Bayesian Algorithmic Mechanism Design

[Extended Abstract]

Jason D. Hartline*
Northwestern University
Evanston, IL, USA
hartline@eecs.northwestern.edu

Brendan Lucier
University of Toronto
Toronto, ON, Canada
blucier@cs.toronto.edu

ABSTRACT

The principal problem in algorithmic mechanism design is in merging the incentive constraints imposed by selfish behavior with the algorithmic constraints imposed by computational intractability. This field is motivated by the observation that the preeminent approach for designing incentive compatible mechanisms, namely that of Vickrey, Clarke, and Groves; and the central approach for circumventing computational obstacles, that of approximation algorithms, are fundamentally incompatible: natural applications of the VCG approach to an approximation algorithm fails to yield an incentive compatible mechanism. We consider relaxing the desideratum of (ex post) incentive compatibility (IC) to Bayesian incentive compatibility (BIC), where truthtelling is a Bayes-Nash equilibrium (the standard notion of incentive compatibility in economics). For welfare maximization in single-parameter agent settings, we give a general black-box reduction that turns any approximation algorithm into a Bayesian incentive compatible mechanism with essentially the same[1] approximation factor.

Categories and Subject Descriptors

J.4 [**Social And Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics, Theory

Keywords

Mechanism design, Bayesian incentive compatibility, algorithms, social welfare.

*Supported in part by NSF Grant CCF-0830773 and NSF Career Award CCF-0846113.

[1]More specifically, we obtain a polynomial time approximation scheme with an ϵ loss that is either additive or multiplicative, depending on the problem setting. This error term arises from statistical methods that seem necessary for a black-box reduction.

STOC'10, June 5–8, 2010, Cambridge, Massachusetts, USA.

1. INTRODUCTION

Can any approximation algorithm be converted into an approximation mechanism for selfish agents? This question is framed by a fundamental incompatibility between the standard economic approach for the design of mechanisms for selfish agents (the *Vickrey-Clarke-Groves* (VCG) mechanism) and the standard algorithmic approach for circumventing computational intractability (approximation algorithms). The conclusion from this incompatibility, driving much of the field of algorithmic mechanism design, is that incentive and algorithmic constraints must be dealt with simultaneously (See e.g., [15]). For a large, important class of problems, we arrive at the opposite conclusion: *there is a general approximation-preserving reduction from mechanism design to algorithm design!*

The goal of mechanism design is to construct the rules for a system of agents so that in the equilibrium of selfish agent behavior a desired objective is obtained. For settings of incomplete information the standard game theoretic equilibrium concept is *Bayes-Nash equilibrium* (BNE), which is defined by mutual best response when the *prior distribution* of agent payoffs is *common knowledge*. The *revelation principle* [18] suggests that when looking for mechanisms with desirable Bayes-Nash equilibria, one must look no further than those with truthtelling as a Bayes-Nash equilibrium, also known as *Bayesian incentive compatible* (BIC) mechanisms. Almost all of the computer science literature has focused on the BIC subclass of *ex post incentive compatible* (IC) mechanisms where truthtelling is a *dominant strategy*. While IC is aesthetically appealing because it is congruous with worst-case-style results, it is not generally without loss!

This loss is evident in the computer science theory of IC mechanism design, which is described most characteristically by impossibility. For instance, for single-minded combinatorial auctions of m items, the optimal worst-case approximation factor (under standard complexity theoretic assumptions) is $\sqrt{m}$ [17]. With such impossibility, a relevant theory must make relaxations. For many problems within the realm of computer systems; e.g., online auctions (eBay), advertising auctions (Google, Yahoo!, MSN), file sharing (BitTorrent), routing (TCP/IP), scheduling (SETI@home), and video streaming (YouTube); high volume should enable demand distributions to be estimated. With demand distributions, the natural algorithmic and mechanism design problems are Bayesian.

Agent incentives in Bayesian mechanism design are very well understood in single-parameter settings, where each agent has a single independent private value for receiving a service (see, e.g, [18]). For the single parameter setting it is known that a mechanism is BIC if and only if (a) the probability an agent is served (a.k.a. the *allocation rule*) is monotone non-decreasing in the agent's value for service, and (b) the agent's expected payment (a.k.a. the *payment rule*) is identified precisely from the allocation rule.[2]

The main challenge in reducing BIC (or IC) mechanism design to algorithm design is that approximation algorithms do not generally have monotone allocation rules. Our reduction shows that in a Bayesian setting we can convert any non-monotone allocation rule into a monotone one without compromising its social welfare. The main technical observation that enables this reduction is that, in a Bayesian setting, we can focus on a single agent for whom the allocation rule is not monotone, apply a transformation that fixes this non-monotonicity (and weakly improves our objective), and *no other agents are affected* (in a Bayesian sense). Therefore, we can apply the transformation independently to each agent. Our reduction is as follows:

1. For each agent, identify intervals in which the agent's allocation rule is non-monotone. (This is a property of the distribution and algorithm and can be done prior to considering any agent bids.)

2. For each agent, if their bid falls in an (above identified) interval, redraw the agent's bid from the prior distribution conditioned on being within the interval.

3. Run the approximation algorithm on the resulting bids and output its solution.

Notice that under the assumption that the original values are drawn according to the common prior, the redrawing of values does not alter this prior.

Three items must be clarified. First, there are many ways one might try to choose intervals in Step 1 of the reduction and most of them are incompatible with mechanism design. To address this issue, we develop a monotonizing technique for allocation rules (adapted from the standard *ironing procedure* from the field of Bayesian optimal mechanism design [18]). Second, we are unlikely to have access the functional form of the allocation rule. To address this issue, we estimate the allocation rule by sampling the distribution and making black-box calls to the algorithm. These estimates can be made precise enough to enable arbitrary small loss in welfare (i.e., a fully polynomial time approximation scheme). Finally, we must also determine payments for our monotonized allocation rule. For this, a general approach of Archer et al. [3] suffices.

Our results apply generally to single-parameter agent settings where the designer's objective is to maximize the social welfare (e.g., single-minded combinatorial auctions). Agents each have a private value for service. The designer either has a feasibility constraint on the set of agents that can be simultaneously served or a cost function over the set of served agents. The social welfare is the sum of the values of the agents served less the designer's cost.

For any ϵ, we give a black box reduction that, in polynomial time in the number of agents and $1/\epsilon$, converts any approximation algorithm into a BIC mechanism with an additive loss of ϵ to the social welfare. We also give a pseudo-polynomial time reduction to a BIC mechanism with a multiplicative loss of ϵ, and a fully polynomial time approximation scheme for the special case of downward-closed feasibility problems. Thus, the approximation complexity of social welfare in single-parameter settings is the same for algorithms and BIC mechanisms.

For the most studied single-parameter mechanism design problems, the performance of the best ex post IC approximation mechanism matches the best approximation algorithm (e.g., single-minded combinatorial auctions [17] and related machine scheduling [12]). None-the-less, our approach gives the best known BIC approximation mechanism for many problems, such as auctions under various graph constraints [2] and auctions of convex bundles [5].

Related Work. The design of ex post IC mechanisms for social welfare problems is well studied, notably for the specific settings of combinatorial auctions [3, 5, 16, 17]. Lehmann et al. [17] introduced the problem of polynomial time approximation of social welfare for single-minded combinatorial auctions and give a mechanism that matches the best algorithmic approximation factor. Archer et al. [3] considered the setting where there are many (at least logarithmic) copies of each item and gave a $(1 + \epsilon)$-approximation mechanism. Archer and Tardos [4] gave a (single-parameter) related machine scheduling mechanism that approximates the makespan. Dhangwatnotai et al. [12] gave a mechanism for related machine scheduling that approximates makespan and matches the algorithmic lower bound. All of the above results are for ex post incentive compatible mechanisms.

