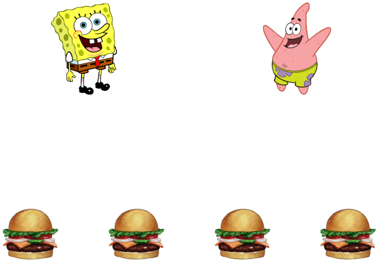
The Complexity of Black-Box Mechanism Design with Priors

Evangelia Gergatsouli[1], Brendan Lucier[2], Christos Tzamos[1]

[1]University of Wisconsin-Madison
[2]Microsoft Research

EC, Phoenix AZ, June 2019

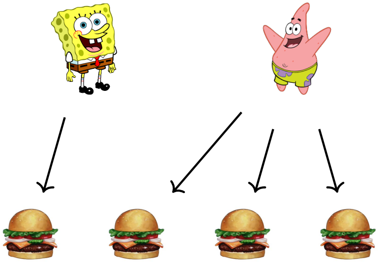
Mechanism Design for Welfare



- n agents, m items
- Agent i has *private* value $v_i(S)$ for set S of items
- Feasibility constraint $\mathcal{F}$ (e.g. at most 2 items per agent)
- Goal: allocate items to maximize welfare

Mechanism $\mathcal{M}$: Given reported values $\mathbf{v}'$ decide:

Mechanism Design for Welfare

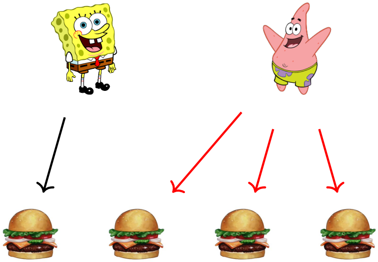


- n agents, m items
- Agent i has *private* value $v_i(S)$ for set S of items
- Feasibility constraint $\mathcal{F}$ (e.g. at most 2 items per agent)
- Goal: allocate items to maximize welfare

Mechanism $\mathcal{M}$: Given reported values $\mathbf{v}'$ decide:

- Allocation rule $\mathcal{A}$: how to give out items

Mechanism Design for Welfare

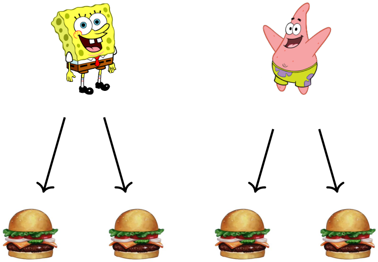


- n agents, m items
- Agent i has *private* value $v_i(S)$ for set S of items
- Feasibility constraint $\mathcal{F}$ (e.g. at most 2 items per agent)
- Goal: allocate items to maximize welfare

Mechanism $\mathcal{M}$: Given reported values $\mathbf{v}'$ decide:

- Allocation rule $\mathcal{A}$: how to give out items

Mechanism Design for Welfare

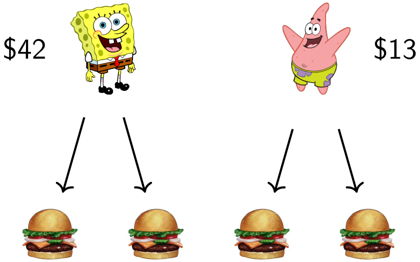


- n agents, m items
- Agent i has *private* value $v_i(S)$ for set S of items
- Feasibility constraint $\mathcal{F}$ (e.g. at most 2 items per agent)
- Goal: allocate items to maximize welfare

Mechanism $\mathcal{M}$: Given reported values $\mathbf{v}'$ decide:

- Allocation rule $\mathcal{A}$: how to give out items

Mechanism Design for Welfare



- n agents, m items
- Agent i has *private* value $v_i(S)$ for set S of items
- Feasibility constraint $\mathcal{F}$ (e.g. at most 2 items per agent)
- Goal: allocate items to maximize welfare

Mechanism $\mathcal{M}$: Given reported values $\mathbf{v}'$ decide:

- Allocation rule $\mathcal{A}$: how to give out items
- Payment rule $\mathcal{P}$: Who pays what

Black-Box Mechanism Design

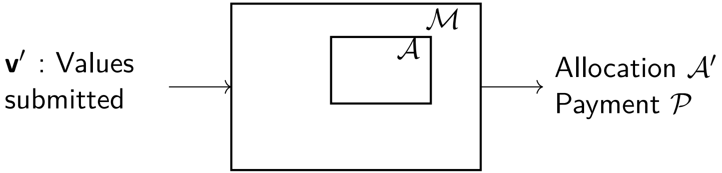
Algorithmic problem: Find allocation rule $\mathcal{A}$ that maximizes welfare

Can we turn this into truthful mechanism?

