Generating k Best Solutions to Winner Determination Problems: Algorithms & Application to Procurement

Terence Kelly, Andrew Byde
HP Laboratories Palo Alto

HPL-2006-40
March 3, 2006*

auctions, procurement, knapsack problems, k-shortest paths, decision support, preference elicitation

Auction participants often cannot easily articulate their requirements and preferences. The buyer in a procurement auction, for instance, may hesitate to quantify the value of non-price solution attributes, and she may have difficulty delineating between hard and soft constraints. Consequently it can be difficult to formulate the winner determination problem (WDP) as a straightforward optimization problem. Existing decision-support aids for such situations, including scenario navigation and preference elicitation, address this difficulty through extensions to an optimization framework.

This paper presents a complementary approach that frames the procurement decision problem as one of *exploration* rather than optimization. The foundation of our approach is an algorithm that generates *k* best solutions to auction WDPs. Our algorithm can scale to practical problem sizes for an interesting class of procurement auctions, and furthermore can incorporate hard constraints into the generation process. We describe ways of extracting useful guidance for the decision-maker from *k*-cheapest WDP solutions. We evaluate our method using real bids submitted by real suppliers in an HP material parts procurement auction.

Generating k Best Solutions to Winner Determination Problems: Algorithms & Application to Procurement

Terence Kelly
Hewlett-Packard Laboratories
Palo Alto, California, USA
terence.p.kelly@hp.com

Andrew Byde
Hewlett-Packard Laboratories
Bristol, UK
andrew.byde@hp.com

ABSTRACT

Auction participants often cannot easily articulate their requirements and preferences. The buyer in a procurement auction, for instance, may hesitate to quantify the value of non-price solution attributes, and she may have difficulty delineating between hard and soft constraints. Consequently it can be difficult to formulate the winner determination problem (WDP) as a straightforward optimization problem. Existing decision-support aids for such situations, including scenario navigation and preference elicitation, address this difficulty through extensions to an optimization framework.

This paper presents a complementary approach that frames the procurement decision problem as one of *exploration* rather than optimization. The foundation of our approach is an algorithm that generates *k* best solutions to auction WDPs. Our algorithm can scale to practical problem sizes for an interesting class of procurement auctions, and furthermore can incorporate hard constraints into the generation process. We describe ways of extracting useful guidance for the decision-maker from *k*-cheapest WDP solutions. We evaluate our method using real bids submitted by real suppliers in an HP material parts procurement auction.

1. INTRODUCTION

Goodness is easy to recognize but hard to define.
— W. H. Auden [3]

Winner determination problems (WDPs) in auctions are *computationally* difficult for many interesting auction types, e.g., combinatorial auctions [19]. In practice, WDPs can furthermore pose *cognitive* challenges to participants. For instance, agents may be unsure of the value they place on non-price attributes of solutions, or unsure of whether to express various business considerations in the objective function or constraints of a formal optimization problem. These issues can be vexing even in single-agent decision problems such as that faced by a bid-taking buyer in a reverse (procurement) auction. The buyer's preferences and constraints may be so difficult to articulate as to preclude even the formulation of the WDP as a proper optimization problem. In many such situations the buyer essentially reports, "I know a good solution when I see one," but can't precisely and explicitly state the *properties* of a good solution.

Document date: March 2, 2006 Document version:
$Id: bgk.tex,v 1.61.1.7 2006/03/03 03:20:56 kterence Exp $

Two existing decision-support techniques can be applied in such cases. The buyer in a reverse auction may resort to *scenario navigation*, repeatedly running a price-optimizing WDP solver with different sets of constraints in the hope of finding an attractive solution. More sophisticated *preference elicitation* methods attempt to assist or even automate scenario navigation by allowing the expression of imprecise preferences over non-price allocation features and/or by systematically interrogating the decision-maker to progressively refine a model of her latent utility function [6]. Preference elicitation offers numerous advantages over ad hoc scenario navigation, but it is not without shortcomings: It typically imposes strong assumptions on the functional form of preferences, it can require an excessive number of queries, and revealed preferences can be intransitive or otherwise problematic [7, 20].

This paper presents a very different complementary approach to decision support in auctions. The foundation of our method is an algorithm that generates *k* best solutions to an auction WDP in descending order of objective function quality. In the most general case, our algorithm can generate solutions that maximize gains from trade in a combinatorial exchange, albeit with limited scalability. When specialized for reverse auctions, our method can scale to practical problem sizes. Our method supports multi-sourcing of items, and bids may express volume discounts and volume surcharges. Hard constraints can be incorporated to prevent unacceptable solutions from being generated.

Once computed, the table of *k*-cheapest solutions to a procurement auction can be post-processed in a variety of ways to aid the decision-maker. If the buyer defines ordinal preferences over solution features, the Pareto frontier of undominated solutions may be computed. More importantly, the table of solutions can be used to associate prices with bundles of constraints, which can focus the buyer's attention on the most pressing tradeoffs in a WDP instance. This in turn can inform decisions about which considerations are best regarded as constraints vs. aspects of the objective function in an optimization formulation of the WDP. Finally, we show how the *k*-cheapest solutions table supports a variety of informative visualizations.

In summary, the main contribution of this paper is an algorithm that allows us to cast the procurement decision problem as one of *exploration* rather than *optimization*. Both scenario navigation and preference elicitation are grounded in optimization: They use samples of the decision-maker's constraints or estimates of her preferences and they seek to compute a single optimal solution. By contrast, we require only seller bids and we systematically extract a large set of candidates from the most promising region of the solution space: the solutions that entail minimal expense. The table of *k* best solutions allows us to leverage a wide range of existing analysis methods including clustering, data visualization, and dominance pruning. In addition to revealing satisfactory WDP solutions, such exploration can yield valuable qualitative as well as quantitative insight into the cost of constraints and the nature of the competitive landscape. "Mining" large databases of candidate solutions is an increasingly popular paradigm in automated design [23], and our algorithm opens the possibility of applying similar strategies to procurement decision-making.

2. GENERAL APPROACH

This section describes how to generate *k* best solutions to any combinatorial auction or exchange. The scalability of this method is limited in the most general case. However the general method is easy to explain and it illustrates principles that we later exploit in a specialized variant that scales to procurement auctions of practical size. Our approach follows from four observations:

1. The WDP in sealed-bid combinatorial auctions and exchanges is a generalized knapsack problem [15].
2. Dynamic programming can solve such problems [14].
3. Any dynamic program can be expressed as a longest- or shortest-path graph search problem [1].
4. We can generate the *k* shortest paths in a graph [10].

Individually, these observations are well known. However to the best of our knowledge they have never been composed to yield a general method for computing *k* best solutions to any sealed-bid combinatorial auction or exchange. The method proceeds as follows: We express the dynamic program corresponding to the auction WDP as a shortest- or longest-paths problem on a graph in which path length corresponds to the value of our objective function (e.g., aggregate surplus in an exchange or buyer expenditure in a reverse auction). We then apply a suitable *k*-shortest paths algorithm to generate a table of *k* best solutions to the WDP.

