Bernoulli Factories and Black-Box Reductions in Mechanism Design

Shaddin Dughmi[1], Jason Hartline[2], Bobby Kleinberg[3] and Rad Niazadeh[3]

[1]Department of Computer Science, University of Southern California,
[2]Department of Computer Science, Northwestern University,
[3]Department of Computer Science, Cornell University.

November 3, 2016

Abstract

We provide 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. The key technical barrier preventing exact incentive compatibility in prior black-box reductions is that repairing violations of incentive constraints requires understanding the distribution of the mechanism's output, which is typically #P-hard to compute. Reductions that instead estimate the output distribution by sampling inevitably suffer from sampling error, which typically precludes exact incentive compatibility.

We overcome this barrier by employing and generalizing the computational model in the literature on *Bernoulli Factories*. In a Bernoulli factory problem, one is given a function mapping the bias of an "input coin" to that of an "output coin", and the challenge is to efficiently simulate the output coin given only sample access to the input coin. Consider a generalization which we call the *expectations from samples* computational model, in which a problem instance is specified by a function mapping the expected values of a set of input distributions to a distribution over outcomes. The challenge is to give a polynomial time algorithm that exactly samples from the distribution over outcomes given only sample access to the input distributions.

In this model we give a polynomial time algorithm for the function given by *exponential weights*: expected values of the input distributions correspond to the weights of alternatives and we wish to select an alternative with probability proportional to its weight. This algorithm is the key ingredient in designing an incentive compatible mechanism for bipartite matching, which can be used to make the approximately incentive compatible reduction of Hartline et al. [2015] exactly incentive compatible.

1 Introduction

We resolve a five-year-old open question from Hartline et al. [2011, 2015]: *There is a polynomial time reduction from Bayesian incentive compatible mechanism design to Bayesian algorithm design for welfare maximization problems.*[1] The key distinction between our result and those of Hartline et al. [2011, 2015] is that both (a) the agents' preferences can be multi-dimensional and from a continuous space (rather than single-dimensional or from a discrete space), and (b) the resulting mechanism is exactly Bayesian incentive compatible (rather than approximately Bayesian incentive compatible).

A mechanism solicits preferences from agents, e.g., how much each agent prefers each outcome, and then chooses an outcome. *Incentive compatibility* of a mechanism requires that, though agents could misreport their preferences, it is not in any agent's best interest to do so. A quintessential research problem at the intersection of mechanism deign and approximation algorithms is to identify blackbox reductions from approximation mechanism design to approximation algorithm design. It is well known that incentive compatible mechanisms, from any individual agent's perspective, are *maximal-in-range*, specifically, the outcome selected maximizes the agent's utility less some cost that is a function of the outcome. The cost function can depend on other agents' reported preferences.

The blackbox reductions from Bayesian mechanism design to Bayesian algorithm design in the literature are based on obtaining an understanding of the distribution of outcomes produced by the algorithm through simulating the algorithm on samples from agents' preferences. Notice that, even for structurally simple problems, calculating the probability that a given outcome is selected by an algorithm can be #P-hard. For example, Hartline et al. [2015] show such a result for calculating the probability that a matching in a bipartite graph is optimal, for a simple explicitly given distribution of edge weights. A blackbox reduction for mechanism design must therefore produce exactly maximal-in-range outcomes merely from samples.

In traditional algorithm design, we think of inputs as being specified to the algorithm exactly. In this paper, we consider "expectations from samples" problems described by a function $f : \mathbb{R}^n \to \Delta_X$ where X is an abstract set of feasible solutions and Δ_X is the family of probability distributions over X. For any n input distributions with expectations $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$, an algorithm for such a problem, with only sample access to each of the n input distributions, must produce sample an outcome from X that is distributed exactly according to $f(\mu_1, \ldots, \mu_n)$.

Whereas sampling an object distributed approximately according to $f(\boldsymbol{\mu})$ is often amenable to Monte-Carlo sampling approaches, exact implementation of many natural functions f is either impossible for information theoretic reasons or requires sophisticated techniques. This is particularly the case if f is discontinuous, but often also when f is continuous. The literature on *Bernoulli Factories*, which inspired this work and provides some of the basic building blocks for our results, restricts attention to the special case where the input distribution and output distribution are both Bernoullis (i.e., supported on $\{0, 1\}$).

To illustrate this computational model, consider one of the basic building blocks of our main result: the *single-agent multiple-urns problem*. In this setting, a single agent faces a set X of urns, and each urn contains a random object whose distribution is unknown, but can be sampled. The agent's type determines his utility for each object; fixing this type, urn i is associated with a random real-valued reward with unknown expectation μ_i. Our goal is to allocate the agent his favorite urn, or close to it. If we could exactly calculate the expected values $\mu_1, \ldots, \mu_n$ from the agent's type, this problem is trivial both algorithmically and from a mechanism design perspective: simply solicit the agent's type t then allocate him the urn with the maximum $\mu_i = \mu_i(t)$. It is not too hard to see

[1] A Bayesian algorithm is one that performs well in expectation when the input is drawn from a known distribution. By polynomial-time, we mean polynomial in the number of agents and the combined "size" of their type spaces.

that this optimal urn selection rule — while maximal-in-range — is impossible to exactly implement if all we had was sample access to the random variables for the agent's utility for objects from each urn. This is unsurprising, since the distribution over urns chosen by this rule is discontinuous in the expectations of our random variables. In the expectations from samples model, discontinuous functions can not be exactly implemented using a finite number of samples.

Our solution to this problem is to design a different allocation rule f which is also maximal in range and approximately utility maximizing, but continuous as a function from the expected values $\boldsymbol{\mu}$ to the space of randomized outcome Δ_X. The rule we design is based on regularization; specifically, we let $f(\boldsymbol{\mu})$ be the distribution $\boldsymbol{q}$ over outcomes maximizing the agent's expected utility $\sum_i \mu_i q_i$ plus a small multiple of the Shannon entropy $H(\boldsymbol{q})$ of $\boldsymbol{q} = (q_1, \ldots, q_n)$. This problem admits a closed form using the KKT conditions: in the optimal solution, q_i is proportional to an exponential in μ_i.[2] We show how to select an urn in accordance with the distribution $\boldsymbol{q}$ using only a polynomial number of samples from the distributions of each urn. We refer to the algorithm that solves this problem as a *logistic Bernoulli race*.

The reduction from Bayesian mechanism design to Bayesian algorithm design of Hartline et al. [2011, 2015] is based on solving a matching problem[3] between multiple agents and outcomes, where an agent's value for an outcome is the expectation of a random variable which can be accessed only through sampling. Specifically, this generalizes the above-described single-agent multiple-urns problem to the problem of matching agents to urns with the goal of approximately maximizing the total weight of the matching (the social welfare). Again, for incentive compatibility we require this expectations from samples algorithm to be maximal in range from each agent's perspective. Using methods from Agrawal and Devanur's [2015] work on stochastic online convex optimization, we reduce this matching problem to the single-agent multiple-urns problem.

As stated in the opening paragraph, our main result – obtained through the approach outlined above – is a polynomial time reduction from Bayesian incentive compatible mechanism design to Bayesian algorithm design. The analysis assumes that agents' values are normalized to the $[0, 1]$ interval and gives additive loss in the welfare. The reduction is an approximation scheme and the dependence of the runtime on the additive loss is inverse polynomial. The reduction depends polynomially on a suitable notion of the size of the space of agent preferences. For example, applied to environments where agents have preferences that lie in high-dimensional spaces, the runtime of the reduction depends polynomially on the number of points necessary to approximately cover each agent's space of preferences. More generally, the bounds we obtain are polynomial in the bounds of Hartline et al. [2011, 2015] but the resulting mechanism, unlike in the proceeding work, is exactly Bayesian incentive compatible.

2 Preliminaries

We consider the general multi-parameter Bayesian mechanism design for strategic agents with independent private valuations. We first go over the setting and introduce the basic notations. Then we talk about Bayesian truthfulness and social welfare maximization problem.

[2] This is a standard relationship that has, for example, been employed in previous work in mechanism design [Huang and Kannan, 2012].

[3] Bei and Huang [2011] independently discovered a similar reduction based on solving a fractional assignment problem. Their reduction applies to finite, discrete type spaces and is approximately Bayesian incentive compatible.

2.1 Notations and Definitions

Multi-parameter Bayesian setting. Suppose there are n agents, where agent k has private *type* t^k from *type space* $\mathcal{T}^k$. The *type profile* of all agents is denoted by $\mathbf{t} = (t^1, \ldots, t^n) \in \mathcal{T}^1 \times \ldots \times \mathcal{T}^n$. Moreover, we assume types are drawn independently from known prior distributions. For agent k, let F^k be the distribution of $t^k \in \mathcal{T}^k$ and $\mathbf{F} = F^1 \times \ldots \times F^n$ be the joint distribution of types. Suppose there is an *outcome space* denoted by $\mathcal{O}$. Agent k with type t^k has valuation $v(t^k, o)$ for outcome $o \in \mathcal{O}$, where $v : (\mathcal{T}^1 \cup \ldots \cup \mathcal{T}^n) \times \mathcal{O} \to [0, 1]$ is a fixed function. Note that we assume agent values are non-negative and bounded, and w.l.o.g in $[0, 1]$. Finally, we allow charging agents with non-negative money *payments* and we assume agents are *quasi-linear*, i.e. an agent with private type t has *utility* $u = v(t, o) - p$ for the outcome-payment pair (o, p).

Algorithms, mechanisms and interim rules. An *allocation algorithm* $\mathcal{A}$ is a mapping from type profiles $\mathbf{t}$ to outcome space $\mathcal{O}$. A (direct revelation) mechanism $\mathcal{M}$ is a pair of *allocation rule* and *payment rule* $(\mathcal{A}, \mathbf{p})$, in which $\mathcal{A}$ is an allocation algorithm and $\mathbf{p} = (p^1, \ldots, p^n)$ where each p^k (denoted by the payment rule for agent k) is a mapping from type profiles $\mathbf{t}$ to $\mathbb{R}_+$. In fact, one can think of the interaction between strategic agents and a mechanism as following: agents submit their *reported types* $\mathbf{s} = (s^1, \ldots, s^n)$ and then the mechanism $\mathcal{M}$ picks the outcome $o = \mathcal{A}(\mathbf{s})$ and charges each agent k with its payment $p^k(\mathbf{s})$. We also consider *interim allocation rule*, which is the allocation from the perspective of one agent when the other agent's types are drawn from their prior distribution. More concretely, we abuse notation and define $\mathcal{A}^k(s_k) \triangleq \mathcal{A}(s^k, \mathbf{t}^{-k})$ to be the distribution over outcomes induced by $\mathcal{A}$ when agent k's type is s^k and other agent types are drawn from $\mathbf{F}^{-k}$. Similarly, for agent k we define *interim payment rule* $p^k(s^k) \triangleq \mathbb{E}_{\mathbf{t}^{-k} \sim \mathbf{F}^{-k}}\{p^k(s^k, \mathbf{t}^{-k})\}$, and *interim value* $v^k(s^k) \triangleq \mathbb{E}_{\mathbf{t}^{-k} \sim \mathbf{F}^{-k}}\{v(s^k, \mathcal{A}^k(s^k, \mathbf{t}^{-k}))\}$. In most parts of this paper, we focus only on one agent, e.g. agent k, and we just work with the interim allocation algorithm $\mathcal{A}^k(.)$. When it is clear from the context, we drop the agent's superscript, and therefore $\mathcal{A}(s)$ denotes the distribution over outcomes induced by $\mathcal{A}(s, \mathbf{t}^{-k})$ when $\mathbf{t}^{-k} \sim \mathbf{F}^{-k}$.

