

On the Impossibility of Black-Box Transformations in Mechanism Design

Shuchi Chawla* Nicole Immorlica† Brendan Lucier‡

Abstract

We consider the problem of converting an arbitrary approximation algorithm for a single-parameter optimization problem into a computationally efficient truthful mechanism. We ask for reductions that are black-box, meaning that they require only oracle access to the given algorithm and in particular do not require explicit knowledge of the problem constraints. Such a reduction is known to be possible, for example, for the social welfare objective when the goal is to achieve Bayesian truthfulness and preserve social welfare in expectation. We show that a black-box reduction for the social welfare objective is not possible if the resulting mechanism is required to be truthful in expectation and to preserve the worst-case approximation ratio of the algorithm to within a subpolynomial factor. Further, we prove that for other objectives such as makespan, no black-box reduction is possible even if we only require Bayesian truthfulness and an average-case performance guarantee.

*Department of Computer Science, University of Wisconsin - Madison.

†Department of Electrical Engineering and Computer Science, Northwestern University.

‡Department of Computer Science, University of Toronto.

1 Introduction

Mechanism design studies optimization problems arising in settings involving selfish agents, with the goal of designing a system or protocol whereby agents’ individual selfish optimization leads to global optimization of a desired objective. A central theme in *algorithmic* mechanism design is to reconcile the incentive constraints of selfish participants with the requirement of computational tractability and to understand whether the combination of these two considerations limits algorithm design in a way that each one alone does not.

In the best-case scenario, one might hope for a sort of equivalence between the considerations of algorithm design and mechanism design, manifested through, for example, general reductions that convert arbitrary algorithms into incentive compatible mechanisms. The classic result of Vickrey, Clarke and Groves provides a positive result along these lines for social welfare maximization, where the goal is to select an outcome that maximizes the total value to all participants. This result demonstrates that for any social welfare maximization problem there exists a mechanism with very robust incentive properties (namely, it is *ex post incentive compatible*). The construction requires that the mechanism optimize social welfare precisely, and so can be thought of as a reduction from incentive compatible mechanism design to exact algorithm design. In fact the reduction can be extended to more general scenarios beyond social welfare. In the “single-parameter” setting, where the preferences of every selfish agent can be described by a single scalar parameter, (Bayesian or ex post) incentive compatibility is essentially equivalent to a per-agent monotonicity condition on the allocation returned by the mechanism. Therefore, for objective functions that are “monotone” in the sense that exact optimization of the objective leads to a monotone allocation function, there is a reduction from mechanism design to exact algorithm design along the lines of the VCG mechanism for social welfare.

However, many settings of interest involve constraints which are computationally infeasible to optimize precisely, and so exact algorithms are not known to exist. Hartline and Lucier [12] recently showed that in Bayesian settings of partial information, the reduction from mechanism design to algorithm design for social welfare can be extended to encompass arbitrary approximate algorithms with arbitrarily small loss in expected performance. In contrast with the VCG mechanism, Hartline and Lucier’s reduction achieves (weaker) Bayesian incentive compatibility and preserves the algorithm’s performance only in expectation over the randomness in the input; on the positive side, it works for arbitrary algorithms and can be extended to multi-parameter settings [11, 4]. Moreover, the reduction is black-box, meaning that it need not understand the underlying structure of the given algorithm or problem constraints.

In light of these results, two natural questions arise:

- Are there black-box reductions transforming arbitrary algorithms for social welfare into *ex post* incentive compatible mechanisms with little loss in *worst-case* approximation ratio?
- Does *every* monotone objective admit a black-box reduction that transforms arbitrary algorithms into (Bayesian or ex-post) incentive compatible mechanisms with little loss in (expected or worst-case) performance?

In this paper we answer both of these questions in the negative. Our impossibility results apply to the simpler class of single-parameter optimization problems.

Our first result strengthens the demands of the reduction for social welfare beyond those of the aforementioned positive results [12, 11, 4] in two significant ways. First, it requires the stronger solution concept of ex post incentive compatibility, rather than Bayesian incentive compatibility.

Second, it requires that the approximation factor of the original algorithm be preserved in the worst case over all possible inputs, rather than in expectation. We show that there exist *single-parameter* instances for which every black-box transformation from algorithms to mechanisms must degrade the algorithm’s worst-case performance by an unbounded factor—even when given access to a constant factor approximation algorithm, the transformation must some times return a mechanism with an approximation ratio that is polynomial in the problem size. A key component of our construction is a feasibility constraint that specifies which allocations are allowed. The feasibility constraint is unknown and only revealed to the mechanism through access to a (non-truthful) algorithm for the problem. In particular, the mechanism is unable to exploit any nice properties of the feasibility constraint. Our impossibility holds also for randomized transformations that aim to achieve truthfulness in expectation over the random bits used; In this case, we use the usual definition of an algorithm’s approximation ratio—in expectation over the random bits but in worst case over the input.

Our second result shows that there exist monotone objective functions for which any black-box reduction from mechanism design to algorithm design must worsen the algorithm’s performance by a factor that is polynomial in the problem size, even when we only desire Bayesian incentive compatibility with respect to a *known* distribution and measure the algorithm’s performance in expectation over the distribution. In particular, we construct an instance of a single-parameter scheduling problem with multiple jobs and machines, where the goal is to minimize the maximum makespan of the schedule. Once again our impossibility result hinges on an unknown feasibility constraint that specifies which schedules are allowed and that is revealed to the mechanism through the underlying algorithm. While the non-trivial feasibility constraint makes this an unnatural optimization problem, our construction serves to demonstrate that results like those for social welfare cannot exist for arbitrary monotone objectives even under the weakest possible requirements on truthfulness and approximability.

Finally, we ask whether there are objectives between the linear welfare objective and the highly non-linear makespan objective that admit reductions in the style of [11] and others. At a high level, the black-box reductions for social welfare perform “ironing” operations for each agent independently fixing non-monotonicities in the algorithm’s output in a local fashion without hurting the overall social welfare. One property of social welfare that enables such an approach is that it is additive across agents. In our final result we show that even restricting attention to objectives that are additive across agents, for almost any objective other than social welfare no per-agent ironing procedure can simultaneously ensure Bayesian incentive compatibility as well as a bounded loss in performance. The implication for mechanism design is that any successful reduction must take a holistic approach over agents and look very different from those known for social welfare.

Our results and techniques. The existence of a black-box reduction from mechanism design to algorithm design can depend on the objective function we are optimizing, the incentive requirements, as well as whether we are interested in a worst-case or average-case performance guarantee. We distinguish between two kinds of incentive requirements (see formal definitions in Section 2). Bayesian incentive compatibility (BIC) implies that truth-telling forms a Bayes-Nash equilibrium under the assumption that the agents’ value distributions are common knowledge. The stronger notion of ex post incentive compatibility (EPIC), a.k.a. universal truthfulness, implies that every agent maximizes her utility by truth-telling regardless of others’ actions and the mechanism’s coin flips. For randomized mechanisms there is a weaker notion of truthfulness called truthfulness in

expectation (TIE) which implies that every agent maximizes her utility in expectation over the randomness in the mechanism by truth-telling regardless of others’ actions. We further distinguish between social welfare and arbitrary monotone objectives, and between the average performance of the algorithm and its worst case performance.

Table 1 below summarizes our findings as well as known results along these three dimensions. Essentially, we find that there is a dichotomy of settings: some allow for essentially lossless transformations whereas others suffer an unbounded loss in performance.

| Objective: social welfare | | | Objective: arbitrary monotone (e.g. makespan) | | |
|---------------------------|-----------------|-------------------|---|-----------------|-------------------|
| | Avg-case approx | Worst-case approx | | Avg-case approx | Worst-case approx |
| BIC | Yes [12, 4, 11] | ? | BIC | No (Section 4) | No |
| TIE | ? | No (Section 3) | TIE | No | No |

Table 1: A summary of our results on the existence of black-box transformations. A “yes” indicates that a reduction exists and gives an arbitrarily small loss in performance; a “no” indicates that every reduction satisfying incentive constraints suffers an arbitrarily large loss in performance.

One way to establish our impossibility results would be to demonstrate the existence of single-parameter optimization problems for which there is a gap in the approximating power of arbitrary algorithms and incentive compatible algorithms. This is an important open problem which has resisted much effort by the algorithmic mechanism design community, and is beyond the scope of our work. Instead, we focus upon the black-box nature of the reductions with respect to, in particular, the feasibility constraint that they face. In our constructions, the black-box transformation has full knowledge of the given instance including the distribution from which agents’ values are drawn. In addition it is given black-box access to an algorithm that it can query at any input. Given an input value vector, the goal of the transformation is to query the algorithm polynomially many times and return an outcome (allocation) that is nearly as good as the one that the original algorithm returns on the same input and is also consistent across inputs in that it satisfies the monotonicity conditions necessary for incentive compatibility. Since the transformation does not know the feasibility constraint explicitly, it can essentially only output an allocation that it has observed while querying the algorithm at different vectors. This last property is crucial in our constructions and allows us to “hide” good allocations from the transformation.

In order to explain how our constructions work, we first discuss some potential approaches towards black-box reductions. One general approach towards achieving IC, namely a maximal-in-range mechanism, is based on the VCG mechanism: instead of reducing a mechanism to an exact algorithm, it restricts the range of outcomes to a (proper) subset of feasible outcomes, and reduces the mechanism design problem to an exact algorithm design problem over this subset. An appropriate subset of outcomes can be constructed by querying the given algorithm at representative inputs and therefore automatically satisfies the unknown feasibility constraint. Moreover if the range is small enough, then the exact algorithm design problem becomes trivial. The main downside of this approach is that in order to achieve good performance the range of outcomes may need to be prohibitively large. A potential variation on this approach is to exploit the structure in the given instance to construct different (and smaller) ranges for different input vectors in such a way that ... Hartline and Lucier [12] (and follow up work in [11, 4]) employ a very different approach to achieve BIC. Their mechanism looks at the agents’ allocations in aggregate over other agents’ values and uses this aggregate information to construct a map from the agents’ values to themselves;

Then, given an input vector, it looks at the vector that the input is mapped to, and returns the original algorithm’s allocation at that vector. [12] show that it is possible to construct the map in such a way that welfare improves and monotonicity (in a Bayesian sense) is achieved. A main achievement of this approach is that it breaks up the problem of achieving monotonicity over a complex high-dimensional value space into single-dimensional (per-agent) problems. Once again the transformation essentially ignores the feasibility constraint and satisfies it by default by returning allocations that it observes while querying the algorithm.

