Algorithmic Mechanism Design with Investment*

Mohammad Akbarpour[†] Scott Duke Kominers[‡]
Kevin Michael Li[§] Shengwu Li[¶] Paul Milgrom[‖]

First posted: February 25, 2020
This version: November 6, 2022

Abstract

We study the investment incentives created by truthful mechanisms that allocate resources using approximation algorithms. Some approximation algorithms guarantee nearly 100% of the optimal welfare, but have only a zero guarantee when one bidder can invest before participating. An algorithm's worst-case allocative and investment guarantees coincide if and only if that algorithm's *confirming negative externalities* are sufficiently small. We introduce new fast approximation algorithms for the knapsack problem that have no confirming negative externalities, with guarantees close to 100% both with and without investments.

Keywords: Combinatorial optimization, Knapsack problem, Investment, Auctions, Approximation, Algorithms

JEL classification: D44, D47, D82

*An extended abstract of this work appeared in the Proceedings of the 22nd ACM Conference on Economics and Computation. We thank Eric Tang for fantastic research assistance, and thank Moshe Babaioff, Ben Brooks, Peter Cramton, Michal Feldman, Matthew Gentzkow, Paul Goldsmith-Pinkham, Yannai Gonczarowski, Nima Haghpanah, Andy Haupt, John William Hatfield, Nicole Immorlica, Matthew Jackson, Emir Kamenica, Zi Yang Kang, Eric Maskin, Ellen Muir, Rad Niazadeh, Noam Nisan, Amin Saberi, Eric Tang, Mitchell Watt, numerous seminar audiences, the editor (Bart Lipman), and several referees for helpful comments. We thank Broadsheet Cafe for inspiration and coffee. Akbarpour and Kominers gratefully acknowledge the support of the Washington Center for Equitable Growth. Additionally, Kominers gratefully acknowledges the support the National Science Foundation (grant SES-1459912) and both the Ng Fund and the Mathematics in Economics Research Fund of the Harvard Center of Mathematical Sciences and Applications. Milgrom gratefully acknowledges support from the National Science Foundation (grant SES-1947514). All errors remain our own.

[†]Stanford University. Email: mohamwad@stanford.edu
[‡]Harvard University and a16z crypto. Email: kominers@fas.harvard.edu
[§]Stanford University. Email: kli317@stanford.edu
[¶]Harvard University. Email: shengwu_li@fas.harvard.edu
[‖]Stanford University and Auctionomics. Email: milgrom@stanford.edu

1 Introduction

Many real-world allocation problems are too complex for exact optimization. For example, it is computationally difficult—even under full information—to optimally pack indivisible cargo for transport (Dantzig, 1957; Karp, 1972), to coordinate electricity generation and transmission (Lavaei and Low, 2011; Bienstock and Verma, 2019), to assign radio spectrum broadcast rights subject to legally-mandated interference constraints (Leyton-Brown et al., 2017), or to find welfare-maximizing allocations in combinatorial auctions (Sandholm, 2002; Lehmann et al., 2006). Rising to the challenge for these and many other problems, researchers have developed approximation algorithms that are fast but inexact.

Approximation algorithms can be combined with pricing rules to produce truthful mechanisms, provided that the algorithm is "monotone" (Lavi et al., 2003). In this paper, we study the *ex ante* investment incentives created by such mechanisms.

Suppose that one bidder can make a costly investment to change its value before participating in a truthful mechanism. As an initial result, we show that all truthful mechanisms using the same allocation algorithm entail the same investment incentives, so we can regard the investment incentives as properties of the algorithm itself.

If the allocation algorithm exactly maximizes total welfare, then the corresponding truthful mechanism is a Vickrey-Clarke-Groves (VCG) mechanism. VCG mechanisms have efficient investment incentives, in the sense that any single bidder's investment is profitable if and only it improves total welfare (Rogerson, 1992). In this respect, the VCG mechanisms are essentially unique. We find that a truthful mechanism has efficient investment incentives only if there is some set of allocations such that, for generic valuation profiles, its allocation algorithm exactly maximizes welfare over that set. Many practical approximation algorithms do not have this structure and, as a result, lack efficient investment incentives.

One might hope that if an algorithm approximately maximizes total welfare, then it generates approximately efficient investment incentives—but as we show, this is not generally true. Given some instance of a problem, it is standard to evaluate the algorithm's performance by the welfare it yields divided by the maximum welfare. The worst-case ratio over all instances is the algorithm's *performance guarantee*.[1] We refer to the worst-case ratio when all values are exogenous as the *allocative guarantee*, and the worst-case ratio when one bidder can make an *ex ante* investment as the *investment guarantee*.[2] (The investment guarantee

[1] Approximation algorithms are evaluated with multiplicative guarantees, rather than the additive guarantees that are more familiar to some economists. Additive guarantees are typically infeasible for computationally hard problems.

[2] Our results partially extend to the case of multiple bidders who make simultaneous investments, as we discuss in Section 2.4.6.

measures welfare net of investment costs.) We take the worst case over instances and over investment technologies, so the investment guarantee is no more than the allocative guarantee. We characterize the algorithms that are robust to the addition of investment opportunities; *i.e.* those that have investment guarantees equal to their allocative guarantees. We then apply these results to evaluate and improve upon standard approximation algorithms.

1.1 The knapsack example

We introduce and illustrate our results using the knapsack problem (Dantzig, 1957). An instance of the knapsack problem is described by a list of indivisible items, each having a positive size and value, and a capacity constraint. Each item's size is no more than the capacity constraint. The problem is to select ("pack") a set of items to maximize the total value, subject to the sum of the item sizes not exceeding the capacity constraint. Finding an exact optimal solution to the knapsack problem is computationally difficult—it is NP-hard.

Suppose that each item is associated with a different bidder and that all the item sizes are publicly observed, but the value of being packed is the bidder's private information. An algorithm for the knapsack problem is *monotone* if when any packed bidder's value is increased, the algorithm still selects that bidder to be packed. Any monotone algorithm can be paired with a payment rule to create a truthful mechanism. One such payment rule—the *threshold rule*—charges zero to each unpacked bidder and charges each packed bidder its *threshold price*, which is the infimum of the set of values that would result in the bidder being packed.

A well-known fast and monotone algorithm for the knapsack problem is the *greedy algorithm*, which arranges the items in decreasing order of value-to-size, packs items one-by-one in that order, and stops as soon as it encounters some item that is too large to be included along with the items already packed. A variant is the *smart greedy algorithm*, which compares the value of this greedy packing with the highest value of any single item and takes the better of those two feasible solutions. By a textbook argument, the smart greedy algorithm's allocative guarantee is $\frac{1}{2}$.[3] As we see below, when one of the bidders can make a costly investment before participating in the mechanism, the smart greedy algorithm's performance guarantee is unchanged: it is again $\frac{1}{2}$.

Some algorithms have investment guarantees that are strictly worse than their allocative guarantees. Here we offer a simple example that strips away irrelevant complexity to highlight

[3]The argument is as follows. Suppose we relax the integer constraints; the solution to the resulting linear program (LP) is an upper bound for the maximum. The LP solution consists of the greedy solution plus a fraction of the next item on the list. The smart greedy solution is weakly better than the greedy solution and also weakly better than the next item on the list, so it achieves at least $\frac{1}{2}$ of the LP solution.

bidder	A	B	C
value	ε	1	.5
size	.4	.6	.5

Table 1: A knapsack instance, with capacity 1. Assume $0 < \varepsilon < .2$.

a vulnerability also found in some canonical algorithms. Our example consists of a collection of *satisficing algorithms* indexed by a parameter $\delta \in (0, 1)$: If the most valuable item is worth at least δ times the sum of all values, then pack that item and stop. Otherwise, solve the maximization exactly. For each δ, the satisficing algorithm is monotone, so its corresponding threshold auction is truthful. By construction, the satisficing algorithm's allocative guarantee is δ, which can be arbitrarily close to 1, but its investment guarantee is 0.

To see why, consider the knapsack instance specified by Table 1. The satisficing algorithm packs bidders A and B, for a total welfare of $1+\varepsilon$. Since Bidder A is packed for any positive ε, A's threshold price is 0. For a numerical example, fix $\delta = 0.99$ and suppose that bidder A can invest at a cost of 200 to raise its value to $200+2\varepsilon$. This investment is profitable for Bidder A, and it causes the satisficing algorithm to pack just A, for net welfare of $200 + 2\varepsilon - 200 = 2\varepsilon$. But the social optimum is to invest and pack both A and B, for net welfare $1 + 2\varepsilon$. The ratio of net welfare under the satisficing algorithm to the optimal net welfare is therefore no more than $\frac{2\varepsilon}{1+2\varepsilon}$. The parameter ε can be made arbitrarily small, so the satisficing algorithm's investment guarantee—the worst-case ratio over all knapsack instances and all investment technologies—is zero.

By contrast, the smart greedy algorithm's performance guarantee is unaffected by the possibility of investment. In the Table 1 example, the smart greedy algorithm packs just B. Bidder A's threshold price is .4, so it is not profitable to make the investment—and net welfare is 1, compared to the optimum of $1 + 2\varepsilon$. Our main theorem implies that in truthful mechanisms based on the smart greedy algorithm, the net welfare when Bidder A chooses investments to maximize profit is always at least $\frac{1}{2}$ of the maximum net welfare, regardless of the investment technology for Bidder A and the values of the other bidders.

What distinguishes the satisficing and smart greedy algorithms that accounts for their starkly different investment guarantees? Both are approximation algorithms that sometimes fail to maximize welfare. Both pack either the most valuable single item or a selection of items, depending on an inequality condition. The important difference is that, for our satisficing example but not for smart greedy, bidder A's investment both 'confirms' A's original outcome and causes a negative externality, reducing the welfare of bidder B. We show that if an algorithm has no such *confirming negative externalities*, then its investment guarantee is the same as its allocative guarantee.

1.2 Summary of main results

For our general treatment, we assume a finite set of outcomes. Each bidder's value is a vector v_n, with element $v_{n,o}$ capturing bidder n's value for outcome o. The bidder's possible values are a product of intervals, one for each outcome. An allocation assigns one outcome to each bidder; there is an arbitrary set of feasible allocations. An algorithm selects an allocation given the value profile v and the set of feasible allocations.

We allow that investments may be made under uncertainty about all the inputs to the algorithm, including the values resulting from the bidder's investment, the values reported by the other bidders, and the set of feasible allocations. Each input is a random variable described by a function from a finite state space S. We allow any probability distribution on S; our only restriction is that in every state, the realization is an allowable instance of the deterministic problem. We assume that the investor selects an investment from some finite set to maximize the investor's own expected payoff, net of the price it pays in the truthful mechanism and any investment cost. We compare the resulting expected welfare to that of the *optimum*, which results from the *ex ante* efficient investment and the *ex post* efficient allocation. An algorithm is a *β-approximation for investment* if for any instance of the investment problem, the expected welfare of the mechanism is at least β times the optimum welfare.

We prove that introducing uncertainty does not affect the investment guarantee: if x is a β-approximation for investment for all singleton state spaces, then it is also a β-approximation for investment for any finite state space.

To study algorithm performance under certainty, we introduce two new concepts, *algorithmic externalities* and *confirming changes.*

When a bidder invests under a truthful mechanism, that bidder changes its reported value. We define the *algorithmic externality* from that change to be the increase (positive, negative or zero) in the sum of other bidders' values for the resulting allocation, plus the increase in the price the bidder pays. For example, if the bidder's change in value reduces the welfare of other bidders but also increases the bidder's payment to the auctioneer by a larger absolute amount, we count that as a positive externality, because it increases the total welfare of the other bidders plus the auctioneer. As we show, any two truthful mechanisms that use the same underlying allocation algorithm result in identical externalities.

Suppose that at some value profile (v_n, v_{-n}), the algorithm allocates outcome o to bidder n. A change in bidder n's values from v_n to $\tilde{v}_n$ is *confirming* if for any other outcome $\hat{o}$, we have $\tilde{v}_{n,o} - v_{n,o} \geq \tilde{v}_{n,\hat{o}} - v_{n,\hat{o}}$; that is, n's value for the original outcome increases at least as much as its value for any other outcome. If the inequalities were all strict, then monotonicity of the algorithm would imply that such a value change must leave n's outcome unchanged, but others' outcomes may change. With weak inequalities, it is possible that n's outcome changes as well, with a compensating change to n's payments. If given a confirming change to n's report, the algorithm's allocation changes in a way that results in a negative externality, we call that a *confirming negative externality*.

Our first main result establishes a necessary and sufficient condition for the investment and allocative guarantees to coincide, in the form of a bound on the magnitude of confirming negative externalities. Suppose we start at value profile (v_n, v_{-n}) and make a confirming change to $\tilde{v}_n$; our condition requires that any resulting negative externality must not exceed the slack in the allocative guarantee β^* at the original value profile. This bound can be hard to assess, however, because the slack depends on the optimal welfare, which is hard to compute or characterize for many problems of interest. The second result is a corollary of the first, that may be more useful because it is easier to check: if an algorithm excludes confirming negative externalities (XCONE), then its investment and allocative guarantees coincide.

To explore the intuition for our results, we limit attention here to packing problems, in which bidder n faces only two outcomes—winning (being "packed") or losing—when its price for being packed is p.[4] A preliminary observation is that the worst-case investment performance must occur when a bidder who chooses to invest c makes zero additional profit, because reducing c leaves the investment decision unchanged while increasing the ratio of the algorithm's welfare to the optimal welfare.

Consider a bidder n who would not be packed without investment, but by investing at cost $c > 0$ can increase its value to $p + c$, resulting in a net profit of zero. The intuitive argument hinges on decomposing this investment into two parts. Suppose that the bidder first has the option of investing at zero cost to raise its value just to the threshold price p. This is a zero-profit investment, since the result is that the bidder is packed but pays its full value p for that. Because the investment cost is zero, this results in the same welfare as if the bidder's value were fixed at p, so the allocative guarantee implies that welfare is at least β^* of the optimum. In the actual problem, bidder n can invest $c > 0$ to increase its value above the threshold to $p + c$. This option does not change total welfare under the optimal benchmark and also does not change the investor's welfare under the algorithm (net of investment costs). Thus, the investment, which confirms n's packing, affects the

[4] Under the assumptions listed below, the same arguments and intuition can be extended to the case with multiple outcomes.

algorithm's performance if and only if it changes the welfare of the other bidders. If this confirming externality is positive, then the algorithm's performance will strictly exceed β^* of the optimum. If it is negative, then it will reduce the performance and, if it is large enough, it may drag performance down below β^*. In particular, an XCONE algorithm, which excludes confirming negative externalities entirely, cannot lead to the investment guarantee falling below β^*.

