On the Impossibility of Black-Box Transformations in Mechanism Design

Rafael Pass[⋆] and Karn Seth

Cornell University
rafael, karn@cs.cornell.edu

Abstract. A fundamental question in algorithmic mechanism design is whether any approximation algorithm for a single-parameter social-welfare maximization problem can be turned into a dominant-strategy truthful mechanism for the same problem (while preserving the approximation ratio up to a constant factor). A particularly desirable type of transformations—called *black-box transformations*—achieve the above goal by only accessing the approximation algorithm as a black box.
A recent work by Chawla, Immorlica and Lucier (STOC 2012) demonstrates (unconditionally) the impossibility of certain *restricted* classes of black-box transformations—where the tranformation is oblivious to the feasibility constrain of the optimization problem. In this work, we remove these restrictions under standard complexity-theoretic assumptions: Assuming the existence of one-way functions, we show the impossibility of *all* black-box transformations.

1 Introduction

A central area in mechanism design focuses on designing *dominant-strategy truthful mechanisms* that 1) maximize some global objective function (e.g., social welfare) in some feasible outcome space, while 2) incentivizing agents to truthfully report their private values, no matter what everyone else does (that is, being truthful is a dominant strategy).

Our focus is on computational aspects of this task; we restrict to NP-optimization problem (that is, problems for which there exists an efficient procedure to check whether an outcome is feasible,) and are interested in computationally-efficient mechanisms that satisfy the above properties. A fundamental question in algorithmic mechanism design is whether any algorithm for solving some single-parameter social-welfare optimization problem (either exactly or approximately) can be transformed into a dominant-strategy truthful mechanism for the same problem. Ideally, we would like these transformations to be *black-box*—that is, given a description of optimization problem f (i.e., the agents'

[⋆] Work supported in part by a Microsoft Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

utility functions, the global objective function, and the feasibility constraints), they only require black-box access to the algorithm $\mathcal{A}$.

The celebrated Vickrey-Clarke-Groves mechanism gives a strong positive result of this kind for any single-parameter social welfare maximization problem with quasi-linear utilities. It shows how to efficiently convert any *exact* algorithm for a social welfare maximization problem into a dominant-strategy truthful mechanism for the same problem, by appropriately charging agents prices.

However, in many cases, we do not have access to efficient exact algorithms, but only an *approximation algorithm*, which finds an outcome that is within some approximation factor of the optimal outcome. It is well-known that the VCG transformation fails when applied to approximation algorithms [NR00], leaving open the question of whether some other black-box transformation can be used to achieve the above goal:

> *Does there exists an efficient* black-box transformation $\mathcal{T}$ *such that for any (single-parameter) class of social welfare maximization problems* $F \in \mathsf{NPO}$[1], *any instance* $\mathsf{f} \in F$, *any approximation algorithm* $\mathcal{A}$ *for* f, $\mathcal{T}^{\mathcal{A}}(\mathsf{f}, F)(\cdot)$ *is a dominant-strategy truthful mechanism for* f *which preserves the approximation ratio of* $\mathcal{A}$ *(up to a constant factor)?*

For a relaxed version of this problem, considering a Bayesian setting where players' values are sampled from some publicly known distribution and we only require a weaker notion of Bayes Nash truthfulness, we can essentially replicate the success of VCG: work by [HL09,BH11] shows that any approximation algorithm can be efficiently converted into a mechanism that is "Bayes-Nash truthful" for social welfare maximization problems, while preserving the approximation ratio of the original algorithm to within an arbitrarily small factor. ([HKM11] show how to achieve a similar result even in a multi-parameter setting, such as multi-item auctions). But the Bayesian setting makes strong assumptions in terms of the knowledge of both the mechanism designer and the players—in many settings it is unreasonable to assume there is a publicly known distribution over player values.

For the case of dominant-strategy truthfulness, Chawla, Immorlica and Lucier [CIL12] gave an elegant partial negative result. That is, they presented a family of problem instances, together with a corresponding family of approximation algorithms for these instances, and showed that every polynomial time transformation must fail to yield a worst-case approximation-preserving dominant-strategy truthful mechanism[2] on *some* member of this family, given black-box access to the corresponding approximation algorithm. However, their result only applies to a *restricted* class of transformations that on top of having black-box access to the algorithm, have *no* access to feasibility constraint of the problem statement f (they only give the transfomation access to the utility

[1] Recall that NPO is the class of NP-optimization problems.

