



# Algorithmic Mechanism Design with Investments

Mohammad Akbarpour, Scott Kominers,  
Kevin Li, Shengwu Li, Paul Milgrom  
October 26, 2023

# Initial Research Question

---

Algorithmic mechanism design identifies

- algorithms that produce nearly efficient allocations and
- pricing that supplements the algorithm to create a truthful mechanism.

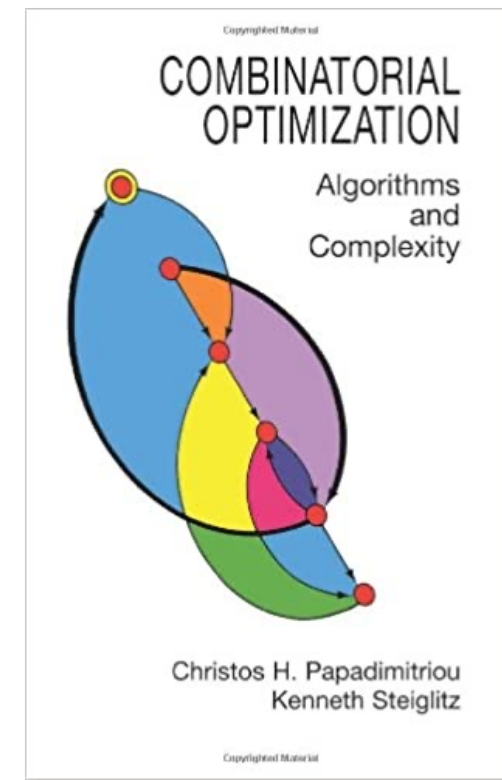
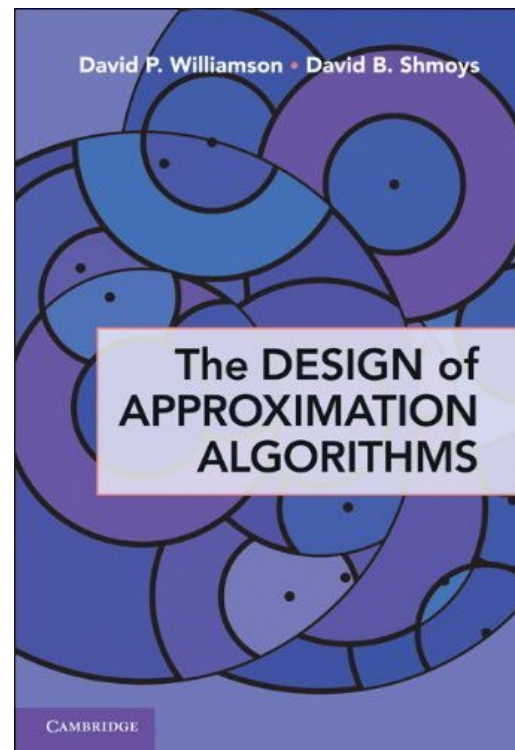
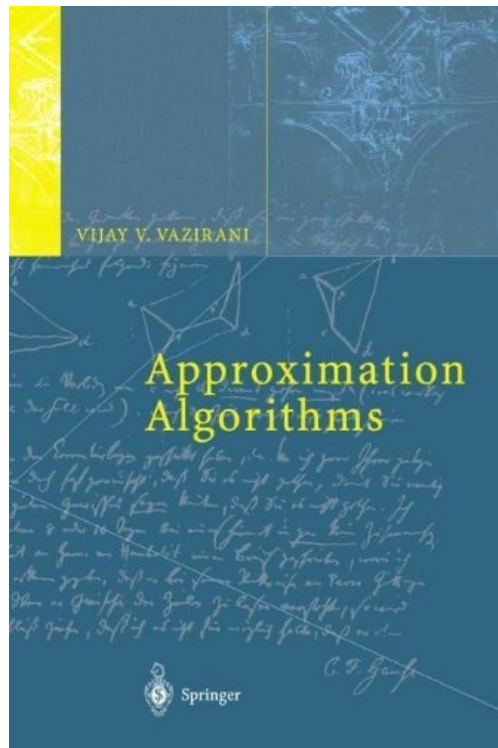
Do near-efficient truthful mechanisms incentivize near-efficient investments?