The table of *k* best WDP solutions then invites a wide range of analyses. For instance, if ordinal preferences are defined over solution attributes, we might eliminate dominated solutions to obtain solutions on the Pareto frontier. Furthermore, the table implicitly *defines prices on bundles of constraints*: For any bundle of constraints that is satisfied by a solution in the *k*-best table, its price is simply the difference in objective function value between the best solution that satisfies the constraints and the best unconstrained solution. Note that we impose no restrictions whatsoever on the mathematical form of constraints; in particular, arbitrary non-linear constraints pose no special difficulties. Finally, a number of informative decision aids and visualizations can be extracted from the table; Section 5 presents several examples. Of course, the *k* best solutions are useful only to the extent that they differ in "interesting" ways; solution diversity is an empirical question considered in Section 5.

Computing *k*-best solutions is obviously no easier than computing the first-best solution, so the inherent computational difficulty of WDPs applies to the general method sketched in this section. The WDP of a multi-unit combinatorial auction or exchange is a multidimensional multiple-choice knapsack problem. If the dimensionality of the problem is small—i.e., if the number of *types* of goods is small—dynamic programming solvers with modest pseudo-polynomial time and memory requirements are available [15]. Such solvers are usable in many practical low-dimensional WDPs, and they may be extended to compute *k* best solutions using the approach of this section. However if the number of types of goods is large, merely to find the *first*-best solution is computationally prohibitive: The WDP of most combinatorial auction variants is NP-hard [19] and approximation results for multidimensional knapsack problems are not encouraging [14].

In summary, while it is conceptually straightforward to compute *k* best solutions to the most general sealed-bid combinatorial exchanges, in practice this is computationally infeasible for auctions involving many types of goods. To scale in the number of good types we must restrict the auction. The remainder of this paper considers a class of procurement auctions and presents a *k*-best solutions algorithm that can scale to important real-world problems.

3. PROCUREMENT AUCTIONS

Businesses increasingly rely on procurement auctions to obtain needed goods, e.g., material inputs to manufacturing processes. Hewlett-Packard alone has spent roughly \$21 billion over the past four years via online procurement auctions, and US firms will collectively spend hundreds of billions of dollars through such auctions in 2006 [4]. Winner determination in procurement auctions can be conceptually trivial and computationally easy if the buyer seeks only to obtain goods at minimal overall cost. However, sophisticated decision support is needed in practice because buyer preferences typically encompass non-price solution attributes and because side constraints are often present [6]. For example, the buyer might

1. constrain the number of sellers included in the solution;
2. insist on between two and four suppliers for each item;
3. impose "XOR" constraints across winners, e.g., if supplier A is chosen, then supplier B must be excluded;
4. wish to spread expenditure evenly across sellers, in order to preserve diversity in the marketplace or to maintain a perception of fairness.

Furthermore, any of these constraints may be "soft" in the sense that the buyer would willingly relax them in exchange for sufficiently large savings.

This section presents a *k*-best-solutions algorithm for procurement auctions with a single buyer, multiple sellers, and multiple items (types of goods) available in multiple units. Seller bids may encode volume discounts and volume surcharges. Multi-sourcing is permitted, i.e., the buyer may obtain units of a single item from multiple suppliers, and the buyer may constrain multi-sourcing on individual items before *k* best solutions are generated. We present computational complexity analyses showing that our technique scales well in terms of problem size parameters. Section 4 extends our algorithm to include *global* constraints on solutions, and Section 5 applies the algorithm to real bids from an actual material-parts procurement auction.

3.1 Definitions and Notation

Let I denote the number of *items* (distinct types of goods) that the buyer wishes to acquire; we may think of the overall procurement auction as consisting of I sub-auctions that are to be cleared simultaneously. Global granularity parameter Q specifies the number of *quantiles* (shares of an item)

	item i_1		item i_2		item i_3	
seller	50%	100%	50%	100%	50%	100%
s_A	\$3	\$6	\$4	\$7	\$5	\$11
s_B	\$2	\$7	\$5	\$8	\$4	\$10

Table 1: Bids in example problem.

that bids offer to supply. If $Q = 4$, for instance, then bids offer to supply 25%, 50%, 75%, or 100% of the total number of demanded units of each item. Let S denote the number of sellers. For each item i, seller s submits a bid B_{is} that is a list of (q, p) pairs, where q is a quantity in the range $1 \ldots Q$ and p is the payment that the seller requires for supplying q/Q of the buyer's demand for item i. In an *acceptable* solution to the auction WDP the buyer obtains exactly Q units of each item.

The practical problems that motivate our work typically involve well under a dozen sellers but can include scores of items. Small values of Q, e.g., $Q = 4$ quantiles, permit sufficiently expressive bids in most cases. We show that these parameter values imply that the number of acceptable solutions to each individual-item sub-auction is small enough to consider exhaustively. However, the number of acceptable solutions to the *overall* procurement auction is far too large to generate. Therefore we generate all acceptable solutions to individual-item sub-auctions, but generate only the k best acceptable solutions to the overall auction using a specialized k-shortest paths algorithm. Before describing the algorithms respectively used for these purposes in Sections 3.3 and 3.4, we introduce an example used throughout the remainder of the paper.

3.2 Example Problem

Consider a procurement auction with $I = 3$ items (i_1, i_2, and i_3) and $S = 2$ sellers (s_A and s_B). Bid granularity parameter $Q = 2$, so sellers may supply 0%, 50%, or 100% of the quantity of each item that the seller wishes to acquire. Table 1 shows the bids in our example. Seller s_A offers a volume discount on item i_2 and imposes a volume surcharge on i_3. Seller s_B imposes a volume surcharge on items i_1 and i_3 but offers a volume discount on i_2.

The number of acceptable solutions to each sub-auction is obviously three: The buyer may acquire 100% from s_A, 100% from s_B, or 50% from each. Three possible solutions in each of three sub-auctions implies a total of $3^3 = 27$ acceptable solutions to the overall clearing problem. Table 2, computed using our prototype solver, lists these solutions and the total payment associated with each. In the table, solutions to sub-auctions are encoded as two-character strings where "AA" means that seller s_A supplied 100% of the item, "AB" means that each seller supplied 50%, etc. The best solution, AB AA AB, costs \$21. However, if we must satisfy a side constraint that seller s_B must supply at least 50% of each item, then the best solution is AB BB AB and costs \$22. We therefore say that the price of this side constraint is \$1. If we further require that seller s_B must supply all of item i_1, then the best solution (BB BB BB) costs \$25 and the price of both constraints together is \$4.

3.3 Generating Individual-Item Outcomes

The number of acceptable solutions to each sub-auction is given by a classic occupancy problem: How many ways can Q indistinguishable balls (representing quantiles) be placed in S distinguishable boxes (representing sellers)? Formally, what

auction i_1 i_2 i_3	$\sum p$ (\$)	auction i_1 i_2 i_3	$\sum p$ (\$)	auction i_1 i_2 i_3	$\sum p$ (\$)
AA AA AA	24	AB AA AA	23	BB AA AA	25
AA AA AB	22	AB AA AB	21	BB AA AB	23
AA AA BB	23	AB AA BB	22	BB AA BB	24
AA AB AA	26	AB AB AA	25	BB AB AA	27
AA AB AB	24	AB AB AB	23	BB AB AB	25
AA AB BB	25	AB AB BB	24	BB AB BB	26
AA BB AA	25	AB BB AA	24	BB BB AA	26
AA BB AB	23	AB BB AB	22	BB BB AB	24
AA BB BB	24	AB BB BB	23	BB BB BB	25

Table 2: Solutions in example problem.

are the solutions to $q_1 + q_2 + \cdots + q_S = Q$ where all q_s are non-negative integers? The *number* of solutions is given by the multiset formula (see [11, p. 38] or [5, p. 32]):

$$R(S,Q) \quad = \quad \left(\begin{array}{c} Q+S-1 \\ Q \end{array} \right) \quad = \quad \frac{(Q+S-1)!}{Q!(S-1)!} \qquad (1)$$

For the example problem of Section 3.2, $R(S = 2, Q = 2) = 3$.

