On the Limits of Black-Box Reductions in Mechanism Design

Shuchi Chawla*
University of Wisconsin - Madison
shuchi@cs.wisc.edu

Nicole Immorlica†
Northwestern University
nicimm@gmail.com

Brendan Lucier
MSR New England
brlucier@microsoft.com

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.

Categories and Subject Descriptors

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

General Terms

Theory

Keywords

Mechanism design, algorithms, social welfare, makespan

1. INTRODUCTION

Mechanism design studies optimization in the presence of selfish agents, where the goal is to design a system in which agents' individual selfish behaviour leads to global optimization of a desired objective. A central theme in *algorithmic* mechanism design is to reconcile the incentive constraints of selfish agents 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 requirements of algorithm design and mechanism design, manifested via reductions that convert arbitrary algorithms into incentive compatible mechanisms. One example is the seminal work of Vickrey, Clark, and Groves for converting *exact* algorithms into mechanisms for the social welfare objective. Much recent research has focused on generalizing such reductions to encompass approximation algorithms. Recently, striking positive results have been obtained for many important cases, especially single-parameter problems, in both Bayesian [12, 11, 4] and ex post (i.e. non-Bayesian) [15, 3, 8, 10, 14] settings. Given this progress and the notable lack of impossibility results for single-parameter problems, it is tempting to guess that a fully general reduction[1] is possible.

In this paper, we show that there is no such general black-box reduction, even for single-parameter problems. By *black-box reduction* we mean a mechanism that is given query access to an algorithm, and must return an allocation discovered by querying the algorithm. Our two main results address mechanisms in two different settings:

1. For the standard optimization objective of *social welfare*, or sum of agents' values, we prove that no general black-box reduction can guarantee ex post truthfulness while approximately preserving the worst-case approximation of the underlying algorithm.

2. In the Bayesian setting, we prove that there are monotone[2] optimization objectives (including makespan) for which no black-box reduction can guarantee truthfulness while approximately preserving the average-case approximation of the underlying algorithm.

Our results stand in contrast to existing constructions in Bayesian mechanism design [12, 11, 4]. In that prior work, it was shown that general black-box reductions for social welfare exist in a Bayesian setting, where the input is drawn from a commonly known distribution **F** and the optimization

*This author was supported in part by NSF awards CCF-0830494, CCF-0643763, and CCF-1101429.

†This author was supported in part by NSF CAREER grant AF-1055020, a Microsoft New Faculty Fellowship, and an Alfred P. Sloan Foundation Fellowship.

STOC'12, May 19–22, 2012, New York, New York, USA.

[1] That is, a reduction that applies to single-parameter optimization problems with arbitrary feasibility constraints.

[2] An objective is *monotone* if exact optimization leads to a truthful (i.e. monotone) allocation rule; see eq. (1).

and incentive compatibility requirements are with respect to **F**. Our first result demonstrates that no such reduction is possible when the solution concept is ex post incentive compatibility (i.e. truthfulness in dominant strategies) and the optimization goal is worst-case approximation. Specifically, we show that every black-box mechanism must degrade the worst-case performance of some algorithms (including constant approximations) by an unbounded amount. Our impossibility holds also for randomized transformations that aim to achieve truthfulness in expectation over their random bits[3].

In the Bayesian setting, we ask whether the known constructions for social welfare can be extended to other monotone objectives. In our second result, we prove that for the objective of makespan minimization, 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 **F** and measure the algorithm's performance in expectation over **F**. Our construction serves to demonstrate that mechanisms like those known 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 that lie between the linear welfare objective and the highly non-linear makespan objective that admit reductions. Roughly speaking, known black-box mechanisms for Bayesian 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 to be optimized, the incentive requirements, and the performance metric. We distinguish between social welfare and arbitrary monotone objectives, and between the average performance of the algorithm (over a distribution of inputs) and its worst case performance. We further distinguish between two kinds of incentive requirements (defined formally in Section 2). Bayesian incentive compatibility (BIC) implies that truthtelling forms a Bayes-Nash equilibrium given the agents' value distributions. The stronger notion of truthfulness in expectation (TIE) implies that every agent maximizes her expected utility by truthtelling regardless of others' actions, with expectation taken over any randomness in the mechanism.

Table 1 below summarizes our findings as well as known results along these three dimensions. 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

	Avg-case approx	Worst-case approx
BIC	Yes [12, 4, 11]	?
TIE	?	No (Section 3)

Objective: arbitrary monotone (e.g. makespan)

	Avg-case approx	Worst-case approx
BIC	No (Section 4)	No
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 a polynomially large loss in performance.

One way to establish our impossibility results would be to find 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 that has resisted much effort by the algorithmic mechanism design community, and is beyond the scope of our work. Instead, we focus on the black-box nature of a reduction from mechanism design to algorithm design. In our model, a black-box mechanism has full knowledge of the problem instance including the distribution from which agents' values are drawn. In addition it is given oracle access to an algorithm that it can query at any input. Given an input value vector, the goal of the mechanism 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 mechanism is meant to be "black-box," we will require that it can only output an allocation that it has observed while querying the algorithm at different vectors[4]. This last property is crucial in our constructions and allows us to "hide" good allocations from the transformation.

While the restriction to black-box mechanisms may appear strong at first glance, we note that many existing constructions for general mechanisms satisfy these requirements. For instance, the standard technique of maximal in range (MIR) mechanism design proceeds by restricting the range of outcomes to a subset of feasible allocations and then solving the optimization problem exactly over this range. An appropriate set of outcomes can be constructed by querying a given algorithm $\mathcal{A}$ at representative inputs [10, 14]. An alternative technique [12, 11, 4] proceeds by constructing a mapping π between input profiles: given input vector $\mathbf{v}$, the mechanism returns the original algorithm's allocation at vector $\pi(\mathbf{v})$. In both cases, the mechanism always satisfies any feasibility constraints of the problem, since it always returns an allocation in the range of the given algorithm.

[3]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.

[4]We can think of this requirement as being necessitated by the presence of arbitrary feasibility constraints that are not known to the mechanism, but are satisfied by the algorithm.

To prove our impossibility result for social welfare, we describe a class of "public project" optimization problems and show that achieving ex post incentive compatibility essentially requires a MIR approach. Furthermore, an underlying algorithm can hide a "long-distance" non-monotonicity in such a way that, at some value vectors, the transformation must either query the algorithm at exponentially many inputs in order to detect the non-monotonicity, or return a default outcome that guarantees monotonicity but has a poor approximation ratio.

For our impossibility result in the Bayesian setting, we show that the per-agent mapping approach of [12] must fail for almost any objective other than social welfare. We consider the makespan objective, and construct a constant approximation algorithm for which any polytime BIC transformation must have large expected makespan. The key to our construction is that the algorithm will return good allocations only on typical (i.e. concentrated) inputs; this limits a mechanism's ability to explore the allocation space by querying at atypical inputs.

All of our constructions involve simple instances with symmetric value spaces: each agent has a high or a low value and gets a high or a low (or medium in one case) allocation. The complexity in our constructions lies in the feasibility constraints satisfied by the algorithm to be transformed. In particular, our constructions involve feasibility constraints that are asymmetric across agents. This is essential: when agents have few possible types, good mechanisms can always be constructed in the absence of a feasibility constraint, or when the constraint is symmetric across agents[5].

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) TIE β-approximation mechanism from any β-approximation that verifies an integrality gap. Dughmi, Roughgarden and Yan [9] extend this approach and obtain TIE mechanisms for a broad class of submodular combinatorial auctions. Dughmi and Roughgarden [8] give a construction that converts any FPTAS algorithm for a social welfare problem into a TIE mechanism.

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 restricted to lie in $[v_{min}, v_{max}]$. Huang et al. [14] prove that any algorithm for a symmetric single-parameter social welfare problem can be converted into a truthful MIR mechanism with arbitrarily small loss in approximation, building upon a reduction with logarithmic loss due to Goel et al. [10].

