View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Harvard University - DASH

Approximately-Strategyproof and Tractable Multi-Unit Auctions

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters.

Citation	Kothari, Anshul, David C. Parkes, and Subhash Suri. 2005. Approximately-strategyproof and tractable multi-unit auctions. Decision Support Systems 39(1): 105-121. Previously published in Proceedings of the 4th ACM Conference on Electronic Commerce: San Diego, California, USA, June 9-12, 2003. New York, N.Y.: ACM Press, 166-175.
Published Version	doi:10.1016/j.dss.2004.08.009
Accessed	February 18, 2015 2:04:11 PM EST
Citable Link	http://nrs.harvard.edu/urn-3:HUL.InstRepos:4031557
Terms of Use	This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

(Article begins on next page)

Approximately-Strategyproof and Tractable Multi-Unit Auctions

Anshul Kothari [1]

Department of Computer Science, University of California at Santa Barbara, CA 93106.

David C. Parkes [2]

Division of Engineering and Applied Sciences, 33 Oxford Street, Harvard University, Cambridge, MA 02138.

Subhash Suri [1]

Department of Computer Science, University of California at Santa Barbara, CA 93106.

Abstract

We present an approximately-efficient and approximately-strategyproof auction mechanism for a single-good multi-unit allocation problem. The bidding language allows marginal-decreasing piecewise constant curves and quantity-based side constraints. We develop a fully polynomial-time approximation scheme for the multi-unit allocation problem, which computes a $(1+\epsilon)$-approximation in worst-case time $T = O(n^3/\epsilon)$, given n bids each with a constant number of pieces. We integrate this approximation scheme within a Vickrey-Clarke-Groves mechanism and compute payments for an asymptotic cost of $O(T \log n)$. The maximal possible gain from manipulation to a bidder in the combined scheme is bounded by $\epsilon V/(1+\epsilon)$, where V is the total surplus in the efficient outcome.

Key words: Approximation Algorithm, Multi-unit Auctions, Strategyproof, Approximately-Strategyproof, Bidding Language

Email addresses: kothari@cs.ucsb.edu (Anshul Kothari), parkes@eecs.harvard.edu (David C. Parkes), suri@cs.ucsb.edu (Subhash Suri).

[1] Supported in parts by NSF grant IIS-0121562.

[2] Supported in parts by NSF grant IIS-0238147.

1 Introduction

Electronic markets facilitate new methods for procurement through expressive bidding and automated winner-determination. Electronic markets have been used to sell wireless spectrum [6], to procure school meals by the Chilean government [8], for course-registration at Chicago GSB [13], and for the procurement of logistics services [22]. Both economic and computational considerations are central to the design of useful electronic markets. Economic desiderata for markets can include allocative-efficiency and revenue-optimality [19]. Computational desiderata for markets can include tractable bidding strategies [26], minimal preference elicitation [27], and tractable winner determination [29].

Supporting simple truth-revealing bidding strategies in a dominant-strategy equilibrium has received attention in the literature [33,26]. This property is called *strategyproofness*. A strategyproof auction is useful because it simplifies the bidding problem: the optimal bidding strategy is known to bidders and does not require that a bidder models the preferences or strategies of other participants. What often emerges is an interesting tradeoff between providing tractable winner-determination algorithms and supporting strategyproofness. Well known economic mechanisms that provide strategyproofness can require that the marke-tmaker solves intractable winner-determination problems, and truth-revelation can quickly unravel when approximations are introduced [25]. Our results are positive: we present auction models that are tractable and for which truthful bidding is almost the dominant strategy for a bidder.

We consider a multi-unit allocation problem, that models both the problem of a seller with multiple identical units of a good and the problem of a buyer that seeks to procure multiple identical units of a good. The problem is motivated by recent trends in electronic commerce; for instance, corporations are increasingly using auctions for their strategic sourcing of commodity goods [15]. We provide a compact and expressive bidding language that allows marginal-decreasing piecewise constant price-schedules together with quantity-based side constraints. Our main contribution is to present a fully polynomial-time approximation scheme, that is both approximately efficient and approximately strategyproof.

In the procurement setting we consider a buyer with value $V > 0$ for M units of a good, and n suppliers each with a marginal-decreasing piecewise-constant cost function. The bidding language also allows each supplier to express an *upper bound* (or capacity constraint) on the number of units she can supply. This allows a supplier to express an infinite ask price for supplying large numbers of units that are beyond her capacity and is an important consideration in practical settings. As a concrete example, the procurement auction models the procurement of circuit boards in flexible sized lots from multiple suppliers each of which can state a capacity constraint.

In the forward auction setting we consider a seller with M units of a good, and n buyers each with a marginal-decreasing piecewise-constant valuation function. Here, the language also allows a buyer to express a *lower bound* (or minimum lot size) on the number of units that she will buy. This allows a buyer to submit a bid price of zero for initial units, and is an important consideration in practical settings. As a concrete example, our forward auction models a setting in which a PC manufacturer would like to sell excess inventory in flexible-size lots and each buyer has a minimal number of units that she must procure.

We consider the computational complexity of implementing the Vickrey-Clarke-Groves (VCG) [32,5,14] mechanism for this multi-unit allocation problem. The VCG mechanism is strategyproof for suppliers in the procurement auction and strategyproof for buyers in the forward auction, and supports allocative-efficiency such that purchasing (selling) decisions are made to maximize the total economic surplus in the economy. There is an asymmetry between the procurement and the forward auction directions because we model the buyer in the procurement setting with a finite value for the goods, while the seller in the forward auction is assumed to have no intrinsic value for the goods on sale. This asymmetry limits the application of our procurement auction to those in which the total payments collected by suppliers are less than the buyer's value. No such restriction is required in the forward auction direction.

The winner-determination problem in the multiunit allocation problem is (weakly) intractable, and has the classic 0/1 knapsack problem as a special case. The quantity-based side constraints preclude the adoption of a simple greedy allocation scheme. The winner-determination problem is a novel and interesting generalization of the classic knapsack problem. We provide a fully polynomial-time approximation scheme, computing a $(1+\epsilon)$-approximation in worst-case time $T = O(n^3/\varepsilon)$, where each bid has a fixed number of piecewise constant pieces.

We demonstrate that the approximate VCG mechanism, in which this $(1+\epsilon)$-scheme is used for winner-determination and to compute payments is $(\frac{\varepsilon}{1+\varepsilon})$-strategyproof. This means that a bidder can gain at most $(\frac{\varepsilon}{1+\varepsilon})V$ from a non-truthful bid, where V is the total surplus from the efficient allocation. As such, this is an example of a computationally-tractable ε-dominance result.[3] In practice, we can have good confidence that bidders without good information about the bidding strategies of other participants will have little to gain from attempts at manipulation. Formally, we can justify truthful bidding in equilibrium by modeling bidders as indifferent to payments that are within $(\frac{\varepsilon}{1+\varepsilon})V$ of each other.

[3] However, this may not be an example of what Feigenbaum & Shenker [10] refer to as a *tolerably-manipulable* mechanism because we have not tried to bound the effect of such a manipulation on the efficiency of the outcome. VCG mechanism do have a natural "self-correcting" property, though, because a useful manipulation to an agent is a reported value that *improves* the total value of the allocation based on the reports of other agents and the agent's own value.

The main innovation in this paper is to provide a fast method to integrate the approximation algorithm for winner-determination into the calculation of VCG payments. A straightforward scheme would require an asymptotic time $O(nT)$ to compute payments to all n bidders. Our scheme determines approximate VCG payments in worst-case time $O(\alpha T \log(\alpha n/\varepsilon))$, where α is a constant that quantifies a reasonable "no-monopoly" assumption. Specifically, in the reverse auction, suppose that $C(\mathcal{I})$ is the minimal cost for procuring M units with all suppliers $\mathcal{I}$, and $C(\mathcal{I} \setminus i)$ is the minimal cost without supplier i. Then, the constant α is defined as an upper bound for the ratio $C(\mathcal{I} \setminus i)/C(\mathcal{I})$, over all suppliers i. This upper-bound tends to 1 as the number of suppliers increases.

Section 2 formally defines the forward and reverse auctions, and defines the VCG mechanisms. We also prove our claims about ε-strategyproofness. Section 3 provides the generalized knapsack formulation for the multi-unit allocation problems and introduces the fully polynomial time approximation scheme. Section 4 defines the approximation scheme for the payments in the VCG mechanism. Section 5 concludes.

1.1 Related Work

There has been considerable interest in recent years in characterizing polynomial-time or approximable special cases of the general combinatorial allocation problem, in which there are multiple different items. The combinatorial allocation problem (CAP) is both NP-complete [29] and inapproximable ([30]). Although some polynomial-time cases have been identified for the CAP (e.g. [7]), introducing an expressive *exclusive-or* bidding language quickly breaks these special cases. In this work we identify a non-trivial but approximable allocation problem with an expressive exclusive-or bidding language—the bid taker in our setting is allowed to accept at most one point on the bid curve.