For problem sizes of practical interest, $R(S,Q)$ is remarkably small, e.g., $R(12,10) = 352,716$. By contrast, the number of solutions including unacceptable ones is $(Q+1)^S$, which is far larger. Therefore in order to generate all acceptable solutions we cannot employ the naïve expedient of generating all solutions and eliminating the unacceptable ones.

We generate all acceptable solutions to individual-item sub-auctions using a recursive algorithm, IISAGEN, presented in pseudocode in Figure 1. It generates exactly those acceptable solutions enumerated by the multiset formula of Equation 1, i.e., it fills the output table `solutions` with the $R(S,Q)$ ways of allocating Q quantiles among S sellers. Note that the algorithm assumes that seller IDs range from zero to $S-1$. The meaning of an (initial or recursive) invocation "`iisagen(q, s)`" is: "generate all ways of allocating `q` remaining quantiles among sellers zero through `s`, and store them starting in element `solno` of the `solutions[]` table." Alternative generation methods are available; see the discussions of compositions and multicombinations in Knuth's recent volume on combinatorial generation [16, 17].

The asymptotic time and memory requirements of IISAGEN as presented in Figure 1 are both $O(R(S,Q))$. We could easily eliminate the storage requirement entirely and simply output solutions as they are completed. Our implementation internally stores an array of individual-item sub-auction solutions because the array is used by the k-shortest paths algorithm described below. Our ongoing research is developing more sophisticated and more computationally efficient ways of incorporating individual-item outcomes into the process of generating k-best solutions to the overall auction.

3.4 Generating k Best Overall Solutions

The number of acceptable solutions to our *overall* procurement auction WDP is $R(S,Q)^I$, which is far too large to generate exhaustively for problems of practical size. This section describes how we *selectively* generate overall solutions in ascending order of expense to the buyer.

We construct a *solutions graph* in which paths correspond to acceptable solutions to our overall WDP; Figure 2 illustrates the construction for the example of Section 3.2. We introduce source and destination nodes s and t and intermediate dummy nodes $d_1, d_2, \ldots, d_{I-1}$. Directed edges from s to d_1 represent acceptable solutions to the sub-auction for item i_1, edges from

```
input:S: integer; number of sellers
      Q: integer; number of quantiles
output:  solutions[][]:  2-D array of ints;
         element [n][s] contains number of
         quantiles supplied by seller s in
         the nth solution
global variables:
   solno = 0:  # of solutions generated so far
   qstk[]:  array of ints, initially zero; holds
            partial solution during search
solver function:
   iisagen(qty:  integer, seller:  integer) {
      local variable:  q:  integer
      if (qty = 0 OR seller = 0) {
         local variable:  s:  integer
         if (seller = 0)
            qstk[seller] ← qty
         for (s ← 0; s < S; s++)
            solutions[solno][s] ← qstk[s]
         solno++
      }
      else {
         for (q ← 0; q ≤ qty; q++) {
            qstk[seller] ← q
            iisagen(qty - q, seller - 1)
            qstk[seller] ← 0
         }
      }
   }
solver invocation:
   iisagen(Q, S-1)
```

Figure 1: Algorithm IISAGEN for generating all solutions to an individual-item sub-auction.

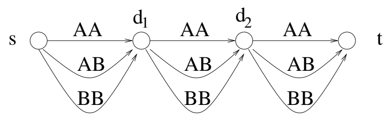

Figure 2: Solutions graph for example of Section 3.2.

d_1 to d_2 represent solutions to the second sub-auction, etc. In the figure, edges are labeled as in Table 2. Edge weights (not shown) represent the cost of the corresponding sub-auction solution, e.g., edge "AA" from s to d_1 has weight \$6 from Table 1. There is a one-to-one correspondence between paths from s to t in the solutions graph and acceptable solutions to the overall WDP, and path lengths correspond to total buyer expenditure. The problem of generating k-cheapest acceptable solutions to the procurement auction WDP then becomes the problem of computing k-shortest paths in the solutions graph. We avoid much of the complexity of published k-shortest paths algorithms because our solutions graph is acyclic and we can safely ignore the possibility of cyclic paths during the generation process.

The foundation of our k-shortest paths algorithm is a memory-efficient representation of solution graph paths as a length-ordered heap. This representation is based on the concept of *deviations*, first introduced by Hoffman & Pavley [12]. First, let a shortest-paths tree be superimposed on the graph.

DEFINITION 3.1 (HOFFMAN & PAVLEY [12]). *A **deviation** from a path p is a path p', having the same origin and destination as p, which is initially part of the shortest-paths tree, which then contains exactly one link, called the deviation link, which is not a link of p, but whose terminal node is the terminal node of a link of p. The final portion of p' coincides with p.*

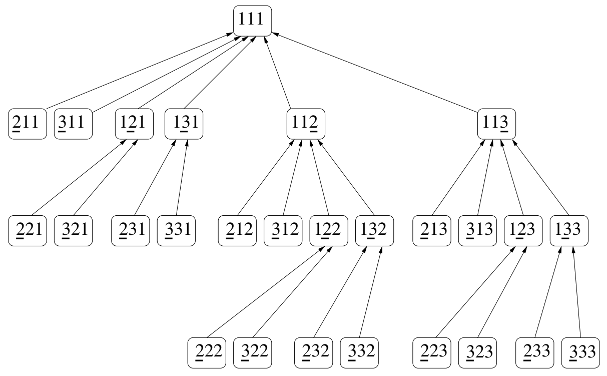

Figure 3: Heap ordering of paths from example problem.

```
 1: compute tree of shortest paths
 2: insert shortest path into PQ
 3: while fewer than k paths have been generated
 4:     p = extract-min(PQ)  /* next shortest path */
 5:     if p is NULL
 6:         terminate  /* fewer than k paths exist */
 7:     output p
 8:     for each deviation d of path p
 9:         if PQ contains fewer than k paths
10:             insert d into PQ
11:         else  /* PQ full, w/ exactly k paths */
12:             if d is longer than longest path in PQ
13:                 discard d
14:             else
15:                 remove & discard longest path in PQ
16:                 insert d into PQ
```

Figure 4: The SKSP algorithm. Priority queue PQ holds paths, ordered by length.

Because edge weights are non-negative, the length of a deviation p' is not less than the length of its "parent" path p. Furthermore any path in a graph is uniquely defined by a parent and a deviation edge (the parent of the shortest path is null). We can therefore represent all paths in our solution graph as a heap-ordered tree in which the shortest path is the root, and the children of each node are its deviations.

Figure 3 illustrates the heap of paths for our example problem. Node numbers represent the rank of the solutions chosen for individual-item sub-auctions, e.g., "312" is the path corresponding to the third-cheapest solution to auction 1, the cheapest solution to auction 2, and the second-cheapest solution to auction 3. Underscores in child nodes represent the deviation edge; the "3" is underlined in "312" because that is the edge that differs from its parent node, "112." The root of the tree, and the top of the heap, is "111," representing the minimal-cost solution obtained by choosing the cheapest solution in every auction.