Some familiar approximation algorithms are XCONE. These include the greedy algorithm, the smart greedy algorithm, and the clock auction algorithm used for the 2016 Federal Communication Commission's broadcast incentive auction (Leyton-Brown et al. (2017), Milgrom and Segal (2020)).

We also show that XCONE is closely related to non-bossiness. An algorithm is non-bossy if a change in one bidder's report that does not affect that bidder's outcome also does not change any other bidder's outcome. For allocation problems with two outcomes, such as the knapsack problem, every non-bossy algorithm is XCONE. Moreover, we show that any algorithm that, like smart greedy, takes the best output from a family of non-bossy algorithms is XCONE.[5]

We have used simple algorithms for illustration, but our results also illuminate the performance of some canonical algorithms for the knapsack problem. Briest et al. (2005) (henceforth BKV) developed an collection of monotone knapsack algorithms indexed by a parameter $\varepsilon > 0$, with allocative guarantees of $1-\varepsilon$ and maximum run-times bounded by a polynomial function of ε^{-1} and the input length. Such a collection is called a "fully polynomial-time approximation scheme" (FPTAS). However, the BKV FPTAS algorithms are not XCONE. More strongly, we show that the BKV FPTAS is vulnerable in the same way as the satisficing algorithms considered earlier; even if the parameter ε is chosen so that the allocative guarantee is arbitrarily close to 1, the investment guarantee can be 0. We modify the BKV FPTAS to make a new FPTAS that is XCONE, so its allocative guarantee and its investment guarantee are both equal to $1-\varepsilon$.

1.3 Related work

Economists have studied *ex ante* investment in mechanism design at least since the work of Rogerson (1992), who demonstrated that Vickrey mechanisms induce efficient investment. Bergemann and Välimäki (2002) extended this finding in a setting with uncertainty, in which bidders invest in information before participating in an auction. Relatedly, Arozamena and Cantillon (2004), studied pre-market investment in procurement auctions, showing that while

[5] This result extends to problems with more than two outcomes under a tie-breaking condition.

second-price auctions induce efficient investment, first-price auctions do not. Hatfield et al. (2014, 2019) extended these findings to characterize a relationship between the degree to which a mechanism fails to be truthful and/or efficient and the degree to which it fails to induce efficient investment. While like us, Hatfield et al. (2014, 2019) dealt with the connection between (near-)efficiency at the allocation stage and (near-)efficiency at the investment stage, they use additive error bounds, rather than the multiplicative worst-case bounds that are standard for the analysis of computationally hard problems. Gershkov et al. (2020) studied the construction of revenue-maximizing mechanisms with *ex ante* investment. Tomoeda (2019) studied full implementation of exactly-efficient social choice rules with investment.

Our paper is not the first to study investment incentives in an NP-hard allocation setting. Milgrom (2017) introduced a "knapsack problem with investment" in which the items to be packed are owned by individuals, and owners may invest to make their items either more valuable or smaller (and thus easier to fit into the knapsack). In the present paper, we reformulate the investment question in terms of worst-case guarantees and broaden the formulation to study truthful mechanisms for a wide class of resource allocation problems.

Our paper is naturally connected to the large literature on algorithmic mechanism design, started by the seminal paper of Nisan and Ronen (1999). This literature considers computational complexity in mechanism design, and explores properties of approximately optimal mechanisms. Among these works are those of Nisan and Ronen (2007) and Lehmann et al. (2002). Nisan and Ronen (2007) showed that in settings where identifying the optimal allocation is an NP-hard problem, VCG-based mechanisms with nearly-optimal allocation algorithms are generically non-truthful, while Lehmann et al. (2002) introduced a truthful mechanism for the knapsack problem in which the allocation is determined by a greedy algorithm. In addition, Hartline and Lucier (2015) developed a method for converting a (non-optimal) algorithm for optimization into a Bayesian incentive compatible mechanism with weakly higher social welfare or revenue; Dughmi et al. (2017) generalized this result to multidimensional types. For a more comprehensive review of results on approximation in mechanism design, see Hartline (2016).

There is also a literature on greedy algorithms of the type we study here, which sort bidders based on some intuitive criteria and choose them for packing in an irreversible way; see Pardalos et al. (2013) for a review. Lehmann et al. (2002) studied the problem of constructing truthful mechanisms from greedy algorithms; similarly, Bikhchandani et al. (2011) and Milgrom and Segal (2020) proposed clock auction implementations of greedy allocation algorithms.

Finally, our concept of an XCONE algorithm is closely related to the definition of a "bitonic" algorithm, introduced by Mu'Alem and Nisan (2008) to construct truthful mechanisms in combinatorial auctions. Bitonicity is defined for binary outcomes; with the restriction to binary outcomes, every XCONE algorithm is bitonic, but not vice versa.

2 The model and results

2.1 Approximation algorithms

Consider a set of **bidders** N and a set of **outcomes** O, both finite. For instance, in the knapsack problem, the set of outcomes is $\{\text{packed}, \text{unpacked}\}$. The value of bidder n for outcome o is $v_{n,o} \in \mathbb{R}_{\geq 0}$. We write $v_n \equiv (v_{n,o})_{o \in O}$ to denote n's **values**, and we write $v \equiv (v_n)_{n \in N}$ to denote a full **value profile**. An **allocation** $a \in O^N$ assigns one outcome to each bidder; a_n denotes the outcome of bidder n.

An **allocation instance** (N, O, v, A) consists of a set of bidders N, a set of outcomes O, a value profile v, and a set of **feasible allocations** $A \subseteq O^N$. To simplify notation, we often write instances as a pair (v, A), leaving N and O implicit.

The standard approach in computer science is to assess an algorithm's worst-case performance over a domain of instances. Hence, we define an **allocation problem** Ω to be a collection of instances.

Assumption 2.1. We assume that the value profiles in Ω have a product structure. That is, let $V_{n,o}$ be a closed interval of $\mathbb{R}_{\geq 0}$ capturing the possible values that bidder n might have for outcome o. We define $V_n \equiv \prod_{o \in O} V_{n,o}$ and require that $\{v : (N, O, v, A) \in \Omega\} = \prod_{n \in N} V_n$.

In some settings, one outcome $o \in O$ is an **outside option** known to be valued at 0; we capture this with $V_{n,o} = \{0\}$.

An allocation algorithm is a computational procedure that takes as input the value profile and the set of feasible allocations and then outputs an allocation. Each algorithm induces an allocation rule, that is, a function from inputs to outputs. Practical algorithms must run quickly, but most of our results do not depend on running time, so we often use "algorithm" to refer both to the computational procedure and to the function that it induces.

We restrict attention to deterministic allocation algorithms, i.e., that select a single allocation for each instance. This rules out algorithms that randomize over multiple feasible allocations; such algorithms are useful for some computationally hard problems, but our definitions and results do not extend straightforwardly to that case. Formally, an **algorithm** x is a function that selects a feasible allocation for each instance $(v, A) \in \Omega$, that is, $x(v, A) \in A$.[6] We denote the outcome assigned to bidder n under x by $x_n(v, A)$. We abuse notation

[6] In complexity theory, we often are not given the feasible allocations A directly, but instead only a de-

and identify outcomes o with binary vectors of length $|O|$, with one element equal to 1 and all others equal to 0, which allows us to write the **welfare** of algorithm x on instance (v, A) as

$$W_x(v, A) \equiv \sum_n \left[v_n \cdot x_n(v, A)\right].$$

The **optimal welfare** at instance (v, A) is

$$W^*(v, A) \equiv \max_{a \in A} \left\{\sum_n \left[v_n \cdot a_n\right]\right\}.$$

Given some $\beta \in [0, 1]$, algorithm x is a **β-approximation for allocation** if for all $(v, A) \in \Omega$, we have that $W_x(v, A) \geq \beta W^*(v, A)$. We refer to the largest such β as the algorithm's **allocative guarantee**.

2.2 Truthful mechanisms

Suppose that the bidder's values are private information, so that the algorithm cannot directly input each bidder n's value v_n but must instead rely on each bidder's *reported* value $\hat{v}_n$. To elicit these reports, we use a **mechanism** (x, p), which is a pair consisting of an algorithm x and a payment rule p that maps any reported instance $(\hat{v}, A)$ into an allocation $x(\hat{v}, A) \in A$ and a profile of payments $p(\hat{v}, A) \in \mathbb{R}^N$. We adopt the sign convention that payments are made by the participants and to the auctioneer. A mechanism is **truthful** if for all instances $(v, A) \in \Omega$ and all $\hat{v}_n \in V_n$, we have that

$$v_n \cdot x_n(v, A) - p_n(v, A) \geq v_n \cdot x_n(\hat{v}_n, v_{-n}, A) - p_n(\hat{v}_n, v_{-n}, A).$$

When can an algorithm be paired with a pricing rule to produce a truthful mechanism? Algorithm x is **weakly monotone** (**W-Mon**) if for any two instances (v, A) and $(\tilde{v}_n, v_{-n}, A)$, we have

$$[\tilde{v}_n - v_n] \cdot [x_n(\tilde{v}_n, v_{-n}, A) - x_n(v, A)] \geq 0. \tag{1}$$

For **packing problems**, we have $O = \{\text{packed}, \text{unpacked}\}$, a value for being "packed" $v_n^{\text{packed}} \geq 0$, and an outside option with $v_n^{\text{unpacked}} = 0$. In this special case, (1) reduces to

scription that implies which allocations are feasible. For instance, a description could specify the bidders' sizes and the capacity of the knapsack. In principle, algorithms for the knapsack problem could output different allocations for two instances with different item sizes but the same feasible allocations. Our formulation ignores this description-dependence, but we could easily accommodate it by specifying a function from descriptions to feasible allocations, and defining an instance as consisting of a value profile v and a description d; none of our results would materially change with this adjustment.

the requirement that, under x, if n is packed at (v_n, v_{-n}, A) and $\tilde{v}_n^{\text{packed}} \geq v_n^{\text{packed}}$, then n is packed at $(\tilde{v}_n, v_{-n}, A)$.

A necessary condition for the existence of a pricing rule p such that (x, p) is truthful is that the algorithm is weakly monotone—and since V_n is convex, this is also sufficient.[7]

Lemma 2.2 (Lavi et al. (2003); Saks and Yu (2005))**.** *An algorithm x is weakly monotone if and only if there exists a payment rule p such that (x, p) is truthful.*

Pricing rules in truthful mechanisms can be defined in terms of threshold prices, one for each outcome. The least value for bidder n to achieve outcome o is denoted by

$$\hat{\tau}_{n,o}(v_{-n}, A, x) = \inf_{v_n \in V_n} \{v_{n,o} : x_n(v_n, v_{-n}, A) = o\}$$

and the **threshold price** is

$$\tau_{n,o}(v_{-n}, A, x) \equiv \min\{\hat{\tau}_{n,o}(v_{-n}, A, x), \sup V_{n,o}\}.$$

Our results hold trivially if $\tau_{n,o}(v_{-n}, A, x) = \infty$.[8] To focus on the non-trivial case, we assume that $\tau_{n,o}(v_{-n}, A, x) < \infty$; since $V_{n,o}$ is closed, it then follows that $\tau_{n,o}(v_{-n}, A, x) \in V_{n,o}$. We denote the **threshold vector** by $\tau_n(v_{-n}, A, x) \equiv (\tau_{n,o}(v_{-n}, A, x))_{o \in O}$. The set of possible values V_n has a product structure, so we have $\tau_n(v_{-n}, A, x) \in V_n$.

Now, V_n is path-connected, so a standard argument by the envelope theorem yields the following lemma (Milgrom and Segal, 2002).

Lemma 2.3. *If (x, p) is a truthful mechanism, then for each n, there exists a real-valued function $f_n(v_{-n}, A)$ such that*

$$p_n(v, A) = \tau_n(v_{-n}, A, x) \cdot x_n(v, A) + f_n(v_{-n}, A).$$

Lemma 2.3 states that in a truthful mechanism, each bidder pays the threshold price to achieve its assigned outcome plus a strategically irrelevant term that does not depend on the bidder's own report. Truthfulness of (x, p) implies that x assigns each bidder an outcome that maximizes its value minus its threshold price.

[7]Bikhchandani et al. (2006) provided other domain assumptions such that weak monotonicity is sufficient.

[8]Observe that if $\tau_{n,o}(v_{-n}, A, x) = \infty$, then $\hat{\tau}_{n,o}(v_{-n}, A, x) = \infty$ and $\sup V_{n,o} = \infty$, i.e., bidder n is never allocated outcome o and can have arbitrarily large values for o, which in turn implies that x has a zero allocative guarantee.

2.3 Algorithmic externalities

Given mechanism (x, p) and instance (v, A), the **externality** of changing n's value from v_n to $\tilde{v}_n$ is

$$\mathcal{E}_{x,p}(\tilde{v}_n, (v, A)) \equiv \underbrace{p_n(\tilde{v}_n, v_{-n}, A) - p_n(v, A)}_{\text{change in } n\text{'s payment}} + \underbrace{\sum_{m \neq n} v_m \cdot [x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A)]}_{\text{effect on others' welfare}}. \qquad (2)$$

Expression (2) is the portion of n's effect on other participants' welfare that is not fully reflected by n's price.[9] Equivalently, if we treat the auctioneer as the residual claimant to any surplus or deficit of the mechanism, then (2) is the change in the sum of the payoffs of other participants, including the auctioneer.