The idea of using approximations within the allocation rules of mechanisms, while retaining either full-strategyproofness or ε-dominance has received some previous attention. For instance, Lehmann *et al.* [23] propose a greedy and strategyproof approximation to a single-minded combinatorial auction problem. Nisan & Ronen [25] discussed approximate VCG-based mechanisms, but either appealed to particular *maximal-in-range* approximations to retain *full* strategyproofness, or to resource-bounded agents with information or computational limitations on the ability to compute strategies.

Feigenbaum & Shenker [10] have defined the concept of *strategically faithful* approximations, and proposed the study of approximations as an important direction for algorithmic mechanism design. Schummer [31] and Parkes *et al.* [28] have previously considered ε-dominance, in the context of economic impossibility results,

for example in combinatorial exchanges. Archer *et al.* [1] adopt similar notions of approximate-strategyproofness in their work on single-minded combinatorial auctions.

A recent characterization due to Lavi *et al.* [21] suggests that it is in fact *necessary* to relax full strategyproofness and consider approximate-strategyproofness in our multi-unit problem: their analysis suggests that no worst-case polynomial time algorithm can be strategyproof and have good approximation properties for the multi-unit allocation problem.

Eso *et al.* [9] have studied a similar procurement problem, but for a different volume discount model. This earlier work formulates the problem as a general mixed integer linear program, and gives some empirical results on simulated data. Kalagnanam *et al.* [18] address double auctions, where multiple buyers and sellers submit demand and supply curves for a *divisible good*, and investigate *competitive-equilibrium* outcomes for myopically-rational agents.

Ausubel [2] has proposed an *ascending-price* multi-unit auction for buyers with marginal-decreasing values and no lower-bounds on lot size, with an interpretation as a primal-dual algorithm [3]. Iwasaki *et al.* [17] generalize Ausubel's methods to allow general valuations in this multi-unit auction problem, and thus their auction applies to the models in the current paper. However, the primary focus in Iwasaki *et al.* is on providing robustness to false-name bids [34] in which bidders participate under multiple identities. Their auction runs in pseudo-polynomial time, but does not provide any worst-case approximation guarantees.

2 Approximately-Strategyproof VCG Auctions

In this section, we first describe the marginal-decreasing piecewise bidding language that is used in our forward and reverse (procurement) auctions. Continuing, we introduce the VCG mechanism for the problem and the ε-dominance results for approximations to VCG outcomes. We also discuss the economic properties of VCG mechanisms in these forward and reverse auction multi-unit settings.

2.1 Marginal-Decreasing Piecewise Bids

We provide a piecewise-constant and marginal-decreasing bidding language. This bidding language is expressive for a natural class of valuation and cost functions: fixed unit prices over intervals of quantities. See Figure 1 for an example. We relax the marginal-decreasing requirement to allow a bidder in the forward auction to state a *minimal purchase amount*, to reflect a zero value for quantities smaller

than that amount. Similarly, a supplier in the reverse auction can state a *capacity constraint* to reflect an (effectively) infinite cost to supply quantities in excess of a particular amount.

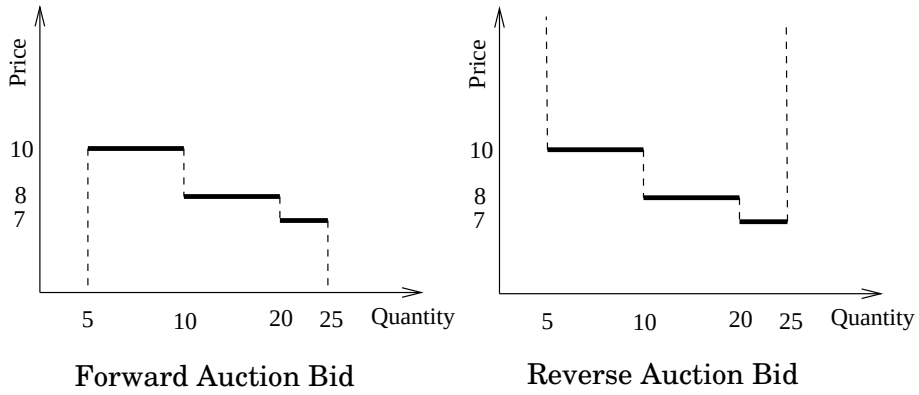

Fig. 1. Marginal-decreasing, piecewise constant bids. In the forward auction bid, the bidder offers \$10 per unit for quantity in the range $[5, 10)$, \$8 per unit in the range $[10, 20)$, and \$7 in the range $[20, 25]$. Her valuation is zero for quantities outside the range $[10, 25]$. In the reverse auction bid, the cost of the supplier is ∞ outside the range $[10, 25]$.

In detail, in a forward auction, a bid from buyer i can be written as a list of (quantity-range, unit-price) tuples, $((u_i^1, p_i^1), (u_i^2, p_i^2), \ldots, (u_i^{m_i-1}, p_i^{m_i-1}))$, with an upper bound $u_i^{m_i}$ on the quantity. The interpretation is that the bidder's valuation in the (semi-open) quantity range $[u_i^j, u_i^{j+1})$ is p_i^j for each unit. Additionally, it is assumed that the valuation is 0 for quantities less than u_i^1 as well as for quantities more than u_i^m. This is implemented by adding two dummy bid tuples, with zero prices in the range $[0, u_i^1)$ and $(u_i^{m_i}, \infty)$. We interpret the bid list as defining a price function, $p_{\text{bid},i}(q) = qp_i^j$, if $u_i^j \leq q < u_i^{j+1}$, where $j = 1, 2, \ldots, m_i - 1$. In order to resolve the boundary condition, we assume that the bid price for the upper bound quantity $u_i^{m_i}$ is $p_{\text{bid},i}(u_i^{m_i}) = u_i^{m_i} p_i^{m_i-1}$.

A supplier's bid is similarly defined in the reverse auction. The interpretation is that the bidder's cost in the (semi-open) quantity range $[u_i^j, u_i^{j+1})$ is p_i^j for each unit. Additionally, it is assumed that the cost is ∞ for quantities less than u_i^1 as well as for quantities more than u_i^m. Equivalently, the unit prices in the ranges $[0, u_i^1)$ and (u_{ij}^m, ∞) are infinity. We interpret the bid list as defining a price function, $p_{\text{ask},i}(q) = qp_i^j$, if $u_i^j \leq q < u_i^{j+1}$.

We assume quasilinear utility functions, with $u_i(q, p) = v_i(q) - p$, for a buyer i with valuation $v_i(q)$ for q units at price p, and $u_i(q, p) = p - c_i(q)$ for a supplier i with cost $c_i(q)$ at price p. This is a standard assumption in the auction literature, and equivalent to assuming risk-neutral agents [19]. We will use the term *payoff* interchangeably for *utility*.

2.2 VCG-Based Multi-Unit Auctions

We construct the tractable and approximately-strategyproof multi-unit auctions around a standard VCG mechanism. In the forward auction, we model a seller with M units and no intrinsic value for the items. Given a set of bids from $\mathcal{I}$ agents, let $V(\mathcal{I})$ denote the maximal value given bids, and subject to the condition that at most one point on the bid curve can be selected from each agent and no more than M units of the item can be sold. Let $x^* = (x_1^*, \ldots, x_N^*)$ denote the solution to this winner-determination problem, where x_i^* is the number of units sold to agent i. Let $V(\mathcal{I} \setminus i)$ denote the maximal value to the seller without bids from agent i. The VCG mechanism for this problem has the following steps:

(1) Collect piecewise-constant bid curves and capacity constraints from all the buyers.
(2) Implement the outcome, x^*, that solves the winner-determination problem.
(3) Collect payment $p_{\text{vcg},i} = p_{\text{bid},i}(x_i^*) - [V(\mathcal{I}) - V(\mathcal{I} \setminus i)]$ from each buyer, and pass the payments to the seller.

In this forward auction, the VCG mechanism is *strategyproof* for buyers, which means that truthful bidding is a dominant strategy and utility maximizing whatever the bids of other buyers. In addition, the VCG mechanism is *allocatively-efficient*, and the total payments maximize the revenue to the seller in expectation across all efficient auctions, even allowing for Bayesian-Nash implementations [20]. Each buyer pays less than its value, and receives payoff $V(\mathcal{I}) - V(\mathcal{I} \setminus i)$ in equilibrium. This is precisely the marginal-value that buyer i contributes to the economy.

In the reverse (or procurement) auction we model a buyer with value $V > 0$ to purchase at least M units and no value otherwise and n suppliers. Each supplier has a marginal-decreasing cost function, subject to a capacity constraint. To simplify the mechanism design problem we assume a straightforward buyer that will truthfully announce this value to the mechanism. [4] The VCG mechanism remains strategyproof for suppliers without this assumption, but the efficiency of the outcome can be compromised.