# When exact optimization is infeasible, approximate!

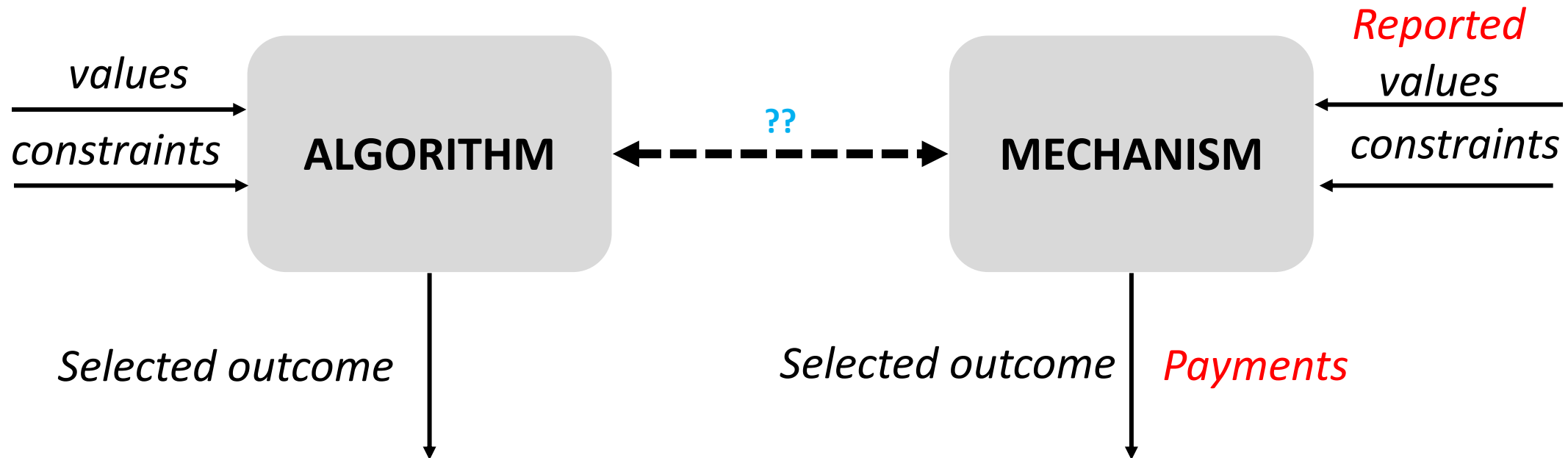
A vast OR/MS literature studies fast approximation algorithms for hard problems.

Theorems emphasize worst-case guarantees for (i) quality of approximation and (ii) computation time.



# Algorithms → Mechanisms

If values are *private information*, computations must rely on reported values.  
The analysis of incentives for *truthful reporting* is part of *mechanism design* theory.



When do there exist payment rules to incentivize truthful reporting?  
Which payment rules do that?



# Packing Problems

A special case to ease exposition

# Notation for Packing Problems

$v_n \in \mathbb{R}_+$  is  $n$ 's the value of being packed (value=0 if not packed).

$\Omega \subseteq \{0,1\}^N$  is the set of feasible packing outcomes.

$x: \mathbb{R}_+^N \rightarrow \Omega$  is the algorithm's packing function.

