Efficiently Generating k-Best Solutions to Procurement Auctions

Andrew Byde, Terence Kelly, Yunhong Zhou, Robert Tarjan

HP Laboratories
HPL-2009-163

Keyword(s):
procurement, auctions, decision support, combinatorial optimization, knapsack problems, k-shortest paths

Abstract:
Procurement executives often find it difficult to articulate their preferences and constraints regarding auctions, making it difficult to cast procurement decisions as straightforward optimization problems. This paper presents an efficient algorithm to aid decision support in such situations. Instead of trying to compute a single optimal solution for the auction winner determination problem, we generate many candidate solutions in ascending order of buyer expenditure. Standard techniques such as clustering and dominance pruning can then trim this list to a compact yet diverse menu of alternatives; other analyses can illuminate the cost of constraints and the competitive landscape. Our efficient solution-generation algorithm addresses sealed-bid procurement auctions with multiple suppliers and multiple types of goods available in multiple units. It supports multi-sourcing and volume discounts/surcharges in bids. Our algorithm may optionally incorporate certain classes of hard constraints, generating only solutions that satisfy them.

External Posting Date: July 21, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: July 21, 2009 [Fulltext]

Presented at AAIM 2009, San Francisco, June 15, 2009

Efficiently Generating *k*-Best Solutions to Procurement Auctions

Andrew Byde, Terence Kelly, Yunhong Zhou**, and Robert Tarjan

Hewlett-Packard Laboratories
Palo Alto, California, USA
{andrew.byde, terence.p.kelly, robert.tarjan}@hp.com,
yunhong.zhou@gmail.com

Abstract. Procurement executives often find it difficult to articulate their preferences and constraints regarding auctions, making it difficult to cast procurement decisions as straightforward optimization problems. This paper presents an efficient algorithm to aid decision support in such situations. Instead of trying to compute a single optimal solution for the auction winner determination problem, we generate many candidate solutions in ascending order of buyer expenditure. Standard techniques such as clustering and dominance pruning can then trim this list to a compact yet diverse menu of alternatives; other analyses can illuminate the cost of constraints and the competitive landscape. Our efficient solution-generation algorithm addresses sealed-bid procurement auctions with multiple suppliers and multiple types of goods available in multiple units. It supports multi-sourcing and volume discounts/surcharges in bids. Our algorithm may optionally incorporate certain classes of hard constraints, generating only solutions that satisfy them.

1 Introduction

The problem of clearing a sealed-bid auction—i.e., determining how goods and payments change hands among participants as a function of auction rules and bids—is conventionally known as the *winner determination problem* (WDP). For most kinds of auctions it is easy to define the WDP as a straightforward optimization problem, e.g., an integer linear program [1]. In practice, however, it can be difficult for auction participants to supply all of the inputs required to solve the WDP, particularly the constraints that define the space of permissible solutions and the preferences that allow a WDP solver to select the best solution.

Our primary focus in this paper is on the buyer's decision problem in sealed-bid procurement auctions, also known as reverse auctions. A procurement executive given seller bids in such an auction might "know the right solution when she sees one." However if she cannot articulate its *properties* in terms of hard constraints and soft tradeoffs among conflicting desiderata, a straightforward optimization formulation of the WDP does not by itself allow the auction to be cleared.

** Currently at Rocket Fuel, Inc.

Existing decision-support techniques such as scenario navigation and preference elicitation extend an optimization framework by requiring additional inputs from the decision maker, e.g., replies to elicitation queries. This paper considers an alternative framework that provides the buyer in a procurement auction with additional outputs rather than demanding additional inputs. Specifically, we use seller bids to generate a large list of alternatives from the most promising region of the solution space: the solutions that entail minimal buyer expenditure.

The key to our approach is generating k-best (i.e., k-cheapest) solutions to the auction WDP. An earlier paper described experiments based on real bids submitted to a real procurement auction; the results demonstrated the usefulness of our k-best solutions approach [2]. The present paper presents a far more efficient solution-generation algorithm, shows how to incorporate into the same decision framework the risk of supplier failure to deliver, and theoretically analyzes the relationship between solution rank k and buyer expenditure.