There has been a large literature on multi-parameter combinatorial auctions and approximation, but this is only tangentially related to our work so we do not cite it exhaustively.

There have been a few reductions of varying degrees of generality. Lavi and Swamy [16] consider IC mechanisms for multi-parameter packing problems and give a technique for constructing a (randomized) β-approximation mechanism from any β-approximation algorithm that verifies an integrality gap. Babaioff et al. [6] look at the equilibrium notion of *algorithmic implementation in undominated strategies* and gives a technique for turning a β-algorithm into a $\beta(\log v_{max})$-approximation mechanism. This solution concept requires that no agent plays a strategy that is dominated by an easy to find strategy. Their approach applies to single-valued combinatorial auctions and does not require the mechanism to know which bundles each agent desires.

There have been a few related studies of Bayes-Nash equilibrium. Christodoulou et al. [11] consider Bayes-Nash equilibria of simultaneous Vickrey auctions in a combinatorial setting and show that these give a 2-approximation when agents' valuations are submodular. Gairing et al. [14] consider Bayes-Nash equilibria of a routing game and study worst-case performance. Borodin and Lucier [8] study worst-case performance of Bayes-Nash equilibria in combinatorial mechanisms based on greedy algorithms.

There are many papers on profit maximization that consider Bayesian design settings. These papers do not tend to consider computational constraints and for many of these (non-computational) settings the restriction to ex post in-

[2]Probabilities and expectations above are taken with respect to both the distribution of agent values and possible randomization in the mechanism.

centive compatibility is without loss.[3] One exception exception is Bhattacharya et al. [7]. They focus on the problem of selling heterogeneous goods to agents with linear valuations, for which they construct a polynomial time 4-approximation mechanism. Their approximation result requires that the type distributions satisfy the monotone hazard rate assumption. In a spirit similar to this paper, they make heavy use of the Bayesian setting to obtain a polynomial runtime.

Organization. We describe in detail the model for single-parameter agents, Bayesian approximation, Bayesian incentive compatibility, and foundational economic theory in Section 2. In Section 3 we give the reduction in an ideal setting where the allocation rule of the algorithm for the given distribution on agent values is precisely known. This reduction is lossless. In Section 4 we develop the reduction in the black-box model where we must sample the distribution and run the algorithm to determine its allocation. Conclusions and open problems are discussed in Section 5.

2. MODEL AND DEFINITIONS

Algorithms. We consider algorithms for binary single-parameter agent settings. An algorithm in such a setting must select a set of agents to serve. This *allocation* is denoted by $\mathbf{x} = (x_1, \ldots, x_n)$ where x_i is an indicator for whether or not agent i is served. Agent i has *valuation* v_i for being served. Without loss for non-negative bounded-support distributions, we will assume $v_i \in [0, 1]$.[4] The vector $\mathbf{v} = (v_1, \ldots, v_n)$ of valuations is the *valuation profile*.

In the *general costs setting*, the seller may have some cost function $c(\cdot)$ over allocations representing the cost of serving the allocated set (e.g., Steiner tree problems [13]). The *general feasibility setting* is the special case where costs are zero (feasible) or infinity (infeasible). These include scheduling and public good problems. An important subclass are *downward-closed settings* where any subset of a feasible set is feasible. Downward closed settings include single-minded combinatorial auctions [17] and knapsack auctions [1].

An algorithm $\mathcal{A}$ is simply an *allocation rule* that maps valuation profiles to allocations. The allocation rule for $\mathcal{A}$ we will denote by $\mathbf{x}(\mathbf{v})$. Our objective is the *social welfare* which is $\mathcal{A}(\mathbf{v}) = \sum_i v_i x_i(\mathbf{v}) - c(\mathbf{x}(\mathbf{v}))$. We allow $\mathcal{A}$ to be randomized in which case $x_i(\mathbf{v})$ is a random variable; $\mathcal{A}(\mathbf{v})$ denotes the expected welfare of the algorithm for valuation profile $\mathbf{v}$. $\mathrm{OPT}(\mathbf{v})$ will denote the maximum social welfare.

We will consider these algorithmic problems in a Bayesian (a.k.a. stochastic) setting where the valuations of the agents are drawn from a product distribution $\mathbf{F} = F_1 \times \cdots \times F_n$. Agent i's *cumulative distribution* and *density* functions are denoted F_i and f_i, respectively. The distribution is assumed to be *common knowledge* to the agents and designer.

The pair $(c(\cdot), \mathbf{F})$ defines a setting for single-parameter algorithm design which we will take as implicit. For this setting, the optimal expected welfare is $\mathrm{OPT} = \mathbf{E}_{\mathbf{v}\sim\mathbf{F}}[\mathrm{OPT}(\mathbf{v})]$ and the algorithm's expected welfare is $\mathcal{A} = \mathbf{E}_{\mathbf{v}\sim\mathbf{F}}[\mathcal{A}(\mathbf{v})]$. An algorithm is a *worst-case β-approximation* if for all $\mathbf{v}$, $\mathcal{A}(\mathbf{v}) \geq \mathrm{OPT}(\mathbf{v})/\beta$. An algorithm is a *Bayesian β-approximation* if $\mathcal{A} \geq \mathrm{OPT}/\beta$.

Mechanisms. A mechanism $\mathcal{M}$ consists of an *allocation rule* and a *payment rule*. We denote by $\mathbf{x}(\mathbf{v})$ and $\mathbf{p}(\mathbf{v})$ the allocation and payment rule of an implicit mechanism $\mathcal{M}$. We assume agents are *risk neutral* and individually desire to maximize their expected *utilities*. Agent i's utility for allocation $\mathbf{x}$ and payments $\mathbf{p}$ is $v_i x_i - p_i$. We consider single-round, sealed-bid mechanisms where agents simultaneously bid and the mechanism then computes the allocation and payments.

Our goal is a mechanism that has good social welfare in equilibrium. The standard economic notion of equilibrium for games of incomplete information is *Bayes-Nash equilibrium* (BNE). The revelation principle says that any equilibrium that is implementable in BNE is implementable with truthtelling as the BNE strategies of the agents.[5] Meaning: an agent that believes the other agents are reporting their values truthfully as given by the distribution has a best response of also reporting truthfully. A mechanism with truthtelling as a BNE is *Bayesian incentive compatible* (BIC).[6]

It will be useful to consider agent i's expected payment and probability of allocation conditioned on their value. To this end, denote $p_i(v_i) = \mathbf{E}_{\mathbf{v},\mathcal{A}}[p_i(\mathbf{v}) \mid v_i]$ and $x_i(v_i) = \mathbf{E}_{\mathbf{v},\mathcal{A}}[x_i(\mathbf{v}) \mid v_i]$. The following theorem characterizes BIC mechanisms.

THEOREM 2.1. *[18] A mechanism is BIC if and only if for all agents i:*

- *$x_i(v_i)$ is monotone non-decreasing, and*
- *$p_i(v_i) = v_i x_i(v_i) - \int_0^{v_i} x_i(z)dz + p_i(0)$.*

Usually, $p_i(0)$ is assumed to be zero.

This motivates the following definition:

DEFINITION 2.1. *An allocation rule $\mathbf{x}(\cdot)$ is* monotone *for distribution $\mathbf{F}$ if $x_i(v_i)$ is monotone non-decreasing for all i. An algorithm is* monotone *if its allocation rule is monotone.*

From Theorem 2.1, BIC and monotone are equivalent and we will use them interchangeably for both algorithms and mechanisms, though we will prefer "BIC" when the focus is incentive properties and "monotone" when the focus is algorithmic properties.

Computation. Our main task in demonstrating that the approximation complexity of algorithms and BIC mechanisms is the same by giving an approximation-preserving reduction from the BIC mechanism design problem to the algorithm design problem. In other words, we use the algorithm's allocation rule to compute the mechanism's allocation and payment rules. As we are in a Bayesian setting this computation will also need access to the distribution.