Black-Box Mechanism Design

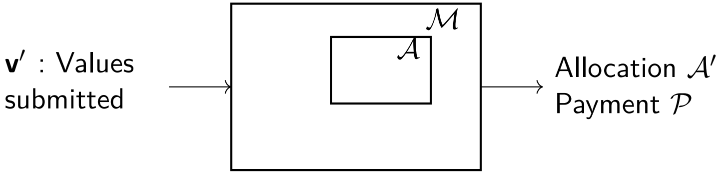
Algorithmic problem: Find allocation rule $\mathcal{A}$ that maximizes welfare

Can we turn this into truthful mechanism?



Black-Box Mechanism Design

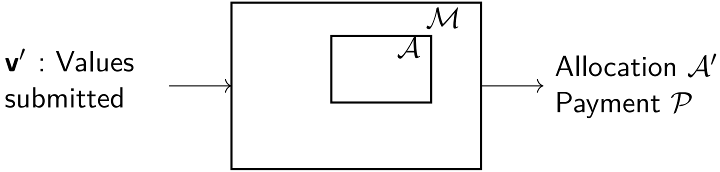
Algorithmic problem: Find allocation rule $\mathcal{A}$ that maximizes welfare
Can we turn this into truthful mechanism?



- If $\mathcal{A}$ maximizes welfare *exactly* in poly-time
  Implement the allocation of $\mathcal{A}$ + charge suitable payments
  $\rightarrow$ VCG is truthful! [Vickrey 1961, Clarke 1971, Groves 1973]
- If $\mathcal{A}$ only *approximately* maximizes welfare

Black-Box Mechanism Design

Algorithmic problem: Find allocation rule $\mathcal{A}$ that maximizes welfare
Can we turn this into truthful mechanism?

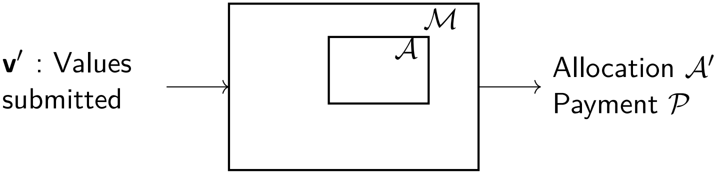


- If $\mathcal{A}$ maximizes welfare *exactly* in poly-time
  Implement the allocation of $\mathcal{A}$ + charge suitable payments
  $\rightarrow$ VCG is truthful! [Vickrey 1961, Clarke 1971, Groves 1973]
- If $\mathcal{A}$ only *approximately* maximizes welfare
  $\rightarrow$ VCG is not truthful!
  $\rightarrow$ There may not exist $\mathcal{P}$ such that $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is truthful.
  $\rightarrow$ Goal: find a modified allocation $\mathcal{A}'$ and truthful payments?

Is designing truthful $\mathcal{M}$ harder than the algorithmic problem?

Black-Box Mechanism Design

Algorithmic problem: Find allocation rule $\mathcal{A}$ that maximizes welfare
Can we turn this into truthful mechanism?



- If $\mathcal{A}$ maximizes welfare *exactly* in poly-time
  Implement the allocation of $\mathcal{A}$ + charge suitable payments
  $\rightarrow$ VCG is truthful! [Vickrey 1961, Clarke 1971, Groves 1973]
- If $\mathcal{A}$ only *approximately* maximizes welfare
  $\rightarrow$ VCG is not truthful!
  $\rightarrow$ There may not exist $\mathcal{P}$ such that $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is truthful.
  $\rightarrow$ Goal: find a modified allocation $\mathcal{A}'$ and truthful payments?

Is designing truthful $\mathcal{M}$ harder than the algorithmic problem?
One possible answer: Black Box reductions!

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

We implement $\mathcal{M}(\mathcal{A}', \mathcal{P})$ s.t.

- Returns feasible allocations
- Have comparable welfare to $\mathcal{A}$
- Is incentive compatible (DSIC/BIC...)

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

Can we implement $\mathcal{M}(\mathcal{A}', \mathcal{P})$ s.t **?**

- Returns feasible allocations
- Have comparable welfare to $\mathcal{A}$
- Is incentive compatible (DSIC/BIC...)

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

Can we implement $\mathcal{M}(\mathcal{A}', \mathcal{P})$ s.t **?**

- Returns feasible allocations
- Have comparable welfare to $\mathcal{A}$
- Is incentive compatible (DSIC/BIC...)