[2] In fact their negative result also encompassed a slightly weaker notion of truthfulness that arises in the case of randomized mechanisms, namely *truthfulness in expectation* (TIE). A mechanism is TIE if each agent wants to be truthful given knowledge of the other players' values, but not the random coin flips of the mechanism. This notion can be contrasted with full ex-post incentive compatibility (EPIC), where the agents want to be truthful even given the coin flips of the mechanism.

functions of the players). In fact, the [CIL12] analysis actually fails if the transformation knows the feasible space.

In this work, we address the above question with respect to *arbitrary* black-box transformations.

1.1 Our Results

Our central theorem shows how to remove the restrictions on the transformation from [CIL12] by using standard complexity-theoretic hardness assumptions: Assuming the existence of one-way functions, we show the impossibility of *all* black-box transformations.[3] Comparing our results to [CIL12], we rule out all black-box transformation (whereas [CIL12] only rules out restricted classes of transformation that are oblivious to the feasibility constraints). On the other hand, the impossibility result of [CIL12] is *unconditional* (whereas we assume one-way functions). We finally observe that we cannot hope to obtain our result without making complexity-theoretic assumptions (unless we prove strong complexity-theoretic hardness results): ruling our arbitrary black-box transformation requires assuming $\mathsf{NPO} \neq \mathsf{P}$[4]; roughly, this follows from the observation that if $\mathsf{NPO} = \mathsf{P}$, then there exists an exact algorithm for every social-welfare optimization problem, and we can then rely on the VCG mechanism. We defer a full proof of this observation to the full version of the paper.

At a high level, the approach in our impossibility result is to consider a family of problem instances that have their feasibility constraints expressed in an "obfuscated" form. This obfuscated description fully determines the set of feasible allocations, and enables efficiently checking whether a particular allocation is feasible, but the obfuscation makes it computationally hard to find the explicit set of feasible allocations for the problem instance given just this description.

1.2 Overview of the Construction

Before describing our results, let us briefly review the construction of [CIL12]. At a high level, their problem instances consist of choosing two privileged subsets of players, U and $V \subseteq [n]$, with $V \subseteq U$. There are three feasible allocations: a high allocation to U, a low allocation to U, and a "dummy" allocation that gives a tiny amount to all n players. When the players in V have high value, the approximation algorithm $\mathcal{A}$ returns the high allocation on U. However, when the players in U have high value, $\mathcal{A}$ returns the low allocation on U. Since $V \subseteq U$, this arrangement creates an essential non-monotonicity[5] in the output of $\mathcal{A}$: increasing the valuations of players in $U \setminus V$ causes their allocations to decrease.

[3] Our results also apply to black-box TIE transformations.

[4] Our results assume the existence of one-way functions, which imply that $\mathsf{NPO} \neq \mathsf{P}$, but it is a major open question whether $\mathsf{NPO} \neq \mathsf{P}$ implies one-way functions.

[5] It is well known that, in the single parameter setting, if an algorithm produces allocations that are monotone in the valuations of the individual players, then there is a simple pricing rule that converts it into a truthful mechanism. Hence their impossibility result must necessarily use an approximation algorithm that is non-monotone.

Additionally, $\mathcal{A}$ has the property that given an input that produces one of the high or low allocation on U, it is hard to find an input that produces the other kind of allocation. This is the crucial property that causes any transformation $\mathcal{T}$ to fail: given an input that produces one kind of allocation on U, $\mathcal{T}$ must proceed as if the other allocation didn't exist, since it can never encounter it. [CIL12] show that this leads every transformation $\mathcal{T}$ to unfairly punish the players in the set V, leading to a low approximation ratio in the transformed mechanism.

Hiding the feasible allocation set from the transformation is essential to their analysis: the argument for not being able to find the second allocation given the first depends on it. In our construction, we would like the a description of the feasible set to be accessible to the transformation.