Many papers have explored the limitations of truthful mechanisms for approximating social welfare in settings with multi-parameter types. Papadimitriou, Schapira and Singer [16] show a gap between the power of deterministic algorithms and truthful mechanisms, and a similar gap was established by Dobzinski [7] for randomized TIE mechanisms. However, prior to our work, it was not known whether a lossless black-box reduction could exist for the class of single-parameter problems.

For the makespan objective in multi-parameter settings (i.e. when the sizes of jobs on different machines are unrelated), Ashlagi et al. [2] proved a gap between the approximating power of algorithms and "anonymous" incentive compatible mechanisms. The situation for single-parameter (a.k.a. related machines) settings is quite different: a line of work on incentive compatible mechanisms [1, 6] recently culminated in the development of a deterministic truthful PTAS by Christodoulou and Kovács [5], matching the approximation of the best possible approximation algorithm [13]. Our negative result for makespan minimization differs in that we consider black-box mechanisms in a setting where the underlying algorithm may satisfy an additional feasibility constraint.

2. PRELIMINARIES

Optimization Problems. In a single-parameter optimization problem we are given as input a value vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, where each v_i is drawn from a known set $V_i \subseteq \mathbb{R}$. Let $V = V_1 \times \cdots \times V_n$ denote the input space. The goal is to choose an allocation $\mathbf{x} \in \mathcal{F} \subseteq \mathbb{R}^n$ from a set of feasible allocations $\mathcal{F}$ such that a given objective function $\phi : \mathcal{F} \times V \to \mathbb{R}$ 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. For example, the social welfare objective is $\phi(\mathbf{x}, \mathbf{v}) = \mathbf{x} \cdot \mathbf{v}$ and the makespan objective is $\phi(\mathbf{x}, \mathbf{v}) = \max_i x_i/v_i$.

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. In general $\mathcal{A}$ can be randomized, in which case $\mathcal{A}(\mathbf{v})$ is a random variable. Given an instance $(\mathcal{F}, \phi)$, we will write $OPT_{\mathcal{F}}(\mathbf{v})$ for the allocation in $\mathcal{F}$ that optimizes $\phi(\mathbf{x}, \mathbf{v})$. Let $approx_{\mathcal{F}}(\mathcal{A})$ denote the worst-case approximation ratio of $\mathcal{A}$ for problem $\mathcal{F}$. That is, $approx_{\mathcal{F}}(\mathcal{A}) = \min_{\mathbf{v} \in V} \frac{\mathbf{E}[\mathcal{A}(\mathbf{v})]}{OPT_{\mathcal{F}}(\mathbf{v})}$ for a maximization problem, where the expectation is over any randomness in $\mathcal{A}$. Note that $approx_{\mathcal{F}}(\mathcal{A}) \leq 1$ for all $\mathcal{F}$ and $\mathcal{A}$.

In the Bayesian version of our optimization problem, there is publicly-known product distribution $\mathbf{F} = F_1 \times \ldots \times F_n$ over input vectors. Given $\mathbf{F}$, the expected objective value of algorithm $\mathcal{A}$ is $\overline{\phi}(\mathcal{A}) = \mathbf{E}_{\mathbf{v} \sim \mathbf{F}}[\phi(\mathcal{A}(\mathbf{v}), \mathbf{v})]$. The goal in this setting is to optimize the expected objective value.

Mechanisms. We consider a mechanism design setting with n rational agents, where each agent possesses one value from the input vector as private information. An outcome $\mathbf{x}$ represents an *allocation* to the agents, where x_i is the allocation to agent i. A mechanism proceeds by eliciting declared values $\mathbf{b} \in \mathbb{R}^n$ from the agents, then applying an allocation algorithm $\mathcal{A} : \mathbb{R}^n \to \mathcal{F}$ that maps $\mathbf{b}$ to an allocation $\mathbf{x}$, and

[5]In particular, when agents have few possible types and the feasibility constraint is symmetric, an input profile is determined entirely by the number of agents of each type, of which there are only polynomially many such choices. One can therefore implement a MIR mechanism that queries the given algorithm on every type profile, up to permutation.

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 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. 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 it is *monotone*: that is, for all i, all $\mathbf{v}_{-i}$, and all $v_i \leq v_i{}'$,

$$\mathbf{E}[x_i(v_i, \mathbf{v}_{-i})] \leq \mathbf{E}[x_i(v_i{}', \mathbf{v}_{-i})]. \quad (1)$$

We say that a 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). 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 and all $v_i \leq v_i{}'$,

$$\mathbf{E}_{\mathbf{v}_{-i}}[x_i(v_i, \mathbf{v}_{-i})] \leq \mathbf{E}_{\mathbf{v}_{-i}}[x_i(v_i{}', \mathbf{v}_{-i})]. \quad (2)$$

Transformations. A *transformation* $\mathcal{T}$ is an algorithm that has black-box access to some algorithm $\mathcal{A}$. Write $\mathcal{T}(\mathcal{A}, \mathbf{v})$ for the output of $\mathcal{T}$ on input $\mathbf{v}$, given that its black-box access is to $\mathcal{A}$. We write $\mathcal{T}(\mathcal{A})$ for the allocation rule[6] $\mathcal{T}(\mathcal{A}, \cdot)$. Note that $\mathcal{T}$ is not parameterized by $\mathcal{F}$; informally speaking, $\mathcal{T}$ has no knowledge of the feasibility constraint $\mathcal{F}$, so in particular we can assume that $\mathcal{T}(\mathcal{A}, \mathbf{v})$ is always an allocation observed by querying $\mathcal{A}$. We do assume that $\mathcal{T}$ is aware of the objective function ϕ, the input domain V, and (in Bayesian settings) the distribution $\mathbf{F}$.

Transformation $\mathcal{T}$ is truthful in expectation (TIE) if $\mathcal{T}(\mathcal{A})$ is TIE for all $\mathcal{A}$. Likewise, $\mathcal{T}$ is Bayesian incentive compatible (BIC) for distribution $\mathbf{F}$ if, for all $\mathcal{A}$, $\mathcal{T}(\mathcal{A})$ is BIC for $\mathbf{F}$. Note that whether or not $\mathcal{T}$ is TIE or BIC is independent of the objective ϕ and feasibility constraint $\mathcal{F}$.

3. A LOWER BOUND FOR TIE TRANSFORMATIONS

We will prove the following theorem, which states that *any* TIE transformation must convert some constant factor approximate algorithms into mechanisms with polynomial approximation factors.

THEOREM 3.1. *There are constants c_1, c_2 such that, for any polytime TIE transformation $\mathcal{T}$, there is an algorithm $\mathcal{A}$ and problem instance $\mathcal{F}$ with $approx_{\mathcal{F}}(\mathcal{A}) \geq c_1$ such that $approx_{\mathcal{F}}(\mathcal{T}(\mathcal{A})) \leq 1/n^{c_2}$.*

Before describing our construction, let us build some intuition for the requirements of Theorem 3.1. Our goal is to construct an algorithm $\mathcal{A}$ that is difficult to monotonize (i.e. satisfy (1)). Since $\mathcal{T}$ is limited to making a polynomial number of queries on each input, a natural approach is to hide a non-monotonicity in the (exponentially large) input space. This idea is impeded by the locality of the monotonicity requirement: on input $\mathbf{v}$, transformation $\mathcal{T}$ can query $\mathcal{A}$ on each input that differs from $\mathbf{v}$ on a single coordinate, and hence find any pair $\mathbf{v}, \mathbf{v}'$ that violates (1).