Our impossibility result for social welfare maximization shows that in order to achieve EPIC the transformation must essentially use an MIR type of approach, and trying to understand the algorithm’s behavior in aggregate in the style of [12] does not work. Furthermore, the underlying algorithm in our construction hides “long-distance” non-monotonicities in such a way that at some value vectors, the transformation must either query the algorithm at exponentially many vectors in order to find the non-monotonicity, or return a default outcome that guarantees monotonicity but has a poor approximation ratio.

For objectives other than social welfare, we show that the per-agent mapping approach of [12] is fundamentally flawed and fails for almost any objective. Specifically for makespan minimization, once again the transformation must try to find “good” allocations by querying the algorithm at exponentially many value vectors, or risk being non-monotone.

All of our constructions involve very simple instances with value spaces that are symmetric across agents—each agent has a high or a low value and gets a high or a low (or medium in one case) allocation;. The complexity of the constructions then lies in the feasibility constraint that is imposed on the mechanism. A weakness of our constructions is that the hidden feasibility constraints that we construct are unnatural (and, for example, asymmetric across agents). This is essential—good mechanisms can be constructed in the absence of a feasibility constraint, or when the constraint is symmetric across agents¹.

Related Work. Reductions from mechanism design to algorithm design in the Bayesian setting were first studied by Hartline and Lucier [12], who showed that any approximation algorithm for a single-parameter social welfare problem can be converted into a Bayesian incentive compatible mechanism with arbitrarily small loss in expected performance. This was extended to multi-parameter settings by Hartline, Kleinberg and Malekian [11] and Bei and Huang [4].

Some reductions from mechanism design to algorithm design are known for prior-free settings, for certain restricted classes of algorithms. Lavi and Swamy [15] consider mechanisms for multi-parameter packing problems and show how to construct a (randomized) β -approximation mechanism that is truthful in expectation, from any β -approximation that verifies an integrality gap. Dughmi, Roughgarden and Yan [10] extend the notion of designing mechanisms based upon randomized rounding algorithms, and obtain truthful in expectation mechanisms for a broad class of submodular combinatorial auctions. Dughmi and Roughgarden [9] give a construction that converts any FPTAS algorithm for a social welfare problem into a mechanism that is truthful in expectation, by way of a variation on smoothed analysis.

Babaioff et al. [3] provide a technique for turning a β -algorithm for a single-valued combinatorial auction problem into a truthful $\beta(\log v_{max}/v_{min})$ -approximation mechanism, when agent values are

¹In particular, when each agent has few different types and the feasibility constraint is symmetric, the following MIR-style transformation works: ...

restricted to lie in $[v_{min}, v_{max}]$. This reduction applies to single-parameter problems with downward-closed feasibility constraints and binary allocations (each agent’s allocation can be either 0 or 1).

Many recent papers have explored limitations on the power of deterministic ex post incentive compatible mechanisms to approximate social welfare, albeit in multi-parameter settings. Papadimitriou, Schapira and Singer [16] gave an example of a social welfare problem for which constant-factor approximation algorithms exist, but any polytime ex post incentive compatible mechanism attains at best a polynomial approximation factor. A similar gap for the submodular combinatorial auction problem was established by Dobzinski [8]. For the general combinatorial auction problem, such gaps have been established for the restricted class of max-in-range mechanisms by Buchfuhrer et al. [5]. However, prior to our work, it was not known whether a lossless black-box reduction could exist for the important special case of single-parameter problems.

For the makespan objective in multi-parameter settings (that is, when the sizes of jobs on different machines are unrelated), Ashlagi et al. [2] showed that ex post incentive compatibility imposes a huge cost: while constant factor approximations can be obtained in the absence of incentive constraints, no “anonymous” mechanism can obtain a sublinear approximation ratio under the requirement of incentive compatibility. The situation for single-parameter settings is quite different. In single-parameter (a.k.a. related) settings, each machine has a single private parameter, namely its speed, and each job has a known intrinsic size; the load that a job places on a machine is its size divided by the speed of the machine. Truthful scheduling to minimize makespan on related machines was studied by Archer and Tardos [1], who designed a truthful-in-expectation 3-approximation. Dhangwatnotai et al. [7] gave a randomized PTAS that is truthful-in-expectation, which was then improved to a deterministic truthful PTAS by Christodoulou and Kovács [6], matching the performance of the best possible approximation algorithm [13]. Our work on makespan minimization differs in that we consider the goal of minimizing makespan subject to an arbitrary feasibility constraint.

A preliminary version of this work [14] proved an impossibility result for EPIC black-box reductions for single-parameter social welfare problems. In this paper we extend that result to apply to (the broader class of) TIE reductions.

2 Preliminaries

Optimization Problems. In a single-parameter real-valued optimization problem we are given an input vector $\mathbf{v} = (v_1, v_2, \dots, v_n)$. Each v_i is assumed to be drawn from a known set $V_i \subseteq \mathbb{R}$, so that $V = V_1 \times \dots \times V_n$ is the set of possible input vectors. The goal is to choose some allocation $\mathbf{x} \in \mathcal{F} \subseteq \mathbb{R}^n$ from among a set of feasible allocations \mathcal{F} such that a given objective function $\phi : \mathcal{F} \times V \rightarrow \mathbb{R}$ is optimized (i.e. either maximized or minimized, depending on the nature of the problem). We think of the feasibility set \mathcal{F} and the objective function ϕ as defining an instance of the optimization problem. We will write $\mathbf{x} = (x_1, x_2, \dots, x_n)$, where each $x_i \in \mathbb{R}$.

An algorithm \mathcal{A} defines a mapping from input vectors \mathbf{v} to outcomes \mathbf{x} . We will write $\mathcal{A}(\mathbf{v})$ for the allocation returned by \mathcal{A} as well as the value it obtains; the intended meaning should be clear from the context. In general an algorithm can be randomized, in which case $\mathcal{A}(\mathbf{v})$ is a random variable.

Given an instance \mathcal{F} of the social welfare problem, we will write $OPT_{\mathcal{F}}(\mathbf{v})$ for the allocation in \mathcal{F} that maximizes $\phi(\mathbf{x}, \mathbf{v})$, as well as the value it obtains. Given algorithm \mathcal{A} , let $approx_{\mathcal{F}}(\mathcal{A})$ denote the worst-case approximation ratio of \mathcal{A} for problem \mathcal{F} . That is, $approx_{\mathcal{F}}(\mathcal{A}) = \min_{\mathbf{v} \in V} \frac{\mathcal{A}(\mathbf{v})}{OPT_{\mathcal{F}}(\mathbf{v})}$

for a maximization problem; here ϕ is implicit and should be clear from the context. Note that $\text{approx}_{\mathcal{F}}(\mathcal{A}) \leq 1$ for all \mathcal{F} and \mathcal{A} .

We also consider a Bayesian version of our optimization problem, in which there is publicly-known product distribution \mathbf{F} on input vectors. That is, $\mathbf{F} = F_1 \times \dots \times F_n$ and each v_i is distributed according to F_i . Given \mathbf{F} , the expected objective value of a given algorithm \mathcal{A} is given by $\bar{\phi}(\mathcal{A}) = \mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[\phi(\mathcal{A}(\mathbf{v}), \mathbf{v})]$. The goal of the optimization problem in this setting is to optimize the expected objective value.

Mechanisms. We will consider our optimization problems in a mechanism design setting with n rational agents, where each agent possesses one value from the input vector as private information. We think of an outcome \mathbf{x} as representing an *allocation* to the agents, where x_i is the allocation to agent i . A (direct-revelation) mechanism for our optimization problem then proceeds by eliciting declared values $\mathbf{b} \in \mathbb{R}^n$ from the agents, then applying an allocation algorithm $\mathcal{A} : \mathbb{R}^n \rightarrow \mathcal{F}$ that maps \mathbf{b} to an allocation \mathbf{x} , and a payment rule that maps \mathbf{b} to a payment vector \mathbf{p} . We will write $\mathbf{x}(\mathbf{b})$ and $\mathbf{p}(\mathbf{b})$ for the allocations and payments that result on input \mathbf{b} . The *utility* of agent i , given that the agents declare \mathbf{b} and his true private value is v_i , is taken to be $v_i x_i(\mathbf{b}) - p_i(\mathbf{b})$.

A (possibly randomized) mechanism is *truthful in expectation* (TIE) if each agent maximizes its expected utility by reporting its value truthfully, regardless of the reports of the other agents, where expectation is taken over any randomness in the mechanism. That is, $\mathbf{E}[v_i x_i(v_i, \mathbf{b}_{-i}) - p_i(v_i, \mathbf{b}_{-i})] \geq \mathbf{E}[v_i x_i(b_i, \mathbf{b}_{-i}) - p_i(b_i, \mathbf{b}_{-i})]$ for all i , all $v_i, b_i \in V_i$, and all $\mathbf{b}_{-i} \in V_{-i}$. We say that an algorithm is TIE if there exists a payment rule such that the resulting mechanism is TIE. It is known that an algorithm is TIE if and only if, for all i and all \mathbf{v}_{-i} , $\mathbf{E}[x_i(v_i, \mathbf{v}_{-i})]$ is monotone non-decreasing as a function of v_i , where the expectation is over the randomness in the mechanism.

We say that a (possibly randomized) mechanism is *Bayesian incentive compatible* (BIC) for distribution \mathbf{F} if each agent maximizes its expected utility by reporting its value truthfully, given that the other agents' values are distributed according to \mathbf{F} (and given any randomness in the mechanism). That is, $\mathbf{E}_{\mathbf{v}_{-i}}[v_i x_i(v_i, \mathbf{v}_{-i}) - p_i(v_i, \mathbf{v}_{-i})] \geq \mathbf{E}_{\mathbf{v}_{-i}}[v_i x_i(b_i, \mathbf{v}_{-i}) - p_i(b_i, \mathbf{v}_{-i})]$ for all i and all $v_i, b_i \in V_i$, where the expectation is over the distribution of others' values and the randomness in the mechanism. We say that an algorithm is BIC if there exists a payment rule such that the resulting mechanism is BIC. It is known that an algorithm is BIC if and only if, for all i , $\mathbf{E}_{\mathbf{v}_{-i}}[x_i(v_i, \mathbf{v}_{-i})]$ is monotone non-decreasing as a function of v_i .