The winner-determination problem is to determine the allocation, x^*, that *minimizes* the cost to the buyer, and forfeit trade if this minimal cost is greater than value, V. Let $C(\mathcal{I})$ denote the minimal cost given bids from all suppliers, and let $C(\mathcal{I} \setminus i)$ denote the minimal cost without bids from supplier i. When there are no gains from trade the outcome of the VCG mechanism is no trade and no payments. Assume otherwise, with $V \geq C(\mathcal{I})$. The VCG mechanism implements the outcome x^* that minimizes cost based on bids from all suppliers, and then provides payment $p_{\text{vcg},i} = p_{\text{ask},i}(x_i^*) + [V - C(\mathcal{I}) - \max(0, V - C(\mathcal{I} \setminus i))]$ to each supplier. The

[4] Without this assumption, the Myerson-Satterthwaite [24] impossibility result would already imply that we should not expect an efficient trading mechanism in this setting.

total payment is collected from the buyer. The payoff to each supplier is equal to the marginal value contributed to the system, and precisely $C(\mathcal{I} \setminus i) - C(\mathcal{I})$ when $V \geq C(\mathcal{I} \setminus i)$ for all suppliers.

The total payments to suppliers in the reverse VCG auction can be greater than the buyer's value for the goods. This is not a problem with the VCG auction *per se* but rather a problem with the efficient multi-unit allocation problem in this reverse direction. The single-item, one buyer and one seller, bargaining problem is a special case of this problem, and a setting in which the well known Myerson-Satterthwaite [24] impossibility result holds. It is not possible to construct an efficient mechanism for the bargaining problem, that satisfies participation (i.e. with the buyer paying less than her value), without sometimes running at a deficit. In fact, the VCG mechanism maximizes the expected revenue to the buyer across all efficient auctions [20] and there can be no efficient auction in which the buyer always pays less than her value when the VCG mechanism fails to satisfy this property.

Formally, the total payment collected by suppliers in the VCG mechanism in equilibrium is less than the buyer's value for the items if and only if the following condition holds:

$$V - C(\mathcal{I}) \geq \sum_i [V - C(\mathcal{I}) - \max\{0, V - C(\mathcal{I} \setminus i)\}] \tag{1}$$

This states that the total payoff to the suppliers is no greater than the total payoff from the efficient allocation. In particular, we need that there are no "pivotal" suppliers for which there is no efficient trade without the supplier (i.e. $V - C(\mathcal{I}) > 0$ but $V - C(\mathcal{I} \setminus i') < 0$. A pivotal supplier's payment provides her with all the reported surplus, which is the outcome that she could achieve in some equilibrium of the underyling bargaining problem. In the absence of pivotal suppliers, condition (1) simplifies to:

$$V - C(\mathcal{I}) \;\; \geq \;\; \sum_i (C(\mathcal{I} \setminus i) - C(\mathcal{I})) \tag{2}$$

This condition states that marginal value contributed by each supplier to the economy, summed across all suppliers, must be less than the marginal value provided by the suppliers when they act as a single coalition. [5] In this case the buyer's payment will be no greater than her value for the outcome and the efficient trade can be implemented.

Consider an example with 3 agents $\{1, 2, 3\}$, and $V = 150$ and $C(123) = 50$. Condition (2) holds when $C(12) = C(23) = 70$ and $C(13) = 100$, but not when $C(12) = C(23) = 80$ and $C(13) = 100$. In the first case, the agent payoffs

[5] This condition is implied by *agents are substitutes* [4], which is necessary and sufficient to support VCG payments in the core in a combinatorial allocation problem.

$\pi = (\pi_0, \pi_1, \pi_2, \pi_3)$, where 0 denotes the buyer, are $(10, 20, 50, 20)$. In the second case, the payoffs are $\pi = (-10, 30, 50, 30)$ and the buyer's payment is \$10 greater than her value. Pragmatically, the buyer needs an opt-out to cancel the auction in this second case and avoid purchasing the items at a loss. However, this would affect strategyproofness on the sell-side. For instance, if the auction was never to be repeated than seller 3 would prefer to understate her cost for supplying goods in combination with seller 1 so that $C(13) = 80$. The effect would be to adjust payoffs to $\pi = (10, 30, 30, 30)$ (with supplier 2 receiving a smaller payment).

Thus the consequence of this asymmetry between the reverse and forward auctions is that the reverse (procurement) auction is only applicable in settings in which supplier costs and the buyer's value is such that condition (2) is sure to hold.

2.3 ε-Strategyproofness

We now consider the strategic consequences of introducing an approximation schemes into the VCG mechanism. It is well known that any approximation that does not remain optimal on some fixed range of outcomes must lead to a failure of full strategyproofness [25]. However, we derive a simple ε-strategyproofness result, that bounds the maximal gain in payoff that an agent can expect to achieve through a unilateral deviation from following a simple truth-revealing strategy. We describe the result for the forward auction problem, but it is quite a general observation.

Let $\hat{V}(\mathcal{I})$ and $\hat{V}(\mathcal{I} \setminus i)$ denote the value of the allocation computed with an approximation scheme, and assume that the approximation satisfies:

$$(1 + \epsilon)\hat{V}(\mathcal{I}) \geq V(\mathcal{I})$$

for some $\epsilon > 0$. We provide such an approximation scheme for our setting later in the paper. Let $\hat{x}$ denote the allocation implemented by the approximation scheme, and consider the VCG mechanism with this approximation. The payoff to agent i, for announcing valuation $\hat{v}_i$, is:

$$v_i(\hat{x}_i) + \sum_{j \neq i} \hat{v}_j(\hat{x}_j) - \hat{V}(\mathcal{I} \setminus i) \tag{3}$$

The final term is independent of the agent's announced value, and can be ignored in an incentive analysis. However, agent i can try to improve its payoff through the effect of its announced value on the allocation $\hat{x}$ implemented by the mechanism. In particular, agent i wants the mechanism to select $\hat{x}$ to maximize the sum of its *true* value, $v_i(\hat{x}_i)$, and the reported value of the other agents, $\sum_{j \neq i} \hat{v}_j(\hat{x}_j)$. If the mechanism's allocation algorithm is optimal, then all the agent needs to do is truthfully state its value and the mechanism will do the rest. However, faced with

an approximate allocation algorithm and the reports of other agents, the agent can try to improve its payoff by announcing a value that *corrects* for the approximation.

Let $V(v_i, \hat{v}_{-i})$ denote the total value of the *efficient* allocation given the *reported* values $\hat{v}_{-i}$ of agents $j \neq i$, and given the true value of agent i. We say a mechanism is ϵ-strategyproof if an agent can gain at most ϵ through some non-truthful strategy.

Theorem 1 *A VCG-based mechanism with a $(1+\varepsilon)$-allocation scheme is $(\frac{\epsilon}{1+\epsilon})V(v_i, \hat{v}_{-i})$-strategyproof for agent i given bids $\hat{v}_{-i}$ from other agents.*

PROOF. Recall from Eq. (3) that the agent's payoff, given outcome $\hat{x}$, is $v_i(\hat{x}_i) + \sum_{j \neq i} \hat{v}_j(\hat{x}_j) - \hat{V}(\mathcal{I} \setminus i)$. Thus the maximal benefit to agent i from reporting a non-truthful $\hat{v}_i \neq v_i$ occurs when the initial approximation is as bad as possible given the approximation bounds. This occurs when the value from solution $\hat{x}$ is $V(v_i, \hat{v}_{-i})/(1+\varepsilon)$. In this case, agent i can hope to report $\hat{v}_i$ that will cause the winner-determination algorithm to select outcome x' that maximizes $v_i(x'_i) + \sum_{j \neq i} \hat{v}_j(x'_j)$ and achieves total value $V(v_i, \hat{v}_{-i})$. The agent's gain in utility in this case, in comparison with truthful bidding, is

$$V(v_i, \hat{v}_{-i}) - \frac{V(v_i, \hat{v}_{-i})}{1+\varepsilon} \;=\; \frac{\varepsilon}{1+\varepsilon} V(v_i, \hat{v}_{-i})$$

□

Formally, we interpret approximate strategyproofness as a statement that truth-revelation is a dominant strategy equilibrium for an agent that is indifferent between payments that are within $(\epsilon/1+\epsilon)V_{\max}$ where $V_{\max}$ is the *maximal* value from trade across all possible economies.

Note that although we do not need to bound the accuracy of the estimated optimal value without agent i to demonstrate approximate strategyproofness, this bound is required to provide a good approximation to the revenue properties of the VCG mechanism.

3 The Generalized Knapsack Problem

In this section, we design a fully polynomial approximation scheme for the generalized knapsack, which models the winner-determination in the multi-unit allocation problem. We describe our results for the reverse auction variation, but the formulation is completely symmetric for the forward-auction.