Alternatively, one could embed a "high-dimensional" non-monotonicity into $\mathcal{A}$ by choosing input vectors $\mathbf{v}, \mathbf{v}'$ that differ on many coordinates and setting $\mathcal{A}_i(\mathbf{v}) > \mathcal{A}_i(\mathbf{v}')$ for each agent i with $v_i < v_i{}'$. Such a situation seems harder to detect, but does not directly imply non-truthfulness. For example, if $\mathcal{A}$ satisfies $\mathcal{A}(1,1) = (10,10)$ and $\mathcal{A}(2,2) = (5,5)$, then condition (1) is not necessarily violated: it may be that $\mathcal{A}(1,2) = (4,11)$ and $\mathcal{A}(2,1) = (11,4)$. However, if we additionally impose a feasibility constraint that the agents involved in the high-dimensional non-monotonicity *always receive the same allocation*, then we could infer that (1) must be violated somewhere. Thus, under certain feasibility constraints, $\mathcal{T}$ must find and correct high-dimensional non-monotonicities.

Motivated by this observation, our approach to proving Theorem 3.1 is as follows. We will construct an approximation algorithm $\mathcal{A}$ containing a high-dimensional non-monotonicity and a feasibility constraint that together imply that $\mathcal{A}$ is not TIE. Our algorithm will be such that $\mathcal{T}$ cannot detect the non-monotonicity (or even that one exists) with non-negligible probability. Thus, to guarantee incentive compatibility, $\mathcal{T}$ will be forced to modify the algorithm's output aggressively in order to address any possible non-monotonicity that could be present. However, our algorithm will be constructed so that such a modification must degrade its approximation factor significantly.

Various technical hurdles must be overcome to implement this plan. First, we must restrict the set of allocations that can be found by $\mathcal{T}$ (with non-negligible probability) when adaptively querying the algorithm, so that $\mathcal{T}$ cannot find a way to correct non-monotonicities without harming the algorithm's approximation factor. At the same time, we require that our algorithm obtain a constant approximation, which limits our ability to hide allocations from $\mathcal{T}$ via suboptimality. The details of the construction involve carefully balancing the tension between these requirements.

3.1 Construction

In the instances we consider, each private value v_i is chosen from $\{v, 1\}$, where $v \in (0,1)$ is a parameter of our construction. We interpret an input vector as a subset $y \subseteq [n]$, corresponding to those agents with the "high" value 1. For $a \geq 0$ and $S \subseteq [n]$, we will write $\mathbf{x}^a_S$ for the allocation in which each agent $i \in S$ is allocated a, and each agent $i \notin S$ is allocated 0.

Feasible Allocations. We consider a family of feasibility constraints, which will be specified by parameters α, γ with $0 < \gamma < \alpha < 1$ and sets $S, T \subseteq [n]$ of agents. We think of S and T as being polynomially large sets, with a polynomially large intersection; see Figure 1(a). The feasible allocations will be $\mathbf{x}^\gamma_{[n]}$, $\mathbf{x}^1_S$, and $\mathbf{x}^\alpha_T$. That is, we can allocate γ to every agent, 1 to all agents in S, or α to all agents in T. We write $\mathcal{F}_{S,T,\alpha,\gamma}$ for this feasibility constraint.

The Algorithm. Fix feasibility constraint $\mathcal{F}_{S,T,\alpha,\gamma}$. Let $U = S \cap T$ and choose a set $V \subset U$. We will define an approximation algorithm $\mathcal{A} = \mathcal{A}_{V,S,T,\alpha,\gamma}$ for problem $\mathcal{F}_{S,T,\alpha,\gamma}$. This will be our candidate for the algorithm in the statement of Theorem 3.1.

[6]Note that a polynomial-time transformation can make only polynomially many queries to compute $\mathcal{T}(\mathcal{A}, \mathbf{v})$ for *each* $\mathbf{v}$, but $\mathcal{T}(\mathcal{A})$ may depend on the behaviour of $\mathcal{A}$ on a super-polynomially large subset of the input space.

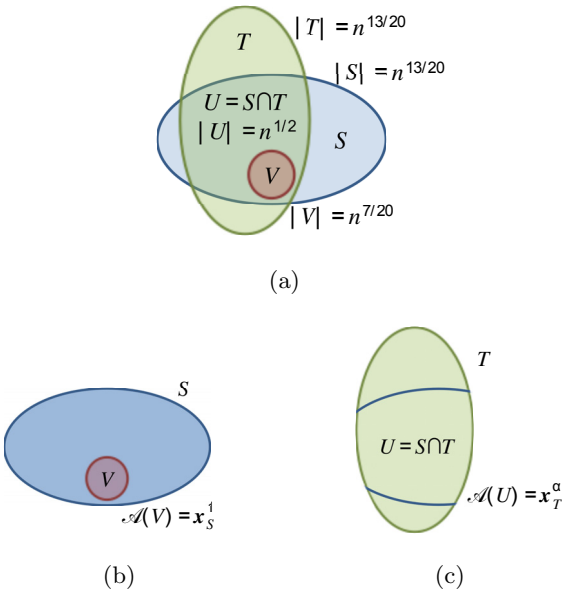

Figure 1: (a) Visualization of typical admissible sets of bidders V, S, and T, and the corresponding allocations of algorithm $\mathcal{A}$ on (b) input V and (c) input $U = S \cap T$.

We begin with an informal description of Algorithm $\mathcal{A}$ (formally listed as Algorithm 1 below). Recall that $\mathcal{A}$ accepts as input a vector $\mathbf{v} \in \{v, 1\}^n$. $\mathcal{A}$ will return $\mathbf{x}_T^\alpha$ if sufficiently many of the high-valued agents lie in set T, and will return $\mathbf{x}_S^1$ if sufficiently many of the high-valued agents lie in set S. When most high-valued agents lie in $S \cap T$, set V is used to determine the outcome: if sufficiently many high-valued agents lie in V then we return $\mathbf{x}_S^1$, otherwise we return $\mathbf{x}_T^\alpha$. If there are not enough high-valued agents, or if too many lie outside $S \cup T$, then we return $\mathbf{x}_{[n]}^\gamma$. See Figure 1.

Algorithm $\mathcal{A}$ has the following properties that will be important to our analysis. First, all outcomes of $\mathcal{A}$ are constant on U. Also, $\mathcal{A}(V) = \mathbf{x}_S^1$ and $\mathcal{A}(U) = \mathbf{x}_T^\alpha$, so if $\alpha < 1$ then $\mathcal{A}$ contains a high-dimensional non-monotonicity as described at the beginning of this section. Moreover, $\mathcal{A}$ will return $\mathbf{x}_S^1$ for any input y that is sufficiently close to V (i.e. small symmetric difference), and for a randomly chosen subset of S (with high probability). On the other hand, any input sufficiently close to U or chosen randomly as a subset of T will generate outcome $\mathbf{x}_T^\alpha$. These last properties will be important in arguing that a TIE transformation is unlikely to detect the non-monotonicity between inputs V and U. Finally, as we demonstrate in the next section, $\mathcal{A}$ obtains a constant approximation when parameters are chosen appropriately.

Before presenting the formal description of Algorithm $\mathcal{A}$ we require some definitions. 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 $\mathbf{x} \in \mathcal{F}_{V,S,T,\alpha}$

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

3.2 Analysis

We now derive the key lemmas for the proof of Theorem 3.1.

Parameter Conditions. All of the results in this section will require that we impose certain technical constraints on the parameters α and γ and the sets V, S, and T. Roughly, parameter $1/\gamma$ and sets V, $S \cap T$, S, and T should be sufficiently large, and α should be sufficiently small (as functions of n and v).

Formally, we will choose parameters $t \geq r \geq 1$ (polynomially large in n) which will dictate the required sizes of sets V, S, and T. We will choose these parameters so that

$$t \gg r \gg \frac{1}{\gamma} \gg \frac{1}{\alpha},\ r^5 t \leq n,\ \text{and}\ \frac{t}{\gamma n} \ll 1.$$

Given this choice of r and t, we say that sets V, S, and T are *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. $|V| = rt$.

See Figure 1(a) for an illustration of the relationship between the sets in an admissible triple. For the remainder of this section we will assume that V, S, and T are feasible. We will set the values of parameters r, t, α, γ, and v in Section 3.3.

