorithm. In fact, the optimal dual is the minimizer of the convex Lagrangian dual function (which in our case is basically a linear function that is sum of the allocation slacks on the surrogate side), and this fact can be used to apply a no-regret learning algorithm for online convex optimization (or even online linear optimization in our case) to learn the dual.

We start from the all-zero vector for the dual, and once we process a replica i we update the dual variables by sending a query to the update rule of the no-regret learning algorithm. We then use this updated dual $\boldsymbol{\alpha}^{(i)}$ to pre-process the coins (i, j) for all surrogates j as described in §4.1. We can then use the fast exponential Bernoulli race over these shifted coins to sample the next edge. However, we still need to avoid violating the budgets on the surrogate side of the bipartite graph, so that the algorithm outputs a perfect B-to-1 matching. In [Dughmi et al. 2017], this is achieved by running the fast exponential Bernoulli race for each replica *only* over surrogates with non-zero remaining budget. This procedure also samples a cycle monotone matching. Moreover, by using standard techniques introduced in [Chen and Wang 2013; Agrawal and Devanur 2015], we can analyze this algorithm and show that for sufficiently large B, polynomial in m and $1/\epsilon$, the welfare loss is bounded by ϵ per-replica.

4.4 Final reduction

To just put the pieces together, in [Dughmi et al. 2017] we develop an online algorithm for the entropy regularized replica-surrogate matching problem under random order, and we then sample from this algorithm truthfully by the fast exponential Bernoulli race per-replica. The final reduction is polynomial time in m and $1/\epsilon$, and will sample a cycle monotone perfect B-to-1 matching whose per-replica welfare loss is bounded by ϵ. We then resample the agent's type by finding the edge in the sampled B-to-1 matching that contains the agent's true type (on the replica side) and outputting the surrogate type at the opposite endpoint of this edge. We repeat the same procedure for all agents to obtain a profile of surrogate types which is fed into the black-box allocation function to yield the mechanism's outcome. To bound the total welfare loss (summed over all agents) by ϵ, we scale the per-replica welfare loss from ϵ down to ϵ/n; this scaling inflates the running time by an additional poly(n) factor. Finally, we can compute payments that combine with this allocation procedure to yield a BIC mechanism using the implicit payment computation method of [Babaioff et al. 2015].

5. CONCLUSION

In this letter, we surveyed the line of work on black-box reductions in Bayesian mechanism design. We briefly explained how techniques and tools in the expectations from samples computational model are applied in [Dughmi et al. 2017] to achieve an exact BIC reduction with only a negligible additive loss in the expected social welfare. While the existence of such a reduction is good news for Bayesian mechanism design, there are limitations that are mostly unavoidable.

(1) *Beyond expected social welfare:* It is tempting to try converting an arbitrary algorithm for an optimization problem into a computionally efficient Basyesian truthful mechanism. Interestingly, this is not possible for all optimization objectives. In particular, Chawla et al. [2012] show that no black-box reduction is possible for the objective of makespan, even if we only require Bayesian truthfulness and an average-case performance guarantee. This precludes extending our result beyond the expected-welfare objective in a general fashion.

(2) *Exponential dependence on dimension:* Notably, the reduction in [Dughmi et al. 2017] is a fully polynomial time approximation scheme to the reduction of [Hartline et al. 2011]. However, the running time of [Hartline et al. 2011] has exponential dependence on Δ, where Δ roughly corresponds to the dimension of the type space. Therefore, our reduction in [Dughmi et al. 2017] also suffers from the same exponential dependence. Intuitively, this seems to be unavoidable for reductions that can only access the type space by sampling and can only access the outcome space by calling the allocation function on sampled type profiles.

We conclude with some open questions. The first natural question, directly related to the second limitation above, is to determine whether or not the exponential dependence on Δ in the black-box reduction is unavoidable. Are there black-box reductions whose running time exhibits a milder dependence on the structure of the type space? Another interesting question is to find more connections between Bayesian mechanism design and the expectations from samples computational model. Hopefully, these connections will lead to finding simpler or compu-

tationally more efficient reductions for specific environments, e.g. combinatorial auctions.