The heap-ordered tree of paths defined by deviations is computationally useful because it can be constructed *incrementally* by generating paths in ascending order of length. Figure 4 sketches our algorithm, SKSP. Step 1 is trivial for our problem: The shortest path consists of the shortest edge at each "hop" (see Figure 2). The loop starting at line 3 incrementally constructs the tree of paths, and the loop starting at line 8 explores the children (deviations) of paths that are known to be among the k shortest. The algorithm is memory efficient because lines 12–16 maintain the loop invariant that priority queue PQ contains at most k paths. If PQ is full (i.e., contains

k paths already), any new path that is longer than the longest in PQ may safely be discarded; neither it nor any of its children can be among the k shortest paths. Likewise, if a new path is discovered that is shorter than the longest already in PQ, the latter may safely be discarded.

Numerous improvements to SKSP are possible but we omit them for the sake of simplicity. E.g., if i paths have already been output, then PQ need not contain more than $k-i$ paths.

3.5 Computational Complexity

Memory Referring to the simple solutions graph of Figure 2, we require $O(1)$ memory for each of N nodes and E edges, where $N = O(I)$, $E = O(IR)$, and $R = R(S,Q)$ is the multiset expression of Equation 1. Storing the shortest-paths tree computed in step 1 requires $O(1)$ memory per node, so it does not increase overall asymptotic memory requirements. Finally, we also need constant memory for every record in priority queue PQ, for a total memory requirement of $O(I+IR+k) = O(IR+k)$.

Time Due to the special structure of our outcome graph, we require only $E = O(IR)$ time to compute the shortest paths tree in step 1 by simply finding the shortest edge at each "hop." Because we must be able to extract both the shortest and longest paths from priority queue PQ (steps 4 and 15), we implement PQ as a pair of binary heaps; all heap operations require $O(\log(k))$ time. The main loop (step 3) iterates k times, and each iteration requires a constant number of heap operations. Each iteration also generates all child deviations of a path and computes their lengths. The number of child deviations is at most the number of edges, $E = IR$. Generating a deviation requires $O(1)$ time but computing its length requires worst-case $O(k)$ time in our current implementation. In summary, SKSP requires $O(IR(1+k\log(k))) = O(k\log(k)IR)$ time in the worst case.

Single seller per item In the special case where the number of quantiles/shares $Q = 1$ (i.e., multi-sourcing is forbidden and the buyer acquires all units of each item from a single seller), SKSP requires $O(IS+k)$ memory and $O(k\log(k)IS)$ time, because $R(S,Q) = S$ in this case.

The k-shortest paths literature contains a wide variety of sophisticated algorithms; some have better asymptotic time and memory complexity than SKSP. Our goal in this paper is to explore an interesting new application of k-shortest paths algorithms rather than to advance or even to fully exploit the state of the art; we believe that the development of improved k best solutions algorithms for WDPs is a promising area for future research. We find that SKSP is more than adequate for our present purposes, and the fact that it admits convenient implementation and analysis weighs heavily in its favor.

4. GLOBAL CONSTRAINTS

Some constraints on a full solution can be imposed within the framework described in Sections 3.3 and 3.4 by restricting the set of outcomes of each individual-item sub-auction. For example, to produce solutions for which no single seller provides more than 30% of an item, it is clearly sufficient to remove individual-item outcomes that violate this constraint; any combination of remaining individual-item solutions must also obey the constraint.

Hard constraints on a full solution are those whose satisfying global solutions are not the product of restricted sets of individual-item outcomes. This section describes an approach for modifying the simple graph representation of Section 3.4 to incorporate certain types of hard constraints, in the sense that solutions that violate the constraints are not generated when the k-shortest paths algorithm operates on the modified graph. We call the expanded graph that encodes global constraints a *constrained solutions graph*, to distinguish it from the simple linear solutions graph depicted in Figure 2. The method of this section is not generally efficient, but several useful global constraints do have efficient representations; we enumerate some in Section 4.4.

In the process of expanding the graph to permit structural representation of complex constraints we inevitably increase the complexity of the k-shortest paths algorithm, and so the question arises as to whether it is better to do so, or to generate a larger list of candidate solutions more quickly and filter out those that violate the constraints. In practice the approach described in this section scales best when the constraint is most stringent—i.e. when the proportion of all paths failing the constraint is significant. This is in contrast to approaches in the literature, such as in Villeneuve & Desaulniers [22], that rely on forbidding a relatively small set of paths, whose computation time scales with the set of *forbidden* paths rather than the set of *permitted* paths.

4.1 Constrained Number of Winners

In this section we will demonstrate how to expand the solutions graph to form a constrained solutions graph G_f whose paths correspond to outcomes with a fixed number S_f of sellers receiving non-zero allocations. We choose this constraint for several reasons: it is useful, being a common concern of procurement executives; it is clearly only globally computable; and it has a relatively straightforward representation.

Define $\sigma_{inc}(o_i)$ to be the set of sellers that receive non-zero quantiles (that are *included*) in some auction given the outcome o_i, let $\sigma_{inc}(o_1,\ldots,o_i)$ likewise be the set of sellers that are included in the outcome $(o_1,\ldots,o_i)$ of a collection of auctions. The graph we construct is essentially a product of that defined in Section 3.4 with a state-transition graph whose states are the sets of sellers included after evaluation of the first $i-1$ individual-item auctions, and whose transitions correspond to adding additional sellers whose bids win in the i^{th} auction. We ensure that the constraint is satisfied by removing transitions that would include more than S_f sellers.

Let Σ_f be the set of sets of S_f or fewer sellers. The set of nodes $V(G_f)$ is

$$V(G_f) = \{\mathtt{s},\mathtt{t}\} \cup \left(\{1,\ldots I\} \times \Sigma_f\right).$$

The nodes s and t are a convenient source and sink for the k-shortest paths algorithm. For each vertex (i,σ) and outcome o_{i+1} of the individual-item auction $i+1$ $(1 \le i < I)$ such that $\sigma \cup \sigma_{inc}(o_{i+1}) \in \Sigma_f$ we add a directed edge from (i,σ) to $(i+1, \sigma \cup \sigma_{inc}(o_{o+1}))$, labeled with o_{i+1}, and with length $c(o_{i+1})$ equal to the cost of outcome o_{i+1}. In addition, for each outcome o_1 to the first auction we connect the source node s to $(1,\sigma_{inc}(o_1))$ with an edge of length $c(o_1)$ labeled with o_1, and we connect every node of the form (I,σ) for which $|\sigma| = S_f$ to the sink node t via an edge of length zero.

PROPOSITION 4.1. *The labeling of edges establishes a 1–1 mapping from paths in G_f connecting s to t, to global outcomes o satisfying the constraint $|\sigma_{inc}(o)| = S_f$.*

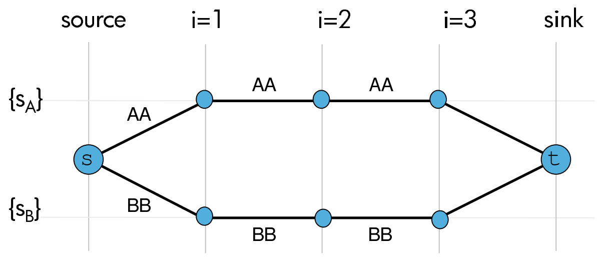

Figure 5: Constrained solutions graph G_1: $S = 2$, $I = 3$, $Q = 2$.