Algorithm Analysis. For our first lemma, we bound the approximation factor of $\mathcal{A}_{V,S,T,\alpha,\gamma}$ for problem $\mathcal{F}_{V,S,T,\alpha,\gamma}$. In particular, we note that if $\alpha = 1$ then $\mathcal{A}_{V,S,T,\alpha,\gamma}$ is a constant approximation.

LEMMA 3.2. $approx_{\mathcal{F}_{S,T,\alpha,\gamma}}(\mathcal{A}_{V,S,T,\alpha,\gamma}) \geq \alpha/6$.

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

The allocation $\mathbf{x}^{\gamma}_{[n]}$ 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 $\mathbf{x}^{\alpha}_T$ 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 $\mathbf{x}^{\gamma}_{[n]}$ for a welfare of at least $\gamma(|y| + v(n - |y|)) \geq t$. The allocation $\mathbf{x}^{1}_S$ obtains welfare at most $|y \cap S| + t \leq n_S(y) + t \leq 2t$, and allocation $\mathbf{x}^{\alpha}_T$ 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 $\mathbf{x}^{\gamma}_{[n]}$ for a welfare of at least $\gamma(|y| + v(n - |y|)) \geq \gamma|y|$. The allocation $\mathbf{x}^{1}_S$ obtains welfare at most $|y \cap S| + t \leq 2n_S(y) \leq 2\gamma|y|$, and allocation $\mathbf{x}^{\alpha}_T$ 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. □

Our next claim restricts the behaviour of TIE algorithms for problem $\mathcal{F}_{S,T,\alpha,\gamma}$. The result uses the fact that all allocations in $\mathcal{F}_{S,T,\alpha,\gamma}$ are constant on U.

Claim 3.3. *Suppose $\mathcal{A}'$ is a truthful-in-expectation algorithm for problem $\mathcal{F}_{S,T,\alpha,\gamma}$. 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 relative to its allocation on input V. Since all allocations are constant on U, this means that the expected allocation to each agent in U on input W must not decrease relative to allocation on input V. By the same argument, $\mathcal{A}'$ returns an allocation to each agent in U at least this large for all W such that $V \subseteq W \subseteq U$, and in particular for $W = U$. □

Transformation Analysis. In light of Claim 3.3, our strategy for proving Theorem 3.1 will be to show that a polytime transformation $\mathcal{T}$ is unlikely to encounter the allocation $\mathbf{x}^{\alpha}_T$ 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 satisfying the conditions of Claim 3.3 by giving an expected allocation of α to the agents in U on input V. Similarly, a transformation is unlikely to encounter the allocation $\mathbf{x}^{1}_S$ during its sampling on input U, and therefore cannot satisfy Claim 3.3 by allocating 1 to agents in U on input U.

Our next two lemmas demonstrate that if the sets V, S, and T were chosen at random, then it is exponentially unlikely that a transformation would observe allocation $\mathbf{x}^{\alpha}_T$ during its sampling when given input V (Lemma 3.4), or observe allocation $\mathbf{x}^{1}_S$ given input U (Lemma 3.5).

Lemma 3.4. *Fix V and S satsifying the requirements of admissibility. Then for any $y \subseteq [n]$, $\Pr_T[\mathcal{A}_{V,S,T,\alpha}(y) = x^{\alpha}_T] \leq e^{-O(\frac{t}{r+1})}$, 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 $m_T = |y \cap (T \backslash S)|$.

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

$$m_T + 2n_V + 2m_U \geq t, \tag{3}$$

$$m_T + 2n_V + 2m_U \geq \gamma(n_V + n_S + n_*), \tag{4}$$

$$m_T + 2n_V + 2m_U \geq n_S + 3n_V. \tag{5}$$

We will show that the probability of these three inequalities being true is exponentially small. To see this, note that (5) implies that $m_T + 2m_U \geq n_V$. Thus, (3) 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 (4). □

Lemma 3.5. *Fix U and T satsifying the requirements of admissibility. Then for any $y \subseteq [n]$, $\Pr_S[\mathcal{A}_{V,S,T,\alpha}(y) = x^{1}_S] \leq e^{-O(\frac{t}{r+1})}$, 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 $m_S = |y \cap (S \backslash T)|$.

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

$$m_S + n_U + 2m_V \geq t, \tag{6}$$

$$m_S + n_U + 2m_V \geq \gamma(n_U + n_T + n_*), \tag{7}$$

$$m_S + n_U + 2m_V \geq n_T + 2n_U. \tag{8}$$

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 (8) implies $m_S + 2m_V \geq n_U$. Thus, (6) 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 (7). □

We emphasize that the results of Lemma 3.4 and 3.5 hold for *every* possible input vector y. This is crucial, since we wish for these lemmas to apply even when inputs are chosen adaptively by transformation $\mathcal{T}$ to gain information about the given algorithm and feasibility constraint.

3.3 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 cannot determine the value of α on input V (by Lemma 3.4), and since it cannot find the "good" allocation of $\mathbf{x}^1_S$ 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 $\mathcal{T}(\mathcal{A}_{V,S,T,\alpha})$. Consider choosing V, S, and T uniformly at random over all admissible choices of these sets. Lemma 3.5 implies that, with all but exponentially small probability, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation $\mathbf{x}^1_S$ on input U (where the probability is taken over the choice of sets and any randomness in the transformation). Furthermore, Lemma 3.4 implies that, with all but exponentially small probability, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation $\mathbf{x}^\alpha_T$ on input V. We conclude that there exists an admissible choice of sets V, S, and T such that, with all but exponentially small probability, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation $\mathbf{x}^1_S$ on input U nor allocation $\mathbf{x}^\alpha_T$ on input V.

Fix this choice of V, S, and T. In the exponentially unlikely event that $\mathcal{A}'_{V,S,T,\alpha}(U)$ encounters $\mathbf{x}^1_S$ or $\mathcal{A}'_{V,S,T,\alpha}(V)$ encounters $\mathbf{x}^\alpha_T$, algorithm $\mathcal{A}'_{V,S,T,\alpha}$ can obtain a welfare of at most $|V| = n^{7/20}$ on input V.

Suppose now that this exponentially unlikely event does not occur. We then have that, on input U, $\mathcal{A}'_{V,S,T,\alpha}$ 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.

On input V, $\mathcal{A}'_{V,S,T,\alpha}$ will not encounter allocation $\mathbf{x}^\alpha_T$ and thus is unaware of the value of α. 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 at most $|V|n^{-1/20} + t \leq n^{1/5} + n^{3/10}$.

We conclude that the expected welfare of $\mathcal{A}'_{V,S,T,1}$ on input V (over all events) is at most $n^{6/20}$, whereas a total of $|V| = n^{7/20}$ is possible with allocation $\mathbf{x}^1_S$. 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. □

4. A LOWER BOUND FOR MAKESPAN

We now consider the makespan objective, which differs from the welfare objective in that it is not linear in agent values or allocations, and in that it is a minimization problem rather than a maximization. We will 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 performance measure to expected makespan with respect to the *known* prior distribution.

In a Bayesian setting, the makespan problem can be defined as follows. There are n selfish machines (i.e. agents) which must be allocated a set of m jobs. Each agent has a private parameter or value v_i representing its speed. If machine i is allocated jobs with a total length x_i, the finishing time of machine i is x_i/v_i. The selfish objective of a machine is to maximize its payment minus its finishing time. The makespan of allocation $\mathbf{x}$ given speeds $\mathbf{v}$ is the maximum finishing time of any machine: $\phi(\mathbf{x}, \mathbf{v}) = \max_i \frac{x_i}{v_i}$. The social objective is to output an allocation that *minimizes* the makespan. An instance of the (Bayesian) makespan problem is given by a feasibility constraint $\mathcal{F}$ specifying the set of permissable allocations and a value distribution $\mathbf{F}$ of machine speeds. Given an algorithm $\mathcal{A}$, we use $\overline{\phi}(\mathcal{A})$ to denote its expected makespan with respect to $\mathbf{F}$. Our main result is that any BIC transformation $\mathcal{T}$ must degrade the expected makespan of some algorithms by a polynomially large factor[7].