One attempt could be to release a cryptographic "commitment" to the feasible sets. Roughly speaking, a commitment scheme can be viewed as the cryptographic analog of a "sealed enveloped". It enables a party to "commit" to a value v in a way that hides it with respect to computationally-bounded (i.e., polynomial-time) parties—this is referred to as the *hiding property* of the commitment. Yet at a later stage the party may reveal the value v by releasing some "decommitment string" and it is garanteed that it can only decommit to the actual value v—this is referred to as the *binding property* of the commitment scheme. A bit more formally, a non-interactive commitment scheme Com is a polynomial-time computable function that given a message m and a random string r outputs a commitment $c = \mathsf{Com}(m, r)$ that determines m, yet hides it with respect to polynomial-time parties; the string r is the decommitment information.

The binding property of the commitment scheme would now ensure that the released description fully determines the set of feasible allocations, while the hiding property would make it computationally infeasible for the transformation to recover the actual feasible set without the decommitment information. However, this solution is unsatisfactory, because it makes it impossible to efficiently check whether a particular allocation is feasible or not (and thus the class of optimization problems wouldn't be in NPO). Given just a set of commitments, there is no efficient way to verify if a particular allocation is the value committed to in one of them.

A natural second approach would be to release an "obfuscated" version of the functionality that checks whether a given allocation is one in the feasible set. In theory, this would allow us to release a description of the feasible allocations, together with an efficient functionality to check whether a given allocation is feasible, and still make it hard for the transformation to explicitly find the feasible sets. However, we do not know how to obfuscate general functionalities [BGI$^+$12]. Although there are results for specific functionalities [Wee05,HRV$^+$07,CRV10,BR13], and also some recent progress for weakened notions of obfuscation [GGH$^+$13], it is not clear how to use these weakened version for our purposes. Finally, even if such obfuscation were possible, we could not longer rely on the result of [CIL12]—this result no longer holds when the black-box transformation can just check whether an allocation is feasible or not.

Our key idea for overcoming these problems is by embedding the [CIL12] instance in larger instance such that the impossibility results still applies even if one can check whether an allocation to the "larger" instance is feasible (but this does not permit checking whether an allocation to the "smaller" embedded instance is feasible). More precisely, we release commitments to each of the feasible allocations, and additionally modify the problem to add in some "dummy" agents with no values, that can be allocated either 1 or 0. An allocation to these players can be interpreted as a binary decommitment string r for one of the released commitments. Thus a feasible allocation now consists of an allocation to the "real" agents, together with an allocation to the dummy agents that shows that the allocation to the real agents is in fact a correct decommitment to one of the released commitments. This scheme now satisfies our requirements: it allows for a public description of the feasibility constraints that enables efficiently checking whether a particular allocation is feasible.

To ensure that the approximation algorithm $\mathcal{A}$ can efficiently find a solution, we provide it with the decommitment information r for all commitments. This will allow $\mathcal{A}$ to compute the feasible allocations efficiently, and thus provide a good approximation. This decommitment information—which we refer to as $\mathcal{A}$'s "trapdoor" (or "secret-sauce")—however, is not publicly released and in particular, cannot be directly accessed by the transformation—since we only consider *black-box* transformations $\mathcal{T}$, $\mathcal{T}$ only gets black-box access to $\mathcal{A}$ and can only access the "secret-sauce" indirectly through the output of $\mathcal{A}$. (As is always the case with black-box separation results, we need to provide the algorithm that the reduction/transformation operates on with some extra "power" (e.g., a trapdoor, or the ability to perform super-polynomial-time computation) that the reduction/transformation itself does not get; see e.g., [IR88]. If we did not do this, we would rule out not only black-box transformations, but also *non-black-box* ones.)

1.3 Paper Layout

The layout of our paper is as follows: In Section 2 we provide preliminaries. Section 3 contains the main results of the paper. In Section 3.2 we give both a description of the family of problem instances and approximation algorithms for those instances. Section 3.3 gives a very high-level idea of the analysis of the construction; the actual proof is deferred to the full version of the paper.

