Sample-efficient black-box reductions in **Bayesian mechanism design** with **independent items**

Arya Maheshwari
MIT



S. Matthew Weinberg
Princeton



Eric Xue
Princeton

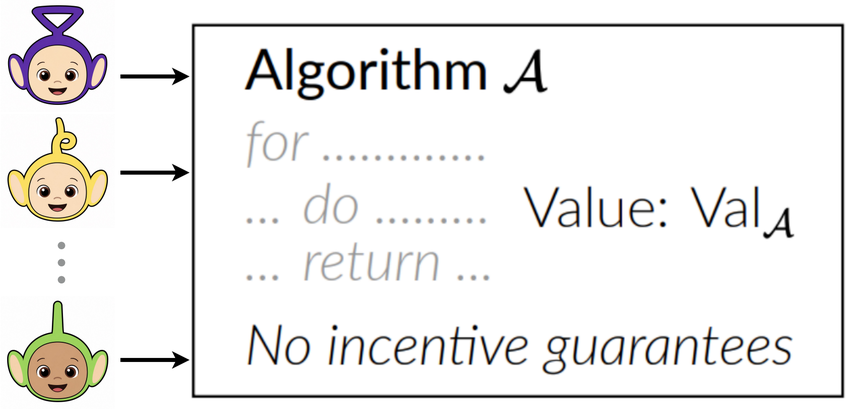


Sample-efficient black-box reductions in Bayesian mechanism design with independent items

Sample-efficient black-box reductions in Bayesian mechanism design with independent items

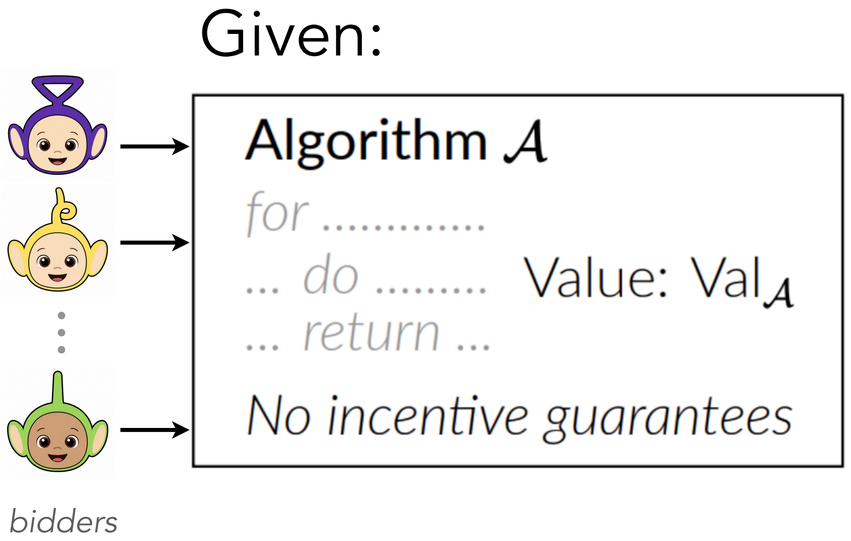
Black-box Reductions in Mechanism Design

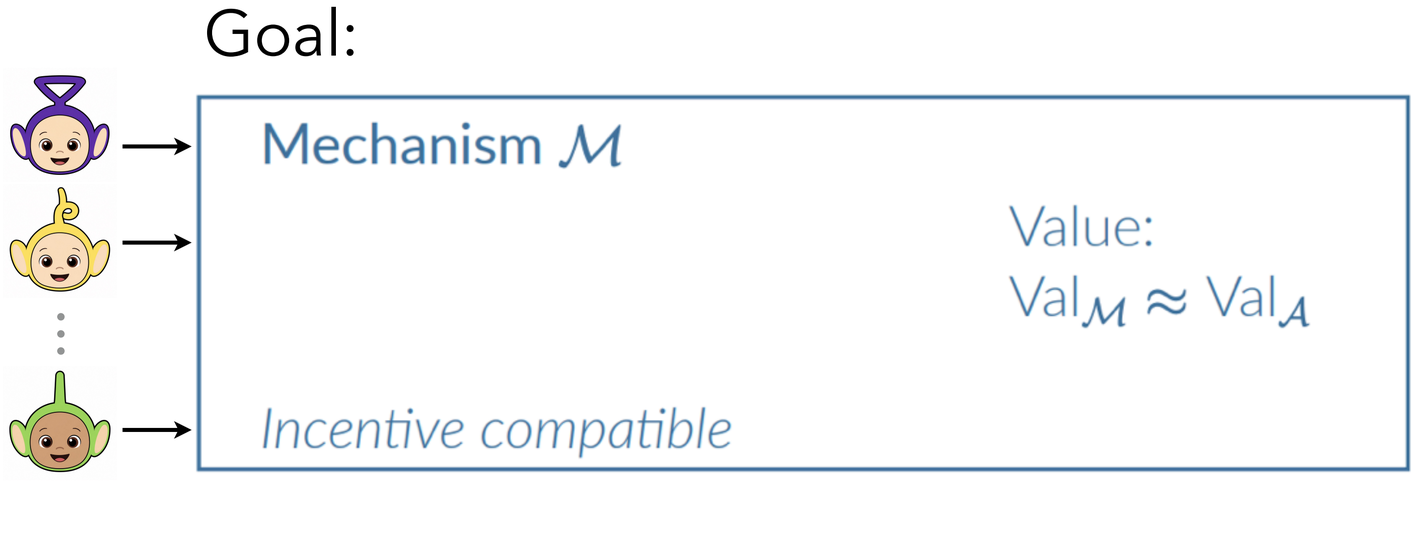
Given:



bidders

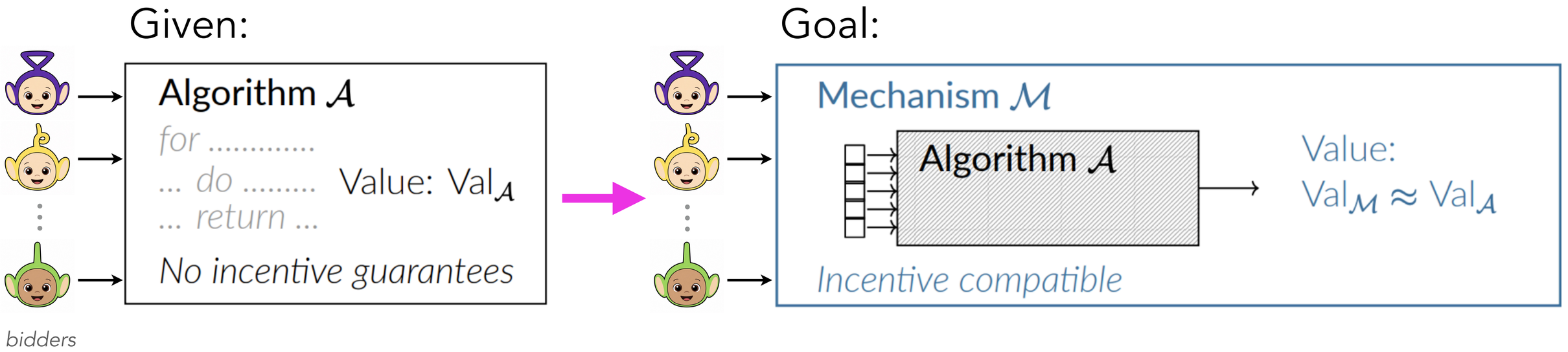
Black-box Reductions in Mechanism Design





