Black-Box Reductions in Mechanism Design

Jessica Taylor – jessica.liu.taylor@gmail.com

March 19, 2014

---

1 Introduction

In their paper *On the Impossibility of Black-Box Transformation in Mechanism Design* (2011) [2], Chawla et al. prove some impossibility results about black-box mechanisms. I will give an overview of the problem, summarize prior work, summarize the results of this paper, go through some proofs, and offer additional commentary and open questions.

In mechanism design, it is necessary to design a mechanism that will ask bidders for their bids (representing how they value different allocations) and then produce an allocation that satisfies some problem-specific feasibility constraint. Chawla et al. consider the following criteria to be necessary for all mechanisms:

1. When bidders bid honestly and bidders' bids come from some distribution known to the mechanism, it must, in expectation, approximately optimize an objective (usually social welface), within a constant factor of the optimal objective value. This criterion is called average-case approximation.

2. If a prior on all bidders' preferences is common knowledge, then some payment rule exists so that honest bidding is a Bayes-Nash equilibrium (Bayesian Information Criterion/BIC).

3. The mechanism must run in polynomial time.

In addition, they look at 2 additional criteria that may be added to these bare minimum criteria:

1. Given honest bids, the mechanism must, in expectation over randomization in the mechanism, approximately optimize the objective within a constant factor (worst-case approximation).

2. If other bidders' bids are unknown and no prior is given, then some allocation rule exists so that bidding honestly yields the highest expected utility regardless of others' bids (Truthful in Expectation/TIE). TIE is a stronger condition than BIC.

Often, it is impossible to satisfy the bare minimum criteria, as the problem does not admit a polynomial-time constant-factor approximation. Therefore, it is appealing to look at black-box reductions from mechanism design to algorithm design. A black-box reduction is a mechanism that is given access to an algorithm that approximately optimizes the objective given valuations. The reduction may make a polynomial number of calls to this algorithm.

Note the subtle difference between average-case approximation and worst-case approximation. Average-case approximation states that the mechanism must approximately optimize the objective in expectation when valuations are drawn from some distribution known to the mechanism. Worst-case approximation states that, regardless of what the valuations are, the mechanism must always approximately optimize the objective in expectation, where expectation is taken over randomness in the mechanism. Using an average-case mechanism usefully requires knowing the distribution over valuations, while using a worst-case mechanism usefully does not.

In addition to looking at black-box reductions for maximizing social welfare, the authors also look at alternative objects. The main alternative objective they look at is makespan: the minimum ratio of allocation to valuation.

2 Previous Work

Hartline and Lucifer (2010) [4] showed that a polynomial-time average-case approximating BIC black-box reduction exists in single-parameter settings. Hartline, Kleinberg, and Malekian (2011) [3] extended this result to multi-parameter settings.

We will focus on the single-parameter setting. To convert an allocation rule to a BIC mechanism, it is sufficient to make a modified allocation rule that is monotone. We will transform each bidder's valuation (replacing v_i with $f_i(v_i)$) so that $f_i(v_i)$ follows the same distribution as v_i. The allocation rule receives the $f_i(v_i)$s rather than v_is. As long as this modified allocation rule is monotone, this is a BIC mechanism. Since bidders' valuations are independent and applying f does not change the distribution, each bidder can reason independently as if no other bidder had their valuations modified. Furthermore, the transformation will not worsen social welfare.

The actual definition of f_i is an ironing of the original distribution over valuations. Single out a bidder i and define f_i as follows. There will be some function g from this bidder's valuation to their expected allocation. If g is non-monotonic, then it is necessary to iron it out so that it is. Specifically, find an interval so that g is non-monotonic on part of this interval. If the bid is within this interval, then resample a different valuation within the interval (proportional to the valuation's prior probability). The result of this is that the new function from valuation to expected allocation (g') is constant on the interval, as the value was thrown away and resampled. If the appropriate interval is selected, this will cause g' to be monotone without changing the distribution of the valuation or reducing social welfare.

This mechanism is not a black-box reduction because it relies on having access to the expected allocation function and being able to find nonmonotonicities in it. One problem is computing payments after we already have a monotone allocation rule. Typically, one would use Myerson's lemma compute payments based on an integral of this allocation rule. However, it is not possible to take the integral of a function provided as a black box. Archer et al. (2003) [1] provide an algorithm to create an unbiased estimate of this integral, which allows one to compute appropriate payments.