We also use standard incentive compatibility notions, i.e. BIC and DSIC. In this paper, we focus on the expected welfare objective. For more details, see Appendix B.

2.2 Bernoulli Factory and Extensions

In this work, we introduce the applications of the Bernoulli factory machinery for black-box reductions in mechanism design. In this section, we review this toolbox and some of its extensions which will be used in later parts of this paper. We interchangeably use $\{1, 0\}$ or $\{\texttt{heads}, \texttt{tails}\}$ to refer to the state of a coin after flipping ($\texttt{heads}$ = "1".)

Basics. The Bernoulli factory problem is about generating new coins from old ones. Formally speaking, it can be defined as the following problem.

Definition 2.1 (Keane and O'Brien [1994])**.** Suppose $f(.) : (0, 1) \to (0, 1)$ is given. The *Bernoulli factory problem* is to output a single sample of a Bernoulli variable with bias $f(p)$ (i.e. $f(p)$-coin), given black-box access to independent samples of a Bernoulli variable with bias p (i.e. p-coin), where p is constrained to lie in a subset $\mathcal{S}$ of $[0, 1]$ but unknown otherwise.

Example To have more intuition on Bernoulli factories, consider the examples where $f(p) = p^2$ and $f(p) = e^{p-1}$. For the former one, it is enough to flip the p-coin twice and output 1 if both flips where 1, and 0 otherwise. For the latter one, the Bernoulli factory is still simple but more

interesting: draw $k \sim$ Poission(1), flip the p-coin k times and output 1 if all coin flips where 1, and 0 otherwise (See Corollary 2.3).

Not surprisingly, the question of chracterizing functions f for which there is an algorithm simulating $f(p)$-coins from p-coins has been the main subject of interest in this literature Keane and O'Brien [1994], Nacu et al. [2005]. In particular, Keane and O'Brien [1994] provides necessary and sufficient conditions for f, under which an algorithm for the Bernoulli factory exists. Moreover, Nacu et al. [2005] suggests an algorithm for simulating an $f(p)$-coin based on polynomial envelopes of f. For a survey on this topic, we suggest the interested reader to look into Łatuszyński [2010].

Multivariate Bernoulli factory. Consider the following generalization of the basic Bernoulli factory problem to multiple coins. Suppose $f : (0,1)^m \to (0,1)$ is a multivariate function. The goal is to simulate a Bernoulli variable with bias $f(p_1, p_2, \ldots, p_m)$ given black-box access to m independent Bernoulli variables with biases $\mathbf{p} = [p_1, \ldots, p_m]$, where $\mathbf{p} \in \mathcal{S} \subseteq [0,1]^m$ but otherwise unknown. On multivariate Bernoulli factory problem much less is known and it is almost merely studied for special cases. One prominent special case is the *linear multivariate Bernoulli factory*, in which $f(p_1, \ldots, p_m) = \sum_{i \in [m]} C_i p_i$. In particular, Huber [2015] showed the following interesting result on how to design fast algorithms for this problem (in terms of runtime and sample complexity).

Theorem 2.1. *Given sample access to Bernoulli random variables $B_1, \ldots, B_m$ with $B_i \sim Bern(p_i)$ for uknown p_i, and constants $C_1, \ldots, C_m$ with the guarantee that $\sum_i C_i p_i$ is at most $1 - \delta$, there exists a multivariate Bernoulli factory that generates $Z \sim Bern(\sum_i C_i p_i)$. Moreover, the factory runs in time $O(\frac{\sum_i C_i}{\delta})$.*

BF based building blocks. We design some useful building blocks (See Appendix A for proofs).

Lemma 2.2. ***(BF for Probability Generating Functions)*** *Given a p-coin and a distribution $\mathcal{D}$ over $\mathbb{Z}$ with moment generating function $f(x) = \mathbb{E}_{k \sim \mathcal{D}}\{x^k\}$, the following procedure samples a $f(p)$-coin with $\mathbb{E}_{k \sim \mathcal{D}}\{k\}$ coin flips in expectation:*

- *Draw $k \sim \mathcal{D}$ and flip the coin k times.*
- *Output 1 if all the coin flips are 1, and output 0 otherwise.*

Corollary 2.3. *Given a coin with bias p and a constant λ, we use Lemma 2.2 with $\mathcal{D} = Poisson(\lambda)$ to sample a coin with bias $f(p) = \exp(\lambda(p-1))$ with only λ coin flips in expectation.*

Lemma 2.4. *We have the following simple building blocks.*

1. ***(Continuous-to-Coin)*** *Given access to samples of distribution $\mathcal{D}$ over $[0,1]$, the following simple procedure outputs a coin B with bias $\mathbb{E}_{X \sim \mathcal{D}}\{X\}$.*
   - *Draw $v \sim \mathcal{D}$ and $u \sim unif\,[0,1]$.*
   - *If $u \leq v$ then $B = 1$, otherwise $B = 0$.*
2. ***(Scaling Down a Coin)*** *Given access to a p-coin and a constant $\lambda \in [0,1]$, the following simple procedure outputs a coin B with bias $p \cdot \lambda$:*
   - *Draw $u \sim unif\,[0,1]$ and flip the coin.*
   - *If $u \leq \lambda$ and coin flip$= 1$ then $B = 1$, otherwise $B = 0$.*

3 Bernoulli Racing

In this section, we define and solve two related problems which can be thought of as *combinatorial Bernoulli factory* problems. In the *Bernoulli race* problem, one is given sample access to a collection of coins, and must select one of them such that each coin is selected with probability proportional to its bias. In the *logistic Bernoulli race* problem, each coin must be selected with probability proportional to an exponential in its bias. The logistic Bernoulli race is a key building block of our mechanism design reduction, and we use our Bernoulli race as a subroutine in its solution. The runtime of all our algorithms in this section will be linearly related to their stated sample complexity.

3.1 Bernoulli Race

Definition 3.1 (**Random Selection with Linear Weights**)**.** Given sample access to Bernoulli random variables with unknown biases $\mathbf{v} = [v_1, \ldots, v_m]$, output a distribution over $[m]$ with marginals $\mathbf{q}$, where

$$\forall i \in [m]: \quad q_i = \frac{v_i}{\sum_{j \in [m]} v_j}$$

We propose Algorithm 1 for the Bernoulli race. The algorithm repeatedly picks a coin uniformly at random and flips it. The winning coin is the first one to come up heads in this process.

Algorithm 1 Bernoulli Race

1: **input** sample access to m coins with biases $v_1, \ldots, v_m$.
2: **initialize** success = `FALSE`.
3: **repeat**
4: Pick coin $i^* \in [1:m]$ uniformly at random and flip it.
5: If the coin comes up heads, then output the coin i^* and let success $\leftarrow$ `TRUE`.
6: **until** success = `TRUE`

Theorem 3.1. *Algorithm 1 uses $\frac{m}{\sum_{i \in [m]} v_i}$ sample queries in expectation before termination, and it outputs the linear weights, a.k.a Bernoulli race distribution, as in Definition 3.1.*

Proof At each iteration, the algorithm terminates if the flipped coin outputs 1 and iterates otherwise. Since the coin is chosen uniformly at random, the probability of termination at each iteration is $\frac{1}{m} \sum_{i \in [m]} v_i$. The total number of iterations (and number of samples) is therefore a geometric random variable with expectation $\frac{m}{\sum_{i \in [m]} v_i}$.

The winning coin also follows the desired distribution, as shown below.

$$\begin{aligned}\Pr\{\text{winner} = i\} &= \sum_{k=1}^{\infty} \Pr\{(\text{winner} = i) \;\&\; \text{Algorithm 1 does not terminate before interation } k\} \\ &= \frac{v_i}{m} \sum_{k=1}^{\infty} \left(1 - \frac{1}{m} \sum_{j \in [m]} v_j\right)^{k-1} = \frac{\frac{v_i}{m}}{\frac{1}{m} \sum_{j \in [m]} v_j} = \frac{v_i}{\sum_{j \in [m]} v_j}.\end{aligned}$$

□

3.2 The Logistic Bernoulli Race

Definition 3.2 (**Random Selection with Exponential Weights**)**.** Given a parameter $\lambda > 0$ and sample access to Bernoulli random variables with unknown biases $\mathbf{v} = [v_1, \ldots, v_m]$, output a distribution over $[m]$ with marginals $\mathbf{q}$, where

$$\forall i \in [m]: \quad q_i = \frac{e^{\lambda v_i}}{\sum_{j \in [m]} e^{\lambda v_j}} \tag{1}$$

We start by presenting a simple implementation of the logistic Bernoulli race in Algorithm 2 — albeit one which is inefficient as far as our intended application is concerned.

Algorithm 2 Non-Polynomial-Time Logistic Bernoulli Race

1: **input** Constant $\lambda > 0$.
2: **input** Sample access to m coins with biases $v_1, \ldots, v_m$.
3: $\forall i \in [m]$, use the Bernoulli factory from Corollary 2.3 to implement a new coin with bias $\tilde{v}_i = e^{\lambda(v_i - 1)}$.
4: Run Algorithm 1 with the new coins $\text{Bern}(\tilde{v}_i)$.

Theorem 3.2. *Algorithm 2 outputs the exponential weights, a.k.a. the logistic Bernoulli race distribution, as in Definition 3.2. Also it uses at most $\lambda m e^{\lambda(1-v_{max})}$ sample queries in expectation where $v_{max} := \max_i v_i$.*

Proof Correctness follows immediately from Corollary 2.3 and Theorem 3.1. Runtime follows from Corollary 2.3, Theorem 3.1, Wald's identity, and simple manipulation. □

Observe that the runtime of Algorithm 2 is exponential in λ if $v_{\max}$ is bounded away from 1. This is inadequate for our main application, where λ will be an approximation parameter which grows polynomially with the number of agents in the exact BIC reduction problem (Section 5). The following fact comes to the rescue: the logistic Bernoulli race distribution is invariant to applying a uniform additive shift to all the coin biases! Next up, we will show how to do this efficiently so that the maximum bias is roughly on the order of $1 - \frac{O(1)}{\lambda}$, and in doing so guarantee runtime polynomial in λ and m for our logistic Bernoulli race.

3.3 Speeding up the Logistic Bernoulli Race

We start with a simple observation: For any given parameter ϵ, we can easily implement a Bernoulli random variable Z whose bias z is within an additive ϵ of $v_{\max}$. We note that — unlike our other results in this paper — we don't require or state a precise relationship between z and $v_1, \ldots, v_m$.

