Generalized Knapsack Solvers for Multi-unit Combinatorial Auctions: Analysis and Application to Computational Resource Allocation

Terence Kelly

Hewlett-Packard Laboratories, Palo Alto CA 94304 USA
kterence@hpl.hp.com

Abstract. The problem of allocating discrete computational resources motivates interest in general multi-unit combinatorial exchanges. This paper considers the problem of computing optimal (surplus-maximizing) allocations, assuming unrestricted quasi-linear preferences. We present a solver whose pseudo-polynomial time and memory requirements are linear in three of four natural measures of problem size: number of agents, length of bids, and units of each resource. In applications where the number of resource *types* is inherently a small constant, e.g., computational resource allocation, such a solver offers advantages over more elaborate approaches developed for high-dimensional problems.

We also describe the deep connection between auction winner determination problems and generalized knapsack problems, which has received remarkably little attention in the literature. This connection leads directly to pseudo-polynomial solvers, informs solver benchmarking by exploiting extensive research on hard knapsack problems, and allows E-Commerce research to leverage a large and mature body of literature.

1 Introduction

Recent years have witnessed an explosion of interest in combinatorial auctions (CAs), which permit agents to define utility over *bundles* of different types of goods. Although CAs are applicable to a wide range of allocation problems, the U.S. Federal Communications Commission's spectrum allocation problem largely motivated the 1990s surge of CA research [1,2]. Special properties of spectrum auctions—particularly the restriction that only a single unit of each type of good is available—received much attention in E-commerce research literature. An important measure of problem size in a single-unit CA is the number of good types, and for this measure the winner determination problem (WDP) is NP-hard by reduction from the weighted set packing problem [3].

An unfortunate consequence of excessive attention to single-unit CAs has been excessive pessimism regarding efficient and exact winner determination in more general problems. The few papers that have considered multi-unit CAs (MUCAs) report that the WDP is NP hard when problem size is measured by number of good types [4,5,1]. Other natural measures, e.g., number of available *units* of each good, number of agents, and the length of bids, receive far less attention.

This paper follows a very different trajectory from practical motivation to conclusions regarding the computational complexity of CA WDPs. We begin with the problem of allocating resources in large computing centers. The number of resource *types* in this problem is a *small constant*, whereas the number of *units* of each resource is large and variable. The optimal allocation problem is a generalized multi-dimensional knapsack problem (MDKP): allocating a bundle of goods to an agent reduces the pool of available goods, just as placing an item in a container with multiple capacity constraints (e.g., weight, volume) reduces its remaining capacity along each dimension.

The deep connection between WDPs and KPs leads to pseudo-polynomial exact algorithms for problems of fixed dimensionality. Very simple exact solvers exist whose time and memory requirements are *linear* in the number of agents, length of bids, and number of units of each resource. Such solvers are entirely practical for low-dimensional problem instances (i.e., few resource types) and are an attractive default solution method whenever their computational costs are not prohibitive. In all cases they provide a well-understood baseline for comparison with more elaborate methods.

Straightforward MUCA WDP solvers inspired by the auction-knapsack connection invite more detailed, more balanced, and more nuanced analyses than are typically performed on complex heuristic solvers designed for high-dimensional problems. Knapsack-based WDP solvers furthermore support very general combinatorial exchanges with essentially no restrictions on the expression of agent utility functions. The connection between CA WDPs and generalized KPs allows us to retain much of the flexibility and generality of integer programming [6] while exploiting the special structure of KPs to obtain simple and efficient exact solvers. In special cases such as *single-good* multi-unit auctions, textbook uni-dimensional KP solvers compare rather well with specialized WDP algorithms. Finally, WDP benchmarks can draw upon extensive Operations Research literature on hard KP instances.

The boundaries of the present investigation are as follows: We consider only one-shot sealed-bid auctions, an important subset of auction types in a comprehensive taxonomy [7]. We consider only discrete allocation (integral quantities of goods). Our results apply to the allocator of proper economic mechanisms such as the Generalized Vickrey Auction (GVA) [8] or Vickrey-Clarke-Groves (VCG) mechanisms [9], but we do not consider incentive issues surrounding auctions. Finally, although a wide range of approximation schemes for KPs have been proposed, we restrict attention to exact methods. This is appropriate in light of recent results on the necessity of exact solvers for incentive-compatible mechanisms [10,11,12,13]. A longer version of this paper [14] includes material omitted due to space limitations.

The remainder of this paper is structured as follows: Section 2 motivates interest in low-dimensional MUCAs with a discussion of resource allocation in large computing centers. Section 3 formulates our general allocation problem and explains its relation to auction winner determination. Section 4 presents a general solver for multi-unit CAs with unrestricted preference expression and analyzes its computational complexity. Section 4.2 discusses hard KP instances, Section 5 reviews related work, and Section 6 concludes with a discussion.