[3] One non-computational setting where ex post incentive compatibility is with loss is when the profit maximizing seller has a strict no-deficit constraint [10].

[4] The bounded support assumption is unnecessary except for our results using sampling, where we believe it is realistic.

[5] The revelation principle holds even in computational settings; any BNE for which the agent strategies and the mechanism can be computed in polynomial time can be converted into a polynomial time BIC mechanism.

[6] Much of the computer science literature on mechanism design focuses on dominant strategy equilibrium (DSE) and *ex post incentive compatibility* (IC). This is not without loss in many settings and therefore should be considered with care when addressing computational questions in mechanism design.

We consider two models of computation: an *ideal model* and a *black-box model*. In the ideal model, we will assume we have explicit access to the functional form of the distribution and allocation rule and we will assume we can perform calculus on these functions. While this model is not realistic, we present it for the sake of clarity in explaining the economic theory that drives our results. In the black-box model we will assume we can query the algorithm on any input and that we can sample from the distribution on any subinterval of its support. Our philosophy is that the ideal model is predictive of what is implementable in polynomial time and we verify this philosophy by instantiating approximately the same reduction under the black-box model.

3. REDUCTION: IDEAL MODEL

In this section we prove that, in the ideal model, any Bayesian algorithm can be made BIC without loss of performance.

THEOREM 3.1. *In the ideal model and general cost settings, a BIC algorithm $\bar{\mathcal{A}}$ can be computed from any algorithm $\mathcal{A}$. Its Bayesian social welfare satisfies $\bar{\mathcal{A}} \geq \mathcal{A}$.*

Theorem 3.1 implies an immediate corollary for Bayesian approximation.

COROLLARY 3.2. *In the ideal model and general cost settings, a BIC Bayesian β-approximation $\bar{\mathcal{A}}$ can be computed from any Bayesian β-approximation algorithm, $\mathcal{A}$.*

Let us build some intuition for the requirements of Theorem 3.1. Suppose that we are given an algorithm $\mathcal{A}$ that is monotone for the distribution $\mathbf{F}$. Then $\mathcal{A}$ is already BIC and specifies the allocation rule $\mathbf{x}(\cdot)$, so we must only compute the payment rule. In our ideal model this is trivial given the formula from Theorem 2.1.

Now suppose we have a non-monotone Bayesian β-approximation algorithm $\mathcal{A}$ with allocation rule $\mathbf{x}(\cdot)$. We would like to use $\mathcal{A}$ to construct a monotone algorithm $\bar{\mathcal{A}}$ from which we can obtain a BIC mechanism by simply computing the payment rule as above. We must make sure that in doing so we do not reduce the algorithm's expected welfare. The key property of our approach which makes it tractable is that we monotonize each agent's allocation rule independently without changing (in a Bayesian sense) the allocation rule any other agent faces. This property is also important for the approximation factor as $\mathcal{A}$ is guaranteed to be a Bayesian β-approximation only for the given distribution $\mathbf{F}$, and may not be a good approximation for some other distribution.

In summary, the desiderata for monotonizing agent i are:

D1. monotone $\bar{x}_i(v_i)$,

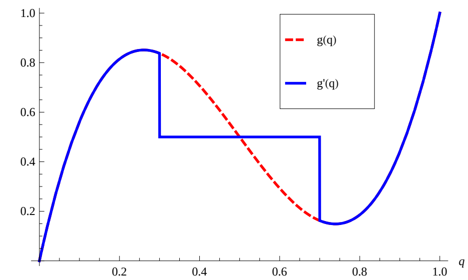
D2. (weakly) improved social welfare $\mathbf{E}_{v_i}[v_i\bar{x}_i(v_i)] \geq \mathbf{E}_{v_i}[v_i x_i(v_i)]$, and

D3. other agents unaffected.

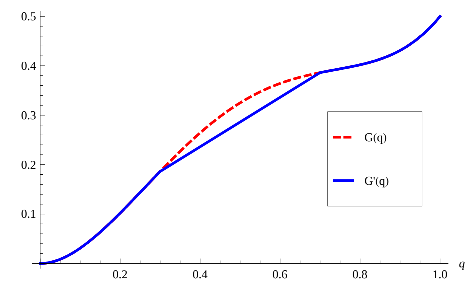
Notice that if we satisfy the last condition we can apply the process simultaneously to all agents.

3.1 Ironing via Resampling

There is a history of fixing non-monotonicities in Bayesian mechanism design. Myerson invented the technique of *ironing* which relies on the fact that if an allocation rule is constant over some interval then any agent within that interval is effectively equivalent to a canonical "average" agent from that interval. Myerson applied this theory to iron *virtual valuation functions* which are used in Bayesian profit maximization [18]. We will apply this theory directly to allocation rules.



(a)



(b)

Figure 1: (a) A non-monotone ironing g' (solid) of curve g (dashed). (b) The corresponding integral curves G' (solid) and G (dashed) in probability space.

Before we describe our ironing procedure in full, let us develop some more intuition. Suppose allocation rule $x_i(\cdot)$ of $\mathcal{A}$ is non-monotone for agent i. A simple approach to flattening non-monotonicities is to choose some interval $[a, b]$ on which $x(\cdot)$ is non-monotone, and to treat the agent identically whenever on this interval. For example, whenever $v_i \in [a, b]$ we could choose to pretend that v_i is actually some other fixed value v' (e.g. $v' = a$) and pass this "pretend" value v' to the algorithm. Unfortunately, if we take this naïve approach, we would have changed the distribution of agent i's input to the algorithm (in particular, the probability of value v' would be increased) and violated D3. In order to maintain D3 we make a minor modification: instead of picking a fixed v', we will draw v' from F_i restricted to the interval $[a, b]$. Thus, we are replacing $v_i \in [a, b]$ with v' drawn from the same distribution. Other agents cannot tell the difference – this operation does not change the distribution of agent i's input! Moreover, agent i will indeed be treated identically whenever $v_i \in [a, b]$: the new probability of allocation will be precisely the distribution weighted average of $x_i(\cdot)$ over the interval $[a, b]$.

Let ${x_i}'(\cdot)$ represent the allocation rule we obtain from the following procedure (where $\mathbf{v}_{-i} \sim \mathbf{F}_{-i}$):

- if $v_i \in [a, b]$, redraw $v' \sim F_i$ restricted to $[a, b]$; else, set $v' = v_i$.

- run $\mathcal{A}(v', \mathbf{v}_{-i})$.

We say that ${x_i}'(\cdot)$ is the curve $x_i(\cdot)$ *ironed on interval* $[a, b]$. We note that ${x_i}'(v_i) = x_i(v_i)$ for $v_i \notin [a, b]$ and ${x_i}'(v_i) = \mathbf{E}_{v' \sim F_i}[x_i(v') \mid v' \in [a, b]]$ otherwise. Note that we can easily iron along multiple disjoint intervals, redrawing v' from whichever interval contains ${v_i}'$ (if any).

We now explore a method for choosing intervals on which to iron in order to obtain monotonicity. It will be instructive to consider the *allocation rule in probability space* instead of valuation space, and the *cumulative allocation rule* (also in probability space).

- Let $g(q) = x_i({F_i}^{-1}(q))$ be the allocation rule in probability space.
- Let $G(q) = \int_0^q g(z)dz$ be the cumulative allocation rule.

Notice that monotonicity of $x_i(\cdot)$ is equivalent to monotonicity of $g(\cdot)$ which is equivalent to convexity of $G(\cdot)$.