Lemma 3.3. *Given sample access to coins with biases $v_1, \ldots, v_m$, and a parameter $\epsilon > 0$, there is an algorithm which runs in time $O(\frac{m}{\epsilon^2} \cdot \log(\frac{m}{\epsilon}))$ and outputs a sample from a Bernoulli random variable $Z \sim Bern(z)$ satisfying $z \in [v_{max} - \epsilon, v_{max} + \epsilon]$ where $v_{max} = \max_i v_i$.*

Proof The algorithm is as follows: Sample $\frac{4}{\epsilon^2} \log(\frac{4m}{\epsilon})$ times from each of the m coins, let $\hat{v}_i$ be the empirical estimate of coin i's bias obtained by averaging, then flip a coin with bias $\hat{v}_{\max} := \max_i \hat{v}_i$ using the continuous-to-coin transformation in Lemma 2.4.

Standard tail bounds imply that $|\hat{v}_{\max} - v_{\max}| < \epsilon/2$ with probability at least $1 - \epsilon/2$, and therefore $z = \mathbb{E}[\hat{v}_{\max}] \in [v_{\max} - \epsilon, v_{\max} + \epsilon]$. □

Since we are interested in a fast logistic Bernoulli race as λ grows large, we restrict attention to $\lambda > 4$. We set $\epsilon = 1/\lambda$, and seek to use Z and the multivariate Bernoulli factory of Theorem 2.1 to boost the bias of each each coin so that the maximum bias is $1 - O(\epsilon)$. However, we use multiplicative scaling to ensure that no coin's bias gets too close to 1, as this would lead to a blowup in the runtime of the multivariate Bernoulli factory. We present the details in Algorithm 3.

Algorithm 3 Polynomial-Time Logistic Bernoulli Race

1: **input** Constant $\lambda > 4$.
2: **input** Sample access to m coins with biases $v_1, \ldots, v_m$.
3: Let $\epsilon = 1/\lambda$.
4: Use Lemma 3.3 to implement a coin with bias $z \in [v_{\max} - \epsilon, v_{\max} + \epsilon]$.
5: Flip Z and use coin scaling (Lemma 2.4) to implement a coin with bias $(1 - 2\epsilon)(1 - z)$.
6: $\forall i \in [m]$, use coin scaling (Lemma 2.4) to implement a coin with bias $(1 - 2\epsilon)v_i$.
7: $\forall i \in [m]$, use the Multivariate Bernoulli Factory from Theorem 2.1 to generate coin $B'_i \sim \text{Bern}(v'_i)$ for $v'_i = (1 - 2\epsilon)v_i + (1 - 2\epsilon)(1 - z)$.
8: Run Algorithm 2 with with parameter $\lambda' = \frac{\lambda}{1-2\epsilon}$ and new coins $B'_1, \ldots B'_m$.

Theorem 3.4. *Algorithm 3 outputs the logistic Bernoulli race distribution as in Definition 3.2, and uses at most $O(\lambda^4 m^2 \log(\lambda m))$ sample queries in expectation.*

Proof Correctness follows from Lemmas 3.3 and 2.4 and Theorems 2.1 and 3.4, in particular since $\lambda' v'_i = \lambda v_i + (1 - z)$, where the additive shift $1 - z$ is is independent of i.

Since the algorithm builds a number of sampling subroutines in a hierarchy, we analyze the runtime of the algorithm and the various subroutines in a bottom up fashion. Steps 4 and 5 implement a coin with bias $(1 - 2\epsilon)(1 - z)$ with runtime $O(\lambda^2 m \cdot \log(\lambda m))$ per sample, as per the bound of Lemma 3.3. The coin implemented in Step 6 is sampled in constant time. Observe that $v'_i \leq (1-2\epsilon)(1+v_i-v_{\max}+\epsilon) \leq 1-\epsilon$, and therefore Theorem 2.1 implies that $O(\lambda)$ samples from the coins of Steps 5 and 6 suffice for sampling B'_i; using Wald's identity, we conclude that the coin B'_i can be sampled in time $O(\lambda^3 m \cdot \log(\lambda m))$. Finally, note that for the i^* such that $v_{i^*} = v_{\max}$ we have $v'_{i^*} \geq 1 - 3\epsilon$; Theorem 3.2 then implies that Step 8 samples at most $\lambda' m e^{3\lambda'/\epsilon} \leq 2e^6 \lambda m = O(\lambda m)$ times from the new coins B'_i; using Wald's identity, we conclude the claimed runtime. □

4 The Single-Agent Multiple-Urns Problem

In this section, we investigate incentive compatible mechanism design in the *single-agent multiple-urns* setting, formally defined below. This provides us with the main ingredient of our black-box reduction in Section 5.

4.1 Problem Definition

Consider the following basic mechanism design setting with only one agent. There are m urns, and urn i is associated with a probability distribution $\mathcal{D}_i$ over outcomes in $\mathcal{O}$. The agent has types in $\mathcal{T}$, and an agent with type $t \in \mathcal{T}$ has value $v(t, O) \in [0, 1]$ for an outcome $O \in \mathcal{O}$. The agent must report a type t to the mechanism, which then assigns her an urn i^* and charges her some nonnegative payment p. The realized outcome is then a fresh draw $o \sim \mathcal{D}_{i^*}$ from the chosen urn. The agent's utility is her value for the outcome o less her payment p. The goal is the usual one: approximately maximize the expected social welfare (i.e. the agent's expected value for the realized outcome) subject to (exact) incentive compatibility — i.e., the agent maximizes her expected utility

by reporting her true type. Here, expectation is over the random realization of the outcome and any internal randomness of the mechanism.

If the mechanism designer knows the distributions $\mathcal{D}_i$ for all urns i, this problem admits a trivial optimal mechanism: simply select the urn maximizing the agent's expected value $v_i(t) \triangleq \mathbb{E}_{O\sim\mathcal{D}_i}\{v(t,O)\}$ according to her reported type t, and charge her a payment of zero. We consider the more involved scenario in which the mechanism can only sample from the urns. In this case, the following Monte-carlo adaptation of the trivial mechanism is tempting: sample from each urn sufficiently many times to obtain a close estimate $\tilde{v}_i(t)$ of $v_i(t)$ with high probability (up to any desired precision $\delta > 0$), then choose the urn i maximizing $\tilde{v}_i(t)$ and charge a payment of zero. This mechanism is not incentive compatible, as illustrated by a simple example.

Example Consider two urns. Urn A contains only outcome o_1, whereas B two contains a mixture of outcomes o_0 and o_2, with o_0 slightly more likely than o_2. Now consider an agent who has (true) values 0, 1, and 2 for outcomes o_1, o_2, and o_3 respectively. If this agent reports her true type, the trivial Monte-carlo mechanism — instantiated with any desired finite degree of precision — assigns her urn A most of the time, but assigns her urn B with some nonzero probability. The agent gains by misreporting her value of outcome o_2 as 0, since this guarantees her preferred urn A.

The above example might seem counter-intuitive, since the trivial Monte-carlo mechanism appears to be doing its best to maximize the agent's utility, up to the limits of (unavoidable) sampling error. One intuitive rationalization is the following: an agent can slightly gain by procuring (by whatever means) more precise information about the distributions $\mathcal{D}_i$ than that available to the mechanism, and using this information to guide her strategic misreporting of her type. This raises the following question, which we will resolve in the affirmative in this section.

Question: *Is there an incentive-compatible mechanism for the single-agent multiple-urns problem which achieves welfare within ϵ of the optimal, and samples only poly$(m, \frac{1}{\epsilon})$ times (in expectation) from the urns?*

The constraints of exact incentive compatiblity require a precise relationship between the (unknown) profile of urn distributions and the distribution of outputs of the mechanism. This relationship must hold despite the fact that the mechanism has only sample access to the profile of urn distributions. Achieving such a design goal is precisely the domain of Bernoulli factories and our Bernoulli races!

Additional Notation. As defined previously, $v_i(t)$ is expected value of an agent with type t for an outcome $o \sim \mathcal{D}_i$. We think of a mechanism as a pair of functions $(\mathbf{q}, p)$, where $\mathbf{q}(t) \in \Delta_m$ is the distribution over urns chosen by the (randomized) mechanism given reported type t, and $p(t) \geq 0$ is the associated expected payment. Abusing notation, we sometimes write $\mathbf{q}(\mathbf{v})$ in place of $\mathbf{q}(t)$, where $\mathbf{v} = [v_1(t), \ldots, v_m(t)]$ is the agent's value profile.

4.2 Entropy Regularization and Bernoulli Racing

It is well known that one way to design an incentive compatible mechanism is through the design of an *affine maximizing* allocation rule (see e.g. Nisan et al. [2007]). In the special case of our single-agent multiple-urns problem, this is a rule which selects a randomized allocation $\mathbf{q} = \mathbf{q}(t) \in \Delta_m$ maximizing $\mathbf{q} \cdot \mathbf{v}(t) + \beta(\mathbf{q})$, where $\mathbf{q} \cdot \mathbf{v}(t)$ is the agent's expected value for the allocation, and $\beta : \Delta_m \to \mathbb{R}$ assigns a type-independent weight to each randomized allocation $\mathbf{q} \in \Delta_m$. There

exists a payment scheme which converts such a rule to an incentive compatible and individually rational mechanism.

Our approach is to design an affine maximizer which achieves expected welfare within ϵ of the optimum, and can be simulated *exactly* through sampling the urns using the Bernoulli race and Bernoulli factories. A report of $t \in \mathcal{T}$ should result in a choice of urn distributed as $\mathbf{q}(t)$ in aggregate over the internal random coins of the mechanism and the samples taken from the urns, even though the vector $\mathbf{q}(t)$ is never computed explicitly. Our affine maximizer uses the *entropy regularlization* function as the choice of β; specifically, our allocation rule solves the following convex program:

$$\underset{\mathbf{q}\in\Delta_m}{\operatorname{argmax}} \quad \mathbf{q}.\mathbf{v} + \frac{\epsilon}{\log(m)} H(\mathbf{q}), \tag{2}$$

where $H(q) = -\sum_i q_i \log q_i$ is the Shannon entropy function.

Our allocation rule achieves an additive ϵ-approximation of the optimal welfare, as can be seen by observing that the output of H is bounded by $\log(m)$ everywhere. Moreover, using the KKT conditions it is easy to show the following closed form expression for our allocation rule:

$$\forall i \in [m]: \quad q_i(\mathbf{v}) = \frac{e^{\frac{v_i \log m}{\epsilon}}}{\sum_{j\in[m]} e^{\frac{v_j \log m}{\epsilon}}} \tag{3}$$

This closed form is is a familiar one: the urn can be chosen by running a logistic Bernoulli race with parameter $\lambda = \frac{\log m}{\epsilon}$ on the coins $v_1, \ldots, v_m$. Theorem 3.4, combined with an implicit payment computation procedure as described in Section 2, implies the following result.

Theorem 4.1. *There is an incentive-compatible and individually-rational mechanism for the single-agent multiple-urns problem which achieves an additive ϵ-approximation to the optimal welfare in expectation, and runs in time $O(m^2(\frac{\log m}{\epsilon})^5)$ in expectation.*

5 Exact BIC Black-Box Reduction

We now investigate the *Bayesian BIC black-box reduction problem*, in which given black-box access to an allocation algorithm $\mathcal{A}$, one wants to simulate a BIC allocation algorithm $\tilde{\mathcal{A}}$ that approximately preserves welfare[4], i.e. $\text{val}(\tilde{\mathcal{A}}) \geq \text{val}(\mathcal{A}) - \epsilon$, and runs in time $\text{poly}(n, \frac{1}{\epsilon})$ given the oracle $\mathcal{A}$. After giving some preliminaries, we describe our polynomial black-box reduction for the general multi-parameter Bayesian mechanism design setting with general type spaces. We explain how the solution to the single-agent multiple-urns problem will help us to implement a BIC allocation algorithm $\tilde{\mathcal{A}}$ for the n-agent multi-parameter mechanism design problem, by only polynomial in n and ϵ^{-1} many calls to $\mathcal{A}$, such that $\text{val}(\tilde{\mathcal{A}}) \geq \text{val}(\mathcal{A}) - \epsilon$.