Transformations. A *polytime transformation* \mathcal{T} is an algorithm that is given black-box access to an algorithm \mathcal{A} . We will write $\mathcal{T}(\mathcal{A}, \mathbf{v})$ for the allocation returned by \mathcal{T} on input \mathbf{v} , given that its black-box access is to algorithm \mathcal{A} . Then, for any \mathcal{A} , we can think of $\mathcal{T}(\mathcal{A}, \cdot)$ as an algorithm that maps value vectors to allocations; we think of this as the algorithm \mathcal{A} *transformed by* \mathcal{T} . We write $\mathcal{T}(\mathcal{A})$ for the allocation rule that results when \mathcal{A} is transformed by \mathcal{T} . Note that \mathcal{T} is not parameterized by \mathcal{F} ; informally speaking, \mathcal{T} has no knowledge of the feasibility constraint \mathcal{F} being optimized by a given algorithm \mathcal{A} . However, we do assume that \mathcal{T} is aware of the objective function ϕ , the domain V_i of values for each agent i , and (in Bayesian settings) the distribution \mathbf{F} over values.

We say that a transformation \mathcal{T} is truthful in expectation (TIE) if, for all \mathcal{A} , $\mathcal{T}(\mathcal{A})$ is a TIE algorithm. In a Bayesian setting with distribution \mathbf{F} , we say that transformation \mathcal{T} is Bayesian incentive compatible (BIC) for \mathbf{F} if, for all \mathcal{A} , $\mathcal{T}(\mathcal{A})$ is a BIC algorithm. Note that whether or not \mathcal{T} is TIE or BIC is independent of the objective function ϕ and feasibility constraint \mathcal{F} .

3 A Lower Bound for TIE Transformations for social welfare

In this section we consider the problem of maximizing social welfare. For this problem, BIC transformations that approximately preserve expected performance are known to exist. We prove that if we strengthen our solution concept to truthfulness in expectation and our performance metric to worst-case approximation, then such black-box transformations are not possible.

3.1 Problem definition and main theorem

The social welfare objective is defined as $\phi(\mathbf{x}, \mathbf{v}) = \mathbf{v} \cdot \mathbf{x}$.

Our main result is that, for any TIE transformation \mathcal{T} , there is a problem instance \mathcal{F} and algorithm \mathcal{A} such that \mathcal{T} degrades the worst-case performance of \mathcal{A} by a polynomially large factor.

Theorem 3.1. *There is a constant $c > 0$ such that, for any polytime TIE transformation \mathcal{T} , there is an algorithm \mathcal{A} and problem instance \mathcal{F} such that $\frac{\text{approx}_{\mathcal{F}}(\mathcal{A})}{\text{approx}_{\mathcal{F}}(\mathcal{T}(\mathcal{A}))} \geq n^c$.*

The high-level idea behind our proof of Theorem 3.1 is as follows. We will construct an algorithm \mathcal{A} and input vectors \mathbf{v} and \mathbf{v}' such that, for each agent i in some large subset of the players, $v_i' > v_i$ but $\mathcal{A}_i(\mathbf{v}') < \mathcal{A}_i(\mathbf{v})$. This does not immediately imply that \mathcal{A} is non-truthful, but we will show that it does imply non-truthfulness under a certain feasibility condition \mathcal{F} , namely that any allocation is constant on the players i with $v_i' > v_i$. Thus, any TIE transformation \mathcal{T} must alter the allocation of \mathcal{A} either on input \mathbf{v} or on input \mathbf{v}' . However, we will craft our algorithm in such a way that, on input \mathbf{v} , the only allocations that the transformation will observe given polynomially many queries of \mathcal{A} will be $\mathcal{A}(\mathbf{v})$, plus allocations that have significantly worse social welfare than $\mathcal{A}(\mathbf{v})$, with high probability. Similarly, on input \mathbf{v}' , with high probability the transformation will only observe allocation $\mathcal{A}(\mathbf{v}')$ plus allocations that have significantly worse social welfare than $\mathcal{A}(\mathbf{v}')$. Furthermore, we ensure that the transformation can not even find the magnitude of the allocation to players i in v' when presented with input v , thereby preventing the transformation from randomizing between the high allocation of $\mathcal{A}(\mathbf{v})$ and an essentially empty allocation to simulate the $\mathcal{A}(\mathbf{v}')$ allocation without directly observing it. Instead, in order to guarantee that it generates an TIE allocation rule, the transformation will be forced to assume the worst-case and offer players i the smallest possible allocation on input \mathbf{v} . This significantly worsens the worst-case performance of the algorithm \mathcal{A} .

3.2 Construction

In the instances we consider, each private value v_i is chosen from $\{v, 1\}$, where $0 < v < 1$ is a parameter that we set below. That is, we will set $V_i = \{v, 1\}$ for all $i \in [n]$. We can therefore interpret an input vector as a subset $y \subseteq [n]$, corresponding to those agents with value 1 (the remaining agents have value v). Accordingly we define $\mathcal{A}(y)$, $OPT_{\mathcal{F}}(y)$, etc., for a given subset $y \subseteq [n]$. Also, for $a \geq 0$ and $y \subseteq [n]$, we will write \mathbf{x}_y^a for the allocation in which each agent $i \in y$ is allocated a , and each agent $i \notin y$ is allocated 0.

Feasible Allocations. We now define a family of feasibility constraints. Roughly speaking, we will choose $\alpha, \gamma \in (0, 1)$ with $\gamma < \alpha$ and sets $S, T \subseteq [n]$ of agents. The feasible allocations will be $\mathbf{x}_{[n]}^\gamma$, \mathbf{x}_S^1 , and \mathbf{x}_T^α . That is, we can allocate γ to every agent, 1 to all agents in S , or α to all agents

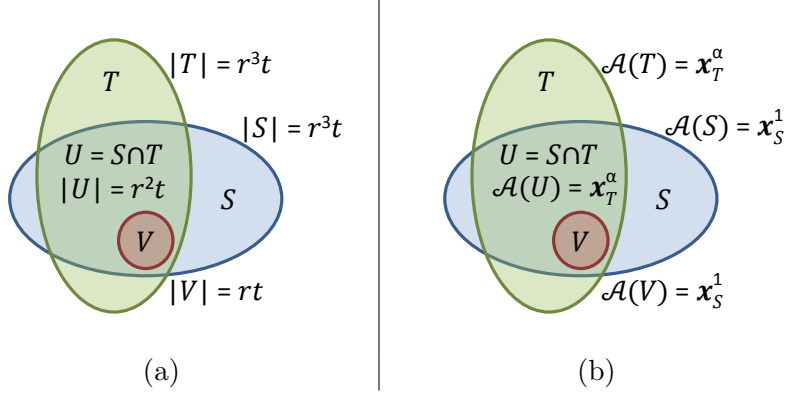


Figure 1: (a) Visualization of typical admissible sets of bidders V , S , and T , given size parameters r and t , and (b) the corresponding allocations of algorithm $\mathcal{A} = \mathcal{A}_{V,S,T,\alpha}$.

in T . We will also require that S and T satisfy certain properties, which essentially state that S and T are sufficiently large and have a sufficiently large intersection.

More formally, define parameters $\gamma \in (0, 1)$, $\alpha \in (0, 1)$, $r \geq 1$, and $t \geq 1$ (which we will fix later to be functions of n), such that $t \gg r \gg \gamma^{-1} \gg \alpha^{-1}$, $r^5 t \leq n$, and $\frac{t}{\gamma^n} \ll 1$. We think of t as a bound on the size of “small” sets, and we think of r as a ratio between the sizes of “small” and “large” sets.

Suppose that V , S , and T are subsets of $[n]$. We say that the triple V, S, T is *admissible* if the following conditions hold:

1. $|S| = |T| = r^3 t$,
2. $|S \cap T| = r^2 t$,
3. $V \subset S \cap T$, and,
4. $|V| = r t$.

In general, for a given admissible V , S , and T , we will tend to write $U = S \cap T$ for notational convenience. See Figure 1(a) for an illustration of the relationship between the sets in an admissible triple. In order to hide the feasibility constraint \mathcal{F} from the transformation, we will pick the sets V , S , and T uniformly at random from all admissible triples, and the value α from an appropriate range. For each admissible tuple V, S, T , and value α , we define a corresponding feasibility constraint

$$\mathcal{F}_{V,S,T,\alpha} = \{x_S^1, x_T^\alpha, x_{[n]}^\gamma\}.$$

Note that $\mathcal{F}_{V,S,T,\alpha}$ does not depend on V ; we include set V purely for notational convenience. We remark that all of the feasible allocations allocate the same amount to agents in U .

Recall that agents have values chosen from $\{v, 1\}$. We will choose $v = \frac{t}{\gamma^n}$, where we recall that our parameters have been chosen so that $\frac{t}{\gamma^n} \ll 1$.

The Algorithm. We now define the algorithm $\mathcal{A}_{V,S,T,\alpha}$ corresponding to an admissible tuple V, S, T and value α . We think of $\mathcal{A}_{V,S,T,\alpha}$ as an approximation algorithm for the social welfare

problem $\mathcal{F}_{V,S,T,\alpha}$ and later show that there is no TIE transformation of $\mathcal{A}_{V,S,T,\alpha}$ without a significant loss in worst-case approximation for some value of α .

Given $y \subset [n]$, we define

$$n_T(y) = |y \cap T| + |y \cap U|$$

and

$$n_S(y) = |y \cap S| + 2|y \cap V|.$$

That is, $n_T(y)$ is the number of elements of y that lie in T , with elements of U counted twice. Likewise, $n_S(y)$ is the number of elements of y that lie in S , with elements of V counted thrice.

The algorithm $\mathcal{A}_{V,S,T,\alpha}$ is then described as Algorithm 1.

Algorithm 1: Allocation Algorithm $\mathcal{A}_{V,S,T,\alpha}$

Input: Subset $y \in [n]$ of agents with value 1

Output: An allocation $\mathbf{x} \in \mathcal{F}_{V,S,T,\alpha}$

```

1 if  $n_S(y) \geq t$ ,  $n_S(y) \geq \gamma|y|$ , and  $n_S(y) \geq n_T(y)$  then
2   | return  $\mathbf{x}_S^1$ 
3 else if  $n_T(y) \geq t$ ,  $n_T(y) \geq \gamma|y|$ , and  $n_T(y) \geq n_S(y)$  then
4   | return  $\mathbf{x}_T^\alpha$ 
5 else
6   | return  $\mathbf{x}_{[n]}^\gamma$ 
7 end

```

3.3 Analysis

In this section, we derive the key lemmas for the proof of Theorem 3.1. First, we bound the approximation factor of algorithm $\mathcal{A}_{V,S,T,\alpha}$ for problem $\mathcal{F}_{V,S,T,\alpha}$.