To produce an approximately monotone allocation rule, the black-box reduction first discretizes each bidder's valuation. Then, it uses sampling to estimate each bidder's expected allocation as a function of valuation. Finally, it uses ironing on this estimated expected allocation

function to produce an approximately monotone allocation rule. Due to slight nonmonotonicity, this algorithm might not be BIC, but it will be ϵ-BIC, meaning that honest bidding is a ϵ-Bayes-Nash equilibirum.

3 Results

Chawla et al. summarize the existence of black-box reductions optimizing social welfare with the following table:

	Average-case approximation	Worst-case approximation
BIC	Yes (Hartline and Lucifer)	?
TIE	?	No (proven in paper)

Hartline and Lucifer (2010) previously proved the top left result: a black-box reduction from algorithm design to mechanism design exists that runs in polynomial-time, approximates the objective in expectation, and is BIC. The main result of the paper is the "no" in the bottom right: no black-box reduction exists that runs in polynomial time, always approximates the objective, and is TIE.

It is important to note what black-box impossibility theorems prove and what they don't. If no black-box reduction satisfying certain criteria exists, this does not necessarily rule out the statement that, whenever an approximation algorithm for this problem exists, a mechanism satisfying the criteria exists.

The fact that the authors only prove TIE + worst-case approximation impossible is disappointing, because it still leaves open the possibility that TIE + average-case approximation is possible. A polynomial-time mechanism that is TIE and approximates welfare in expectation would be limited by the fact that it requires knowing a distribution over valuations a priori, but it would still be more useful than a BIC mechanism, because it does not require all bidders to believe in this distribution as the prior.

The authors additionally prove a result for makespan: no polynomial-time black-box reduction exists that approximately minimizes makespan in expectation and is BIC. This is a strong result, as average-case approximation and BIC are the weakest criteria considered. It is not completely clear when it is worth doing something other than maximizing social welfare, but this result shows that one particular objective cannot be approximated with a black-box reduction.

The makespan objective is not additive over bidders (i.e. it cannot be expressed as $\sum_{i=1}^{n} h_i(x_i)v_i$). The authors do not prove a similarly general result for an additive objective other than social welfare. However, they do show that one particular strategy for constructing a polynomial-time BIC black-box reduction that maximizes social welfare does not work for other additive objectives. This is a weak result, because other strategies for constructing a black box reduction for additive objectives may still work.

4 Proof for Social Welfare

The paper's main result is that no polynomial-time black-box reduction exists that is TIE and always approximately optimizes welfare. To prove this, the authors construct a set of possible

social welfare problems. Optimizing welfare (even approximately) requires knowing a certain parameter of the problem. The parameter can only be learned if the mechanism makes certain queries to the welfare optimization black box. As it is unlikely to make these queries, it is also unlikely to approximately optimize welfare.

In the constructed family of problems, each bidder has a valuation v_i either equal to v or 1, where $0 < v < 1$. So the input to the black box is a subset $y \subseteq [n]$ of bidders with valuation 1, where $[n] = \{1..n\}$. All feasible allocations take the form that some subset $y \subseteq [n]$ of bidders are allocated a each, written $\mathbf{x}^a_y$.

Let $0 < \gamma < 1$, $r \in \mathbb{N}$, and $t \in \mathbb{N}$ be constants that will be fixed later. We will set $v = t/(\gamma n)$. Then the constructed family of problems is parameterized by sets $V, S, T \subseteq [n]$ and a number $\gamma < \alpha < 1$, where

$$\begin{aligned} |S| &= |T| = r^3 t \\ |S \cap T| &= r^2 t \\ V &\subset S \cap T \\ |V| &= rt \end{aligned}$$

It will be useful to set $U = S \cap T$. There are 3 feasible allocations: $\mathbf{x}^\gamma_{[n]}$, $\mathbf{x}^1_S$, and $\mathbf{x}^\alpha_T$. Note that the feasibile set does not depend on V; instead, V is used to define the allocation algorithm.