Many different flavours to the problem:

1. Worst Case performance **vs** Average Performance when $v_i \sim \mathcal{D}$

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

Can we implement $\mathcal{M}(\mathcal{A}', \mathcal{P})$ s.t **?**

- Returns feasible allocations
- Have comparable welfare to $\mathcal{A}$
- Is incentive compatible (DSIC/BIC...)

Many different flavours to the problem:

1. Worst Case performance **vs** Average Performance when $v_i \sim \mathcal{D}$
2. Achieving $\mathcal{A}$'s welfare exactly **vs** approximately

Black-Box Mechanism Design

Someone gives us $\mathcal{A}$ that

- Returns feasible allocations
- Has some welfare guarantee

Can we implement $\mathcal{M}(\mathcal{A}', \mathcal{P})$ s.t **?**

- Returns feasible allocations
- Have comparable welfare to $\mathcal{A}$
- Is incentive compatible (DSIC/BIC...)

Many different flavours to the problem:

1. Worst Case performance **vs** Average Performance when $v_i \sim \mathcal{D}$
2. Achieving $\mathcal{A}$'s welfare exactly **vs** approximately
3. Truthfulness: DSIC **vs** BIC (*Bayesian Incentive Compatible* $\rightarrow$ truthful in expectation over other agents reports)

Previous Results

Can we find such a reduction from mechanism design to algorithm design?

Flavours of the problem studied:

preserve **worst case** approx.

- Prior-Free Settings
  - Cannot find reduction to get **DSIC** Mechanism even for single parameter
    [Chawla et al 2012]

preserve **expected welfare** within ε

- Bayesian Settings ($v_i \sim \mathcal{D}$)
  - Can find **BIC** Mechanism, **single**-parameter
    [Hartline, Lucier 2010]
  - Can find ϵ-**BIC** Mechanism, **multi**-parameter
    [Hartline et al 2011 and Bei, Huang 2011]
  - Can find **BIC** Mechanism, **multi**-parameter
    [Dughmi et al 2017]

What's left to do?

- The picture so far
  - ✗ DSIC reduction, worst-case performance, single-parameter
  - ✓ BIC reduction, expected performance, multi-parameter

What's left to do?

- The picture so far
  - ✗ DSIC reduction, worst-case performance, single-parameter
  - ✓ BIC reduction, expected performance, multi-parameter
- Some questions still remain:
  1. *Can we find a "stronger than BIC" reduction that preserves expected welfare, even for single-parameter agents?*

What's left to do?

- The picture so far
  - ✗ DSIC reduction, worst-case performance, single-parameter
  - ✓ BIC reduction, expected performance, multi-parameter
- Some questions still remain:
  1. *Can we find a "stronger than BIC" reduction that preserves expected welfare, even for single-parameter agents?*
  2. Previous BIC results: runtime is polynomial in typespace* size.
     → example: additive agent, with independent values over each item, typespace is exponential.

     Can we avoid runtime dependence on typespace?
     → get a BIC reduction that runs in time poly(n,m)?

*Typespace:
discrete: possible different input profiles

continuous: support size of $\mathcal{D}$

Main Results (Informal)

- ✗ No BIC reduction, even for single additive agent over independent items, with subexponential query complexity
- ✗ No MIDR reduction even for single parameter settings, with subexponential query complexity
  $\rightarrow$ MIDR $\subseteq$ DSIC $\subseteq$ BIC

*✗ $= \mathcal{M}$ **degrades welfare by a polynomial factor** with subexponential queries to $\mathcal{A}$

Main Results (Informal)

- ✗ No BIC reduction, even for single additive agent over independent items, with subexponential query complexity
- ✗ **No MIDR reduction even for single parameter settings, with subexponential query complexity**
  $\rightarrow$ MIDR $\subseteq$ DSIC $\subseteq$ BIC

*✗ $= \mathcal{M}$ **degrades welfare by a polynomial factor** with subexponential queries to $\mathcal{A}$

Up next: intuition for second result.

Lower Bound for MIDR transformations

- Objective: maximize welfare
- Single-parameter setting with n agents
- For every agent: $v_i \in \{0, 1\}$, outcome $\in \{0, 1\}$

Definition 1 (MIDR)

$\mathcal{A}$ is MIDR if for every $\mathbf{v}, \mathbf{v}'$: $\mathbb{E}_{\mathcal{A}(\boldsymbol{v})}[\mathbf{v} \cdot \mathcal{A}(\mathbf{v})] \geq \mathbb{E}_{\mathcal{A}(\boldsymbol{v}')}[\mathbf{v} \cdot \mathcal{A}(\mathbf{v}')]$
Essentially: $\mathcal{A}(\boldsymbol{v})$ is best outcome for $\boldsymbol{v}$ in $\mathcal{A}$'s range.