2 Motivation: Data Center Allocation

Large tightly-coupled computers remain popular for enterprise computing, and today entire data centers comprising large numbers of loosely-coupled hosts are offered as commercial products [15]. Resource allocation in both contexts has several properties that recommend auction-mediated negotiation, and knapsack-based optimal allocators are ideal WDP solvers for these contexts.

The number of abstract resource types in computational allocation problems is inherently a *small constant*, because only a few fundamental operations can be performed on data: data can be manipulated, stored, and transported. Corresponding resource types—processing, storage, and bandwidth—often suffice in models of computational resource allocation [16]. For reasons of fault isolation, security, and performance isolation, most computing resources are allocated in *integral* quantities; examples include CPUs, switch ports, and logical devices (LDEVs) in consolidated storage arrays. By contrast, the number of *units* of each resource is large and expands with user needs.

Data centers are partitioned so that an application's performance depends only upon the resources it receives; in auction contexts this property is sometimes called "no externalities" [17]. Multi-tiered applications for large computing environments are *horizontally scalable* by design, i.e., they exploit variable quantities of resources at each tier. Application performance exhibits both complementarities and substitutabilities across resource types. For example, one application may require minimal quantities of both memory and bandwidth in order to perform acceptably; another may compensate for lack of bandwidth by exploiting an additional CPU for data compression. The utility that accrues to an application is a complex function of the *bundle* of resources it receives; this property recommends combinatorial auctions.

While the number of applications simultaneously sharing an enterprise computing center may be large, the number of self-interested agents among whom resources are allocated may be small. Agents might correspond to departments or projects within a firm, or to firms within a consortium that jointly owns a data center. If the number of agents is so large that each agent's potential influence on allocative outcomes is negligible, competitive (i.e., non-strategic) behavior may be a reasonable normative assumption. However strategic behavior is to be expected if few agents are involved. Incentive-compatible mechanisms (in which truth-telling is a dominant strategy for agents) are therefore desirable, even for allocation within a hierarchical organization [18]. Given that the incentive properties of GVA/VCG mechanisms sometimes require *exact* WDP solvers [10, 11, 12, 13], we prefer exact solvers to approximate ones where possible.

Computational resource allocation can be formalized as a generalized knapsack problem [19]; Section 3 describes a suitable formulation. Our straightforward solver, presented in Section 4, is appropriate to the special properties of data-center allocation. Its computational complexity is exponential in the number of resource *types* but is linear in the number of available *units* of each resource and in all other natural measures of problem size. A simple implementation of the solver produces, as a side effect, a table describing the aggregate utility of any subset of the data center's resource pool, thereby providing a wealth of information about the marginal value of various resource types. This information might be useful for purposes other than allocation, e.g., capacity planning.

3 Problem Formulation

We are given R resource types and T agents. At most N_r indivisible units of resource type r are available, $r = 1, \ldots, R$. Each agent's utility function is represented by defining utility over a list of resource bundles; the list may be arbitrarily long, and may therefore may represent any utility function. If agent utility naturally takes a more compact form than a list of (bundle, utility) pairs, the former may easily be translated into the latter. The length of utility functions when defined explicitly over bundles is not prohibitive in low-dimensional cases (i.e., where the number of resource types R is a small constant).

Our goal is to maximize aggregate utility by choosing exactly one bundle from each list, subject to resource scarcity. Let B_t denote the number of bundles in agent t's utility function, and let $q_{tb} = (q_{1tb}, \ldots, q_{Rtb})$ and u_{tb} respectively denote the quantities of resources in bundles and the utility of bundles, $b = 1, \ldots, B_t$. Binary decision variable $x_{tb} = 1$ if agent t receives the bth resource bundle on its list, zero otherwise. Formally, our "multi-dimensional multiple-choice knapsack problem" (MDMCK) is the following integer program:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{t=1}^{T} \sum_{b=1}^{B_t} x_{tb} u_{tb} & & & (1) \\ \text{subject to} \quad & \textstyle\sum_{b=1}^{B_t} x_{tb} = 1 & & t = 1, \ldots, T & (2) \\ & \textstyle\sum_{t=1}^{T} \sum_{b=1}^{B_t} x_{tb} q_{rtb} \leq N_r & & r = 1, \ldots, R & (3) \end{aligned}$$

The inequality in Equation 3 permits unallocated goods; to forbid them we simply replace it with equality. In the latter case we can express arbitrary disposal costs of unallocated goods via an additional agent utility function. The solver of Section 4 takes a different approach: it accepts an explicit disposal cost function as an input.

MDMCK includes classic knapsack problems as special cases [19]. Extensive literature exists on these special cases, but relatively little on MDMCK itself. Kellerer *et al.* devote roughly three pages to MDMCK and identify approximate heuristic algorithms dating back to 1997 [20]. They report that to the best of their knowledge no exact algorithm for MDMCK has ever been published. In fact, Tennenholtz briefly sketched an exact solver suitable for low-dimensional MDMCK instances, without analyzing its complexity or connecting the WDP to generalized KPs [21].