Define

$$\begin{aligned} n_T(y) &= |y \cap T| + |y \cap U| \\ n_S(y) &= |y \cap S| + 2|y \cap V| \end{aligned}$$

These mostly track $|y \cap T|$ and $|y \cap S|$ respectively but double or triple count some items, so they will be within a factor of 3 of $|y \cap T|$ or $|y \cap S|$ respectively. There exists an approximation algorithm to approximately maximize welfare:

$$\mathcal{A}_{V,S,T,\alpha}(y) = \begin{cases} \mathbf{x}^1_S & \text{if } n_S(y) \geq t, n_S(y) \geq \alpha|y|, n_S(y) \geq n_T(y) \\ \mathbf{x}^\alpha_T & \text{if } n_T(y) \geq t, n_T(y) \geq \gamma|y|, n_T(y) \geq n_S(y) \\ \mathbf{x}^\gamma_{[n]} & \text{otherwise} \end{cases}$$

The algorithm's approximation factor (the ratio between the welfare of its output and the optimal welfare) is at least $\alpha/6$. The proof given in the paper is somewhat tedious, but the basic idea is simple. If the algorithm returns $\mathbf{x}^1_S$, then the condition ensures that $S \cap y$ contains a reasonably high number of items and not much less than $T \cap y$. If the algorithm returns $\mathbf{x}^\alpha_T$, then the condition ensures that $T \cap y$ contains a reasonably high number of items and not much less than $S \cap y$. If the algorithm returns $\mathbf{x}^\gamma_{[n]}$, then neither $S \cap y$ nor $T \cap y$ is especially large, so $\mathbf{x}^\gamma_{[n]}$ is a reasonable allocation in comparison.

The next important claim in the paper is that, if $\mathcal{A}'$ is a TIE mechanism for $F_{V,S,T,\alpha}$, then the expected allocation to each agent in U must be at least as large in $A'(U)$ as in $A'(V)$. This follows from monotonicity of TIE mechanisms and the fact that all feasible allocations allocate the same amount to each bidder in U:

Start with $y = V$ and consider the amount allocated to each bidder in U according to $\mathcal{A}'(y)$. Now consider adding an additional bidder in U to y. As this bidder's valuation has increased,

due to monotonicity the bidder's allocation has not decreased. But this bidder is in U, so all bidders in U have this same allocation which has not decreased from their previous allocation. More bidders can be added to this set, and the same argument applied, until $y = U$.

Now suppose that $\mathcal{T}$ is a black-box reduction (which may or may not be randomized). We write $\mathcal{A}' = \mathcal{T}(\mathcal{A}_{V,S,T,\alpha})$, which must be TIE. As $\mathcal{T}$ is polynomial time, it must make a polynomial number of calls to $\mathcal{A}_{V,S,T,\alpha}$. In order for $\mathcal{T}$ to learn the correct value of α, it must call $\mathcal{A}_{V,S,T,\alpha}(y)$ for some value y such that $\mathcal{A}_{V,S,T,\alpha}(y) = \mathbf{x}_T^\alpha$. This is because the orginial value of α is unknown, $\mathcal{T}$ can only gain information about α by observing the output of $\mathcal{A}_{V,S,T,\alpha}$ on different inputs, and only $\mathbf{x}_T^\alpha$ provides the exact correct value of α.

A key lemma at this point is that, for all y, $P(\mathcal{A}_{V,S,T,\alpha}(y) = \mathbf{x}_T^\alpha) \leq e^{-O\left(\frac{t}{r+1}\right)}$, where S, V, and α are fixed, and T varies uniformly from all feasible values given S, V.

Define the following variables:

$$\begin{aligned} n_V &= |y \cap V| \\ n_S &= |y \cap (S \setminus V)| \\ n_* &= |y \cap ([n] \setminus S)| \\ m_U &= |y \cap (U \setminus V)| \\ m_T &= |y \cap (T \setminus S)| \end{aligned}$$

n_V, n_S, and n_* are constants that add up to $|y|$. m_U and m_T are random variables because they depend on T. It is easy to see from algorithm definition that $\mathcal{A}_{V,S,T,\alpha}(y) = \mathbf{x}_T^\alpha$ if and only if:

$$m_T + 2n_V + 2m_U \geq t$$

$$m_T + 2n_V + 2m_U \geq \gamma(n_V + n_S + n_*)$$

$$m_T + 2n_V + 2m_U \geq n_S + 3n_V$$

The third inequality implies $m_T + 2m_U \geq n_V$. Substituting n_V for $m_T + 2m_U$ in the first equation yields $3n_V \geq t$ and thus $m_T + 2m_U \geq n_V \geq t/3$. Therefore, $m_T + m_U \geq t/6$.