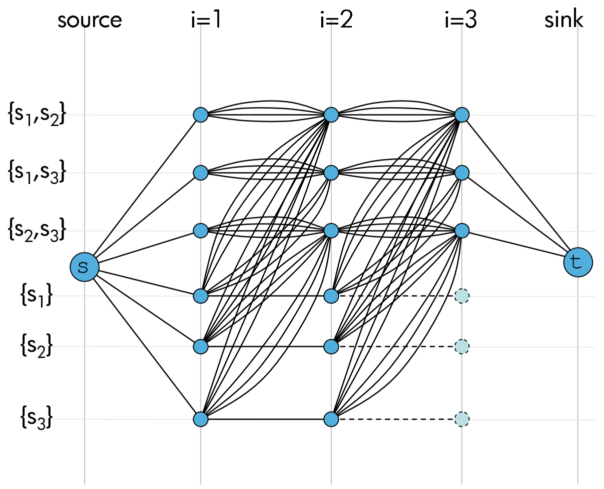

Figure 6: Constrained solutions graph G_2: $S = I = Q = 3$.

Proof. Consider a global outcome $o = (o_1, \ldots, o_I)$. If this outcome is represented by a path in G_f from $\texttt{s}$ to $\texttt{t}$ then by construction the sequence of nodes traversed must be $\texttt{s}$, $(1, \sigma_{inc}(o_1))$, $(2, \sigma_{inc}(o_1, o_2))$, $\ldots$, $(I, \sigma_{inc}(o_1, \ldots, o_I))$, $\texttt{t}$. The only vertices connected to $\texttt{t}$ are of the form (I, σ) with $|\sigma| = S_f$, so it follows that if o is represented by a path from $\texttt{s}$ to $\texttt{t}$ then $|\sigma_{inc}(o)| = S_f$.

Conversely, if o is a global outcome satisfying the constraint then $|\sigma_{inc}(o_1, \ldots, o_i)| \leq |\sigma_{inc}(o)| = S_f$, so that $\sigma_{inc}(o_1, \ldots, o_i)$ is in Σ_f, which implies that the vertices $(i, \sigma_{inc}(o_1, \ldots, o_i))$ are all in G_f. Clearly the edges labeled $\sigma_{inc}(o_1)$, $\ldots$, $\sigma_{inc}(o_I)$ are all in G_f, and the fact that o obeys the constraint implies that the final node $(I, \sigma_{inc}(o))$ is connected to $\texttt{t}$. □

Since the length of a path in G_f is the cost of the corresponding global outcome, the k-shortest paths of G_f correspond to the k-cheapest solutions to the winner determination problem. Figure 5 shows G_1 for the example in Section 3.2 ($I = 3$, $S = 2$, $Q = 2$). A more complicated example, G_2 given $I = S = Q = 3$, is shown in Figure 6. Note that G_f may contain vertices and edges that are not on any path from $\texttt{s}$ to $\texttt{t}$—for example (I, σ) with $|\sigma| < S_f$, as shown in Figure 6.

The complexity of G_f relative to the unconstrained solutions graph depends on the size of Σ_f. For S_f fixed, but S varying,

$$|\Sigma_f| = \binom{S}{S_f} = O(S^{S_f}).$$

4.2 Incremental Function Representation

In general, what we are doing in Section 4.1 is evaluating a function at each step whose value (set of sellers included in first i auctions) depends on the value at the previous step (set of sellers included in first $i-1$ auctions) and the outcome in the i^{th} auction. Any chain of functions of this form such that the suitability of a global outcome can be determined by examining the output of the last function provides a way of restricting the k-best algorithm to generate only those global outcomes (paths) that satisfy the global constraint. In the following definition, O_i is the set of outcomes to the i^{th} single-item auction, and $O = \prod_i O_i$ is the set of global outcomes.

Definition 4.1. *Suppose that $O_{cons} \subseteq O$ is a global constraint represented as a subset of the space of global outcomes (those that are acceptable). An **incremental representation** of O_{cons} is defined as a sequence of sets $X_i, i = 0, \ldots, I$ with $X_0 = \{*\}$, functions $f_i : X_{i-1} \times O_i \to X_i, i = 1, \ldots, I$, and a subset $X_{cons} \subseteq X_I$, such that the function $\mathcal{F} : O \to X_I$ defined by*

$$\mathcal{F}(o_1, \ldots, o_I) = f_I(f_{I-1}(\ldots f_2(f_1(*, o_1), o_2), \ldots o_{I-1}), o_I)$$

*satisfies $\mathcal{F}^{-1}(X_{cons}) = O_{cons}$. Given such a representation, the sets X_i will be referred to as the **partial values**, the functions f_i are the **incremental functions**, and X_{cons} the **final values**.*

In these terms, the constraint in Section 4.1 is represented by partial values equal to the set of subsets of sellers, $X_i = \Sigma_f$; the incremental function is the union of the partial value at step $i-1$ with the set of sellers included in step i, $f_1(*, o_1) = \sigma_{inc}(o_1)$, $f_i(x, o_i) = x \cup \sigma_{inc}(o_i)$, $i > 1$; and the final values are $X_{cons} = \{x \in \Sigma_f : |x| = S_f\}$.

4.2.1 Constrained Solutions Graph

For any incremental representation we can construct a corresponding constrained solutions graph G as in Section 4.1. We let the nodes of G correspond to the partial values, with a source and sink node added,

$$V(G) = \{\texttt{s}, \texttt{t}\} \cup \bigcup_{i=1}^{I} \{i\} \times X_i.$$

We will identify the source node $\texttt{s}$ with $(0, *)$ so as not to make special cases for f_1. For each $i < I$, $x_i \in X_i$ and $o_{i+1} \in O_{i+1}$ we add an edge to G to represent the transition that the outcome o_{i+1} induces from the "state" x_i to the new state $x_{i+1} = f_{i+1}(x_i, o_{i+1})$. This edge, which goes from the node (i, x_i) to $(i+1, f_{i+1}(x_i, o_{i+1}))$ is labeled with o_{i+1} and has length $c(o_{i+1})$. The same reasoning as for Proposition 4.1 gives

Proposition 4.2. *The labeling of edges establishes a 1–1 mapping from paths in G connecting $\texttt{s}$ to $\texttt{t}$ to global outcomes o satisfying the constraint $o \in O_{const}$.* □

4.3 Complexity

The complexity analysis of Section 3.5 extends naturally to constrained solutions graphs. The number of nodes in the graph is now $2 + \sum_{i=1}^{I} |X_i|$, and the number of edges $\sum_{i=1}^{I} |O_i| \times |X_{i-1}| + X_{cons}$. Introduce the constant $X_{max} = \max_i |X_i|$. Clearly the number of nodes is $O(IX_{max})$ and edges $O(IRX_{max})$, where $R = R(S, Q)$ is the multiset formula of Equation 1. Following through the same logic as in Section 3.5, we get that the storage requirements are $O(IRX_{max} + k)$ and time requirements $O(k \log(k) IRX_{max})$. In other words, the problem has become more complex by a factor of at most X_{max}.