THEOREM 4.1. *There are constants c_1, c_2 such that, for any polytime black-box BIC transformation $\mathcal{T}$, there is an instance $(\mathcal{F}, \mathbf{F})$ of the makespan problem and an algorithm $\mathcal{A}$ with $\overline{\phi}(\mathcal{A}) \leq c_1$ and $\overline{\phi}(\mathcal{T}(\mathcal{A})) \geq c_2 n^{1/4}$ for n large enough.*

Our proof of this theorem is based on an instance in which machines have just two potential speeds, *fast* or *slow*, drawn uniformly at random, and jobs are of just two types, *long* or *short*. There are a small number of long jobs and a large number of short jobs. One of the many good algorithms for this instance simply assigns one job per machine, trying to give long jobs to fast machines and short jobs to slow ones. So long as the distributions of speeds is typical (i.e., concentrated around the expectation of half fast and half slow machines), there are enough machines of each type and so this algorithm has low expected makespan. On the other hand, the algorithm is highly non-monotone: since there are only a small number of long jobs, the expected allocation of the fast machines is very small, whereas the large number of short jobs means the expected makespan of slow machines is larger (for an appropriate setting of parameters). Any transformation that fixes this non-monotonicity must either increase the allocation to fast machines or decrease the allocation to slow machines. The crux of our argument is that any such fix necessarily increases expected makespan by a polynomially large factor. Intuitively, on a typical input vector of machine speeds, the transformation may either: 1) query the algorithm on a vector in which the speeds of some fast machines are swapped with the speeds of some slow machines, or 2) query the algorithm on a vector that is atypically unbalanced. In case 1), the transformation's hope is that the returned allocation will give short jobs to machines that appear slow but are truly fast while simultaneously not allocating long jobs to machines that appear fast but are truly slow. We show that in order to sufficiently increase the expected allocation of fast machines, the transformation would have to swap a very large number of machine speeds

[7]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 $\delta > 0$.

and so exposes itself to a high risk of giving a long job to a slow machine. Hence the expected makespan of each returned allocation is high. In case 2), our algorithm returns a uniformly random allocation which does significantly increase the allocation to fast machines. However, as before, there is a significant chance that some truly slow machines receive long jobs and drastically increase the makespan of the transformation.

We now provide the details of the proof suggested by this intuition.

Problem Instance: We will consider the following instance $(\mathcal{F}, \mathbf{F})$ of a makespan minimization problem. Let $\alpha < n^{1/2}$ be a parameter to be determined later. The value distribution $\mathbf{F}$ will be the uniform distribution over $\{1, \alpha\}^n$. That is, every value (i.e. speed) v_i is 1 or α with equal probability. There will be $n^{1/2}$ jobs of length α and $n/2 + n^{3/4}$ of length 1. Our feasibility constraint $\mathcal{F}$ will be that each machine can be assigned at most one job. That is, $\mathcal{F} \subseteq \{0, 1, \alpha\}^n$, and a vector in $\{0, 1, \alpha\}^n$ is feasible if and only if it contains $n^{1/2}$ entries with value α and $n/2 + n^{3/4}$ entries with value 1.

The Algorithm: Our algorithm $\mathcal{A}$ is described as Algorithm 2 below; we think of $\mathcal{A}$ as an approximation algorithm for problem instance described above. Roughly speaking, $\mathcal{A}$ behaves as follows. Recall that $\mathcal{A}$ takes as input a vector of speeds $\mathbf{v} \in \{1, \alpha\}^n$. Given input $\mathbf{v}$, write $H(\mathbf{v}) = \{i : v_i = \alpha\}$ for the set of machines that declare the high speed α. The behaviour of $\mathcal{A}$ will depend on the size of set $H(\mathbf{v})$. If $||H(\mathbf{v})| - \frac{1}{2}n| \leq n^{3/4}$ (i.e. if $|H(\mathbf{v})|$ is close to its expectation), then $\mathcal{A}$ will allocate a short job to each machine in $[n] \backslash H(\mathbf{v})$, and will allocate the $\sqrt{n}$ long jobs as well as any remaining short jobs (of which there are $t = \frac{1}{2}n + n^{3/4} - |H(\mathbf{v})| \leq 2n^{3/4}$) to the machines in $H(\mathbf{v})$, uniformly at random. Otherwise, if $|H(\mathbf{v})|$ is not close to its expectation, $\mathcal{A}$ will ignore the reported speeds and return an allocation chosen uniformly at random from all allocations that assign at most one job to each machine. Although we describe $\mathcal{A}$ as a randomized algorithm, we remark that Theorem 4.1 holds even when $\mathcal{A}$ is restricted to be a deterministic algorithm[8].

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

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

LEMMA 4.3. $\overline{\phi}(\mathcal{A}) \leq 1 + 2\alpha e^{-n^{1/4}/4}$.

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 4.2 implies that this latter case occurs with probability at most $2e^{-n^{1/4}/4}$, the result follows. □

[8]In particular, if we consider the ensemble of deterministic algorithms defined by taking all possible settings of the random variables in Algorithm 2, then a straightforward extension of our results shows that, for every transformation, Theorem 4.1 holds for at least one algorithm in the ensemble.

Algorithm 2: Allocation Algorithm $\mathcal{A}$

Input: Vector $\mathbf{v} \in \{1, \alpha\}^n$
Output: An allocation $\mathbf{x} \in \mathcal{F}$

$H \leftarrow \{i : v_i = \alpha\}$;
if $\frac{1}{2}n - n^{3/4} \leq |H| \leq \frac{1}{2}n + n^{3/4}$ **then**
  $t \leftarrow \frac{1}{2}n + n^{3/4} - |H|$;
  Choose set $S \subset H$ with $|S| = n^{-1/2}$ uniformly at random;
  Choose set $T \subset H \setminus S$ with $|T| = t$ uniformly at random;
  for $i \notin H$ **do** $x_i \leftarrow 1$;
  for $i \in S$ **do** $x_i \leftarrow \alpha$;
  for $i \in T$ **do** $x_i \leftarrow 1$;
  for $i \in H \setminus (S \cup T)$ **do** $x_i \leftarrow 0$;
else
  Choose set $S \subset [n]$ with $|S| = n^{1/2}$ uniformly at random;
  Choose set $T \subset [n] \setminus S$ with $|T| = n/2 + n^{3/4}$ uniformly at random;
  for $i \in S$ **do** $x_i \leftarrow \alpha$;
  for $i \in T$ **do** $x_i \leftarrow 1$;
  for $i \notin S, i \notin T$ **do** $x_i \leftarrow 0$;
end
return x

Transformation: 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 $\mathcal{A}$. Recall that we can assume $\mathcal{T}(\mathcal{A})$ can only return an allocation that it observed during black-box queries to algorithm $\mathcal{A}$.

Let p_{bad} denote the probability, over $\mathbf{v}$, $\mathcal{T}$, and $\mathcal{A}$, that $\mathcal{T}(\mathcal{A})$ returns an allocation with makespan α. We will show that if $\mathcal{T}(\mathcal{A})$ is BIC, then p_{bad} must be large; this will imply that the expected makespan of $\mathcal{T}(\mathcal{A})$ will be large as well.

Intuitively, the reason that $\mathcal{A}$ is not BIC is that 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. We will prove that a sufficiently large increase to these probabilities must imply that p_{bad} is large as well.

We first show that, if approximately half of the machines are fast, then $\mathcal{T}(\mathcal{A})$ is unlikely to find an allocation that allocates 0 to many slow machines but allocates long jobs only to fast machines.

LEMMA 4.4. *Suppose* $\mathbf{v}$ *satisfies* $||H(\mathbf{v})| - \frac{1}{2}n| \leq n^{3/4}$, *and let random variable* $\mathbf{x}$ *denote the allocation returned by* $\mathcal{T}(\mathcal{A}, \mathbf{v})$. *Then the probability that* $\phi(\mathbf{x}, \mathbf{v}) < \alpha$ *and at least* $n^{3/4}$ *agents satisfy* $v_i = 1$ *and* $x_i = 0$ *is at most* $(\ln 4)e^{-n^{1/4}/2}$.