Let ${x_i}'(\cdot)$ be $x_i(\cdot)$ ironed along some interval $[a, b]$, and consider the corresponding curves $g'(\cdot)$ and $G'(\cdot)$. This ironing procedure corresponds to replacing $g(\cdot)$ with its average on $[a, b]$, or equivalently $G(\cdot)$ with the line segment connecting $G(F(a))$ to $G(F(b))$ (See Figure 1).[7] This latter line segment interpretation suggests that we can view our interval selection problem as the problem of replacing portions of curve G with straight line segments so that the resulting curve $\bar{G}$ will be convex. This is precisely the problem of finding the convex hull of G! Thus the choice of intervals that monotonizes $x_i(\cdot)$ (satisfying D1) is precisely the set of intervals defined by the convex hull of $G(\cdot)$. See Figure 2.

Finally, since the convex hull of $G(\cdot)$ lies below $G(\cdot)$, this transformation weakly improves welfare (satisfying D2). Informally speaking, in moving from cumulative allocation rule $G(\cdot)$ to $\bar{G}(\cdot)$, we lower the probability of low-value allocations in exchange for a corresponding increase in the probability that higher-valued allocations occur. This intuition is made more precise in Lemma 3.4, below.

3.2 The Ironed Algorithm

We are now ready to define our ironed algorithm $\bar{\mathcal{A}}$. Given distribution F and interval I, we will write $F[I]$ to mean F restricted to I.

Definition 3.1 (Resample $(\mathcal{A}, \mathbfcal{I})$). *Given algorithm $\mathcal{A}$ and a* profile of disjoint interval sets, $\mathbfcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, *the* resampled algorithm *for $\mathcal{A}$ with intervals $\mathbfcal{I}$ is algorithm* Resample $(\mathcal{A}, \mathbfcal{I})$:

1. *For each agent i, if $v_i \in I \in \mathcal{I}_i$, draw $\bar{v}_i \sim F_i[I]$; else, set $\bar{v}_i = v_i$.*
2. *Run $\mathcal{A}(\bar{\mathbf{v}})$.*

Definition 3.2 (MonoInts $(\mathbf{x})$). *The* set of monotonizing intervals for $\mathbf{x}(\cdot)$ *is* MonoInts $(\mathbf{x}) = (\mathcal{I}_1, \ldots, \mathcal{I}_n)$ *defined by:*

1. *Let $g_i(q) = x_i({F_i}^{-1}(q))$ be the allocation rule in probability space.*
2. *Let $G_i(q) = \int_0^q g_i(z)dz$ be the cumulative allocation rule.*
3. *Let $\bar{G}_i(\cdot)$ be the convex hull of $G_i(\cdot)$.*
4. *Let $\mathcal{I}_i$ be the set of intervals in valuation space on which $G_i(F_i(\cdot)) > \bar{G}_i(F_i(\cdot))$.*

[7]Note that the transformation to probability space (from valuation space) is necessary for obtaining this line-segment interpretation.

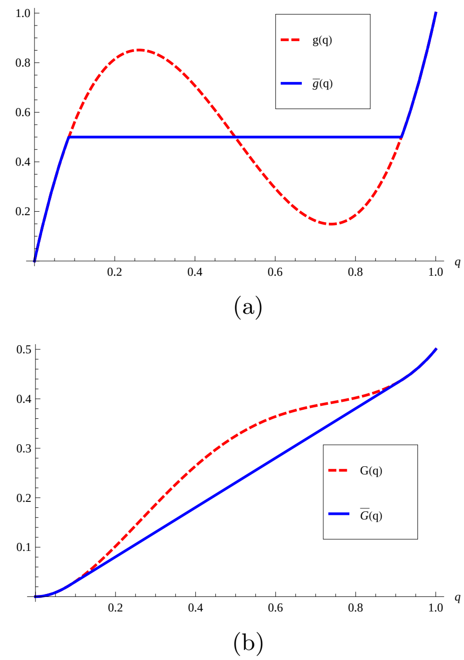


Figure 2: (a) A monotone ironing $\bar{g}$ (solid) of curve g (dashed). (b) The corresponding integral curves $\bar{G}$ (solid) and G (dashed) in probability space. Note $\bar{G}$ is the convex hull of G.

Definition 3.3 ($\bar{\mathcal{A}}$). *The* ironed algorithm *corresponding to algorithm $\mathcal{A}$ is $\bar{\mathcal{A}} =$* Resample $(\mathcal{A}, $ MonoInts $(\mathbf{x}))$.

Lemma 3.3. *$\bar{\mathcal{A}}$ is monotone.*

Proof. We must show that each agent has a monotone allocation rule. The allocation rule for agent i is precisely $\bar{x}_i(v_i) = \bar{g}(F_i(v_i))$, which is the derivative of a convex function and therefore monotone. ☐

Lemma 3.4. *If $\mathcal{A}$ is a Bayesian β-approximation then $\bar{\mathcal{A}}$ is a Bayesian β-approximation.*

Proof. First notice that the two allocation rules produce the same distribution over allocations and therefore expected costs are identical. We will show, for a single agent i, that $\mathbf{E}[v_i\bar{x}_i(v_i)] \geq \mathbf{E}[v_i x_i(v_i)]$, from which linearity of expectation implies the result. We have

$$\begin{aligned}\mathbf{E}[v_i x_i(v_i)] &= \int_0^1 v x_i(v) f_i(v)\, dv = \int_0^1 F_i^{-1}(q) g_i(q)\, dq \\ &= \int_0^1 \int_0^{F_i^{-1}(q)} g_i(q)\, dz\, dq = \int_0^1 \int_{F_i(z)}^1 g_i(q)\, dq\, dz \\ &= \int_0^1 (G_i(1) - G_i(F_i(z)))\, dz\end{aligned}$$

and similarly $\mathbf{E}[v_i\bar{x}_i(v_i)] = \int_0^1 (\bar{G}_i(1) - \bar{G}_i(F_i(z)))dz$. We conclude $\mathbf{E}[v_i\bar{x}_i(v_i)] \geq \mathbf{E}[v_i x_i(v_i)]$ since $\bar{G}_i(1) = G_i(1)$ and $\bar{G}_i(F_i(z)) \leq G_i(F_i(z))$ for all $z \in [0, 1]$. ☐

Theorem 3.1 follows from Lemmas 3.3 and 3.4.

4. REDUCTION: BLACK-BOX MODEL

We will use the ironing procedure from the previous section to monotonize an algorithm in the black-box model. However, instead of using direct knowledge of the allocation rule, we must use sampling to estimate it. This sampling introduces errors in the selection of interval sets for resampling, which must then be dealt with. Our analysis will proceed in the following steps.

1. We describe a method for computing payments in the black-box model.
2. We describe a method for combining sampling with ironing to obtain a nearly monotone algorithm. In fact, this algorithm will be ϵ-Bayesian incentive compatible.
3. We show that a convex combination of this nearly monotone algorithm with a blatantly monotone one will give a monotone algorithm, resulting in a BIC mechanism.

All of these steps approximately preserve social welfare. We obtain the following theorem.

THEOREM 4.1. *In the black-box model and general cost settings, for any $\epsilon > 0$, a BIC algorithm $\mathcal{A}'$ can be computed from any algorithm $\mathcal{A}$. Its Bayesian social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon$, and its runtime is polynomial in n and $1/\epsilon$.*

The additive error in Theorem 4.1 can be converted into a multiplicative error whenever the expected welfare of $\mathcal{A}$ is not too small. We obtain the following corollary.

COROLLARY 4.2. *In the black-box model and general cost settings, for any $\epsilon > 0$, a BIC algorithm $\mathcal{A}'$ can be computed from any algorithm $\mathcal{A}$. Its Bayesian social welfare satisfies $\mathcal{A}' \geq \mathcal{A}/(1+\epsilon)$, and its runtime is polynomial in n, $1/\epsilon$, and $1/\mathcal{A}'$.*

We note that it is possible to improve Theorem 4.1 by considering the expected values of the agents. This improvement is discussed in in Appendix C and enables, for example, a fully polynomial reduction for downward-closed settings.

In the remainder of this section we prove of Theorem 4.1.