4.4 Examples

In this section we detail a selection of global constraints of increasing complexity, and their corresponding incremental representations and constrained solutions graphs.

4.4.1 Worst Case

The first thing to notice about incremental representations is that *every* global constraint has an one: Let $X_i = \prod_{j=1}^{i} O_j$, let f_i be the identity function on $X_i = X_{i-1} \times O_i$, and let $X_{cons} = O_{cons}$. This is a representation because $\mathcal{F}$ is the identity function on $X_I = O$. However, it is not a useful representation, because the number of new nodes and edges required to create its constrained solutions graph scales very poorly: the size of the partial value space X_i can in general be bounded above only by $\prod_j^i |O_j|$, and there are $|O_{cons}|$ edges between nodes of the form (I, x_I) and $\mathtt{t}$.

4.4.2 Trivial Representation

Most simply we can consider the trivial constraint $O_{cons} = O$, which generates the unconstrained outcome graph, as described in Section 3.4[1]. In terms of the formalism of Definition 4.1, $X_i = \{*\}$ for all i. In this graph there are $I+2$ nodes and $1 + \sum_i |O_i|$ edges; it forms a complexity benchmark for the more sophisticated constrained solutions graphs to come.

4.4.3 Monotonic Predicates

The most important feature of the example in Section 4.1 is that the global constraint is evaluated over predicates of the form "is seller s assigned non-zero total volume?" Such predicates have two very nice properties: Most importantly, they can be evaluated incrementally at each step by a simple OR over whether the seller has yet been included and whether the seller is included at the current step. This implies that the space of partial values need be no larger than $2^{|S'|}$, where S' is the set of sellers under consideration in the global constraint. For example, an incremental representation of the constraint "Either seller 1 is included, or seller 2 is included, but not both" exists with $|X_i| = 4$.

Secondly, the value of the predicate is monotonic with respect to step i; if it is true at step i it will be true in all subsequent steps. This fact sometimes gives straightforward upper bounds on the partial value sets. For example, for the canonical representation of "Either seller 1 is included, or seller 2 is included, but not both", the partial value sets will clearly never contain a partial value in which both seller 1 and seller 2 are included: it is not necessary to wait until step I to realize this. If the constraint had been "Either seller 1 is assigned an even number of shares, or seller 2 is assigned an even number of shares, but not both", this would not have been possible. Similarly, it is this monotonicity that allows the upper bound on X_i in Section 4.1.

4.4.4 Price or Quantity Thresholds

Another slightly more complicated monotonic predicate from which global constraints can be built is, "has the value (or the quantity) assigned to seller s exceeded some threshold T".

The case of a quantile threshold is the easiest to tackle, since the total number of quantiles assigned to a seller necessarily takes one of only relatively few values[2]. Consider the constraint that the number of quantiles assigned to s be at least T. We let $X_i = \{0, 1, \ldots, T\}$ be the number of quantiles assigned to s in the first i auctions. The incremental functions aggregate additional quantiles as they are assigned, up to a maximum of T: $f_i(x, o_i) = \min(T, x + o_i(s))$, and there is a unique acceptable final value, $X_{cons} = \{T\}$.

The same representation can clearly be used to constrain the number of quantiles assigned to a seller above, so long as the upper threshold T' is strictly less than T, by choosing $X_{cons} = \{0, \ldots, T'\}$. For a large upper bound this would be expensive, and it might be cheaper to implement the constraint by counting and bounding below the number of quantiles *not* assigned to s.

When constraining value rather than quantiles, the situation is trickier, since it may take far too many values. The obvious approach is to round value to the nearest of a set of representative points. This will inevitably introduce errors into the constrained solutions graph approach. However, irrespective of the quantization chosen, if we always round the total value assigned so far down to the nearest quantum (for upper bounds, up for lower bounds), then this will be an underestimate, and hence we will never exclude from the constrained solutions graph paths corresponding to valid outcomes. Instead the graph may contain paths corresponding to invalid outcomes, having value close to but above the threshold, which must be filtered out from the final list of the k cheapest.

4.5 Summary

We do not yet have a full characterization of the global constraints that are amenable to incremental representations in the sense that they have tight bounds X_{max} relative to the unconstrained graph; this is a topic of our ongoing work. Nonetheless this section has presented several useful examples, namely constraints involving

1. upper or lower bounds on quantiles assigned to a seller;
2. the set of included sellers; and
3. bounds on the value assigned to a seller.

5. EXPERIMENTS

To explore the usefulness of the SKSP algorithm, we compute k best solutions based on actual bids submitted by parts suppliers to a procurement auction in which Hewlett-Packard spent approximately \$3.7 million. Our data set includes, for each of several dozen items, the total number of units that HP wished to acquire in the auction. We also have per-unit prices on each item from each of six sellers who participated in the auction. We define the *minimal procurement cost* of an item to be the lowest cost of satisfying HP's total demand for the item, and we rank items in descending order of minimal procurement cost.

Table 3 summarizes the five subsets of bids used in our experiments. The name of each bid set encodes its size and other properties. For instance, $B_{50,1}$ includes the top 50 items in the auction and sets granularity parameter Q to 1 (i.e., multi-sourcing items is forbidden). Table 3 shows that the top 50 items account for 99% of the buyer's total cost, assuming that

[1] G as defined in Definition 4.1 is actually the graph from Section 3.4 with one extra node and one extra edge added.

[2] Note that in the case $Q = 1$, constraining quantiles is the same as constraining the total number of items assigned to a given seller.

data set	# items I	# quantiles Q	vol discount?	% total cost	# undominated	# winners	# subsets	$\overline{H}$ range
$B_{50,1}$	50	1	N	99	33	5	1	0.461–0.545
$B_{25,1}$	25	1	N	90	30	3,4,5	4	0.304–0.691
$B_{25,4}$	25	4	N	90	35	4,5	2	0.417–0.548
$B_{25,4,D}$	25	4	Y	90	26	3,4,5	5	0.307–0.674
$B_{15,4,D}$	15	4	Y	75	46	2,3,4,5	8	0.096–0.756

Table 3: Summary of experiments and results.

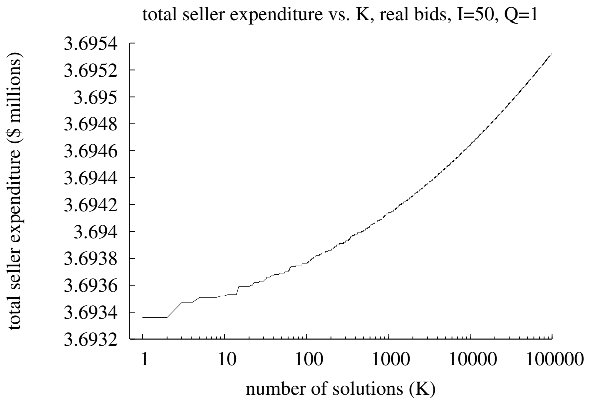

Figure 7: Total expenditure vs. solution rank.

the buyer acquires every item at its minimal procurement cost. Bid set $B_{25,4}$ includes only the top 25 items, which account for 90% of the buyer's minimal total expenditure. This data set furthermore sets $Q = 4$, meaning that a seller may supply 0%, 25%, 50%, or 100% of the buyer's demand for any item. Two of our bid sets, $B_{25,4,D}$ and $B_{15,4,D}$, include randomly-generated volume discounts; seller bids for $q = Q$ are still the real-world bids, but we randomly generate bids for quantities $q = 1, \ldots, Q-1$ that charge more per unit for smaller purchases.