Incentive compatibility = in each bidder's interest to report truthfully

Black-box Reductions in Mechanism Design



Incentive compatibility = in each bidder's interest to report truthfully

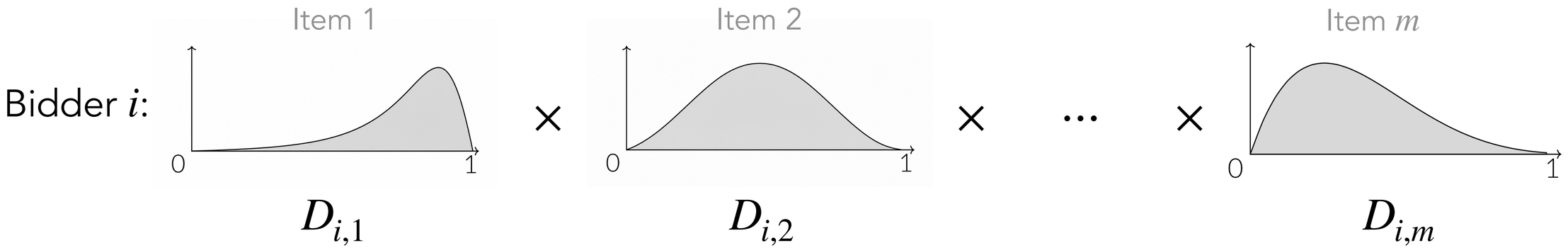
Morally: what is the "price of incentive compatibility"?

Sample-efficient black-box reductions in **Bayesian mechanism design** with **independent items**

Bayesian Mechanism Design

- n bidders, m items
- Bidder i has **type** $t_i \in$ type space $T_i \subseteq [0,1]^m \quad \forall i \in [n]$ $\qquad T := \times_i T_i$
- Prior D_i s.t. $t_i \sim D_i$ independently. **Independent items:** each $D_i = \times_{j=1}^{m} D_{i,j}$

(for this talk: **additive** bidders: bidder i's value for $S \subseteq [m] = \Sigma_{j \in S} t_{i,j}$)



Bayesian Mechanism Design

- n bidders, m items
- Bidder i has **type** $t_i \in$ type space $T_i \subseteq [0,1]^m \quad \forall i \in [n] \qquad T := \times_i T_i$
- Prior D_i s.t. $t_i \sim D_i$ independently. **Independent items:** each $D_i = \times_{j=1}^{m} D_{i,j}$

- **Mechanism** M: $T \to (\Delta(O), \Delta(\mathbb{R}^n))$: types $\mapsto$ (outcome, payments)
- Maximize **expected revenue** $\mathrm{Rev}(M; D)$ from mechanism M over $t \sim D$

Bayesian Mechanism Design

- n bidders, m items
- Bidder i has **type** $t_i \in$ type space $T_i \subseteq [0,1]^m \quad \forall i \in [n] \qquad T := \times_i T_i$
- Prior D_i s.t. $t_i \sim D_i$ independently. **Independent items:** each $D_i = \times_{j=1}^{m} D_{i,j}$
- **Mechanism** M: $T \to (\Delta(O), \Delta(\mathbb{R}^n))$: types $\mapsto$ (outcome, payments)
- Maximize **expected revenue** $\mathrm{Rev}(M; D)$ from mechanism M over $t \sim D$
- Much of past work: **distribution D is known** [Mye81, CHK07, CHMS10, HN17, LY13, BILW20, RW18, …]

Bayesian Mechanism Design

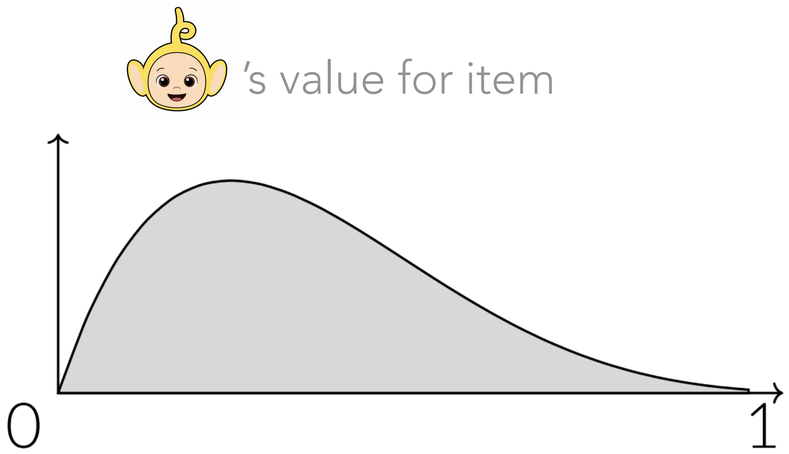
- n bidders, m items
- Bidder i has **type** $t_i \in$ type space $T_i \subseteq [0,1]^m \quad \forall i \in [n] \qquad T := \times_i T_i$
- Prior D_i s.t. $t_i \sim D_i$ independently. **Independent items:** each $D_i = \times_{j=1}^{m} D_{i,j}$
- **Mechanism** M: $T \to (\Delta(O), \Delta(\mathbb{R}^n))$: types $\mapsto$ (outcome, payments)
- Maximize **expected revenue** $\mathrm{Rev}(M; D)$ from mechanism M over $t \sim D$
- Much of past work: **distribution D is known** [Mye81, CHK07, CHMS10, HN17, LY13, BILW20, RW18, …]

But how does the mechanism designer actually form this prior D?