Lower Bound for MIDR transformations

- Objective: maximize welfare
- Single-parameter setting with n agents
- For every agent: $v_i \in \{0, 1\}$, outcome $\in \{0, 1\}$

Definition 1 (MIDR)

$\mathcal{A}$ is MIDR if for every $\mathbf{v}, \mathbf{v}'$: $\mathbb{E}_{\mathcal{A}(\boldsymbol{v})}[\mathbf{v} \cdot \mathcal{A}(\mathbf{v})] \geq \mathbb{E}_{\mathcal{A}(\boldsymbol{v}')}[\mathbf{v} \cdot \mathcal{A}(\mathbf{v}')]$
Essentially: $\mathcal{A}(\boldsymbol{v})$ is best outcome for $\boldsymbol{v}$ in $\mathcal{A}$'s range.

Theorem 1

For any MIDR black-box transformation $\mathcal{M}$ with sub exponential query complexity there exists an algorithm $\mathcal{A}$ and distribution $\mathcal{D}$ such that $\textsc{Wel}(\mathcal{M}_{\mathcal{A}}) \leq \frac{\textsc{Wel}(\mathcal{A})}{poly(n)}$.

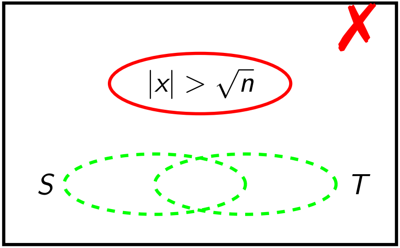
Construction Details

Construction: family of algorithms $\mathcal{A}_{ST}$, distribution $\mathcal{D}$ such that $\mathcal{M}$ degrades welfare

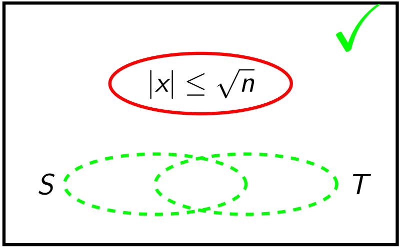
- Input distribution $\mathcal{D}$: $x_i = 1$ w.p. $1/(\sqrt{n})$,
- Uniformly random hidden sets S, T of size $\sim O(\sqrt{n})$ with “big enough“ intersection $|S \cap T|$
- Algorithm $\mathcal{A}_{ST}(x) = x$ or 0 depending on x
  Serve everyone with value 1 **or** serve no one.
  Example:
  $\mathcal{A}_{S,T}(x) = x$: 0010101 → 0010101 ✓
  $\mathcal{A}_{S,T}(x) = 0$: 00**1**0101 → 00**0**0000 ✗

Illustration of Allocation

- If x is too large then $\mathcal{A}_{ST}(x) = \emptyset$
- If x is not too large and has no intersection with S and T, then $A_{ST}(x) = x$



(a)

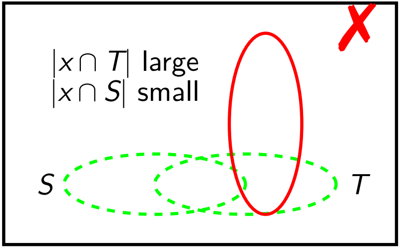


(b)

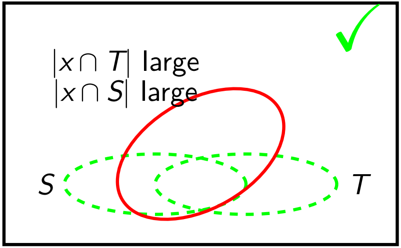
Illustration of Allocation

If x is not too large then:

- If $|x \cap T|$ is large, and $|x \cap S|$ is small then $\mathcal{A}_{ST}(x) = \emptyset$
- else $\mathcal{A}_{ST}(x) = x$



(c)



(d)

Lower Bound - Proof Idea

Lemma 1

$\mathcal{A}$ has high expected welfare ($\Omega(\sqrt{n})$)

Lemma 2

$\mathcal{M}$ has polynomially lower welfare than $\mathcal{A}$ ($O(n^{1/4})$)

Lemma 1 + Lemma 2 $\rightarrow$ Theorem 1

Lower Bound - Proof Idea

Lemma 1

$\mathcal{A}$ has high expected welfare ($\Omega(\sqrt{n})$)

Proof Idea.