Figure 7 plots total buyer expenditure as a function of solution rank k for the top 100,000 solutions based on the $B_{50,1}$ bid set. The figure shows that seller expenditure increases very gradually with k (note the logarithmic horizontal scale); the 100,000th-best solution is only 0.054% more expensive than the cheapest solution. This remarkable result is robust across a wide range of experiments that we do not report in detail due to space limitations: *The top k solutions entail nearly equal expenditure, even for large values of k.*

But are the top k solutions *diverse*, i.e., do they differ from one another in "interesting" ways? We address this crucial question by first defining several summary measures of procurement WDP solutions and then describing how they vary among the top $k = 25,000$ solutions generated for each of our five bid sets.

One important feature of a procurement WDP solution is the way in which it distributes the buyer's money across sellers. Consider an *expenditure vector*, $\vec{x} = (x_1, x_2, \ldots, x_S)$, where x_s is the fraction of the buyer's total expenditure given to seller s. We quantify uniformity of expenditure by treating $\vec{x}$ as a probability vector and computing its information-theoretic entropy [9],

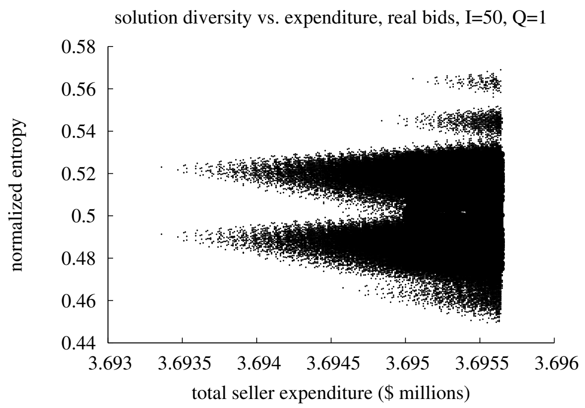

Figure 8: Normalized entropy vs. expenditure.

normalized to lie in the range $[0,1]$: $\overline{H} = \sum_{s=1}^{S} -x_s \log_2 x_s / M$, where M is a normalization constant. $\overline{H} = 1$ when all sellers receive an equal amount of the buyer's money and it is zero when a single seller supplies all of the buyer's demand. Other important WDP solution attributes include the number of sellers included and also the specific subset of sellers who win.

The rightmost three columns of Table 3 present three views of solution diversity among the top $k = 25,000$ solutions for our bid sets. Column 7 shows the numbers of winners. For instance, *all* of the top 25,000 solutions based on the $B_{50,1}$ bids involve exactly five sellers, but the $B_{25,1}$ bids generated solutions involving three, four, or five sellers. Since we have $S = 6$ sellers, one of $2^6 - 1 = 63$ subsets of sellers are winners in any particular solution. Column 8 shows the number of such subsets that actually occur among the top 25,000 solutions. All solutions to $B_{50,1}$ involve the *same* subset of sellers, but four subsets appear among the solutions to $B_{25,1}$. The last column in Table 3 shows the range of normalized entropy $\overline{H}$ across the top 25,000 solutions.

The three diversity measures in Table 3 suggest similar conclusions: Comparing $B_{50,1}$ with $B_{25,1}$, and comparing $B_{25,4,D}$ with $B_{15,4,D}$, we see that *including fewer top items increases solution diversity.* An analogy to road networks explains why: The second-shortest path between two cities is likely to involve a brief and trivial detour from the shortest path if the network includes minor streets; we can ensure that paths differ substantially by including only major highways. The analogous operation for procurement auctions is to include only the top-ranked items in terms of minimal procurement cost.

A comparison of $B_{25,1}$ with $B_{25,4}$ suggests that multi-sourcing items by increasing granularity parameter Q may decrease solution diversity. However the $B_{25,4,D}$ results suggest that volume discounts can restore the diversity lost by increasing Q. Finally, $B_{15,4,D}$ shows that the combined effect of volume discounts and a greatly reduced item count dramatically increases diversity among the top solutions.

How useful are the k best solutions when the procurement WDP is viewed as a multi-criteria optimization problem? For instance, what if greater uniformity of expenditure across sellers is always desirable (all else being equal), but the decision-maker is reluctant to quantify the dollar value of increasing $\overline{H}$? Figure 8 is a scatterplot of $\overline{H}$ versus total expenditure for over 250,000 of the top $B_{50,1}$ solutions. The Pareto frontier

	number of sellers in solution			
$\overline{H}$ range	2	3	4	5
[0.0, 0.1)	0.819%			
[0.1, 0.2)	0.772%	0.276%		
[0.2, 0.3)	0.594%	0.098%	0.512%	
[0.3, 0.4)	0.655%	0%	0.290%	0.883%
[0.4, 0.5)		0.053%	0.112%	0.638%
[0.5, 0.6)		0.172%	0.112%	0.416%
[0.6, 0.7)			0.284%	0.416%
[0.7, 0.8)				0.527%

Table 4: Percent premium vs. $\overline{H}$ and number of winners.

of undominated solutions appears to be remarkably small, and column 6 in Table 3 confirms that this is indeed the case for all of our data sets: For the bi-criteria problem involving expenditure and normalized entropy, the top 25,000 solutions contain fewer than fifty undominated solutions. Experiments with other criteria, not reported here, lead to the same conclusion: For practical procurement problems, the Pareto frontier is conveniently small.[3]

Note that scenario navigation and preference elicitation techniques based on off-the-shelf integer program solvers will encounter difficulty if preferences or constraints involve normalized entropy, because the logarithmic terms in $\overline{H}$ turn the WDP into a difficult *non-linear* mixed integer program. This observation illustrates an important and very general advantage of our method: Our framework places no restrictions on the form of preferences or constraints, and it does not force a tradeoff between convenience of implementation and fidelity in modeling the problem domain.

Now let us suppose that the decision-maker's preferences involve the number of sellers included in a solution as well as the uniformity of expenditure across sellers. Table 4 shows how to compactly summarize the 25,000 best solutions from our most diverse solution set (based on the $B_{15,4,D}$ bids). For each number of sellers and each range of $\overline{H}$, the table shows the additional cost of the best solution expressed as a percentage of the cheapest solution. For example, the cheapest solution involving five sellers and $\overline{H}$ in the range $[0.7, 0.8)$ is 0.527% more expensive than the globally cheapest solution. The prices of bundles of constraints can be read directly from Table 4. For example, if the decision-maker insists on a solution involving five sellers and $\overline{H} \geq 0.4$, the cheapest solution satisfying this bundle of constraints costs 0.416% more than the globally cheapest solution.

The examples of this section show how an unwieldy table of *k*-cheapest solutions can be condensed to a convenient size to aid the decision-maker in a procurement auction. Our ongoing research explores other ways of summarizing the *k* best solutions, e.g., via clustering. We do not report run times for our SKSP prototype, but all of our experiments ran in a matter of seconds on an aging laptop computer. The entire solver consists of under 700 lines of C, excluding comments.