Two-resource MDMCK admits simple graphical illustration (Figure 1). A bundle/utility pair in a utility function is represented as a rectangle labeled with agent ID (left). Utility functions are collections of such rectangles (center). The solution is illustrated at right: a bundle is chosen from each utility function such that utility is maximized while total resource usage does not exceed any capacity dimension.

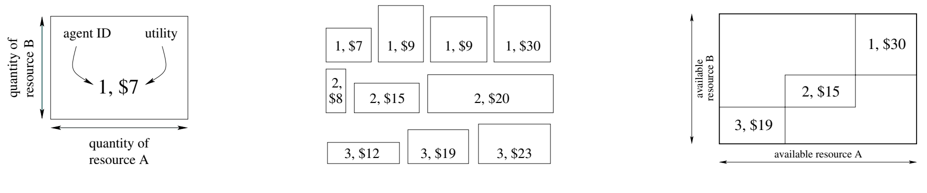

Fig. 1. 2-D MDMCK. *Left:* resource bundle. *Center:* utility functions. *Right:* optimal allocation.

3.1 Application to Auctions

In an auction setting, we refer to an agent's list of (q_{tb}, u_{tb}) pairs as its *bid*. We shall ignore the relationship between an agent's *reported* and *true* utility functions except to note that they may differ and that our allocator receives the former. The constraint of Equation 3 ensures that each agent receives exactly one bundle defined by its bid. In other words, we permit "XOR bids," which in turn permit the expression of arbitrary preferences [17].

The MDMCK formulation requires that each agent's utility depends only on the bundle of resources the agent itself receives ("no externalities" [17]). No other restrictions on agent preferences are inherent. For example, MDMCK allows goods to be "bads," i.e., free disposal is not required. Furthermore agent utility need not be "normalized" in the sense that no change in goods owned implies no change in utility.

Some prior work on single-good-type/multi-unit auctions has restricted the form of bids, e.g., demand must be monotonic in per-unit price [22] or atomic bids are forbidden [23]; monotonicity restrictions have also appeared in multi-good CA analyses [24]. In the single-good-type case, divisibility is required for existence of a uniform price that maximizes surplus *according to restricted-form bids* (which might not represent actual agent preferences). Uniform prices are sometimes desirable, e.g., for reasons of perceived fairness. The real motivation for bid restrictions, however, has sometimes been to facilitate efficient WDP algorithms [25].

Computational issues aside, the greater generality and flexibility of a MDMCK formulation makes it attractive if uniform prices are not required. The components of resource vectors q and utilities u may assume both negative and positive values, allowing agents to express willingness to engage in complex *atomic* (all-or-nothing) transactions. Thus the MDMCK formulation supports very general combinatorial exchanges, e.g., the dozen CA variants considered in Ref. [26].

3.2 Auction and KP Taxonomies

CA WDPs are often linked to set packing, even in the multi-unit case [5]. Connections with generalized knapsack problems, however, seem more natural and more useful for several reasons. First, KPs are more widely known among nonspecialists, e.g., implementors in industry; they are intuitive, memorable, and invite graphical interpretation (Figure 1). KPs are also far more widely studied. Most importantly, KPs admit pseudo-polynomial solution under restrictions that are sometimes acceptable in practice. Whereas connections with set packing have led to the pessimistic view that "CA WDPs are NP hard," the knapsack connection encourages cautious optimism.

Consider three aspects of sealed-bid auctions and their knapsack counterparts:

1. number of types of goods in an auction / dimensionality of a KP;
2. number of units of each good / capacity of KP container in each dimension; and
3. number of bundles in bids / the "multiple-choice" aspect of KP.

In each case the characteristic may be single or multiple, e.g., an auction may involve multiple units of a single good type, or single units of multiple good types. Table 1 summarizes the seven meaningful combinations of these possibilities. When KP items

Table 1. Auction types and winner-determination problems (S=single, M=multiple)

good types	units	bundles	common name / examples	winner-determination problem
S	S	S	first price	find max
S	M	S	double auctions, single-quantity bids	0-1 KP; subset-sum if #units $\propto$ utility
S	M	M	double auctions, XOR bids	multiple-choice KP (MCKP)
M	S	S	"combinatorial auctions"	weighted set packing (WSP) [3]
M	S	M	single-unit CA, XOR bids	convert to WSP via "dummy goods"
M	M	S	multi-unit CA, single-bundle bids	multi-dimensional KP (MDKP)
M	M	M	multi-unit CA, XOR bids [4]	MDMCK [19]

are partitioned into disjoint sets and we must choose exactly one item from each set, we say that a "multiple-choice" constraint applies; this corresponds to an XOR constraint across elements of a compound bid. The most general KP shown is MDMCK, which corresponds to multi-unit CAs with arbitrary XOR bids (MMM in Table 1).

It is straightforward to convert an instance of the MSM problem to an MSS instance by adding "dummy goods" to enforce multiple-choice/XOR constraints: introduce an extra good type for each agent, one unit of which is included in each of the agent's bundles and of which exactly one unit is available [4]. MMM instances can be converted to MMS instances in the same way. This transformation increases the dimensionality of problem instances, which may increase computational burdens for some solvers.