Lemma 2.3 implies that any two truthful mechanisms that use the same allocation algorithm x have the same externalities. Consequently, we henceforth suppress the dependence of $\mathcal{E}_{x,p}$ on p, writing $\mathcal{E}_x$ and calling this an **algorithmic externality**. VCG mechanisms have zero externalities, so it follows that if x is exactly maximizing, then x has no algorithmic externalities.

For an algorithm to yield efficient investment incentives, it must have zero externalities. Suppose that bidder n changes its value from v_n to $\tilde{v}_n$. If $\mathcal{E}_x(\tilde{v}_n, (v, A)) \neq 0$, then the change in n's payment does not fully capture the effect on others' welfare. Thus, we can find cost $c \in \mathbb{R}$ such that paying c to change n's value from v_n to $\tilde{v}_n$ is privately profitable but not socially optimal, or socially optimal but not privately profitable.

We characterize the zero-externality algorithms. An algorithm x is **maximal-in-range** if for each set of feasible allocations A, there exists a $R \subseteq A$ such that for all v we have

$$x(v, A) \in \operatorname*{argmax}_{a \in R} \left\{ \sum_n [v_n \cdot a_n] \right\}.$$

We say that algorithm x is **welfare-equivalent** to algorithm x' if for every instance (v, A) we have $W_x(v, A) = W_{x'}(v, A)$. Note that if two algorithms are welfare-equivalent, then they yield identical allocations except when two allocations yield exactly the same welfare.

Theorem 2.4. *If algorithm x is weakly monotone, then x has no algorithmic externalities ($\mathcal{E}_x \equiv 0$) if and only if x is welfare-equivalent to a maximal-in-range algorithm.*

Theorem 2.4 implies that it is (essentially) only VCG mechanisms that have no externalities, since any truthful mechanism based on a maximal-in-range algorithm is just a

[9] In some parts of the economics and mechanism design literatures, the word "externality" is used to refer just to the second term, but our definition here is faithful to the traditional Pigouvian concept of externality.

VCG mechanism with restricted range.[10] Many standard approximation algorithms are not maximal-in-range, including the smart greedy algorithm for the knapsack problem. However, some allocation problems have fast maximal-in-range algorithms with meaningful allocative guarantees (Holzman et al., 2004; Dobzinski and Nisan, 2007).

Theorem 2.4 is substantially the same as Theorem 3.2 of Nisan and Ronen (2007). However, their proof does not apply directly to our setting, because it requires the possible values V_n to be unbounded above, whereas we allow V_n to be any product of closed intervals.

2.4 Performance under investment

In mechanisms with algorithmic externalities, selfish investment decisions do not always maximize social welfare. Thus, we study the connection between algorithmic externalities and performance guarantees under investment.

Given a truthful mechanism (x, p), we assess whether the mechanism's performance guarantee applies also to problems in which a single bidder, denoted $\iota \in N$, can decide *ex ante* whether to invest and/or what investment to make. In our formulation, the bidder may be uncertain about what the situation will be when the mechanism is run, including potential uncertainty as to the values that would result from each of its possible investments, the values of the other bidders, and the feasible set that will apply. We compare the expected social welfare from the bidder's selfish investment choice and the given mechanism's allocation to the expected welfare from making the *ex ante* efficient investment and using the *ex post* efficient allocation.

We model the investor's uncertainty using a probability space with a finite number of states S. Each uncertain **investment** opportunity is a pair (ν_ι, c) consisting of a function $\nu_\iota : S \to V_\iota$ and a cost $c \in \mathbb{R}$, while other bidders' values and the feasible set are the function $\nu_{-\iota} : S \to V_{-\iota}$ and correspondence $\mathcal{A} : S \rightrightarrows A$.[11]

Formally, an **investment instance** $(N, O, S, g, \iota, I, \nu_{-\iota}, \mathcal{A})$ consists of:

1. Sets of bidders N and outcomes O.

2. A finite set of states S and a probability distribution $g \in \Delta S$.

3. A distinguished bidder $\iota \in N$, we call the **investor**.

[10] Here is an example of a weakly monotone zero-externality algorithm that is not maximal-in-range: It applies to the problem of selecting two auction winners from among four bidders. If all four bidders have the same value of winning, then the algorithm selects bidders 1 and 2; otherwise, it selects a pair of bidders to maximize welfare from the set Φ consisting of the other five bidder pairs. This algorithm is welfare-equivalent to the algorithm that always selects the welfare-maximizing allocation from Φ.

[11] We do not restrict the correlations among these uncertain elements.

4. A finite set of investments I for ι. To represent the status quo, we require that this set includes at least one pair (ν_ι, c) with $c = 0$.[12]

5. A function from states to the other bidders' values, $\nu_{-\iota} : S \to V_{-\iota}$.

6. A correspondence from states to feasible allocations, $\mathcal{A} : S \rightrightarrows O^N$.

We require that each state $s \in S$ and each investment $(\nu_\iota, c) \in I$ together result in an instance of the original allocation problem, i.e., that $(N, O, \nu(s), \mathcal{A}(s)) \in \Omega$. To simplify our notation, we write each investment instance in the form $\overline{\omega} = (g, I, \nu_{-\iota}, \mathcal{A})$, suppressing N, O, S, and ι. (We use a line above variables to distinguish functions or variables related to an investment problem.)

Suppose that the investor participates in some truthful mechanism (x, p). After an investment is chosen and the state is realized, the investor can do no better than to report the resulting value to the mechanism truthfully. Hence, its best response investment choice at instance $\overline{\omega} = (g, I, \nu_{-\iota}, \mathcal{A})$ is

$$\mathrm{BR}(x, p, \overline{\omega}) \equiv \operatorname*{argmax}_{(\nu_\iota, c) \in I} \left\{ \left(\sum_{s \in S} g(s) \left[\nu_\iota(s) \cdot x_\iota(\nu_\iota(s), \nu_{-\iota}(s), \mathcal{A}(s)) - p_\iota(\nu_\iota(s), \nu_{-\iota}(s), \mathcal{A}(s)) \right] \right) - c \right\}.$$

By Lemma 2.3, the price $p_\iota(v, A)$ paid by the investor consists of a term entirely pinned down by the algorithm x, plus a term that does not depend on its own report. Thus, for any two truthful mechanisms that use the same algorithm, (x, p) and (x, p'), the bidder has the same privately optimal investments—$\mathrm{BR}(x, p, \overline{\omega}) = \mathrm{BR}(x, p', \overline{\omega})$—so we henceforth suppress the payment rule argument p from $\mathrm{BR}(\cdot)$.[13]

The **welfare** of algorithm x at investment instance $\overline{\omega} = (g, I, \nu_{-\iota}, \mathcal{A})$ is

$$\overline{W}_x(\overline{\omega}) \equiv \min_{(\nu_\iota, c) \in \mathrm{BR}(x, \overline{\omega})} \left\{ \left(\sum_{s \in S} g(s) W_x(\nu_\iota(s), \nu_{-\iota}(s), \mathcal{A}(s)) \right) - c \right\}.$$

We benchmark performance relative to the net welfare delivered by *ex ante* efficient investment and *ex post* efficient allocations. That is, the **optimal welfare** at investment instance $\overline{\omega} = (g, I, \nu_{-\iota}, A)$ is

$$\overline{W}^*(\overline{\omega}) \equiv \max_{(\nu_\iota, c) \in I} \left\{ \left(\sum_{s \in S} g(s) W^*(\nu_\iota(s), \nu_{-\iota}(s), \mathcal{A}(s)) \right) - c \right\}.$$

[12] Negative costs $c < 0$ represent disinvestments compared to the status quo.

[13] By assuming that the investor best-responds, we are abstracting from computational limitations that the investor might face when there are many states.

The benchmark $\overline{W}^*(\varpi)$ is equal to the net welfare from selfish investment under a VCG mechanism. Given some $\beta \in [0,1]$, algorithm x is a **β-approximation for investment** if for every investment instance ϖ, we have that $\overline{W}_x(\varpi) \geq \beta\overline{W}^*(\varpi)$. Notably, since we are quantifying over $\iota \in N$ and I, this requires the inequality to hold regardless of which bidder is the investor and which investments are available.[14] We refer to the largest such β as the algorithm's **investment guarantee**.

Adding investment opportunities weakly reduces the algorithm's performance guarantee.

Proposition 2.5. *If x is a β-approximation for investment, then x is a β-approximation for allocation.*

Proof. Any instance of the allocation problem (v, A) is equivalent to the instance of the investment problem $(g, I, \nu_{-\iota}, \mathcal{A})$ with the singleton investment technology $I = \{(\nu_\iota, 0)\}$, $\nu_\iota \equiv v_\iota$, $\nu_{-\iota} \equiv v_{-\iota}$, and $\mathcal{A} \equiv A$; the result then follows. $\square$

The converse of Proposition 2.5 does not hold in general—investment opportunities may strictly reduce the algorithm's performance guarantee. But when is an algorithm's allocative guarantee equal to its investment guarantee? We now determine the answer.

2.4.1 Reduction to the case without uncertainty

First, we simplify the problem by observing that, for our purposes, it is without loss of generality to focus on the case without uncertainty. That is, a **certain investment instance** ϖ is an investment instance with just one state, so $|S| = 1$, and we abuse notation by writing such an instance as $(I, v_{-\iota}, A)$. An algorithm is a **β-approximation for certain investment** if $\overline{W}_x(\varpi) \geq \beta\overline{W}^*(\varpi)$ for any certain investment instance ϖ.

The next theorem states that an algorithm's performance guarantee with investment is the same as its performance guarantee with certain investment.

Theorem 2.6. *For any weakly monotone algorithm x and any $\beta \in [0,1]$, x is a β-approximation for investment if and only if x is a β-approximation for certain investment.*

The intuition for Theorem 2.6 is as follows: Suppose we start from some investment instance with uncertainty. We can construct a related generalized investment instance with the same values in every state but a *state-dependent* cost that makes the realized profit in each state equal to the original *ex ante* expected profit. This leaves the expected profit

[14] Our results extend naturally if some bidders are known in advance to be unable to make investments. In that case, our necessary conditions weaken to pertain only to those bidders who can make investments.

and costs from investing unchanged, so selfish investment in the original instance yields the same expected welfare as selfish investment in the new instance. If the algorithm x is a β-approximation for certain investment, then in the new instance, in every state, the selfish investment achieves at least a fraction β of the welfare from the *ex post* efficient investment and the *ex post* efficient allocation. In turn, this is an upper bound for the expected welfare from the *ex ante* efficient investment and the *ex post* efficient allocation in the new instance, which is the same as in the original instance. It follows that x is a β-approximation for investment.

2.4.2 Performance under certain investment

Having reduced the problem with uncertain investment to the problem with certain investment, we now derive a necessary and sufficient condition for an algorithm x to be a β-approximation for certain investment.

We will show a link between investment guarantees and algorithmic externalities. We can simplify the problem by focusing on the externalities that result from value changes in particular directions. Changing from v_n to $\tilde{v}_n$ **confirms** outcome $\tilde{o}$ if

$$[\tilde{v}_n - v_n] \cdot [\tilde{o} - o] \geq 0 \text{ for all outcomes } o. \tag{3}$$

Intuitively, (3) means that changing n's value from v_n to $\tilde{v}_n$ raises n's value for $\tilde{o}$ at least as much as it raises n's value for any other outcome—equivalently, n's marginal gain from switching from o to $\tilde{o}$ does not fall. The system of inequalities (3) defines a convex cone with vertex at v_n. If x is weakly monotone, then any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$ implies that

$$[\tilde{v}_n - v_n] \cdot [x_n(\tilde{v}_n, v_{-n}, A) - x_n(v, A)] = 0; \tag{4}$$

this follows by combining (1) and (3), with $\tilde{o} = x_n(v, A)$ and $o = x_n(\tilde{v}_n, v_{-n}, A)$. For any truthful (x, p), type v_n cannot profitably imitate $\tilde{v}_n$ and *vice versa*, so

$$v_n \cdot [x_n(\tilde{v}_n, v_{-n}, A) - x_n(v, A)] \leq p_n(\tilde{v}_n, v_{-n}, A) - p_n(v, A) \leq \tilde{v}_n \cdot [x_n(\tilde{v}_n, v_{-n}, A) - x_n(v, A)]. \tag{5}$$

From (4) and (5), it follows that

$$p_n(\tilde{v}_n, v_{-n}, A) - p_n(v, A) = v_n \cdot [x_n(\tilde{v}_n, v_{-n}, A) - x_n(v, A)]\,; \tag{6}$$

that is, the bidder with value v_n is indifferent between reporting v_n and reporting the confirming change $\tilde{v}_n$ when facing (v_{-n}, A).

The externalities from confirming changes reduce to a simple expression. In particular, they are equal to the difference between the welfare yielded by the new allocation at the old values and the welfare yielded by the old allocation at the old values.

Proposition 2.7. *For any weakly monotone x, any instance (v, A), and any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$, we have*

$$\mathcal{E}_x(\tilde{v}_n, (v, A)) = \sum_m \left[v_m \cdot \left[x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A) \right] \right]. \tag{7}$$

Proof. Substituting (6) into (2) yields (7). □

The key condition for our characterization is a lower bound on the externalities resulting from confirming value changes.

Definition 2.8. For some $\beta \in [0, 1]$, algorithm x has β**-bounded confirming externalities** if given any instance (v, A) and any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$, we have

$$\mathcal{E}_x(\tilde{v}_n, (v, A)) \geq \beta W^*(v, A) - W_x(v, A). \tag{8}$$

The inequality (8) requires the algorithmic externality of the confirming change to exceed the lower bound, which is the negative of the slack in the allocative guarantee at instance (v, A). Definition 2.8 is a necessary condition for an algorithm to be a β-approximation for certain investment, as we now prove.