Our overall approach is in principle applicable to auctions other than procurement auctions (see Section 2), but its efficiency depends on the precise form of the WDP considered—an unavoidable fact due to the NP-hardness of solving general WDPs [3]. When specialized for procurement auctions, in which portions of a procurement order must be assigned among sellers such that the total order is filled exactly, our algorithm can scale to practical problem sizes. We can incorporate certain kinds of hard constraints to prevent unacceptable solutions from being generated, and our approach allows multi-sourcing and the expression of volume discounts and surcharges in bids.

Once generated, the k-cheapest solutions to a procurement auction can be post-processed in several helpful ways. Ordinal preferences over solution attributes enable dominance pruning that yields a smaller Pareto frontier of solutions. The buyer may also reduce the number of candidate solutions by clustering them and considering only the cheapest in each cluster. Furthermore, the k-best solutions define prices on bundles of constraints: The price of any bundle of constraints satisfied by a generated solution is the cost difference between the cheapest satisfying solution and the cheapest unconstrained solution. These prices can focus the buyer's attention on the auction's most pressing tradeoffs. No restrictions on the mathematical form of constraints or preferences are necessary; arbitrary non-linearities pose no special difficulties. Finally, the k-cheapest solutions admit a wide range of informative visualizations. See [2] for a more detailed discussion of post-processing, including empirical results.

2 General Approach

Before refocusing attention on procurement auctions in Section 3, we briefly consider the fully general case of arbitrary sealed-bid combinatorial auctions/exchanges. This very general context makes it easy to sketch the basic ideas underlying our approach and explain why the restrictions of less general WDPs are necessary to obtain a computationally efficient solution generator.

The basic recipe for generating k-best solutions to any WDP follows from linking four observations [2]:

1. the WDP in combinatorial exchanges is a generalized knapsack problem [4];
2. dynamic programming can solve such problems [5];
3. dynamic programs are equivalent to shortest path problems [6]; and
4. we can generate the k shortest paths in a graph [7].

We can therefore construct a graph whose *paths* correspond to WDP solutions and whose path *lengths* correspond to objective function values in the WDP optimization problem (e.g., path lengths might represent buyer expenditure in a procurement auction). By computing k-shortest paths on this graph we obtain k-best solutions to the WDP.

Unfortunately, while this approach is straightforward, there is good reason to believe that it cannot be computationally tractable for the fully general case of arbitrary combinatorial auction/exchange WDPs: Merely computing the *first-best* solution to a combinatorial auction is NP-hard [3]; computing k-best solutions can be no easier.

We gain more detailed insight into the computational complexity of combinatorial exchange WDPs by considering four natural measures of problem size: the number of *types* of goods, the number of *units* of each good available, the number of participating agents, and the length of agent bids. The computational difficulty of solving a fully general combinatorial exchange WDP is remarkably modest in terms of three of these four measures: Practical solvers with pseudo-polynomial time and memory requirements are available if the number of types of goods is a small constant [4]. The parameter responsible for intractability in the fully general context is the number of types of goods.

In this paper we present an approach that achieves computational efficiency by restricting attention to a class of procurement auctions, defined precisely in Section 3. The most important benefit of the restrictions that differentiate our procurement auctions from the fully general case of arbitrary combinatorial exchanges is that the procurement WDP admits efficient solvers whose computational demands scale pseudo-polynomially in all problem size parameters, including the number of good types. The k-best solution generation algorithms that we present in this paper have computational demands quadratic in a granularity parameter that divides the seller's demand for each type of good into equal shares. In practical procurement auctions this parameter is reasonably small, so the quadratic cost is acceptable.

The remainder of this paper is organized as follows: Section 3 formalizes our class of procurement auctions. Section 4 presents a k-best-solutions algorithm for the case of unconstrained solutions. Section 5 expands the scope of our method to incorporate constraints at the local level (e.g., no seller may supply more than 80% of any one item) and at the global level (e.g., at least three sellers must be involved in the global solution). Section 6 describes an extension of the notion of "cost" from the obvious monetary interpretation to a risk-based interpretation. Section 7 briefly summarizes the results of experiments on real bids from an actual material-parts procurement auction [2], demonstrating that

our approach yields useful insights for procurement executives. Section 8 analyzes the distribution of buyer cost among the k-cheapest solutions assuming that supplier bids are random variables constrained in reasonable ways. Section 9 reviews related work, and we conclude in Section 10.