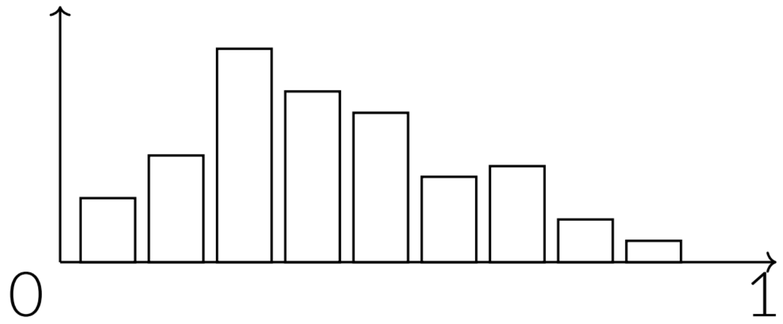
Model: **Sample Access** to Type Distribution

More realistic: **limited sample access** to D

e.g., data from previous auctions



Ideal world: **full knowledge of each D_i**



Real world: **limited # of samples**

Model: **Sample Access** to Type Distribution

More realistic: **limited sample access** to D

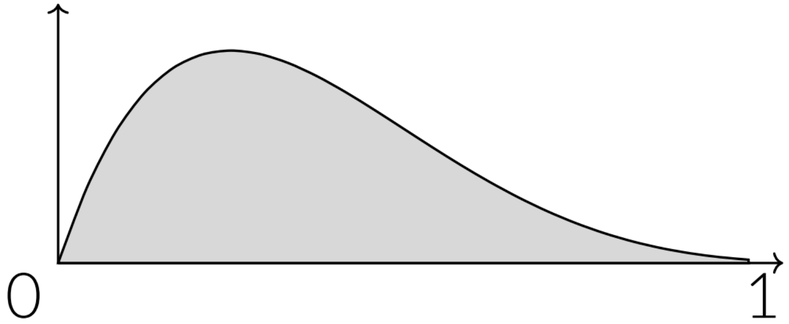
e.g., data from previous auctions

⇒ **mechanism design from samples**: introduced in [CR14], now lots of work

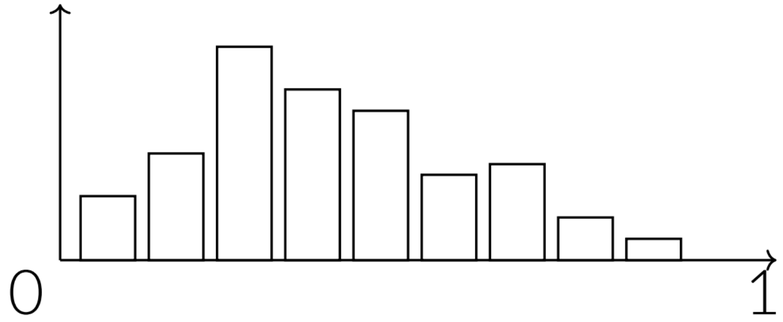
[CR14, DHN14, DHP16, RS16, MR16, BSV16, GN17, Syr17, GW18, GHZ19, GHTZ21, …]



's value for item



Ideal world: **full knowledge of each D_i**



Real world: **limited # of samples**

Sample-efficient black-box reductions in Bayesian mechanism design with independent items

Motivation: **Poly-Sample BIC Auction Learning**

Motivation: **Poly-Sample BIC Auction Learning**

Existing Result [GW18]

For all $\varepsilon, \delta > 0$, the **sample complexity of learning** (w.p. 1-δ) an $O(\epsilon)$-**Bayesian Incentive Compatible, up-to-ε revenue-optimal mechanism** for n bidders with independent values for m items is $\mathrm{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

Motivation: **Poly-Sample BIC Auction Learning**

Existing Result [GW18]

For all $\varepsilon, \delta > 0$, the **sample complexity of learning** (w.p. 1-δ)

☹️ an $O(\epsilon)$-**Bayesian Incentive Compatible, up-to-ε revenue-optimal mechanism** for n bidders with independent values for m items is $\mathrm{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

Motivation: **Poly-Sample BIC Auction Learning**

Existing Result [GW18]

For all $\varepsilon, \delta > 0$, the **sample complexity of learning** (w.p. 1-δ)

☹️ an $O(\epsilon)$-**Bayesian Incentive Compatible, up-to-ε revenue-optimal mechanism** for n bidders with independent values for m items is $\text{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

Bayesian Incentive Compatibility (BIC): for each i, truthfully reporting type t_i is no worse than any untruthful report t_i' in expectation (over $t_{-i} \sim D_{-i}$)

- ϵ-**BIC:** can gain at most ϵ by misreporting

Motivation: **Poly-Sample BIC Auction Learning**

Existing Result [GW18]

For all $\varepsilon, \delta > 0$, the **sample complexity of learning** (w.p. 1-δ)

☹️ an $O(\epsilon)$-**Bayesian Incentive Compatible, up-to-ε revenue-optimal mechanism** for n bidders with independent values for m items is $\mathrm{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

What we'd love: a black-box reduction to convert

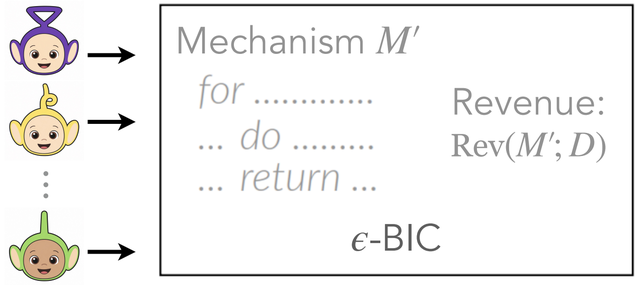
ε-BIC $M' \rightarrow$ exactly-BIC M

"**ε-BIC-to-BIC reduction**" [DW12, RW18, COVZ19]

Problem Statement

Given:

- Query access to ε-Bayesian Incentive Compatible mechanism M' for D
- Sample access to $D = \times_{i,j} D_{i,j}$ (bidders $i \in [n]$, items $j \in [m]$)

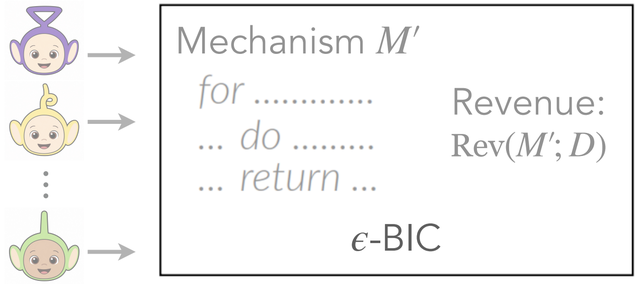




Problem Statement

Given:

- Query access to ε-Bayesian Incentive Compatible mechanism M' for D
- Sample access to $D = \times_{i,j} D_{i,j}$ (bidders $i \in [n]$, items $j \in [m]$)

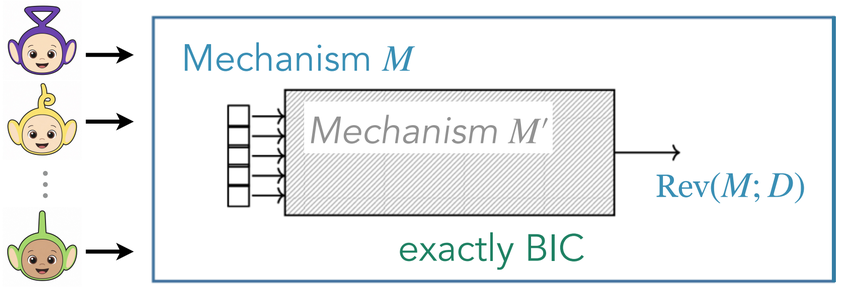




Output:

Exactly BIC mechanism M for D with revenue close to that of M': $\mathrm{Rev}(M; D) \gtrsim \mathrm{Rev}(M'; D)$

with **efficient sample complexity.**



Challenge: **Zero Error** with Only **Sample Access**

Can “learn” product distribution D by “learning marginals” $\{D_{i,j}\}_{i\in[n],j\in[m]}$ …

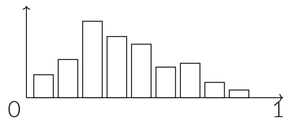
Challenge: **Zero Error** with Only **Sample Access**

Can "learn" product distribution D by "learning marginals" $\{D_{i,j}\}_{i\in[n],j\in[m]}$ ⋯

But **sample access** ⇒ learning will have *some error* (think: (ϵ, δ)−PAC)

yet want to ensure **exact BIC** — *zero tolerance for error!*

(intuitively: want to know 

to ensure algorithm is BIC, but only have 

)

Challenge: **Zero Error** with Only **Sample Access**

Can "learn" product distribution D by "learning marginals" $\{D_{i,j}\}_{i\in[n],j\in[m]}$ ⋯

But **sample access** ⇒ learning will have *some error* (think: (ϵ, δ)−PAC)

yet want to ensure **exact BIC** — *zero tolerance for error!*

Prior work*:

- Black-box reductions with *full knowledge* of D [HKM11, BH11, DW12, RW18]
- Black-box reductions with only sample access to D [DHKN17, COVZ21]

  but exponential sample complexity: poly$(n, \exp(m), 1/\epsilon)$ ☹️

*in more general models, but still require exponential sample complexity/full knowledge even in the independent items setting

Challenge: **Zero Error** with Only **Sample Access**

Can "learn" product distribution D by "learning marginals" $\{D_{i,j}\}_{i\in[n],j\in[m]}$ ⋯

But **sample access** ⇒ learning will have *some error* (think: (ϵ, δ)−PAC)

yet want to ensure **exact BIC** — *zero tolerance for error!*

Question: is poly$(n, m, 1/\epsilon)$ sample complexity achievable? [GW18]

Challenge: **Zero Error** with Only **Sample Access**

Can "learn" product distribution D by "learning marginals" $\{D_{i,j}\}_{i\in[n],j\in[m]}$ ⋯

But **sample access** ⇒ learning will have *some error* (think: (ϵ, δ)−PAC)

yet want to ensure **exact BIC** — *zero tolerance for error!*

Question: is poly$(n, m, 1/\epsilon)$ sample complexity achievable? [GW18]

Our main result: **yes!**

Main Result: **Poly-Sample Black-box Reduction**

Given:

- Sample access to $D = \times_{i,j} D_{i,j}$ (bidders $i \in [n]$, items $j \in [m]$)
- Query access to ε-Bayesian Incentive Compatible mechanism M' over D

Main Result: **Poly-Sample Black-box Reduction**

Given:

- Sample access to $D = \times_{i,j} D_{i,j}$ (bidders $i \in [n]$, items $j \in [m]$)
- Query access to ε-Bayesian Incentive Compatible mechanism M' over D

Main Theorem (informal)

We can construct an **exactly BIC** mechanism M for D with revenue

$$\mathrm{Rev}(M; D) \geq \mathrm{Rev}(M'; D) - O(n\sqrt{\epsilon})$$

using only $\mathrm{poly}(n, m, 1/\varepsilon)$ samples from D.

Corollary: **Poly-Sample BIC Auction Learning**

Corollary (Main Theorem + [GW18])

For all $\varepsilon, \delta > 0$, the **sample complexity of learning** (w.p. 1-δ)
an **exactly BIC, up-to-ε revenue-optimal mechanism**
for n bidders with independent values for m items is

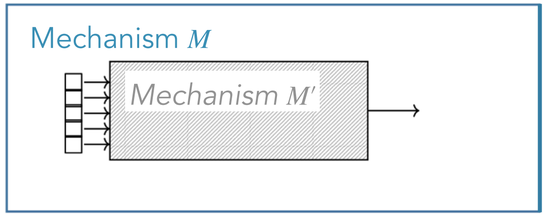
$$\text{poly}(n, m, 1/\varepsilon, \log(1/\delta))$$

Roadmap

- **Main Tool:** Replica-Surrogate Matching
- **Three Algorithmic Ideas**

Background: **Replica-Surrogate Matching** [HKM11]

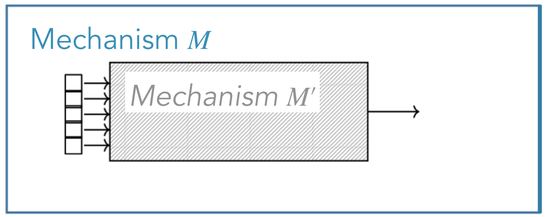
Goal: use process of participating in M' as a black-box



We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Goal: use process of participating in M' as a black-box



Create an interface layer around M', separately for each bidder:

Each bidder i will **purchase a "surrogate" to represent her in M'**, s.t.

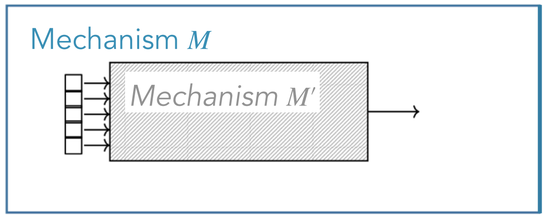
1. Bidder i gets matched to her surrogate in a way that induces **BIC**

2. The interface layer incurs only negligible revenue loss

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Goal: use process of participating in M' as a black-box



Create an interface layer around M', separately for each bidder:

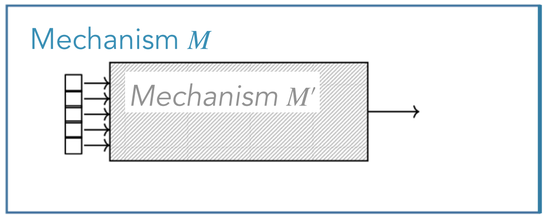
Each bidder i will **purchase a "surrogate" to represent her in M'**, s.t.

1. Bidder i gets matched to her surrogate in a way that induces **BIC**
   via bipartite matching market between "replicas" R_i and "surrogates" S_i
2. The interface layer incurs only negligible revenue loss

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Goal: use process of participating in M' as a black-box



Create an interface layer around M', separately for each bidder:

Each bidder i will **purchase a "surrogate" to represent her in M'**, s.t.