Theorem 2.9. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, if x is a β-approximation for certain investment, then x has β-bounded confirming externalities.*

Proof. We prove the contrapositive. Suppose x does not have β-bounded confirming externalities. Take any instance (v, A) and value change $\tilde{v}_n$ such that (8) does not hold, so

$$\mathcal{E}_x(\tilde{v}_n, (v, A)) < \beta W^*(v, A) - W_x(v, A). \tag{9}$$

Choose investment cost c so that n is indifferent between v_n at cost 0 and $\tilde{v}_n$ at cost c, and denote the resulting investment instance by $\overline{\omega}$. By construction, it is a best response for n to choose $\tilde{v}_n$ at cost c, so we have

$$\overline{W}_x(\overline{\omega}) \leq W_x(\tilde{v}_n, v_{-n}, A) - c. \tag{10}$$

Moreover, the welfare of the best allocation when n chooses v_n at cost 0 is no more than the optimal benchmark at $\overline{\omega}$, so we have

$$W^*(v, A) \leq \overline{W}^*(\overline{\omega}). \tag{11}$$

Combining the inequalities (10), (9), and (11) yields

$$\begin{aligned}
\overline{W}_x(\overline{\omega}) &\leq W_x(\tilde{v}_n, v_{-n}, A) - c \\
&= W_x(v, A) + \underbrace{\tilde{v}_n \cdot x_n(\tilde{v}_n, v_{-n}, A) - v_n \cdot x_n(v, A) - c}_{=p_n(\tilde{v}_n, v_{-n}, A) - p_n(v, A) \text{ by construction of } c} + \sum_{m \neq n} v_m \cdot [x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A)] \\
&= W_x(v, A) + \mathcal{E}_x(\tilde{v}_n, (v, A)) \\
&< \beta W^*(v, A) \\
&\leq \beta \overline{W}^*(\overline{\omega}),
\end{aligned}$$

showing that x is not a β-approximation for certain investment. □

We now state the converse of Theorem 2.9, *i.e.* that Definition 2.8 is a sufficient condition.

Theorem 2.10. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, if x has β-bounded confirming externalities, then x is a β-approximation for certain investment.*

We summarize the preceding results in a corollary:

Corollary 2.11. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, the following statements are equivalent:*

1. *x is a β-approximation for investment.*

2. *x is a β-approximation for certain investment.*

3. *x has β-bounded confirming externalities.*

2.4.3 A tractable sufficient condition

Corollary 2.11 characterizes the allocation algorithms that attain performance guarantee β under investment. However, expression (8) in the definition of β-bounded confirming externalities requires that we assess the *optimal* welfare W^* at some instance. This raises a practical challenge given that our main interest is in problems for which optimal allocations are hard to compute. Thus, even though Corollary 2.11 yields necessary and sufficient conditions for an algorithm to attain some guarantee β, direct application of these results may sometimes be intractable.

We thus introduce a tractable sufficient condition that implies that the algorithm's allocative guarantee and investment guarantee are equal.

Definition 2.12. Algorithm x **excludes confirming negative externalities** (**"is XCONE"**) if given any instance (v, A) and any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$, we have

$$\mathcal{E}_x(\tilde{v}_n, (v, A)) \geq 0.$$

Theorem 2.13. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, if x is XCONE and a β-approximation for allocation, then x is a β-approximation for investment.*

Proof. Take any (v, A) and any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$. We have

$$\mathcal{E}_x(\tilde{v}_n, (v, A)) \geq 0 \geq \beta W^*(v, A) - W_x(v, A);$$

the first inequality follows since x is XCONE and the second inequality follows since x is a β-approximation for allocation. Thus, we see that x has β-bounded confirming externalities. By Corollary 2.11, x is a β-approximation for investment. □

2.4.4 Relating XCONE to non-bossiness

Our XCONE condition is related to the standard mechanism design concept of non-bossiness. Algorithm x is **non-bossy** if $x_n(\tilde{v}_n, v_{-n}, A) = x_n(v, A)$ implies that $x(\tilde{v}_n, v_{-n}, A) = x(v, A)$, that is, if changing n's value does not change n's outcome it must not change others' outcomes, either. Algorithm x is **consistent** if (4) implies that $x_n(\tilde{v}_n, v_{-n}, A) = x_n(v, A)$; this holds, for instance, if whenever bidder n is indifferent between several outcomes at the threshold prices, the algorithm breaks ties according to some fixed order on outcomes.

Proposition 2.14. *In packing problems, every algorithm is consistent.*

Proof. We prove the contrapositive. Suppose $x_n(\tilde{v}_n, v_{-n}, A) \neq x_n(v, A)$; in a packing problem, this implies that $\tilde{v}_n^{\text{packed}} \neq v_n^{\text{packed}}$. Without loss of generality, suppose n is packed under $x_n(\tilde{v}_n, v_{-n}, A)$ and not packed under $x_n(v, A)$. Then the left-hand side of (4) is equal to $\tilde{v}_n^{\text{packed}} - v_n^{\text{packed}}$, which is non-zero. □

Proposition 2.15. *If x is weakly monotone, consistent, and non-bossy, then x is XCONE.*

Proof. Because x is weakly monotone, any change from v_n to $\tilde{v}_n$ that confirms $x_n(v, A)$ implies (4). Next, because x consistent and non-bossy, (4) implies that $x(\tilde{v}_n, v_{-n}, A) = x(v, A)$. Thus, we have

$$0 = \sum_m v_m \cdot [x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A)] = \mathcal{E}_x(\tilde{v}_n, (v, A)),$$

where the final equality follows from Proposition 2.7. □

2.4.5 Combinations of XCONE algorithms

A standard technique for addressing computationally difficult allocation problems is to run several candidate algorithms and select the best of their solutions; this can yield a better allocative guarantee than each individual algorithm. However, the resulting algorithm may be bossy, even if the candidate algorithms are non-bossy. By contrast, if the candidate algorithms are XCONE, then the resulting algorithm is XCONE.[15]

Proposition 2.16. *Let X be a collection of weakly monotone XCONE algorithms. If y is a weakly monotone algorithm that at each instance $(v, A) \in \Omega$ outputs a welfare-maximizing allocation from the collection $\{x(v, A)\}_{x \in X}$, then y is XCONE.*[16]

Proof. Consider any instance (v, A) and any $\tilde{v}_n$ that confirms $x(v, A)$, Let $x \in X$ be such that $y(v, A) = x(v, A)$. Because x is weakly monotone and XCONE, Proposition 2.7 implies that

$$0 \leq \mathcal{E}_x(\tilde{v}_n, (v, A)) = \sum_m v_m \cdot [x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A)]\,; \tag{12}$$

hence, we have

$$\begin{aligned}
\sum_m v_m \cdot y_m(v, A) &= \sum_m v_m \cdot x_m(v, A) \\
&\leq \sum_m v_m \cdot x_m(\tilde{v}_n, v_{-n}, A) \\
&\leq \sum_m v_m \cdot y_m(\tilde{v}_n, v_{-n}, A),
\end{aligned} \tag{13}$$

where the first inequality follows from (12), and the second uses the definition of y. Rear-

[15] To see this, consider an allocation problem with three bidders; bidder n's value for being packed is v_n^{packed} and its value for not being packed is $v_n^{\text{unpacked}} = 0$. Algorithm x packs bidders 1 and 2 if $v_2^{\text{packed}} + 3 > v_3^{\text{packed}}$ and packs bidders 1 and 3 otherwise. Algorithm x' always packs just bidder 3. Let x'' select the best solution from x and x'. When $v_1^{\text{packed}} = 1$, $v_2^{\text{packed}} = 2$, $v_3^{\text{packed}} = 4$, x'' packs bidder 3, but if we raise v_3^{packed} to 8, then x'' packs bidders 1 and 3. Thus, while x and x' are non-bossy, x'' is bossy—yet all three algorithms are XCONE.

[16] Our necessary and sufficient condition, Definition 2.8, also has this property. That is, if we replace the supposition that every algorithm in X is XCONE with the supposition that every algorithm in X has β-bounded confirming externalities, then a parallel proof yields the conclusion that y has β-bounded confirming externalities.

ranging (13) yields

$$\begin{aligned} 0 &\leq \sum_m v_m \cdot [y_m(\tilde{v}_n, v_{-n}, A) - y_m(v, A)] \\ &= \mathcal{E}_y(\tilde{v}_n, (v, A)), \end{aligned}$$

where the equality follows from Proposition 2.7 because y is weakly monotone. □

Proposition 2.16 assumes that y is weakly monotone. Yet weak monotonicity of every algorithm in X does not necessarily imply weak monotonicity of y, even though y is a welfare-maximizing selection from X (see Example 2.18 below). One other advantage of XCONE algorithms is that such a y *does* inherit weak monotonicity from X when there are only two outcomes.

Proposition 2.17. *Suppose that $|O| = 2$, and let X be a collection of weakly monotone XCONE algorithms. If y is an algorithm that at each instance $(v, A) \in \Omega$ outputs a welfare-maximizing allocation from the collection $\{x(v, A)\}_{x \in X}$, then y is weakly monotone.*

Proposition 2.17 does not generalize to $|O| > 2$. Indeed, there exist pairs of candidate algorithms, both weakly monotone and XCONE, such that the resulting y is not weakly monotone—as the following example illustrates.

Example 2.18. Consider an allocation problem with two bidders and three outcomes, and suppose that $V_1 = [0, 4] \times [0, 4] \times \{0\}$ and $V_2 = \{(5, 0, 0)\}$. We suppose that algorithm x always allocates outcome 2 to bidder 1 and outcome 3 to bidder 2, while algorithm $\tilde{x}$ allocates outcome 1 to both bidders if $v_1 \geq 1$ and allocates outcome 3 to both bidders otherwise. Both x and $\tilde{x}$ are weakly monotone and XCONE. Let y be an algorithm that outputs a welfare-maximizing allocation from the set $\{x(v_1, v_2), \tilde{x}(v_1, v_2)\}$. Under algorithm y, bidder 1 gets outcome 2 when $v_1 = (0, 1, 0)$ and outcome 1 when $v_1 = (2, 4, 0)$, so y is not weakly monotone.

2.4.6 Allowing multiple investors

Suppose that each bidder n has a finite set of feasible investments I_n and, as before, an investment consists of a function $\nu_n : S \to V_n$ and a cost $c \in \mathbb{R}$. Suppose that all bidders simultaneously choose investments, knowing that in each state $s \in S$, the resulting allocation and payments will be $x(\nu(s), A(s))$ and $p(\nu(s), A(s))$, for truthful mechanism (x, p). The resulting investment game has a Nash equilibrium, possibly in mixed strategies.

Even for VCG mechanisms, not every Nash equilibrium of the investment game is efficient. Complementarities between the bidders can result in inefficient Nash equilibria, as the following example illustrates.

Example 2.19. Consider a packing problem with three bidders. It is feasible to pack any single bidder, or to pack bidder 2 and bidder 3 simultaneously. There is only one state and so no uncertainty: $|S| = 1$. Bidder 1 has a *status quo* value 10 for being packed, that is, its technology is the singleton $I_1 = \{(10, 0)\}$. Bidders 2 and 3 have the technology $I_2 = I_3 = \{(0, 0), (9, 1)\}$. Total welfare is maximized if both bidders 2 and 3 choose the investment $(9, 1)$, which leads to both being packed. However, if only one of them invests, then it is optimal to pack just Bidder 1. In the VCG auction, there are two pure strategy Nash equilibrium investment profiles. In one Nash equilibrium, no bidder invests and Bidder 1 is packed, for net welfare 10. In the efficient Nash equilibrium, both Bidders 2 and 3 invest and both are packed, for net welfare 16.

Nevertheless, VCG mechanisms satisfy a different efficiency criterion: Conditional on any belief about the strategies of the other bidders, every best response for bidder n maximizes interim social welfare net of bidder n's investment costs.

Our results extend this observation to include approximate efficiency. Any best response of bidder n to its belief about the other bidders' investments yields social welfare (net of n's investment costs) that is at least a fraction β of what would be achieved by the interim efficient investment for bidder n and the *ex post* efficient allocation. The next proposition states this formally.

Proposition 2.20. *Let $h \in \Delta(I_{-n})$, with $h(\nu_{-n})$ denoting the marginal distribution. Let $(\nu_n, c) \in I_n$ be a best response for bidder n to the belief h given algorithm x. If x has β-bounded confirming externalities, then*

$$\left(\sum_{\nu_{-n}} h(\nu_{-n}) \sum_{s \in S} g(s) W_x(\nu_n(s), \nu_{-n}(s), \mathcal{A}(s))\right) - c$$
$$\geq \beta \max_{(\nu'_n, c') \in I_n} \left(\sum_{\nu_{-n}} h(\nu_{-n}) \sum_{s \in S} g(s) W^*(\nu'_n(s), \nu_{-n}(s), \mathcal{A}(s))\right) - c'.$$

Proof. Let us define a new single-investor instance that is payoff-equivalent for n, incorporating n's belief h using an expanded state space $S \times S'$ and functions $\tilde{\nu}_{-n} : S \times S' \to V_{-n}$. For each of bidder n's investments $(\nu_n, c) \in I_n$, we define a corresponding investment $(\tilde{\nu}_n, c)$ with $\tilde{\nu}_n(s, s') \equiv \nu_n(s)$, and similarly define $\tilde{\mathcal{A}}(s, s') \equiv \mathcal{A}(s)$. By Corollary 2.11, if x has β-bounded confirming externalities, then the desired inequality follows. □

3 Application: Knapsack algorithms

The **knapsack problem** is a special case of the allocation problem introduced in Section 2.1, in which there are two outcomes, $\{\text{packed}, \text{unpacked}\}$. Each bidder n has possible values $V_n^{\text{packed}} = [0, \infty)$ and $V_n^{\text{unpacked}} = \{0\}$. Thus for knapsack problems, we abuse notation and use v_n to denote v_n^{packed}, bidder n's value for being packed, since $v_n^{\text{unpacked}} \equiv 0$ uniformly.

Each bidder has **size** $q_n \geq 0$, and the knapsack has **capacity** Q. Without loss of generality, suppose no bidder's size is more than Q.[17] The set of feasible allocations is any subset of bidders $K \subseteq N$ such that $\sum_{n \in K} q_n \leq Q$. As before, let A denote the set of feasible allocations and let a be an element of A.