4.1 Computing Payments

Suppose that $\mathcal{A}$ has monotone allocation rules. The problem of designing a mechanism to implement $\mathcal{A}$ then reduces to calculating appropriate payments. These payments are completely determined by the allocation rule of $\mathcal{A}$, but in the black-box model we do not have direct access to the functional form of the allocation rule. Archer et al. [3] solve this problem by computing an unbiased estimator of the desired payment rule using only black-box calls to the algorithm. For completeness we now summarize their approach.

DEFINITION 4.1 (BLACK-BOX PAYMENTS). *If algorithm $\mathcal{A}$ does not allocate to agent i, then agent i pays 0. Otherwise, we compute the payment of agent i as follows:*

1. *Choose ${v_i}'$ uniformly from $[0, v_i]$*
2. *Draw $\mathbf{v}'_{-i} \sim \mathbf{F}_{-i}$ and run $\mathcal{A}({v_i}', \mathbf{v}'_{-i})$*
3. *If $\mathcal{A}$ allocated to agent i in the previous step set $X = v_i$, otherwise set $X = 0$.*
4. *If $X \neq 0$, repeatedly draw values $\mathbf{v}'_{-i} \sim \mathbf{F}_{-i}$ and run $\mathcal{A}(v_i, \mathbf{v}'_{-i})$ until the algorithm allocates to player i, and let T be the number of iterations required.*
5. *Agent i's payment is $p_i = v_i - TX$.*

As was shown by Archer et al., this computation attains the appropriate expected payment.

CLAIM 4.3 (ARCHER ET AL. [3]). *In the black-box payment procedure, the expected payments are $p_i(v_i) = v_i x_i(v_i) - \int_0^{v_i} x_i(z)dz$.*

We note that since we execute this procedure for agent i only if he receives an allocation, which occurs with probability $x_i(v_i)$, the expected number of calls to $\mathcal{A}$ for each player is at most

$$x_i(v_i)\left(1 + \frac{1}{x_i(v_i)}\right) \leq 2.$$

Thus, in expectation, all payments can be computed with $2n$ calls to $\mathcal{A}$.

Note that any mechanism paired with the above payment scheme will be *individually rational* (IR), meaning that a truthtelling agent will never obtain negative utility. This is true even if the allocation rule is not monotone. This follows immediately from the fact that the payment for an agent that declares value v_i is never greater than v_i (indeed, it is defined as v_i minus a non-negative value).

4.2 Sampling and ϵ-Bayesian Incentive Compatibility

We will be estimating the allocation rule of a non-monotone algorithm and attempting to iron it. This will fail to result in an absolutely monotone rule. In this section we show that a nearly monotone rule results in truthtelling as an *ϵ-Bayes-Nash equilibrium* (ϵ-BNE): the most an agent can gain from a non-truthtelling strategy is an additive ϵ. We call such a mechanism *ϵ-Bayesian incentive compatible* (ϵ-BIC).

DEFINITION 4.2 (ϵ-BIC). *A mechanism is ϵ-Bayesian incentive compatible if truthtelling obtains at least as much utility as any other strategy, up to an additive ϵ, assuming all other agents truthtell. That is, for all i, v_i, and v', $v_i x_i(v_i) - p_i(v_i) \geq v_i x(v') - p_i(v') - \epsilon$.*

The main theorem of this section is the following.

THEOREM 4.4. *In the black-box model and general cost settings, for any $\epsilon > 0$, an ϵ-BIC algorithm $\mathcal{A}'$ can be computed from any algorithm $\mathcal{A}$. Its Bayesian social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon$, and its runtime is polynomial in n and $1/\epsilon$.*

The resampling procedure from the previous section is the main workhorse for Theorem 4.4. The construction of algorithm $\mathcal{A}'$ consists primarily to choosing interval sets on which to resample.

4.2.1 ϵ-closeness

We now formalize a closeness property under which an allocation rule that is close to monotone is ϵ-BIC (for some related ϵ).

Definition 4.3 (ϵ-close). *Given allocation rules $x(\cdot)$ and $x'(\cdot)$ are ϵ-close if $|x(v) - x'(v)| < \epsilon$ for all v. Two algorithms or mechanisms are ϵ-close if each agent's allocation rules are ϵ-close.*

Lemma 4.5. *If non-monotone $\mathcal{A}'$ is ϵ-close to a monotone $\mathcal{A}$, then $\mathcal{A}'$ is (2ϵ)-BIC.*

Lemma 4.6. *If $\mathcal{A}$ and $\mathcal{A}'$ have the same expected costs*[8] *and are ϵ-close then $\mathcal{A}' \geq \mathcal{A} - n\epsilon$.*

Lemma 4.7. *If algorithms $\mathcal{A}$ and $\mathcal{A}'$ are ϵ-close, then for any collection $\boldsymbol{\mathcal{I}} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ of interval sets, resampled algorithms* Resample $(\mathcal{A}, \boldsymbol{\mathcal{I}})$ *and* Resample $(\mathcal{A}', \boldsymbol{\mathcal{I}})$ *are ϵ-close.*

Appropriate payments to turn an algorithm that is ϵ-close to monotone into a mechanism that is 2ϵ-BIC can be computed by the same process we would use for monotone algorithms.

4.2.2 Discretization

A key step in our reduction will be in discretizing the allocation rules of the algorithm. This reduces the problem of estimating an allocation rule to estimating its value at a polynomial number of points. Moreover, our resulting allocation will not necessarily be monotone, but there will be only a polynomial number of points at which it can be non-monotone; we will use this to our advantage when fixing non-monotonicities in Section 4.3.

An algorithm is *k-piece piecewise constant* if for each i there is a partition of valuation space into at most k intervals such that the allocation rule for agent i is constant on each interval.

Definition 4.4 ($\text{Disc}_\epsilon(\mathcal{A})$). *For a given $\epsilon > 0$ and algorithm $\mathcal{A}$, the* discretization of algorithm $\mathcal{A}$, $\text{Disc}_\epsilon(\mathcal{A})$, *is* Resample $(\mathcal{A}, \boldsymbol{\mathcal{I}})$, *where $\boldsymbol{\mathcal{I}} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ is the collection of intervals defined by*

$$\mathcal{I}_i = \{[0, \epsilon)\} \cup \left\{\left[\epsilon(1+\epsilon)^t, \epsilon(1+\epsilon)^{t+1}\right)\right\}_{0 \leq t \leq \log_{1+\epsilon}(1/\epsilon)}.$$

Lemma 4.8. $\text{Disc}_\epsilon(\mathcal{A})$ *is $\log_{1+\epsilon}(1/\epsilon)$-piece piecewise constant and* $\text{Disc}_\epsilon(\mathcal{A}) \geq \mathcal{A} - 2n\epsilon$.

4.2.3 Statistical Estimation

We next describe a sampling procedure for estimating an allocation rule. This procedure will not form an algorithm, but rather generates an estimated allocation curve, which we will denote by $\mathbf{y}(\cdot)$. This estimate behaves like an allocation rule, but is not associated with an actual algorithm (and, in particular, need not be feasibly implementable).

Definition 4.5 (estimate allocation rule). *Given algorithm $\mathcal{A}$ which is k-piece piecewise constant and $\epsilon > 0$, an* estimated allocation rule *for $\mathcal{A}$ is a curve $\mathbf{y}(\cdot)$ found as follows:*

1. *for each agent i and valuation-space piece I_j, draw $\frac{4}{\epsilon^2}\log(2kn/\epsilon)$ samples from $\mathbf{F}$ conditional on $v_i \in I_j$, and run $\mathcal{A}$ on each of these samples.*
2. *let y_{ij} be the average allocation over the invocations to $\mathcal{A}$ above, for each i and j.*
3. *Define $\mathbf{y}$ by $y_i(v) = y_{ij}$ for all $v \in I_j$*

[8]Recall that the *expected cost* of an algorithm A with allocation rule $\mathbf{x}(\cdot)$ is $\mathbf{E}_{\mathbf{v}\sim\mathbf{F}}[c(\mathbf{x}(\mathbf{v}))]$.