PROOF. Let B denote the (bad) event, over randomness in $\mathbf{v}$, $\mathcal{T}$, and $\mathcal{A}$, that $\mathcal{T}(\mathcal{A})$ returns an allocation with makespan

α. 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 $x_i = 0$. Recall that C denotes 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 $\mathbf{v}$.

Fix input $\mathbf{v}$ and suppose that event $C \wedge U(\mathbf{v}) \wedge \neg B$ occurs. Then $\mathbf{v}$ satisfies the event C, and moreover 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_1 denote the number of agents with $v_i = 1$. Then C implies $t_1 \geq n/2 - n^{3/4}$.

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_1 agents has an $n^{-1/2}$ probability of being allocated α (from the definition of $\mathcal{A}$). The probability that none of the t_1 agents is allocated a load of α is at most $(1 - n^{-1/2})^{t_1} < e^{-n^{1/4}}$.

On the other hand, suppose that $\mathbf{v}'$ satisfies the event C. Then $U(\mathbf{v})$ implies that at least $t_2 \geq n^{3/4}$ agents satisfy $v_i = 1$ and $v_i{}' = \alpha$, since only agents for which $v_i{}' = \alpha$ can be allocated 0. In this case, each of these t_2 agents has probability at least $n^{-1/2}$ of being allocated α. The probability that none of them is allocated a load of α is at most $(1 - n^{-1/2})^{t_2} < e^{-n^{1/4}}$.

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. Since the allocations returned by $\mathcal{A}$ on any two queries are independent, we can 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

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

as required. □

Now consider the probability with which $\mathcal{T}$ allocates a given job to an agent. For agent i, value v, and allocation x, 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 agent i.

We can express the fact that $\mathcal{T}$ satisfies BIC in terms of conditions on these probabilities (Lemma 4.5 below): either $p_i^0(1)$ should be large or one of $p_i^1(\alpha)$ and $p_i^\alpha(\alpha)$ should be large.

LEMMA 4.5. *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. □

On the other hand, in Lemmas 4.6, 4.7, and 4.8, we will show that on average over all agents, each of these probabilities ($p_i^0(1)$, $p_i^1(\alpha)$, and $p_i^\alpha(\alpha)$) is small if p_{bad} is small. The proofs of these lemmas are deferred to the end of this section. In Lemma 4.9 we put these results together to argue that p_{bad} must be large.

LEMMA 4.6. $\frac{1}{2n}\sum_i p_i^\alpha(\alpha) \leq n^{-\frac{1}{2}}$.

LEMMA 4.7. $\frac{1}{2n}\sum_i p_i^0(1) \leq n^{-1/4} + p_{bad} + (\ln 4)e^{-n^{1/4}/2}$.

LEMMA 4.8. $\frac{1}{2n}\sum_i p_i^1(\alpha) \leq 3n^{-1/4} + p_{bad} + (\ln 4)e^{-n^{1/4}/2} + 2e^{-n^{1/4}/4}$.

We can now derive a lower bound on p_{bad}: since one of $p_i^0(1)$, $p_i^1(\alpha)$, or $p_i^\alpha(\alpha)$ must be large, the bad event of allocating a long job to a slow machine must occur with non-negligible probability.

LEMMA 4.9. *If $\mathcal{T}$ is a BIC transformation, then $p_{bad} \geq n^{-1/4}(1 - o(1))$.*

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 first that the third inequality is true. Lemma 4.6 implies

$$n^{-\frac{1}{2}} \geq 1/2\alpha$$

which is a contradiction, since $\alpha = \frac{1}{4}n^{1/2}$.
Suppose the first inequality is true. Then by Lemma 4.7 we know

$$p_{bad} \geq 1/18 - n^{-1/4} - (\ln 4)e^{-n^{1/4}/2}$$

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).
Finally, suppose the second inequality is true. Then we know from Lemma 4.8

$$p_{bad} \geq 1/18 - 5n^{-1/4} - 2(\ln 4)e^{-n^{1/4}/2} - 2e^{-n^{1/4}/4}$$

which again implies the desired result. □

To complete the proof of Theorem 4.1, note Lemma 4.9 implies $\overline{\phi}(\mathcal{T}(\mathcal{A})) = 1 + \alpha p_{bad} = \Omega(n^{1/4})$. Since $\overline{\phi}(\mathcal{A}) = \theta(1)$ (Lemma 4.3), $\mathcal{T}$ degrades the expected makespan of $\mathcal{A}$ by a factor of $\Omega(n^{1/4})$.

Proofs of bounds on the allocation probabilities.

To conclude the analysis, we now present proofs of Lemmas 4.6, 4.7, and 4.8.

Proof of Lemma 4.6 : Note that, for any $\mathbf{v}$, exactly $n^{1/2}$ agents are allocated α by $\mathcal{A}$. Thus, keeping $\mathbf{v}$ fixed, we have

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

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

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

□

Proof of Lemma 4.7 : For each input vector $\mathbf{v}$, either event $\neg U(\mathbf{v})$ occurs, event B occurs, event $\neg C$ occurs, or event $C \wedge U(\mathbf{v}) \wedge \neg B$ 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 $\mathbf{v}$ and using Lemma 4.4, we have

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

as required. □

Proof of Lemma 4.8 : 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 $\mathbf{v}'$. Let us consider whether $\mathbf{v}'$ satisfies event C. Note that, as in the proof of Lemma 4.4, the probability that $\mathcal{T}$ is able to find an allocation $\mathbf{v}'$ that does not satisfy event C and for which event B does not occur is at most $e^{-n^{1/4}/2}$. Thus the probability that $\neg B \wedge \neg U(\mathbf{v}) \wedge C$ occurs and moreover $\mathbf{v}'$ does not satisfy event C is at most $(\ln 4)e^{-n^{1/4}/2}$. Given that this (unlikely) event does occur, the number of agents satisfying $v_i = \alpha$ and $x_i = 1$ is at most $n/2 + n^{3/4} < n$.

Next suppose that $\mathbf{v}'$ does satisfy event C. Recall that we wish to bound the number of agents for which $v_i = \alpha$ and $x_i = 1$. Note, from the definition of $\mathcal{A}$, that at most $2n^{3/4}$ agents can have ${v_i}' \neq 1$ and $x_i = 1$. So what remains is to bound the number of agents for which $v_i = \alpha$ and ${v_i}' = 1$. Since $\mathbf{v}$ also satisfies event C, we have

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

Furthermore, $\neg U(\mathbf{v}) \wedge \neg B$ implies that $|H(\mathbf{v}') \backslash H(\mathbf{v})| \leq n^{3/4}$. Combining with the inequalities above, we conclude that $|H(\mathbf{v}) \backslash H(\mathbf{v}')| \leq 2n^{3/4} + n^{3/4} = 3n^{3/4}$. Note that this is a bound on the number of agents such that $v_i = \alpha$ and ${v_i}' = 1$. We conclude that the number of agents such that $v_i = \alpha$ and $x_i = 1$ is at most $3n^{3/4} + 2n^{3/4} = 5n^{3/4}$.