Now it will be useful to place a probabilistic lower bound on $m_T + m_U$. Let S and V be fixed and consider the possible values for the set T. T can be any set such that $V \subseteq T$, $|T \cap S| = r^2 t$, and $|T| = r^3 t$. Therefore, T will contain every element in V, a uniformly random subset of $S \setminus V$ of size $r^2 t - rt$, and a uniformly random subset of $[n] \setminus S$ of size $r^3 t - r^2 t$. So, T will contain every element in V with probability 1, every element in $S \setminus V$ with probability $\frac{r^2 t - rt}{r^3 t - rt} = \frac{r-1}{r^2-1} = \frac{1}{r+1}$, and every element in $n \setminus S$ with probability $\frac{r^3 t - r^2 t}{n - r^3 t} \leq \frac{r^3 t - r^2 t}{r^5 t - r^3 t} = \frac{r-1}{r^2-1} = \frac{1}{r+1}$.

As a result, as n approaches infinity, Chernoff bounds imply that it becomes highly unlikely (all but $e^{-O(t/(r+1))}$ probability) that $m_T + m_U = |y \cap (T \setminus V)|$ will be any less than twice its expectation (which is at least $\frac{1}{r+1}|y \setminus V| = n_* + n_S$). Strictly speaking, Chernoff bounds only work for sums of independent variables, but these random variables (indicators of whether a particular item intersects with $T \setminus V$) are anticorrelated with each other, so Chernoff bounds should still apply.

So, we have with high probability $n_* + n_S \geq \frac{r}{2}(m_T + m_U)$, which implies:

$$\frac{m_T + 2n_V + 2m_U}{n_V + n_S + n_*} \leq \frac{m_T + 2n_V + 2m_U}{n_S + n_*} \leq \frac{m_T + 2(m_T + 2m_U) + m_U}{n_S + n_*} \leq \frac{6(m_T + m_U)}{n_S + n_*} \leq \frac{12}{r+1} < \gamma$$

which contradicts inequality 2.

A consequence of this lemma is that $A'(V)$ is unlikely to learn the true value of α. Conditioned on knowing V and S (which is more than it knows initially), A' must assign a maximum entropy distribution to T. It must feed $A_{V,S,T,\alpha}$ a sequence of y values. By the lemma, none of these individual y values is at all likely to result in $A_{V,S,T,\alpha}$ returning $\mathbf{x}_T^\alpha$.

It may be pointed out that that A' may discover information about the problem by observing whether A returns $\mathbf{x}_S^1$ or $\mathbf{x}_{[n]}^\gamma$. However, provided that A does not return $\mathbf{x}_T^\alpha$, this provides no information about T or α. This can be seen in 2 cases:

- Suppose $n_S(y) \geq n_T(y)$. Then the deciding factor is whether or not $n_S(y) \geq t \wedge n_S(y) \geq \gamma|y|$. As n_S does not have any dependence on T, this does not provide information about T or α beyond what is provided by knowing S or V.
- Suppose $n_S(y) < n_T(y)$. Then A must return either $\mathbf{x}_T^\alpha$ or $\mathbf{x}_{[n]}^\gamma$. We have assumed that the first cannot happen, so no information is provided.

Induction shows that, no matter what strategy A' uses, no y value it passes to A is likely to provide any information on T or α.

A similar lemma states that, for all y, $P(\mathcal{A}_{V,S,T,\alpha}(y) = \mathbf{x}_S^1) \leq e^{-O(\frac{T}{r+1})}$, where U and T are fixed, and V and S vary from all feasible values given U, T. The proof is quite similar to the previous lemma. Analogously, it implies that, on input U, A' is unlikely to discover the true value of S.

At this point, it will be useful to fix some parameters:

$$\begin{aligned} t &= n^{4/20} \\ r &= n^{3/20} \\ \gamma &= n^{-2/20}\alpha && \in \{n^{-1/20}, 1\} \\ v &= t/(\gamma n) = n^{-14/20} \end{aligned}$$

On input U, $\mathcal{A}'$ cannot allocate more than α in expectation to each agent in U, because it does not know S. So by monotonicity, it must not allocate more than α in expectation to each agent in U on input V either.