1. Bidder i gets matched to her surrogate in a way that induces **BIC**
   via bipartite matching market between "replicas" R_i and "surrogates" S_i
2. The interface layer incurs only negligible revenue loss
   so that $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Replica-surrogate matching: for each bidder i,

1. Draw set of surrogates S_i each from D_i

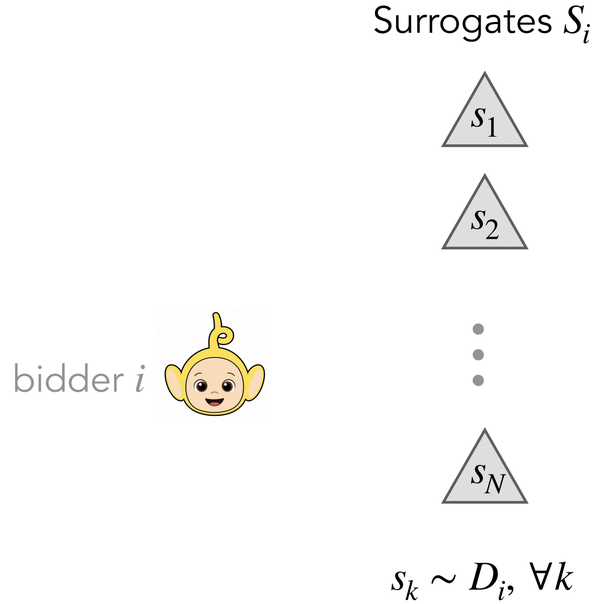
bidder i 

We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Replica-surrogate matching: for each bidder i,

1. Draw set of surrogates S_i each from D_i



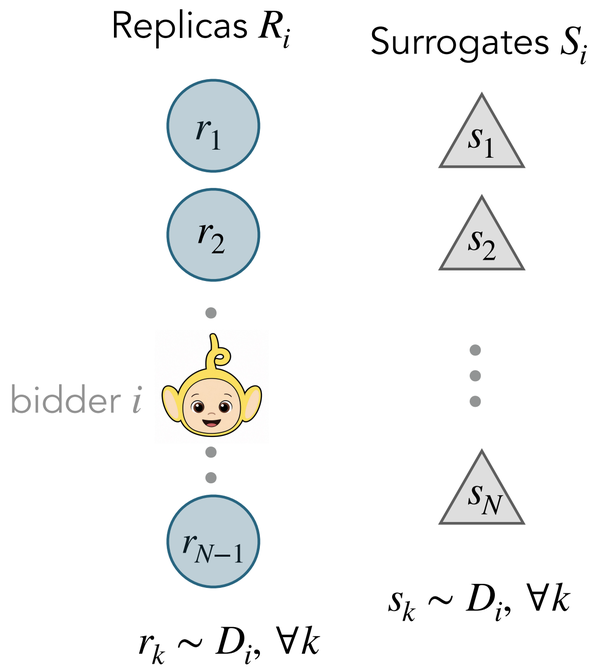
We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Replica-surrogate matching: for each bidder i,

1. Draw set of surrogates S_i each from D_i
2. Draw set of replicas from D_i to "simulate competition" for bidder i

Replicas R_i := simulated replicas ∪ {bidder i}



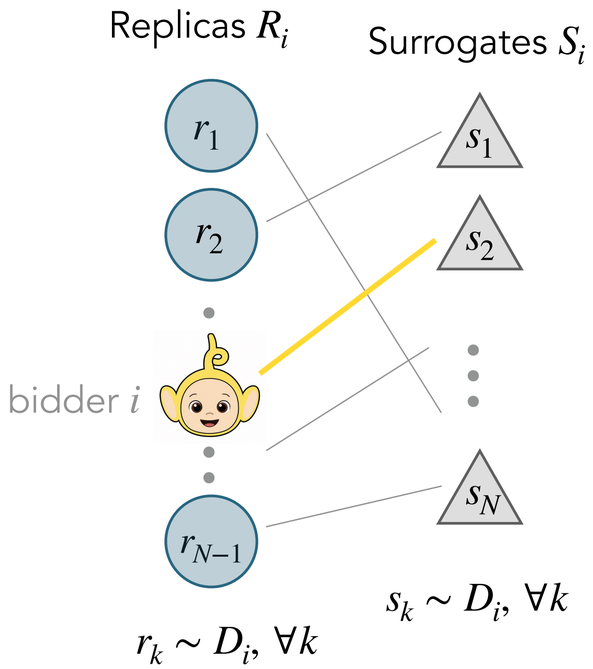
We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Replica-surrogate matching: for each bidder i,

1. Draw set of surrogates S_i each from D_i
2. Draw set of replicas from D_i to "simulate competition" for bidder i

Replicas R_i := simulated replicas ∪ {bidder i}



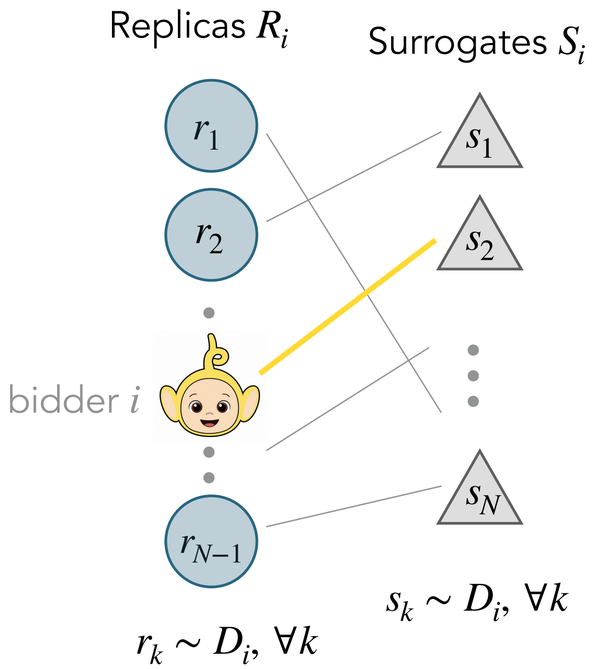
We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]