Lemma 3.2. $\text{approx}_{\mathcal{F}_{V,S,T,\alpha}}(\mathcal{A}_{V,S,T,\alpha}) \geq \alpha/6$.

Proof. Choose $y \subseteq [n]$ and consider the three cases for the output of $\mathcal{A}_{V,S,T,\alpha}$.

Case 1: $n_S(y) \geq t$, $n_S(y) \geq \gamma|y|$, **and** $n_S(y) \geq n_T(y)$. Our algorithm returns allocation \mathbf{x}_S^1 and obtains welfare at least $|y \cap S|$. Note that

$$|y \cap S| \geq \frac{1}{3}n_S(y) \geq \frac{1}{3}t$$

and

$$|y \cap S| \geq \frac{1}{3}n_S(y) \geq \frac{1}{3}n_T(y) \geq \frac{1}{3}|y \cap T|.$$

The allocation \mathbf{x}_T^α obtains welfare at most $\alpha(|y \cap T| + |T \setminus y|v) \leq \alpha(|y \cap T| + nv\gamma) \leq |y \cap T| + t \leq 6|y \cap S|$. Note that here we used $|T| \leq n\gamma$, which follows since $r > \gamma^{-1}$.

The allocation $\mathbf{x}_{[n]}^\gamma$ obtains welfare at most $\gamma|y| + t \leq 2n_S(y) \leq 6|y \cap S|$. So we obtain at least a 1/6-approximation in this case.

Case 2: $n_T(y) \geq t$, $n_T(y) \geq \gamma|y|$, **and** $n_T(y) \geq n_S(y)$. Our algorithm returns allocation \mathbf{x}_T^α and obtains welfare at least $\alpha|y \cap T|$. The same argument as case 1 shows that our approximation factor is at least $\alpha/6$ in this case.

Case 3: $n_S(y) \leq t$ and $n_T(y) \leq t$. Our algorithm returns allocation $\mathbf{x}_{[n]}^\gamma$ for a welfare of at least $\gamma(|y| + v(n - |y|)) \geq t$. The allocation \mathbf{x}_S^1 obtains welfare at most $|y \cap S| + t \leq n_S(y) + t \leq 2t$, and allocation \mathbf{x}_T^α obtains welfare at most $2\alpha t \leq 2t$. So our approximation factor is at least $1/2$ in this case.

Case 4: $n_S(y) \leq \gamma|y|$ and $n_T(y) \leq \gamma|y|$. Our algorithm returns allocation $\mathbf{x}_{[n]}^\gamma$ for a welfare of at least $\gamma(|y| + v(n - |y|)) \geq \gamma|y|$. The allocation \mathbf{x}_S^1 obtains welfare at most $|y \cap S| + t \leq 2n_S(y) \leq 2\gamma|y|$, and allocation \mathbf{x}_T^α obtains welfare at most $|y \cap T| + t \leq 2\alpha\gamma|y| \leq 2\gamma|y|$. So our approximation factor is at least $1/2$ in this case. \square

Suppose now that \mathcal{A}' is any algorithm for problem $\mathcal{F}_{V,S,T,\alpha}$ that is TIE. We will show that \mathcal{A}' is then very restricted in the allocations it can return on inputs $y = V$ and $y = U$. Furthermore, we note that if \mathcal{A}' has a good enough approximation ratio, then its allocations on inputs $y = V$ and $y = U$ are restricted further still. In particular, the optimal allocation on both V and U is x_S^1 ; so to obtain a good approximation factor, on both U and V , the algorithm should allocate a large enough amount to agents in U . As any TIE transformation of \mathcal{A} is itself an algorithm for problem $\mathcal{F}_{V,S,T,\alpha}$, these observations will later play a key role in our impossibility result.

Claim 3.3. *Suppose \mathcal{A}' is a truthful-in-expectation algorithm for problem $\mathcal{F}_{V,S,T,\alpha}$. Then the expected allocation to each agent in U must be at least as large in $\mathcal{A}'(U)$ as in $\mathcal{A}'(V)$.*

Proof. Take any set W with $V \subseteq W \subseteq U$, $|W| = |V| + 1$. Then, on input W , the expected allocation to the agent in $W \setminus V$ must not decrease. Since all allocations are constant on U , this means that the expected allocation to each agent in U must not decrease. By the same argument, \mathcal{A}' returns an allocation at least this large for all W such that $V \subseteq W \subseteq U$, and in particular for $W = U$. \square

In light of these claims, our strategy for proving Theorem 3.1 will be to show that a polytime transformation \mathcal{T} is unlikely to encounter the allocation \mathbf{x}_T^α during its sampling when the input is V , given that the sets V , S , and T are chosen uniformly at random over all admissible tuples. This means the transformation will be unable to learn the value of α . This is key since it prevents the transformation from using the value of α to appropriately randomize between the allocation of \mathbf{x}_S^1 and the essentially empty allocation of $\mathbf{x}_{[n]}^\gamma$ to achieve an effective allocation of α for agents in U on input V thereby satisfying the conditions of Claim 3.3. Similarly, a transformation is unlikely to encounter the allocation \mathbf{x}_S^1 during its sampling on input U , and therefore can not satisfy Claim 3.3 by allocating 1 to agents in U on input U .

Lemma 3.4. *Fix V and S satisfying the requirements of admissibility. Then for any $y \subseteq [n]$, $\Pr_T[\mathcal{A}_{V,S,T,\alpha}(y) = x_T^\alpha] \leq e^{-O(\frac{t}{r+1})}$, with probability taken over all choices of T that are admissible given V and S .*

Proof. Fix any y . Write $n_V = |y \cap V|$, $n_S = |y \cap (S - V)|$, and $n_* = |y \cap ([n] - S)|$. Note that $|y| = n_V + n_S + n_*$. Define the random variables m_U and m_T by $m_U = |y \cap (U - V)|$ and $m_T = |y \cap (T \setminus S)|$.

The event $[\mathcal{A}_{V,S,T,\alpha}(y) = x_T^\alpha]$ occurs precisely if the following are true:

$$m_T + 2n_V + 2m_U \geq t, \tag{1}$$

$$m_T + 2n_V + 2m_U \geq \gamma(n_V + n_S + n_*), \tag{2}$$

$$m_T + 2n_V + 2m_U \geq n_S + 3n_V. \quad (3)$$

We will show that the probability of these three inequalities being true is exponentially small. To see this, note that (3) implies that $m_T + 2m_U \geq n_V$. Thus, (1) implies that $m_T + 2m_U \geq t/3$, and hence $m_T + m_U \geq t/6$. Now each element of y counted in n_S will count toward m_U with probability $\frac{1}{r+1}$, and each element of y counted in n_* will count toward m_T with probability $\frac{1}{r+1}$. Since $t \gg r$, Chernoff bounds imply that with probability at least $1 - e^{-O(t/r)}$, we will have $n_* + n_S \geq \frac{r}{2}(m_T + m_U)$. Then

$$\frac{m_T + 2n_V + 2m_U}{n_V + n_S + n_*} < \frac{3(m_T + 2m_U)}{n_S + n_*} < \frac{12}{r} \ll \gamma$$

contradicting (2). □

Lemma 3.5. *Fix U and T satisfying the requirements of admissibility. Then for any $y \subseteq [n]$, $\Pr_S[\mathcal{A}_{V,S,T,\alpha}(y) = x_S^1] \leq e^{-O(\frac{t}{r+1})}$, with probability taken over all choices of V and S that are admissible given U and T .*

Proof. Fix any y . Write $n_U = |y \cap U|$, $n_T = |y \cap (T - U)|$, and $n_* = |y \cap ([n] - T)|$. Note that $|y| = n_U + n_T + n_*$. Define the random variables m_V and m_S by $m_V = |y \cap V|$ and $m_S = |y \cap (S \setminus T)|$.

The event $[\mathcal{A}_{V,S,T,\alpha}(y) = x_S^1]$ occurs precisely if the following are true:

$$m_S + n_U + 2m_V \geq t, \quad (4)$$

$$m_S + n_U + 2m_V \geq \gamma(n_U + n_T + n_*), \quad (5)$$

$$m_S + n_U + 2m_V \geq n_T + 2n_U. \quad (6)$$

We will show that the probability of these three inequalities being true is exponentially small. To see this, first note that we can assume $n_T = 0$, as this only loosens the requirements of the inequalities. We then have that (6) implies $m_S + 2m_V \geq n_U$. Thus, (4) implies that $m_S + 2m_V \geq t/2$, and hence $m_S + m_V \geq t/4$. Now each element of y counted in n_U will count toward m_V with probability $\frac{1}{r}$, and each element of y counted in n_* will count toward m_S with probability $\frac{1}{r+1}$. Since $t \gg r$, Chernoff bounds imply that with probability at least $1 - e^{-O(t/r)}$, we will have $n_* + n_T \geq \frac{r}{2}(m_S + m_V)$. Then

$$\frac{m_S + n_U + 2m_V}{n_U + n_*} < \frac{4(m_S + m_V)}{n_U + n_*} < \frac{8}{r} \ll \gamma$$

contradicting (5). □

3.4 Proof of Main Theorem

We can now set our parameters t , r , α , and γ . We will choose $t = n^{1/5}$, $r = n^{3/20}$, and $\gamma = n^{-2/20}$. The values of α we will be considering are 1 and $n^{-1/20}$. Note that $t \gg r \gg \gamma^{-1} \gg \alpha^{-1}$ for each choice of α . Note also that $v = t\gamma^{-1}/n = n^{-14/20} \ll 1$.

Our idea now for proving Theorem 3.1 is that since the transformation can not determine the value of α on input V (by Lemma 3.4), and since it can not find the “good” allocation of \mathbf{x}_S^1 on input U (by Lemma 3.5), it must be pessimistic and allocate the minimum possible value of α to

agents in V on input V in order to guarantee that the resulting allocation rule is TIE (by Claim 3.3). This implies a bad approximation on input V and hence a bad worst-case approximation.