The results of this section are based on real bids submitted by parts suppliers to an actual procurement auction. By contrast, most published evaluations of WDP solvers and preference elicitation schemes rely on randomly-generated bids. One problem with random bid generators is that they often inadvertently produce easy problem instances [2, 15]. We experimented with random bids and repeatedly encountered an orthogonal difficulty: We found it remarkably difficult to generate sufficiently realistic random bids. In several cases, seemingly reasonable random bid generators turned out to have subtle deficiencies that became evident only when experimental results were compared with those based on real bids.

[3]If *k* and the number of criteria are large, finding the Pareto frontier via naïve algorithms is infeasible. However, sophisticated and efficient algorithms exist for finding undominated points in high-dimensional data [23].

6. RELATED WORK

Preference Elicitation A range of preference elicitation techniques have been developed for general single-agent decision problems. Applications to auctions are sometimes motivated by a desire to preserve privacy and shorten bids [18]. Conen & Sandholm describe an approach to elicitation in combinatorial auctions that employs queries such as, "In your preferences, what is the rank of bundle A?"; the worst-case number of required queries is exponential in the number of items [7]. Boutilier *et al.* explore elicitation to aid uncertain decision-makers; their approach employs relatively user-friendly queries ("do you prefer outcome x or x'?") and they optimize a measure of minimax regret [6]. Sandholm & Boutilier review preference elicitation in combinatorial auctions [20]. They report that most approaches place strong restrictions on preferences and require exponentially many queries in the worst case.

Auctions & Knapsack Problems The relationship between winner determination in sealed-bid combinatorial auctions/exchanges and generalized knapsack problems was noted several years ago by Holte [13] but attracted little attention until recently. Kellerer *et al.* provide the first extensive discussion of the auction-knapsack connection [14]; see [15] for a detailed literature review. Tennenholtz outlines a method for solving a two-good-type combinatorial auction WDP by converting it to a longest-paths problem, but does not explore the relationship with knapsack problems or consider the applicability of *k*-shortest paths algorithms [21].

***k*-Shortest Paths** The literature on *k*-shortest paths algorithms is extensive; see Eppstein [10] for an introduction. Much of the literature is motivated by applications involving multi-criteria or constrained optimization: The algorithms are used to generate a large number of solutions that are post-processed to eliminate dominated or unsatisfactory solutions. Numerous problem variants are considered: computing cyclic vs. simple (loopless) paths, directed vs. undirected graphs, and single vs. multiple edges between node pairs. Our problem is that of computing *k*-shortest simple paths in a directed acyclic graph containing multiple arcs between node pairs.

Techniques for encoding constraints in graph structure in such a way that a *k*-shortest paths algorithm generates only those paths that satisfy the constraints have been explored. For example, Villeneuve & Desaulniers describe an approach based on string-matching algorithms [22]. Their method differs from ours in that its computational demands increase with the number of forbidden paths; it is not well suited to the global constraints that we consider in Section 4.

Coutinho-Rodrigues *et al.* explore a hybrid technique that combines *k*-shortest paths computations with interactive elicitation queries to explore the Pareto frontier in bi-criteria optimization problems [8]. We believe that a similar fusion of preference elicitation with our *k*-best solutions approach is a promising topic for future research.

7. CONCLUSIONS

This paper has introduced a general method for computing *k* best solutions to a large class of sealed-bid auction winner determination problems. Applied to procurement auctions, our approach scales to real-world problem sizes even if multi-sourcing and volume discounts/surcharges are permitted. For procurement problems with realistic bounds on multi-sourcing (small fixed *Q*), our algorithm's complexity scales polynomially with respect to the number of sellers *S*, less than quadratically with respect to *k*, and linearly with the number of items *I*. Arbitrary non-linearities in constraints or in preferences over solution attributes pose no special difficulties. In addition, via the constrained outcome graph developed in Section 4, many useful global constraints can be accommodated with only a modest increase in computational complexity. Our algorithm can be implemented from scratch with moderate effort and at low cost; it does not depend on an integer program solver or other third-party software.

Empirical results based on real bids from a real procurement auction demonstrate that our algorithm can generate a large number of solutions to practical auction WDPs. Additionally we have shown that the analysis of *k*-best solutions can give the decision-maker valuable qualitative as well as quantitative insight into a promising region of the solution space.

Our approach complements existing preference elicitation techniques and sidesteps several of their shortcomings, e.g., the possibility of intransitive revealed preferences, the need for excessively numerous or vexing queries, and restrictions on the functional form of constraints and utility functions. The set of *k* best solutions that we compute provides rich opportunities to leverage a wide range of existing data analysis techniques; our ongoing research explores these possibilities.

8. ACKNOWLEDGMENTS

We thank Kemal Guler for reviewing numerous drafts of this paper. Donald Knuth provided valuable insights into the combinatorial generation problem of listing all solutions to individual-item sub-auctions. Alex Zhang explained the implications of our non-linear expenditure-uniformity measure. Michael Wellman, Daniel Reeves, and Claudio Bertolini offered detailed feedback on a late draft.

9. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[2] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *ICMAS*, 2000.

[3] W. H. Auden. In C. Fadiman, editor, *I Believe*. Simon & Shuster, 1939.

[4] J. Beckett. The business of bidding: Reinventing auctions for better results, Sept. 2005. `http://www.hpl.hp.com/news/2005/jul-sep/auctions.html`.

[5] E. A. Bender and G. S. Williamson. *Foundations of Applied Combinatorics*. Addison-Wesley, 1991.

[6] C. Boutilier, T. Sandholm, and R. Shields. Eliciting bid-taker non-price preferences in (combinatorial) auctions. In *Proc. AAAI*, 2004.

[7] W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions [extended abstract]. In *Proc. ACM E-Commerce Conf.*, Oct. 2001.

[8] J. Coutinho-Rodrigues, J. Climaco, and J. Current. An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. *Computers & Operations Research*, 26:789–798, 1999.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[10] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[11] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, third edition, 1970.

[12] W. Hoffman and R. Pavley. A method for the solution of the nth best path problem. *Journal of the ACM*, 6(4):506–514, Oct. 1959.

[13] R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In E. Stroulia and S. Matwin, editors, *LNCS*, volume 2056. Springer, 2001.

[14] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[15] T. Kelly. Generalized knapsack solvers for multi-unit combinatorial auctions. In *Agent Mediated E-Commerce (AMEC VI)*, July 2004. `http://ana.lcs.mit.edu/peyman/amec-vi-accepted.htm`. Also HP Labs TR HPL-2004-21 & Springer LNAI.

[16] D. E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating all Combinations and Partitions*. Addison-Wesley, 2005.

[17] D. E. Knuth. Personal communication, Sept. 2005.

[18] S. M. Lahaie and D. C. Parkes. Applying learning algorithms to preference elicitation. In *Proc. ACM E-Commerce Conf.*, pages 180–188, May 2004.

[19] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, Aug. 1998.

[20] T. Sandholm and C. Boutilier. *Combinatorial Auctions*, chapter 10. MIT Press, Jan. 2006.

[21] M. Tennenholtz. Some tractable combinatorial auctions. In *Proc. of 17th Nat'l Conf. on AI*, July 2000.

[22] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operations Research*, 165:97–107, 2005.

[23] M. A. Yukish. *Algorithms to Identify Pareto Points in Multi-Dimensional Data Sets*. PhD thesis, U. Penn. Dept. of Mech. Eng., Aug. 2004.