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

Terence Kelly
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2004-21
February 15th , 2004*

E-mail: kterence@hpl.hp.com

knapsack problems, combinatorial auctions, dynamic programming, optimization, economics, markets, pricing

The problem of allocating discrete computational resources among agents motivates interest in general multi-unit combinatorial exchanges. This paper considers the problem of computing optimal (surplus-maximizing) allocations, assuming that agent utility functions are quasi-linear but otherwise unrestricted. We present a solver whose time and memory requirements are linear (in the pseudo-polynomial sense) 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.

In this context, 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 encourages a clean separation between E-Commerce research and Operations Research.

* Internal Accession Date Only
Approved for External Publication

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

Terence Kelly
kterence@hpl.hp.com
Hewlett-Packard Laboratories
1501 Page Mill Road m/s 1125
Palo Alto CA 94304 USA

15 February 2004

Abstract

The problem of allocating discrete computational resources among agents motivates interest in general multi-unit combinatorial exchanges. This paper considers the problem of computing optimal (surplus-maximizing) allocations, assuming that agent utility functions are quasi-linear but otherwise unrestricted. We present a solver whose time and memory requirements are linear (in the pseudo-polynomial sense) 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.*

In this context, 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 encourages a clean separation between E-Commerce research and Operations Research.

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 FCC's spectrum allocation problem largely motivated the 1990s surge of CA research [10, 51]. 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 [57].

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 [10, 15, 37]. 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 environments. The number of resource *types* in reasonable formulations of this problem is a *small constant*, whereas the number of *units* of each resource is large and variable. The optimal allocation problem corresponding to WDP is a generalized multi-dimensional knapsack problem (KP): 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 and low dimensionality. If the coefficients describing a problem instance are bounded, 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)

* A shorter but substantively similar version of this paper was submitted to the Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04) on 22 January 2004. Author notifications are due 23 March 2004. http://www.aamas-conference.org/
$Revision: 1.43 $ $Date: 2004/02/16 07:55:51 $

and provide an attractive default solution method for instances where their computational costs are not prohibitive. In all cases they provide a well-understood baseline against which more elaborate methods may be compared.

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 [2] 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 [74]. 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) [69] or Vickrey-Clarke-Groves (VCG) mechanisms [42], 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 [31, 34, 47, 48].

The remainder of this paper is structured as follows: Section 2 motivates interest in low-dimensional MUCAs with a discussion of multi-agent resource allocation in large computing environments. Section 3 formulates our general allocation problem and explains the relationship between auction winner determination and knapsack problems. Section 4 presents a general solver for multi-unit CAs with unrestricted agent preference expression and analyzes its time and memory requirements in detail. Section 5 defines a succinct mode of expression for rational agent preferences in the context of multi-unit CAs, derives a method of generating synthetic test inputs directly from the definition, and relates MUCA inputs to Operations Research literature on hard KP instances. Section 6 reviews related work and Section 7 concludes with a discussion.

2. Motivation: Data Center Allocation

Large tightly-coupled computers remain popular for enterprise computing [19], and today entire data centers comprising large numbers of loosely-coupled hosts are offered as commercial products [20]. 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 [17]. 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 storage arrays [18]. The number of *units* of each resource is typically large and expands with user needs: Modern data centers contain thousands of hosts and their consolidated storage systems comprise comparable numbers of LDEVs.

Applications and the resources allocated to them are partitioned so that an application's performance depends only on the resources it receives; this corresponds to a property sometimes called "no externalities" in auction contexts [46]. 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. These properties suggest that agents will define utility over *bundles* of resources, which in turn 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 small it is reasonable to suppose that each will behave strategically. Incentive-compatible mechanisms (in which truth-telling is a dominant strategy for agents) are therefore desirable, even for allocation within a hierarchical organization [32]. Given that the incentive properties of GVA/VCG mechanisms sometimes require *exact* WDP solvers [31, 34, 47, 48], we prefer exact solvers to approximate ones where possible.