Several of the correspondences in Table 1 have been noted previously. Kothari *et al.* mention in a footnote that their single-good multi-unit WDP is similar "in spirit" to MCKP, citing a 1970s reference [22]. However they quickly dismiss the connection on grounds that MCKP leads to an infeasible formulation. In fact, simple MCKP solvers in modern texts scale rather well with problem size (see Section 5.2), and efficient specialized solvers are the subject of sophisticated recent research [27]. Holte observes that Operations Researchers have long investigated MDKPs that are substantively identical to multi-unit CA WDPs [28], contrary to claims in recent E-commerce literature that MUCA WDPs were never before studied [4]. Years later, however, MUCA WDP research that cites Holte does not mention the connection he made [29]. A very recent text on KPs discusses Holte's insight in considerable detail but does not make the connection between MDMCK and multi-unit CAs with XOR bids; instead it suggests the use of dummy goods to enforce XOR constraints for a MDKP solver [20]. Overall, we find remarkably few references to knapsack problems in recent literature on auction WDPs, and nothing approaching a comprehensive treatment of the relationship between the two in the E-commerce literature. Section 5 considers in greater detail the state of the E-commerce literature in this regard.

4 Dynamic Programming Solver

This section presents a simple dynamic programming (DP) algorithm for MDMCK; it generalizes multi-dimensional and multiple-choice KP solvers [30, 20].

Let $N = (N_1, \ldots, N_R)$ denote the multi-dimensional "size" of our resource pool, and let 0 denote the R-vector consisting entirely of zeros. We say that $a \geq b$ if every component of vector a is not less than the corresponding component of b.

Given an integer $\hat{t}$ and a resource pool size n, we define $F_{\hat{t}}(n)$ to be the optimal value of our objective function (Equation 1) for the sub-instance of MDMCK involving only agents $1, \ldots, \hat{t}$ and a resource pool of size n. $F_0(n)$ defines the utility of unallocated resources for feasible "leftovers" $n \geq 0$ and defines utility as $-\infty$ for infeasible allocations. Similarly we define $A_{\hat{t}}(n)$ as the bundle assigned to agent $\hat{t}$ by the optimal assignment for the sub-instance defined by $\hat{t}$ and n. F and A may be defined recursively:

$$F_{\hat{t}}(n) = \begin{cases} -\infty & \hat{t} = 0,\ \neg(n \geq 0) \\ D(n) & \hat{t} = 0, \quad n \geq 0 \\ \max\limits_{b \in \mathcal{B}_{\hat{t}}} \{F_{\hat{t}-1}(n - q_{\hat{t}b}) + u_{\hat{t}b}\} & 1 \leq \hat{t} \leq T \end{cases} \tag{4}$$

$$A_{\hat{t}}(n) = \arg\max_{b \in \mathcal{B}_{\hat{t}}} \{F_{\hat{t}-1}(n - q_{\hat{t}b}) + u_{\hat{t}b}\} \quad 1 \leq \hat{t} \leq T \tag{5}$$

where $\mathcal{B}_{\hat{t}} = \{1, \ldots, B_{\hat{t}}\}$ and D expresses the (dis)utility of unallocated resources. To permit unallocated goods at no cost we simply set $D = 0$; to forbid unallocated goods we set $D = -\infty$. $F_T(N)$ is the value of an optimal solution, and the corresponding choices of bundles may be recovered as $A_T(N)$, $A_{T-1}(N - q_{TA_T(N)})$, etc.; conversion to decision variables x_{tb} of Equations 1 through 3 is trivial.

We may evaluate the dynamic program in at least two ways: by constructing tables corresponding to $F(\cdot)$ and $A(\cdot)$ in bottom-up fashion, or by recursively evaluating $F_T(N)$ and $A_T(N)$. The former strategy yields a full $F_T(n)$ table containing information about the marginal utilities of every resource type for every resource pool size $n : 0 \leq n \leq N$; this may be useful for purposes other than allocation, e.g., capacity planning. A disadvantage of the bottom-up approach is that it achieves worst-case performance on *all* inputs. Top-down evaluation may save time on some inputs by evaluating $F(\cdot)$ and $A(\cdot)$ for fewer $(\hat{t}, n)$ pairs, and may permit more space-efficient representation of the tables than naïve arrays. Top-down evaluation admits a variety of optimizations and elaborations, including lower-bound heuristics and pruning via upper bounds; with such embellishments it resembles branch-and-bound (B&B) search. A no-frills top-down C implementation of our solver runs to several dozen lines of code, comparable to succinct uni-dimensional KP solvers [31].

4.1 Computational Complexity

The worst-case time and memory complexity of a straightforward implementation of the the dynamic program are easy to analyze. We assume a bottom-up implementation that stores $F(\cdot)$ and $A(\cdot)$ values in ordinary arrays. We assume that the coefficients describing a problem instance are integers from a bounded range, and without loss of generality we assume that all coefficients are non-negative. (A natural expression of a fully general two-sided exchange WDP might be an instance of MDMCK with negative coefficients, but such an instance can be efficiently transformed to one with non-negative coefficients in a simple pre-processing step without altering the optimal values of decision variables.)