The knapsack problem is NP-Hard (Karp, 1972); there is no known polynomial-time algorithm that outputs optimal allocations (Cook, 2006; Fortnow, 2009). Dantzig (1957) suggested applying a **greedy algorithm** to the knapsack problem. Formally:

Algorithm (Greedy)**.** Sort bidders by the ratio of their values to their sizes so that

$$\frac{v_1}{q_1} \geq \frac{v_2}{q_2} \cdots \geq \frac{v_{|N|}}{q_{|N|}}. \tag{14}$$

Add bidders to the knapsack one by one in the sorted order, so long as the sum of the sizes does not exceed the knapsack's capacity. When encountering the first bidder that would violate the capacity constraint, stop.

Although the Greedy algorithm performs well on some instances—including ones for which all bidders are small in relation to the capacity of the knapsack—its allocative guarantee is 0, as illustrated by the following example.

Example 3.1. Consider a knapsack with capacity 1 and two bidders. For some arbitrarily small $\varepsilon > 0$, let $v_1 = \varepsilon$, $q_1 = \frac{\varepsilon}{2}$, $v_2 = 1$, and $q_2 = 1$. The Greedy algorithm picks bidder 1 and stops, whereas the optimal algorithm picks bidder 2. Thus, the worst-case performance of Greedy is no better than ε of the optimum.

There is a standard modification of the Greedy algorithm that improves the allocative guarantee for the knapsack problem (Williamson and Shmoys, 2011, p. 77). Let us define the **"smart greedy" algorithm** as follows.

Algorithm (SmartGreedy)**.** Run the Greedy algorithm. Compare the Greedy algorithm's packing to the packing that just packs the most valuable individual bidder, and output whichever has higher welfare.

[17] Bidders with $q_n > Q$ can be deleted with no substantial change in an algorithm's runtime.

SmartGreedy's allocative guarantee is much better than Greedy's.

Proposition 3.2. SmartGreedy *is a* $\frac{1}{2}$*-approximation for the Knapsack problem.*

Proof. For any instance ω, order the bidders by value/size as in (14). If Greedy packs all bidders, then trivially $W^*(\omega) = W_{\text{SmartGreedy}}(\omega)$. Otherwise, let k be the lowest index of a bidder not packed by Greedy and let K be the index of a bidder with maximum value. Optimal welfare $W^*(\omega)$ is no more than the best solution to the linear program in which we can pack fractional bidders, which—given that we have sorted the bidders in descending order of value-to-size—in turn is no more than $\sum_{n=1}^{k} v_n$. It follows that

$$\begin{aligned} W^*(\omega) &\leq \sum_{n=1}^{k} v_n \\ &= W_{\text{Greedy}}(\omega) + v_k \\ &\leq W_{\text{Greedy}}(\omega) + v_K \\ &\leq 2 \max \{W_{\text{Greedy}}(\omega), v_K\} \\ &= 2W_{\text{SmartGreedy}}(\omega); \end{aligned}$$

hence, we see that SmartGreedy is a $\frac{1}{2}$-approximation, as desired. □

SmartGreedy is bossy, as our next example shows.

Example 3.3. Consider the knapsack instance with capacity 10 and three bidders; $v_1 = 2$, $v_2 = 1$, $v_3 = 8$, $q_1 = q_2 = 1$, and $q_3 = 9$. At this instance, SmartGreedy packs just bidder 3. If bidder 3 instead reports $v_3 = 10$, then SmartGreedy instead packs bidder 1 and bidder 3. Thus, we see that SmartGreedy is bossy. However, the adjustment just described is a confirming *positive* externality; raising the value of a packed bidder has strictly increased the welfare of other bidders.

The Greedy and SmartGreedy algorithms are XCONE.

Proposition 3.4. *For the knapsack problem, the* Greedy *algorithm and the* Smart-Greedy *algorithm are both XCONE.*

Proof. Consider the bidders sorted by the Greedy algorithm as in (14), and suppose the Greedy algorithm packs bidders 1 through k. If we raise the value of a packed bidder (without changing sizes), then the Greedy algorithm again packs bidders 1 through k. If we lower the value of an unpacked bidder, then the Greedy algorithm terminates no earlier than before, packing at least bidders 1 through k. The only confirming externalities are positive ones; hence the Greedy algorithm is XCONE.

Meanwhile, the algorithm that selects the most valuable single bidder is monotone and non-bossy and so is XCONE by Proposition 2.15, as well. Thus, by Proposition 2.17, the SmartGreedy algorithm is monotone, and so by Proposition 2.16 it is XCONE. □

Proposition 3.2, Proposition 3.4, and Theorem 2.13 yield the following corollary.

Corollary 3.5. *The* SmartGreedy *algorithm is a* $\frac{1}{2}$*-approximation for investment.*

The SmartGreedy algorithm has both confirming externalities (Example 3.3) and negative externalities, as the next example demonstrates. Crucially, it has no confirming negative externalities.

Example 3.6. Suppose we have three bidders with sizes $(.5, .5, .6)$ and values $(1, 1, 0)$, and a knapsack with capacity 1. The SmartGreedy algorithm packs the first two bidders. Raising the third bidder's value to 2 raises its payment by 1.2 but reduces the welfare of the other bidders by 2. However, this value change is not confirming.

3.1 Fully polynomial-time approximation schemes

If we restrict attention to XCONE algorithms for the knapsack problem, does that lead to a loss in allocative efficiency? State-of-the-art knapsack algorithms that run in polynomial time have stronger allocative guarantees than the SmartGreedy algorithm. Can fast XCONE algorithms be constructed that match their performance?

We will shortly answer these questions. Our construction below can be followed at two levels suitable for different readers. To follow it in full detail, readers should be acquainted with the theory of computation—in particular with how instances are represented as input strings and how running time is defined as a function of input size. These formalisms can be found in Arora and Barak (2009, pp. 9–37). Alternatively, readers can review the proofs that our new algorithms are XCONE while observing that that they inherit their performance guarantees and polynomial runtimes from the other algorithms used in the construction.

As is standard for computational analyses, we now assume that the bidders' values are non-negative integers.[18] Under this assumption, the input size is polynomial in the logarithm of the highest value $\log(\max_n\{v_n\})$ and the number of bidders $|N|$.[19]

We have used the SmartGreedy algorithm for illustration, but there are fast knapsack algorithms that do better. In particular, there exist families of algorithms indexed by parameter $\varepsilon > 0$, that yield $(1-\varepsilon)$-approximations for allocation, with running time polynomial in

[18]Real numbers can take infinitely many bits to represent, complicating statements about input size.

[19]It is conventional to take the logarithm with base 2, but this statement is true for any base.

ε^{-1} and the input size. Such a family is called a **fully polynomial-time approximation scheme** (FPTAS).

Briest et al. (2005) (henceforth BKV) constructed a weakly monotone FPTAS for the knapsack problem. Our construction modifies two steps in theirs to ensure that the algorithms have the XCONE property in addition to the weakly monotone FPTAS property.

Suppose we have some allocation instance with value profile v, and our desired allocative guarantee is $(1-\varepsilon)$. The first step of the BKV construction is to round each value to a grid. We define a family of modified value profiles, one for each non-negative integer $\ell \in \mathbb{N}$, essentially, censoring values above $2^{\ell+1}$ and then rounding the values to a grid with step size $\gamma_\ell := \frac{\varepsilon 2^\ell}{|N|}$. Formally, for given $\varepsilon > 0$ and ℓ, let us define a modified value profile $v^{\varepsilon,\ell}$ as follows:

1. $v'_n := \min\{v_n, 2^{\ell+1}\}$ (for all n).
2. $v_n^{\varepsilon,\ell} := \lfloor v'_n/\gamma_\ell \rfloor \cdot \gamma_\ell$ (for all n).

An exact optimum for the modified values $v^{\varepsilon,\ell}$ can be computed in polynomial time (for the textbook algorithm, see Williamson and Shmoys (2011, pp. 65–68)).

Let $x^*(\tilde{v})$ be any selection from the set of optimal allocations at value profile $\tilde{v}$ (implicitly, given the feasible allocations A), that is $x^*(\tilde{v}) \in \operatorname{argmax}_{a\in A}\left\{\sum_n [\tilde{v}_n \cdot a_n]\right\}$. Given parameter $\varepsilon > 0$, the BKV allocation rule (henceforth **the BKV rule**) selects an allocation in the infinite set

$$\left\{x^*\left(v^{\varepsilon,\ell}\right)\right\}_{\ell\in\mathbb{N}}$$

that maximizes performance according to the modified values, *i.e.* $\max_\ell \sum_n [v_n^{\varepsilon,\ell} \cdot x_n^*\left(v^{\varepsilon,\ell}\right)]$. One needs to search only a finite number of non-negative integers ℓ to find the desired maximum, because for large enough ℓ all the modified values round to 0. Combining these steps, the result is an algorithm that computes the BKV rule in polynomial time.

Proposition 3.7 (Briest et al. (2005))**.** *The BKV rule is weakly monotone, a* $(1-\varepsilon)$*-approximation for allocation, and can be computed in* $\operatorname{poly}\left(\varepsilon^{-1}, |N|, \log(\max_n\{v_n\})\right)$ *time.*

Despite the appealing properties described in Proposition 3.7, the BKV rule has confirming negative externalities because a large investment can make only large values of ℓ relevant in the preceding computation, damaging the efficiency of the allocation for other bidders. This is essentially the same vulnerability as arises for the satisficing algorithm in the Introduction. It turns out that the BKV investment guarantee can be arbitrarily bad, as we now state formally.

Proposition 3.8. *For all* $\delta > 0$*, there exists* $\varepsilon < \delta$ *such that the BKV rule with parameter* ε *has an investment guarantee of* 0.

Nevertheless, we can construct a XCONE FPTAS by modifying the BKV rule. First, instead of defining x^* to be an arbitrary maximizer, we limit the selection when there are multiple maximizers so that it never packs any bidders whose values are exactly 0 and breaks ties among maximizers using a strict ordering, so that x^* is non-bossy. Second, where the BKV selection is an allocation in the family $\left\{x^*\left(v^{\varepsilon,\ell}\right)\right\}_{\ell\in\mathbb{N}}$ that maximizes welfare under the *modified* values $\sum_n [v_n^{\varepsilon,\ell} \cdot a_n]$, ours is an allocation in the same family that maximizes welfare under the *actual* values $\sum_n [v_n \cdot a_n]$. For any parameter $\varepsilon > 0$, we use $\breve{x}^\varepsilon$ to denote the resulting allocation rule.

Proposition 3.9. *The allocation rule $\breve{x}^\varepsilon$ is XCONE, weakly monotone, and a $(1-\varepsilon)$-approximation for allocation.*

Proof. Let us define the allocation rule

$$\breve{x}^{\varepsilon,\ell}(v) \equiv x^*\left(v^{\varepsilon,\ell}\right).$$

The allocation rule x^* is weakly monotone, and the censoring and rounding operations are monotone transformations, so $\breve{x}^{\varepsilon,\ell}$ is weakly monotone. Moreover, x^* is non-bossy, so $\breve{x}^{\varepsilon,\ell}$ is non-bossy. Thus, $\breve{x}^{\varepsilon,\ell}$ is XCONE by Proposition 2.14 and Proposition 2.15.

Now, $\breve{x}^\varepsilon$ chooses the best output from the collection $\left\{\breve{x}^{\varepsilon,\ell}(v)\right\}_{\ell\in\mathbb{N}}$, so $\breve{x}^\varepsilon$ is weakly monotone by Proposition 2.17. Next, applying Proposition 2.16 yields the conclusion that $\breve{x}^\varepsilon$ is XCONE.

The BKV rule is a $(1-\varepsilon)$-approximation for allocation and chooses the allocation from the collection $\{\breve{x}^{\varepsilon,\ell}(v)\}_{\ell\in\mathbb{N}}$ that maximizes welfare under the modified values. The allocation $\breve{x}^\varepsilon(v)$ is selected from the same collection to maximize welfare under the actual values, so it achieves a weakly higher welfare than the BKV rule. Hence $\breve{x}^\varepsilon$ is a $(1-\varepsilon)$-approximation for allocation. □

Corollary 3.10. *The allocation rule $\breve{x}^\varepsilon$ is a $(1-\varepsilon)$-approximation for investment.*

Moreover, since $\breve{x}^\varepsilon$ is computed by tweaking the BKV algorithm, it inherits BKV's polynomial time property, resulting in a FPTAS.

Proposition 3.11. *The allocation rule $\breve{x}^\varepsilon$ can be computed in* $\operatorname{poly}\left(\varepsilon^{-1}, |N|, \log(\max_n\{v_n\})\right)$ *time.*

Proposition 3.9 and Proposition 3.11 demonstrate that good investment guarantees need not come at the cost of allocative guarantees; there is a FPTAS that achieves both. Although both the BKV FPTAS and our modification run in polynomial time, ours requires additional steps that incur a computational cost: our suggested FPTAS is slower than the BKV FPTAS. Further details are in the proof of Proposition 3.11.

We have focused on the knapsack problem for ease of exposition, but BKV showed how to construct a monotone FPTAS for a range of weakly NP-complete problems, such as job scheduling with deadlines and the constrained shortest-path problem.[20] Our method adapts easily to convert the BKV FPTAS to a XCONE FPTAS for those problems as well.

4 Discussion

Mechanism design analysis in economics has traditionally focused mostly on mechanisms that exactly optimize some objective like welfare, revenue, or consumer surplus, neglecting issues of computational hardness. Yet exact optimization is tractable only for small problems or problems with special structure.

Practical mechanisms without optimization can be created by using the large corpus of fast approximation algorithms developed by computer scientists, but doing that raises new questions. Approximation algorithms have heretofore been designed for short-run problems in which participants' values are fixed exogenously, but in practice, participants can often make *ex ante* investments that alter their values. In this paper, we study investment incentives in a class of environments in which there is a finite number of outcomes for each bidder and the bidder's possible values lie in a product of intervals. Focusing on deterministic mechanisms, we asked three general questions:

1. *Can mechanisms based on approximation algorithms avoid distorting participants' investment incentives, as VCG mechanisms, based on optimization, do?*

2. *When can mechanisms based on approximation algorithms preserve their performance guarantees even when an investment decision is added?*

3. *Does adding a requirement of robustness to the addition of investment decisions reduce the speed and allocative guarantees achievable by approximation algorithms?*

To frame the first question, we began by showing that that the externalities from any truthful mechanism depend only on the algorithm, and not on which prices are used to promote truthful reporting. For that reason, we call these "algorithmic externalities." Then, for the first question, we find a negative answer: unless the algorithm mimics welfare maximization on some possibly limited set of allocations, there are necessarily non-zero externalities

[20]If an allocation problem is NP-complete, but one can find an optimal allocation with running time polynomial in the *numeric value* of the largest integer in the input, then it is called "weakly NP-complete." Note that such an algorithm might still run in time exponential in the *length* of the input.

that can cause privately profitable investments to reduce welfare or welfare-increasing investments to be privately unprofitable.