Proof of Theorem 3.1 : For each admissible V, S, T and $\alpha \in \{1, n^{-1/20}\}$, write $\mathcal{A}'_{V,S,T,\alpha}$ for $\mathcal{T}(\mathcal{A}_{V,S,T,\alpha})$. Lemma 3.5 implies that, with all but exponentially small probability, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation \mathbf{x}_S^1 on input U . Thus, on input U , it can allocate at most α to each agent in U in expectation (using the fact that $\alpha > \gamma$). Then, since $\mathcal{A}'_{V,S,T,\alpha}$ is incentive compatible, Claim 3.3 implies that $\mathcal{A}'_{V,S,T,\alpha}$ must allocate at most α to each agent in U on input V .

Now Lemma 3.4 implies that, with all but exponentially small probability, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation \mathbf{x}_T^α on input V , and thus is unaware of the value of α on input V . Thus, to ensure incentive compatibility, $\mathcal{A}'_{V,S,T,1}(V)$ must allocate at most $n^{-1/20}$ to each agent in U . It therefore obtains a welfare of $|V|n^{-1/20} + t \leq n^{1/5} + n^{3/10} < n^{6/20}$, whereas a total of $|V| = n^{7/20}$ is possible with allocation \mathbf{x}_S^1 . Thus $\mathcal{A}'_{V,S,T,1}$ has a worst-case approximation of $n^{-1/20}$, whereas $\mathcal{A}_{V,S,T,1}$ has an approximation factor of $1/6$. \square

We conclude with a remark about extending our impossibility result to TIE transformations under the weaker goal of preserving the expected social welfare under a given distribution \mathbf{F} . We would like to prove that, when agents' values are drawn according to a distribution \mathbf{F} , any TIE transformation necessarily degrades the average welfare of some algorithm by a large factor. The difficulty with extending our techniques to this setting is that a transformation may use the distribution \mathbf{F} to “guess” the relevant sets V and U (i.e. if the distribution is concentrated around the sets V and U in our construction). One might hope to overcome this difficulty in our construction by hiding a “true” set V (that generates a non-monotonicity) in a large sea of sets that could potentially take the role of V . Then, if the transformation is unlikely to find a good allocation on input U , and unlikely to determine the value of α on any of these potential sets, and is further unable to determine which set is the “true” V , then it must be pessimistic and allocate the minimum potential value of α on any of these potential sets in order to guarantee truthfulness. Unfortunately, our construction assumes that all allocations are constant on U , and this makes it difficult to hide a set V while simultaneously making it difficult to discover a good allocation on input U . We feel that, in order to make progress on this interesting open question, it is necessary to remove the assumption that all allocations are constant on U which, in hand, seems to make it much more difficult to derive necessary conditions for a transformation to be TIE.

4 An Impossibility Result for Makespan

We now consider an objective function, namely makespan, that differs from the social welfare objective in that it is not linear in agent values or allocations. Informally we show that black-box reductions for approximation algorithms for makespan are not possible even if we relax the notion of truthfulness to Bayesian incentive compatibility and relax the measure of performance to expected makespan, where both the notions are defined with respect to a certain *fixed* and *known* distribution over values. As in the previous section, our impossibility result hinges on the fact that the transformation is not aware of the feasibility constraint that an allocation needs to satisfy and can learn this constraint only by querying the algorithm at different inputs.

4.1 Problem definition and main theorem

We consider the following minimization problem in a Bayesian setting. In this problem n selfish machines (a.k.a. agents) are being allocated jobs. Each agent has a private value v_i representing its speed. If machine i is allocated jobs with a total length x_i , the load of machine i is x_i/v_i . The makespan of allocation \mathbf{x} to machines with speeds \mathbf{v} is the maximum load of any machine:

$$\phi(\mathbf{x}, \mathbf{v}) = \max_i \frac{x_i}{v_i}.$$

An instance of the (Bayesian) makespan problem is given by a feasibility constraint \mathcal{F} and a distribution over values \mathbf{F} ; the goal is to map every value vector to allocations so as to minimize the expected makespan:

$$\mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[\phi(\mathbf{x}(\mathbf{v}), \mathbf{v})] \text{ subject to } \mathbf{x}(\mathbf{v}) \in \mathcal{F} \text{ for all } \mathbf{v}.$$

Given an algorithm \mathcal{A} , we use $\bar{\phi}(\mathcal{A})$ to denote its expected makespan.

Our main result is the following:

Theorem 4.1. *Let n be large enough and \mathcal{T} be any black-box BIC transformation that makes at most $e^{n^{1/4}/2}$ black-box queries to the given algorithm on each input. There exists an instance $(\mathcal{F}, \mathbf{F})$ of the makespan problem and a deterministic algorithm \mathcal{A} such that $\mathcal{T}(\mathcal{A})$ either returns an infeasible allocation with positive probability, or has makespan $\bar{\phi}(\mathcal{T}(\mathcal{A})) = \Omega(n^{1/4})\bar{\phi}(\mathcal{A})$. Here \mathbf{F} is the uniform distribution over $\{1, \alpha\}^n$ for an appropriate α and is known to \mathcal{T} .*

We note that the algorithm \mathcal{A} in the statement of Theorem 4.1 is deterministic. A BIC transformation \mathcal{T} must therefore degrade the makespan of some algorithms by a polynomially large factor even when we limit ourselves to deterministic algorithms. For simplicity of exposition, we prove a gap of $\Omega(n^{1/4})$, however, our construction can be tweaked to obtain a gap of $\Omega(n^{1/2-\delta})$ for any $\delta > 0$.

Problem Instance. We now describe the problem instance $(\mathcal{F}, \mathbf{F})$ in more detail. Let $\alpha < n^{1/2}$ be a parameter to be determined later. As mentioned earlier, \mathbf{F} is the uniform distribution over $\{1, \alpha\}^n$. That is, every value (i.e. speed) v_i is 1 or α with equal probability. There are $2n$ jobs in all, n of length α and n of length 1. Our feasibility constraint will have the property that each machine can be assigned at most one job. So a valid allocation will set the allocation to each machine to a value in $\{0, 1, \alpha\}$.²

Of all such allocations (i.e. all $\mathbf{x} \in \{0, 1, \alpha\}^n$), all but one will be feasible. This one infeasible allocation is thought of as a parameter of the problem instance. Given $\mathbf{x} \in \{0, 1, \alpha\}^n$, we will write $\mathcal{F}_{\mathbf{x}}$ as the set $\{0, 1, \alpha\}^n \setminus \mathbf{x}$, and $\Gamma(\mathbf{x}) = (\mathcal{F}_{\mathbf{x}}, \mathbf{F})$ as the corresponding problem instance in which \mathbf{x} is infeasible. We will use \mathbf{x}_{bad} to denote the forbidden allocation in the remainder of this section.

The algorithm. We will first describe a randomized algorithm $\mathcal{A}(\mathbf{x}_{bad})$ (Algorithm 2 below) which we think of as an approximation algorithm for problem instance $\Gamma(\mathbf{x}_{bad})$.

We first note that $\mathcal{A}(\mathbf{x}_{bad})$ must terminate.

²A makespan assignment must allocate each job to a machine, but we will sometimes wish to specify an allocation in which not all jobs are allocated. For ease of exposition, we will therefore assume that there is an extra agent with value $n(\alpha + 1)$; this agent will always be allocated all jobs not allocated to any other agent. Note that the load of this machine is always at most 1.

Algorithm 2: Allocation Algorithm $\mathcal{A}(\mathbf{x}_{bad})$

Input: Vector $\mathbf{v} \in \{1, \alpha\}^n$ **Output:** An allocation $\mathbf{x} \neq \mathbf{x}_{bad}$

```
1  $H \leftarrow \{i : v_i = \alpha\};$ 
2 if  $\frac{1}{2}n - n^{3/4} \leq |H| \leq \frac{1}{2}n + n^{3/4}$  then
3   | Choose set  $S \subset H$  with  $|S| = n^{-1/2}|H|$  uniformly at random;
4   | for  $i \in S$  do  $x_i \leftarrow \alpha;$ 
5   | for  $i \in H \setminus S$  do  $x_i \leftarrow 0;$ 
6   | for  $i \notin H$  do  $x_i \leftarrow 1;$ 
7 else
8   | Choose set  $S \subset [n]$  with  $|S| = n^{3/4}$  uniformly at random;
9   | for  $i \in S$  do  $x_i \leftarrow \alpha;$ 
10  | for  $i \notin S$  do  $x_i \leftarrow 0;$ 
11 end
12 if  $\mathbf{x} = \mathbf{x}_{bad}$  then go to line 1;
13 return  $\mathbf{x}$ 
```

Claim 4.2. For all \mathbf{v} , algorithm $\mathcal{A}(\mathbf{x}_{bad})$ terminates with probability 1.

Proof. This follows from noting that at least two distinct allocations can be chosen by $\mathcal{A}(\mathbf{x}_{bad})$ on each branch of the condition on line 2, so $\mathcal{A}(\mathbf{x}_{bad})$ must eventually choose an allocation that is not \mathbf{x}_{bad} . \square

We now use $\mathcal{A}(\mathbf{x}_{bad})$ to define a set of deterministic algorithms³. Let $D(\mathbf{x}_{bad})$ (or D for short) denote the set of deterministic algorithms in the support of $\mathcal{A}(\mathbf{x}_{bad})$. That is, for every $\mathcal{A} \in D$, $\mathcal{A}(\mathbf{v})$ is an allocation returned by $\mathcal{A}(\mathbf{x}_{bad})$ on input \mathbf{v} with positive probability for every $\mathbf{v} \in \{1, \alpha\}^n$. Moreover, for every combination of allocations that can be returned by $\mathcal{A}(\mathbf{x}_{bad})$ on each input profile, there is a corresponding deterministic algorithm in D .

For any \mathbf{v} , let $H(\mathbf{v}) = \{i : v_i = \alpha\}$ be the set of high speed agents. Let C denote the event that $\frac{1}{2}n - n^{3/4} \leq |H(\mathbf{v})| \leq \frac{1}{2}n + n^{3/4}$, over randomness in \mathbf{v} . We think of C as the event that the number of high-speed agents is concentrated around its expectation. We note the following immediate consequence of Chernoff bounds.

Observation 4.3. $\Pr_{\mathbf{v}}[C] \geq 1 - 2e^{-n^{1/4}/4}$.

This allows us to bound the expected makespan of each $\mathcal{A} \in D$.

Lemma 4.4. For each $\mathcal{A} \in D$, $\bar{\phi}(\mathcal{A}) \leq 1 + 2\alpha e^{-n^{1/4}/4}$ where the expectation is taken over \mathbf{v} .