In describing our approximation scheme, we begin with a simple property (the *Anchor property*) of an optimal knapsack solution. We use this property to reduce

our problem to a simpler but restricted problem. We present a 2-approxmation for the restricted problem and then use this basic approximation to develop a fully polynomial-time approximation scehme for this restricted problem (FPTAS). Latter we use the FPTAS for the restricted problem to develop an FPTAS for the generalized knapsack problem.

One of the major appeals of our piecewise bidding language is its *compact representation* of the bidder's valuation functions. We strive to preserve this, and present an approximation scheme that will depend only on the number of bidders, and not the maximum quantity, M, demanded by the buyer, which can be very large in realistic procurement settings.

The FPTAS implements an $(1+\varepsilon)$ approximation to the optimal solution x^*, in worst-case time $T = O(n^3/\varepsilon)$, where n is the number of bidders, and where we assume that the piecewise bid for each bidder has $O(1)$ pieces. The dependence on the number of pieces is also polynomial: if each bid has a maximum of c pieces, then the running time can be derived by substituting nc for each occurrence of n.

3.1 Preliminaries

Before we begin, let us recall the classic 0/1 knapsack problem: we are given a set of n items, where the item i has *value* v_i and *size* s_i, and a knapsack of capacity M; all sizes are integers. The goal is to determine a subset of items of maximum value with total size at most M. Since we want to focus on a reverse auction, the equivalent knapsack problem will be to choose a set of items with *minimum value* (i.e. cost) whose size *exceeds* M. The *generalized knapsack problem* of interest to us can be defined as follows:

Generalized Knapsack:
Instance: A target M, and a set of n lists, where the ith list has the form

$$B_i \;=\; \langle (u_i^1, p_i^1), \ldots, (u_i^{m_i-1}, p_i^{m_i-1}), (u_i^{m_i}(i), \infty) \rangle,$$

where u_i^j are increasing with j and p_i^j are decreasing with j, and u_i^j, p_i^j, M are positive integers.

Problem: Determine a set of integers x_i^j such that
(1) (One per list) At most one x_i^j is non-zero for any i,
(2) (Membership) $x_i^j \neq 0$ implies $x_i^j \in [u_i^j,\ u_i^{j+1})$,
(3) (Target) $\sum_i \sum_j x_i^j \;\geq\; M$, and
(4) (Objective) $\sum_i \sum_j p_i^j x_i^j$ is minimized.

This generalized knapsack formulation is a clear generalization of the classic 0/1 knapsack. In the latter, each list consists of a single *point* (s_i, v_i).

The connection between the generalized knapsack and our auction problem is transparent. Each list encodes a bid, representing multiple *mutually exclusive* quantity intervals, and one can choose any quantity in an interval, but at most one interval can be selected. Choosing interval $[u_i^j, u_i^{j+1})$ has cost p_i^j per unit. The goal is to procure at least M units of the good at minimum possible cost. The generalized knapsack problem has some flavor of the *continuous* knapsack problem. However, there are two major differences that make our problem significantly more difficult: (1) intervals have boundaries, and so to choose interval $[u_i^j, u_i^{j+1})$ requires that at least u_i^j and at most u_i^{j+1} units must be taken; (2) unlike the classic knapsack, we cannot sort the items (bids) by value/size, since different intervals in one list have different unit costs.

In fact, because of the "one per list" constraint, the generalized problem is closer in spirit to the *multiple choice knapsack* problem [11], where the underling set of items is partitioned into disjoint subsets $U_1, U_2, \ldots, U_k$, and one can choose at most one item from each subset. PTAS do exist for the multiple choice knapsack problem [12], and indeed, one can convert our problem into a *huge* instance of this problem, by creating one group for each list; put a (quantity, price) point tuple (x, p) for *each possible quantity* for a bidder into his group (subset). However, this conversion explodes the problem size, making it infeasible for all but the most trivial instances.

3.2 Anchor Property

We begin with a definition. Given an instance of the generalized knapsack, we call each tuple $t_i^j = (u_i^j, p_i^j)$ an *anchor*. Recall that these tuples represent the breakpoints in the piecewise constant curve bids. We say that the *size* of an anchor t_i^j is u_i^j, the minimum number of units available at this anchor's price p_i^j. The *cost* of the anchor t_i^j is defined to be the minimum total price associated with this tuple, namely, $cost(t_i^j) = p_i^j u_i^j$ if $j < m_i$, and $cost(t_i^{m_i}) = p_i^{m_i-1} u_i^{m_i}$.

In a feasible solution $\{x_1, x_2, \ldots, x_n\}$ of the generalized knapsack, we say that an element $x_i \neq 0$ is an anchor if $x_i = u_i^j$, for some anchor u_i^j. Otherwise, we say that x_i is *midrange*. We observe that an optimal knapsack solution can always be constructed so that at most one solution element is midrange. If there are two midrange elements x and x', for bids from two different agents, with $x \leq x'$, then we can increment x' and decrement x, until one of them becomes an anchor. See Figure 2 for an example.

Lemma 1 *[Anchor Property] There exists an optimal solution of the generalized knapsack problem with at most one midrange element. All other elements are anchors.*

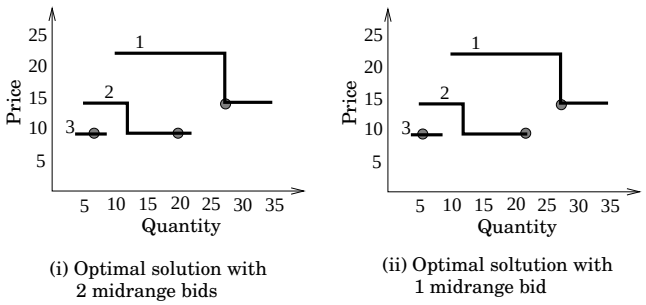

Fig. 2. (i) An optimal solution with more than one bid not anchored (2,3); (ii) an optimal solution with only one bid (3) not anchored.

3.3 Algorithm Roadmap

First we consider a restricted generalized knapsack problem, where we assume that the midrange element corresponding to the optimal solution is known. By anchor property, we know that any other element, x_i, is going to be one of the anchor point in the list i of the generalized knapsack problem. Therefore, the restricted problem is very similar to the multiple choice knapsack problem. We use the approximation scheme for multiple choice knapsack problem to develop an approximation scheme for the restricted problem. The approximation scheme for the restricted problem, in turn, is used to develop an approximation scheme for the generalized knapsack problem. This is done by iterating over all the choices for midrange element, solving the corresponding restricted problem and choosing the solution with the minimum cost as the final solution. Following pseudo-code presents the high level idea. In the pseudo code, the tuple (ℓ, j) represents a midrange element, where x_ℓ lies in $[u_\ell^j, u_\ell^{j+1})$.

- for ℓ = 1 to n do
- for j in list ℓ of generalized knapsack do
- Solve the problem assuming (ℓ, j) as midrange element. Let $A(\ell, j)$ be the solution.
- Return $\min_{(\ell,j)}\{A(\ell, j)\}$.

3.4 2-approximation

Let us assume that the midrange element corresponding to the optimal solution is known. Suppose that, in the optimal solution, x_ℓ is midrange element and agent ℓ lies in its jth range, $[u_\ell^j, u_\ell^{j+1})$. Now we can reduce the generalized knapsack problem to a simpler problem where x_ℓ has to lie between $[u_\ell^j, u_\ell^{j+1})$ and any other x_i has to be either zero or one of the anchor points in list i. The objective of the new problem is same as the original problem, that is, to obtain M units at the minimum cost. It is worth noting that an optimal solution for this restricted problem is also

going to be an optimal solution for the generalized knapsack problem.

The new problem contains n groups. There are $n-1$ groups of potential anchors, where ith group contains all the anchors of the list i in the generalized knapsack. The group for agent ℓ contains two elements. The first element, t_ℓ^1, is the anchor point (u_ℓ^j, p_ℓ^j). The second element, t_ℓ^2, represents the *interval* $[0, u_\ell^{j+1} - u_\ell^j)$, and the associated unit-price p_ℓ^j. Since x_ℓ lies between u_ℓ^j and u_ℓ^{j+1}, any solution should choose t_ℓ^1 to ensure that x_ℓ is at least u_ℓ^j. The second element, t_ℓ^2, represents the excess number of units that can be taken from agent ℓ in addition to u_ℓ^j, which has already been committed. In any other group, we can choose at most one anchor.

The following pseudo-code describes our algorithm for this restricted generalized knapsack problem. U is the union of all the tuples in n groups, including tuple t_ℓ^2 for agent ℓ. The *size* of this special tuple is defined as $u_\ell^{j+1} - u_\ell^j$, and the *cost* is defined as $p_l^j(u_\ell^{j+1} - u_\ell^j)$. R is the number of units that remain to be acquired. S is the set of tuples accepted in the current tentative solution. Since any solution should contain t_ℓ^1, we add it to S and initialize R to be $M - u_\ell^j$. $Best$ is the best solution found so far. Variable $Skip$ is only used in the proof of correctness.