2 Preliminaries

2.1 Notation

Let $\mathbb{N}$ denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \ldots, n\}$. We call a function negligible if it grows more slowly than the inverse of any polynomial. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, $M(x)$ represents the distribution of outputs of $M(x)$ when the random tape is chosen uniformly. By $x \leftarrow S$, we denote an element x is sampled from a distribution S. If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F. To denote the ordered sequence in which the experiments happen we use semicolon, e.g. $(x \leftarrow S; (y, z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot, \cdot)$ denotes a predicate, then $\Pr[x \leftarrow S; (y, z) \leftarrow A(x) : p(y, z)]$ is the probability that the predicate

$p(y, z)$ is true in the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$. The notation $\{(x \leftarrow S; (y, z) \leftarrow A(x) : (y, z))\}$ denotes the resulting probability distribution $\{(y, z)\}$ generated by the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$.

2.2 Optimization Problems

In this work, we restrict ourselves to single-parameter social welfare maximization problems with quasilinear utilities, which is a central class of problems in the mechanism design literature.

In this class of problems we are given an input vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, representing the valuations of each of n players. Each v_i is assumed to be drawn from a known set $V_i \subseteq \mathbb{R}$ corresponding to the possible valuations for player i, and $V = V_1 \times V_2 \times \cdots \times V_n$ is the set of possible input vectors. Each problem also defines allocations $x \in \mathsf{f} \subseteq \mathbb{R}^n$, where f is called the set of feasible allocations. In allocation x, player i is allocated x_i, and that player's utility is $v_i \cdot x_i - p_i$, where p_i is the price charged to player i. The goal in a social welfare maximization problem is to choose $x \in \mathsf{f}$ such that the total social welfare of all players, $\sum_i v_i \cdot x_i$, is maximized. Different sets f and V can be thought of as defining different instances of the problem.

An algorithm $\mathcal{A}$ for the problem defines a mapping from input vectors $\mathbf{v}$ to outcomes $\mathbf{x} \in \mathbb{R}^n$. We use $\mathcal{A}(v)$ to represent the output of $\mathcal{A}$ on input v. If $\mathcal{A}$ is randomized, then $\mathcal{A}(v)$ is a random variable. We may also allow $\mathcal{A}$ to take additional inputs as "hints".

We use OPT_{f} to refer to the optimal algorithm for a social welfare problem f and the social welfare $\phi(\mathbf{x}, \mathbf{v})$, that is, for all $\mathbf{v} \in V$, $OPT_{\mathsf{f}}(\mathbf{v}) = \arg\max_{\mathbf{x} \in \mathsf{f}} \phi(\mathbf{x}, \mathbf{v})$ (for a maximization problem, as in the case of social welfare). For an arbitrary algorithm $\mathcal{A}$, we let $approx_{\mathsf{f}}(\mathcal{A})$ denote the worst-case approximation ratio of $\mathcal{A}$ for f. For a maximization problem, $approx_{\mathsf{f}}(\mathcal{A}) = \min_{\mathbf{v} \in V} \frac{\mathbb{E}[\mathcal{A}(\mathbf{v})]}{OPT_{\mathsf{f}}(\mathbf{v})}$.

An family F of optimization problems is in the class NPO if the following conditions hold:

- For every $\mathsf{f} \in F$, f's feasible allocations are each of size at most polynomial in $|\mathsf{f}|$.
- There exists a verifier Ver such that, for every $\mathsf{f} \in F$, $\mathsf{Ver}(\mathsf{f}, \mathbf{x}) = 1 \iff \mathbf{x}$ is in the set of feasible allocations for f. Additionally, Ver is polynomial-time computable.
- Given the valuations of players, the utility Φ of an allocation is polynomial-time computable (this is the case when Φ is the social-welfare maximization objective).

2.3 Mechanisms

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

A (possibly randomized) mechanism $\mathcal{M}$ is called truthful in expectation (TIE) if each agent maximizes it's expected utility by reporting its value truthfully, regardless of the reports of the other agents (the expectation is over the randomness of the mechanism. That is, $\mathbb{E}[u_i(v_i, \mathbf{v}_{-i})] \geq \mathbb{E}[u_i(b_i, \mathbf{v}_{-i})]$ for all i, for all $b_i \in V_i$ and for all $\mathbf{b}_{-i} \in V_{-i}$. 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 $\mathbf{v}_{-i}$, $\mathbb{E}[x_i(v_i, \mathbf{v}_{-i})]$ is a monotone non-decreasing function of v_i, where again the expectation is over the randomness of the mechanism.