Realized welfare is:

$$W_x(v) \stackrel{\text{def}}{=} \sum_{n \in N} v_n x_n(v)$$

Constrained optimizer

$$x^*(v|S) \in \arg \max_{z \in S \cap \Omega} \sum_{n \in N} v_n z_n$$

Unconstrained optimizer

$$x^*(v) \stackrel{\text{def}}{=} x^*(v|\Omega)$$

Optimal welfare

$$W^*(v) \stackrel{\text{def}}{=} \sum_{n \in N} v_n x_n^*(v)$$

Payment to  $n$

$$p_n(v_n, v_{-n})$$

# Externalities and Investment Incentives

When player  $n$ 's value changes from  $v_n$  to  $\tilde{v}_n$ , the *externality* is the wedge between the change in  $n$ 's payment and the change in others' welfare.

$$\mathcal{E}_{x,p}(\tilde{v}_n|v) = (p_n(\tilde{v}_n, v_{-n}) - p_n(v)) + \sum_{m \neq n} v_m (x_m(\tilde{v}_n, v_{-n}) - x_m(v))$$

If the investment cost is low enough, player  $n$  may invest to increase its *net* profit by some  $\Delta$ . Then, *net* welfare increases by  $\Delta + \mathcal{E}_{x,p}(\tilde{v}_n|v)$ , which can be negative.

# Main Findings

---

1. All non-optimizing algorithms entail **non-zero externalities**, which can lead to welfare-reducing investments.  
*OK, sure. But won't good approximations have small externalities?*
2. No. An FPTAS can consist entirely of algorithms that guarantee only a **0-fraction of the possible net welfare** when a single agent can invest.  
*Ouch! That's bad. Do all approximation-based algorithms have bad investment guarantees?*
3. No. Any algorithm that “excludes confirming negative externalities” – **XCONE** algorithms – has the **same investment guarantee as its allocative guarantee**.  
*Nice! But are there any useful XCONE algorithms?*
4. Yes. We introduce a new **XCONE FPTAS for the knapsack problem**.  
*Cool! Does the guarantee change if the investor must decide subject to uncertainty?*
5. No, uncertainty does not affect the guarantee. We show that the **worst-case problem is always deterministic**.  
*OK, but surely there is some drawback.*
6. Yes, there is. The XCONE theorem depends on its **unrestricted value domain** assumption, which does not extend to randomized algorithms.



# Truthful Packing Mechanisms

## Definitions

1. A packing algorithm is **monotone** if for each item  $n$ , there exists a **threshold**  $\tau_n(v_{-n})$  such that  $n$  is packed if  $v_n > \tau_n(v_{-n})$  and not packed if  $v_n < \tau_n(v_{-n})$ .
2. A direct mechanism  $(x, p)$  is **truthful** if for all  $v$ , truthful reporting is a Nash equilibrium of the mechanism.
3. The **threshold payment rule** is

$$p^t(v) = \begin{cases} \tau_n(v_{-n}) & \text{when } n \text{ wins} \\ 0 & \text{when } n \text{ loses} \end{cases}$$

## Theorem

The direct mechanism  $(x, p)$  is truthful if and only if

1. The algorithm  $x$  is monotone and
2. For some functions  $\{f_n\}_{n=1}^N$ , we have  $p_n(v) = p_n^t(v) + f_n(v_{-n})$ .

# Proof

For sufficiency, by the usual “second-price/threshold” logic,  $(x, p^t)$  is truthful.

For necessity, suppose that  $(x, p)$  is truthful and denote the maximum payoff to bidder  $n$  by

$$U_n(v_n, v_{-n}|p) \stackrel{\text{def}}{=} \max_{\hat{v}_n} v_n x_n(\hat{v}_n, v_{-n}) - p_n(\hat{v}_n, v_{-n})$$

Since the mechanism is truthful, the maximizer is  $v_n$ :

$$U_n(v_n, v_{-n}|p) = v_n x_n(v_n, v_{-n}) - p_n(v_n, v_{-n})$$

...and by the *envelope theorem*

$$U_n(v_n, v_{-n}|p) = -g_n(v_{-n}) + \int_0^{v_n} x_n(s, v_{-n}) ds$$

where  $-g_n(v_{-n})$  is a constant of integration.