Algorithm Greedy

(1) Sort all tuples of U in the ascending order of unit price; in case of ties, sort in ascending order of unit quantities.
(2) Set $mark(i) = 0$, for all lists $i = 1, 2, \ldots, n$.
Initialize $R = M - u_\ell^j$, $S = \{t_\ell^1\}$, $Best = Skip = \emptyset$.
(3) Scan the tuples in U in the sorted order. Suppose the next tuple is t_i^k, i.e. the kth anchor from agent i.
If $mark(i) = 1$, ignore this tuple;
otherwise do the following steps:
- if $size(t_i^k) > R$ and $i = \ell$
  return min $\{cost(S) + Rp_\ell^j,\ cost(Best)\}$;
- if $size(t_i^k) > R$ and $cost(t_i^k) \leq cost(S)$
  return min $\{cost(S) + cost(t_i^k),\ cost(Best)\}$;
- if $size(t_i^k) > R$ and $cost(t_i^k) > cost(S)$
  add t_i^k to $Skip$; Set $Best$ to $S \cup \{t_i^k\}$ if cost improves;
- if $size(t_i^k) \leq R$ then
  add t_i^k to S; $mark(i) = 1$; subtract $size(t_i^k)$ from R.

The approximation algorithm is very similar to the approximation algorithm for knapsack [16]. Since we wish to minimize the total cost, we consider the tuples in order of increasing per unit cost. If the size of tuple t_i^k is smaller than R, then we add it to S, update R, and delete from U all the tuples that belong to the same group as t_i^k. If $size(t_i^k)$ is greater than R, then S along with t_i^k forms a feasible solution. However, this solution can be far from optimal if the size of t_i^k is much larger than

R. If total cost of S and t_i^k is smaller than the current best solution, we update $Best$. One exception to this rule is the tuple t_ℓ^2. Since this tuple can be taken fractionally, we update $Best$ if the sum of S's cost and fractional cost of t_ℓ^2 is an improvement.

The algorithm terminates in either of the first two cases, or when all tuples are scanned. In particular, it terminates whenever we find a t_i^k such that $size(t_i^k)$ is greater than R but $cost(t_i^k)$ is less than $cost(S)$, or when we reach the tuple t_ℓ^2 and it gives a feasible solution.

Lemma 2 *Suppose A^* is an optimal solution of the generalized knapsack with element (l, j) being midrange. Then, the solution A, returned by Greedy is a 2-approximation to A^*.*

PROOF. Let A be the value returned by Greedy. Consider the set $Skip$ at the termination of Greedy. There are two cases to consider: either some tuple $t \in Skip$ is also in A^*, or no tuple in $Skip$ is in A^*. In the first case, let S_t be the tentative solution S at the time t was added to $Skip$. Because $t \in Skip$ then $size(t) > R$, and S_t together with t forms a feasible solution, and we have:

$$cost(A) \;\leq\; cost(Best) \;\leq\; cost(S_t) + cost(t).$$

Again, because $t \in Skip$ then $cost(t) > cost(S_t)$, and we have $cost(A) < 2cost(t)$. On the other hand, since t is included in A^*, we have $A^* \geq cost(t)$. These two inequalities imply the desired bound:

$$cost(A^*) \;\leq\; cost(A) \;<\; 2cost(A^*).$$

In the second case, imagine a modified instance of the problem, which *excludes* all the tuples of the set $Skip$. Since none of these tuples were included in A^*, the optimal solution for the modified problem should be same as one for the original. Suppose our approximation algorithm returns the value A' for this modified instance.

Let t' be the last tuple considered by the approximation algorithm before termination of the modified instance, and let $S_{t'}$ be the corresponding tentative solution set in that step. Running the greedy algorithm for the modified instance can be thought as running it for the original problem except instead of adding a tuple to $Skip$, we ignore it. Therefore, even for the original problem, t' is going to be the last tuple considered and $S_{t'}$ is going to be corresponing tentative solution. Since the $Skip$ of the modified problem remains empty, the $Best$ for the modified instance remains NULL. Therefore, $cost(A')$ is the sum of the cost of $S_{t'}$ and t'. On the other hand, $cost(A)$ is the minimum of the same sum and $Best$ for the original problem. Therefore,

$$cost(A) \leq cost(A')$$

Also, for the modified instance, as we consider tuples in order of increasing per unit price, and none of the tuples are going to be placed in the set $Skip$, we must have $cost(S_{t'}) < cost(A^*)$ because $S_{t'}$ is the optimal way to obtain $size(S_{t'})$ in the modified problem.

We also have $cost(t) \leq cost(S_{t'})$. Therefore,

$$\begin{aligned} cost(A) \leq cost(A') &\leq cost(S_{t'}) + cost(t') \\ &< 2cost(A^*) \end{aligned}$$

This completes the proof of Lemma 2. □

So far, we have considered the restricted problem where we assume that the midrange element corresponding to the optimal solution is known. We convert the 2 approximation algorithm for the restricted problem to one for generalized knapsack problem by using the idea presented in the section 3.3. We run the algorithm Greedy once for each tuple (ℓ, j) as a candidate for midrange element and choose the solution with the minimum cost as our final solution.

There are $O(n)$ tuples and we execute the algorithm Greedy for each of them. It is easy to see that, after an initial sorting of the tuples, the algorithm takes $O(n)$ time. Also the sorting needs to be done only once and hence the total cost of the algorithm is $O(n^2)$. Thus, we have our first polynomial time approximation algorithm.

Theorem 2 *A 2-approximation of the generalized knapsack problem can be found in time $O(n^2)$, where n is number of item lists (each of constant length).*

The dependence on the number of pieces is also polynomial: if each bid has a maximum of c pieces, then the running time is $O((nc)^2)$.

3.5 An Approximation Scheme

We now use the 2-approximation algorithm presented in the preceding section to develop a fully polynomial approximation (FPTAS) for the generalized knapsack problem. The high level idea is fairly standard, but the details require technical care. Similar to the 2-approximation algorithm, we first consider the restricted problem where the midrange element is known.

Let the midrange element be x_ℓ, which falls in the range $[u_\ell^j, u_\ell^{j+1})$. Our FPTAS runs in two phases. In the first phase, we solve a multiple choice knapsack prob-

lem, *mKnapsack*, where we construct a dynamic programming table to compute the minimum cost at which at least $M - u_\ell^{j+1}$ units can be obtained using the remaining $n-1$ lists in the generalized knapsack. In the second phase, we go through the last row of the dynamic programming table, searching for the entry, which along with the midrange element minimizes the cost of obtaining at least M units.

Suppose $G[i,r]$ denotes the *maximum* number of units that can be obtained at cost at most r using only the first i lists (ignoring the ℓth list) in the generalized knapsack. Then, the following recurrence relation describes how to construct the dynamic programming table:

$$G[0,r] = 0$$

$$G[i,r] = \max \left\{ \begin{array}{l} G[i-1,r] \\ \max\limits_{j \in \beta(i,r)} \{G[i-1,\ r - cost(t_i^j)] + u_i^j\} \end{array} \right\}$$

where $\beta(i,r) = \{j \ : \ 1 \leq j \leq m_i,\ cost(t_i^j) \leq r\}$, is the set of anchors for agent i. As convention, agent i will index the *row*, and cost r will index the *column*.

This dynamic programming algorithm is only pseudo-polynomial, since the number of column in the dynamic programming table depends upon the total cost. However, we can convert it into a FPTAS by *scaling* the cost dimension.

Let A denote the 2-approximation to the generalized knapsack problem, with total cost, $cost(A)$. Let ε denote the desired approximation factor. We compute the *scaled cost* of a tuple t_i^j, denoted $scost(t_i^j)$, as

$$scost(t_i^j) \ = \ \lceil \frac{n \ cost(t_i^j)}{\varepsilon cost(A)} \rceil \tag{4}$$

This scaling improves the running time of the algorithm because the number of columns in the modified table is at most $\lceil \frac{n}{\varepsilon} \rceil$, and independent of the total cost. However, the computed solution might not be an optimal solution for the original problem. We show that the error introduced is within a factor of ε of the optimal solution.

As a prelude to our approximation guarantee, we first show that if two different solutions to the *mKnapsack* problem have equal scaled cost, then their *original* (unscaled) costs cannot differ by more than $\varepsilon cost(A)$.