The analysis of investment guarantees hinges on a new category of externalities that we dub "confirming" externalities. These arise when a bidder changes its report in a way that raises the relative value of its outcome but results in an externality to others. We show that the worst-case performance guarantee is robust to investments if and only if its confirming negative externalities are not too large.[21] That condition, however, can be hard to verify, so we also offer a sufficient condition—XCONE—that can be easier to check. An XCONE algorithm is one that excludes confirming negative externalities, but may have confirming positive externalities. We show that, for some algorithms for the knapsack problem including Greedy and SmartGreedy, the XCONE condition can be checked and verified without much difficulty. However, the XCONE condition also fails for some algorithms with very good—even arbitrarily good—performance for the short-run allocation problem.

For the third question, we study a particular FPTAS for the knapsack problem, modifying it with robustness to investments in mind. The result is a new XCONE FPTAS—a collection of algorithms that, for every ε, is XCONE, always achieves at least a $1-\varepsilon$ fraction of the optimum, and runs in time that is polynomial in the size of the input and ε^{-1}.

More broadly, there is a long tradition in economics of studying the performance of a competitive equilibrium, which assumes that all decisions, short-run and long-run, are guided by optimization. Because some problems are too hard for optimization, it is important to generalize that conception is to study economies in which approximation algorithms replace optimization, but that change raises many new considerations. Besides investment incentives, approximation can also affect how participants understand mechanisms in practice, raise new opportunities for coordination or collusion, and influence post-auction resale markets. Given the close connection between weakly monotone algorithms and truthful mechanisms, it seems possible—and important—to analyze how these and other economic properties of mechanisms reflect properties of their underlying algorithms.

[21] Randomized algorithms are not covered by this analysis because they can create additional inefficiencies, besides those from CONEs, by making investment returns uncertain. For example, consider the randomized algorithm that packs a knapsack optimally or leaves it empty, each with probability 1/2. This algorithm is a 1/2-approximation for allocation and has no externalities, so it is XCONE. It is also maximal-in-distributional-range, in the sense of Dughmi and Roughgarden (2014). Yet consider its investment performance in an instance with just one bidder that fits into the knapsack. Suppose that the bidder can choose value 0 at cost 0 or value 3 at cost 2. In the threshold auction based on the approximation algorithm, the bidder will find that it is not profitable to invest, so the net welfare will be 0. But it is socially optimal for the bidder to invest and be packed, for net welfare 1. Thus, the randomized algorithm's investment guarantee is 0.

References

Arora, S. and B. Barak (2009): *Computational complexity: a modern approach*, Cambridge University Press.

Arozamena, L. and E. Cantillon (2004): "Investment incentives in procurement auctions," *Review of Economic Studies*, 71, 1–18.

Bergemann, D. and J. Välimäki (2002): "Information Acquisition and Efficient Mechanism Design," *Econometrica*, 70, 1007–1033.

Bienstock, D. and A. Verma (2019): "Strong NP-hardness of AC power flows feasibility," *Operations Research Letters*, 47, 494–501.

Bikhchandani, S., S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen (2006): "Weak Monotonicity Characterizes Deterministic Dominant-Strategy Implementation," *Econometrica*, 74, 1109–1132.

Bikhchandani, S., S. De Vries, J. Schummer, and R. V. Vohra (2011): "An ascending vickrey auction for selling bases of a matroid," *Operations Research*, 59, 400–413.

Briest, P., P. Krysta, and B. Vöcking (2005): "Approximation techniques for utilitarian mechanism design," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 39–48.

Cook, S. (2006): "The P versus NP problem," in *The Millennium Prize Problems*, ed. by J. A. Carlson, A. Jaffe, and A. Wiles, American Mathematical Society Providence, 87–104.

Dantzig, G. B. (1957): "Discrete-variable extremum problems," *Operations research*, 5, 266–288.

Dobzinski, S. and N. Nisan (2007): "Mechanisms for multi-unit auctions," in *Proceedings of the 8th ACM conference on Electronic commerce*, 346–351.

Dughmi, S., J. D. Hartline, R. Kleinberg, and R. Niazadeh (2017): "Bernoulli factories and black-box reductions in mechanism design," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 158–169.

Dughmi, S. and T. Roughgarden (2014): "Black-box randomized reductions in algorithmic mechanism design," *SIAM Journal on Computing*, 43, 312–336.

Fortnow, L. (2009): "The status of the P versus NP problem," *Communications of the ACM*, 52, 78–86.

Gershkov, A., B. Moldovanu, P. Strack, and M. Zhang (2020): "A Theory of Auctions with Endogenous Valuations," *Journal of Political Economy.*

Hartline, J. D. (2016): "Mechanism design and approximation," *Book draft*, 122.

Hartline, J. D. and B. Lucier (2015): "Non-optimal mechanism design," *American Economic Review*, 105, 3102–24.

Hatfield, J. W., F. Kojima, and S. D. Kominers (2014): "Investment Incentives in Labor Market Matching," *American Economic Review Papers & Proceedings*, 104, 436–441.

——— (2019): "Strategy-Proofness, Investment Efficiency, and Marginal Returns: An Equivalence," Becker Friedman Institute Working Paper.

Holzman, R., N. Kfir-Dahav, D. Monderer, and M. Tennenholtz (2004): "Bundling equilibrium in combinatorial auctions," *Games and Economic Behavior*, 47, 104–123.

Karp, R. M. (1972): "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ed. by R. E. Miller and J. W. Thatcher, Springer, 85–103.

Lavaei, J. and S. H. Low (2011): "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, 27, 92–107.

Lavi, R., A. Mu'Alem, and N. Nisan (2003): "Towards a characterization of truthful combinatorial auctions," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 574–583.

Lehmann, D., R. Müller, and T. Sandholm (2006): "The Winner Determination Problem," in *Combinatorial Auctions*, ed. by P. Cramton, Y. Shoham, and R. Steinberg, MIT Press, 297–318.

Lehmann, D., L. I. Oćallaghan, and Y. Shoham (2002): "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM*, 49, 577–602.

Leyton-Brown, K., P. Milgrom, and I. Segal (2017): "Economics and computer science of a radio spectrum reallocation," *Proceedings of the National Academy of Sciences*, 114, 7202–7209.

Milgrom, P. (2017): *Discovering Prices*, Columbia University Press.

Milgrom, P. and I. Segal (2002): "Envelope theorems for arbitrary choice sets," *Econometrica*, 70, 583–601.

——— (2020): "Clock auctions and radio spectrum reallocation," *Journal of Political Economy*, 128, 1–31.

Mu'Alem, A. and N. Nisan (2008): "Truthful approximation mechanisms for restricted combinatorial auctions," *Games and Economic Behavior*, 64, 612–631.

Nisan, N. and A. Ronen (1999): "Algorithmic mechanism design," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 129–140.

——— (2007): "Computationally feasible VCG mechanisms," *Journal of Artificial Intelligence Research*, 29, 19–47.

Pardalos, P. M., D.-Z. Du, and R. L. Graham (2013): *Handbook of Combinatorial Optimization*, Springer.

Rogerson, W. P. (1992): "Contractual solutions to the hold-up problem," *Review of Economic Studies*, 59, 777–793.

Saks, M. and L. Yu (2005): "Weak monotonicity suffices for truthfulness on convex domains," in *Proceedings of the 6th ACM Conference on Electronic Commerce*, 286–293.

Sandholm, T. (2002): "Algorithm for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, 135, 1–54.

Tomoeda, K. (2019): "Efficient investments in the implementation problem," *Journal of Economic Theory*, 182, 247–278.

Vickrey, W. (1961): "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, 16, 8–37.

Williamson, D. P. and D. B. Shmoys (2011): *The Design of Approximation Algorithms*, Cambridge University Press.

A Proofs omitted from the main text

Preliminaries & Notation

Before getting into the proofs, we introduce a few notations and conventions. The value profile for bidders other than n, v_{-n}; the set of possible allocations, A; the algorithm, x; and the probability distribution, g, usually do not change within a given proof. Therefore we often suppress the dependence on these parameters to ease notation (see Table 2).

Meanwhile, truthfulness of the mechanism (x,p) implies that for every allocation instance (v,A), x assigns each bidder n an outcome that maximizes its value minus its threshold price. We call this maximum the bidder's **normalized utility** and denote it as follows:

$$u_n(v,A,x) \equiv [v_n - \tau_n(v_{-n},A,x)] \cdot x_n(v,A) = \max_{o\in O} \left\{v_{n,o} - \tau_{n,o}(v_{-n},A,x)\right\}.$$

The normalized utility corresponds to the bidder's utility in the mechanism with prices $p_n(v,A) = \tau_n(v_{-n},A,x)\cdot x_n(v,A)$; other truthful mechanisms may shift prices and utility by a strategically irrelevant additive term. By construction, we have $u_n(v,A,x) \geq 0$.

We extend the normalized utility notation u to the case of investment in a natural way as follows:

$$u_\iota((v_\iota,c),v_{-\iota},A,x) \equiv \max_{o\in O} \left\{v_{\iota,o} - \tau_{\iota,o}(v_{-\iota},A,x)\right\} - c,$$

$$u_\iota(I,v_{-\iota},A,x) \equiv \max_{(v_\iota,c)\in I} \left\{u_\iota((v_\iota,c),v_{-\iota},A,x)\right\}.$$

This allows us to talk about normalized utility for investor facing a cost or an investment opportunity.

Notation	Description	Suppressed Notation
$x(v_n,v_{-n},A)$	allocation of algorithm x	$x(v_n)$
$W_x(v_n,v_{-n},A)$	welfare of algorithm x	$W_x(v_n)$
$W^*(v_n,v_{-n},A)$	welfare of the optimal algorithm	$W^*(v_n)$
$\mathrm{BR}(x,g,I,v_{-n},A)$	best response for the investor	$\mathrm{BR}(x,I)$
$\tau_n(v_{-n},A,x)$	threshold price	τ_n
$u_n(v_n,v_{-n},A,x)$	normalized utility of bidder n	$u_n(v_n)$
$u_\iota((v_\iota,c),v_{-\iota},A,x)$	normalized utility of bidder ι	$u_\iota(v_\iota,c)$
$u_\iota(I,v_{-\iota},A,x)$	normalized utility of bidder ι	$u_\iota(I)$
$p_n(v_n,v_{-n},A)$	price paid by bidder n	$p_n(v_n)$
$\mathcal{E}_x(\tilde{v}_n,(v,A))$	externality from changing v_n to $\tilde{v}_n$	$\mathcal{E}_x(\tilde{v}_n,v_n)$

Table 2: Correspondence table for the simplified notation we often use in this section

Proof of Theorem 2.4

Theorem 2.4. *If algorithm x is weakly monotone, then x has no algorithmic externalities ($\mathcal{E}_x \equiv 0$) if and only if x is welfare-equivalent to a maximal-in-range algorithm.*

If x is welfare-equivalent to a maximal-in-range algorithm with some range R, then the welfare coincides with that of the VCG mechanism restricted to R. Since a VCG mechanism has zero externalities, x also has zero externalities because of the following lemma.

Lemma A.1. *If algorithm x and x' are weakly monotone and welfare-equivalent, then they have the same threshold prices and externalities.*

Proof. Assume x and x' are welfare-equivalent and have different threshold prices, say $\tau_{n,o}(x) < \tau_{n,o}(x')$. Then there exists some value v_n where $v_{n,o} < \tau_{n,o}(x')$ and $x_n(v_n) = o$. Let us define v'_n with $v'_{n,o} \in [v_{n,o}, \tau_{n,o}(x'))$, and $v'_{n,o'} = v'_{n,o'}$ for $o' \neq o$. By monotonicity, $x_n(v'_n) = o$—but by the definition of threshold price, $x'_n(v'_n) \neq o$. However while there are infinitely many such $v'_{n,o} \in [v_{n,o}, \tau_{n,o}(x'))$, there are only a finite set of values $v'_{n,o}$ where an allocation a with $a_n = o$ can have the same welfare as an allocation a' with $a'_n \neq o$. Thus, we obtain a contradiction to the hypothesis that x and x' are welfare-equivalent. Therefore x and x' must have the same threshold prices. Now,

$$\begin{aligned}
\mathcal{E}_{x,p}(\tilde{v}_n, v_n) \equiv \; & \underbrace{p_n(\tilde{v}_n) - p_n(v_n)}_{\text{change in } n\text{'s threshold payment}} + \underbrace{\sum_{m \neq n} v_m \cdot [x_m(\tilde{v}_n) - x_m(v_n)]}_{\text{effect on others' welfare}} \\
&= W_x(\tilde{v}_n) - u_n(\tilde{v}_n) - W_x(v_n) + u_n(v_n) \\
&= W_x(\tilde{v}_n) - \max_{o \in O} \{\tilde{v}_{n,o} - \tau_{n,o}\} - W_x(v_n) + \max_{o \in O} \{v_{n,o} - \tau_{n,o}\},
\end{aligned}$$

so the externalities of x only depend on the welfare and threshold price functions—and similarly for x'.[22] Thus, since x and x' welfare-equivalent (by hypothesis) and have the same threshold prices (by the first part of the lemma), we see that they have the same externalities. □

Let $W(v, a) \equiv \sum_n [v_n \cdot a_n]$ denote the welfare of allocation a at a value profile v. We write $a \simeq a'$ if $W(v, a) = W(v, a')$ for every value profile v. Define the **modified domain** as $D = \{v \in V \mid W(v, a) = W(v, a') \implies a \simeq a'\}$ and the **modified range** as $R = x(D)$.