Equating the two expressions for  $U_n(v_n, v_{-n}|p)$  and solving for prices leads to:

$$p_n(v_n, v_{-n}) = g_n(v_{-n}) + v_n x_n(v_n, v_{-n}) - \int_0^{v_n} x_n(s, v_{-n}) ds$$

A similar formula applies to  $p_n^t$ , so  $p_n - p_n^t = g_n(v_{-n}) - g_n^t(v_{-n}) = f_n(v_{-n})$ . ■

# Example: VCG Pivot Mechanism

## Uniform price threshold auctions

- There are more passengers booked ( $P$ ) than seats ( $S$ ) on the plane.
- Each passenger is asked to report her value.
- The  $S$  passengers with highest values are seated and charged a price equal to the  $(S + 1)^{\text{th}}$  highest value.
- Others are not seated and pay zero

## VCG “pivot mechanism” (a more general case)

- Set of packed items  $S$  (indicator  $z^S$ ) is chosen from feasible set  $\Omega$  to maximize total packed value
- Each packed item/bidder  $n \in S$  is charged its threshold price:

$$p_n = \max_{z \in \Omega} \sum_{j \neq n} z_j v_j - (W^*(v) - v_n)$$

- Others are not packed and pay zero.

1. How Algorithms Determine Investment Incentives

2. Algorithms and Externalities: Worst-Case Analysis

3. An FPTAS that is Robust to Investments

# Algorithm + Truthfulness Determines Investment Incentives

*All truthful mechanisms that use algorithm  $x$  have the same investment incentives.*

## Theorem 1

Let  $(x, p)$  and  $(x, \hat{p})$  be two truthful mechanisms with the same algorithm  $x$ . For any  $v_{-n}$ , bidder  $n$ 's net return for investing to change from  $v_n$  to  $v'_n$  is the same for both mechanisms.

**Proof.** Let  $c$  be the cost of investment. Since  $\hat{p}_n(\cdot, v_{-n}) \equiv p_n(\cdot, v_{-n}) + f_n(v_{-n})$ , when  $n$ 's investment changes its value vector from  $v_n$  to  $v'_n$ , its net return using  $p_n$  is

$$\begin{aligned} & (v'_n \cdot x_n(v'_n, v_{-n}) - p_n(v'_n, v_{-n}) - c) - (v_n \cdot x_n(v_n, v_{-n}) - p_n(v_n, v_{-n})) \\ &= (v'_n \cdot x_n(v'_n, v_{-n}) - \hat{p}_n(v'_n, v_{-n}) - c) - (v_n \cdot x_n(v_n, v_{-n}) - \hat{p}_n(v_n, v_{-n})) \blacksquare \end{aligned}$$

The numbered theorems are new in this paper.

# VCG Mechanisms $\Rightarrow$ Efficient Investments

## Theorem

In the VCG pivot mechanism, bidder  $n$ 's profit is  $W^*(v) - W^*(0, v_{-n})$ .

## Theorem (Rogerson)

In any VCG mechanism, bidder  $n$ 's profit increase from an investment at cost  $c$  that changes its value from  $v_n$  to  $\hat{v}_n$  is  $W^*(\hat{v}_n, v_{-n}) - W^*(v) - c$ . It is individually profitable for  $n$  to invest *if and only if* the investment increases net welfare.

# Efficient Investments $\implies$ Constrained Optimization

## Theorem 2

Suppose  $x$  has the property that an investment  $(c, v_n \rightarrow \hat{v}_n)$  is individually profitable if and only if it increases net welfare:

$$W_x(\hat{v}_n, v_{-n}) - c > W_x(v_n, v_{-n}).$$

Then there there exists some  $S \subseteq \Omega$  such that

$$W_x(v) = \max_{z \in S} \sum_{n \in N} v_n(z_n).$$

This is almost, but not quite the same as saying  $x(v) \in \arg \max_{z \in S} \sum_{n \in N} v_n(z_n)$ . The subtlety is that we do not guarantee for all  $v$  that  $x(v) \in S$ .

1. How Algorithms Determine Investment Incentives

2. Algorithms and Externalities: Worst-Case Analysis

3. An FPTAS that is Robust to Investments



# An Extreme but Simple Example

Here is a *satisficing algorithm* for packing problems:

1. If the most valuable item is at least 99% of the total value, pack it alone.
2. Otherwise, optimize.

This algorithm has a **99% allocative guarantee**.