3 Procurement Auctions

Businesses increasingly obtain goods through procurement auctions. Such auctions account for tens of billions of dollars of HP's expenditures in recent years [8], and US firms spend hundreds of billions of dollars via procurement auctions per year. In practice, buyer preferences typically encompass non-price solution attributes and side constraints, e.g., a desire to have 2–4 suppliers for each type of good; XOR constraints on winners, e.g., "supplier B must be excluded if A is chosen"; constraints on the total number of winning sellers; constraints on the distribution of expenditure across sellers. Many of these constraints are motivated by risk-management concerns related to delivery failure or delay, a topic to which we shall return in Section 6. Furthermore many are "soft" in the sense that the buyer would waive them in exchange for sufficiently large savings.

3.1 Definitions and Notation

Let S denote the number of *sellers*, a term that we will use interchangeably with *suppliers*; we assume that $S \geq 2$. Let I denote the number of *items* (distinct types of goods) that the buyer wishes to acquire; the overall procurement auction consists of I single-item sub-auctions that are cleared simultaneously. Global granularity parameter Q specifies the number of *quantiles* (shares of an item) 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. In Section 4.3 we shall consider a more general case, in which the number of quantiles depends on the item i; it makes no difference to the construction or complexity, so for the sake of clarity we will assume until then that the total number of quantiles is uniform across single-item auctions.

A vector of quantity assignments $\mathbf{q} = \{q_{i,s}\}$ is a *solution* (or *outcome*) of the auction if the constraint $\sum_{s=1}^{S} q_{i,s} = Q$ is satisfied for all items i. Our objective will be to rank such solutions in order of some *cost* function:

$$c(\mathbf{q}) = \sum_{i=1}^{I} \sum_{s=1}^{S} B_{is}(q_{i,s}), \tag{1}$$

where $B_{is}(q)$ is any non-negative function that is calculable from an assignment of a given quantity of a given item to a given seller, and for which $B_{is}(0) = 0$. The canonical example is the amount of money that seller s demands for providing q quantiles of item i, but others definitions are useful (see Section 6). When we refer to "cheapest" we will implicitly mean cost in this general sense.

The data (I, S, Q, B) specifies a procurement auction WDP. For a given auction we will construct a weighted, directed acyclic graph G with special source and sink vertices `s` and `t` such that the k-shortest paths from `s` to `t` correspond to the k cheapest solutions to the WDP. Eppstein [7] demonstrates that given a graph G with n vertices and m edges, the k shortest paths can be calculated implicitly in time $O(m + n \log n + k)$. The $n \log n$ term comes from Dijkstra's algorithm [9] for constructing the tree of shortest paths from `s` to each other vertex; if such a tree has already been constructed Eppstein's algorithm takes time $O(m + k)$. Happily, since our graph is directed and acyclic, it is possible to construct the shortest-path tree in time $O(m)$, so that our graph's k shortest paths can be found implicitly in time $O(m + k)$. To extract explicit representations takes additional time proportional to the number of edges in each path, for which we will derive good bounds in Section 4.2.

Graph construction is significantly simpler in the unconstrained case, so we discuss this first; a full discussion of the constrained case is presented in Section 5.

4 Unconstrained Solutions

It is natural to decompose the problem by item, constructing a sub-graph for each item (with the appropriate correspondence between shortest paths and cheapest solutions), and then chaining the sub-graphs together; a concatenation of paths will correspond to a multi-item solution, and its length to total cost.

4.1 Single-Item Sub-Graph

To construct a single-item graph G, we consider a set of vertices with coordinates (s, q). We choose the source `s` to be the vertex $(0, 0)$, the sink `t` to be (S, Q), and the intermediate vertices to be all those for which $s = 1, \ldots, S-1$ and $q = 0, \ldots, Q$. Given these vertices for G, we add an edge from (s, q) to $(s{+}1, q{+}q')$ with label q' and length $B_{s+1}(q')$, whenever both of these vertices are in G. This edge corresponds to an assignment of quantity q' to seller $s + 1$.

The graph for $S = Q = 3$ is shown in Figure 1. Each edge corresponds to assigning 0, 1, 2 or 3 of the 3 available quantiles to a particular seller; the "length" of each edge is the corresponding bid $B_{is}(q)$. The reader can verify that there are exactly ten paths from `s` to `t` (edges are directed, left to right), corresponding to the ten ways of allocating 3 quantiles among 3 sellers. In general the total number of distinct paths through the unconstrained single-item graph can be shown to be $R(S, Q) = (Q + S - 1)!/(Q!(S - 1)!)$, which is well known to be the number of ways of placing Q indistinguishable balls into S distinguishable cells.