5.1 Preliminaries: The Black-Box Reduction Problem

We summarize the approach and related results in Hartline et al. [2015] and Hartline and Lucier [2010] for this problem, which provides the foundation of our result.

[4]One could also consider approximately preserving objectives other than welfare. However, Chawla et al. [2012] have shown that BIC black-box reductions for the makespan objective cannot be computationally efficient in general.

The ideal model vs. the black-box model. Under the *ideal model* (Hartline et al. [2015]), we have access to an oracle that for any two types $t^k, s^k \in \mathcal{T}^k$ outputs $\mathbb{E}_{\mathbf{t}^{-k}\sim\mathbf{F}^{-k}}\{v(t^k, \mathcal{A}s^k, \mathbf{t}^{-k}\}$. However, under the *black-box model* (Hartline et al. [2015]), we only have access to an oracle that for any type profile $\mathbf{t} \in \prod_{k\in[n]} \mathcal{T}^k$ outputs the distribution over outcomes $\mathcal{A}(\mathbf{t})$.

The surrogate selection rule. The heart of the reduction in Hartline and Lucier [2010], Hartline et al. [2015] is the *surrogate selection rule* Γ that for each agent maps the agent's reported type in $\mathcal{T}$ to a randomized type in $\mathcal{T}$ (which is called a *surrogate*). Using this mapping, they define a generic reduction, called Γ-*reduction*, that applies the mapping Γ independently to each agent's reported type to get a surrogate profile and then calls allocation algorithm $\mathcal{A}$ on the surrogate profile to return an outcome. To make this reduction work, as explained in [Hartline et al., 2015], the mapping Γ should satisfy *distribution preservation property*, i.e. for any agent k, $\Gamma(t_k) \sim F^k$ if $t_k \sim F^k$. From the perspective of an agent k, this ensures that the type distribution of other agents will remain the same after the mapping takes place. With this property, the black-box reduction decomposes across agents and boils down to showing that the composition of the interim allocation algorithm $\mathcal{A}^k(.) \triangleq \mathcal{A}(., \mathbf{t}^{-k})$ with the surrogate selection rule Γ is BIC and has a welfare-loss bounded by ϵ/n compared to $\mathbb{E}_{t\sim F^k}\{\mathcal{A}^k(t)\}$.

The Replica-Surrogate Matching (RSM). The choice of surrogate selection rule in Hartline et al. [2015] for the ideal model is based on the *replica-surrogate matching*. In this paper, we use a simple generalization of this matching problem to the k-to-1 budgeted matching, formally defined as following for a given agent with reported type t and $m, k \in \mathbb{N}$.

Definition 5.1. *The k-to-1 replica-surrogate matching* refers to the maximum-weighted k-to-1 (fractional or integral) matching problem in an instance generated as follows:

1. Pick i^* uniformly at random in $[mk]$, which is referred as *real agent replica*.
2. Sample the *surrogate type profile* $\mathbf{s}$, an m-tuple of i.i.d samples from the type distribution F and define the *surrogate outcome profile* $\mathbf{O} = (O_1, \ldots, O_m)$, where $O_j = \mathcal{A}(s_j)$ is a distribution over outcomes $\mathcal{O}$. We call the set of these surrogate as "*right-side vertices*".
3. Define the *replica type profile* $\mathbf{r}$, an mk-tuple of types by letting $r_{i^*} = t$, the real agent's reported type, and sampling the remaining replica types $\mathbf{r}_{-i^*}$ i.i.d. from the type distribution F. We call the set of these replicas as "*left-side vertices*" or "*replica-agents*".
4. For $(i, j) \in [mk] \times [m]$, consider the random variable $v(r_i, o_j)$ where $o_j \sim O_j$. Let $v_{i,j} \triangleq \mathbb{E}_{o_j\sim O_j}\{v(r_i, o_j)\}$. Now define a weighted bipartite graph between replicas (left-side vertices) and surrogate outcomes (right-side vertices) with the weight of the edge (i, j) equal to $v_{i,j}$.
5. Let each surrogate outcome $j \in [m]$ (right-side vertex) to have a budget equal to k.

RSM Black-box reduction in ideal model or discrete types. Suppose $X(\mathbf{r}, \mathbf{s})$ be a a feasible k-to-1 matching for the instance in Definition 5.1. Let Γ^X be the surrogate selection rule that for each agent's reported type t computes the randomized k-to-1 matching $X(\mathbf{r}, \mathbf{s})$, and assigns it to the surrogate that real agent replica i^* gets matched to. Now, similar to [Hartline et al., 2015], it is not hard to see this selection is distribution preserving as long as $X(\mathbf{r}, \mathbf{s})$ is a perfect matching (see Lemma A.1). Moreover, consider the mechanism design setting that given $\mathbf{s}$ we elicit replica agent's types $\mathbf{r}$ first and then assign to each of them a surrogate outcome. We can think of $X(\mathbf{r}, \mathbf{s})$ as an allocation algorithm for this problem. It is not again hard to see that if X forms a DSIC allocation

algorithm, then the composition of Γ^X and interim allocation algorithm $\mathcal{A}(.)$ forms a BIC allocation algorithm (see Lemma A.2).

Finally, as shown in Hartline et al. [2015], by calculating the maximum-weighted matching (for $k = 1$) given the weights of edges in the ideal model we get a BIC reduction whose welfare-loss is bounded by ϵ/n for each agent and requires only $\text{poly}(n, \frac{1}{\epsilon})$ calls to the ideal model allocation oracle. Under the black-box model, they also provide a similar, yet more complicated, reduction to show how to get a polynomial BIC reduction when type spaces are discrete. However, this result does not hold for general type spaces and in fact they can only show a *non-exact* ϵ-BIC polynomial reduction by estimating the expected value over edges through sampling. The question of having an exact polynomial BIC reduction remained open since then, which we settle in this paper.

Implicit payment computation. Given a BIC allocation algorithm $\tilde{\mathcal{A}}$, there are standard techniques to compute payments $\tilde{\mathbf{p}}$ such that $\tilde{\mathcal{M}} = (\tilde{\mathcal{A}}, \tilde{\mathbf{p}})$ is a BIC mechanism. These methods, called *implicit payment computations*, are randomized reductions that call $\tilde{\mathcal{A}}$ a limited number of times in black-box manner and perform some postprocessing to compute unbiased estimates of the truthful payments $\tilde{\mathbf{p}}$ which combine with $\tilde{\mathcal{A}}$ to form a BIC mechanism. Appendix C provides more details.

5.2 Replica-Surrogate Logistic Matching (RSLM)

Similar to Hartline et al. [2015, 2011] and Hartline and Lucier [2010], the base of our reduction is a surrogate selection rule Γ, as explained in Section 5.1. In particular, we design our reduction using a k-to-1 replica-surrogate matching algorithm as introduced in Definition 5.1, such that it satisfies:

1. *Dominant strategy incentive compatible.* We give a randomized allocation algorithm for the mechanism design problem of matching replica agents to surrogate outcomes, given replica types $\mathbf{r}$ and surrogate types $\mathbf{s}$, that is DSIC in expectation.

2. *Polynomial sample complexity.* We show how to implement this randomized algorithm by only polynomial number of sample queries to the interim allocation black-box $\mathcal{A}(.)$.

3. *Small welfare loss.* We show how our allocation algorithm does not lose more than ϵ/n welfare compared to the interim allocation algorithm $\mathcal{A}$.

Notation remark. In the remaining parts of this section, as we are only focusing on one agent i when the types of the other agents are drawn form the joint type distribution, we abuse the notation and use $\mathcal{A}(t)$ to denote the distribution over outcomes induced by the allocation algorithm $\mathcal{A}$ when the agent reports t and others report $\mathbf{t}^{-i} \sim \mathbf{F}{-i}$.

We start by considering a convex program, similar to the maximum-weighted k-to-1 replica-surrogate matching program, and show how this helps us to have a DISC allocation rule $X(\mathbf{r}, \mathbf{s})$ for the replica-surrogate k-to-1 matching problem that is implemetable with polynomial calls to the interim allocation black-box $\mathcal{A}$.

Definition 5.2. The *Replica-Surrogate Logistic Matching (RSLM)* convex program, parametrized by $\delta > 0$, is defined by the following mathematical program, in which $\mathbf{v}_i$ denotes the vector $(v_{i,1}, \ldots, v_{i,m})$:

$$\begin{aligned} \underset{\{\mathbf{x}_i\}_{i\in[mk]}}{\text{maximize}} \quad & \textstyle\sum_{i\in[mk]} \mathbf{x}_i \cdot \mathbf{v}_i + \delta \sum_{i\in[mk]} H(\mathbf{x}_i) & & \\ \text{s.t.} \quad & \mathbf{x}_i \in \Delta^m & & i = 1, \ldots, mk \\ & \sum_{i\in[mk]} x_{i,j} \leq k & & j = 1, \ldots, m \end{aligned}$$

where $H(\cdot)$ is the Shannon's entropy function, i.e. $H(\mathbf{x}_i) = -\sum_{j=1}^{m} x_{i,j} \ln(x_{i,j})$.

One can think of the convex program in Definition 5.2 as an extension to the maximum weighted budgeted matching with an entropy regularizer term added to the objective function. The following observation, which can easily be proven by writing the KKT conditions, is the reason that this program is interesting for our application.

Observation 1. *Suppose $\boldsymbol{\alpha}^* \in \mathbb{R}^m$ are the optimal Lagrangian dual variables corresponding to the budget constraints of the RSLM convex program (Definition 5.2), i.e.*

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha} \geq 0} \ \max \left\{ \mathcal{L}(\mathbf{x}_1, \dots, \mathbf{x}_{mk}, \boldsymbol{\alpha}) \mid \forall i \in [mk]; \mathbf{x}_i \in \Delta^m \right\}$$

where $\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) \triangleq \sum_{i \in [mk]} \mathbf{x}_i \cdot \mathbf{v}_i + \delta \sum_{i \in [mk]} H(\mathbf{x}_i) + \sum_{j \in [m]} \alpha_j (k - \sum_{i \in [mk]} x_{i,j})$ is the Lagrangian function. Then, the following solution to the (primal) convex program is optimal.

$$x^*_{i,j} = \frac{\exp\left(\frac{v_{i,j} - \alpha^*_i}{\delta}\right)}{\sum_{r \in [m]} \exp\left(\frac{v_{i,r} - \alpha^*_r}{\delta}\right)}, \quad \forall i \in [mk], j \in [m].$$

Here is how Observation 1 will be helpful. Suppose replica-agent i has replica type r. Then his value for surrogate outcome distribution O_j is a random value $v(r, o_j)$, where $o_j \sim O_j$. This is an instance of the single-agent multiple urns problem where we have an urn for each surrogate outcome distribution O_j. Interestingly, based on the result of Section 4, we know how to efficiently implement the exponential weight allocations (similar to the optimal allocation described in Observation 1) by using the logistic Bernoulli race algorithm (Algorithm 3). The only difference is this new allocation requires an additive shift of values by dual variables first, which can easily be taken care of truthfully. More concretely, we have the following lemma.