Computational resource allocation can be formalized as a generalized knapsack problem [27]; 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 defines utility over a list of resource bundles. 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 $\vec{q}_{tb} = (q_{1tb},\ldots,q_{Rtb})$ 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:[1]

$$\text{maximize} \quad \sum_{t=1}^{T}\sum_{b=1}^{B_t} x_{tb}u_{tb} \tag{1}$$

$$\text{subject to} \quad \sum_{b=1}^{B_t} x_{tb} = 1 \qquad t = 1,\ldots,T \tag{2}$$

$$\sum_{t=1}^{T}\sum_{b=1}^{B_t} x_{tb}q_{rtb} \le N_r \qquad r = 1,\ldots,R \tag{3}$$

The inequality in Equation 3 permits unallocated goods; to forbid them we simply replace the inequality 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 reduces to classic knapsack problems as special cases [27]. 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 [26]. 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 MUCA WDP solver suitable for low-dimensional MDMCK instances, without providing time or memory complexity analyses and without connecting the WDP to generalized KPs [68].

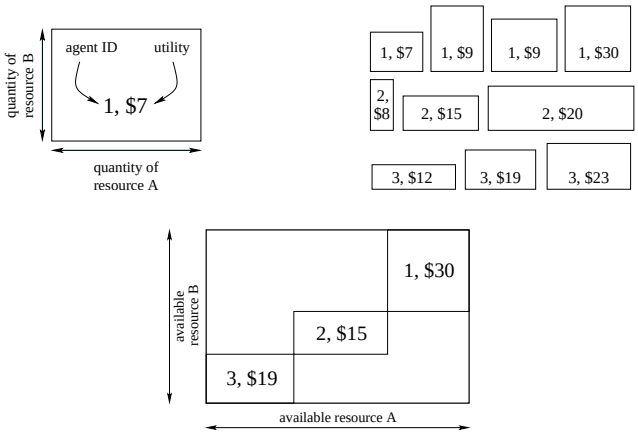

Figure 1. Illustration of 2-D MDMCK. *Top left:* resource bundle. *Top right:* utility functions. *Bottom:* optimal allocation.

Two-resource MDMCK admits a simple graphical interpretation illustrated in Figure 1. A rectangle labeled with an agent ID and a utility value can represent a resource bundle (top left). Utility functions are collections of such rectangles (top right). The allocator's output is illustrated at bottom: exactly one bundle is chosen from each utility function such that utility is maximized while the sum of resource usage does not exceed the resource pool in any dimension.

A note on terminology: Neither MDMCK nor its special cases should be confused with the *multiple knapsack* problem, which involves several containers/resource pools and which we do not consider in this paper. Furthermore note that in MDMCK we do not seek to fill a container in the sense of tiling in two dimensions; the term "multi-dimensional knapsack" has been applied to this very different kind of packing problem [14]. Some authors prefer "multi-constraint" for problems like MDMCK to avoid confusion but "multi-dimensional" appears to be more standard [26, 43].

3.1. Application to Auctions

In an auction setting, we refer to the list of $(\vec{q}_{tb}, u_{tb})$ pairs that an agent submits to the allocator as its *bid*. In general this *reported* utility function may differ from the agent's *true* utility function; we shall ignore the relationship between the two 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

1 The definition here differs slightly from that of earlier work on MDMCK: Whereas Reference [27] requires that *at most one* bundle on each list be chosen, here we generalize the conventional definition of the classic multiple-choice KP and require that *exactly one* bundle be chosen. Otherwise the two formulations are identical. To obtain "at most one bundle" allocation in the present formulation, we simply append a zero-utility null bundle to every agent utility function.

defined by its bid. In other words, we permit "XOR bids," which in turn permit the expression of arbitrary preferences over bundles [46].

The MDMCK formulation requires that each agent's utility depends only on the bundle of resources the agent itself receives ("no externalities" [46]). 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 multi-unit auctions has restricted the expression of agent utility functions, e.g., by requiring that demand be monotonic in per-unit price [29] or that bids be divisible, i.e., "all-or-nothing" bids are forbidden [73]; monotonicity restrictions have also appeared in multi-good CA analyses [33]. In the single-good-type case, bid divisibility is required to ensure that a uniform price exists corresponding to any allocation that maximizes surplus according to agent bids (which is not the same as maximizing surplus, because divisible bids might not represent actual agent utility functions). Uniform prices are sometimes desired, e.g., for reasons of perceived fairness. The real motivation for restrictions on the form of bids, however, has often been to facilitate efficient clearing algorithms [71].