Lemma 3 *Let x and y be two distinct feasible solutions of mKnapsack. If x and y have equal scaled costs, then their unscaled costs cannot differ by more than $\varepsilon cost(A)$.*

PROOF. Let I_x and I_y, respectively, denote the indicator functions associated with the anchor vectors x and y—there is 1 in position $I_x[i,k]$ if the $x_i^k > 0$. Since x and y has equal scaled cost,

$$\sum_{i \neq \ell} \sum_k scost(t_i^k) I_x[i,k] \;=\; \sum_{i \neq \ell} \sum_k scost(t_i^k) I_y[i,k] \tag{5}$$

However, by (4), the scaled costs satisfy the following inequalities:

$$\frac{(scost(t_i^k) - 1)\varepsilon\, cost(A)}{n} \;\leq\; cost(t_i^k) \;\leq\; \frac{scost(t_i^k)\varepsilon\, cost(A)}{n} \tag{6}$$

Substituting the upper-bound on scaled cost from (6) for $cost(x)$, the lower-bound on scaled cost from (6) for $cost(y)$, and using equality (5) to simplify, we have:

$$cost(x) - cost(y) \;\leq\; \frac{\varepsilon\, cost(A)}{n} \sum_{i \neq \ell} \sum_k I_y[i,k] \;\leq\; \varepsilon\, cost(A),$$

The last inequality uses the fact that at most n components of an indicator vector are non-zero; that is, any feasible solution contains at most n tuples. □

Finally, given the dynamic programming table for *mKnapsack*, we consider all the entries in the last row of this table, $G[n-1, r]$. These entries correspond to optimal solutions with all agents except ℓ, for different levels of cost. In particular, we consider the entries that provide at least $M - u_\ell^{j+1}$ units. Together with a contribution from agent ℓ, we choose the entry in this set that *minimizes* the total cost, defined as follows:

$$cost(G[n-1,r]) \;+\; \max\; \{u_\ell^j,\; M - G[n-1,r]\} p_\ell^j,$$

where $cost()$ is the original, unscaled cost associated with entry $G[n-1, r]$.

The following lemma shows that we achieve a $(1+\varepsilon)$-approximation.

Lemma 4 *Suppose A^* is an optimal solution of the generalized knapsack problem, and suppose that element (l, j) is midrange in the optimal solution. Then, the solution V from running our FPTAS satisfies*

$$cost(V) \leq (1 + 2\varepsilon)\, cost(A^*)$$

PROOF. Let $x_{-\ell}$ denote the vector of the elements in solution A^* without element ℓ. Then, by definition, $cost(A^*) \;=\; cost(x_{-\ell}) + p_\ell^j x_\ell^j$. Let $r \;=\; scost(x_{-\ell})$ be the

scaled cost associated with the vector $x_{-\ell}$. Now consider the dynamic programming table constructed for *mKnapack*, and consider its entry $G[n-1, r]$. Let A denote the 2-approximation to the generalized knapsack problem, and V denote the solution from our FPTAS.

Suppose $y_{-\ell}$ is the solution associated with this entry in our dynamic program; the components of the vector $y_{-\ell}$ are the quantities from different lists. Since both $x_{-\ell}$ and $y_{-\ell}$ have equal scaled costs, by Lemma 3, their unscaled costs are within $\varepsilon\, cost(A)$ of each other; that is,

$$cost(y_{-\ell}) - cost(x_{-\ell}) \;\leq\; \varepsilon\, cost(A).$$

Now, define $y_\ell^j \;=\; \max\{u_\ell^j,\; M - \sum_{i\neq\ell}\sum_j y_i^j\}$; this is the contribution needed from ℓ to make $(y_{-\ell}, y_\ell^j)$ a feasible solution. Among all the equal cost solutions, our dynamic programming tables chooses the one with maximum units. Therefore,

$$\sum_{i\neq\ell}\sum_j y_i^j \;\geq\; \sum_{i\neq\ell}\sum_j x_i^j$$

Therefore, it must be the case that $y_\ell^j \;\leq\; x_\ell^j$. Because $(y_\ell^j,\; y_{-\ell})$ is also a feasible solution, if our algorithm returns a solution with cost $cost(V)$, then we must have

$$\begin{aligned} cost(V) &\leq cost(y_{-\ell}) \;+\; p_\ell^j\, y_\ell^j \\ &\leq cost(x_{-\ell}) \;+\; \varepsilon cost(A) \;+\; p_\ell^j\, x_\ell^j \\ &\leq (1+2\varepsilon)\, cost(A^*), \end{aligned}$$

where we use the fact that $cost(A) \leq 2\, cost(A^*)$. $\square$

The presented scheme assumes that the midrange element is known. Using the ideas presented in section 3.3, we convert the approximation scheme for restricted problem to one for the generalized knapsack problem. The approximation scheme for generalized knapsack problem solves $O(n)$ restricted problems, corresponding to $O(n)$ choices of midrange element, and chooses one with the minimum cost as the final solution.

For a given midrange, the most expensive step in the algorithm is the construction of dynamic programming table, which can be done in $O(n^2/\varepsilon)$ time assuming constant intervals per list. Thus, we have the following result.

Theorem 3 *We can compute an $(1+\varepsilon)$ approximation to the solution of a generalized knapsack problem in worst-case time $O(n^3/\varepsilon)$.*

The dependence on the number of pieces is also polynomial: if each bid has a maximum of c pieces, then the running time can be derived by substituting cn for

each occurrence of n.

4 Computing VCG Payments

A naive approach to computing the VCG payments requires solving the allocation problem n times, removing each agent in turn. We extend our approximation scheme for the generalized knapsack, and determine all n payments in *total* time $O(\alpha T \log(\alpha n/\varepsilon))$, where $1 \leq C(\mathcal{I} \setminus i)/C(\mathcal{I}) \leq \alpha$, for a constant upper bound, α, and T is the complexity of solving the allocation problem once. This α-bound can be justified as a "no monopoly" condition, because it bounds the marginal value that a single buyer brings to the auction. Similarly, in the reverse variation we can compute the VCG payments to each supplier in time $O(\alpha T \log(\alpha n/\varepsilon))$, where α bounds the ratio $C(\mathcal{I} \setminus i)/C(\mathcal{I})$ for all i.

Our overall strategy will be to build two dynamic programming tables, one forward and one backward, for each midrange element (l, j), and use these tables to solve all subproblems without each agent. The forward table is built by considering the agents in the order of their indices, where as the backward table is built by considering them in the reverse order. The optimal solution corresponding to $C(\mathcal{I} \setminus i)$ can be broken into two parts: one corresponding to the first $(i-1)$ agents and the other corresponding to the last $(n-i)$ agents. As the $(i-1)$th row of the forward table corresponds to the suppliers with the first $(i-1)$ indices, an approximation to the first part will be contained in $(i-1)$th row of the forward table. Similarly, the $(n-i)$th row of the backward table will contain an approximation for the second part. We first present a simple but inefficient way of computing the approximate value of $C(\mathcal{I} \setminus i)$, which serves to illustrate the main idea of our algorithm. Then we present an improved scheme, which uses the fact that the elements in the rows are sorted, to compute the approximate value more efficiently.

In the following, we concentrate on computing an allocation in which x_ℓ^j is midrange, and some agent $i \neq \ell$ removed. This will be a component in computing an approximation to $C(\mathcal{I} \setminus i)$, the value of the solution to the generalized knapsack without bids from agent i. We begin with the simple scheme.

4.1 A Simple Approximation Scheme

Let *mKnapsack*(ℓ, j) be the multiple choice knapsack problem, when (ℓ, j) is considered as the midrange element. We implement the scaled dynamic programming algorithm for *mKnapsack*(ℓ, j) with two alternate orderings over the other suppliers, $k \neq \ell$, one with suppliers ordered $1, 2, \ldots, n$, and one with suppliers ordered $n, n-1, \ldots, 1$. We call the first table the *forward* table, and denote it F_ℓ, and the

second table the *backward* table, and denote it B_ℓ. The subscript ℓ reminds us that the agent ℓ is midrange. [6]

In building these tables, we use the same scaling factor as before; namely, the cost of a tuple t_i^j is scaled as follows:

$$scost(t_i^k) \;=\; \lceil \frac{n\,cost(t_i^k)}{\varepsilon\, cost(A)} \rceil$$

where $cost(A)$ is the upper bound on $C(\mathcal{I})$, given by our 2-approximation scheme. In this case, because $C(\mathcal{I} \setminus i)$ can be α times $C(\mathcal{I})$, the scaled value of $C(\mathcal{I} \setminus i)$ can be at most $n\alpha/\varepsilon$. Therefore, the cost dimension of our dynamic program's table will be $n\alpha/\varepsilon$.