Lemma 4.9. *If algorithm $\mathcal{A}$ is k-piece piecewise constant then, for any $\epsilon > 0$, an estimated allocation rule $\mathbf{y}(\cdot)$ for $\mathcal{A}$ is k-piece piecewise constant, and is $\frac{\epsilon}{2}$-close to $\mathbf{x}(\cdot)$ with probability at least $1 - \frac{\epsilon}{2}$. The number of black-box calls to $\mathcal{A}$ used in the construction of $\mathbf{y}(\cdot)$ is polynomial in n, k, and $1/\epsilon$.*

We now complete the proof of Theorem 4.4 by combining our sampling procedure with the ironing procedure from the ideal model.

Definition 4.6 ($\text{Iron}_\epsilon(\mathcal{A})$). *Given piecewise constant algorithm $\mathcal{A}$, the* statistically ironed algorithm *for $\mathcal{A}$ with error $\epsilon > 0$ is* $\text{Iron}_\epsilon(\mathcal{A}) = \text{Resample}(\mathcal{A}, \text{MonoInts}(\mathbf{y}))$ *where $\mathbf{y}(\cdot)$ is the estimated allocation rule for $\mathcal{A}$.*

Note that $\text{Iron}_\epsilon(\mathcal{A})$ is not simply a resampling of $\mathcal{A}$, but rather a convex combination of resamplings; construction of the interval set $\text{MonoInts}(\mathbf{y})$ is randomized.

Lemma 4.10. $\text{Iron}_\epsilon(\mathcal{A})$ *is 2ϵ-BIC and* $\text{Iron}_\epsilon(\mathcal{A}) \geq \mathcal{A} - n\epsilon$.

Proof of Theorem 4.4: Define $\mathcal{A}'$ to be the algorithm $\text{Iron}_{\epsilon'}(\text{Disc}_{\epsilon'}(\mathcal{A}))$, where $\epsilon' = \epsilon/3n$. Then, by Lemmas 4.8 and 4.10, $\mathcal{A}'$ is $2\epsilon'$-BIC, and hence ϵ-BIC, and $\mathcal{A}' \geq \text{Disc}_{\epsilon'}(\mathcal{A}) - n\epsilon' \geq \mathcal{A} - 3n\epsilon' = \mathcal{A} - \epsilon$. The runtime of $\mathcal{A}'$ (which is dominated by sampling in the construction of $\mathbf{y}$) is $O(nk\epsilon'^{-2}\log(2kn/\epsilon'))) = \tilde{O}(n^3\epsilon^{-3}\log(\epsilon^{-1}))$, where recall $k = \frac{1}{\epsilon}\log(1/\epsilon)$ is the number of discrete intervals in $\text{Disc}_{\epsilon'}(\mathcal{A})$. □

4.3 Bayesian Incentive Compatibility

In the previous section we showed how to construct an ϵ-BIC mechanism from any algorithm with almost no loss to the social welfare. Our goal now is to take such an ϵ-BIC algorithm $\mathcal{A}$ and make it BIC. In other words, we would like to "fix" the (small) non-monotonicities in $\mathcal{A}$. Fortunately, since each allocation curve of $\mathcal{A}$ is discretized, any non-monotonicities must occur only at a small number of predetermined points. Our approach for removing these points of non-monotonicity is simple: we will construct an alternative algorithm $\mathcal{A}'$ whose allocation curves are stair functions, with jumps in allocation probability occurring at each of those points. A convex combination of $\mathcal{A}$ and $\mathcal{A}'$ will then be monotone. This convex combination will be our final BIC algorithm.

It is important that this convex combination process not reduce social welfare by too much. This requires two things. First, we need the convex combination to be mostly $\mathcal{A}$ as only it has provably good welfare. This is possible by taking ϵ so small that the explicit monotonicities in $\mathcal{A}'$ heavily outweigh the non-monotonicities in $\mathcal{A}$ (which are at most ϵ). Second, we need to ensure that the expected social welfare of $\mathcal{A}'$ is not extremely negative.

How should we construct $\mathcal{A}'$? Suppose first that we are in a downward-closed feasibility setting. In this case, the singleton allocation $\{i\}$ is feasible for each agent i. The construction of $\mathcal{A}'$ with stair-function allocation curves is then straightforward: an agent i is chosen uniformly at random;

the algorithm then either allocates to agent i or not, with the probability of allocation following a stair function. Since $\mathcal{A}'$ only returns feasible outcomes, its expected social welfare must be non-negative.

We would like to follow this same approach in general cost settings. However, it may be that, for some i, the particular allocation $\{i\}$ has an extremely high (or infinite) cost, in which case the above algorithm may have an extremely negative social welfare. Note, though, that in our construction we can replace $\{i\}$ with *any* allocation that includes agent i. It is therefore sufficient to find, for each i, some allocation that includes agent i and whose cost is not too high. Once these allocations are found, we can use them to construct the stair algorithm $\mathcal{A}'$.

In some cases finding low-cost allocations may be highly non-trivial. To get around this problem, we observe that as long as algorithm $\mathcal{A}$ has a reasonable probability of allocating to agent i, there must exist low-cost allocations that include i that are returned by $\mathcal{A}$. We can therefore find such allocations by repeatedly sampling outcomes of $\mathcal{A}$. If, on the other hand, we were to take many samples and not find any allocations that include agent i, then we can safely assume that agent i does not contribute much to the expected social welfare of $\mathcal{A}$. In this case, we can trivially monotonize agent i's allocation curve by ironing his entire allocation curve into a single interval, removing the need to find allocations that include him.

4.3.1 The Stair Algorithm

DEFINITION 4.7 ($\textsc{Stair}(\mathcal{A})$). *Let $\mathcal{A}$ be a k-piece piecewise constant algorithm, and suppose $S_1, \ldots, S_n$ and $T_1, \ldots, T_n$ are allocations such that $i \in S_i$ and $i \notin T_i$ for all i. The* stair algorithm for $\mathcal{A}$, $\textsc{Stair}(\mathcal{A})$, *does the following:*

1. *Pick an agent i uniformly from the n agents.*
2. *If v_i is in the jth highest piece of k pieces, allocate to S_i with probability $(j-1)/(k-1)$ and T_i otherwise.*

DEFINITION 4.8 ($\textsc{Comb}_\epsilon(\mathcal{A})$). *Suppose algorithm $\mathcal{A}$ is k-piece piecewise constant. Then $\textsc{Comb}_\epsilon(\mathcal{A})$ is the convex combination of $\mathcal{A}$ with probability $1-\delta$ and $\textsc{Stair}(\mathcal{A})$ with probability δ, where $\delta = 2(k-1)n\epsilon$.*

LEMMA 4.11. *If $\mathcal{A}$ is ϵ-close to a monotone $\mathcal{A}'$, then algorithm $\textsc{Comb}_\epsilon(\mathcal{A})$ is BIC.*

4.3.2 Finding Low-Cost Sets

We now describe the choice of sets $S_1, \ldots, S_n$ and $T_1, \ldots, T_n$ for algorithm $\textsc{Stair}(\mathcal{A})$. In many settings this is trivial (e.g., for downward-closed feasibility problems we can take $S_i = \{i\}$ and $T_i = \emptyset$), but for some problems it might be difficult to find feasible (or low-cost) allocations. Our approach is as follows. Since $\mathcal{A}$ never makes an allocation that generates negative social welfare, we can bound the cost of any allocation made by $\mathcal{A}$. This motivates us to look for a set $S_i \ni i$ returned by $\mathcal{A}$ on some input, for each i. This can be accomplished by sampling. In the event that we do not find a set S_i, it is likely that the probability of allocating to agent i is very low; we can therefore *iron together all intervals* for agent i, effectively removing the need for S_i, without causing much loss to the expected welfare. This operation can be viewed as trimming away agents that are very rarely allocated. The same holds for finding T_i.