Lemma 1. *Each path in G from `s` to `t` corresponds, via the edge labeling, to a non-negative integer solution $\mathbf{q}$ of the equation $\sum_{s=1}^{S} q_s = Q$, and vice-versa. Furthermore the length of this path is exactly the cost to the buyer of the outcome $\mathbf{q}$.*

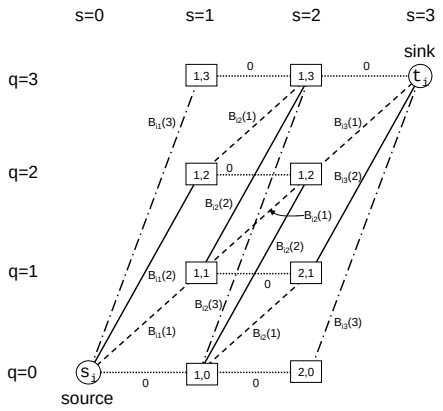

Fig. 1. Individual-item solutions graph for $S = Q = 3$. Each edge is directed, left to right. Finely dashed edges have label $q = 0$; roughly dashed edges have label $q = 1$; solid edges have label $q = 2$ and dash-cut edges have label $q = 3$. Edge lengths are shown next to each edge.

Proof. Suppose a path in G has edge labels q_s. By induction on s, any path starting at $\mathtt{s}$ with the labels q_s, $k = 1, \ldots, s$ must end at $(s, \sum_{k \le s} q_k)$, so the fact that the sink vertex has label (S, Q) proves the equation. For the converse statement, the equation implies that the vertices $(s, \sum_{k \le s} q_k)$, $s = 0, \ldots, S$ are all necessarily in G; the path that links these vertices in order of s clearly has edge labels q_s by the definition of edge labels in G.

4.2 Multi-item Graph

Having understood the intuition behind a single-item subgraph, we present the formal definition of the unconstrained multi-item graph, as a concatenation of single-item graphs for each item:

Definition 1. *Let (I, S, Q, B) be a WDP; the **unconstrained solutions graph** $G_u(I, S, Q, B)$ is defined as follows: The set of vertices is the set of tuples (i, s, q), where $i = 1, \ldots, I$ and either*

- $(s, q) = (0, 0)$*; or*
- $s = 1, \ldots, S-1$ *and* $q = 0, \ldots, Q$*; or*
- $(s, q) = (S, Q)$.

The set of edges is constructed by adding an edge from (i, s, q) to $(i, s+1, q+q')$ with label q' and length $B_{is+1}(q')$ whenever both vertices are in G_u; we also add connecting edges from (i, S, Q) to $(i+1, 0, 0)$ for each $i = 1, \ldots, I-1$, with no label and length 0. We identify $(1, 0, 0)$ as the source $\mathtt{s}$ of G_u, and (I, S, Q) as the sink $\mathtt{t}$.

Proposition 1. *There is a one-to-one correspondence between solutions to the WDP of an auction (I, S, Q, B) and paths in $G_u(I, S, Q, B)$ from $\mathtt{s}$ to $\mathtt{t}$.*

Proof. The proposition is a simple consequence of Lemma 1, since G_u is a concatenation of single-item graphs.

Complexity Each single-item sub-graph of G_u has $(Q+1)(S-1)+2 = O(SQ)$ vertices and

$$(S-2)\left(\frac{(Q+1)(Q+2)}{2}\right) + 2(Q+1) \;=\; O(SQ^2)$$

edges. It follows that $n = O(I\,S\,Q)$, and $m = O(I\,S\,Q^2)$, so that the complexity of implicitly finding the k-shortest paths is $O(I\,S\,Q^2 + k)$. To explicitly extract a path requires additional computation in proportion to the number of edges in the path [7], which is $I \times S$, so explicit enumeration of the k-shortest paths takes time $O(I\,S(Q^2 + k))$.

4.3 The General Problem

For real-world auctions, instead of a fixed number of quantiles for all items, there is an exact number of units to acquire for each item. Quantile is a heuristic we use to obtain reasonable approximate solutions by dividing the number of units for each item into the same number of quantiles (rounding if necessary). In this section we consider *the general problem* where items have various desired number of units, and denote the procurement problem where all items have the same number of quantiles as *the simplified problem*.