Similarly, on input V, $\mathcal{A}'$ will not know α, and so it can only allocate up to $n^{-1/20}$ in expectation to each agent in U. By monotonicity, it can only allocate up to $n^{-1/20}$ in expectation to each agent in U on input U. On input U, its welfare will be at most $|V|n^{-1/20} + t = n^{4/20}n^{3/20}n^{-1/20} + n^{4/20} < 2n^{6/20}$. On the other hand, the allocation $\mathbf{x}_S^1$ yields welfare $n^{7/20}$. This is asymptotically better. When $\alpha = 1$, $\mathcal{A}$ will get an approximation factor of $1/6$, a constant factor. Although it may be pointed out that we don't always get an approximation factor of $1/6$ on all problem instances considered (specifically, we get $\alpha/6$), the black-box reduction does not know the set of problem instances that the algorithm works for, so it must conservatively assume that α may be $n^{-1/20}$.

5 Discussion

The result is only proven for worst-case approximation, not average-case approximation. The authors note that, to extend this impossibility result to average-case approximation, it is nec-

essary to construct a distribution over valuation profiles under which the black-box reduction achieves poor welfare in expectation. However, because the black-box reduction knows the distribution, it can use this information to gain information about the problem. In particular, the proof relies on running the reduction on sets V and U without the mechanism knowing U if it given input V, or V if it is given input U. As a result, extending this result to average-case approximation is difficult.

A significant issue with the proof is that the reduction is only allowed to query the approximation algorithm with valuation profiles of the form $\{v, 1\}^n$, and the approximation algorithm is only guaranteed to approximate the objective if the valuation profile is of this form. However, if the approximation algorithm could accept non-standard valuation profiles and still approximately maximize welfare, then it would be possible to give the algorithm a valuation profile in which a single bidder has an extremely high valuation, and all others have 0. If in fact this bidder is in $S \cup T$, this would force the mechanism to reveal the allocation $\mathbf{x}_S^1$ or $\mathbf{x}_T^\alpha$.

At this point it will be useful to distinguish between problems and problem instances. These definitions are slightly different from those in the paper, as the paper allows problems to restrict valuations to a given set. A problem instance is a set of feasible allocations. A problem is a set of valid parameters plus a mapping from parameters to problem instances. For example, a knapsack problem instance with known weights of items (but unknown valuations) is a problem instance, while a mapping from parameters (weights) to problem instance is a problem. Note that it is trivial to create a problem that contains one particular problem instance, so the following results apply to problem instances and not just problems.

It will be useful to consider a stricter definition of an approximation algorithm. We could require that an approximation algorithm must perform well on all instances of a problem and all valuation profiles. Since the approximation algorithm presented in the paper is only guaranteed to approximately maximize welfare for valid valuation profiles, it would not count under this definition. So it is still an open question whether or not a polynomial-time TIE black-box reduction exists that approximately maximizes welfare, whenever there is an approximation algorithm that approximately maximizes welfare on all instances of a problem and all valuation profiles.

6 White-Box Reductions

As the paper's main proof relies on the black-box nature of the reduction, it is theoretically possible that, whenever a polynomial-time approximation algorithm exists for a problem, a polynomial-time worst-case-approximating TIE (PWATIE) mechanism exists for this problem. After all, for the problem used in the proof, there exists such a mechanism (assuming the mechanism is given as input the parameters specifying the problem instance). Since there are only 3 possible allocations, it is easy to exactly compute the optimal allocation and use VCG for payments.

Suppose it were true that a PWATIE mechanism always exists for a problem when an approximation algorithm exists for the problem. This is a major open quesiton in algorithmic game theory, but it will be useful to imagine what a positive answer would entail. Also assume that we have some mathematical system for proving things about algorithms (say, ZFC). We will now consider white-box reductions, which instead of having access to an approximation

black box, know what the problem is. Then there are 3 possibilities:

1. There exists a PWATIE white-box reduction that takes as input a definition of the polynomial-time-approximable problem (as a set of parameters plus a mapping from parameter to problem instance) in ZFC in addition to parameters specifying the problem instance.

2. (1) is not true, but for every polynomial-time-approximable problem, there exists a provably PWATIE mechanism that takes as input paramaters specifying the problem.