DEFINITION 4.9 ($\textsc{Trim}_\epsilon(\mathcal{A})$). *The* trimmed algorithm *for piece-wise constant $\mathcal{A}$ is $\textsc{Trim}_\epsilon(\mathcal{A})$:*

1. *For each agent i and valuation-space piece $I_j \in \mathcal{I}_i$, draw $\frac{4}{\epsilon^2}\log(2n/\epsilon)$ samples from $\mathbf{F}$ conditional on $v_i \in I_j$, and run $\mathcal{A}$ on each of these samples.*
2. *If $\mathcal{A}$ is the same (always or never allocating) for i on every sample, define $\mathcal{I}'_i = \{[0,1]\}$; otherwise, $\mathcal{I}'_i = \mathcal{I}_i$ and we define S_i to be any observed allocation that includes agent i and T_i to be any observed allocation that does not include agent i.*
3. *Run $\textsc{Resample}(\mathcal{A}, \boldsymbol{\mathcal{I}}')$.*

Note that, for each i, either sets $S_i \ni i$ and $T_i \not\ni i$ will be found during the execution of $\textsc{Trim}_\epsilon(\mathcal{A})$, or else the allocation rule of agent i will be made constant.

LEMMA 4.12. $\textsc{Trim}_\epsilon(\mathcal{A}) \geq \mathcal{A} - n\epsilon$.

We are now ready to combine our tools into a BIC mechanism, proving Theorem 4.1.

DEFINITION 4.10 ($\textsc{Mono}_\epsilon(\mathcal{A})$). *Given an algorithm $\mathcal{A}$ and $\epsilon > 0$, the* monotonization of $\mathcal{A}$, $\textsc{Mono}_\epsilon(\mathcal{A})$, *is the algorithm* $\textsc{Comb}_\epsilon(\textsc{Iron}_\epsilon(\textsc{Trim}_\epsilon(\textsc{Disc}_\epsilon(\mathcal{A}))))$.

LEMMA 4.13. $\textsc{Mono}_\epsilon(\mathcal{A})$ *is BIC, and* $\textsc{Mono}_\epsilon(\mathcal{A}) \geq \mathcal{A} - 6kn^2\epsilon$.

Proof of Theorem 4.1: Let $\mathcal{A}'$ be the monotonized algorithm $\textsc{Mono}_{\epsilon'}(\mathcal{A})$, where $\epsilon' = \epsilon/6kn^2$. The result then follows immediately from Lemma 4.13. The runtime, which is dominated by sampling, is $O(kn(\epsilon')^{-2}) = \tilde{O}(\frac{n^5}{\epsilon^5}\log^3(1/\epsilon))$. □

5. CONCLUSIONS

Our main result is for single-parameter agents and the objective of social welfare where we give a black-box reduction that converts any Bayesian approximation algorithm into a Bayesian incentive compatible mechanism. For these settings there is no gap separating the approximation complexity of algorithms and BIC mechanisms.

It is notable that our transformation from an approximation algorithm to a BIC mechanism cannot be duplicated by the agents acting on their own: there are non-monotone algorithms that, when coupled with an appropriate payment rule, do not have any BNE with near the expected welfare as the original algorithm on the true values. A concrete example is given in Appendix B.

While our main theorem is extremely general, the situations not covered by it are of notable interest.

1. Multi-parameter Bayesian mechanism design is not yet well understood, but there is every reason to believe that approximation (which has not been pursued much by the economics literature) has a very interesting and relevant role to play in providing positive results (See, e.g., [9]). For social welfare maximization, a recent lower bound by Papadimitriou et al. [19] for the combinatorial public project problem that shows that there is a gap separating the approximation complexity of algorithms and ex post IC mechanisms. This result, however, does not immediately extend to show a separation for BIC mechanisms. *Is there a black-box reduction from BIC mechanism design to algorithm design in multi-parameter settings?*

2. In the special case where our reduction is applied to a worst-case β-approximation (recall: our reduction applies more generally to Bayesian β-approximations), the resulting BIC mechanism is still only a β-approximation in the weaker Bayesian sense. *Is there a polynomial-time black-box reduction that turns any worst-case β-approximation algorithm into a BIC mechanism that is also a worst-case β-approximation?*

3. While Bayes-Nash equilibrium (i.e., BIC) is the standard equilibrium concept for implementation in economics, the stronger dominant strategy equilibrium (i.e., ex post IC) is the standard concept in computer science. The main challenge in obtaining a similar reduction for IC mechanisms is that the valuation space is exponentially big and monotonizing all points seems to require an exhaustive procedure. One potential approach would be to apply our ironing technique repeatedly, re-ironing an agent's curve whenever it is affected by an ironing of another agent's curve. Such a procedure will not generally give a monotone allocation rule; A concrete example is given in the full version of the paper. *Is there a polynomial-time reduction for turning any $f(n)$-approximation (worst-case or Bayesian) algorithm for a single-parameter domain into an ex post IC mechanism that is a (worst-case or Bayesian) $\Theta(f(n))$-approximation?*

The final question above is a refinement of what we consider to be the main open question of this work. *Is there a gap separating the approximation complexity of implementation by Bayesian incentive compatible and ex post incentive compatible mechanisms for single-parameter social welfare maximization?*

6. REFERENCES

[1] G. Aggarwal and J. Hartline. Knapsack auctions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[2] K. Akcoglu, J. Aspens, B. Dasgupta, and M. Kao. An opportunity-cost algorithm for combinatorial auctions. In *Applied Optimization: Computational Methods in Decision-Making, Economics, and Finance*, 2002.

[3] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. 14th ACM Symp. on Discrete Algorithms*. ACM/SIAM, 2003.

[4] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.

[5] M. Babaioff and L. Blumrosen. Computationally-feasible truthful auctions for convex bundles. In *Proc. 7th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2004.

[6] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *Journal of the ACM*, 2009.

[7] S. Bhattacharya, G. Goel, S. Gollapudi, and K. Munagala. Budget constrained auctions with heterogeneous items. In *Proc. 41st ACM Symp. on Theory of Computing*, 2010.

[8] A. Borodin and B. Lucier. Price of anarchy for greedy auctions. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.

[9] S. Chawla, J. Hartline, D. Malec, and B. Sivan. Sequential posted pricing and multi-parameter mechanism design. In *Proc. 41st ACM Symp. on Theory of Computing*, 2010.

[10] S. Chawla, J. Hartline, U. Rajan, and R. Ravi. Bayesian optimal no-deficit mechanism design. In *Workshop on Internet and Network Economics (WINE)*, 2006.

[11] G. Christodoulou, A. Kovács, and Michael Schapira. Bayesian combinatorial auctions. In *Proc. 35st Intl. Colloq. on Automata, Languages and Programming*, pages 820–832, 2008.

[12] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, 2008.

[13] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

[14] M. Gairing, B. Monien, and K. Tiemann. Selfish routing with incomplete information. In *Proc. 17th ACM Symp. on Parallel Algorithms and Architectures*, 2005.

[15] R. Lavi. Computationally efficient approximation mechanisms. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 12, pages 301–329. Cambridge University Press, 2007.

[16] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, 2005.

[17] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proc. 1st ACM Conf. on Electronic Commerce*, pages 96–102. ACM Press, 1999.

[18] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[19] C. Papadimitriou, M. Schapira, and Y. Singer. On the hardness of being truthful. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, 2008.