For item i, let Q_i denote the total number of units desired, for $i = 1, \ldots, I$. If $Q_i = Q$ for all i the general problem degenerates into the simplified problem. For each item i, Q_i is no longer very small, and it could be exponentially large (e.g., a large computer firm usually needs to procure millions of units for each computer part). For the general problem, we can solve it as in Section 4. We now require $\sum_{s=1}^{S} q_{i,s} = Q_i$ for each item i. We construct the single-item subgraph G_i for each item i, identical to the construction in Section 4.1. Thus G_i has $O(SQ_i)$ vertices and $O(SQ_i^2)$ edges. The multi-item graph G is a concatenation of single-item subgraphs G_i for all i, thus it has $O(S\sum_i Q_i)$ vertices and $O(S\sum_i Q_i^2)$ edges. Using Eppstein's algorithm, we obtain:

Theorem 1. *The general procurement problem (demanding Q_i units of item i) is pseudo-polynomial-time solvable. We can compute the k-cheapest solutions in time $O(S\sum_i Q_i^2 + k)$ implicitly and time $O(S\sum_i Q_i^2 + kSI)$ explicitly.*

5 Constrained Solutions

Some constraints on a full solution can be imposed within the framework described in Section 4 by simply removing some edges from graph. Because the full graph is a chain of single-item graphs, any constraint that can be incorporated in this way can be factorized into constraints on each single-item auction, and for this reason we call them "local". An example of a local constraint is that no seller provide more than 80% of any item; this can be represented by removing all edges whose label q is greater than $0.8Q$.

Since the incorporation of local constraints is so straightforward, we will turn our attention to *hard* constraints, whose satisfying global solutions are *not* the product of restricted sets of individual-item outcomes; the canonical example, which has great practical importance for risk management, is that of restricting the number of sellers involved in the global solution.

5.1 Hard Constraints

This section describes an approach for modifying the simple graph representation of Section 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*. The method of this section is not generally efficient, but several useful global constraints do have efficient representations; we enumerate some in Section 5.3.

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 [10], 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.

5.2 Constrained Solutions Graph

In this section we consider the problem of constructing a graph all of whose paths correspond to solutions satisfying some constraint $\mathcal{C}$. Our method is, as in Section 4, to construct a graph G_c with vertices indexed by item-seller-quantity triples, but now with an auxiliary variable, $x \in X$, which represents the "state" of the solution so far constructed. By restricting those edges that are added to G_c on the basis of their state, we can exclude paths that are bound to violate the constraint. X will therefore represent the intermediate states in the evaluation of the acceptability of an outcome as the outcome is constructed by assigning quantities to suppliers.

We can formalize this description in the following way. We consider the edges $edges(G_u)$ of the unconstrained graph, and because of the need to bootstrap the evaluation of the auxiliary variable, pay particular attention to those edges $edges(\mathtt{s})$ that originate at the source vertex $\mathtt{s}$. A **representation function** is defined to be a tuple (X, f), such that $f : edges(\mathtt{s}) \cup \big(X \times \big(edges(G_u) \setminus edges(\mathtt{s})\big)\big) \to X$ is a function mapping a source edge, or a non-source edge and a state value, to another state value. A representation function can be extended from edges to paths starting at $\mathtt{s}$, by repeated application: for any sequence of edges $e_1, \ldots, e_m$

starting at $\mathtt{s}$ we define

$$f(e_1, \ldots, e_m) := f(f(\ldots f(f(e_1), e_2), \ldots, e_{m-1}), e_m) \tag{2}$$

By construction, paths in the unconstrained graph from $\mathtt{s}$ to $\mathtt{t}$ are in one-to-one correspondence with solutions to the WDP; we say that a function (X, f) and the set of final states $X_{\mathtt{t}}$ **represents** a constraint $\mathcal{C}$ if the set of solutions obeying the constraint corresponds in this way to exactly the set of paths $(e_1, \ldots, e_m)$ such that $f(e_1, \ldots, e_m) \in X_{\mathtt{t}}$.