Computational issues aside, the greater generality and flexibility of a MDMCK formulation makes it attractive in auction settings where uniform prices are not required. The components of resource vectors $\vec{q}$ and utilities u may assume both negative and positive values, allowing agents to express willingness to engage in complex *atomic* (all-or-nothing) transactions. Properties including bid divisibility, monotonicity, free disposal, and normalization are permitted but not required. Thus the MDMCK formulation supports very general two-sided multi-unit combinatorial exchanges, e.g., the dozen CA variants considered in Ref. [59].

3.2. Auction and KP Taxonomies

To some extent, relationships between problem families are an aesthetic issue; whether a particular correspondence appears natural or promising is partly in the eye of the beholder. CA WDPs are frequently described as generalizations of set packing, even in the multi-unit case [15]. 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., algorithm implementors in industry; they are intuitive, memorable, and invite simple graphical interpretation (Figure 1). KPs are also far more widely studied in Operations Research. Most importantly, KPs admit pseudo-polynomial solution under restrictions that are acceptable in a wide range of practical situations. Whereas connections with set packing have led to the oversimplified pessimistic view that "CA WDPs are NP hard," the knapsack connection encourages cautious optimism and more balanced and thorough analyses.

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	classic 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) [57]
M	S	M	single-unit CA, XOR bids	reducible 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 [37]	MDMCK [27]

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

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

1. the number of types of goods in an auction, or the dimensionality of a KP;
2. the number of units of each good available, or the capacity of a KP container in each dimension; and
3. the number of bundles over which agents define utility, or 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 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 [37]. 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 [29]. 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 6.2), and efficient specialized solvers are the subject of sophisticated recent research [53]. Holte observes that Operations Researchers have long investigated MDKPs that are substantively identical to multi-unit CA WDPs [22], contrary to claims in recent E-commerce literature that MUCA WDPs were never before studied [37]. Years later, however, MUCA WDP research that cites Holte does not mention the connection he made [35]. 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 [26]. 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 6 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 the MDMCK problem of Section 3; it generalizes multi-dimensional and multiple-choice KP solvers [26, 41, 43].

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

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

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

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

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 $D = -\infty$. $F_T(\vec{N})$ is the value of an optimal solution, and the corresponding choices of bundles may be recovered as $A_T(\vec{N})$, $A_{T-1}(\vec{N} - \vec{q}_{TA_T(\vec{N})})$, etc.; conversion to decision variables x_{tb} of Equations 1 through 3 is trivial.

We may evaluate the dynamic program in two ways: by constructing tables corresponding to $F(\cdot)$ and $A(\cdot)$ in bottom-up fashion, or by recursively evaluating $F_T(\vec{N})$ and $A_T(\vec{N})$. The former strategy yields a full $F_T(\vec{n})$ table containing information about the marginal utilities of every resource type for every resource pool size $\vec{n} : \vec{0} \leq \vec{n} \leq \vec{N}$; this information may be useful for purposes other than allocation, e.g., as a guide for capacity planning. A major 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 $(\hat{t}, \vec{n})$ 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; depending on how it is embellished, it may resemble 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 [63].

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 that the coefficients describing a problem instance (q_{rtb} and u_{tb}) are integers from a *bounded* range. We furthermore assume a bottom-up implementation that stores $F(\cdot)$ and $A(\cdot)$ values in ordinary arrays. The arguments of $F(\cdot)$ and $A(\cdot)$ are vectors whose components may in general exceed the corresponding components of $\vec{N}$, because bundles $\vec{q}$ may include negative components. Furthermore an implementation must store values of $F(\vec{n})$ and $A(\vec{n})$ even for $\vec{n}$ vectors with negative components. Let V_r denote the smallest "width" of our F and A tables in the rth dimension such that these requirements are satisfied. Conservative values for V_r may be computed by a simple pass over the

input; we omit the details. In the special case where no input coefficients are negative, $V_r = N_r + 1$ for all r.