The dynamic program requires storage proportional to $T\prod_{r=1}^{R} N_r$. Evaluating Equations 4 and 5 requires time proportional to $R\sum_{t=1}^{T}(B_t\prod_{r=1}^{R} N_r)$ where the R term is due to the R-dimensional vector subtraction in the recursive calls to F. If $N_r = N$ for each resource, and if each agent defines utility over B resource bundles, then the storage requirement is $O(TN^R)$ and the time requirement is $O(RTBN^R)$. If each agent defines utility over all N^R possible resource bundles (the case of rational preferences) then the time requirement becomes $O(RTN^{2R})$.

The classic 0-1 and integer knapsack problems are NP-hard [32, 33]. MDMCK includes these as special cases, and therefore it too is NP-hard. However knapsack problems are *not* NP-hard in the strong sense, i.e., they admit pseudo-polynomial solution if dimensionality is fixed. See Papadimitriou & Steiglitz for a good discussion of pseudo-polynomial complexity analysis applied to classic KPs [33]. If we need not support problem instances with enormous coefficients, pseudo-polynomial bounds are the most natural and insightful description of algorithmic complexity. Restricting u_{tb}, q_{rtb}, and N_r to the length of modern machine words, e.g., 64 bits, is unlikely to be problematic in practical allocation problems.

For classic uni-dimensional problems, branch-and-bound algorithms are often favored over DP *except for hard problem instances*, where DP usually performs better [30, page 36]. For high-dimensional problems the computational costs of DP are prohibitive and the best method may be general integer programming (IP). Modern IP solvers support convenient and rapid solution of a wide range of WDPs [6] and compute approximate solutions to large MDMCK instances very rapidly [19].

4.2 Hard Knapsack Problems

Real-world CA WDP instances are not available for solver benchmarking, so we must rely on synthetic benchmarks. A thorough evaluation of any WDP solver should include instances intended to mimic typical inputs, such as those generated by CATS [34], as well as hard instances to expose worst-case behavior. The connection between WDPs and KPs allows us to exploit many years of research on hard KP instances for WDP solver evaluation.

There are two ways to construct hard instances of classic uni-dimensional knapsack problems. The first is to make the coefficients enormous; Chvátal describes how large they must be in order to foil a range of common solution methods [35]. We shall continue to assume that coefficients are bounded and therefore focus on the second method, which involves the relationship between bundle size and utility.

The size/utility relationship is easy to visualize in the uni-dimensional case. Figure 2, after Pisinger [36], illustrates four possibilities; Martello *et al.* and Kellerer *et al.* describe others [37, 20]. Strongly-correlated instances are among the hardest

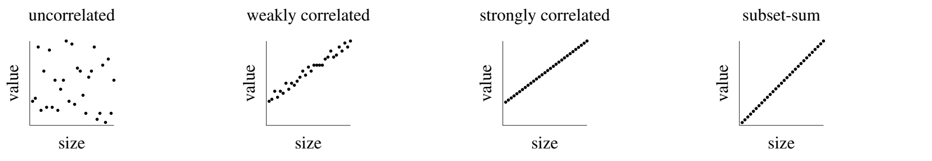

Fig. 2. Uni-dimensional KPs

for today's best KP solvers and are the subject of ongoing research [38, 37]. An extended version of this paper describes how to construct generalized multi-dimensional strongly-correlated instances [14].

It is interesting to note that early empirical evaluations of KP solvers focused excessively on "easy" problem instances, specifically the uncorrelated and weakly-correlated cases of Figure 2; only later did attention within the OR literature shift to characterizing hard instances and using them in solver benchmarks [37]. A similar pattern is evident in evaluations of WDP solvers many years later, as Andersson *et al.* have noted [6]; see also Section 5.2. It is reasonable to speculate that mis-steps in WDP benchmarking might have been avoided if connections between WDPs and KPs had been more prominent in E-commerce research.

5 Related Work

(This section has been reduced due to space limitations; see [14] for the full version.) The literature on knapsack problems is vast and growing. An excellent text by Martello & Toth [30] is now out of print, but a very recent book by Kellerer *et al.* provides updated and expanded coverage, including multi-dimensional problems and MDMCK itself [20]. Martello *et al.* review recent research on exact solutions for large hard instances of 0-1 KP [37]. Pisinger summarizes the state of the art in uni-dimensional KP research c. 1995 [36], much of which is directly applicable to subsequent research on single-good-type/multi-unit WDPs [23, 22].