**Investment Example:** Three items. Bidder 1 value 0 while bidders 2 and 3 have value 1 each. It is feasible to pack all items. 1's threshold price is zero.

- By investing at cost 200, bidder 1 can raise its value to  $200 + \varepsilon$ , earning  $\varepsilon > 0$ , so investing is strictly profitable.
- Result of the investment: 1 is packed alone, reducing net value from 2 to  $\varepsilon$ .  
The **investment guarantee** for algorithm  $x$  is no more than  $\inf_{\varepsilon > 0} \frac{\varepsilon}{2 + \varepsilon} = 0$ .

# Is That Example Too Special?

---

No. The example exploits the fact that approximation algorithms for the knapsack problem can have good allocative guarantees even when they are “careless” about packing items with relatively low values.

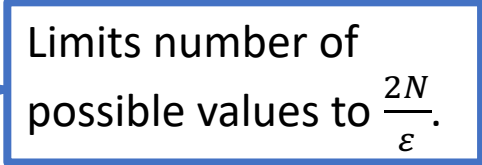
What follows are “real” algorithms with similarly bad investment performance.



# The BKV FPTAS for Knapsack Problems

Briest, Krysta and Vöcking (2005) introduced a **monotone** FPTAS, as follows:

Given an integer value profile  $v \in \mathbb{N}_+^N$  and a target approximation ratio  $1 - \varepsilon$ , construct approximate a series of knapsack problems indexed by  $\ell = 0, 1, \dots$  by replacing the values as follows.

1. **Step size:** Define  $\gamma_\ell \stackrel{\text{def}}{=} \frac{\varepsilon 2^\ell}{N}$ .
  2. **Truncate:**  $v'_n \stackrel{\text{def}}{=} \min(v_n, 2^{\ell+1})$ .
  3. **Round values down to step:**  $v_n^{\varepsilon\ell} \stackrel{\text{def}}{=} \left\lfloor \frac{v'_n}{\gamma_\ell} \right\rfloor \gamma_\ell$
  4. **Optimize:** For all  $\ell \in \mathbb{N}$  such that some  $v_n^{\varepsilon\ell} \neq 0$ , compute  $x^*(v^{\varepsilon\ell})$ . There is a polynomial time dynamic programming algorithm for this problem.
  5. **Choose:** The BKV allocation is  $x_n^*(v^{\varepsilon\ell^*})$  where  $\ell^* \in \arg \max \sum_n v_n^{\varepsilon\ell} x_n^*(v^{\varepsilon\ell})$ .
- 
- Limits number of possible values to  $\frac{2N}{\varepsilon}$ .

## Theorem (BKV FPTAS)

The BKV algorithm is monotone, guarantees  $1 - \varepsilon$  of the maximum, and runs in time that is polynomial in input size and  $\frac{1}{\varepsilon}$ .

The BKV allocation is  $x_n^*(v^{\varepsilon \ell^*})$  where  $\ell^* \in \arg \max \sum_n v_n^{\varepsilon \ell} x_n^*(v^{\varepsilon \ell})$ .

## How BKV Can Go Wrong

- When  $l$  is *low*, high item-value gets truncated to  $2^l$ , which can make low  $\ell^*$  suboptimal.
- When  $l^*$  is *high*, low items-values get rounded to zero and those items go unpacked.
- In our example, a large investment in BKV contribute little to *net* surplus but leads small items to go unpacked, crashing performance just as in the satisficing example.



# BKV: Careless Packing of Low-Value Items

Briest, Krysta and Vöcking (2005) introduced a **monotone** FPTAS, as follows:

Given an integer value profile  $v \in \mathbb{N}_+^N$  and a target approximation ratio  $1 - \varepsilon$ , construct approximate a series of knapsack problems indexed by  $\ell = 0, 1, \dots$  by replacing the values as follows.