APPENDIX

A. IRONING ALLOCATION RULES VS. IRONING VIRTUAL VALUATIONS

The ironing procedure described in this paper is reminiscent of the ironing procedure used by Myerson for maximizing revenue [18]. Myerson's optimal mechanism allocates to maximize virtual value (defined by $\phi_i(v_i) = v_i - \frac{1-F_i(v_i)}{f_i(v_i)}$ for agent i). When the virtual valuation functions are non-monotone, this mechanism first irons the virtual valuation functions, then allocates to maximize ironed virtual value. A natural question is whether this is identical to ironing the non-monotone allocation rule that would result from maximizing non-ironed virtual values. We now give an example demonstrating that these two mechanisms are distinct.

Consider an auction of a single indivisible item to multiple bidders with values drawn i.i.d. from distribution F. Suppose $\mathcal{A}$ assigns the item to the agent with highest virtual value. Let $\bar{\mathcal{A}}$ be the ironed algorithm corresponding to $\mathcal{A}$. Let $\mathcal{A}'$ be Myerson's algorithm, which assigns the item to the agent with the highest *ironed* virtual value. Let $x(\cdot)$, $x'(\cdot)$, and $\bar{x}(\cdot)$ denote the allocation curves of $\mathcal{A}$, $\mathcal{A}'$, and $\bar{\mathcal{A}}$, respectively (by symmetry, allocation curves are the same for each player). Our goal is to show that $\bar{x}(\cdot) \neq x'(\cdot)$.

Consider the following distribution F: with probability 1/2, the value is drawn uniformly from $[10, 11]$; otherwise, it is drawn uniformly from $[11, 15]$. The virtual valuation function corresponding to this distribution is

$$\phi(v) = \begin{cases} 2v - 12 & v \in [10, 11] \\ 2v - 15 & v \in (11, 15]. \end{cases}$$

Let $\phi'(\cdot)$ be the ironed function corresponding to $\phi(\cdot)$. It can be verified that $\phi'(10) < \phi'(v)$ for all $v \in [10, 15]$. This implies that $x'(10) = 0$.

On the other hand, $x(10) > 0$ (since $\phi(v) < \phi(10)$ for $v \in (11, 11.5)$), and furthermore $x(v) \geq 0$ for all $v \in [10, 15]$. Thus, from the definition of the ironing procedure, there is some $z \in [10, 15]$ such that $\bar{x}(10) = \mathbf{E}_v[x(v) \mid v \leq z] > 0$. We therefore conclude $x(\cdot) \neq \bar{x}(\cdot)$, and hence these two different ironing procedures result in different mechanisms.

B. EQUILIBRIA OF NON-MONOTONE ALGORITHMS

We have shown how to transform a non-monotone algorithm into a monotone one to obtain a mechanism that is BIC. It is notable that the agents could not do this on their own: there are non-monotone algorithms that, when coupled with any individually-rational and no-positive-transfer[9] payment rule, do not have any BNE with near the expected welfare as the original algorithm on the true values.

Choose parameter $X \gg n$. Consider an auction of a single indivisible item to n bidders with values drawn i.i.d. from the following distribution: with probability $1/n$ the value is X; with the remaining probability it is drawn uniformly from $[0, 1]$. Let $\mathcal{A}$ allocate to the bidder with the largest value in $\left[0, \frac{1}{n^2}\right]$, if any and breaking ties randomly; otherwise it allocates to the bidder with the largest value.

Consider the expected welfare of $\mathcal{A}$. Since with high probability an agent has value X and no agent has value $1/n^2$ or below, $\mathcal{A} = \Omega(X)$.

Next we show that in any BNE most agents will bid $1/n^2$ and the expected welfare will be the average value of the agents which is $O(X/n)$. Thus, the equiligrium is far from the algorithms Bayesian performance, i.e., the *price of stability* is linear.

Consider any mechanism that pairs $\mathcal{A}$ with an ex-post IR and no-positive-transfer payment scheme. We claim that in any BNE of such a mechanism, an agent with value greater than $\frac{1}{n^2}$ would instead report value $\frac{1}{n^2}$. To see this, consider a BNE and let p denote the probability that some agent declares value $\frac{1}{n^2}$. Suppose that $p < 1 - \frac{1}{n}$. If agent i has value $v_i \in \left[\frac{1}{8}, \frac{3}{8}\right]$ and he does not bid $\frac{1}{n^2}$, then his probability of allocation is at most $p(3/4 + o(1))$ (since otherwise, with high probability, there will be an agent with value at least $2v_i$ who could improve his utility by copying agent i's strategy). The expected utility of agent i is therefore at most $p(3v_i/4 + o(1))$. On the other hand, agent i could bid $\frac{1}{n^2}$ for an expected utility of at least $p\left(v_i - \frac{1}{n^2}\right) > p(v_i/2 + o(1))$ (since $v_i \geq 1/8$). Thus any agent with a value in $\left[\frac{1}{8}, \frac{3}{8}\right]$ will bid $\frac{1}{n^2}$, so $p \geq 1 - \frac{1}{n}$. We conclude by noting that if $p \geq 1 - \frac{1}{n}$, every player with value above $\frac{1}{n^2}$ maximizes his utility by declaring $\frac{1}{n^2}$.

[9] No positive transfers implies that losers have zero payment.

C. EXTENSION: GENERAL VALUATIONS

Theorem 4.1 states that an algorithm can be made BIC with an additive loss of ϵ to its social welfare, in time polynomial in ϵ^{-1}. As noted in Corollary 4.2, this error can be made multiplicative, but at a cost: our reduction is pseudo-polynomial, running in time polynomial in $1/\mathcal{A}$. This might be problematic if (for example) the expected value of each agent is exponentially small. In cases where expected values are very small, we require a stronger theorem to obtain meaningful results for fully polynomial time mechanisms.

In this section we discuss the following extension to Theorem 4.1, which is tailored to cases where the expected welfare of the given algorithm is very small. Let $\mu_{\max} = \max_i \mathbf{E}[v_i]$ be the maximum expected value of any agent.

THEOREM C.1. *In the black-box model and general cost settings, for any $\epsilon > 0$, a BIC algorithm $\mathcal{A}'$ can be computed from any Bayesian algorithm $\mathcal{A}$. Its social welfare satisfies $\mathcal{A}' \geq \mathcal{A} - \epsilon\mu_{\max}$, and its runtime is polynomial in n, $1/\epsilon$, and $\log(1/\mu_{\max})$.*

For the special case of downward-closed set systems for feasibility problems, we can assume that $\mathcal{A} \geq \mu_{\max}$, since the trivial algorithm that simply allocates to the single player with the highest input value attains this value. This implies the following corollary.

COROLLARY C.2. *In the black-box model and downward-closed feasibility settings, For any $\epsilon > 0$, a BIC Bayesian $\beta(1+\epsilon)$-approximation algorithm $\mathcal{A}'$ can be computed from any Bayesian β-approximation algorithm $\mathcal{A}$. Its runtime is polynomial in n, $1/\epsilon$, and $\log(1/\mu_{\max})$.*

The proof of Theorem C.1 closely follows the proof of Theorem 4.1 from Section 4. The main difference is in the conversion of an ϵ-BIC algorithm to a purely BIC algorithm (i.e. corresponding to the proof of Theorem 4.1 from Theorem 4.4 in Section 4.3). Our high-level approach is the same: we consider a convex combination of the almost-monotone algorithm from Theorem 4.4 with the blatantly monotone stair algorithm. Recall that in Section 4.3 some care was necessary when finding sets $S_1, \ldots, S_n$ such that $S_i \ni i$. This task becomes even more difficult in our extended setting, since we wish to bound our error with respect to $\mu_{\max}$. Thus, in addition to requiring that $S_i \ni i$ for each i, we also require that the cost of each set S_i is not too large relative to $\mu_{\max}$. We must therefore modify the sampling procedure in algorithm $\text{TRIM}_\epsilon(\mathcal{A})$ to search specifically for low-cost allocations. The details of this modification are deferred to the full version of the paper.