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

It is interesting to note that straightforward DP solutions to the classic 0-1 KP and MCKP problems in the uni-dimensional case have identical asymptotic time complexity in terms of the total number of items that may be packed [41, pages 39 and 78]. In other words, we pay no price for the multiple-choice constraint. The same is true for multi-dimensional knapsack problems in the dynamic programming approach of this paper. In the context of auctions, it has been suggested that XOR constraints over bundles within compound bids be enforced through the use of dummy goods as described in Section 3.2. This trick would be a disaster for our DP solver because it expands the dimensionality of the problem. Fortunately, dummy goods are entirely unnecessary for the algorithm of Equations 4 and 5, which incorporates XOR constraints directly.

The classic 0-1 and integer knapsack problems are NP-hard [11, 49]. Because MDMCK reduces to these as special cases, it too is NP-hard. However knapsack problems are *not* NP-hard in the strong sense, i.e., they admit pseudo-polynomial solution if the coefficients describing instances are bounded, as we have assumed in our analysis. See Papadimitriou & Steiglitz for a good discussion of pseudo-polynomial complexity analysis applied to classic KPs [49]. Restricting agent utilities u_{tb} to the length of modern machine words, e.g., 64 bits, does not seem problematic. Similar observations apply to the q_{rtb} and N_r coefficients.[2]

For classic uni-dimensional problems, branch-and-bound algorithms are often favored over DP *except for hard problem instances*, where DP usually performs better [41, 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 [2] and compute approximate solutions to large MDMCK instances very rapidly [27]. Specialized KP algorithms for the low-dimensional case are the subject of our ongoing research.

2 To put the issue in perspective, note that 48 bits suffices to represent in cents the annual budget of the U.S. Federal Government.

5. Generating Test Instances

The most basic normative assumption that can be made regarding agent preferences is that they are *rational*, i.e., transitive and complete. Merely to express arbitrary rational preferences requires storage exponential in the number of good types, but if this is a small constant the expression and transmission of arbitrary rational preferences is feasible. This section describes a simple and fully general constructive definition of rational agent utility functions with free disposal for the MDMCK problem. Our contribution is to describe how to apply the definition to construct *hard* problem instances, which may complement research toward benchmarks that mimic *typical* CA WDP instances [36].

Let $M(\vec{q})$ be an arbitrary non-negative scalar function defined over resource bundles $\vec{q}$, and let $\vec{e}_i$ denote the unit vector in direction i, i.e., the vector whose ith component is 1 and whose other components are all zero. We define the utility of bundle $\vec{q}$ as

$$u(\vec{q}) \;=\; \begin{cases} u_0 & \vec{q} = \vec{0} \\ M(\vec{q}) + \max_i\{u(\vec{q} - \vec{e}_i)\} & \vec{q} > \vec{0} \end{cases} \qquad (6)$$

The values of u_0 and $M(\vec{e}_i)$ define the utility of a single unit of each good; other values of $M(\vec{q})$ define how utility increases as we move away from the origin $\vec{0}$. To normalize utility we set $u_0 = 0$. The max term in Equation 6 ensures that every bundle is more valuable than all of its proper subsets (free disposal).

5.1. 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 [36], 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 [9]. 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 [52], illustrates four possibilities; Martello *et al.* and Kellerer *et al.* describe others [26, 40]. Strongly-correlated instances are hardest for today's best KP solvers and are the subject of ongoing research [40, 54]. A generalized multi-dimensional form of strongly-correlated instances may be obtained by

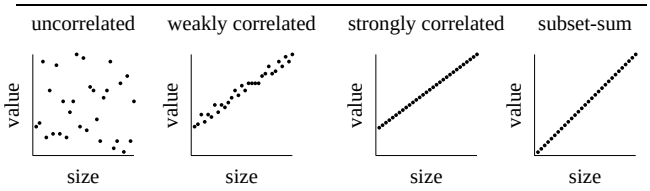

Figure 2. Uni-dimensional KPs.

setting u_0 and $M(\cdot)$ of Equation 6 to strictly positive constants. If random perturbation is added, generalized weakly-correlated instances are obtained. (Little research appears to exist on hard multi-dimensional KPs, and it is not obvious that instances generated in this way are the hardest classes of MDKPs; they are simply a natural starting point for an investigation of hard MDKPs.)