Proof. If event C occurs, then \mathcal{A} returns an allocation in which each agent i with $v_i = 1$ receives allocation at most 1. Since each agent with $v_i = \alpha$ also receives allocation at most α , we conclude that if event C occurs then the makespan of $\mathcal{A}(\mathbf{v})$ is at most 1. Otherwise, the makespan of $\mathcal{A}(\mathbf{v})$ is trivially bounded by α . Since Observation 4.3 implies that this latter case occurs with probability at most $2e^{-n^{1/4}/4}$, the result follows. \square

³More precisely, we will define a set of deterministic allocation rules that map type profiles to allocations; in particular we will not be concerned with implementations of these allocation rules.

4.2 Transformation Analysis

We now present a proof of Theorem 4.1. Let \mathcal{T} denote a BIC transformation that can make at most $e^{n^{1/4}/2}$ black-box queries to an algorithm for makespan. Write $\mathcal{T}(\mathcal{A})$ for the mechanism induced when \mathcal{T} is given black-box access to an algorithm \mathcal{A} .

We first note that if $\mathcal{T}(\mathcal{A})$ returns only feasible allocations with probability 1, then it can only return an allocation that it observed during black-box queries to algorithm \mathcal{A} . This is true even if we consider only algorithms of the form $\mathcal{A} \in D(\mathbf{x}_{bad})$ for some choice of \mathbf{x}_{bad} .

Claim 4.5. *Suppose that for some $\mathcal{A} \in D$, with positive probability, \mathcal{T} returns an allocation not returned by a black-box query to \mathcal{A} . Then there exists an algorithm \mathcal{A}' such that $\mathcal{T}_{\mathcal{A}'}$ returns an infeasible allocation with positive probability.*

Proof. Suppose that with positive probability $\mathcal{T}_{\mathcal{A}}$ returns the allocation \mathbf{x}' on $\mathcal{A} \in D(\mathbf{x})$ without encountering it in a black box query to the algorithm \mathcal{A} . Then there exists $\mathcal{A}' \in D(\mathbf{x})$ such that \mathcal{A}' and \mathcal{A} agree on each input queried by \mathcal{T} in some instance where $\mathcal{T}_{\mathcal{A}}$ returns \mathbf{x}' , and furthermore \mathcal{A}' never returns allocation \mathbf{x}' on any input. Note, then, that $\mathcal{T}_{\mathcal{A}'}$ also returns allocation \mathbf{x}' with positive probability. But $\mathcal{A}' \in D(\mathbf{x}')$, so if we set $\mathbf{x}_{bad} = \mathbf{x}'$ then we conclude that $\mathcal{T}_{\mathcal{A}'}$ returns the infeasible allocation \mathbf{x}' with positive probability. \square

In the remainder of the analysis we assume that \mathcal{T} only returns observed allocations. We will now think of \mathbf{x}_{bad} as being fixed, and \mathcal{A} as being drawn from $D(\mathbf{x}_{bad})$ uniformly at random. Let B denote the (bad) event, over randomness in \mathbf{v} , \mathcal{T} , and the choice of $\mathcal{A} \in D$, that $\mathcal{T}(\mathcal{A})$ returns an allocation with makespan α . Our goal will be to show that if $\mathcal{T}(\mathcal{A})$ is BIC for every $\mathcal{A} \in D$, then $\Pr[B]$ must be large; this will imply that the expected makespan of $\mathcal{T}(\mathcal{A})$ will be large for some $\mathcal{A} \in D$.

Intuitively, the reason that an algorithm $\mathcal{A} \in D$ may not be truthful is because low-speed agents are often allocated 1 while high-speed agents are often allocated 0. In order to fix such non-monotonicities, \mathcal{T} must either increase the probability with which 1 is allocated to the high-speed agents, or increase the probability with which 0 is allocated to the low-speed agents. To this end, let $U(\mathbf{v})$ be the event that, on input \mathbf{v} , $\mathcal{T}(\mathcal{A})$ returns an allocation \mathbf{x} in which at least $n^{3/4}$ agents satisfy $v_i = 1$ and $x_i = 0$.

As the following lemma shows, the event $U(\mathbf{v})$ is unlikely to occur unless B also occurs. Then to fix the non-monotonicity while avoiding B , \mathcal{T} must rely on allocating 1 more often to the high-speed agents. However this would require \mathcal{T} to query \mathcal{A} on speed vectors \mathbf{v}' that are near-complements of \mathbf{v} , and in turn imply a large-enough probability of allocating α to low-speed agents, i.e. event B . We now make this intuition precise.

Lemma 4.6. *For each \mathbf{v} , $\Pr[U(\mathbf{v}) \wedge \neg B | \mathbf{v}] < (\ln 4)e^{-n^{1/4}/2}$.*

Proof. Fix input \mathbf{v} , and suppose that event $U(\mathbf{v}) \wedge \neg B$ occurs. Recall that \mathcal{T} only returns an allocation that \mathcal{A} outputs on a query \mathbf{v}' . We will refer to a query of \mathcal{A} on input \mathbf{v}' as a *successful* query, if it returns an \mathbf{x} that satisfies $U(\mathbf{v}) \wedge \neg B$. Let us bound the probability of a single query being successful. Let t denote the number of agents with $v_i = 1$. Then $U(\mathbf{v})$ implies $t \geq n^{3/4}$.

First, suppose that \mathbf{v}' does not satisfy the event C , that is, the number of high speed agents in \mathbf{v}' is far from its mean $n/2$. Then each of the t agents has an $n^{-1/4}$ probability of being allocated α (taken over the choice of \mathcal{A} from D). The probability that none of the t agents is allocated a load of α is at most $(1 - n^{-1/4})^t < e^{-n^{1/2}}$.

On the other hand, suppose that \mathbf{v}' satisfies the event C . Then $U(\mathbf{v})$ implies that at least $t' \geq n^{3/4}$ agents satisfy $v_i = 1$ and $v_i' = \alpha$. In this case, each of these t' agents has probability $n^{-1/2}$ of being allocated α . The probability that none of them is allocated a load of α is at most $(1 - n^{-1/2})^{t'} < e^{-n^{1/4}}$.

In either case, the probability that a single query is successful is at most $e^{-n^{1/4}}$. Transformation \mathcal{T} can make at most $e^{n^{1/4}}/2$ queries on input \mathbf{v} . We will now bound the probability that any one of them is successful. First note that since \mathcal{A} is deterministic, we can assume that \mathcal{T} does not query \mathcal{A} more than once on the same input. Furthermore, we can think of the choice of $\mathcal{A} \in D$ as independently selecting the behaviour of \mathcal{A} for each input profile, so that the allocations returned by \mathcal{A} on different input profiles are independent with respect to the choice of \mathcal{A} from D . We can therefore think of the $e^{n^{1/4}}/2$ queries as independent trials that are successful with probability at most $e^{-n^{1/4}}$. Thus, the probability that at least one of these queries is successful is at most

$$\begin{aligned} 1 - (1 - e^{-n^{1/4}})^{e^{n^{1/4}}/2} &= 1 - (1 - e^{-n^{1/4}})^{e^{n^{1/4}}/2} \\ &< 1 - (1/4)^{e^{-n^{1/4}}/2} \\ &= 1 - (1/e)^{(\ln 4)e^{-n^{1/4}}/2} \\ &< (\ln 4)e^{-n^{1/4}/2} \end{aligned}$$

as required. \square

We now consider the specific probabilities with which \mathcal{T} returns high or low allocations on high or low values. For agent i , value v , and allocation x , we will write $p_i^x(v) = \Pr_{\mathbf{v}_{-i}, \mathcal{A}, \mathcal{T}}[x_i(v, \mathbf{v}_{-i}) = x]$. That is, $p_i^x(v)$ is the probability that conditioned on agent i 's value being v , $\mathcal{T}(\mathcal{A})$ allocates x to the agent; Here the probability is over the values of the other agents, any randomness in \mathcal{T} , and the choice of $\mathcal{A} \in D$.

Observation 4.7. $\sum_i p_i^x(v) = 2 \sum_i \Pr_{\mathbf{v}}[(v_i = v) \wedge (x_i(\mathbf{v}) = x)]$.

We can express the fact that \mathcal{T} satisfies BIC in terms of conditions on these probabilities (Lemma 4.8 below): either $p_i^0(1)$ should be large, i.e. low-speed agents get a low allocation, or one of $p_i^1(\alpha)$ and $p_i^\alpha(\alpha)$ should be large, i.e. high-speed agents get a high allocation. On the other hand, in Lemmas 4.9, 4.10, and 4.11 we show that on average over all agents, each of these probabilities is small if the probability of the bad event B is small. The proofs of these lemmas are deferred to the end of this section. In Lemma 4.12 we put these results together to argue that B occurs with a large probability.

Lemma 4.8. *Let \mathbf{x} be the allocation rule of $\mathcal{T}(\mathcal{A})$. Then if $x_i(1) \leq x_i(\alpha)$, $p_i^0(1) < 1/3$, and $p_i^1(\alpha) < 1/3$, then $p_i^\alpha(\alpha) > 3/\alpha$.*

Proof. Since $p_i^0(1) < \frac{1}{3}$, we have $x_i(1) > \frac{2}{3}$. Since $p_i^1(\alpha) < \frac{1}{3}$, we have $x_i(\alpha) < \frac{1}{3} + p_i^\alpha(\alpha)\alpha$. We conclude that $\frac{2}{3} < \frac{1}{3} + p_i^\alpha(\alpha)\alpha$ which implies the desired result. \square

Lemma 4.9. $\frac{1}{2n} \sum_i p_i^\alpha(\alpha) \leq n^{-\frac{1}{2}} + \Pr[B]n^{-1/4} + 2e^{-n^{1/4}}/4n^{-1/4} + (\ln 4)e^{-n^{1/4}}/2n^{-1/4}$.

Lemma 4.10. $\frac{1}{2n} \sum_i p_i^0(1) \leq n^{-1/4} + \Pr[B] + (\ln 4)e^{-n^{1/4}}/2$.

Lemma 4.11. $\frac{1}{2n} \sum_i p_i^1(\alpha) \leq 3n^{-1/4} + \Pr[B] + (\ln 4)e^{-n^{1/4}/2} + 2e^{-n^{1/4}/4}$.

The above lemmas put together give a lower bound for $\Pr[B]$. Set $\alpha = \frac{1}{4}n^{1/2}$.