5.1 WDP-KP Connections

An extensive literature search revealed little mention of the connection between auction WDPs and KPs and nothing approaching a comprehensive treatment. Recent surveys on combinatorial auctions and auction theory [3, 2, 39, 1, 40, 26] do not discuss knapsack problems. The string "knap" appears in exactly five papers among all past proceedings of the ACM Conference on E-Commerce (EC). Two are unrelated to our interests, two mention in passing a relationship between special cases of WDP and KPs [22, 5], and one uses reduction from KP to prove NP-hardness [24]. Occasionally papers in other fora note that single-good-type auction WDPs are KPs, usually to establish intractability and sometimes to note the existence of pseudo-polynomial algorithms [41].

In several cases E-commerce research has missed opportunities to build upon relevant prior work—and failed to acknowledge it—perhaps because the WDP-KP connection has been overlooked. For instance, the fact that multi-dimensional KPs do not admit fully-polynomial approximation, even in the two-dimensional case, has been known since 1979 [20, p. 252]. Twenty years later, the question of whether CA WDPs admit approximation was described as "open" [42, p. 10].

Queries to six literature search engines for "auction," "knapsack," and "auction AND knapsack" yielded results summarized in Table 2. In all cases the conjunctive query yielded far fewer hits than the two basic queries; none of the "auction AND knapsack" papers contained a detailed or systematic treatment of the WDP-KP connection. A few papers mention in passing a deep relationship between WDPs and KPs and a handful casually state that the connection is well known, without saying by whom;

Table 2. Summary of keyword searches, December 2003 and January 2004

Source	Total docs	"auction"	"knapsack"	both
Springer Link	?	152	103	zero
IEEE Xplore	990,765	313	150	zero
ACM Digital Library	125,779	802	427	10
CiteSeer	?	1,686	922	12
Science Citation Index	33,117,604	2,379	989	zero
Elsevier Science Direct	"over 4M"	5,143	2,084	11

see [14] for citations. Somewhat ironically, the only detailed discussion of the connection between combinatorial auction WDPs and generalized KPs of which we are aware occurs in a very recent text written primarily by Operations Researchers with little interest in E-commerce [20, pp. 478–482].

In summary, the WDP-KP relationship is neither noted nor exploited widely in E-commerce research at the intersection of computer science and auction theory. The remainder of this section reviews selected literature on multi-unit auction WDPs, showing how the KP literature can enhance several of these contributions.

5.2 Multi-unit Auction WDPs

Kothari *et al.* consider *single-good-type* multi-unit auctions and introduce a fully-polynomial algorithm to compute approximately surplus-maximizing allocations [22]. Bids are restricted in several ways: they are divisible, the utility they express is monotonic in per-unit price, and their length is bounded. This paper mentions in passing that its allocation problem can be solved by a multiple-choice KP solver and that fully-polynomial approximation algorithms exist for MCKP. However it offers no detailed comparison with earlier approximate MCKP solvers or with simple exact algorithms.

A textbook DP algorithm for MCKP [30, page 78] applied to the single-good multi-unit WDP supports a completely general two-sided exchange with unrestricted bids. In the special case of a forward auction with N units for sale and T agents whose bids define utility over all possible quantities $0,\dots,N$, the (pseudo-polynomial) time and memory requirements of this very simple exact method are respectively $O(TN^2)$ and $O(TN)$. Similar computational properties apply to the special case of a reverse auction; more sophisticated algorithms with improved asymptotic bounds exist [20]. The algorithm of Kothari *et al.* computes a $(1+\varepsilon)$ approximation for the restricted-bid problem and requires $O(T^3/\varepsilon)$ time. A detailed comparison with the textbook DP solver would place the new contribution in better perspective and would illuminate the tradeoffs between computational complexity and generality that are available to us. Discussion of the need for fully-polynomial (vs. pseudo-polynomial) algorithms would help to motivate the new method.

Bassamboo *et al.* consider *online* bid processing in single-good-type multi-unit auctions with indivisible (all-or-nothing) single-quantity bids [43]. They describe a remarkably storage-efficient algorithm for maintaining a small set of potentially winning bids prior to clearing; bids that cannot potentially win at the time they arrive are rejected, permitting the bidder to adjust her bid if desired. These authors note that literature on online knapsack problems exists, but does not precisely match the auction rules they consider.

Tennenholtz notes that the multi-good-type/multi-unit WDP is "tractable" when the number of types of goods is fixed, and describes a longest-paths dynamic programming algorithm in the context of a two-good-type example [21]. It is not clear whether the intended meaning is that polynomial or pseudo-polynomial solutions exist (the former cannot be true, because this WDP includes NP-hard problems MCKP and 0-1 KP as special cases). Neither knapsack problems nor their close relationship with longest-path problems [44, p. 100] are mentioned, nor are time and memory complexity analyses presented. A later version of the paper omits the DP algorithm entirely [45].

WDP solver research for multi-good-type/multi-unit CAs has emphasized heuristic branch-and-bound algorithms [4, 5]. Such approaches are entirely reasonable, particularly for high-dimensional problems in which DP solvers are likely to be infeasible. Comparisons with DP-based KP solvers could enhance B&B investigations by encouraging more detailed analyses of worst-case time and memory requirements in terms of all measures of problem size. B&B research to date has emphasized the number of good *types*, sometimes without detailed quantitative analysis of computational requirements [4]. Furthermore, benchmarks for multi-unit CAs could draw upon extensive research on hard KP instances. Empirical evaluations of MUCA WDP solvers to date have employed similar input synthesis procedures [4, 5, 26], which produce multi-dimensional variants of the uncorrelated and weakly correlated cases of Figure 2; for uni-dimensional KPs, these are not hard instances.