A mechanism $\mathcal{M}$ is ε-TIE if an agent cannot gain more than ε by lying, regardless of the reports of the other agents (the expectation is over the randomness of the mechanism. That is, $\mathbb{E}[u_i(v_i, \mathbf{v}_{-i})] \geq \mathbb{E}[u_i(b_i, \mathbf{v}_{-i})] - \varepsilon$ for all i, for all $b_i \in V_i$ and for all $\mathbf{b}_{-i} \in V_{-i}$.

2.4 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 black-box access to algorithm $\mathcal{A}$. We can think of $\mathcal{T}^{\mathcal{A}}(\cdot)$ as the algorithm $\mathcal{T}$ that runs with multiple black-box accesses to $\mathcal{A}$. We say $\mathcal{T}$ is TIE (resp. ε- TIE) if for every algorithm $\mathcal{A}$, $\mathcal{T}^{\mathcal{A}}(\cdot)$ is TIE (resp. ε-TIE).

We write $\mathcal{T}^{\mathcal{A}}$ for the allocation rule that results when $\mathcal{T}$ runs with black-box access to $\mathcal{A}$. We assume $\mathcal{T}$ is aware of the objective function ϕ and the domain V_i of values for each agent i.

A transformation $\mathcal{T}$ for a family $F \in \mathsf{NPO}$ additionally takes as input an instance $\mathsf{f} \in F$, the utility function Φ and the verifier Ver for the family F. We say $\mathcal{T}$ is TIE for F (resp. ε- TIE) if for every instance $\mathsf{f} \in F$, for every algorithm $\mathcal{A}$ for f, $\mathcal{T}^{\mathcal{A}}(\mathsf{f}, \Phi, \mathsf{Ver})(\cdot)$ is TIE (resp. ε-TIE)

2.5 Commitment Schemes

Recall that, roughly speaking, a non-interactive commitment scheme Com is a polynomial-time computable function that given a message m and a random string r outputs a commitment $c = \mathsf{Com}(m, r)$ that determines m, yet hides it with respect to polynomial-time parties. More formally,

Definition 1 (Commitment Schemes[Gol01]) *A non-interactive commitment scheme* Com *is a polynomial-time computable function satisfying the following two properties:*

- ***computational hiding****: for every pair of message sequences* $\{m_{0,n}\}_{n\in\mathbb{N}}, \{m_{1,n}\}_{n\in\mathbb{N}}$ *such that* $m_{0,n}, m_{1,n} \in \{0,1\}^n$ *for all* $n \in \mathbb{N}$*, for every non-uniform probabilistic polynomial time adversary* A*, there exists a negligible function* $\mu(\cdot)$ *such that for all* $n \in \mathbb{N}$*:*

  $$|\Pr[r \leftarrow \{0,1\}^n : A(\mathsf{Com}(m_{0,n}, r)) = 0] - \Pr[r \leftarrow \{0,1\}^n : A(\mathsf{Com}(m_{1,n}, r)) = 0]| \leq \mu(n)$$

  (In other words, A *should not be able to distinguish between commitments to any pair of messages* $m_{0,n}$ *and* $m_{1,n}$ *except with negligible probability.)*

– ***perfect binding**: there do not exist $m_0, m_1, r_0, r_1 \in \{0,1\}^n$ such that $m_0 \neq m_1$ but $\mathsf{Com}(m_0, r_0) = \mathsf{Com}(m_1, r_1)$. (In other words, no commitment c can be opened to two different values m_0 and m_1.)*

Non-interactive commitment schemes can be constructed based on any one-to-one one-way function (see Section 4.4.1 of [Gol01]).

In many applications—and in particular ours—it, however, suffices to consider a *family* of non-interactive commitments that are parametrized by some initialization string k, and the security properties of the commitment (i.e., hiding and binding) need only hold with all but negligible probability over the choice of k. Such commitment schemes can be constructed based on any "plain" one-way function [Nao91,HILL99]. For ease of notation, in this extented abstract, we here simply assume the existence of non-interactive commitment schemes and note that our construction can be easily modified to work also with any family of non-interactive commitments (and thus be based on any one-way function).