Lemma 4.12. $\Pr[B] \geq n^{-1/4} - 2e^{-n^{1/4}/4} - (\ln 4)e^{-n^{1/4}/2}$.

Proof. We know that, for each i , either $p_i^0(1) \geq 1/3$, $p_i^1(\alpha) \geq 1/3$, or $p_i^\alpha(\alpha) > 3/\alpha$. So one of these inequalities must be true for at least one third of the agents, and hence one of the following must be true:

$$\frac{1}{2n} \sum_i p_i^0(1) \geq 1/18$$

$$\frac{1}{2n} \sum_i p_i^1(\alpha) \geq 1/18$$

$$\frac{1}{2n} \sum_i p_i^\alpha(\alpha) \geq 1/2\alpha.$$

Suppose the first inequality is true. Then by Lemma 4.10 we know

$$\Pr[B] \geq 1/18 - n^{-1/4} - (\ln 4)e^{-n^{1/4}/2}$$

which implies the desired result for sufficiently large n (as the right hand side is at least a constant for large n , whereas $n^{-1/4} - 2e^{-n^{1/4}/4} - (\ln 4)e^{-n^{1/4}/2}$, from the statement of the lemma, vanishes as n grows).

Suppose the second inequality is true. Then we know from Lemma 4.11

$$\Pr[B] \geq 1/18 - 3n^{-1/4} - (\ln 4)e^{-n^{1/4}/2} - 2e^{-n^{1/4}/4}$$

which again implies the desired result.

Finally, suppose the third inequality is true. Then we know from Lemma 4.9

$$n^{-\frac{1}{2}} + \Pr[B]n^{-1/4} + 2e^{-n^{1/4}/4}n^{-1/4} + (\ln 4)e^{-n^{1/4}/2}n^{-1/4} \geq 1/2\alpha$$

which implies (recalling $\alpha = \frac{1}{4}n^{1/2}$)

$$\Pr[B] \geq (2n^{-\frac{1}{2}} - n^{-\frac{1}{2}})n^{1/4} - 2e^{-n^{1/4}/4} - (\ln 4)e^{-n^{1/4}/2} = n^{-1/4} - 2e^{-n^{1/4}/4} - (\ln 4)e^{-n^{1/4}/2}$$

as required. □

We can now prove our main result.

Proof of Theorem 4.1. Write $\Pr[B \mid \mathcal{A}]$ for the probability of event B given that \mathcal{T} is given black-box access to algorithm \mathcal{A} , with probability over the choice of input profile \mathbf{v} . Choose $\mathcal{A}' \in \operatorname{argmax}_{\mathcal{A} \in D} \{\Pr[B \mid \mathcal{A}]\}$. Then in particular $\Pr[B \mid \mathcal{A}] \geq \Pr[B]$, where recall that $\Pr[B]$ is the probability of event B when \mathcal{A} is chosen uniformly at random from D .

Recall that we set $\alpha = \frac{1}{4}n^{1/2}$. By Lemma 4.4 the expected makespan of \mathcal{A}' is at most $1 + 2\alpha e^{-n^{1/4}/4} = 1 + \frac{1}{2}n^{1/2}e^{-n^{1/4}/4} < 2$ for large enough n . Using Lemma 4.12, the expected makespan of $\mathcal{T}_{\mathcal{A}'}$ is at least

$$\begin{aligned}
1 + \alpha \Pr[B|\mathcal{A}'] &\geq 1 + \alpha \Pr[B] \\
&\geq 1 + \alpha(n^{-1/4} - 2e^{-n^{1/4}/4} - (\ln 4)e^{-n^{1/4}/2}) \\
&= 1 + \frac{1}{4}n^{1/4} - \frac{1}{2}n^{1/2}e^{-n^{1/4}/4} - \left(\frac{\ln 4}{4}\right)n^{1/2}e^{-n^{1/4}/2} \\
&\geq \frac{1}{4}n^{1/4}
\end{aligned}$$

as required. \square

Proofs of bounds on the allocation probabilities. To conclude the analysis, we now present proofs of Lemmas 4.9, 4.10, and 4.11.

Proof of Lemma 4.9. Let us first condition on the event $\neg B \wedge \neg U(\mathbf{v}) \wedge C$. That is, we consider the output of $\mathcal{T}(\mathcal{A})$ on a value vector \mathbf{v} that satisfies the concentration event C , and further assume that the makespan of $\mathcal{T}(\mathcal{A})$ is small ($\neg B$) and few agents with $v_i = 1$ have a 0 allocation ($\neg U(\mathbf{v})$).

C implies that many of the agents (at least $\frac{1}{2}n - n^{3/4}$) in \mathbf{v} are low-speed agents. Along with $\neg B$ and $\neg U(\mathbf{v})$ this implies that most of these agents have an allocation of 1; Call this set of agents L . Then $|L| > \frac{1}{2}n - 2n^{3/4}$. In particular L is non-empty. Now suppose that $\mathcal{T}(\mathcal{A})$ returns an allocation \mathbf{x} that is returned by \mathcal{A} on input \mathbf{v}' . Then, since L is non-empty, \mathbf{v}' must satisfy the condition on line 2 of $\mathcal{A}(\mathbf{x}_{bad})$ (as this is the only way in which any agent can be allocated 1, regardless of the choice of $\mathcal{A} \in D$). This implies that at most $n^{1/2}$ agents get an allocation of α because S is of size at most $n^{1/2}$.

We conclude that for fixed \mathbf{v} satisfying C and conditioning on $\neg B \wedge \neg U(\mathbf{v})$,

$$\frac{1}{n} \sum_i \Pr[(v_i = \alpha) \wedge (x_i(\alpha, \mathbf{v}_{-i}) = \alpha)] \leq \frac{1}{n}n^{1/2} = n^{-1/2}.$$

For the case that events B , $\neg C$, or $U(\mathbf{v})$ occur, we note that every allocation returned by $\mathcal{T}(\mathcal{A})$ allocates α to at most $n^{3/4}$ agents. So, conditioning on either of these events, we have

$$\frac{1}{n} \sum_i \Pr[(v_i = \alpha) \wedge (x_i(\alpha, \mathbf{v}_{-i}) = \alpha)] \leq n^{-1/4}.$$

We conclude, taking probabilities over all \mathbf{v} , that

$$\begin{aligned}
\frac{1}{2n} \sum_i p_i^0(1) &\leq n^{-1/2} + \Pr[B]n^{-1/4} + \Pr[\neg C]n^{-1/4} + \Pr[U(\mathbf{v}) \wedge \neg B]n^{-1/4} \\
&\leq n^{-1/2} + \Pr[B]n^{-1/4} + 2e^{-n^{1/4}/4}n^{-1/4} + (\ln 4)e^{-n^{1/4}/2}n^{-1/4}
\end{aligned}$$

as required. \square

Proof of Lemma 4.10. For each input vector \mathbf{v} , either event $\neg U(\mathbf{v})$ occurs, event B occurs, or event $U(\mathbf{v}) \wedge \neg B$ occurs. Event $\neg U(\mathbf{v})$ by definition gives us a bound on the number of agents with value 1 that receive an allocation of 0. So conditioning on this event (and keeping \mathbf{v} fixed) we have

$$\frac{1}{n} \sum_i \Pr[(v_i = 1) \wedge (x_i(1, \mathbf{v}_{-i}) = 0)] \leq \frac{1}{n} n^{3/4} = n^{-1/4}.$$

Thus, taking probabilities over all \mathbf{v} , we have

$$\frac{1}{2n} \sum_i p_i^0(1) \leq n^{-1/4} + \Pr[B] + \Pr_{\mathbf{v}}[U(\mathbf{v}) \wedge \neg B | \mathbf{v}] \leq n^{-1/4} + \Pr[B] + (\ln 4)e^{-n^{1/4}/2}$$

as required. \square

Proof of Lemma 4.11. Let us first fix \mathbf{v} and condition on the event $\neg B \wedge \neg U(\mathbf{v}) \wedge C$. Suppose that $\mathcal{T}(\mathcal{A})$ returns an allocation \mathbf{x} that is returned by \mathcal{A} on input \mathbf{v}' .

As in the proof of Lemma 4.9, event $\neg B \wedge \neg U(\mathbf{v}) \wedge C$ implies that $\mathcal{T}(\mathcal{A})$ returns an allocation in which some agents are allocated 1. We therefore conclude that \mathbf{v}' satisfies the condition on line 2 of $\mathcal{A}(\mathbf{x}_{bad})$. This implies

$$-2n^{3/4} \leq |H(\mathbf{v})| - |H(\mathbf{v}')| \leq 2n^{3/4}.$$

Furthermore, $\neg U(\mathbf{v}) \wedge \neg B$ implies that $|H(\mathbf{v}') \setminus H(\mathbf{v})| \leq n^{3/4}$. Combining with the inequalities above, we conclude that $|H(\mathbf{v}) \setminus H(\mathbf{v}')| \leq 2n^{3/4} + n^{3/4}$. Note that this is a bound on the number of agents such that $v_i = \alpha$ and $v_i' = 1$, which is also a bound on the number of agents such that $v_i = \alpha$ and $x_i = 1$.

We conclude that, conditioning on event $\neg B \wedge \neg U(\mathbf{v}) \wedge C$ and keeping \mathbf{v} fixed, we have

$$\frac{1}{n} \sum_i \Pr[(v_i = \alpha) \wedge (x_i(\alpha, \mathbf{v}_{-i}) = 1)] \leq \frac{1}{n} (3n^{3/4}).$$

Thus, taking probabilities over all \mathbf{v} and all events, we have

$$\begin{aligned} \frac{1}{2n} \sum_i p_i^0(1) &\leq 3n^{-1/4} + \Pr[B] + \Pr_{\mathbf{v}}[U(\mathbf{v}) \wedge \neg B | \mathbf{v}] + \Pr[\neg C] \\ &\leq 3n^{-1/4} + \Pr[B] + (\ln 4)e^{-n^{1/4}/2} + 2e^{-n^{1/4}/4} \end{aligned}$$

as required. \square

5 Additive objective functions

In Section 4 we showed that no BIC approximation-preserving transformations are possible for the makespan objective. One of the properties of the social welfare objective that allows a BIC transformation where one cannot exist for makespan is that the objective function is additive across agents. This allows a transformation to focus on each agent individually while taking an aggregate view over other agents and preserving the performance with respect to the respective component of the objective function alone. [12] and [11] formalize this idea as follows: for each agent i they construct a mapping g_i from the value space of i to itself, and on input \mathbf{v} return the output of the algorithm on $g(\mathbf{v}) = (g_1(v_1), g_2(v_2), \dots)$. The mappings g_i ensure the following three properties:

- (P.1) the mapping preserves the distribution over values of i ,
- (P.2) the expected allocation of agent i upon applying the mapping, i.e. $x_i(g_i(v_i))$, is monotone non-decreasing in v_i , and,
- (P.3) the contribution of agent i to the overall social welfare is no worse than in the original algorithm.