In outline, our proof of the forwards direction of Theorem 2.4 constructs the x' welfare-equivalent to x as follows. We have just defined a subdomain of values $D \subseteq V$ on which

[22] Here we use the normalized utility notation introduced in the Preliminaries & Notation section at the start of the appendix.

no two essentially different allocations have the same welfare. Any algorithm x' that is welfare-equivalent to x must satisfy $x'(v) = x(v)$ for $v \in D$. For $v \notin D$, we use the monotonicity and zero-externality properties of x to show that there exists $a \in R$ such that $W(v, a) = W(v, x(v))$ and set $x'(v) := a$. This ensures that x and x' are welfare-equivalent. We then use the same properties to establish that x' is a maximal-in-range algorithm with range R, which finishes the proof.

Claim A.2. *Algorithm x is welfare-equivalent to a maximal-in-range algorithm with range R.*

Proof. The proof relies on the following two lemmatta. The first characterizes properties of the welfare function for an algorithm that has no externalities. The second shows that the modified domain D is dense.

Lemma A.3. *If algorithm x has no externalities, then W_x is*

- *non-decreasing in v_n and*
- *1-Lipschitz in v_n*

in the sup *norm.*

Proof. Recall that the formula for externalities is

$$\begin{aligned}
\mathcal{E}_{x,p}(\tilde{v}_n, (v, A)) &\equiv \underbrace{p_n(\tilde{v}_n, v_{-n}, A) - p_n(v, A)}_{\text{change in } n\text{'s threshold payment}} + \underbrace{\sum_{m \neq n} v_m \cdot [x_m(\tilde{v}_n, v_{-n}, A) - x_m(v, A)]}_{\text{effect on others' welfare}} \\
&= W_x(\tilde{v}_n) - u_n(\tilde{v}_n) - W_x(v_n) + u_n(v_n).
\end{aligned}$$

Since x has no externalities, $\mathcal{E}_{x,p}(\tilde{v}_n, (v, A)) = 0$, so $W_x(v_n) - u_n(v_n) = W_x(\tilde{v}_n) - u_n(\tilde{v}_n)$. Therefore, $W_x(\cdot)$ is a constant plus $u_n(\cdot)$. It is clear that $u_n(\cdot)$ is nondecreasing and 1-Lipschitz ; hence we see that W_x is nondecreasing and 1-Lipschitz, as well. □

Lemma A.4. *The modified domain D is non-empty. In addition, for every $v \in V$, $d \in D$, and $\varepsilon > 0$, there exists $v' \in D$ such that*

- *$\|v' - v\| < \varepsilon$, and*
- *$\prod_{n,o}\{v'_{n,o}, d_{n,o}\} \subseteq D$.*

Proof. To show D is non-empty, we construct a value profile $d \in D$. We describe its components $d_{n,o}$ by ordering the pairs (n, o), beginning with the pairs (n, o) for which $V_{n,o}$ is a singleton and listing the rest in arbitrary order, indexed by k. In step $k = 0$, for each pair (n, o) such that $V_{n,o}$ is a singleton, we fix $d_{n,o}$ to be equal to the sole element of $V_{n,o}$. Set $B_0 := \{d_{n,o} \in V_{n,o} : V_{n,o} \text{ is a singleton}\}$ (and $B_0 = \emptyset$ if no $V_{n,o}$ is a singleton). For each step $k \geq 1$, given a finite set B_{k-1} and any two subsets $B', B'' \subseteq B_{k-1}$, consider these equations:

$$d_{n,o} + \sum_{b \in B'} b = \sum_{b \in B''} b.$$

There are finitely many such equations and, for $k \geq 1$, the interval $V_{n,o}$ has non-empty interior, so there exists some $d_{n,o} \in V_{n,o}$ that satisfies none of these equations. We set $B_k := B_{k-1} \cup \{d_{n,o}\}$ and iterate until d has been constructed. Suppose that allocations a and a' satisfy $W(d, a) = W(d, a')$. Then, for all n, either $a_n = a'_n$ or V_{n,a_n} and V_{n,a'_n} are both singletons, so $a \simeq a'$ and hence $d \in D$.

The second half of the lemma is proved by constructing v' in a similar way. Call a value v' **generic (with respect to d)** if $\prod_{n,o}\{v'_{n,o}, d_{n,o}\} \subseteq D$. Start with $v' = d$. If $V_{n,o}$ is a singleton, then $v'_{n,o} = v_{n,o}$. Otherwise, $V_{n,o}$ is an interval with non-empty interior, so there exists some value $v'_{n,o}$ within $\frac{\varepsilon}{|N| \cdot |O|}$ of $v_{n,o}$ that keeps v' generic (because only a finite number of choices for $v'_{n,o}$ would result in v' being non-generic). □

Now we show that algorithm x is welfare-equivalent to some algorithm whose feasible set is R. Because the welfare $W(v, a)$ is continuous in v and the set D is dense in V, the welfare of x at any value profile (even outside of D) must be equal to the welfare of some allocation in R at that value profile.[23]

Formally, for any $\varepsilon > 0$ and $v \in V$, we can find $d \in D$ so that $\|v - d\| < \varepsilon$.[24] By A.3, we get

$$|W_x(v) - W_x(d)| < \varepsilon$$

and we have

$$|W_x(d) - W(v, x(d))| < \varepsilon.$$

By the triangle inequality, we then have

$$W_x(v) < W(v, x(d)) + 2\varepsilon \leq \left(\max_{r \in R}\{W(v, r)\}\right) + 2\varepsilon. \tag{15}$$

[23]The formal proof that follows only shows the welfare is less than or equal to $\max_{r \in R}\{W(v, r)\}$ since this is all that is necessary for the proof of Theorem 2.4.

[24]This follows from Lemma A.4.

Since (15) is true for any $\varepsilon > 0$,

$$W_x(v) \le \max_{r \in R}\{W(v, r)\}. \tag{16}$$

Now we show that the welfare of x at a value profile v is bounded below by the welfare of any allocation $r \in R$ at that value profile. This is because if we start at a value profile d where the allocation is r and look at the welfare as we change the value profile to v, weak monotonicity ensures certain changes do not change the allocation r, and the welfare function being 1-Lipschitz ensures that the other changes weakly increase the algorithm's welfare compared to the welfare of allocation r.

Formally, for any $r \in R$, let $d \in D$ be a value profile where $r = x(d)$. For any $v \in V$ and $\varepsilon > 0$ we can construct v' as in Lemma A.4. Consider the profile v'' where

$$v''_{n,o} = \begin{cases} \max\{d_{n,o}, v'_{n,o}\} & x_n(d) = o \\ \min\{d_{n,o}, v'_{n,o}\} & x_n(d) \neq o. \end{cases}$$

We now prove that $x(v'') \simeq r$. Consider changing the value profile from d to v'' one element at a time, and let v^k denote the value profile in step k. At each step we have $x(v^k) \simeq r$. We argue by contradiction; suppose at some step we are at value profile v^k and $x(v^k) \simeq r$, but $x(v^{k+1}) \not\simeq r$. We define $\tilde{\alpha} \equiv \inf\{\alpha \in [0,1] : x(\alpha v^{k+1} + (1-\alpha)v^k) \not\simeq r\}$, and $\tilde{v} \equiv \tilde{\alpha} v^{k+1} + (1-\tilde{\alpha})v^k$. We can find an allocation $a \not\simeq r$ and a sequence of value profiles $(v^l)_{l=1}^{\infty}$ such that $x(v^l) = a$, v^l is a convex combination of v^k and v^{k+1}, and $\lim_{l\to\infty} v^l = \tilde{v}$. By continuity of W_x (from Lemma A.3), we have

$$W(\tilde{v}, a) = \lim_{l\to\infty} W_x(v^l) = W_x(\tilde{v}) = \lim_{l\to\infty} W_x((1/l)v^k + (1-1/l)\tilde{v}) = W(\tilde{v}, r). \tag{17}$$

Moreover because x is weakly monotone we have $a_{n,o} = r_{n,o}$ for the n, o pertaining to step k. The value profiles v^k and $\tilde{v}$ are identical except for the element pertaining to n, o, so (17) implies that $W(v^k, a) = W(v^k, r)$. But since the value profile v^k is in D (by construction of v' and v''), it must be that $a \simeq r$, a contradiction. We have proven that $x(v'') \simeq r$, so

$$W_x(v'') = W(v'', r).$$

By Lemma A.3, the welfare function is nondecreasing and 1-Lipschitz, so[25]

$$W_x(v') \geq W_x(v'') - \sum_{x_n(d)=o} v''_{n,o} - v'_{n,o} = W(v', r).$$

Since we could construct v' for any $\varepsilon > 0$ (by Lemma A.4) and the welfare function $W_x(\cdot)$ is continuous, we have

$$W_x(v) \geq W(v, r);$$

since this is true for all $r \in R$, we have

$$W_x(v) \geq \max_{r \in R}\{W(v, r)\}. \tag{18}$$

Combining (16) and (18), we get

$$W_x(v) = \max_{r \in R}\{W(v, r)\},$$

so x is welfare-equivalent to a maximal-in-range algorithm with range R. □

Proof of Theorem 2.6

Theorem 2.6. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, x is a β-approximation for investment if and only if x is a β-approximation for certain investment.*

The certain investment instances are a subset of the investment instances. Thus, if x is a β-approximation for investment, then x is a β-approximation for certain investment.

We now prove the other direction. Suppose we have some investment instance $(g, I, \nu_{-\iota}, \mathcal{A})$ and some algorithm x that is a β-approximation for certain investment. Going off the intuition described in the main text, we seek to construct state-dependent cost functions $\tilde{c} : S \to \mathbb{R}$ to make the *ex post* normalized utility from the investment constant (and equal to the *ex ante* normalized expected utility from that investment).

Now, the state-dependent cost function $\tilde{c} : S \to \mathbb{R}$ that gives constant utility in each state for investment (ν_ι, c) is given by

$$\tilde{c}(s) \equiv u_\iota(\nu(s), \mathcal{A}(s), x) - \underbrace{\left(\left[\sum_{s'} g(s')u_\iota(\nu(s'), \mathcal{A}(s'), x)\right] - c\right)}_{\textit{ex ante} \text{ normalized utility from } (\nu_\iota, c)}.^{26}$$

[25] By construction, $v''_{n,o} \geq v'_{n,o}$ if $x_n(d) = o$ and $v''_{n,o} \leq v'_{n,o}$ if $x_n(d) \neq o$. We use the fact that the welfare is 1-Lipschitz in the first case and nondecreasing in the second case.

[26] Here we use the normalized utility notation introduced in the Preliminaries & Notation section at the

Observe that $\sum_s g(s)\tilde{c}(s) = c$. By construction, the *ex post* normalized utility of choosing the investment with state-dependent-cost $(\nu_\iota, \tilde{c})$, *i.e.* $u_\iota(\nu(s), \mathcal{A}(s), x) - \tilde{c}(s)$, is in every state equal to the *ex ante* normalized utility of the original investment (ν_ι, c).

However, we need a status quo alternative so that the realization in each state is a valid investment instance, so we introduce a new investment option with 0 normalized utility and 0 cost. Formally, we define an investment option that yields a value in each state equal to the investor's threshold prices

$$\nu'_\iota(s) \equiv \tau_\iota(\nu_{-\iota}(s), \mathcal{A}(s), x).$$

Let I' consist of the investments modified to have state-dependent costs, as well as the status quo alternative, that is $I' \equiv \{(\nu_\iota, \tilde{c}) : ((\nu_\iota, c) \in I\} \cup \{(\nu'_\iota, 0)\}$. For each state s, the tuple $(I'(s), \nu_{-\iota}(s), \mathcal{A}(s))$ is a certain investment instance. Since x is a β-approximation for certain investment, for any $(\nu_\iota, c) \in \mathrm{BR}(x, g, I, \nu_{-\iota}, \mathcal{A})$ and any state s, we have $W_x(\nu(s), \mathcal{A}(s)) - \tilde{c}(s) \geq \beta \overline{W}^*(I'(s), \nu_{-\iota}(s), \mathcal{A}(s))$.

Thus, for any $(\nu_\iota, c) \in \mathrm{BR}(x, g, I, \nu_{-\iota}, \mathcal{A})$ such that $\overline{W}_x(I, \mathcal{A}) = [\sum_s g(s) W_x(\nu_\iota(s), \mathcal{A}(s))] - c$, we conclude that

$$\begin{aligned}\overline{W}_x(I, \nu_{-\iota}, \mathcal{A}) &= \left[\sum_s g(s) W_x(\nu(s), \mathcal{A}(s))\right] - c = \sum_s g(s) \left[W_x(\nu(s), \mathcal{A}(s)) - \tilde{c}(s)\right] \\ &\geq \sum_s g(s) \beta \overline{W}^*(I'(s), \nu_{-\iota}(s), \mathcal{A}(s)) \geq \beta \overline{W}^*(I', \nu_{-\iota}, \mathcal{A}) \geq \beta \overline{W}^*(I, \nu_{-\iota}, \mathcal{A}),\end{aligned}$$

where the penultimate inequality holds because the expectation of the maximum is no less than the maximum of the expectation, and the final inequality holds because for all $(\nu_\iota, c) \in I$, we have $(\nu_\iota, \tilde{c}) \in I'$ with $\sum_s g(s)\tilde{c}(s) = c$.

Proof of Theorem 2.10

Theorem 2.10. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, if x has β-bounded confirming externalities, then x is a β-approximation for certain investment.*

We start with the following technical lemma.