Replica-surrogate matching: for each bidder i,

1. Draw set of surrogates S_i each from D_i
2. Draw set of replicas from D_i to "simulate competition" for bidder i

Replicas R_i := simulated replicas ∪ {bidder i}

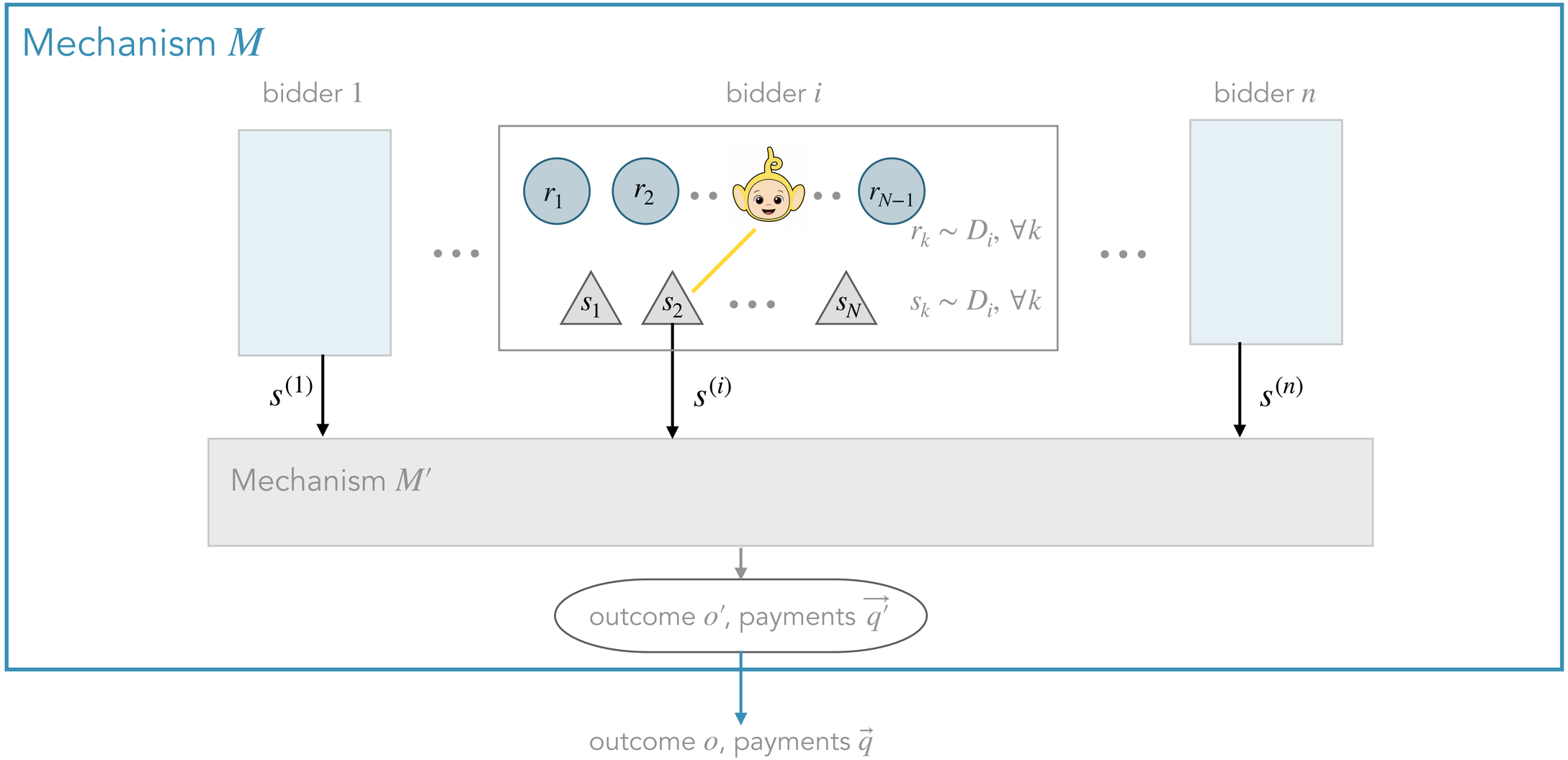


Interpretation: simulated replicas **induce a useful payment rule** faced by bidder i to be matched to a surrogate

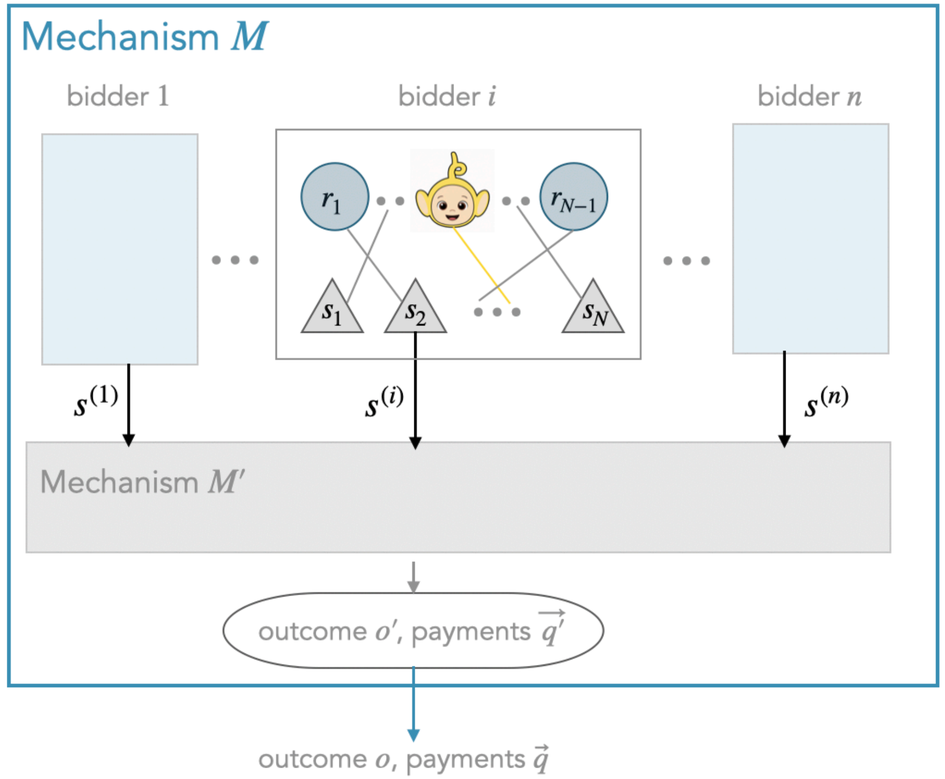
⇒ yields desired BIC guarantee overall

We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]



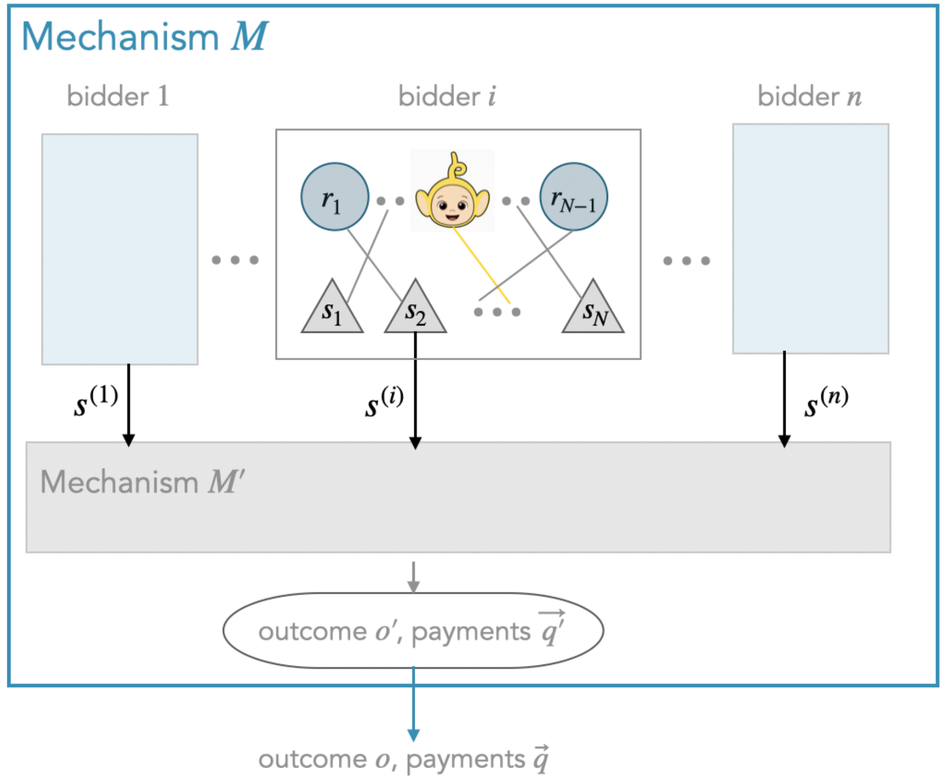
Background: **Replica-Surrogate Matching** [HKM11]