Lemma 5.1. *Given $\boldsymbol{\alpha} \in [0, h]^m$ and a replica $i \in [km]$, allocating a surrogate outcome j to replica i, where j is randomly drawn from the following distribution over $[m]$,*

$$x_{i,j} = \frac{\exp\left(\frac{v_{i,j} - \alpha_j}{\delta}\right)}{\sum_{r \in [m]} \exp\left(\frac{v_{i,r} - \alpha_r}{\delta}\right)}, \quad \forall j \in [m], \tag{4}$$

is a DSIC (in expectation) allocation algorithm from the perspective of this replica-agent. Moreover, this allocation algorithm can be implemented by only $O\left(\frac{h^5 m^2}{\log^4(m/\delta)\delta^5}\right)$ calls to $\mathcal{A}$.

Proof We show this allocation algorithm is an affine maximizer in expectation, and therefore is DSIC. Consider the following affine maximization.

$$\max \quad \left(1 - \frac{\epsilon}{2}\right) \sum_{j=1}^{m} x_{i,j} \cdot \frac{v_{i,j} + h - \alpha_j}{h+1} + \left(\frac{\epsilon}{2 \log m}\right) H(\mathbf{x}_i) \quad s.t. \quad \mathbf{x}_i \in \Delta^n, \tag{5}$$

Similar to Section 4, it is easy to see that by setting $\epsilon = \frac{2\delta \log m}{\delta \log m + h + 1} \geq \frac{\delta \log m}{h}$, the probability distribution in Equation (4) will be the optimal solution for this maximization program, and therefore the randomized allocation algorithm in the lemma statement is an affine maximizer in expectation. For efficient implantation, let r_i be the replica type of replica-agent i. First, transform random variables $\{v(r_i, o_j)\}_{j \in [m]}$ to random variables $\left\{\frac{v(r_i, o_j) + h - \alpha_j}{h+1}\right\}_{j \in [m]}$. Then by running Algorithm 3 on the transformed value random variables, one can draw a random choice of surrogate outcome distribution from

$\mathbf{x}_i$. Moreover, following Theorem 3.4 and the fact that $\epsilon \geq \frac{\delta \log m}{h}$, we only need $O(\frac{h^5 m^2}{\log^4(m/\delta)\delta^5})$ many calls to sampling oracles, or equivalently in this case to $\mathcal{A}$ black-box. Note that the transformed random variables form a valid input to Algorithm 3 as $v_j = \mathbb{E}_{o_j \sim O_i}\{\frac{v(r_i,o_j)+h-\alpha_j}{h+1}\} = \frac{v_{i,j}+h-\alpha_j}{h+1} \in [0,1]$ when $\alpha_j \in [0,h]$ and $v_{i,j} \in [0,1]$. □

Now, one might think that implementing an optimal solution to the RSLM truthfully can be done by just using Lemma 5.1 with $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$. However, such an allocation rule cannot be truthful as α^* depends on the entire replica type profile, and therefore from the perspective of any replica-agent i the allocation in (4) is not a DSIC allocation.

5.3 Online RSLM Under Random Arrival Order

In order to find a near-optimal solution to the RSLM convex program that can be implemented truthfully, we look at the "*online RSLM under random arrival order*" problem, where replica-agents arrive online based on a uniform random permutation. Inspired by online algorithms for stochastic online convex programming [Agrawal and Devanur, 2015, Chen and Wang, 2013] or stochastic online packing [Agrawal et al., 2009, Devanur et al., 2011, Badanidiyuru et al., 2013, Kesselheim et al., 2014], we propose an online algorithm (Algorithm 4 below) that at each step uses multiplicative weight updates to learn the Lagrangian dual variables using the history so far, and then uses Lemma 5.1 to implement the appropriate single-agent incentive compatible allocation rule with the learned dual variables (that just depend on the replica-agent types arriving before this agent). We analyze three aspects of Algorithm 4. The first aspect, which is easy at this point to answer, is whether this allocation is truthful. The second aspect is how much welfare do we lose by this allocation algorithm. We answer the second question in two parts. We start by showing that the online algorithm for RSLM is a near-optimal competitive algorithm with respect to the offline RSLM problem. We then show how this result will be translated into a small additive welfare loss compared to $\mathcal{A}$. The final aspect is the sample complexity of Algorithm 4, i.e. how many calls to $\mathcal{A}$ it requires.

Remark Algorithm 4 initializes a parameter γ such that with probability at least $1-\eta$, we have $\frac{\texttt{OPT}}{k} \leq \gamma \leq O(1) \cdot \frac{\texttt{OPT}}{k}$. Interestingly, such a parameter can be calculated truthfully by only efficient sampling and estimating the optimal solution, which will be explained in more details later.

Estimating γ through sampling. We need to initialize γ such that $\frac{\texttt{OPT}}{k} \leq \gamma \leq O(1) \cdot \frac{\texttt{OPT}}{k}$. To this end, we find the empirical estimate $\hat{\mathbf{v}}_i$ of $\mathbf{v}_i$ for all $i \neq i^*$ by sampling (Recall that i^* denotes the index of the real agent replica, i.e. the replica that corresponds to the real agent's reported type). We then use those estimates to solve the following convex program, denoted by *empirical* RSLM.

$$\begin{aligned} \underset{\{\mathbf{x}_i\}_{i\neq i^*}}{\text{maximize}} \quad & \textstyle\sum_{i\neq i^*} \mathbf{x}_i \cdot \hat{\mathbf{v}}_i + \delta \sum_{i\neq i^*} H(\mathbf{x}_i) & \\ \text{s.t.} \quad & \mathbf{x}_i \in \Delta_m & i \in [mk] \setminus i^* \\ & \sum_{i\neq i^*} x_{i,j} \leq k & j = 1,\ldots,m \end{aligned}$$

First of all, as we never ask the real agent for its type, this step is obviously incentive compatible. Moreover, suppose $\mathbf{x}^*$ is the optimal solution of the empirical RSLM convex program and $\mathbf{x}^{**}$ is the optimal solution of the original RSLM convex program. Suppose $\forall(i,j) : \hat{v}_{i,j} \geq v_{i,j} - \epsilon$ for some

Algorithm 4 Online Replica-Surrogate Logistic Matching (online RSLM)

given: parameters $\delta, \eta \in \mathbb{R}_+$, positive integers m, k.
Shuffle the replicas uniformly at random, indexed by $1, \ldots, mk$.
initialize: $\boldsymbol{\alpha}^{(1)} = \frac{1}{m}\mathbf{1}$ and $\hat{\mathbf{x}}_i = \mathbf{0}, i \in [mk]$. ▷ setting the initial dual variables and the assignment.
initialize: γ. ▷ constant approximation to the offline optimal problem.
initialize: $i = 1$.
repeat
    Using Algorithm 3, assign a random surrogate outcome $\hat{\jmath}_i$ to replica-agent i such that:

$$\Pr(\hat{\jmath}_i = j) = \frac{\exp\left(\frac{v_{i,j} - \gamma\alpha_j^{(i)}}{\delta}\right)}{\sum_{r\in[m]} \exp\left(\frac{v_{i,r} - \gamma\alpha_r^{(i)}}{\delta}\right)}, \quad \forall j \in [m]$$

    Let $\hat{x}_{i,j} = 1$ for $j = \hat{\jmath}_i$, and let $\hat{x}_{i,j} = 0$ for all $j \neq \hat{\jmath}_i$.
    Update $\boldsymbol{\alpha}^{(i+1)}$ using multiplicative weight update rule:

$$\forall j \in [m] : w_j^{(i)} = \exp\left(\eta \cdot \sum_{i' \le i} \hat{x}_{i',j}\right) \ , \quad \boldsymbol{\alpha}^{(i+1)} = \frac{\mathbf{w^{(i)}}}{\mathbf{w^{(i)}} \cdot \mathbf{1}}$$

    $i \leftarrow i + 1$.
until $(\exists\, j \in [m] : \sum_{i' \le i} \hat{x}_{i',j} = k)$ or $(i > mk)$.
if $i \le mk$ **then**
    Let $\tau = i$ (termination time).
    for $i' = \tau, \tau + 1, \ldots, mk$ **do**
        Assign a surrogate outcome with non-zero remaining budget to replica-agent i' at random.
    end for
end if

$\epsilon > 0$ (This event holds with high probability with enough samples). As $\{\mathbf{x}_i^{**}\}_{i\neq i^*}$ also forms a feasible solution for the empirical RSLM, we have:

$$\begin{aligned}
\frac{\texttt{OPT}}{k} \ge \frac{\texttt{OPT}_{\text{empirical}}}{k} \triangleq & \frac{1}{k}\left(\sum_{i\neq i^*} \mathbf{x}_i^* \cdot \hat{\mathbf{v}}_i + \delta \sum_{i\neq i^*} H(\mathbf{x}_i^*)\right) \ge \frac{1}{k}\left(\sum_{i\neq i^*} \mathbf{x}_i^{**} \cdot \hat{\mathbf{v}}_i + \delta \sum_{i\neq i^*} H(\mathbf{x}_i^{**})\right) \\
\ge & \frac{1}{k}\left(\sum_{i} \mathbf{x}_i^{**} \cdot \hat{\mathbf{v}}_i + \delta \sum_{i} H(\mathbf{x}_i^{**}) - (1 + \delta\log(m))\right) \\
\ge & \frac{1}{k}\left(\sum_{i} \mathbf{x}_i^{**} \cdot \mathbf{v}_i + \delta \sum_{i} H(\mathbf{x}_i^{**}) - (\epsilon \cdot km + 1 + \delta \cdot \log(m))\right) \\
= & \frac{\texttt{OPT}}{k} - \epsilon \cdot m - \frac{1}{k} - \frac{\delta \cdot \log(m)}{k} \ge \frac{\texttt{OPT}}{k}\left(1 - \frac{\epsilon}{\delta \cdot \log(m)} - \frac{1}{\delta \cdot mk\log(m)} - \frac{1}{mk}\right)
\end{aligned} \tag{6}$$

where the last inequality holds as $\texttt{OPT}$ is bounded below by the value of the uniform allocation, i.e. $\texttt{OPT} \ge \delta \cdot mk\log(m)$. Now suppose $\delta \cdot m\log(m) = \Omega(1)$. By setting $\epsilon = O(\delta \cdot \log(m))$, the right hand side will be lower-bounded by $c \cdot \frac{\texttt{OPT}}{k}$ for some constant $c < 1$. We now have the following technical lemma.

Lemma 5.2. *Given $\eta \in (0, 1)$, let $\gamma \triangleq 1/(ck) \cdot \texttt{OPT}_{empirical}$, where each $\hat{v}_{i,j}$ is the empirical mean of*

L samples. If $\delta \cdot m\log(m) = \Omega(1)$ and $L = \Omega\left(\frac{\log(\frac{m^2k}{\eta})}{\delta^2\cdot\log^2(m)}\right)$, then with probability at least $1 - O(\eta)$ we have $\frac{\texttt{OPT}}{k} \le \gamma \le O(1)\cdot\frac{\texttt{OPT}}{k}$.