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 [40]. A similar pattern is evident in evaluations of WDP solvers many years later, as Andersson *et al.* have noted [2]; see also Section 6.2. 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.

6. Related Work

The literature on knapsack problems is vast and growing; due to space restrictions we shall cite only a handful of recent surveys. An excellent text by Martello & Toth [41] is now out of print, but a very recent book by Kellerer *et al.* provides an updated and much more comprehensive treatment of the field, including multi-dimensional problems and MDMCK itself [26]. MDKP is routinely treated in OR texts, e.g., Minoux [43]. Martello *et al.* review recent research on exact solutions for large hard instances of 0-1 KP [40]. Pisinger's doctoral dissertation [52] summarizes the state of the art in uni-dimensional KP research c. 1995, much of which is directly applicable to WDPs in single-good multi-unit auctions that attracted attention in subsequent years [29, 73].

6.1. 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 [10, 28, 30, 51, 57, 59] do not discuss knapsack problems, although some describe connections to generalized set packing. The string "knap" appears in exactly five papers among all past proceedings of the ACM Conference on E-Commerce (EC). Two mention in passing a relationship between special cases of WDP and KPs [15, 29], one uses reduction from KP to prove NP-hardness [33], and the remainder are unrelated to our interests [7, 76].

Source	Documents	"auction"	"knapsack"	both
Springer Link	?	152	103	zero
IEEE Xplore	990,765	313	150	zero
ACM Digital Lib	125,779	802	427	10
CiteSeer	?	1,686	922	12
Sci. Cit'n Index	33,117,604	2,379	989	zero
Elsevier Sci Direct	"over 4M"	5,143	2,084	11

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

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, and the documents at the intersection of the two keywords contained no detailed or systematic treatment of the WDP-KP connection. A few papers mention in passing a deep relationship between WDPs and KPs [8, 22], and a handful casually state that the connection is well known, without saying by whom [3, 24, 75]. For completeness we include all of the "auction AND knapsack" results: Ten ACM Digital Library hits [4, 15, 29, 33, 44, 50, 55, 65, 66, 77]; twelve CiteSeer hits [3, 4, 8, 16, 22, 24, 29, 38, 45, 55, 56, 58]; and eleven Science Direct hits [13, 23, 25, 39, 60, 61, 64, 67, 70, 72, 75].

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 [26, pp. 478–482]. Because their main interest is in knapsack problems rather than auctions, Kellerer *et al.* do not explicitly compare taxonomies of auction WDPs and KPs. It is but a short step, however, from their discussion to the correspondences made explicit in Section 3.2. (The present paper was written and submitted for publication before Kellerer *et al.*'s excellent text became available in the United States.)

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.

6.2. 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 [29]. 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 [41, 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 $0,\ldots,N$, the (pseudo-polynomial) time and memory requirements of this very simple exact method are respectively $O(TN^2)$ and $O(TN)$. 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 [6]. 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 [68]. 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 [1, p. 100] are mentioned, nor are time and memory complexity analyses presented.

WDP solver research for multi-good-type/multi-unit CAs has emphasized heuristic branch-and-bound algorithms [15, 37]. 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 [37]. 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 [15, 37, 59], which produce multi-dimensional variants of the uncorrelated and weakly correlated cases of Figure 2; for uni-dimensional KPs, these are not hard instances.

	single-unit/ high-dimensional	multi-unit/ low-dimensional
practical motivation	spectrum auctions	computational resource alloc'n
# good types	variable, high	low, fixed
# units/type	fixed at 1	variable, high
WDP	weighted set packing	generalized KP
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

Table 3. Trajectories of CA research.