Fact: for **revenue approximation** guarantee, want to match "**close**" replicas & surrogates

We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Background: **Replica-Surrogate Matching** [HKM11]



Fact: for **revenue approximation** guarantee, want to match "**close**" replicas & surrogates

⇒ Need R_i and S_i to form an $O(\varepsilon)$-cover of type space $[0,1]^m$ for each i

⇒ Need $\exp(m)$ replicas and surrogates… ☹️

with only poly(m) samples from D?

We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Idea 1: **Sampling by Products**

⚠️ How to obtain $\exp(m)$-size R_i / S_i with only $\text{poly}(m)$ samples from D_i?

💡 **Independent items**: draw poly samples $S_{i,j}$ from $D_{i,j}$ for each item $j \in [m]$
then take the product set $S_i = \times_{j \in [m]} S_{i,j}$ (so $|S_i| = \text{poly}^m$)

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 1: **Sampling by Products**

⚠️ How to obtain $\exp(m)$-size R_i / S_i with only $\text{poly}(m)$ samples from D_i?

💡 **Independent items**: draw poly samples $S_{i,j}$ from $D_{i,j}$ for each item $j \in [m]$
then take the product set $S_i = \times_{j\in[m]} S_{i,j}$ (so $|S_i| = \text{poly}^m$)

Great! Can $O(\varepsilon)$-cover $\text{supp}(D_i) \subseteq [0,1]^m$ like this; each $s_i \sim D_i$. Same for R_i.

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 1: **Sampling by Products**

⚠️ How to obtain $\exp(m)$-size R_i / S_i with only $\text{poly}(m)$ samples from D_i?

💡 **Independent items**: draw poly samples $S_{i,j}$ from $D_{i,j}$ for each item $j \in [m]$
then take the product set $S_i = \times_{j\in[m]} S_{i,j}$ (so $|S_i| = \text{poly}^m$)

Great! Can $O(\varepsilon)$-cover $\text{supp}(D_i) \subseteq [0,1]^m$ like this; each $s_i \sim D_i$. Same for R_i.

Problem: R_i is no longer **i.i.d** → leads to incentive problems

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 1: **Sampling by Products**

⚠️ How to obtain $\exp(m)$-size R_i / S_i with only $\text{poly}(m)$ samples from D_i?

💡 **Independent items**: draw poly samples $S_{i,j}$ from $D_{i,j}$ for each item $j \in [m]$

then take the product set $S_i = \times_{j\in[m]} S_{i,j}$ (so $|S_i| = \text{poly}^m$)

Great! Can $O(\varepsilon)$-cover $\text{supp}(D_i) \subseteq [0,1]^m$ like this; each $s_i \sim D_i$. Same for R_i.

Problem: R_i is no longer **i.i.d** → leads to incentive problems

Given bidder report $t_i = (t_{i,j})_{j\in[m]}$ and samples $R_{i,j}$ from $D_{i,j}$:

Replicas are set to $R_i := \times_j (R_{i,j} \cup \{t_{i,j}\})$.

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 1: **Sampling by Products**

⚠️ How to obtain $\exp(m)$-size R_i / S_i with only $\text{poly}(m)$ samples from D_i?

💡 **Independent items**: draw poly samples $S_{i,j}$ from $D_{i,j}$ for each item $j \in [m]$
then take the product set $S_i = \times_{j\in[m]} S_{i,j}$ (so $|S_i| = \text{poly}^m$)

Great! Can $O(\varepsilon)$-cover $\text{supp}(D_i) \subseteq [0,1]^m$ like this; each $s_i \sim D_i$. Same for R_i.

Problem: R_i is no longer **i.i.d** → leads to incentive problems

Given bidder report $t_i = (t_{i,j})_{j\in[m]}$ and samples $R_{i,j}$ from $D_{i,j}$:

Replicas are set to $R_i := \times_j (R_{i,j} \cup \{t_{i,j}\})$.

⚠️ So now bidder "influences" a fraction of the final set R_i of replicas

⇒ bidder can manipulate surrogate prices by misreporting

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate “training” phase

We want:
1. M is BIC
2. Rev$(M; D) \approx$ Rev$(M'; D)$,
3. poly$(n, m, 1/\varepsilon)$ samples

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not including bidder*
- Learn prices for matching between R_i^P and S_i

Phase 2: Test

- Draw (the real) replicas R_i via products including bidder's $(t_{i,j})_j$
- Use prices from Phase 1 to match R_i & S_i as the real output

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not including bidder*
- Learn prices for matching between R_i^P and S_i

⇒ Bidder i has no influence over this!

Phase 2: Test

- Draw (the real) replicas R_i via products including bidder's $(t_{i,j})_j$
- Use prices from Phase 1 to match R_i & S_i as the real output

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not including bidder*
- Learn prices for matching between R_i^P and S_i

⇒ Bidder i has no influence over this!

Phase 2: Test

- Draw (the real) replicas R_i via products including bidder's $(t_{i,j})_j$
- Use prices from Phase 1 to match R_i & S_i as the real output

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not including bidder*
- Learn prices for matching between R_i^P and S_i

Phase 2: Test

- Draw (the real) replicas R_i via products including bidder's $(t_{i,j})_j$
- Use prices from Phase 1 to match R_i & S_i as the real output

Intuition: prices from Phase 1 will be pretty good for Phase 2 with high probability.

Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not including bidder*
- Learn prices for matching between R_i^P and S_i

Phase 2: Test

- Draw (the real) replicas R_i via products including bidder's $(t_{i,j})_j$
- Use prices from Phase 1 to match R_i & S_i as the real output

Intuition: prices from Phase 1 will be *pretty good* for Phase 2 *with high probability*.



Idea 2: **Separate Phase for Learning Prices**

⚠️ How to fix incentive issue (bidder being able to affect prices)?

💡 **Decouple learning of prices** via a separate "training" phase

Phase 1: Training

- Draw "pricing-replicas" R_i^P via products, *not in*
- Learn prices for between R_i^P and S_i

Phase 2: Test

- Draw (the real) replicas R_i via bidder's $(t_{i,j})_j$
- ase 1 to match R_i & S_i as the real output

Not good enough:
back to fundamental challenge of
zero error with only sample access

Intuition: prices from Phase 1 will be *pretty good*
for Phase 2 *with high probability*.



Idea 3: **Correlated Discards under Failures**

⚠️ How to fix **low-probability failures** (e.g. learning failures from Idea 2)?