Proof By using the Chernoff-Hoeffding inequality, with probability at least $1 - O(\eta)$ we have $\forall(i,j) : \hat{v}_{i,j} \ge v_{i,j} - O(\delta\cdot\log(m))$ when $L = \Omega\left(\frac{\log(\frac{m^2k}{\eta})}{\delta^2\cdot\log^2(m)}\right)$. The lemma then holds due to (6). □

5.4 Incentive Compatibility

Lemma 5.3. *In the mechanism design problem of assigning surrogate outcomes to replica-agents, the allocation algorithm implemented by Algorithm 4 is DSIC in expectation.*

Proof The lemma simply holds by the following observations: (1) γ's calculation only depends on the sampled replica types of non-real replicas (and not the real agent i^*'s reported type), (2) at the time that algorithm processes each replica i (including the real agent i^*) the dual variables $\boldsymbol{\alpha}^{(i)}$ only depend on replica types that arrived before, and (3) the randomized allocation for each single replica i is implemented truthfully in expectation by Lemma 5.1. □

By Lemma 5.3, using Algorithm 4 as the surrogate selection rule will result in a Bayesian truthful reduction. So the following corollary is immediate.

Corollary 5.4. *Given oracle access to allocation algorithm $\mathcal{A}$, the RSLM reduction outputs a BIC allocation algorithm $\hat{\mathcal{A}}$. This allocation together with implicit payments $\mathbf{p}$ (computed as explained in Section 5.1) forms a BIC mechanism.*

5.5 Social Welfare Loss

In this section, we show that the welfare loss of our reduction is bounded by $O(\epsilon)$. We do this in two steps. We first prove that Algorithm 4 is $(1-O(\epsilon))$ competitive, and second show how to translate this competitive ratio to the $O(\epsilon)$ welfare loss.

5.5.1 Competitive Analysis

Lemma 5.5. *For a fixed $\delta > 0, \eta > 0$ and $m \in \mathbb{N}$, Algorithm 4 is $(1-O(\eta))$-competitive with respect to the optimal offline solution of the RSLM convex program (Definition 5.2) if $\frac{m\log m}{\eta^2} \le k \le m^{O(1)}$.*

Proof Suppose $\frac{\texttt{OPT}}{k} \le \gamma \le O(1)\cdot\frac{\texttt{OPT}}{k}$ (this happens with probability at least $1-\eta$). For $i \in [1, \tau-1]$, let $\texttt{ALG}_i \triangleq \mathbf{x}_i\cdot\mathbf{v}_i + \delta H(\mathbf{x}_i)$ be the reward of algorithm at time i, where

$$x_{i,j} \triangleq \frac{\exp\left(\frac{v_{i,j}-\gamma\alpha_j^{(i)}}{\delta}\right)}{\sum_{r\in[m]}\exp\left(\frac{v_{i,r}-\gamma\alpha_r^{(i)}}{\delta}\right)}, \quad \forall j\in[m], i\in[1:\tau-1].$$

Moreover, let $\mathcal{H}_{i-1}$ denote all the observations, decisions, and realized randomness until time $i-1$. Given $\mathcal{H}_{i-1}$, Observation 1 implies that

$$\mathbf{x}_i = \underset{\mathbf{x}_i'\in\Delta^m}{\operatorname{argmax}}\ \mathcal{L}^{(i)}(\mathbf{x}_i', \gamma\boldsymbol{\alpha}^{(i)}) \tag{7}$$

where $\mathcal{L}^{(i)}(\mathbf{x}'_i, \boldsymbol{\alpha}^{(i)}) \triangleq \mathbf{x}'_i \cdot \mathbf{v}_i + \delta H(\mathbf{x}'_i) + \sum_{j\in[m]} \gamma\alpha_j^{(i)}(\frac{1}{m} - x'_{i,j})$. Let $\{\mathbf{x}_i^*\}_{i\in[mk]}$ be the offline optimal solution and $r_i^* = \mathbf{x}_i^* \cdot \mathbf{v}_i + \delta H(\mathbf{x}_i^*)$. (Thus, $\texttt{OPT} = \sum_{i\in[mk]} r_i^*$.) If replicas are arriving in a uniformly random order, then $\mathbb{E}\{r_i^*\} = \frac{1}{mk}\texttt{OPT}$ and $\mathbb{E}\{\mathbf{x}_i^*\} \preceq \frac{1}{m}\mathbf{1}$. By applying (7) for $i \in [1, \tau-1]$, we have

$$\texttt{ALG}_i + \sum_{j\in[m]} \gamma\alpha_j^{(i)}\big(\tfrac{1}{m} - x_{i,j}\big) \geq r_i^* + \sum_{j\in[m]} \gamma\alpha_j^{(i)}\big(\tfrac{1}{m} - x_{i,j}^*\big)$$

so, by rearranging the terms and taking expectations conditioned on the observed history, we have

$$\begin{aligned}
\mathbb{E}\{\texttt{ALG}_i|\mathcal{H}_{i-1}\} \geq\ & \gamma\mathbb{E}\{\boldsymbol{\alpha}^{(i)} \cdot \mathbf{x}_i|\mathcal{H}_{i-1}\} + \mathbb{E}\{r_i^*|\mathcal{H}_{i-1}\} - \gamma\mathbb{E}\{\boldsymbol{\alpha}^{(i)} \cdot \mathbf{x}_i^*|\mathcal{H}_{i-1}\} \\
=\ & \mathbb{E}\{r_i^*\} - \gamma\boldsymbol{\alpha}^{(i)} \cdot \mathbb{E}\{\mathbf{x}_i^*\} + \gamma\boldsymbol{\alpha}^{(i)} \cdot \hat{\mathbf{x}}_\mathbf{i} - (\mathbb{E}\{r_i^*\} - \mathbb{E}\{r_i^*|\mathcal{H}_{i-1}\}) \\
& + \gamma\boldsymbol{\alpha}^{(i)} \cdot (\mathbb{E}\{\mathbf{x}_i^*\} - \mathbb{E}\{\mathbf{x}_i^*|\mathcal{H}_{i-1}\}) + \gamma\boldsymbol{\alpha}^{(i)} \cdot (\mathbb{E}\{\mathbf{x}_i|\mathcal{H}_{i-1}\} - \hat{\mathbf{x}}_i) \\
\geq\ & \frac{1}{mk}\texttt{OPT} + \gamma\boldsymbol{\alpha}^{(i)} \cdot (\hat{\mathbf{x}}_i - \frac{1}{m}\mathbf{1}) - L_i - L'_i
\end{aligned}$$

where

$$\begin{aligned}
L_i &\triangleq \gamma\ \boldsymbol{\alpha}^{(i)}.(\hat{\mathbf{x}}_i - \mathbb{E}\{\mathbf{x}_i|\mathcal{H}_{i-1}\})\ , \\
L'_i &\triangleq |(\mathbb{E}\{r_i^*\} - \mathbb{E}\{r_i^*|\mathcal{H}_{i-1}\}| + \gamma\|\mathbb{E}\{\mathbf{x}_i^*\} - \mathbb{E}\{\mathbf{x}_i^*|\mathcal{H}_{i-1}\}\|\ .
\end{aligned}$$

By summing the above inequalities for $i = 1 : \tau - 1$ we have:

$$\sum_{i=1}^{\tau-1} \mathbb{E}\{\texttt{ALG}_i|\mathcal{H}_{i-1}\} \geq \frac{\tau-1}{mk}\texttt{OPT} + \gamma\sum_{i=1}^{\tau-1} \boldsymbol{\alpha}^{(i)} \cdot (\hat{\mathbf{x}}_i - \frac{1}{m}\mathbf{1}) - \sum_{i=1}^{\tau-1}(L_i + L'_i) \tag{8}$$

In order to bound the term $\gamma\sum_{i=1}^{\tau-1} \boldsymbol{\alpha}^{(i)} \cdot (\hat{\mathbf{x}}_i - \frac{1}{m}\mathbf{1})$, let $g_i(\boldsymbol{\alpha}) \triangleq \boldsymbol{\alpha} \cdot (\hat{\mathbf{x}}_i - \frac{1}{m}\mathbf{1})$. Then, by applying the regret bound of multiplicative weight online learning algorithm for any realization of random variables $\{\hat{\mathbf{x}}_i\}$, we have

$$\sum_{i=1}^{\tau-1} g_i(\boldsymbol{\alpha}^{(i)}) \geq (1-\eta)\max_{\|\boldsymbol{\alpha}\|_1\leq 1, \boldsymbol{\alpha}\geq\mathbf{0}} \sum_{i=1}^{\tau-1} g_i(\boldsymbol{\alpha}) - \frac{\log m}{\eta} \geq (1-\eta)(k - \frac{\tau-1}{m}) - \frac{\log m}{\eta} \tag{9}$$

where the last inequality holds because at the time $\tau-1$, either there exists j such that $\sum_{i=1}^{\tau-1} \hat{x}_{i,j} = k$, or $\tau - 1 = mk$ and all surrogate outcome budgets are exhausted. In the former case, we have

$$\max_{\|\boldsymbol{\alpha}\|_1\leq 1, \boldsymbol{\alpha}\geq\mathbf{0}} \sum_{i=1}^{\tau-1} g_i(\boldsymbol{\alpha}) \geq \sum_{i=1}^{\tau-1} g_i(\mathbf{e}_j) \geq k - \frac{\tau-1}{m},$$

and in the latter case we have

$$\max_{\|\boldsymbol{\alpha}\|_1\leq 1, \boldsymbol{\alpha}\geq\mathbf{0}} \sum_{i=1}^{\tau-1} g_i(\boldsymbol{\alpha}) \geq 0 \geq k - \frac{\tau-1}{m}.$$

Combining (8) and (9), letting $Q_i = L_i + L'_i$, and assuming $\hat{\mathbf{x}}_j = 0$ for $j \geq \tau$, we have:

$$\begin{aligned}
\sum_{i=1}^{mk} \mathbb{E}\{\texttt{ALG}_i|\mathcal{H}_{i-1}\} \geq \sum_{i=1}^{\tau-1} \mathbb{E}\{\texttt{ALG}_i|\mathcal{H}_{i-1}\} &\geq \frac{\tau-1}{mk}\texttt{OPT} + \gamma(1-\eta)(k - \frac{\tau-1}{m}) - \gamma\frac{\log m}{\eta} - \sum_{i=1}^{\tau-1} Q_i \\
&\geq \frac{\tau-1}{mk}\texttt{OPT} + \frac{\texttt{OPT}}{k}(1-\eta)(k - \frac{\tau-1}{m}) - O(1)\cdot\texttt{OPT}\frac{\log m}{k\eta} - \sum_{i=1}^{mk} Q_i \\
&\geq (1-\eta)\texttt{OPT} - O(\eta)\cdot\texttt{OPT} - \sum_{i=1}^{mk} Q_i
\end{aligned} \tag{10}$$

where the last inequality holds simply because $k > \frac{\log m}{\eta^2}$. By taking expectations from both sides, we have

$$\mathbb{E}\{\texttt{ALG}\} \geq (1 - O(\eta)) \cdot \texttt{OPT} - \sum_{i=1}^{mk} (\mathbb{E}\{L_i\} + \mathbb{E}\{L'_i\}) \tag{11}$$

We now bound each term separately. Define $Y_i \triangleq \sum_{i' \leq i} L_i$. Note that $\mathbb{E}\{Y_i - Y_{i-1} | \mathcal{H}_{i-1}\} = 0$, and therefore sequence $\{Y_i\}$ forms a martingale. Now, by using concentration of martingales we have the following lemma.