Lemma A.5. *The optimal welfare function $W^*(v_n)$ is*

- *non-decreasing in v_n and*
- *1-Lipschitz in v_n*

start of the appendix.

in the sup *norm.*

Proof. This follows from Lemma A.3 because the optimal welfare function has zero externalities. $\square$

Next we state a technical condition that we will prove is equivalent to β-approximation for certain investment. The condition compares $W_x(v, A)$ minus that bidder's normalized utility to the optimal welfare when one bidder's value is set to the threshold vector.[27]

Definition A.6. For some $\beta \in [0, 1]$, algorithm x is **β-pivotal** if for any allocation instance (v, A) and any bidder n, we have

$$W_x(v, A) - u_n(v, A, x) \geq \beta \underbrace{W^*(\tau_n(v_{-n}, A, x), v_{-n}, A)}_{\text{optimal welfare at } n\text{'s threshold vector}} .$$

Lemma A.7. *For any weakly monotone algorithm x and any $\beta \in [0, 1]$, x is a β-approximation for certain investment if and only if x is β-pivotal.*

Proof. Being a β-approximation for investment means the welfare for algorithm x must be lower bounded by β times the optimal welfare for any investment instance. However, many of these lower bounds turn out to be redundant, and our argument shows that the only relevant bounds are those in the β-pivotality condition.

Algorithm x is a β-approximation for certain investment if and only if for every set of investments I, any $(v_\iota, c) \in \mathrm{BR}(x, I)$, and any $(v'_\iota, c') \in I$,

$$W_x(v_\iota) - c \geq \beta(W^*(v'_\iota) - c'). \tag{19}$$

If we restrict attention to investment sets of the form

$$I = \{(v_\iota, c), (v'_\iota, c'), (\tau_\iota, 0)\},$$

then we still obtain the bounds (19); however, we can rewrite the requirement that $(v_\iota, c) \in \mathrm{BR}(x, I)$ as $u_\iota(v_\iota, c) \geq \max\{u_\iota(v'_\iota, c'), 0\}$. By formulating (19) as

$$W_x(v_\iota) - u_\iota(v_\iota) \geq \beta(W^*(v'_\iota) - u_\iota(v'_\iota)) - u_\iota(v_\iota, c) + \beta u_\iota(v'_\iota, c'),$$

we notice that the tightest bound occurs when $u_\iota(v_\iota, c) = u_\iota(v'_\iota, c') = 0$. Therefore algorithm

[27] Here we use the normalized utility notation introduced in the Preliminaries & Notation section at the start of the appendix.

x is a β-approximation for certain investment if and only if for any v_ι

$$W_x(v_\iota) - u_\iota(v_\iota) \geq \max_{v'_\iota}\{\beta(W^*(v'_\iota) - u_\iota(v'_\iota))\}. \tag{20}$$

Since by Lemma A.5 we have

$$W^*(v'_\iota) - u_\iota(v'_\iota) \leq W^*(v'_\iota - u_\iota(v'_\iota)) \leq W^*(\tau_\iota),$$

we know the maximum in (20) occurs at $v'_\iota = \tau_\iota$; plugging this in, (20) becomes

$$W_x(v_\iota) - u_\iota(v_\iota) \geq \beta W^*(\tau_\iota),$$

which is exactly the β-pivotality condition. Therefore algorithm x is a β-approximation for certain investment if and only if x is β-pivotal. $\square$

We are now ready to prove Theorem 2.10. Suppose that x has β-bounded confirming externalities. If the change from the threshold price τ_n to any value v_n is a confirming change, then x having β-bounded confirming changes would immediately imply x is β-pivotal. Now, while it is not the case that the change from the threshold price τ_n to any value v_n is confirming, we show that it *is* true that for any value v_n, there exists a value v_n^ε arbitrarily close to τ such that the change from v_n^ε to v_n is a confirming change.

Formally, for any v_n and $\varepsilon \in (0, 1]$, we let

$$v_n^\varepsilon \equiv \varepsilon v_n + (1-\varepsilon)\tau_n.$$

Now, because x is truthful, we have

$$\begin{aligned} x_n(v_n^\varepsilon) \in \operatorname*{argmax}_o\{v_{n,o}^\varepsilon - \tau_{n,o}\} &= \operatorname*{argmax}_o\{(\varepsilon v_{n,o} + (1-\varepsilon)\tau_{n,o}) - \tau_{n,o}]\} \\ &= \operatorname*{argmax}_o\{\varepsilon[v_{n,o} - \tau_{n,o}]\} \\ &= \operatorname*{argmax}_o\{v_{n,o} - \tau_{n,o}\}. \end{aligned}$$

It follows that for any outcome $o' \in O$,

$$v_{n,x_n(v_n^\varepsilon)} - \tau_{n,x_n(v_n^\varepsilon)} \geq v_{n,o'} - \tau_{n,o'}. \tag{21}$$

Thus, we see that the change from v_n^ε to v_n confirms $x_n(v_n^\varepsilon)$ because

$$\begin{aligned} v_{n,x_n(v_n^\varepsilon)} - v_{n,o'} &= [(v_{n,x_n(v_n^\varepsilon)} - \tau_{n,x_n(v_n^\varepsilon)}) - (v_{n,o'} - \tau_{n,o'})] + \tau_{n,x_n(v_n^\varepsilon)} - \tau_{n,o'} \\ &\geq \varepsilon[(v_{n,x_n(v_n^\varepsilon)} - \tau_{n,x_n(v_n^\varepsilon)}) - (v_{n,o'} - \tau_{n,o'})] + \tau_{n,x_n(v_n^\varepsilon)} - \tau_{n,o'} \\ &= v^\varepsilon_{n,x_n(v_n^\varepsilon)} - v^\varepsilon_{n,o'}, \end{aligned}$$

where the inequality follows from (21).

Since the change from v_n^ε to v_n confirms $x_n(v_n^\varepsilon)$ and x has β-bounded confirming externalities, we have

$$p_n(v_n) - p_n(v_n^\varepsilon) + \sum_{m\neq n} (v_m \cdot [x_m(v_n) - x_m(v_n^\varepsilon)]) = \mathcal{E}_x(v_n, v_n^\varepsilon) \geq \beta W^*(v_n^\varepsilon) - W_x(v_n^\varepsilon). \quad (22)$$

Using the definition of normalized utility u_n, the inequality (22) becomes

$$W_x(v_n) - W_x(v_n^\varepsilon) - u_n(v_n) + u_n(v_n^\varepsilon) \geq \beta W^*(v_n^\varepsilon) - W_x(v_n^\varepsilon).$$

Canceling $W_x(v_n^\varepsilon)$ from both sides of (A) and taking the limit as $\varepsilon \to 0$, we get

$$W_x(v_n) - u_n(v_n) \geq \beta W^*(\tau_n).$$

Thus, we see that x is β-pivotal, and by Lemma A.7, x is a β-approximation for certain investment.

Proof of Proposition 2.17

Proposition 2.17. *Suppose that $|O| = 2$, and let X be a collection of weakly monotone XCONE algorithms. If y is an algorithm that at each instance $(v, A) \in \Omega$ outputs a welfare-maximizing allocation from the collection $\{x(v, A)\}_{x\in X}$, then y is weakly monotone.*

Suppose that y and X satisfy the assumptions of Proposition 2.17. We want to prove that for any (v, A) and $\tilde{v}_n$,

$$0 \leq [\tilde{v}_n - v_n] \cdot [y_n(\tilde{v}_n) - y_n(v_n)]. \quad (23)$$

If $y_n(\tilde{v}_n) = y_n(v_n)$ then (23) follows immediately. Suppose $y_n(\tilde{v}_n) \neq y_n(v_n)$. If the change from v_n to $\tilde{v}_n$ confirms $y_n(\tilde{v}_n)$ then (23) follows immediately. Suppose it does not confirm $y_n(\tilde{v}_n)$. Then by $y_n(\tilde{v}_n) \neq y_n(v_n)$ and $|O| = 2$, the change from v_n to $\tilde{v}_n$ confirms $y_n(v_n)$, and the change from $\tilde{v}_n$ to v_n confirms $y_n(\tilde{v}_n)$.

Let us pick $x, \tilde{x} \in X$ such that $x(v_n) = y(v_n)$ and $\tilde{x}(\tilde{v}_n) = y(\tilde{v}_n)$. We have

$$\begin{aligned} v_n \cdot \tilde{x}_n(v_n) + \sum_{m \neq n} v_m \cdot \tilde{x}(v_n) &\leq v_n \cdot x_n(v_n) + \sum_{m \neq n} v_m \cdot x_m(v_n) \\ &\leq v_n \cdot x_n(\tilde{v}_n) + \sum_{m \neq n} v_m \cdot x_m(\tilde{v}_n), \end{aligned} \tag{24}$$

where the first inequality is by construction and the second inequality is by x XCONE and weakly monotone and Proposition 2.7. A symmetric argument yields

$$\begin{aligned} \tilde{v}_n \cdot x_n(\tilde{v}_n) + \sum_{m \neq n} v_m \cdot x_m(\tilde{v}_n) &\leq \tilde{v}_n \cdot \tilde{x}_n(\tilde{v}_n) + \sum_{m \neq n} v_m \cdot \tilde{x}_m(\tilde{v}_n) \\ &\leq \tilde{v}_n \cdot \tilde{x}_n(v_n) + \sum_{m \neq n} v_m \cdot \tilde{x}_m(v_n). \end{aligned} \tag{25}$$

Adding inequalities (24) and (25) and canceling terms yields

$$0 \leq [\tilde{v}_n - v_n] \cdot [\tilde{x}_n(v_n) - x_n(\tilde{v}_n)]. \tag{26}$$

Since the change from v_n to $\tilde{v}_n$ confirms $y_n(v_n) = x_n(v_n)$, and x_n is weakly monotone, we have

$$0 = [\tilde{v}_n - v_n] \cdot [x_n(\tilde{v}_n) - x_n(v_n)]. \tag{27}$$

Similarly, since the change from $\tilde{v}_n$ to v_n confirms $y_n(\tilde{v}_n) = \tilde{x}_n(\tilde{v}_n)$, and $\tilde{x}_n$ is weakly monotone, we have

$$0 = [\tilde{v}_n - v_n] \cdot [\tilde{x}_n(\tilde{v}_n) - \tilde{x}_n(v_n)]. \tag{28}$$

Adding (26), (27), and (28) yields

$$0 \leq [\tilde{v}_n - v_n] \cdot [\tilde{x}_n(\tilde{v}_n) - x_n(v_n)] = [\tilde{v}_n - v_n] \cdot [y_n(\tilde{v}_n) - y_n(v_n)],$$

as desired.

Proof of Proposition 3.8

Proposition 3.8. *For all $\delta > 0$, there exists $\varepsilon < \delta$ such that the BKV rule with parameter ε has an investment guarantee of* 0.

The proof proceeds by reconstructing the satisficing example from the Introduction, but using the more complicated FPTAS algorithm. Both examples share the following properties:

1. Bidder 1 is packed even when its value is low and nearly the total value is derived from the packing of other bidders.

2. When bidder 1 does not invest, the other bidders are packed optimally, providing nearly all the value.

3. When bidder 1 does invest, the algorithm extracts nearly zero from its packing of the other bidders. The bidder's investment thus results in a large confirming negative externality, leading to an investment guarantee of zero.

In the following family of examples, we consider $\varepsilon = \frac{1}{2^i}$ for $i \in \mathbb{N}$.

Let there be $|N| = 2^j$ bidders where the first bidder is the investor, the second bidder has a value of 2, and the remaining $|N| - 2$ bidders have a value of 1. The knapsack can fit either the first two bidders or everyone except for the second bidder.

Consider what happens when the investor has value 0. For the BKV rule, the allocation that maximizes the rounded values will have $\ell \leq i + j$. For $\ell \leq i + j$, the BKV rule will pack everyone except for the second bidder. If $\ell > i + j$, the last $|N| - 2$ bidders' values will be rounded to 0.

If the investor increases its value to 2^{i+j+2}, the allocation that maximizes the rounded values will be for $\ell = i + j + 1$. If ℓ is smaller, the investor's value will be truncated and if ℓ is larger, the second bidder's value will be rounded down to zero. For $\ell = i + j + 1$, $\gamma_\ell = \frac{1}{2}$ so all values will be rounded down to the nearest even integer. Since the last $|N| - 2$ bidders' values are rounded down to 0, the BKV rule will pack the first two bidders.

As shown above, the investor increasing its value from 0 to 2^{i+j+2} results in a confirming negative externality and therefore the BKV rule is not XCONE. If the set of investments is $\{(0, 0), (2^{i+j+2}, 2^{i+j+2})\}$, then the performance under investment is $\frac{2}{|N|-2}$ which goes to 0 as $|N|$ goes to ∞.

Proof of Proposition 3.11

Proposition 3.11. *The allocation rule $\breve{x}^\varepsilon$ can be computed in* $\mathrm{poly}\left(\varepsilon^{-1}, |N|, \log(\max_n\{v_n\})\right)$ *time.*

If $\frac{\varepsilon 2^\ell}{|N|} > \max_n\{v_n\}$, then every value rounds to 0, and by construction $x^*(v^{\varepsilon,\ell})$ packs no bidders and thus yields 0 welfare. Consequently, it suffices to compute $x^*\left(v^{\varepsilon,\ell}\right)$ from $\ell = 0$ to $\ell = \lfloor \log(\varepsilon^{-1}|N| \max_n\{v_n\}) \rfloor + 1$ in order to find the best output from the collection $\left(\breve{x}^{\varepsilon,\ell}\right)_{\ell\in\mathbb{N}}$. Briest et al. (2005) proved that computing $x^*\left(v^{\varepsilon,\ell}\right)$ in each step takes $\mathrm{poly}\left(\varepsilon^{-1}, |N|, \log(\max_n\{v_n\})\right)$ time. Thus, we can compute $\breve{x}^\varepsilon$ in $\mathrm{poly}\left(\varepsilon^{-1}, |N|, \log(\max_n\{v_n\})\right)$ time, which completes the proof of Proposition 3.11.

As an aside, we note that our proposal runs more slowly than the BKV FPTAS. The BKV FPTAS searches ℓ from $\lfloor \log(\max_n\{v_n\}) \rfloor - \lceil \log((1-\varepsilon)^{-1}|N|) \rceil - 1$ to $\lfloor \log(\max_n\{v_n\}) \rfloor$. Our FPTAS searches a larger range, from 0 to $\lfloor \log(\varepsilon^{-1}|N| \max_n\{v_n\}) \rfloor + 1$. This is because we are choosing the maximal allocation according to v instead of $v^{\varepsilon,\ell}$, and must search a larger range to find the relevant maximum.