Definition 2. *Let (I, S, Q, B) be a WDP, and $(X, f, X_{\mathtt{t}})$ a representation of a constraint $\mathcal{C}$ on the set of acceptable solutions. The **constrained solutions graph** for this WDP, $G_c(I, S, Q, B, X, f, X_{\mathtt{t}})$, is defined as follows:*

1. *Let the vertices of G_c be $\mathtt{s} \cup \big(X \times G_u \setminus \{\mathtt{s}\}\big)$, with a special sink vertex $\mathtt{t}$ added (here $G_u(I, S, Q, B)$ is the unconstrained graph defined in Definition 1);*
2. *For each source edge e in the unconstrained graph, from $\mathtt{s}$ to v', add an edge in G_c from $\mathtt{s}$ to $(f(e), v')$, with the label and weight of e;*
3. *For each non-source edge e in the unconstrained graph, from v to v', and each state x, add an edge in G_c from (x, v) to $(f(x, e), v')$; and*
4. *Add an edge of weight zero between (I, S, Q, x) and $\mathtt{t}$ whenever $x \in X_{\mathtt{t}}$.*

Proposition 2. *If $(X, f, X_{\mathtt{t}})$ represents a constraint $\mathcal{C}$, then there is a one-to-one correspondence between solutions to the WDP (I, S, Q, B) satisfying $\mathcal{C}$, and paths in G_c from $\mathtt{s}$ to $\mathtt{t}$.*

Proof. It is clear from 2 and 3 that any path in G_c from $\mathtt{s}$ to $\mathtt{t}$ corresponds to a sequence of edges $e_1, \ldots, e_m$ in G_u along which f is iteratively evaluated; the penultimate vertex in G_c must therefore be $(I, S, Q, f(e_1, \ldots, e_m))$. By the definition of the fact that f represents the constraint, a solution obeys the constraint if and only if the corresponding path in the unconstrained graph satisfies $f(e_1, \ldots, e_m) \in X_{\mathtt{t}}$, which by 4 is true if and only if the corresponding path in G_c goes from $\mathtt{s}$ to $\mathtt{t}$. Therefore there is a one-to-one correspondence between paths in G_c from $\mathtt{s}$ to $\mathtt{t}$, and solutions to the WDP satisfying the constraint.

Complexity It is obvious from the construction that the complexity of incorporating a constraint via a representation with state set X is a factor of $|X|$ worse than that in Section 4.2. Thus the implicit cost is $O(|X|\, I\, S\, Q^2 + k)$ and the explicit cost is $O(|X|\, I\, S(Q^2 + k))$.

5.3 Examples

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

Worst Case The first thing to notice is that *every* global constraint has a representation: Let X be the set of all paths originating at $\mathtt{s}$ in G_u, and define $f(x, e)$ to be the concatenation of the edge e onto the end of x, if it is defined, or the empty path otherwise. Clearly every path starting at $\mathtt{s}$ is mapped by repeated application of f to a unique element of X, namely itself. For an arbitrary set of acceptable outcomes we can therefore let $X_{\mathtt{t}}$ be the set of corresponding paths from the unconstrained graph; only these paths will be connected to the sink vertex in G_c, and so only solutions obeying our arbitrary constraint will be generated by the k-shortest paths method in G_u. This representation is not useful, however, because the number of new vertices and edges required to create its constrained solutions graph scales very poorly.

Constrained Number of Winners In order to clarify the general definitions in Section 5.2 above, in this section we give an explicit representation of the important constraint that the number of sellers allocated non-zero quantities in the global solution should lie in some range.

Suppose that our goal is to bound this number above by some value Σ_f. We proceed by letting X be the collection of sets of sellers with at most Σ_f elements, with a special element $fail$ to denote that the constraint is violated. We define the representation functions $f(e)$ and $f(x, e)$ as follows:

- If e is a source edge, and has label $q > 0$ then $f(e) := \{s_1\}$, otherwise $f(e)$ is the empty set.
- If e is not a source edge,
  - If it has zero weight, then define $f(x, e) = x$;
  - Otherwise the edge ends at a vertex of the form (i, s, q); if $x \cup \{s\} \in X$ then define $f(x, e) := x \cup \{s\}$; otherwise define $f(x, e) := fail$.

As quantities are allocated to sellers, the state keeps an accurate record of the set of sellers so far allocated non-zero quantity, transitioning to state $fail$ if the number of sellers ever gets too high. To represent an upper bound on the number of sellers it is sufficient to let $X_{\mathtt{t}} = X \setminus \{fail\}$. A *lower* bound of σ_f is representable by using $\Sigma_f = \sigma_f - 1$ in the above, and $X_{\mathtt{t}} = \{fail\}$ (only those solutions that fail to use less than or equal to $\sigma_f - 1$ sellers are acceptable). We can impose upper and lower bounds simultaneously by using the state set from the upper bound, and choosing $X_{\mathtt{t}} = \{x \in X : |x| \geq \sigma_f\}$. Because lower bounds thus have multiple representations, choosing a representation wisely is in general a tricky matter.