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

$$\begin{aligned}&\frac{1}{n}\sum_i \Pr[(v_i = \alpha) \wedge (x_i(\alpha, \mathbf{v}_{-i}) = 1)] \\ &\leq \frac{1}{n}(5n^{3/4} + n(\ln 4)e^{-n^{1/4}/2}).\end{aligned}$$

Thus, taking probabilities over all $\mathbf{v}$ and all events and using Lemma 4.4, we have

$$\begin{aligned}\frac{1}{2n}\sum_i p_i^0(1) &\leq 5n^{-1/4} + (\ln 4)e^{-n^{1/4}/2} + \Pr[B] \\ &\quad + \Pr[C \wedge U(\mathbf{v}) \wedge \neg B|\mathbf{v}] + \Pr[\neg C] \\ &\leq 5n^{-1/4} + \Pr[B] + 2(\ln 4)e^{-n^{1/4}/2} + 2e^{-n^{1/4}/4}\end{aligned}$$

as required. □

4.1 Symmetric Feasibility Constraints

Throughout this section, we have assumed that a transformation can only return allocations that it observes by querying the given algorithm on some input. While this captures the notion of a black box transformation, in some settings it may be reasonable to assume that the feasibility constraints exhibit some basic structure that could be exploited by a transformation. One such natural assumption is *symmetry*: if an allocation (x_i) is feasible, then for any permutation π of the agents, $(x_{\pi(i)})$ is also feasible. Symmetry occurs in any environment in which agents are treated anonymously, for example. In this section, we question whether one can circumvent the previous negative result in certain environments by allowing the transformation to leverage knowledge of the structure of the feasibility constraints. We will see that a negative result similar to Theorem 4.1 holds even when the transformation can exploit symmetry, if we consider environments in which agent values are not drawn from identical distributions. In other words, symmetry and identically-distributed priors are in some sense complementary. We leave as an open question whether there exist transformations with small loss in makespan for environments with identically distributed agent types and symmetric feasibility constraints.

We first note that if we allow our transformation to assume symmetry, then there is a trivial transformation for Algorithm 2 presented in the previous setting: given an input, the transformation queries the algorithm on that input, sorts the machines in the input by speed (breaking ties lexicographically), sorts the allocations returned by the algorithm by size, and then assigns each machine an allocation in this order (i.e. so that the fastest machines are assigned the greatest loads). This transformation is clearly monotone and preserves the makespan.

Our negative result for symmetric feasibility constraints is based on a construction very similar to the one in the previous section. Each machine will have a slow speed and a fast speed, as well as a small job and a large job. We use an environment with non-identically distributed values to vary the notions of slow and fast and small and large in order to create instances where allocations can not be permuted without significantly increasing the makespan.

Problem Instance.

There are $2n$ machines $m_1, \ldots, m_n$ and $m'_1, \ldots, m'_n$. Machine m_i has speed v_i chosen uniformly from $\{\alpha^{3i}, \alpha^{3i+1}\}$, where we will choose $\alpha = \frac{1}{4}n^{1/2}$. Machine m'_i has speed α^{3i+2}. We refer to the machines m_i as the "type-1" machines and the machines m'_i as the "type-2" machines.

There are $3n$ jobs: one each of size $\alpha^{3i}, \alpha^{3i+1}$, and α^{3i+2} for $1 \leq i \leq n$. Our feasibility constraint allows each machine to be assigned at most three jobs. We will refer to the jobs

of sizes α^{3i} and α^{3i+1} as "type-1" jobs, and the jobs of sizes α^{3i+2} as "type-2" jobs.

The Algorithm.

Our algorithm will be a slight modification of Algorithm 2 from the previous setting; we will only briefly describe the differences. The type-1 jobs and machines correspond to the jobs and machines used in the setting of Algorithm 2. We define the high speed machines to be the set $H = \{i : v_i = \alpha^{3i+1}\}$; note that H consists only of type-1 machines. Whenever Algorithm 2 assigns machine i a job of length α, our modified algorithm will instead assign machine m_i a job of length α^{3i+1}. Similarly, whenever Algorithm 2 assigns machine i a job of length 1, our modified algorithm will instead assign machine m_i a job of length α^{3i}. For each i, we will assign machine m_i' the unallocated jobs among those of size $\alpha^{3i}, \alpha^{3i+1}$, and α^{3i+2}.

The extra power granted to a transformation in the symmetric setting is that it can modify allocations returned by the original algorithm: given an observed allocation (x_i) and permutation π, the transformation can return allocation $(x_{\pi(i)})$. However, note that our algorithm is designed so that, for every input vector (v_i) and associated allocation (x_i), and every $i' > i$, it is the case that $x_{i'}/v_i \geq \alpha$. This implies that, for every observed allocation (x_i) and permutation π that is not the identity, the makespan of allocation $(x_{\pi(i)})$ is always at least α. (This uses the fact that, for every non-identity permutation π, there exists some i such that $\pi(i) > i$). We conclude that a transformation cannot make use of the symmetry assumption to reduce the probability with which it returns allocations with makespan α.

As in the analysis of Algorithm 2, the makespan of this algorithm is dominated by that of the "typical" case in which the number of high speed and low speed machines is approximately balanced. In this typical case, the makespan is constant. The atypical case, which occurs with exponentially small probability by the Chernoff bound, has makespan at most $\alpha = \theta(n^{1/2})$, and hence the expected makespan due to this atypical case is constant. On the other hand, any BIC black-box transformation that performs a sub-exponential number of queries has approximation factor at least α. This is because, as discussed above, the transformation cannot make use of the symmetry assumption without generating allocations with makespan α. Since the transformation cannot make use of symmetry, our previous analysis for the proof of Theorem 4.1 applies and we obtain the following similar impossibility result.

THEOREM 4.10. *There are constants c_1, c_2 such that, for any polytime black-box BIC transformation $\mathcal{T}$ for symmetric problem instances, there is a symmetric instance $(\mathcal{F}, \mathbf{F})$ of the makespan problem and an algorithm $\mathcal{A}$ with $\overline{\phi}(\mathcal{A}) \leq c_1$ and $\overline{\phi}(\mathcal{T}(\mathcal{A})) \geq c_2 n^{1/4}$ for n large enough.*

We note that our construction involves agent values that lie in the range $[1, n^{\theta(n)}]$, and that our approximability gap is polynomial in n. The multiplicative loss in expected makespan is therefore logarithmic in the range of values that can be assumed by the agents. We leave open the question of whether our construction can be modified so that the loss in approximation factor can be made independent of the input value space for symmetric settings.

5. THE LIMITATIONS OF IRONING

As mentioned earlier, BIC transformations can preserve social welfare in expectation through a technique known as ironing. One of the properties of the social welfare objective that allows such a 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. Can such a per-agent ironing approach be applied to other objectives, for example those that are additive across agents (but otherwise non-linear)? In this section we answer this question in the negative: informally, any kind of non-linearity in the objective leads to a loss in approximation that is related to the degree of non-linearity and can be made arbitrarily large.

The ironing approach of [12] and [11] can be described as follows: for each agent i they construct a mapping σ_i from the value space of i to itself, and on input $\mathbf{v}$ return the output of the algorithm on $\sigma(\mathbf{v}) = (\sigma_1(v_1), \sigma_2(v_2), \cdots)$. The mappings σ_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. $x_i(\sigma_i(v_i))$, 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.

(P.1) coupled with (P.2) implies that the transformation is BIC. (P.3) implies that the social welfare is preserved in expectation. Furthermore, 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 the mappings σ_i becomes easy. We call any mapping g that satisfies the properties (P.1) and (P.2) a per-agent ironing.

Our claim is that for any objective function that is non-linear in the agents' allocations, there exists an instance such that for any per-agent ironing suffers a loss in performance that depends on the degree of non-linearity in the function. Our argument make use of randomness in the original (non-truthful) algorithm. This is essentially without loss of generality: from a single agent's point of view, the algorithm's outcome, dependent on others' a priori random values, appears random. Through most of this section we focus on a single agent setting.

We first deal with maximization problems, $\max \mathbf{E}_v[h(x, v)]$, where the function h is an increasing function of the allocations and values. (Note that maximizing a function that is either decreasing in allocations or in values leads to a non-monotone objective.) Our gap construction depends on whether the function displays convexity or concavity. We argue the convex case; The concave case, which is quite similar, is left as an exercise to the reader.