Finally, awareness of the WDP-KP connection would support more succinct and more precise descriptions of novel WDP algorithms. Leyton-Brown *et al.*, for instance, introduce a "polynomial" subroutine for pre-processing bids for a single good type ("singletons") [4,29]. In fact, this subroutine implements the classic *pseudo*-polynomial DP algorithm for the NP-hard 0-1 knapsack problem.

6 Discussion

This paper has compared two very different trajectories of CA research, summarized in Table 3. Motivated largely by FCC spectrum auctions, most CA research over the past decade has taken the number of *types* of goods as a measure of problem size while fixing the number of *units* of each good at 1. This paper begins with the problem of

Table 3. Trajectories of CA research

	single-unit/high-dimensional	multi-unit/low-dimensional
practical motivation	spectrum auctions	computational resource allocation
# good types	variable, high	low, fixed
# units/type	fixed at 1	variable, high
WDP	weighted set packing	generalized knapsack problem
conventional wisdom	"WDP is NP-hard," rational preferences infeasible	*linear* solvers available, rational preferences okay
solver research	heuristic B&B, restricted prefs	exact DP, any preferences
OR leverage	limited, late	extensive, early

computational resource allocation in modern data centers, which involves few types of goods but many units of each. Whereas comparisons with set packing have led to the conclusion that the WDP is intractable in the single-unit/high-dimensional case, different natural measures of problem size lead us to conclude that the WDP admits pseudo-polynomial solution in the multi-unit/low-dimensional case. Realization that WDPs are special cases of MDMCK leads to a very general solver whose simplicity invites thorough analysis.

By recognizing connections between knapsack problems and winner determination, we bring a wealth of Operations Research knowledge to bear on problems central to multi-agent resource allocation. This eliminates duplication of effort by allowing E-commerce research to focus on *typical* WDP instances while leaving to the OR community the task of characterizing *hard* cases. It also allows WDP solver research to focus on novel methods only when real-world instances offer optimization opportunities that are not exploited by general-purpose KP solvers.

Straightforward dynamic-programming KP solvers offer several attractive properties, including analytic tractability and simplicity of implementation. These in turn reduce errors, which have been discovered in elaborate B&B solvers after publication [46]. If nothing else, DP provides a well-understood baseline for comparisons of more sophisticated methods and highlights tradeoffs between algorithmic intricacy and computational efficiency. Furthermore for hard instances of low-dimensional problems, DP may simply outperform alternatives. In the special case of single-good/multi-unit auctions, textbook KP solvers provide exact solutions for unrestricted inputs and scale remarkably well with problem size; at the very least, they merit detailed comparison with approximation algorithms for restricted problems.

We have shown that a practical multi-agent allocation problem involving computational resources lends itself readily to formulation as a generalized knapsack problem, and that for this low-dimensional problem an extremely simple DP solver scales to instances of non-trivial size. In future work we intend to compare the performance of DP, B&B, and integer program solvers on a range of synthetic MDMCK instances and, if possible, to characterize analytically the instances best suited to each solution method.

Acknowledgments

I thank Hans Kellerer, Anshul Kothari, Daniel Lehmann, Kevin Leyton-Brown, Jeff Mason, David Pisinger, Daniel Reeves, Yoav Shoham, Bill Walsh, and Michael Wellman for discussions and pointers to literature. Moises Goldszmidt, Eugene Nudelman, Ryan Porter and the anonymous reviewers provided detailed feedback on earlier versions. Wei Deng helped with the literature review.

References

1. de Vries, S., Vohra, R.V.: Combinatorial auctions: A survey. INFORMS Journal on Computing **15** (2003) 284–309
2. Pekec, A., Rothkopf, M.H.: Combinatorial auction design. Management Science **49** (2003) 1485–1503