Lemma 5.6. $\mathbb{E}\{\sum_{i=1}^{mk} L_i\} \leq \gamma O(\sqrt{km \log km})$.

Proof of Lemma 5.6 Sequence $\{Y_i\}_{i=1}^{mk}$ forms a martingale, as $\mathbb{E}\{Y_i - Y_{i-1} | \mathcal{H}_{i-1}\} = 0$ and using Cauchy-Schwarz

$$|Y_i - Y_{i-1}| = \gamma |\boldsymbol{\alpha}^{(i)} \cdot (\hat{\mathbf{x}}_i - \mathbb{E}\{\mathbf{x}_i | \mathcal{H}_{i-1}\})| \leq \gamma \|\boldsymbol{\alpha}^{(i)}\| \cdot \|\hat{\mathbf{x}}_i - \mathbb{E}\{\mathbf{x}_i | \mathcal{H}_{i-1}\}\| \leq 2\gamma \ .$$

By using Azuma's inequality, we have

$$\Pr\{|Y_{mk}| \geq t\} \leq \exp\big(-\tfrac{t^2}{4km\gamma^2}\big).$$

Let $t = \gamma\sqrt{2km \log(km)}$, then $\Pr\{|Y_{mk}| \geq \gamma\sqrt{2km \log(km)}\} \leq \frac{1}{\sqrt{km}}$. Therefore,

$$\mathbb{E}\{\sum_{i=1}^{mk} L_i\} \leq \mathbb{E}\{|Y_{mk}|\} \leq \gamma\sqrt{2km \log(km)} + \frac{1}{\sqrt{km}}.2\gamma km = \gamma O(\sqrt{km \log km}). \quad \square$$

To bound the second term, we use an argument based on Lemma 4.1 in Agrawal and Devanur [2015]. In fact, we have the following lemma.

Lemma 5.7. $\mathbb{E}\{\sum_{i=1}^{mk} L'_i\} \leq \gamma O(\sqrt{k \log m})$.

Proof of Lemma 5.7 Using Lemma 4.1 in Agrawal and Devanur [2015] with $S = \{v \in \mathbb{R}^m : v \leq \frac{1}{m}\mathbf{1}\}$, we have $\mathbb{E}\{\sum_{i=1}^{mk} L'_i\} \leq \gamma O(\sqrt{skm \log m})$ where $s = \max_{v \in S} \max_{j \in [m]} v_j$. Obviously, $s = \frac{1}{m}$, which completes the proof. $\square$

Using Lemmas 5.7 and 5.6, combined with the facts that $\gamma \leq O(1) \cdot \frac{\texttt{OPT}}{k}$ and $k \geq \frac{m \log m}{\eta^2}$, we have $\mathbb{E}\{\sum_{i=1}^{mk} L_i + L'_i\} \leq O(\eta)\texttt{OPT}$. Together with (11), we conclude that $\mathbb{E}\{\texttt{ALG}\} \geq (1 - O(\eta))\texttt{OPT}$. This holds conditioned on $\frac{\texttt{OPT}}{k} \leq \gamma \leq O(1) \cdot \frac{\texttt{OPT}}{k}$. Moreover, γ is calculated by sampling such that this event happens with probability at least $(1 - \eta)$, which completes the proof. $\square$

5.5.2 Analyzing the Welfare Loss

In order to finalize our analysis of welfare loss, we have to compare the expected welfare of Algorithm 4 (normalized by a factor mk) to the expected welfare of the agent under allocation $\mathcal{A}$. To do this comparison, we consider the *maximum replica-surrogate logistic matching* and bound its normalized welfare loss compared to the *identity surrogate-to-surrogate-outcome matching*, whose expected normalized welfare is basically the welfare of the agent under allocation $\mathcal{A}$. We heavily use the machinery developed in [Hartline et al., 2015]. In particular, we endow the type space $\mathcal{T}^{\mathtt{a}}$ of an agent $\mathtt{a} \in [n]$ with the *distinguishability metric*,

$$d(t, t') = \sup_{o, o' \in \mathcal{O}} \{v(t, o) - v(t', o) - v(t, o') + v(t', o')\},$$

and we quantify the sample complexity of the reduction in terms of the *doubling dimension* of this metric space.

Definition 5.3 ([Hartline et al., 2015])**.** The doubling dimension of a metric is the smallest constant Δ such that every bounded subset S can be partitioned into at most 2 subsets, each having diameter at most half of the diameter of S.

Lemma 5.8. *If for every agent $\mathtt{a} \in [n]$ the type space of agent $\mathcal{T}^{\mathtt{a}}$ has doubling dimension bounded by $\Delta \geq 1$, then the total expected welfare loss of the RSLM reduction is bounded by $O(\epsilon)$ if $m = \Omega\left((\frac{n}{\epsilon})^{\Delta+1}\right)$, $\delta = O(\Delta^{-1} \cdot (\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))^{-1})$ and $\eta = O(\epsilon)$.*

Proof Fix an agent $\mathtt{a}$. For a given $\mathbf{r} \in \mathcal{T}^{mk}$ and $\mathbf{s} \in \mathcal{T}^{m}$, let $\mathbf{r}^{(l)} \triangleq (r_{(l-1)m+1}, \ldots, r_{lm})$ for $l \in [k]$. Moreover, suppose $\mathbf{x}^*$ denotes the maximum (fractional) replica-surrogate logistic k-to-1 matching, $\mathbf{x}$ denotes the output of Algorithm 4 (reduction's replica-surrogate allocation probabilities), and $\Pi^{(l)}(\mathbf{r}^{(l)}, \mathbf{s})$ denotes the edge set of the maximum matching between replicas $\mathbf{r}^{(l)}$ and surrogate outcomes. Letting $v(r,s) \triangleq \mathbb{E}\{v(r, \mathcal{A}(s))\}$ be the interim values, we have:

$$\begin{aligned}
\mathbb{E}\{\mathrm{val}^{\mathtt{a}}(\tilde{\mathcal{A}}) | \mathbf{r}_{-i^*}, \mathbf{s}\} &\geq \frac{1 - O(\eta)}{km}\left(\sum_{(r,s)} x^*_{r,s} v(r,s) + \delta \sum_{i \in [km]} H(\mathbf{x}^*_i)\right) - \delta \log(m) \\
&\geq \frac{1 - O(\eta)}{km}\left(\sum_{l=1}^{k} \sum_{(r,s) \in \Pi^{(l)}(\mathbf{r}^{(l)}, \mathbf{s})} v(r,s)\right) - \delta \log(m) \qquad (12)
\end{aligned}$$

where this holds simply by using Lemma 5.5 and using the facts that (i) the union of maximum matchings $\Pi^{(l)}(\mathbf{r}^{(l)}, \mathbf{s})$ for $l \in [k]$ forms a feasible solution for the RSLM convex program in Definition 5.2, (ii) the entropy of a distribution over a sample space of size m is bounded by $\log(m)$, and (iii) j^* is a uniformly random replica. For two m tuples $\mathbf{r}^{(l)}$ and $\mathbf{s}$ let $d(\mathbf{r}^{(l)}, \mathbf{s})$ be the *transportation cost* as defined in Proposition 3.1 of [Hartline et al., 2015], i.e.

$$d(\mathbf{r}^{(l)}, \mathbf{s}) \triangleq \min\left\{\frac{1}{m}\sum_{(r,s)\in\Pi} d(r,s) | \Pi \text{ is a perfect matching between } \mathbf{r}^{(l)} \text{ and } \mathbf{s}\right\}. \qquad (13)$$

Now, by taking expectation from both sides of (12) with respect to $(\mathbf{r}, \mathbf{s})$ and using Proposition 3.1 in Hartline et al. [2015], we have:

$$\begin{aligned}
\mathbb{E}\{\mathrm{val}^{\mathtt{a}}(\tilde{\mathcal{A}})\} &\geq \frac{1 - O(\eta)}{km}\left(\sum_{l=1}^{k} \mathbb{E}\left\{\sum_{(r,s) \in \Pi^{(l)}(\mathbf{r}^{(l)}, \mathbf{s})} v(r,s)\right\}\right) - \delta \log(m) \\
&\geq \frac{1 - O(\eta)}{km}\left(\sum_{l=1}^{k} \mathbb{E}\{\sum_{i \in [m]} v(s_i, s_i)\} - \frac{m}{2}\sum_{l=1}^{k} \mathbb{E}\{d(\mathbf{r}^{(l)}, \mathbf{s})\}\right) - \delta \log(m) \\
&\geq (1 - O(\eta))\mathbb{E}_{s\sim F}\{v(s, \mathcal{A}(s))\} - \frac{1}{2}\mathbb{E}\{d(\mathbf{r}^{(1)}, \mathbf{s})\} - \delta \log(m) \overset{(1)}{\geq} \mathbb{E}\{\mathrm{val}^{\mathtt{a}}(\mathcal{A})\} - O(\frac{\epsilon}{n}), \qquad (14)
\end{aligned}$$

where in inequality (1) we use Theorem 3.4 and Theorem 3.2 in Hartline et al. [2015] and the facts that $m = \Omega\left((\frac{n}{\epsilon})^{\Delta+1}\right)$, $\delta = O(\Delta^{-1} \cdot (\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))^{-1})$ and $\eta = O(\epsilon)$. By summing over all agents, we conclude that $\mathbb{E}\{\mathrm{val}(\tilde{\mathcal{A}})\} \geq \mathbb{E}\{\mathrm{val}(\mathcal{A})\} - O(\epsilon)$. □

5.6 Sample Complexity

In order to get polynomial sample complexity and welfare loss bounded by ϵ, let the parameters of the Algorithm 4 be initialized as following:

- $m = \Omega\left((\frac{n}{\epsilon})^{\Delta+1}\right)$, $\eta = O(\epsilon)$, and $k = \frac{m^{1.01}\log m}{\eta^2}$,
- $\delta = O(\Delta^{-1} \cdot (\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))^{-1})$,
- $L = \Omega\left(\frac{\log(\frac{m^2 k}{\eta})}{\delta^2 \cdot \log^2(m)}\right)$,

where constant Δ is an upper-bound on the doubling dimension of type spaces.

Lemma 5.9. *Algorithm 4 requires ploy$(\frac{1}{\epsilon}, n)$ calls to the allocation black-box $\mathcal{A}$.*

Proof Algorithm 4 has $m \cdot k = \text{poly}(n, \frac{1}{\epsilon})$ arriving replicas assuming Δ is a constant. Upon arrival of each replica, this online algorithm runs the logistic Bernoulli race algorithm (Algorithm 3) as explained in Lemma 5.1 to match the arriving replica i to a surrogate outcome j. Note that $\gamma = O(\texttt{OPT}/k) = O(m)$, and therefore $h = O(m)$ in Lemma 5.1. So, this step can be done by $\text{poly}(n, \frac{1}{\epsilon})$ many calls to the allocation algorithm $\mathcal{A}$. Finally, note that by the above choices of m, δ, k and η we have $\delta \cdot m\log(m) = \Omega(1)$, and so by Lemma 5.2 calculating γ only requires $\text{poly}(n, \frac{1}{\epsilon})$ number of calls to $\mathcal{A}$. Putting all pieces together we have the lemma. □

Now by using Algorithm 4 as the surrogate selection rule Γ of the RSM reduction explained in Section 5.1, the following corollary is immediate.