We begin by defining two definitions that the function h must satisfy in order for ironing to fail:

DEFINITION 5.1. *A function h is said to be α-**convex** with respect to a random variable X at v if $E[h(X, v)] \geq \alpha h(E[X], v)$.*

DEFINITION 5.2. *A function h is said to be* α**-responsive** *with respect to v at x if there exists a value v' such that $h(x,v) \geq \alpha h(x,v')$.*

The first definition is a quantitative version of Jensen's inequality and essentially quantifies the function's departure from convexity. The second asserts that the function is "responsive" to the agent's value. For example, consider the function $h(x,v) = vx^2$ defined over $x, v \in [0,1]$ and consider the random variable X that takes on values 0 and 1 with equal probability. Then, $E[h(X,v)] = 0.5v$ and $h(E[x],v) = 0.25v$. Therefore, the function is 2-convex with respect to X at any v. Moreover, it is α-responsive with respect for any $\alpha > 0$ to any v at any x. On the other hand, the function $h(x,v) = v(x+10)^2$ defined over $x, v \in [0,1]$ is not very convex with respect to the random variable X defined above: $E[h(X,v)] = 110.5v$ and $h(E[X],v) = 110.25v$. In fact this function admits a good linear approximation given by $v(21x + 100)$; the ratio between the two functions is at most 1.002 within the range $x, v \in [0,1]$.

It should be immediately evident that both properties are necessary to show that ironing degrades the performance of an approximation algorithm. Indeed, an objective function that is not sensitive to the agents' values can be ironed because no permutation of allocations across values can affect the objective by too much. Likewise, a function that is nearly linear everywhere can be ironed because ironing maximizes a linear approximation to the function. We remark that for our argument it is sufficient for the function to satisfy the two properties over some appropriate interval rather than everywhere.

We are now ready to prove our claim.

THEOREM 5.3. *Let h be a continuous increasing function of x and an increasing function in v. Suppose that there exists a value v and a random variable X in the support of the function such that h is α-convex with respect to X at all values and α^2-responsive with respect to v at any allocation in the support of X. Then 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 $\mathcal{A}$ obtains objective function value a factor of $\Omega(\alpha)$ worse than that of $\mathcal{A}$.*

PROOF. Let $\epsilon = 1/\alpha$. Let $\bar{x} = E[X]$ and $\bar{h}(v) = E[h(X,v)]$. Let $v' < v$ be a value such that $h(x,v) \geq 1/\epsilon^2 h(x,v')$ for all x in the support of X. Let $x' > \bar{x}$ be such that $h(x',v') \leq (1+\epsilon)h(\bar{x},v')$ and $h(x',v) \leq (1+\epsilon)h(\bar{x},v)$. The continuity of h implies that x' exists. The main idea behind the proof is the following. In terms of the objective function h, an allocation of x' is much worse than a randomized allocation of X. However, the expected allocation in the latter case is $\bar{x}$ which is smaller than x'. We will now define an instance where the ironing is forced to replace a randomized allocation of X by an allocation of x', thereby hurting the objective function value.

Consider an instance in which the agent's value is v' with probability $1-\epsilon$ and v otherwise. $\mathcal{A}$ is defined as follows. When the agent reports a value of v', the algorithm allocates x' to the agent. When the agent reports a value of v, the algorithm allocates according to X. Now the expected allocation at v is $\bar{x} < x'$, and therefore ironing is necessary. Suppose that the tranformation maps the value v' to v with probability $z/(1-\epsilon)$ for some z. Then, to preserve the distribution over values, we must have $z \leq \epsilon$ and v must get mapped to v' with probability z/ϵ.

How large does z have to be to fix the non-monotonicity? The new expected allocation at v' is $z/(1-\epsilon)\bar{x} + (1 - z/(1-\epsilon))x'$, while the new expected allocation at v is $(1-z/\epsilon)\bar{x} + x'z/\epsilon$. 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 $(1-\epsilon)h(x',v') + \epsilon\bar{h}(v) > \epsilon\bar{h}(v)$. The new objective function value is given by

$$\begin{aligned}
&(1-\epsilon)\left(\frac{z}{1-\epsilon}\bar{h}(v') + \left(1 - \frac{z}{1-\epsilon}\right)h(x',v')\right) \\
&\quad + \epsilon\frac{z}{\epsilon}h(x',v) + \epsilon\left(1-\frac{z}{\epsilon}\right)\bar{h}(v) \\
&\leq (1-\epsilon^2)\bar{h}(v') + z(1+\epsilon)h(\bar{x},v) + \epsilon\left(1-\frac{z}{\epsilon}\right)\bar{h}(v) \\
&\leq \epsilon^2(1-\epsilon^2)\bar{h}(v) + \epsilon^2(1+\epsilon)\bar{h}(v) + \epsilon^2\bar{h}(v) \\
&< 4\epsilon^2\bar{h}(v)
\end{aligned}$$

Here the first inequality follows from the fact that $h(x',v') \leq (1+\epsilon)h(\bar{x},v') \leq (1+\epsilon)\bar{h}(v')$ and $h(x',v) \leq (1+\epsilon)h(\bar{x},v)$; The second follows from $\bar{h}(v') \leq \epsilon^2\bar{h}(v)$, $h(\bar{x},v) \leq \epsilon\bar{h}(v)$, $z \leq \epsilon$, and $z \geq \epsilon(1-\epsilon)$. This implies that any mapping σ that preserves the value distribution and achieves monotonicity must achieve an approximation fator of $\Omega(1/\epsilon)$. □

We next consider algorithms for minimization problems of the form $\min \mathbf{E}_v[h(x,v)]$, where the function h is non-decreasing in x and non-increasing in v. Once again we focus on the case where h is convex in x; the concave case is similar and we skip it.

THEOREM 5.4. *Let h be a continuous increasing function of x and a decreasing function in v. Suppose that there exists a value v and a random variable X such that h is α-convex with respect to X at v and α-responsive with respect to v at any allocation in the support of X. Then 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 $\mathcal{A}$ obtains objective function value a factor of $\Omega(\alpha)$ worse than that of $\mathcal{A}$.*

PROOF. Let $\epsilon = 1/\alpha$. Let $v' > v$ be a value such that $h(x,v) \geq 1/\epsilon h(x,v')$ for all x in the support of X. Consider an instance where the agent's distribution is uniform over $\{v, v'\}$. Let $\bar{x} = E[X]$. Let $x' > \bar{x}$ be such that $h(x',v) \leq (1+\epsilon)h(\bar{x},v)$. Let $\bar{h}(v) = E[h(X,v)]$.

The algorithm $\mathcal{A}$ is defined as follows. At v, the algorithm returns x'. At v', the algorithm returns X. The expected allocation at v_2 is $\bar{x}$, and therefore the allocation is non-monotone. Suppose that the tranformation maps v to v' with probability z and vice versa. Then it is easy to see that $z \geq 1/2$.

Now let us compute the objective function value. The objective function value of the original algorithm is $1/2h(x',v) + 1/2\bar{h}(v') \leq 1/2(1+\epsilon)\epsilon\bar{h}(v) + 1/2\epsilon\bar{h}(v) = (1+\epsilon/2)\epsilon\bar{h}(v)$.

On the other hand, we can bound from below the objective function value of the transformed mechanism by considering the term corresponding to value v when the allocation is randomized according to X (an event that happens with

probability at least 1/4). The new objective function value is therefore at least $1/4\bar{h}(v)$. This proves the theorem. □

Finally, we remark that while our arguments above focused on settings consisting of a single agent, it is straightforward to extend the argument to settings that involve multiple agents—essentially, we can construct instances where a single agent makes a large contribution to the algorithm's (and transformation's) performance, and the allocations and therefore contributions of other agents to the objective function are fixed. We leave the details to the reader.

6. 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] 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.

[6] 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.

[7] S. Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proc. 42nd ACM Symp. on Theory of Computing*, 2011.

[8] S. Dughmi and T. Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *Proc. 51st IEEE Symp. on Foundations of Computer Science*, 2010.

[9] 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.

[10] G. Goel, C. Karande, and L. Wang. Single parameter combinatorial auctions with partially public valuations. In *Proc. 3rd Intl. Conf. on Algorithmic Game Theory*, 2010.

[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] Z. Huang, L. Wang, and Y. Zhou. Black-box reductions in mechanism design. In *Proc. 14th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 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.