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") [35, 37]. In fact, this subroutine implements the classic *pseudo*-polynomial DP algorithm for the NP-hard 0-1 knapsack problem.

7. 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 computational resource allocation in modern data centers, which in some formulations involves few types of goods but many units of each. Whereas comparisons with set packing prob-

lems 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 [12]. 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

Extensive discussions with Daniel Reeves and Professors Michael Wellman and Jeff Mason at the University of Michigan improved this research enormously. Moises Goldszmidt reviewed an early draft and provided valuable feedback. David Pisinger answered several questions about knapsack problems and their literature promptly and in detail. Kevin Leyton-Brown and Bill Walsh answered many questions about their respective research. Leyton-Brown, Walsh, Daniel Lehmann and Yoav Shoham provided pointers to literature on the need for exact optimization solvers in incentive-compatible mechanisms. Shoham and his Stanford University research group provided feedback; detailed written comments from Eugene Nudelman and Ryan Porter were particularly helpful. Wei Deng assisted with the literature search. The Technical Publications department of Hewlett-Packard Labs halted publication of this report so that a review of Kellerer *et al.* [26] could be included.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, 1993. ISBN 0-13-617549-X.

[2] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, pages 39–46, July 2000. `http://www.computer.org/proceedings/icmas/0625/06250039abs.htm`.

[3] Amitabha Bagchi, Amitabh Chaudhary, Rahul Garg, Michael T. Goodrich, and Vijay Kumar. Seller-focused algorithms for online auctioning. In F. Dehne, J.-R. Sack, and R. Tamassia, editors, *Lecture Notes in Computer Science*, volume 2125, chapter 4b, pages 135–147. Springer, January 2001. Originally appeared in 7th International Workshop on Algorithms & Data Structures (WADS 2001).

[4] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, September 2001.

[5] Achal Bassamboo, Manish Gupta, and Sandeep Juneja. Efficient winner determination techniques for internet single item multi-unit open-cry auctions. Technical Report RI 00027, IBM, December 2000. A later version is Ref. [6].

[6] Achal Bassamboo, Manish Gupta, and Sandeep Juneja. 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, October 2001. Proceedings available as Ref. [62]. Draft dated 28 February 2002 available at `http://www.tcs.tifr.res.in/~sandeepj/avail_papers/ifip.ps`. An earlier version is Ref. [5].

[7] Yuan-Chi Chang, Chung-Sheng Li, and John R. Smith. Searching dynamically bundled goods with pairwise relations. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 135–143, June 2003.

[8] Chunming Chen, Muthucumaru Maheswaran, and Michel Toulouse. Supporting co-allocation in an auctioning-based resource allocator for grid systems. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2002)*, pages 89–96, April 2002.

[9] Vasek Chvátal. Hard knapsack problems. *Operations Research*, 28(6):1402–1411, November 1980.

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

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979. ISBN 0-7167-1045-5.

[12] Kidane Asrat Ghebreamiak and Arne Andersson. Caching in multi-unit combinatorial auctions. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 164–165. ACM Press, 2002. ISBN 1-58113-480-0.

[13] M. Ghiassi and C. Spera. Defi ning the internet-based supply chain system for mass customized markets. *Computers & Industrial Engineering*, 45(1):17–41, June 2003.

[14] P. C. Gilmore and R. E. Gomory. The theory and computation of knapsack functions. *Operations Research*, 14:1045–1074, 1966.

[15] Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 13–20, October 2000.

[16] John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth? In *IEEE Symposium on Foundations of Computer Science*, pages 252–259, October 2001.

[17] Hewlett-Packard Corporation. An economy of IT: Allocating resources in the computing utility, October 2003. `http://www.hpl.hp.com/news/2003/oct_dec/computons.html`.

[18] Hewlett-Packard Corporation. hp StorageWorks disk array xp512, September 2003. `http://www.hp.com/products1/storage/products/disk_arrays/highend/xp512/`.

[19] Hewlett-Packard Corporation. HP Integrity Superdome servers, January 2004. `http://www.hp.com/products1/servers/integrity/superdome_high_end/`.