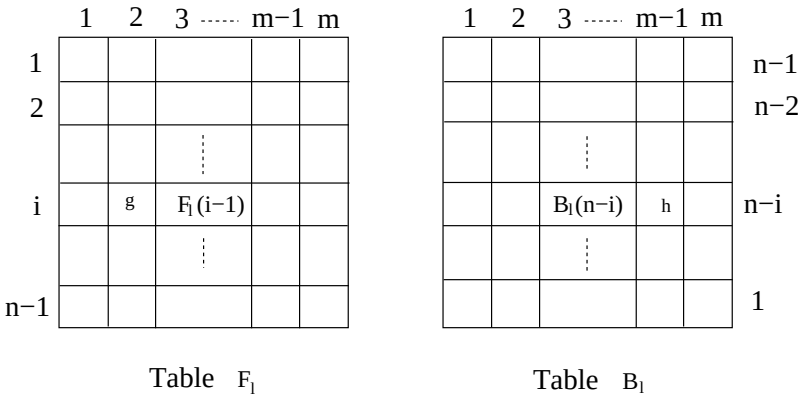

Fig. 3. Computing VCG payments. $m = \frac{n\alpha}{\varepsilon}$

Now, suppose we want to compute a $(1+\epsilon)$-approximation to the generalized knapsack problem restricted to element (ℓ, j) midrange, and further restricted to *remove* bids from some supplier $i \neq \ell$. Call this problem *gKnapsack*$^{-i}(\ell, j)$.

Recall that the ith row of our dynamic progamming table stores the best solution possible using only the first i agents excluding agent ℓ, all of them either cleared at zero, or on anchors. These first i agents are a different subset of agents in the forward and the backward tables. By carefully combining one row of F_ℓ with one row of B_ℓ we can compute an approximation to *gKnapsack*$^{-i}(\ell, j)$. We consider the row of F_ℓ that corresponds to solutions constructed from agents $\{1, 2, \ldots, i-1\}$, skipping agent ℓ. We consider the row of B_ℓ that corresponds to solutions constructed from agents $\{i+1, i+2, \ldots, n\}$, again skipping agent ℓ. The rows are labeled $F_l(i-1)$ and $B_l(n-i)$ respectively. [7] The scaled costs for acquiring these units

[6] We could label the tables with both ℓ and j, to indicate the jth tuple is forced to be midrange, but omit j to avoid clutter.

[7] To be precise, the index of the rows are $(i-2)$ and $(n-i)$ for F_ℓ and B_ℓ when $\ell < i$, and $(i-1)$ and $(n-i-1)$, respectively, when $\ell > i$.

are the column indices for these entries. To solve *gKnapsack*$^{-i}(\ell, j)$ we choose one entry from row $F_\ell(i-1)$ and one from row $B_\ell(n-i)$ such that their total quantity exceeds $M - u_\ell^{j+1}$ and their combined cost is minimum over all such combinations. Formally, let $g \in F_\ell(i-1)$, and $h \in B_\ell(n-1)$ denote entries in each row, with $size(g)$, $size(h)$, denoting the number of units and $cost(g)$ and $cost(h)$ denoting the *unscaled* cost associated with the entry. We compute the following, *subject to the condition* that g and h satisfy $size(g) + size(h) > M - u_\ell^{j+1}$:

$$\min_{g \in F_\ell(i-1), h \in B_\ell(n-i)} \left\{ cost(g) + cost(h) + p_\ell^j \cdot \max\{u_\ell^j,\ M - size(g) - size(h)\} \right\} \quad (7)$$

Lemma 5 *Suppose A^{-i} is an optimal solution of the generalized knapsack problem without bids from agent i, and suppose that element (ℓ, j) is the midrange element in the optimal solution. Then, the expression in Eq. 7, for the restricted problem gKnapsack$^{-i}(\ell, j)$, computes a $(1+\varepsilon)$-approximation to A^{-i}.*

PROOF. From earlier, we define $cost(A^{-i}) = C(\mathcal{I} \setminus i)$. We can split the optimal solution, A^{-i}, into three disjoint parts: x_ℓ corresponds to the midrange supplier, x_i corresponds to first $i-1$ suppliers (skipping agent l if $l < i$), and x_{-i} corresponds to last $n-i$ suppliers (skipping agent l if $l > i$). We have:

$$cost(A^{-i}) = cost(x_i) + cost(x_{-i}) + p_\ell^j x_\ell^j$$

Let $r_i = scost(x_i)$ and $r_{-i} = scost(x_{-i})$. Let y_i and y_{-i} be the solution vectors corresponding to scaled cost r_i and r_{-i} in $F_\ell(i-1)$ and $B_\ell(n-i)$, respectively. From Lemma 3 we conclude that,

$$cost(y_i) + cost(y_{-i}) - cost(x_i) - cost(x_{-i}) \leq \varepsilon\, cost(A)$$

where $cost(A)$ is the upper-bound on $C(\mathcal{I})$ computed with the 2-approximation.

Among all equal scaled cost solutions, our dynamic program chooses the one with maximum units. Therefore we also have,

$$(size(y_i) \geq size(x_i)) \ \text{ and } \ (size(y_{-i}) \geq size(x_{-i}))$$

where we use shorthand $size(x)$ to denote total number of units in all tuples in x.

Now, define $y_\ell^j = max(u_\ell^j, M - size(y_i) - size(y_{-i}))$. From the preceding inequalities, we have $y_\ell^j \leq x_\ell^j$. Since (y_ℓ^j, y_i, y_{-i}) is also a feasible solution to the generalized knapsack problem without agent i, the value returned by Eq. 7 is at most

$$\begin{aligned} cost(y_i) + cost(y_{-i}) + p_\ell^j y_\ell^j &\leq C(\mathcal{I} \setminus i) + \varepsilon\, cost(A) \\ &\leq C(\mathcal{I} \setminus i) + 2\, cost(A^*)\varepsilon \\ &\leq C(\mathcal{I} \setminus i) + 2C(\mathcal{I} \setminus i)\varepsilon \end{aligned}$$

This completes the proof. □

A naive implementation of this scheme will be inefficient because it might check $(n\alpha/\varepsilon)^2$ *pairs* of elements, for any particular choice of (ℓ, j) and choice of dropped agent i. In the next section, we present an efficient way to compute Eq. 7, and eventually to compute the VCG payments.

4.2 Improved Approximation Scheme

Our improved approximation scheme for the winner-determination problem without agent i uses the fact that elements in $F_\ell(i-1)$ and $B_\ell(n-i)$ are sorted; specifically, both unscaled $cost$ and quantity (i.e. $size$) increases from left to right.

As before, let g and h denote generic entries in $F_\ell(i-1)$ and $B_\ell(n-i)$ respectively. To compute Eq. 7, we consider all the tuple pairs, and first divide the tuples that satisfy condition $size(g) + size(h) > M - u_\ell^{j+1}$ into two disjoint sets. For each set we compute the best solution, and then take the best between the two sets.

[case I: $size(g) + size(h) \geq M - u_l^j$**]**

The problem reduces to

$$\min_{g \in F_\ell(i-1),\ h \in B_\ell(n-i)} \left\{ cost(g) + cost(h) + p_l^j u_\ell^j \right\} \tag{8}$$

We define a pair (g, h) to be *feasible* if $size(g) + size(h) \geq M - u_\ell^j$. Now to compute Eq. 8, we do a forward and backward walk on $F_\ell(i-1)$ and $B_\ell(n-i)$ respectively. We start from the smallest index of $F_\ell(i-1)$ and move right, and from the highest index of $B_\ell(n-i)$ and move left. Let (g, h) be the current pair. If (g, h) is feasible, we decrement B's pointer (that is, move backward) otherwise we increment F's pointer. The feasible pairs found during the walk are used to compute Eq. 8. The complexity of this step is *linear* in size of $F_\ell(i-1)$, which is $O(n\alpha/\varepsilon)$.

[case II: $M - u_\ell^{j+1} \leq size(g) + size(h) \leq M - u_\ell^j$**]**

The problem reduces to

$$\min_{g \in F_\ell(i-1),\ h \in B_\ell(n-i)} \left\{ cost(g) + cost(h) + p_l^j (M - size(g) - size(h)) \right\}$$

To compute the above equation, we transform the above problem to another problem using modified cost, which is defined as:

$$mcost(g) = cost(g) - p_l^j \cdot size(g)$$
$$mcost(h) = cost(h) - p_l^j \cdot size(h)$$

The new problem is to compute

$$\min_{g \in F_\ell(i-1),\ h \in B_\ell(n-i)} \left\{ mcost(g) + mcost(h) + p_\ell^j M \right\} \tag{9}$$

The modified cost simplifies the problem, but unfortunately the elements in $F_\ell(i-1)$ and $B_\ell(n-i)$ are no longer sorted with respect to $mcost$. However, the elements are still sorted in quantity and we use this property to compute Eq. 9. Call a pair (g, h) *feasible* if $M - u_\ell^{j+1} \leq size(g) + size(h) \leq M - u_\ell^j$. Define the *feasible set* of g as the elements $h \in B_\ell(n-i)$ that are feasible given g. As the elements are sorted by quantity, the feasible set of g is a contiguous subset of $B_\ell(n-i)$ and shifts left as g increases.