3. For every polynomial-time-approximable problem, there exists a PWATIE mechanism, but there does not always exist a provably PWATIE mechanism.

That is: either a white-box reduction exists, none exists even though there is a provably correct mechanism, or there is no provably correct mechanism. (2) is impossible. Given that there always exists a provably PWATIE mechanism whenever a polynomial-time approximation exists, the following algorithm is a white-box reduction:

- Take as input γ (a ZFC definition of set of possible parameters), f (a ZFC definition of a mapping from parameter to set of feasible outcomes), α (parameters for this problem instance), v (valuations)

- For every string in lexicographic order:

  - If the string encodes a tuple containing a program's source code and a ZFC proof that the program is a PWATIE mechanism for this problem (γ, f), run the program (mechanism) on (α, v) and return the result.

The algorithm is guaranteed to run in polynomial time. As a provably correct mechanism exists, it will be found in a constant number of proof checks. Actually running this mechanism will take polynomial time. It is also PWATIE, as it will always find and run the same PWATIE mechanism regardless of α and v.

Now that (2) is proven impossible, it will be useful to consider cases (1) and (3). How likely you believe (3) is depends on your confidence in ZFC. If (3) is also false, then it may be possible to create a white-box reduction using ZFC descriptions of the problem, but this is not especially practical. Therefore, it will be useful to consider other ways of giving the mechanism information about the feasible set.

7 Feasibility Oracles

We return to black-box reductions, but allow the reduction to query an additional black box in addition to the approximation algorithm: a feasibility oracle. This oracle takes as input some allocation. It may either return some allocation that allocates at least as much to every bidder as the input allocation does, or return null if there is no such allocation. There are many problems that lack efficient optimal allocation algorithms, but do have efficient feasibility oracles. For example, it is easy to construct a feasibility oracle for the knapsack problem (where

valid allocations are 0 if the item is not in the knapsack and 1 if it is): simply check if the items with nonzero allocation fit in the knapsack.

Returning a greater allocation is good if valuations are non-negative. To handle negative valuations, it may be useful to give the feasibility oracle the signs of valuations and have it return a Pareto-superior (or equal) feasible allocation.

For some problems, it is not possible to construct a feasibility oracle. It is sometimes not even possible to find a feasible solution in polynomial time. For example, we could imagine a modified knapsack problem where, in addition to weights and values, items also have shininess. We may define feasible sets of items to be those with under a maximum total weight and above a minimum total shininess. Now, just finding a feasible allocation is an NP-complete problem (in fact, it is equivalent to the ordinary knapsack problem where it is necessary for items to have a total weight under some maximum and a total value above some minimum). Since a feasibility oracle must return a feasible allocation on the input $\mathbf{v} = \mathbf{0}$, there is no polynomial-time feasibility oracle for this problem. On the other hand, if a feasible solution cannot be found in polynomial time, then no polynomial-time approximation algorithm exists either.

For the problem used in the proof, it is easy to see how the black-box reduction could use the feasibility oracle to determine α, T, and S. Single out each bidder and give the feasibility oracle an allocation in which this bidder has an extremely small allocation, and all other bidders are allocated 0. If this bidder is in $S \setminus T$, the feasibility oracle will return $\mathbf{x}_S^1$. If the bidder is in $T \setminus S$, the feasibility oracle will return $\mathbf{x}_T^\alpha$. The proof appears completely impossible to adapt for these black-box reduction with access to feasibility oracles, as it relies on hiding the feasible set from the mechanism.

Unlike a proof-based white-box reduction, a black-box reduction using a feasibility oracle could be very practical, as it would not rely on proof search. Proving that such a reduction does not exist appears to be far harder than proving that no black-box reduction (without access to a feasibility oracle) exists. Whether or not a black-box reduction using a feasibility oracle exists is an important open question.

As before, we can narrow down to a few possible cases, this time related to black-box reductions with feasibility oracles rather than white-box reductions:

1. There exists a PWATIE black-box reduction using a feasibility oracle.
2. Some problem instance exists that admits an approximation algorithm but no PWATIE mechanism.
3. All approximable problem instances have PWATIE mechanisms, but no black-box reduction with access to a feasibility oracle exists. This may indicate that a more powerful oracle is necessary.