[20] Hewlett-Packard Corporation. HP Utility Data Center (UDC) overview, January 2004. `http://h30046.www3.hp.com/solutions/overview.html`.

[21] Robert C. Holte. Bibliography of combinatorial auctions and knapsack problems, June 2001. `http://www.cs.ualberta.ca/~holte/CombinatorialAuctions/cabib.bib`.

[22] Robert C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In E. Stroulia and S. Matwin, editors, *Lecture Notes in Computer Science*, volume 2056, pages 57–66. Springer, January 2001. See Ref. [21] for an extensive bibliography on knapsack problems and auctions.

[23] Srinivasan Jagannathan, Jayanth Nayak, Kevin Almeroth, and Markus Hofmann. On pricing algorithms for batched content delivery systems. *Electronic Commerce Research and Applications*, 1(3–4):264–280, 2002.

[24] Joni L. Jones and Gary J. Koehler. An allocation heuristic for combinatorial auctions, February 2001. This appears to be an unpublished draft or a paper under review; related papers are available at the fi rst author's Web site at the University of South Florida.

[25] Joni L. Jones and Gary J. Koehler. Combinatorial auctions using rule-based bids. *Decision Support Systems*, 34(1):59–74, December 2002.

[26] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004. ISBN 3-540-40286-1. This book considers a wide range of KP variants, including MDMCK. It also discusses in some depth the relationship between multi-dimensional KPs and winner determination in multi-unit CAs.

[27] Terence Kelly. Utility-directed allocation. In *First Workshop on Algorithms and Architectures for Self-Managing Systems*, July 2003. `http://tesla.hpl.hp.com/self-manage03/`. Also available as HP Labs tech report HPL-2003-115.

[28] Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13(3):227–260, July 1999.

[29] Anshul Kothari, David C. Parkes, and Subhash Suri. Approximately-strategyproof and tractable multi-unit auctions. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 166–175, June 2003.

[30] Vijay Krishna. *Auction Theory*. Academic Press, 2002. ISBN 0-12-426297-X.

[31] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions (extended abstract). In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[32] John O. Ledyard. Incentive compatible space station pricing. *American Economic Review*, 76:274–279, May 1987.

[33] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 18–28, October 2001.

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

[35] Kevin Leyton-Brown. *Resource Allocation in Competitive Multiagent Systems*. PhD thesis, Stanford University, August 2003.

[36] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the Second ACM Conference on Electronic Commerce*, pages 66–76, October 2000.

[37] Kevin Leyton-Brown, Yoav Shoham, and Moshe Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, July 2000.

[38] Grace Y. Lin, Yingdong Lu, and David D. Yao. The stochastic knapsack revisited: Structure, switch-over policies, and dynamic pricing. `columbia.edu/~yao/knaps7.pdf`.

[39] Fiona Mackenzie. Exploring the connections: Structural adjustment, gender and the environment. *Geoforum*, 24(1):71–87, February 1993.

[40] Silvano Martello, David Pisinger, and Paolo Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123(2):325–332, June 2000.

[41] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementation*. John Wiley & Sons Ltd., 1990. ISBN 0-471-92420-2. Unfortunately this classic is out of print. See [26] for an updated and very comprehensive treatment of the subject.

[42] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995. ISBN 0-19-507340-1.

[43] Michel Minoux. *Mathematical Programming*. Wiley, 1986. Translated from a French edition of 1983. ISBN 0-471-90170-9.

[44] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions (extended abstract). In *Eighteenth National Conference on Artificial Intelligence*, pages 379–384, 2002.

[45] Chaki Ng, David C. Parkes, and Margo Seltzer. Virtual worlds: Fast and strategyproof auctions for dynamic resource allocation (poster presentation). In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 238–239, June 2003. Discussion of knapsack problems in extended version at `http://www.eecs.harvard.edu/econcs/pubs/virtual.pdf`.

[46] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the Second ACM Conference on Electronic Commerce*, pages 1–12, October 2000.

[47] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 242–252, October 2000.

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

[49] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, second edition, 1998. ISBN 0-486-40258-4.