💡 *Convert* low-probability incentive error into *small revenue loss*:

If detect sampling failure for bidder i's R-S matching,
then **discard every other bidder ℓ's allocation** ($\ell \neq i$) & **charge them 0**

We want:
1. M is BIC
2. $\text{Rev}(M; D) \approx \text{Rev}(M'; D)$,
3. $\text{poly}(n, m, 1/\varepsilon)$ samples

Idea 3: **Correlated Discards under Failures**

⚠️ How to fix **low-probability failures** (e.g. learning failures from Idea 2)?

💡 *Convert* low-probability incentive error into *small revenue loss*:

If detect sampling failure for bidder i's R-S matching,
then **discard every other bidder ℓ's allocation** ($\ell \neq i$) & **charge them 0**

Key insight: this preserves BIC!

1. Bidder i sampling failure only possibly messes up incentives for bidders $\ell \neq i$
2. But if ℓ now gets nothing in this case, no new incentive to deviate, because $\ell \neq i$ *can't control when this happens*

Idea 3: **Correlated Discards under Failures**

⚠️ How to fix **low-probability failures** (e.g. learning failures from Idea 2)?

💡 *Convert* low-probability incentive error into *small revenue loss*:

If detect sampling failure for bidder i's R-S matching,
then **discard every other bidder ℓ's allocation** ($\ell \neq i$) & **charge them 0**

Key insight: this preserves BIC!

1. Bidder i sampling failure only possibly messes up incentives for bidders $\ell \neq i$
2. But if ℓ now gets nothing in this case, no new incentive to deviate, because $\ell \neq i$ *can't control when this happens*

+ loses only negligible revenue, since failure probabilities are tiny

Recap & Future Work

- Sample complexity of ϵ-BIC-to-BIC reductions in Bayesian mech design
- Main result: for independent items, **poly**$(n, m, 1/e)$ samples suffice!
- Main algorithmic ideas:

  1. Exponential cover via ***product set*** of polynomially-many draws
  2. "Training phase" to learn prices: ***pricing-replicas*** before real R-S matching
  3. ***Correlated discards*** to eliminate incentive error (convert into revenue loss)

Recap & Future Work

- Sample complexity of ϵ-BIC-to-BIC reductions in Bayesian mech design
- Main result: for independent items, **poly**$(n, m, 1/e)$ samples suffice!
- Main algorithmic ideas:
  1. Exponential cover via *product set* of polynomially-many draws
  2. "Training phase" to learn prices: *pricing-replicas* before real R-S matching
  3. *Correlated discards* to eliminate incentive error (convert into revenue loss)

Future work

- **Computational complexity:** any efficient solutions possible?
- More general classes of **valuations**? (e.g. subadditive, submodular, …)

See the paper for…

- *Meta-framework:* a unified framework that captures both previous work and our algorithms
- Various extensions: Lipschitz valuations over independent items; unbounded payments; etc
- All proof details!

Paper:



Thank you!

a-m@mit.edu

Sketch: **Full Mechanism**

For each bidder $i \in [n]$:

1. Draw surrogates S_i (via products)
2. Draw pricing-replicas R_i^P (via products) & compute prices using (R_i^P, S_i)
3. Elicit bidder i's type $(t_{i,j})_j$ and draw replicas R_i (via products, using $(t_{i,j})_j$)
4. If price learning error: set $\textbf{Flag}_i$ to `True`
5. Match bidder i to a surrogate =: $s^{(i)}$ using (R_i, S_i) and record price q_i

Sketch: **Full Mechanism**

For each bidder $i \in [n]$:

1. Draw surrogates S_i (via products)
2. Draw pricing-replicas R_i^P (via products) & compute prices using (R_i^P, S_i)
3. Elicit bidder i's type $(t_{i,j})_j$ and draw replicas R_i (via products, using $(t_{i,j})_j$)
4. If price learning error: set **Flag**$_i$ to `True`
5. Match bidder i to a surrogate =: $s^{(i)}$ using (R_i, S_i) and record price q_i

Then:

6. Run $M'(s^{(1)}, \ldots, s^{(n)}) \to$ outcome o and payments $p'_1, \ldots, p'_n$
7. For each bidder i:
   If $\exists\ k \neq i$ such that **Flag**$_k$ = `False`: bidder i gets $\varnothing$, pays 0
   Else: bidder i gets o_i and pays $p'_i + q_i$

Bayesian Mechanism Design

- n bidders, m items
- Bidder i has type $t_i \in$ type space $T_i \subseteq [0,1]^m \quad \forall i \in [n]$
- Prior D_i s.t. $t_i \sim D_i$ independently. **Independent items:** each $D_i = \times_{j=1}^{m} D_{i,j}$
- Outcome space O, valuations* $v_i : T_i \times O \to [0,1]$
- Allocation rule $A : T \to \Delta(O)$, payment rule $p : T \to \Delta(\mathbb{R}^n)$
  $\Rightarrow$ **mechanism** $M = (A, p)$

  $T := \times_i T_i$
- Utilities $u_i(t_i, M(t)) := \mathbb{E}_{(o,q)\sim(A(t),p(t))}[v_i(t_i, o) - q_i]$

*Mild assumption: *Lipschitz* valuations (same as [GW18])

Bayesian Mechanism Design

Objective: maximize **expected revenue** $\mathrm{Rev}(M; D) = \mathbb{E}_{t \sim D}[\Sigma_{i \in [n]} p_i(t)]$ $\quad D := \times_i D_i$

M should be incentive compatible:

Bayesian Incentive Compatibility (BIC): for each bidder i, $\quad D_{-i} := \times_{k \neq i} D_k$

$$\forall t_a, t_b \in T_i,\ \mathbb{E}_{t_{-i} \sim D_{-i}}[u_i(t_a, M(t_a; t_{-i})] \geq \mathbb{E}_{t_{-i} \sim D_{-i}}[u_i(t_a, M(t_b; t_{-i})]$$

for any bidder type t_a, truthful report is no worse than any untruthful report t_b in expectation

for each i, truthfully reporting type t_i is no worse than any untruthful report t_i' in expectation (over $t_{-i} \sim D_{-i}$)

Bayesian Mechanism Design

Objective: maximize **expected revenue** $\mathrm{Rev}(M; D) = \mathbb{E}_{t \sim D}[\Sigma_{i \in [n]} p_i(t)]$ $D := \times_i D_i$

M should be incentive compatible:

ε-Bayesian Incentive Compatibility (ε-BIC): for each bidder i, $D_{-i} := \times_{k \neq i} D_k$

$$\forall t_a, t_b \in T_i,\ \mathbb{E}_{t_{-i} \sim D_{-i}}[u_i(t_a, M(t_a; t_{-i})] \geq \mathbb{E}_{t_{-i} \sim D_{-i}}[u_i(t_a, M(t_b; t_{-i})] - \varepsilon$$

for any bidder type t_a, truthful report is no worse than any untruthful report t_b in expectation