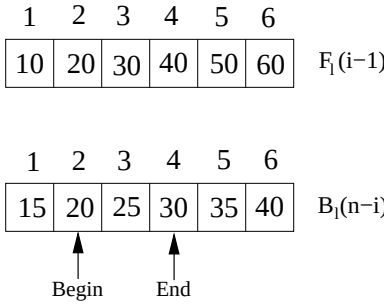

Fig. 4. The *feasible set* of $g = 3$, defined on $B_\ell(n-i)$, is $\{2, 3, 4\}$ when $M - u_\ell^{j+1} = 50$ and $M - u_\ell^j = 60$. $Begin$ and End represent the start and end pointers to the feasible set.

Therefore, we can compute Eq. 9 by doing a forward and backward walk on $F_\ell(i-1)$ and $B_\ell(n-i)$ respectively. We walk on $B_\ell(n-i)$, starting from the highest index, using two pointers, $Begin$ and End, to indicate the start and end of the current feasible set. We maintain the feasible set as a *min heap*, where the key is modified cost. To update the feasible set, when we increment F's pointer(move forward), we walk left on B, first using End to remove elements from feasible set which are no longer feasible and then using $Begin$ to add new feasible elements. For a given g, the only element which we need to consider in g's feasible set is the one with minimum modified cost which can be computed in constant time with the *min heap*. So, the main complexity of the computation lies in heap updates. Since, any element is added or deleted at most once, there are $O(\frac{n\alpha}{\varepsilon})$ heap updates and the time complexity of this step is $O(\frac{n\alpha}{\varepsilon} \log \frac{n\alpha}{\varepsilon})$.

Putting this altogether, the algorithm for to implement the VCG mechanism in our procurement setting works as follows. First, using the 2 approximation algorithm, we compute an upper bound on $C(\mathcal{I})$. We use this bound to scale down the tuple costs. Using the scaled costs, we build the forward and backward tables corresponding to each tuple (ℓ, j). The forward tables are used to compute $C(\mathcal{I})$. To compute $C(\mathcal{I} \setminus i)$, we iterate over all the possible midrange tuples and use the corresponding forward and backward tables to compute the locally optimal solution using the above scheme. Among all the locally optimal solutions we choose one with the minimum total cost.

The most expensive step in the algorithm is computation of $C(\mathcal{I} \setminus i)$. The time complexity of this step is $O(\frac{n^2\alpha}{\varepsilon} \log \frac{n\alpha}{\varepsilon})$ as we have to iterate over all $O(n)$ choices of (ℓ, j), for all $\ell \neq i$, and each time use the above scheme to compute Eq. 7. In the worst case, we might need to compute $C(\mathcal{I} \setminus i)$ for all n suppliers, in which case the final complexity of the algorithm will be $O(\frac{n^3\alpha}{\varepsilon} \log \frac{n\alpha}{\varepsilon})$.

Theorem 4 *We can compute an $\epsilon/(1+\epsilon)$-strategyproof approximation to the VCG mechanism in the forward and reverse multi-unit auctions in worst-case time $O(\frac{n^3\alpha}{\varepsilon} \log \frac{n\alpha}{\varepsilon})$.*

It is interesting to recall that $T = O(\frac{n^3}{\varepsilon})$ is the time complexity of the FPTAS to the generalized knapsack problem with all agents. Our combined scheme computes an approximation to the complete VCG mechanism, including payments to $O(n)$ agents, in time complexity $O(T \log(n/\varepsilon))$, taking the no-monopoly parameter, α, as a constant.

Thus, our algorithm performs much better than the naive scheme, which computes the VCG payment for each agent by solving a new instance of generalized knapsack problem. The speed up comes from the way we solve *gKnapsack*$^{-i}(\ell, j)$. The time complexity of computing *gKnapsack*$^{-i}(\ell, j)$ by creating a new dynamic programming table will be $O(\frac{n^2}{\varepsilon})$ but by using the forward and backward tables, the complexity is reduced to $O(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon})$. We can further improve the time complexity of our algorithm by computing Eq. 7 more efficiently. Currently, the algorithm uses heap, which has logarithmic update time. In worst case, we can have two heap update operations for each element, which makes the time complexity super linear. If we can compute Eq. 7 in linear time then the complexity of computing the VCG payment will be same as the complexity of solving a single generalized knapsack problem.

5 Conclusions

We have presented a fully polynomial-time approximation scheme for the single-good multi-unit auction problem, using a marginal-decreasing piecewise-constant bidding language with quantity-based side constraints. Our scheme is both approximately efficient and approximately strategyproof within any specified factor $\varepsilon > 0$. As such it is an example of computationally tractable ε-dominance result, as well as an example of a non-trivial but approximable allocation problem. It is particularly interesting that we are able to compute the payments to n agents in a VCG-based mechanism in worst-case time $O(T \log n)$, where T is the time complexity to compute the solution to a single allocation problem.

References

[1] Aaron Archer, Christos Papadimitriou, Kunal Talwar, and Eva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 205–214, 2003.

[2] Lawrence M Ausubel and Paul R Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1:1–42, 2002.

[3] Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh V Vohra. Linear programming and Vickrey auctions. In Brenda Dietrich and Rakesh Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, pages 75–116. IMA Volumes in Mathematics and its Applications, Springer-Verlag, 2001.

[4] Sushil Bikhchandani and Joseph M Ostroy. The package assignment model. *Journal of Economic Theory*, 107(2):377–406, 2002.

[5] E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[6] Peter Cramton. Spectrum auctions. In Martin Cave, Sumit Majumdar, and Ingo Vogelsang, editors, *Handbook of Telecommunications Economics*. Elsevier Science B.V., Amsterdam, 2002.

[7] Sven de Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *Informs Journal on Computing*, 15(3):284–309, 2003.

[8] Rafael Epstein, Lysette Henriquez, Jaime Catalan, Gabriel Y Weintraub, and Cristian Martinez. A combinatorial auction improves school meals in Chile. *Interfaces*, 32(6):1–14, 2002.

[9] Marta Eso, Soumyadip Ghosh, Jayant R Kalagnanam, and Laszlo Ladanyi. Bid evaluation in procurement auctions with piece-wise linear supply curves. Technical Report RC 22219, IBM TJ Watson Research Center, 2001. Submitted to Journal of Heuristics.

[10] Joan Feigenbaum and Scott Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[11] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman and Company, New York, 1979.

[12] G V Gens and E V Levner. Computational complexity of approximation algorithms for combinatorial problems. *Mathematical Foundations of Computer Science*, pages 292–300, 1979.

[13] Robert L Graves, Linus Schrage, and Jayaram Sankaran. An auction method for course registration. *Interfaces*, 23(5):81–92, 1993.

[14] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[15] A Hohner, J Rich, E Ng, G Reid, A Davenport, J Kalagnanam, H S Lee, and C An. Combinatorial and quantity discount procurement auctions provide benefts to Mars Incorporated and its suppliers. *Interfaces*, 2003. The Edelman Issue.

[16] O Ibarra and C E Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *JACM*, 22:463–468, 1975.

[17] Atsushi Iwasaki, Makoto Yokoo, and Kenji Terada. A robust open ascending-price multi-unit auction protocol against false-name bids. *Decision Support Systems*, 2004. Special issue dedicated to the Fourth ACM Conference on Electronic Commerce (EC'03).

[18] Jayant R Kalagnanam, Andrew J Davenport, and H S Lee. Computational aspects of clearing continous call double auctions with assignment constraints and indivisible demand. *Electronic Commerce Journal*, 1(3):221–238, 2001.

[19] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[20] Vijay Krishna and Motty Perry. Efficient mechanism design. Technical report, Pennsylvania State University, 2000. Available at: http://econ.la.psu.edu/˜vkrishna/vcg18.ps.

[21] R Lavi, A Mu'alem, and N Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th Annual Symposium on Foundations of Computer Science*, 2003.

[22] John O Ledyard, Mark Olson, David Porter, Joseph A Swanson, and David P Torma. The first use of a combined value auction for transportation services. *Interfaces*, 32:4–12, 2002.

[23] Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, September 2002.

[24] Robert B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

[25] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, pages 242–252, 2000.

[26] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[27] David C. Parkes. Auction design with costly preference elicitation. *Annals of Mathematics and AI*, 2003. Special Issue on the Foundations of Electronic Commerce, Forthcoming.

[28] David C Parkes, Jayant R Kalagnanam, and Marta Eso. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[29] Michael H Rothkopf, Aleksandar Pekeč, and Ronald M Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[30] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[31] James Schummer. Almost dominant strategy implementation. Technical report, MEDS Department, Kellogg Graduate School of Management, 2002.

[32] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[33] Michael P Wellman, William E Walsh, Peter R Wurman, and Jeff K MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[34] Makoto Yokoo, Y Sakurai, and S Matsubara. The effect of false-name bids in combinatorial auctions: New Fraud in Internet Auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.