3 Main Result

3.1 Problem definition and Main Theorem

In this section, we will give the main result of this paper:

Theorem 1 (Main Theorem) *Assuming the existence of one-way functions, there exist constants $c, d > 0$, a family $F \in \mathsf{NPO}$ of single-parameter social-welfare optimization problems with verifier Ver, and a sequence of distributions $\{D_n\}_{n \in \mathbb{N}}$ such that D_n is a distribution over pairs $(\mathsf{f}, \mathcal{A})$ where each $\mathsf{f} \in F$, and $\mathcal{A}$ is a polynomial size approximation algorithm for f, such that for any poly-time ε-TIE transformation $\mathcal{T}$ with $\varepsilon \leq 1/n^d$, for all n, with all but negligible probability over $(\mathsf{f}, \mathcal{A}) \leftarrow D_n$, $\frac{approx_{\mathsf{f}}(\mathcal{A})}{approx_{\mathsf{f}}(\mathcal{T}^{\mathcal{A}}(\mathsf{f}, \mathsf{Ver}))} \geq n^c$.*

As mentioned above, our overall approach will be to appropriately modify the construction of [CIL12]—roughly speaking, embedding each instance into a larger instance, which uses cryptographic commitments to release the feasibility constraints to the transformation.

3.2 Construction

We consider instances where there are $3n$ agents. The first n agents have private values v_i drawn from $\{v, 1\}$, where $0 < v < 1$ is a parameter we set below. The remaining $2n$ agents are dummy agents, that all have value 0, however allocating to these agents is necessary in order to satisfy the feasibility constraints f, which we describe below. We refer to the set of the first n agents as Y_1, the second n agents as Y_2, and the third n agents as Y_3. We can therefore interpret an input vector as a subset $y \subseteq Y_1$, corresponding to those of the first n agents that have value 1 (the remaining agents in Y_1 have value v, and all agents in Y_2 and Y_3 have value 0).

Since every player in sets Y_2 and Y_3 always has value 0, we will sometimes omit these players from the input to $\mathcal{A}$, since their valuations are known by default. Accordingly, for every $y \subseteq Y_1$, we define $\mathcal{A}(y)$, $OPT_{\mathsf{f}}(y)$, and so on, as the output of the algorithms on the set of values corresponding to the subset y, with the valuations of the players in Y_2 and Y_3 set to 0.

Also, for $a_1, a_2, a_3 \geq 0$, and $y_1 \subseteq Y_1, y_2 \subseteq Y_2, y_3 \subseteq Y_3$, we will let $(\mathbf{x}_{y_1}^{a_1}, \mathbf{x}_{y_2}^{a_2}, \mathbf{x}_{y_3}^{a_3})$ denote the allocation where each agent $i \in y_1$ is allocated a_1, each agent $j \in y_2$ is allocated a_2, each agent $k \in y_3$ is allocated a_3 and all remaining agents are allocated 0. We will also sometimes refer to allocations in the form (x_y^a, r, r'), where $a \geq 0$, $y \subseteq Y$, and $r, r' \in \{0,1\}^n$. This corresponds to allocating a to every $i \in y$, allocating 1 to every $j \in Y_2$ such that r_j, the jth bit of r, is 1, and also allocating 1 to every $k \in Y_3$ such that r'_k, the kth bit of r', is 1, while allocating 0 to all remaining agents in $Y_1 \cup Y_2 \cup Y_3$.

Problem instances: We now define the family F of problems corresponding to the problem instances. Each instance f will have a trapdoor trap. Intuitively, the set of feasible allocations for f will be computationally difficult to find without knowledge of trap.