3. Rothkopf, M.H., Pekec, A., Harstad, R.M.: Computationally manageable combinatorial auctions. Management Science **44** (1998) 1131–1147
4. Leyton-Brown, K., Shoham, Y., Tennenholtz, M.: An algorithm for multi-unit combinatorial auctions. In: Proc. of the 17th National Conference on Artificial Intelligence. (2000)
5. Gonen, R., Lehmann, D.: Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In: ACM E-Commerce. (2000)
6. Andersson, A., Tenhunen, M., Ygge, F.: Integer programming for combinatorial auction winner determination. In: ICMAS. (2000)
7. Wurman, P.R., Wellman, M.P., Walsh, W.E.: A parametrization of the auction design space. Games and Economic Behavior **35** (2001) 304–338
8. Varian, H., MacKie-Mason, J.K.: Generalized Vickrey auctions. Technical report, Dept. of Economics, University of Michigan (1994)
9. Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford University Press (1995) ISBN 0-19-507340-1.
10. Nisan, N., Ronen, A.: Algorithmic mechanism design. Games and Economic Behavior **35** (2001) 166–196
11. Lehmann, D., O'Callaghan, L., Shoham, Y.: Truth revelation in approximately efficient combinatorial auctions. Journal of the ACM **49** (2002) 577–602
12. Nisan, N., Ronen, A.: Computationally feasible VCG mechanisms. In: Proceedings of the ACM Conference on Electronic Commerce. (2000) 242–252
13. Lavi, R., Mu'alem, A., Nisan, N.: Towards a characterization of truthful combinatorial auctions (extended abstract). In: IEEE FOCS. (2003)
14. Kelly, T.: Generalized knapsack solvers for multi-unit combinatorial auctions. Technical Report HPL-2004-21, HP Labs (2004)
15. Hewlett-Packard Corporation: HP Utility Data Center (UDC) overview (2004) `http://h30046.www3.hp.com/solutions/overview.html`.
16. Hewlett-Packard Corporation: An economy of IT: Allocating resources in the computing utility (2003) `http://www.hpl.hp.com/news/2003/oct_dec/computons.html`.
17. Nisan, N.: Bidding and allocation in combinatorial auctions. In: Proceedings of the Second ACM Conference on Electronic Commerce. (2000) 1–12
18. Ledyard, J.O.: Incentive compatible space station pricing. American Economic Review **76** (1987) 274–279
19. Kelly, T.: Utility-directed allocation. In: First Workshop on Algorithms and Architectures for Self-Managing Systems. (2003) `http://tesla.hpl.hp.com/self-manage03/`. Also available as HP Labs tech report HPL-2003-115.
20. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004) ISBN 3-540-40286-1.
21. Tennenholtz, M.: Some tractable combinatorial auctions. In: Proc. of the 17th National Conference on Artificial Intelligence. (2000) A later and longer version is Ref. [45].
22. Kothari, A., Parkes, D.C., Suri, S.: Approximately-strategyproof and tractable multi-unit auctions. In: ACM E-Commerce. (2003)
23. Wurman, P.R., Walsh, W.E., Wellman, M.P.: Flexible double auctions for electronic commerce: Theory and implementation. Decision Support Systems **24** (1998) 17–27
24. Lehmann, B., Lehmann, D., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. In: ACM E-Commerce. (2001)
25. Walsh, W.: Personal communication (2003)
26. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Winner determination in combinatorial auction generalizations. In: AAMAS. (2002)
27. Pisinger, D.: A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operations Research **83** (1995) 394–410

28. Holte, R.C.: Combinatorial auctions, knapsack problems, and hill-climbing search. In Stroulia, E., Matwin, S., eds.: Lec. Notes in CS. Volume 2056. Springer (2001)
29. Leyton-Brown, K.: Resource Allocation in Competitive Multiagent Systems. PhD thesis, Stanford University (2003)
30. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementation. John Wiley & Sons Ltd. (1990) ISBN 0-471-92420-2.
31. Sedgewick, R.: Algorithms in C. Addison-Wesley (1998) See page 215 of the 8th printing (August 2001) for a remarkably clear and compact integer knapsack solver in C.
32. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979) ISBN 0-7167-1045-5.
33. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Second edn. Dover (1998) ISBN 0-486-40258-4.
34. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a universal test suite for combinatorial auction algorithms. In: ACM E-Commerce. (2000)
35. Chvátal, V.: Hard knapsack problems. Operations Research **28** (1980) 1402–1411
36. Pisinger, D.: Algorithms for Knapsack Problems. PhD thesis, University of Copenhagen (1995) `http://www.diku.dk/users/pisinger/95-1.pdf`.
37. Martello, S., Pisinger, D., Toth, P.: New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operational Research **123** (2000) 325–332
38. Pisinger, D.: A fast algorithm for strongly correlated knapsack problems. Discrete Applied Mathematics **89** (1998) 197–212
39. Klemperer, P.: Auction theory: A guide to the literature. Journal of Economic Surveys **13** (1999) 227–260
40. Krishna, V.: Auction Theory. Academic Press (2002) ISBN 0-12-426297-X.
41. Sandholm, T., Suri, S.: Market clearability. In: IJCAI. (2001)
42. Sandholm, T.: Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence **135** (2002) 1–54
43. Bassamboo, A., Gupta, M., Juneja, S.: Efficient winner-determination techniques for Internet multi-unit auctions. In: Proceedings of the First IFIP Conference on E-Commerce, E-Business, and E-Government. Volume 202. (2001)
44. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall (1993) ISBN 0-13-617549-X.
45. Tennenholtz, M.: Tractable combinatorial auctions and b-matching. Artificial Intelligence **140** (2002) 231–243
46. Ghebreamiak, K.A., Andersson, A.: Caching in multi-unit combinatorial auctions. In: AAMAS. (2002)