Monotonic Predicates The most important feature of the cardinality constraint example in Section 5.3 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 states need be no larger than $2^{|S'|}$, where S' is the set of sellers

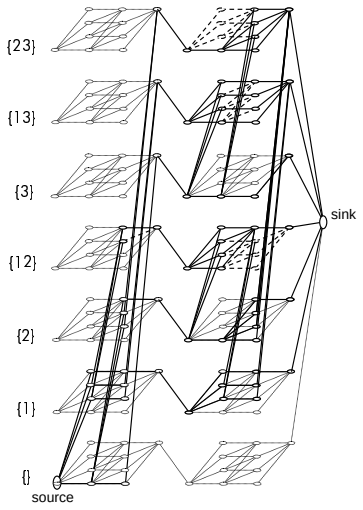

Fig. 2. A representation of the constrained graph $G_c(2, 3, 3, B, X)$ for state variable x equal to the set of suppliers so far included in the solution. Dashed edges lead to a fail state, not shown; a copy of the unconstrained graph based on concatenated copies of Figure 1 for each element $x \in X$ is shown, grayed out, for reference.

under consideration in the global constraint. For example, a representation of the constraint "Either seller 1 is included, or seller 2 is included, but not both" exists with $|X| = 4$.

Secondly, the value of the predicate is monotonic in the sense that as quantities are assigned to sellers, once the predicate is true, it will remain true in all subsequent steps. This fact sometimes gives straightforward upper bounds on the state sets. For example, for the canonical representation of "Either seller 1 is included, or seller 2 is included, but not both", the state sets will clearly never contain a state 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 in Section 5.3.

Constraints on a second metric We can impose an arbitrary constraint with respect to a second cost metric c' (i.e. one expressible as a sum of edge weights B', as in Equation 1) by maintaining it as a state variable $x = c'$: updates to the state are calculable at the edge level by the fact that the cost is a sum of edge values.

In this case X covers all reachable values of c', and so might potentially be very large. A constraint is enforced in the straightforward way, by letting $X_{\mathtt{t}}$ be the set of values of the second metric that are acceptable.

Monotonic constraints on a second metric If the second cost metric is *monotonic* in the sense that the edge weights B' in Equation 1 are all of one sign, and if the constraint is an inequality $c' \leq C$, $c' \geq C$, etc., then the representation of Section 5.3 can be simplified.

Without loss of generality, assume that the second cost metric is positive. For the case of an upper bound on a positive second cost metric, $c' \leq C$, we can restrict X to be the reachable values of c' that are also less than or equal to C, and add a failure state $fail$ as in Section 5.3. Then the constraint representation either accumulates c' in the expected way, or diverts to $fail$ if the constraint is violated: the monotonicity of c' guarantees that once broken the constraint is always broken. The set of final values $X_{\mathtt{t}}$ is equal to $X \setminus fail$.

Symmetrically, for the case of a lower bound on a positive second cost metric, $c' > C$, we can use the same state space, with the failure state replaced by a success state, *success*: the set of final values is then just $X_{\mathtt{t}} = \{success\}$.

Quantized thresholds as a solution filter In Section 5.3, most interesting secondary cost metrics will have very many possible states. Even in cases where this multiplicity makes the full constrained graph impractical to construct, we can still construct a graph whose solutions all satisfy the constraint, but which excludes some solutions that do not. This reduces the computational burden on a final filtering stage, possibly at low cost in terms of the original graph construction.

The basic idea is to quantize the secondary edge costs B' by some step parameter δ. If c' is a positive monotonic secondary cost metric, and the constraint is an upper bound $c' \leq C$, then by rounding all values of B' *down* to the nearest multiple of δ, we can assure ourselves of a smaller space of possible second metric values, while remaining confident that any path excluded corresponds to a solution violating the constraint. For a lower bound we round bid values up instead.