Let Com be a non-interactive commitment scheme.[6] Then each instance f in our family will consist of a choice of $\alpha, \gamma \in (0,1)$ with $\gamma < \alpha$, sets $S, T \subseteq Y_1$ of agents, and commitments to the preceding parameters $c_1 = \mathsf{Com}(S, r_S)$, $c_2 = \mathsf{Com}(T, r_T)$, $c_3 = \mathsf{Com}(\alpha, r_\alpha)$, for $r_S, r_T, r_\alpha \in \{0,1\}^n$. We let $\boldsymbol{c} = \langle c_1, c_2, c_3 \rangle$, $\boldsymbol{r} = \langle r_S, r_T, r_\alpha \rangle$. The feasible allocations are:

- $(\mathbf{x}_{[n]}^{\gamma}, 0^n, 0^n)$
- $(\mathbf{x}_{S'}^{1}, r, 0^n)$ such that $c_1 = \mathsf{Com}(S', r)$
- $(\mathbf{x}_{T'}^{\alpha'}, r, r')$ such that $c_2 = \mathsf{Com}(T', r)$ and $c_3 = \mathsf{Com}(\alpha', r')$.

We also require that S and T satisfy the same properties as in [CIL12], namely that they are sufficiently large, and have a sufficiently large intersection. Formally, we define parameters $r \geq 1$, and $t \geq 1$ (which we fix below to functions of n), such that $t \gg r \gg \gamma^{-1} \gg \alpha^{-1}$, and $\frac{t}{\gamma n} \ll 1$. We think of t as a bound on "small" sets, and r as a ratio between "small" and "large" sets. We also use a third set $V \subseteq [n]$ as part of our description. Then, if V, S and T are subsets of $[n]$, we say that the triple V, S, T is *admissible* if:

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

Additionally, for an admissible V, S, T, we let $U = S \cap T$. Then, each problem instance corresponds to a choice of V, S and T and a value α, together with $\boldsymbol{c}$ corresponding to commitments to S, T and α under randomness given by $\boldsymbol{r}$. The description of a problem instance is thus given by

$$\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}} = (\boldsymbol{c}, \gamma)$$

[6] As mentioned above, such schemes can be constructed based on any one-to-one one-way function. But, by slightly modifying our construction, we can also rely on a family of non-interactive commitment schemes which can be based on any "plain" one-way function. We defer the details to the full version.

and the trapdoor for that instance is given by

$$\mathsf{trap}_{V,S,T,\alpha,\boldsymbol{r}} = \langle S, T, \alpha, \boldsymbol{r} \rangle$$

The utility function Φ for each instance is simply the social welfare maximization objective. The verifier Ver, given an instance $\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}}$ and an allocation **x** simply checks whether **x** falls into one of the three feasible types of allocations described above. This corresponds to performing at most three decommitments, implying Ver is efficient.

Note that, given the trapdoor, it is easy to come up with feasible allocations corresponding to $\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}}$: simply use the decommitments in the trapdoor together with α to create any of the three different allocations. However, as we will argue later that, without the trapdoor, it is *infeasible* to construct any allocation other than the first, and this argument will be the heart of the impossibility result. Also, notice that neither f nor trap depend on V. Note also that each of the three feasible allocations is independent of V, and thus it is not strictly necessary to include V in our description of a problem instance. However, we include it for notational convenience, as it is an important component of the approximation algorithm, as we will see shortly.

Algorithm We now give an approximation algorithm for the family of problem instances described above. Our algorithm closely follows the approximation algorithm of [CIL12], and is designed to resist black-box transformation into a mechanism. We note that the algorithm depends on f, and in particular needs to have trap for f baked into it.

Algorithm 1 Allocation algorithm $\mathcal{A}_{V,S,T,\alpha,\boldsymbol{r}}$

```
Input: A subset $y \in Y_1$ or agents with value 1
Output: An allocation valid with respect to $\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}}$
1: if $n_S(y) \geq t$, $n_S(y) \geq \gamma|n|$, and $n_S(y) \geq n_T(y)$ then
2:     return $(\mathbf{x}_S^1, r_S, 0^n)$
3: else if $n_T(y) \geq t$, $n_T(y) \geq \gamma|n|$, and $n_T(y) \geq n_S(y)$ then
4:     return $(\mathbf{x}_T^\alpha, r_T, r_\alpha)$
5: else
6:     return $(\mathbf{x}_{[n]}^\gamma, 0^n, 0^n)$
7: end if
```

As in [CIL12], we define the following functions used by our algorithm given an input $y \in Y_1$:

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

and

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

We also inherit Lemma 3.2 from [CIL12], which guarantees the approximation ratio of the above algorithm. We restate this lemma here:

Lemma 2 $\mathit{approx}_{\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}}}(\mathcal{A}_{V,S,T,\alpha,\boldsymbol{r}}) \geq \alpha/6$

Finally, we also have the following version of Claim 3.3 from [CIL12], closely following their proof, but modified to handle ε-TIE transformations. The lemma guarantees that any mechanism solving one of the problem instances we described must allocate similar amounts to agents in U on inputs $y = U$ and $y = V$.