$\mathcal{A}_{ST}(x) = x$ almost always, result follows from concentration using construction of S, T and $\mathcal{D}$ □

Lemma 2

$\mathcal{M}$ has polynomially lower welfare than $\mathcal{A}$ ($O(n^{1/4})$)

Lemma 1 + Lemma 2 → Theorem 1

Lower Bound - Proof Idea

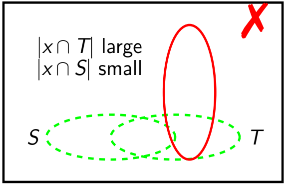
Lemma 2

$\mathcal{M}$ has polynomially lower welfare than $\mathcal{A}$ ($O(n^{1/4})$)

Proof Idea.

Prove in 3 steps:

1. $\mathcal{A}_{ST}(T) = 0$, and $\mathcal{M}$ cannot find set S with subexponentially many samples, so it **cannot find an output with high welfare for T**.



(c)

Note: we don't use any truthfulness constraint here

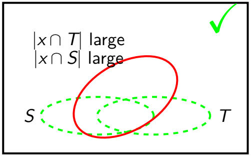
Lower Bound - Proof Idea

Lemma 2

$\mathcal{M}$ has polynomially lower welfare than $\mathcal{A}$ ($O(n^{1/4})$)

Proof Idea.

2. **Idea**: Because of MIDR, $\mathcal{M}_{\mathcal{A}}(S)$ can't return an outcome with high welfare for T. However, on input S, we cannot find $T \rightarrow$ must reduce welfare throughout.



(d)

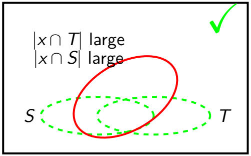
Lower Bound - Proof Idea

Lemma 2

$\mathcal{M}$ has polynomially lower welfare than $\mathcal{A}$ ($O(n^{1/4})$)

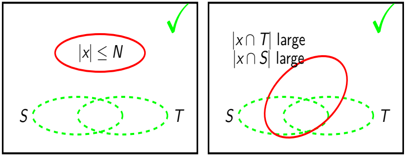
Proof Idea.

2. **Idea**: Because of MIDR, $\mathcal{M}_{\mathcal{A}}(S)$ can't return an outcome with high welfare for T. However, on input S, we cannot find $T \rightarrow$ must reduce welfare throughout.



(d)

3. $\mathcal{M}_{\mathcal{A}}(x)$ gives low welfare (for any input x) **Idea**: Cannot decide if x is the set S or not



(b) (d)

Second Result

- Objective: maximize $\textsc{Wel}$
- Single additive agent, n items

Theorem 2

For any DSIC black-box transformation $\mathcal{M}$ with sub exponential query complexity, there exists an algorithm $\mathcal{A}$ and distribution $\mathcal{D}$ such that $\textsc{Wel}(\mathcal{M}_{\mathcal{A}}) \leq \frac{\textsc{Wel}(\mathcal{A})}{poly(n)}$

Note: for single agent, DSIC = BIC $\Rightarrow$ same result for BIC

Proof follows similarly to MIDR reduction, **but**

- Instead of MIDR condition, uses a characterization of BIC allocation rules due to [Hartline, Kleinberg, Malekian 2011]

Conclusion and Open Problems

Black-box reductions:

reducing mechanism design to algorithm design.

Existing black-box reductions are for BIC mechanisms, and have polynomial dependence on typespace

We showed two negative results

- Remove polynomial dependence on typespace
  ✗ Even for single additive agent over independent types

- Change BIC requirement $\rightarrow$ strengthen BIC to MIDR
  ✗ Even for single parameter settings

Conclusion and Open Problems

Black-box reductions:

reducing mechanism design to algorithm design.

Existing black-box reductions are for BIC mechanisms, and have polynomial dependence on typespace

We showed two negative results

- Remove polynomial dependence on typespace
  ✗ Even for single additive agent over independent types
- Change BIC requirement $\rightarrow$ strengthen BIC to MIDR
  ✗ Even for single parameter settings

Open problem: Can we strengthen BIC to DSIC, while preserving expected welfare? [Chawla et al 2012]

Conclusion and Open Problems

Black-box reductions:

reducing mechanism design to algorithm design.

Existing black-box reductions are for BIC mechanisms, and have polynomial dependence on typespace

We showed two negative results

- Remove polynomial dependence on typespace
  ✗ Even for single additive agent over independent types

- Change BIC requirement $\rightarrow$ strengthen BIC to MIDR
  ✗ Even for single parameter settings

Open problem: Can we strengthen BIC to DSIC, while preserving expected welfare? [Chawla et al 2012]

Thank you!