REFERENCES

Agrawal, S. and Devanur, N. R. 2015. Fast algorithms for online stochastic convex programming. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1405–1424.

Agrawal, S., Wang, Z., and Ye, Y. 2009. A dynamic near-optimal algorithm for online linear programming. *arXiv preprint arXiv:0911.2974*.

Ashlagi, I., Dobzinski, S., and Lavi, R. 2012. An optimal lower bound for anonymous scheduling mechanisms. *Math. Oper. Res. 37,* 2, 244–258.

Babaioff, M., Kleinberg, R., and Slivkins, A. 2015. Truthful mechanisms with implicit payment computation. *J. ACM 62,* 2 (May), 10:1–10:37.

Badanidiyuru, A., Kleinberg, R., and Slivkins, A. 2013. Bandits with knapsacks. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 207–216.

Bei, X. and Huang, Z. 2011. Bayesian incentive compatibility via fractional assignments. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 720–733.

Chawla, S., Immorlica, N., and Lucier, B. 2012. On the limits of black-box reductions in mechanism design. In *Proceedings of the 44th ACM Symposium on Theory of Computing*. ACM, 435–448.

Chen, X. A. and Wang, Z. 2013. A near-optimal dynamic learning algorithm for online matching problems with concave returns. *arXiv preprint arXiv:1307.5934*.

Devanur, N. R., Jain, K., Sivan, B., and Wilkens, C. A. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*. ACM, 29–38.

Dobzinski, S. and Vondrák, J. 2016. Impossibility results for truthful combinatorial auctions with submodular valuations. *J. ACM 63,* 1, 5:1–5:19.

Dughmi, S., Hartline, J. D., Kleinberg, R., and Niazadeh, R. 2017. Bernoulli factories and black-box reductions in mechanism design. In *Proc. 49th Annual ACM Symposium on Theory of Computing, (STOC 2017)*. 158–169.

Dughmi, S. and Vondrák, J. 2015. Limitations of randomized mechanisms for combinatorial auctions. *Games and Economic Behavior 92*, 370 – 400.

Gibbard, A. 1973. Manipulation of voting schemes: A general result. *Econometrica 41,* 4, 587–601.

Hartline, J. D., Kleinberg, R., and Malekian, A. 2011. Bayesian incentive compatibility via matchings. In *SODA*.

Hartline, J. D., Kleinberg, R., and Malekian, A. 2015. Bayesian incentive compatibility via matchings. *Games and Economic Behavior 92,* C, 401–429.

Hartline, J. D. and Lucier, B. 2010. Bayesian algorithmic mechanism design. In *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 301–310.

Hartline, J. D. and Lucier, B. 2015. Non-optimal mechanism design. *The American Economic Review 105,* 10, 3102–3124.

Huber, M. 2015. Optimal linear Bernoulli factories for small mean problems. *CoRR abs/1507.00843*.

Keane, M. and O'Brien, G. L. 1994. A Bernoulli factory. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 4,* 2, 213–219.

Łatuszyński, K. 2010. The Bernoulli factory, its extensions and applications. *Proceedings of IWAP 2010*, 1–5.

McAfee, R. P. and McMillan, J. 1988. Multidimensional incentive compatibility and mechanism design. *Journal of Economic Theory 46*, 335–354.

Nacu, Ş. and Peres, Y. 2005. Fast simulation of new coins from old. *The Annals of Applied Probability 15,* 1A, 93–115.

Nisan, N. and Ronen, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior 35,* 1–2, 166–196.

Papadimitriou, C. H., Schapira, M., and Singer, Y. 2008. On the hardness of being truthful. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, (FOCS 2008).* 250–259.

Rochet, J.-C. 1987. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics 16,* 2 (April), 191–200.

Satterthwaite, M. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Econ. Theory 10,* 2, 187–217.