All of these cases are plausible.

8 Proof for Makespan

The main objective other than social welfare that the paper covers is makespan:

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

Each bidder will get utility $v_i x_i$ as before, but the goal of the mechanism is now to minimize makespan rather than maximize total utility. One way to interpret this is that v_i is the speed of machine i, x_i is the amount of work allocated to machine i, so that makespan is the maximum time taken by any machine to complete its work.

The authors prove that, for any polynomial-time BIC black-box mechanism, there must be some problem instance and allocation algorithm so that the expected makespan of the mechanism grows asymptotically faster than the expected makespan of the allocation algorithm.

The problem instance is simple. Some parameter $\alpha > 1$ is fixed. Each v_i (speed) is taken to be either 1 or α chosen by a fair independent coin flip. Each machine may be assigned a job of length 0, 1, or α; that is, feasible allocations are in $\{0, 1, \alpha\}^n$. An additional feasibility constraint is that the allocation is not equal to some particular $\mathbf{x}_{bad} \in \{0, 1, \alpha\}^n$. $\mathbf{x}_{bad}$ can be taken to be a parameter of the problem.

The authors define a randomized allocation algorithm $\mathcal{A}(\mathbf{x}_{bad})$. The algorithm proceeds as follows. Let H be the set of fast machines (which have speed α). If the number of fast machines $|H|$ is within $n^{3/4}$ of its expectation $n/2$, then the algorithm assigns a random set of $|H|/\sqrt{n}$ machines in H (chosen uniformly) to do a task of size α, the rest in H to do a task of size 0, and all other machines to do a task of size 1. Otherwise, pick a random set of size $n^{3/4}$ (chosen uniformly) to do tasks of size α, and the rest to do tasks of size 0. If the resulting allocation happens to be $\mathbf{x}_{bad}$, then re-run the algorithm.

Chernoff bounds show that for large n, $|H|$ is likely to be close to its expectation, so the first case will almost always happen (with probability at least $1 - 2e^{-n^{1/4}/4}$. This results in a makespan of 1, because high-speed machines are assigned tasks of size 0 or α and low-speed machines are assigned tasks of size 1. In the second case, the makespan will almost always be equal to α (as a slow machine gets a task of size α), and will never be greater. Therefore, the algorithm's expected makespan is no more than $1 + 2\alpha e^{-n^{1/4}/4}$.

It is important to note that this result holds regardless of which random sets the algorithm selects. Therefore, we can define a set of deterministic algorithms $D(\mathbf{x}_{bad})$ such that $\bigcup_{\mathcal{B} \in D(\mathbf{x}_{bad})} \mathcal{B}(\mathbf{v}) = \text{Support}(\mathcal{A}(\mathbf{x}_{bad})(\mathbf{v}))$. Each $\mathcal{B} \in D(\mathbf{x}_{bad})$ can be interpreted as a way of running $\mathcal{A}(\mathbf{x}_{bad})$ with a constant sequence of random numbers replacing the output of the random number generator. As the bound on expected makespan does not depend on the random number generator used by $\mathcal{A}(\mathbf{x}_{bad})$, the bound applies to all these deterministic algorithms.

Now consider the black-box transformation $\mathcal{T}$, which is given some deterministic $\mathcal{B} \in D(\mathbf{x}_{bad})$. It does not know the value of $\mathbf{x}_{bad}$. The only way it can know that an allocation is safe is if it was returned by the algorithm $\mathcal{B}$. This is because, if it makes some queries to the algorithm and then returns some allocation $\mathbf{x}$ that was not returned by $\mathcal{B}$, then we could set $\mathbf{x}_{bad} = \mathbf{x}$ and observe that the algorithm still has a nonzero probability of returning $\mathbf{x}$. So we can assume that $\mathcal{T}$ only returns allocations returned by its queries to $\mathcal{B}$; otherwise, it would have a nonzero probability of returning an infeasible allocation.