Corollary 5.10. *The RSLM reduction (with parameters set as mentioned above) requires only poly$(n, \frac{1}{\epsilon})$ many calls to the allocation algorithm $\mathcal{A}$.*

Acknowledgment

The authors would like to thank Nikhil Devanur, Shipra Agrawal and Pooya Jalaly for helpful discussions and comments on various parts of the paper.

A Omitted Lemmas and Proofs

Proof of Lemma 2.2 This procedure only requires $\mathbb{E}_{K\sim\mathcal{D}}\{K\}$ coin flips in expectation. Moreover,

$$\begin{aligned}\Pr\{\text{output=1}\} &= \mathbb{E}\left\{\mathbf{1}\{\text{all } k \text{ coins-tosses yield 1}\}\right\}\\ &= \mathbb{E}_{k\sim\mathcal{D}}\{\mathbb{E}\{\text{all } k \text{ coins-tosses yield 1}|k\}\}\\ &= \mathbb{E}_{k\sim\mathcal{D}}\{p^k\} = f(p).\end{aligned}$$ □

Proof of Lemma 2.4 The first block works because $\mathbb{E}\{B|v = x\} = \Pr\{u \le x\} = x$, and therefore $\mathbb{E}\{B\} = \mathbb{E}\{v\}$. The second one works because of the independence of u and the coin.

Lemma A.1. *If $X(\mathbf{r}, \mathbf{s})$ is a perfect k-to-1 matching for the instance in Definition 5.1, then Γ^X is distribution preserving.*

Proof Each surrogate s_j is an i.i.d. sample from F. Moreover, by the principle of deferred decisions the index i^* (the real agent's index in the replica type profile) is a uniform random index in $[mk]$, even after fixing the matching. Since this choice of replica is uniform in $[mk]$ and X is a perfect k-to-1 matching, the selection of surrogate outcome is uniform in $[m]$, and therefore the selection of surrogate type associated with this outcome is also uniform in $[m]$. As a result, the output distribution of the selected surrogate type is F. □

Lemma A.2. *If $X(\mathbf{r}, \mathbf{s})$ is a feasible replica-surrogate k-to-1 matching and a DSIC allocation rule (in expectation over allocation's random coins), then the composition of Γ^X and interim allocation algorithm $\mathcal{A}(.)$ forms a BIC allocation algorithm for the original mechanism design problem.*

Proof Each replica-agent $i \in [mk]$ (*including* the real agent i^*) bests off by reporting his true replica type under some proper payments. Now, consider an agent in the original mechanism design problem with true type t. For any given surrogate type profile $\mathbf{s}$, using the Γ^X-reduction the agent receives the same outcome distribution as the one he gets matched to in X in a Bayesian sense, simply because of distribution preservation of Γ^X (Lemma A.1). As allocation X is DSIC, this agent doesn't benefit from miss-reporting his true type as long as the value he receives for reporting t' is $v(t, \mathcal{A}(\Gamma^X(t')))$. Therefore conditioning on $\mathbf{s}$ and non-real replicas in $\mathbf{r}$, the final allocation is BIC from the perspective of this agent. The lemma then follows by averaging over the random choice of $\mathbf{s}$ and non-real agent replicas in $\mathbf{r}$. □

B Basics and Notations of Bayesian Mechanism Design

Bayesian and dominant strategy truthfulness. We are only interested in designing mechanisms that are *interim truthful*, i.e. every agent bests of by reporting her true type assuming all other agent's reported types are drawn independently from their prior type distribution. More precisely, a mechanism $\mathcal{M}$ is *Bayesian Incentive Compatible (BIC)* if for all agents k, and all types $s^k, t^k \in \mathcal{T}^k$,

$$\mathbb{E}_{\mathbf{t}^{-k} \sim \mathbf{F}^{-k}}\{v(t^k, \mathcal{A}^k(t^k))\} - p^k(t^k) \geq \mathbb{E}_{\mathbf{t}^{-k} \sim \mathbf{F}^{-k}}\{v(t^k, \mathcal{A}^k(s^k))\} - p^k(s^k) \tag{15}$$

As a stronger notion of truthfulness than Bayesian truthfulness, one can consider *dominant strategy truthfulness*.More precisely, a mechanism $\mathcal{M}$ is *Dominant Strategy Incentive Compatible (DSIC)* if for all agents k, and all types $s^k, t^k \in \mathcal{T}^k$ and all types $\mathbf{t}^{-k} \in \mathcal{T}^{-k}$,

$$v(t^k, \mathcal{A}(\mathbf{t})) - p^k(\mathbf{t}) \geq v(t^k, \mathcal{A}(s^k, \mathbf{t}^{-k})) - p^k(s^k, \mathbf{t}^{-k}) \tag{16}$$

Moreover, an allocation algorithm $\tilde{\mathcal{A}}$ is said to be BIC (or DSIC) if there exists a payment rule $\tilde{\mathbf{p}}$ such that $\tilde{M} = (\tilde{\mathcal{A}}, \tilde{\mathbf{p}})$ is a BIC (or DSIC) mechanism. Throughout the paper, we use the terms Bayesian (or dominant strategy) truthful and Bayesian (or dominant strategy) incentive compatible interchangeably. For randomized mechanisms, DSIC and BIC solution concepts are defined by considering expectation of utilities of agents over mechanism's internal randomness.

Social welfare. We are considering mechanism design for maximizing *social welfare*, i.e. the sum of the utilities of agents and the mechanism designer. For quasi-linear agents, this quantity is in fact sum of the valuations of the agents under the outcome picked by the mechanism. For the allocation algorithm $\mathcal{A}$, we use the notation $\text{val}(\mathcal{A})$ for the expected welfare of this allocation and $\text{val}^k(\mathcal{A})$ for the expected value of agent k under this allocation, i.e. $\text{val}(\mathcal{A}) \triangleq \mathbb{E}_{\mathbf{t} \sim \mathbf{F}}\{\sum_k v(t^k, \mathcal{A}(\mathbf{t}))\}$ and $\text{val}^k(\mathcal{A}) \triangleq \mathbb{E}_{\mathbf{t} \sim \mathbf{F}}\{v(t^k, \mathcal{A}(\mathbf{t}))\}$.

C Implicit Payment Computations - Details

In this section we describe one standard reduction for computing implicit payments in our general setting, given access to a BIC allocation algorithm $\tilde{\mathcal{A}}$: a multi-parameter counterpart of the single-parameter payment computation procedure used for example by Archer et al. [2004], Hartline and Lucier [2010], which makes $n+1$ calls to $\tilde{\mathcal{A}}$, thus incurring a factor $n+1$ overhead in running time. (A different implicit payment computation procedure, described in Babaioff et al. [2013, 2015], avoids this overhead by calling $\tilde{\mathcal{A}}$ only once in expectation, but incurs a $1-\epsilon$ loss in expected welfare and potentially makes payments of magnitude $\Theta(1/\epsilon)$ *from* the mechanism *to* the agents.)

The implicit payment computation procedure assumes that the agents' type spaces $(\mathcal{T}^k)_{k\in[n]}$ are *star-convex at 0*, meaning that for any agent k, any type $t^k \in \mathcal{T}^k$, and any scalar $\lambda \in [0,1]$, there is another type $\lambda t^k \in \mathcal{T}^k$ with the property that $v(\lambda t^k, o) = \lambda v(t^k, o)$ for every $o \in \mathcal{O}$. (The assumption is without loss of generality, as argued in the next paragraph.) The implicit payment computation procedure, applied to type profile $\mathbf{t}$, samples $\lambda \in [0,1]$ uniformly at random and computes outcomes $o^0 \triangleq \tilde{\mathcal{A}}(\mathbf{t})$ as well as $o^k \triangleq \tilde{\mathcal{A}}(\lambda t^k, \mathbf{t}^{-k})$ for all $k \in [n]$. The payment charged to agent k is $v(t^k, o^0) - v(t^k, o^k)$. Note that, in expectation, agent k pays

$$p^k(\mathbf{t}) = v(t^k, \tilde{\mathcal{A}}(\mathbf{t})) - \int_0^1 v(t^k, \tilde{\mathcal{A}}(\lambda t^k, \mathbf{t}^{-k}))\, d\lambda,$$

in accordance with the payment identity for multi-parameter BIC mechanisms when type spaces are star-convex at 0; see Babaioff et al. [2013] for a discussion of this payment identity.

Finally, let us justify the assumption that $\mathcal{T}^k$ is star-convex for all k. This assumption is without loss of generality for the allocation algorithms $\tilde{\mathcal{A}}$ that arise from the RSM reduction, because we can enlarge the type space $\mathcal{T}^k$ if necessary by adjoining types of the form λt^k with $t^k \in \mathcal{T}^k$ and $0 \le \lambda < 1$. Although the output of the original allocation algorithm $\mathcal{A}$ may be undefined when its input type profile includes one of these artifically-adjoined types, the RSM reduction never inputs such a type into $\mathcal{A}$. It only calls $\mathcal{A}$ on profiles of surrogate types sampled from the type-profile distribution F, whose support excludes the artificially-adjoined types. Thus, even when the input to $\tilde{\mathcal{A}}$ includes an artifically-adjoined type λt^k, it occurs as one of the replicas in the reduction. The behavior of algorithm $\tilde{\mathcal{A}}$ remains well-defined in this case, because replicas are only used as inputs to the valuation function $v(r_i, o_j)$, whose output is well-defined even when $r_i = \lambda t^k$ for $\lambda < 1$.

References

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

Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *arXiv preprint arXiv:0911.2974*, 2009.

Aaron Archer, Christos Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics*, 1(2): 129–150, 2004.

Moshe Babaioff, Robert Kleinberg, and Aleksandrs Slivkins. Multi-parameter mechanisms with implicit payment computation. In *Proceedings of the 14th ACM Conference on Electronic Commerce*, pages 35–52, 2013.

Moshe Babaioff, Robert Kleinberg, and Aleksandrs Slivkins. Truthful mechanisms with implicit payment computation. *J. ACM*, 62(2):10:1–10:37, May 2015. ISSN 0004-5411.

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

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

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

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

Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 29–38. ACM, 2011.

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

Jason D. Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian incentive compatibility via matchings. *SODA*, 2011.

Jason D. Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian incentive compatibility via matchings. *Games and Economic Behavior*, 92(C):401–429, 2015.

Zhiyi Huang and Sampath Kannan. The exponential mechanism for social welfare: Private, truthful, and nearly optimal. In *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science*, pages 140–149. IEEE, 2012.

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

MS Keane and George L O'Brien. A bernoulli factory. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 4(2):213–219, 1994.

Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 303–312. ACM, 2014.

Krzysztof Łatuszyński. The bernoulli factory, its extensions and applications. *Proceedings of IWAP 2010*, pages 1–5, 2010.

Şerban Nacu, Yuval Peres, et al. Fast simulation of new coins from old. *The Annals of Applied Probability*, 15(1A):93–115, 2005.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007. ISBN 0521872820.