The benefit of this approach is that if each agent's value space is single-dimensional or well structured in some other way, the computational problem of finding such a mapping becomes easy.

Given this construction, it is natural to ask whether there are other objectives that are additive across agents and for which such a per-agent ironing procedure works. We show in this section that for almost any objective other than social welfare, such an approach cannot work. In particular, given an objective satisfying some mild properties, we construct an instance such that for any mapping g_i from agent i 's value space to itself that satisfies properties (P.1) and (P.2) above, property (P.3) fails to hold by an arbitrarily large factor.

We focus first on maximization problems. Note that for an objective function of the form $\max \mathbf{E}_v[\sum_i x_i(\mathbf{v})h_i(v_i)]$ where h_i s are non-decreasing functions, the approach of [12] and [11] works as-is to give a BIC approximation preserving transformation. In the sequel, we consider objectives of the form $\max \mathbf{E}_v[\sum_i h_i(x_i(\mathbf{v}))v_i]$ where h_i is an arbitrary non-linear continuous function.

Theorem 5.1. *Consider the objective $\max \mathbf{E}_v[\sum_i h_i(x_i(\mathbf{v}))v_i]$ where each h_i is an arbitrary increasing function. Suppose that there exists an agent i for which h_i is a continuous super-linear function (i.e. $h_i(x) = \omega(x)$) or a continuous sub-linear function (i.e. $h_i(x) = o(x)$). Then for any $\epsilon \in (0, 1)$, there exists a distribution over agent values and an algorithm \mathcal{A} such that any transformation that performs a per-agent ironing of the allocation function of \mathcal{A} achieving properties (P.1) and (P.2) above, must violate property (P.3) by a factor of $\Omega(1/\epsilon)$.*

Proof. We focus on a single agent i and drop the subscript i to improve readability. Our algorithm makes non-zero allocations only to agent i so that the contribution of other agents to the objective is 0. Let h be the corresponding continuous increasing function and assume wlog that $h(0) = 0$ and $h(1) = 1$. In the remainder of the proof we assume that h is super-linear. The proof for the sub-linear case is similar.

With respect to this agent, our goal is to maximize the objective $E[h(x(v))v]$. Fix any $\epsilon > 0$ and consider the following instance and algorithm. The agent's value is 0 with probability $1 - \epsilon$ and 1 with probability ϵ . At 0, the algorithm always allocates an amount $1 + \epsilon'$. At 1, the allocation is 0 with probability $1 - 1/k$ and k with probability $1/k$. Here we pick $\epsilon' > 0$ and k such that $h(1 + \epsilon') < 1/(1 - \epsilon)$ and $h(k) > k/\epsilon$. The existence of ϵ' and k follows from the fact that h is continuous and superlinear.

Now, the expected allocation at 1 is 1, so in order to produce a BIC output, the mapping g must map each of the values to the other with some probability. Suppose that 0 gets mapped to 1 with probability $z/(1 - \epsilon)$ for some z . Then, to preserve the distribution over values, we must have $z \leq \epsilon$ and 1 must get mapped to 0 with probability z/ϵ .

How large does z have to be to fix the non-monotonicity? The new expected allocation at 0 is $z/(1 - \epsilon) + (1 - z/(1 - \epsilon))(1 + \epsilon')$, while the new expected allocation at 1 is $(1 - z/\epsilon) + z/\epsilon(1 + \epsilon')$. Setting the former to be no larger than the latter, and rearranging terms, we get

$$z/(1 - \epsilon) + z/\epsilon \geq 1$$

implying $z \geq \epsilon(1 - \epsilon)$.

Let us now compute the objective function value. The original objective function value is $\epsilon h(k)/k$. The new objective function value is given by

$$\begin{aligned} \epsilon(z/\epsilon h(1 + \epsilon') + (1 - z/\epsilon)h(k)/k) &< z/(1 - \epsilon) + (\epsilon - z)h(k)/k \\ &< h(k)/k(\epsilon - z + \epsilon z/(1 - \epsilon)) \\ &= h(k)/k(\epsilon - (1 - 2\epsilon)z/(1 - \epsilon)) \\ &< 2\epsilon^2 h(k)/k \end{aligned}$$

Here the first inequality follows from $h(1 + \epsilon') < 1/(1 - \epsilon)$, the second from $1 < \epsilon h(k)/k$ and the fourth from $z \geq \epsilon(1 - \epsilon)$. This implies that any mapping g that satisfies properties (P.1) and (P.2) must violate property (P.3) by a factor of at least $1/2\epsilon$.

A similar example can be constructed for sub-linear continuous h , and we skip the details. \square

Next we consider minimization problems of the form $\min \mathbf{E}_v[\sum_i h_i(x_i(\mathbf{v}))h'_i(v_i)]$ where h_i s are non-decreasing functions and h'_i s are non-increasing functions. Once again, if there exists an i such that h_i is non-linear, we get a gap.

Theorem 5.2. *Consider the objective $\max \mathbf{E}_v[\sum_i h_i(x_i(\mathbf{v}))h'_i(v_i)]$ where each h_i is an arbitrary increasing function and each h'_i is an arbitrary continuous decreasing function. Suppose that there exists an agent i for which h_i is a continuous super-linear function (i.e. $h_i(x) = \omega(x)$) or a continuous sub-linear function (i.e. $h_i(x) = o(x)$). Then for any $\epsilon > 0$, there exists a distribution over agent values and an algorithm \mathcal{A} such that any transformation that performs a per-agent ironing of the allocation function of \mathcal{A} achieving properties (P.1) and (P.2) above, must violate property (P.3) by a factor of $\Omega(1/\epsilon)$.*

Proof. Once again we focus on the agent i and present the proof for the case where the function h (the subscript i being implicit) is continuous and super-linear. The proof for the sub-linear case is similar. Assume without loss of generality that $h(0) = 0$ and $h(1) = 1$. Let $k \geq 1$ be such that $h(k) > k/\epsilon$. Let $v_1 = h'^{-1}(1)$ and $v_2 = h'^{-1}(\epsilon/k)$. Agent i 's value distribution is uniform over $\{v_1, v_2\}$.

The algorithm \mathcal{A} is defined as follows. At v_1 , the algorithm returns x_1 with $h(x_1) = 1 + \epsilon$. Note that $x_1 > 1$. At v_2 , the algorithm returns k with probability $1/k$ and 0 otherwise. The expected allocation is 1, and therefore the allocation is non-monotone. Suppose that the transformation maps v_1 to v_2 with probability z and vice versa. Then it is easy to see that $z > 1/2$.

Now let us compute the objective function value. The objective function value of the original algorithm is $1/2(1 + \epsilon) + 1/2\epsilon/k(h(k)/k) = 1/2 + \epsilon/2(1 + h(k)/k^2) < 1 + \epsilon h(k)/2k^2$. We can bound from below the objective function value of the transformed mechanism by considering the term corresponding to value v_1 when the allocation is k (an event that happens with probability $1/2$ times z times $1/k$). The new objective function value is therefore at least

$$\begin{aligned} zh(k)/2k &\geq h(k)/4k \\ &\geq 1/8\epsilon + h(k)/16k^2 = 1/8\epsilon(1 + \epsilon h(k)/2k^2) \end{aligned}$$

Here the second inequality follows by using $h(k) > k/\epsilon$ and $k > 1/2$. This implies that any transformation that satisfies properties (1) and (2) must violate property (3) by a factor of at least $1/8\epsilon$. \square

To conclude this section we note that for minimization problems with objectives of the form $\mathbf{E}_{\mathbf{v}}[\sum_i x_i(\mathbf{v})h'_i(v_i)]$ where h'_i 's are decreasing functions, an approach similar to the ironing approach of [11] gives a BIC approximation-preserving transformation. In particular, if the type space for each agent is finite then we can find a mapping by finding the min-cost perfect matching where edge costs between v_i and v'_i are $h'_i(v_i)\mathbf{E}_{\mathbf{v}_{-i}}[x_i(v'_i, \mathbf{v}_{-i})]$.

References

- [1] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.
- [2] Itai Ashlagi, Shahar Dobzinski, and Ron Lavi. An optimal lower bound for anonymous scheduling mechanisms. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 169–176, 2009.
- [3] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *Journal of the ACM*, 2009.
- [4] X. Bei and Z. Huang. Bayesian incentive compatibility via fractional assignments. In *Proc. 22nd ACM Symp. on Discrete Algorithms*, 2011.
- [5] D. Buchfuhrer, S. Dughmi, H. Fu, R. Kleinberg, E. Mossel, C. Papadimitriou, M. Schapira, Y. Singer, and C. Umans. Inapproximability for vcg-based combinatorial auctions. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.
- [6] George Christodoulou and Annamária Kovács. A deterministic truthful ptas for scheduling related machines. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1005–1016, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.
- [7] 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.
- [8] S. Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proc. 42nd ACM Symp. on Theory of Computing*, 2011.
- [9] S. Dughmi and T. Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *Proc. 51st IEEE Symp. on Foundations of Computer Science*, 2010.
- [10] S. Dughmi, T. Roughgarden, and Q. Yan. From convex optimization to randomized mechanisms: Toward optimal combinatorial auctions. In *Proc. 42nd ACM Symp. on Theory of Computing*, 2011.
- [11] J. Hartline, R. Kleinberg, and A. Malekian. Bayesian incentive compatibility via matchings. In *Proc. 22nd ACM Symp. on Discrete Algorithms*, 2011.
- [12] J. Hartline and B. Lucier. Bayesian algorithmic mechanism design. In *Proc. 41st ACM Symp. on Theory of Computing*, 2010.

- [13] Dorit s. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17:539–551, June 1988.
- [14] N. Immorlica and B. Lucier. On the impossibility of black-box truthfulness without priors. In *Workshop on Bayesian Mechanism Design*, 2011.
- [15] 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.
- [16] C. Papadimitriou, M. Schapira, and Y. Singer. On the hardness of being truthful. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, 2008.