1. **Step size:** Define  $\gamma_\ell \stackrel{\text{def}}{=} \frac{\varepsilon 2^\ell}{N}$ .
2. **Truncate:**  $v'_n \stackrel{\text{def}}{=} \min(v_n, 2^{\ell+1})$ .
3. **Round values down to step:**  $v_n^{\varepsilon\ell} \stackrel{\text{def}}{=} \left\lfloor \frac{v'_n}{\gamma_\ell} \right\rfloor \gamma_\ell$
4. **Optimize:** For all  $\ell \in \mathbb{N}$  such that some  $v_n^{\varepsilon\ell} \neq 0$ , compute  $x^*(v^{\varepsilon\ell})$ . There is a polynomial time dynamic programming algorithm for this problem.
5. **Choose:** The BKV allocation is  $x_n^*(v^{\varepsilon\ell^*})$  where  $\ell^* \in \arg \max \sum_n v_n^{\varepsilon\ell} x_n^*(v^{\varepsilon\ell})$ .

*To take full credit for packing highly valued items, low item-values must be rounded to zero, leaving those items unpacked.*

# The BKV FPTAS Destroys Investment Incentives

## Theorem 3

For all  $\delta > 0$ , there exists  $\varepsilon < \delta$  such that the BKV rule with parameter  $\varepsilon$  has an investment guarantee of 0.

**Proof Sketch:** Bidder 1 is the investor with initial value near 0; bidder 2 has value 2; and the bidders 3, ...,  $N$  have values of 1. It is feasible to pack  $\{1,2\}$  or  $\{1,3,4 \dots, N\}$ , but not to pack all bidders. 1 has threshold price 0.

Investment leads to high value  $v_1$  but is barely profitable. It leads to an optimal step size  $\gamma_{\ell^*} = \frac{\varepsilon 2^{\ell^*}}{N} \in (1,2)$ . In problem  $\ell^*$ , the values of bidders 3, ...,  $N$  are rounded down to zero.

Hence,  $x^*(v^{\varepsilon \ell^*}) = \{1,2\}$  is BKV's selected packing.

The net value with investment is 2, but the optimum net value is  $N - 2$ , and  $\lim_{N \rightarrow \infty} \frac{2}{N-2} = 0$ . ■

# Intuition: How Do Investments Worsen Performance?

Consider any packing algorithm  $x$ , its threshold auction, and an investor: bidder  $n$ .

We study three *barely profitable* investment opportunities for approximation algorithms:

1. Investing at cost  $c$  raises  $n$ 's value from below to barely above the threshold.
  - To be profitable for the bidder,  $c = 0$ . The performance ratio is then the same as for the allocation problem created by the investment.
  - In that case, investment performance is **never worse than the worst allocation performance**.
2. Investing at cost  $c$  raises  $n$ 's value **from above the threshold** and is just profitable.
  - Call this a **confirming** investment. By monotonicity,  $n$  remains packed.
  - If  $x$  **reduces** the total value of the **other packed bidders**, that is a **confirming negative externality (CONE)**. Then, net performance is then **strictly worse** than without investment.
  - See the previous example.
3. Investing at cost  $c$  raises  $n$ 's value from strictly below to strictly above the threshold and is barely profitable.
  - This adds the effects of steps (1) and (2) and can be harmful only with a CONE.

Worst cases involve barely profitable investments, because profit raises investment performance.

# The XCONE Sufficient Condition

## Theorem 4

If an algorithm *excludes confirming negative externalities* (“is XCONE”), then its worst-case investment performance is the *same* as its worst-case allocation performance.



# What About Uncertainty?

Suppose that, at the time of investment, the bidder is uncertain about... *everything*:

- Other bidders' values  $v_{-n}$
- The packing constraints
- The  $v_n$  values and constraints that will result from investing and/or not investing
- All of these depend on an unknown state  $s$  with pdf  $p$  (finite support)

## Theorem 5 (“Uncertainty Changes Nothing”)

1. The worst case for the investment problem is deterministic.
2. Even with uncertainty, XCONE is a sufficient condition for the guarantees to coincide.