Lemma 3 *Suppose $\mathcal{A}'$ is a ε-TIE algorithm for $\mathsf{f}_{V,S,T,\alpha,\boldsymbol{r}}$. Let a_U and a_V be the expected allocation to each agent in U in $\mathcal{A}'(U)$ and $\mathcal{A}'(V)$ respectively. Then $a_V - a_U \leq \varepsilon \cdot (|U| - |V|)$*

Proof. Consider any set W with $V \subseteq W \subseteq U$ and $|W| = |V| + 1$. Then, on input W, the expected allocation to the agent in $W \setminus V$ must not decrease by more than ϵ. Since all allocations are constant on U, this means that the expected allocation to each agent in U must not decrease by more than ε. By the same argument, for each W such that $V \subseteq W \subseteq U$, $\mathcal{A}'$ must allocate at least $a_V - \varepsilon \cdot (|W| - |V|)$ to each agent in U, and in particular, this holds for $W = U$. $\square$

3.3 Analysis of Construction

The crux of the analysis of [CIL12] is showing that when $\mathcal{T}^{\mathcal{A}}(\cdot)$ is given input $y = V$, it has difficulty finding the feasible output $\mathbf{x}_T^{\alpha}$, while on input $y = U$, it cannot find the feasible output $\mathbf{x}_S^1$. Being unable to find these allocations, and simultaneously needing to maintain the TIE property, the transformed algorithm is forced to make poor choices. The proof of these result in [CIL12], however, crucially rely on the fact that the the feasible allocations are not revealed to the transformation. In the full version of the paper, we show how to extend their proof to also work when commitments to the feasible allocations are given to the transformation. The key idea is that to analyze the probability of a "bad event" (i.e., that the transformation finds a feasible output in a situation when it shouldn't), we can consider a mental experiment where the actual commitments are replaced with commitments to 0. In this mental experiment we can then rely on the [CIL12] analysis to bound the probability of the bad event, and finally, we rely on the hiding propery of the commitment scheme to argue that the probability of the bad even in the real experiment is also small. Due to lack of space, the actual analysis is omitted.

4 Acknowledgements

We would like to thank Nicole Immorlica for helpful discussions.

References

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BH11] Xiaohui Bei and Zhiyi Huang. Bayesian incentive compatibility via fractional assignments. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 720–733. SIAM, 2011.

[BR13] Zvika Brakerski and Guy N Rothblum. Obfuscating conjunctions. In *Advances in Cryptology–CRYPTO 2013*, pages 416–434. Springer, 2013.

[CIL12] Shuchi Chawla, Nicole Immorlica, and Brendan Lucier. On the limits of black-box reductions in mechanism design. In *Proceedings of the 44th symposium on Theory of Computing*, pages 435–448. ACM, 2012.

[CRV10] Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography*, pages 72–89. Springer, 2010.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. FOCS, 2013.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[HKM11] Jason D Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian incentive compatibility via matchings. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–747. SIAM, 2011.

[HL09] Jason D. Hartline and Brendan Lucier. Bayesian algorithmic mechanism design. *CoRR*, abs/0909.4756, 2009.

[HRV$^+$07] Susan Hohenberger, Guy N Rothblum, Vinod Vaikuntanathan, et al. Securely obfuscating re-encryption. In *Theory of Cryptography*, pages 233–252. Springer, 2007.

[IR88] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *CRYPTO '88*, pages 8–26, 1988.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NR00] Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. In *ACM Conference on Electronic Commerce*, pages 242–252, 2000.

[WCL98] QS Wu, Kun-Mao Chao, and Richard CT Lee. The npo-completeness of the longest hamiltonian cycle problem. *Information processing letters*, 65(3):119–123, 1998.

[Wee05] Hoeteck Wee. On obfuscating point functions. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 523–532. ACM, 2005.