[50] David C. Parkes. Conference report: The third ACM conference on electronic commerce. *ACM SIGecom Exchanges*, 3(1):57–61, 2002.

[51] Aleksandar Pekec and Michael H. Rothkopf. Combinatorial auction design. *Management Science*, 49(11):1485–1503, November 2003.

[52] David Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, February 1995. `http://www.diku.dk/users/pisinger/95-1.pdf`.

[53] David Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operations Research*, 83:394–410, 1995.

[54] David Pisinger. A fast algorithm for strongly correlated knapsack problems. *Discrete Applied Mathematics*, 89(1–3):197–212, December 1998.

[55] Paul S.A. Reitsma, Peter Stone, Janos A. Csirik, and Michael L. Littman. Randomized strategic demand reduction: Getting more by asking for less. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 162–163, July 2002.

[56] Paul S.A. Reitsma, Peter Stone, Janos A. Csirik, and Michael L. Littman. Self-enforcing strategic demand reduction. In J. Padget, O. Shehory, D. Parkes, N. Sadeh, and W.E. Walsh, editors, *Lecture Notes in Computer Science*, volume 2531, pages 289–306. Springer, January 2002.

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

[58] Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. In *IJCAI 2001 Workshop on Distributed Constraint Reasoning*, August 2001.

[59] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 69–76. ACM Press, 2002. ISBN 1-58113-480-0.

[60] Jayaram K. Sankaran. Column generation applied to linear programs in course registration. *European Journal of Operational Research*, 87(2):328–342, December 1995.

[61] Jayaram K. Sankaran. A monotonic, dual-based bounding procedure for integer programs. *Computers & Operations Research*, 22(5):491–501, May 1995.

[62] Beat Schmid, Katarina Stanoevska-Slabeva, and Volker Tschammer, editors. *Towards The E-Society: E-Commerce, E-Business, and E-Government, The First IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2001), October 3-5, Zürich, Switzerland*, volume 202 of *IFIP Conference Proceedings*. Kluwer, 2001. ISBN 0-7923-7529-7. Available for purchase at `http://www.wkap.nl/prod/b/0-7923-7529-7`.

[63] Robert Sedgewick. *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.

[64] Maiko Shigeno, Yasufumi Saruwatari, and Tomomi Matsui. An algorithm for fractional assignment problems. *Discrete Applied Mathematics*, 56(2–3):333–343, January 1995.

[65] Narayanan Shivakumar, Jan Jannik, and Jennifer Widom. Per-user profile replication in mobile environments: Algorithms, analysis, and simulation results. *Mobile Networks and Applications*, 2(2):129–140, October 1997.

[66] Narayanan Shivakumar and Jennifer Widom. User profile replication for faster location lookup in mobile environments. In *Proceedings of the First Annual International Conference on Mobile Computing and Networking*, pages 161–169, December 1995.

[67] Jeffrey E. Teich, Hannele Wallenius, Jyrki Wallenius, and Otto R. Koppius. Emerging multiple issue e-auctions. *European Journal of Operational Research*, 2004. In press as of late December 2003.

[68] Moshe Tennenholtz. Some tractable combinatorial auctions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, July 2000.

[69] Hal Varian and Jeffrey K. MacKie-Mason. Generalized Vickrey auctions. Technical report, Dept. of Economics, University of Michigan, July 1994.

[70] Jose A. Ventura and Sanjay Radhakrishnan. Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research*, 144(3):598–612, February 2003.

[71] Bill Walsh. Personal communication, February 2003.

[72] D. J. Wu. Software agents for knowledge management: coordination in multi-agent supply chains and auctions. *Expert Systems with Applications*, 20(1):51–64, January 2001.

[73] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27, 1998.

[74] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35:304–338, 2001.

[75] Mu Xia, Gary J. Koehler, and Andrew B. Whinston. Pricing combinatorial auctions. *European Journal of Operational Research*, 154(1):251–270, April 2004.

[76] Daniel D. Zeng, Fei-Yue Wang, and Sudha Ram. Storage allocation in Web prefetching techniques. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 264–265, June 2003.

[77] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Web engineering: Quality driven Web services composition. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 411–421, May 2003.