1. Algorithms and Investment Incentives: Preliminary Analysis

2. Algorithms and Investment Incentives: Worst-Case Analysis

3. An FPTAS that is Robust to Investments

# Finding the CONE

1. In our example, when bidder 1 makes a large investment, the BKV algorithm increases  $\ell^*$ .
2. When  $\ell^*$  increases by enough, the values of bidders  $\{3, \dots, N\}$  are rounded to 0. They are then not packed: there is a confirming negative externality (“CONE”).
3. That externality is so large that it reduces the net value to a near-zero fraction of the optimal investment value.

# Building XCONE Algorithms

## Definition

An algorithm  $x$  is **non-bossy** if for all value profiles  $v, v'$  and all  $n$ ,

$$x_n(v) = x_n(v'_n, v_{-n}) \implies x(v) = x(v'_n, v_{-n}).$$

## Theorem 6

1. Optimization algorithms (as used in BKV) are always non-bossy.
2. For packing problems, every non-bossy algorithm is XCONE.
3. An algorithm that outputs the highest-valued allocation from among other XCONE algorithms is XCONE.

# Adjusting BKV to be XCONE

We make two changes to BKV to create an XCONE algorithm.

1. We define  $x^*$  to always exclude items of value zero and to break ties “systematically.”
  - For large  $\ell$ , with adjusted values of zero, the selected optimum must be  $\emptyset$ .
  - This  $x^*$  is non-bossy.
2. Where BKV uses  $\ell^* \in \arg \max \sum_n v_n^{\varepsilon \ell} x_n^*(v^{\varepsilon \ell})$ , our modification uses  $\ell \in \arg \max \sum_n v_n x_n^*(v^{\varepsilon \ell})$ , based on the actual values  $v$ .
  - Our approximation has values at least as large as BKV, so it is also an FPTAS.
  - Maximization is non-bossy, so **our selection** is a maximum of non-bossy selections, ensuring that our algorithm XCONE.

# XCONE-FPTAS Theorem

## Theorem 7

The modified BKV algorithm is an FPTAS. Its algorithms are XCONE, monotone, and  $1 - \varepsilon$  approximations for the knapsack problem.

...and the same technique can be applied to FPTAS for some other problems as described in the paper.

# Summary (Our Numbered Theorems)

1. Investment incentives are the same for all truthful mechanisms that use the same algorithm.
2. The approximation algorithms that incentivize welfare-increasing investments are the constrained optimizers.
3. The BKV-FPTAS algorithm are *zero*-approximations for the investment problem.
4. XCONE algorithms have the same worst-case performance ratios for both the investment and allocation problems.
5. Investment uncertainty does not change worst-case performance.
6. Every non-bossy packing algorithm is XCONE; max of XCONE algorithms is XCONE.
7. There is an XCONE FPTAS for the knapsack problem (and for several other problems).

# Formulations with Structure Values Including Randomized Approximation Algorithms

A Randomized Approximation Algorithm with **no externalities**:

- With probability  $\frac{1}{2}$ , pack knapsack optimally.
- With probability  $\frac{1}{2}$ , leave knapsack empty
- This randomized algorithm is a  $\frac{1}{2}$  approximation for the allocation problem.
- The value of the random outcome is the mean of the pure values.

Example:

- 1 bidder/item with value 0
- Bidder can invest at cost  $c$  to increase its value to  $1.99c$
- Randomized algorithm: Bidder does not invest. Value remains 0.
- Optimization: Bidder invests. Net value is  $.99c$ .

*Despite **no externalities**, this algorithm, which is a  $\frac{1}{2}$ -approximation for the allocation problem, is a 0-approximation for the investment problem.*



# Beyond Packing Problems

Our general theory makes arguments closely similar to those for packing problems, finding that XONE applies widely to problems satisfying this assumption:

**Assumption.** For each bidder, the possible value vectors is a product of intervals.

This assumption is *not* compatible with *randomized algorithms*, in which some relevant outcomes are randomizations with values equal to the expected value of pure outcomes.

In some formulations with deterministic algorithms, item values are assumed to be additive, so our analysis does not apply to those.

End