The rest of the proof relies on the fact that, to be monotonic, the algorithm must give $n^{3/4}$ slow machines jobs of size 0 with at least $n^{-1/4}$ probability. However, given this (and that the allocation is the result of querying $\mathcal{B}$), it is very likely that its makespan is α. This can be seen in the 2 different cases. Let $\mathbf{v}'$ be the input to $\mathcal{B}$ that results in $n^{3/4}$ slow machines getting jobs of size 1. If $|H|$ is close to its mean, then $\mathbf{v}'$ must overestimate the speeds of some machines (i.e. $v'_i = \alpha, v_i = 1$) because in this case $\mathcal{B}$ will only ever allocate 0 to a machine if its speed is

α. Probably, at least one of these machines will be given a task of size α. If $|H|$ is far from its mean, then all machines have some chance of being given a job of size α, so probably at least one machine with speed 1 will be given a job of size α.

Actually showing that $n^{3/4}$ slow machines must be given jobs of size 0 with $n^{-1/4}$ probability is quite tedious. Eventually, we will set $\alpha = \frac{1}{4}n^{1/2}$, so that the expected makespan of the original allocation algorithm is $\Theta(1)$ while the expected makespan of the black box reduction is $\Theta(\alpha n^{-1/4}) = \Theta(n^{1/4})$. This is asymptotically worse than optimal.

9 Discussion

With a feasibility oracle (modified to handle negative valuations, as explained previously), the problem considered in the paper is easy. Construct an allocation in which all machines get a job of size 1, and use the feasibility oracle (with negative signs of vs) to find a feasible allocation with lower or equal allocations for all machines. One must exist, as there is only one infeasible allocation. Then, always return this allocation regardless of bids. This guarantees a makespan no more than 1 and is obviously monotonic.

On the other hand, if the feasibility oracle does not handle negative valuations, then the problem is not straightforward. The feasibility oracle can be defined to always return an arbitrary strictly greater allocation than its argument other than $\mathbf{x}_{bad}$, unless its argument is the maximum allocation, in which it returns its argument. This results in it being very difficult to gain information about $\mathbf{x}_{bad}$ by consulting the feasibility oracle. This problem is much less likely to afflict social welfare problems, as Pareto-superior allocations will be strictly better for the objective, unlike in makespan.

Makespan is a weird objective, and it is not clear when it would be used in real-life situations. However, the result implies that it is not possible to construct a black-box reduction mechanism to optimize any possible objective functions even under BIC.

We might want to narrow down a set of objectives that may be approximable. For example, we may consider additive objectives of the form $\sum_{i=1}^{n} h_i(x_i)v_i$. Makespan is not additive, so it is still an open question whether a black-box reduction exists to solve additive objectives. The authors present some weak evidence against this by showing that the proof that some mechanism optimizes welfare under BIC fails if welfare is replaced with another additive objective. However, this is not at all conclusive, since some different way of handling additive objectives could exist.

10 Conclusion

Hartline and Lucifer [4] constructed a black-box reduction from algorithm design to mechanism design in the case of BIC and average-case approximation. Chawla et al. prove that this is impossible in the case of TIE and worst-case approximation. They also prove that, even in the case of BIC and average-case approximation, it is impossible to approximately minimize makespan.

Both proofs relies on hiding information about the feasible set from the reduction. There are a few ways that one could construct a PWATIE reduction even in the face of the paper's main result (and a polynomial average-case approximate TIE reduction for makespan). First,

one could require the approximation algorithm to perform well on all valuation profiles, not just ones in a restricted set. Second, one could create a white-box reduction that takes as input a description of the problem. It is possible to prove that this exists (using proof search) as long as there is always a provably PWATIE mechanism whenever there is a polynomial-time approximation algorithm. Third, one could give the black-box reduction access to a feasibility oracle, which it can use to find feasible allocations. If the reduction has access to a feasibility oracle, then the paper's approach of hiding information from the mechanism will no longer work, and an impossibility proof (if it exists) will take a completely different form.

On the Impossibility of Black-Box Transformations in Mechanism Design contributes important impossibility results that disallow certain black-box reductions, but it still leaves many open questions. There are still multiple avenues of attack that could yield useful reductions.

References

[1] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 205–214, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[2] S. Chawla, N. Immorlica, and B. Lucier. On the impossibility of black-box transformations in mechanism design. *CoRR*, abs/1109.2067, 2011.

[3] J. D. Hartline, R. Kleinberg, and A. Malekian. Bayesian incentive compatibility via matchings. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 734–747. SIAM, 2011.

[4] J. D. Hartline and B. Lucier. Bayesian algorithmic mechanism design. *CoRR*, abs/0909.4756, 2009.