6 Pareto Optima Minimizing Both Cost and Risk

We have presented the objective function with respect to which solutions are ranked as being the monetary cost of procuring a particular bundle of quantities from various sellers, but another interesting example is provided by letting the cost of such an assignment be the negative logarithm of the probability of failure:

$$B_{is}(q) = -\log\left(Pr(\text{seller } s \text{ delivers}|\text{quantity} = q, \text{item} = i)\right). \tag{3}$$

Then the total cost of an assignment is a measurement of the *risk of failing to obtain all quantities required* (under the assumption of independence between deliveries). If we care only about minimizing risk of delivery, then it is equivalent to minimizing the total length of a path where $B_{is}(q)$ is defined in Equation 3. And of course we can compute *k-safest* solutions using the same overall approach that we have heretofore employed for computing k-cheapest solutions. However, since both monetary cost (expenditure) and delivery risk are important considerations in procurement auctions, it is natural to consider the bi-criteria optimization problem minimizing both expenditure and risk, and to seek an algorithm to enumerate the Pareto optimal solutions in order of one metric or the other.

Next we show how to compute the Pareto optima path set. Write the unconstrained graph from Section 4 in terms of its vertices and edges: $G = G_u(I, S, B, Q) = (V, E)$. Recall from Section 4.2 that the number of vertices and edges are bounded by $n = O(ISQ)$ and $m = O(ISQ^2)$. For each edge $e \in E$, denote its cost $c_1(e)$ and risk $c_2(e)$. For each vertex $v \in V$ and any non-negative cost value $c \leq V_{\max}$, we use $L_v[c]$ to store the minimum risk distance from the source to vertex v with cost distance exactly c. Here $V_{\max}$ is the maximum cost distance for any path from the source, corresponding to the maximum procurement cost to satisfy demand for all items. L_v is an array with length V_{max}. It is easy to initiate L_s for the source vertex $\mathtt{s}$. Next we use dynamic programming to compute $L_v[c]$ as follows:

$$L_v[c] = \min\{L_u[c - c_1(u, v)] + c_2(u, v) \mid (u, v) \in E\}.$$

The set of Pareto optima paths can be extracted from the array $L_{\mathtt{t}}$, where $\mathtt{t}$ is the destination vertex, through a linear walk of $L_{\mathtt{t}}$ from least to highest cost: the first Pareto path is given by the minimum c value with $L_v[c]$ defined; at step c, if $L_v[c]$ is defined and $L_v[c]$ is smaller than the risk distance of any of the stored Pareto paths, the path corresponding to $L_v[c]$ is Pareto optimal, and we store it.

The total running time to compute the set of Pareto optima is dominated by the dynamic programming step to compute $L_v[c]$ for all $v \in V$, $c \leq V_{\max}$, and it takes time $O(mV_{\max})$ where $m = |E| = O(ISQ^2)$. This completes the description of our algorithm, which runs in time $O(ISQ^2V_{\max})$.

In general we cannot expect to do particularly well on the Pareto enumeration problem, because the procurement auction problem minimizing both expenditure and risk belongs to the bi-criteria shortest path problem, which is NP-hard based on a reduction from the *Partition Problem* (see Garey & Johnson [11], p. 214). The general multiple-objective shortest path problem is one of the most intensively studied problems in multiple-objective combinatorial optimization; here we mention only work most relevant to our setting. We are interested in computing the Pareto optima path set in an acyclic graph with two cost metrics. Henig [12] considered the bi-criteria optimization problem using a utility function to combine both metrics. For an acyclic graph with n vertices, Warburton [13] gives a pseudo-polynomial-time exact algorithm (based on DP) with running time $O(n^2\hat{V}_{\max} \log n)$ where $\hat{V}_{\max}$ is the maximum possible path distance under both metrics. Using standard scaling and rounding techniques, it is converted into a fully-polynomial-time approximation scheme (FPTAS) to compute the *approximately efficient* Pareto optima set in time $O(n^3/\epsilon)$.

For our contributions, first, we show that the procurement auction problem minimizing both cost and risk can be modeled as bi-criteria shortest path problem in graph $G_u(I, S, Q, B)$. Second, we show that a dynamic programming approach simplifying the Walburton method can compute the set of Pareto optima in time $O(ISQ^2V_{\max})$.

7 Experiments

This section briefly summarizes the results of experiments described in detail in a preliminary paper [2] that employed an inefficient solution-generation algorithm inferior to the algorithm of the present paper.

We computed k-best solutions based on actual bids submitted to a multi-million-dollar, multi-item, multi-supplier HP material-parts procurement auction. We then explored the following questions:

1. Are the top k solutions *affordable*?
2. Are the top k solutions *diverse*?
3. Does *dominance pruning* aid multi-criteria decision problems?
4. Can our method *assign prices to bundles of side constraints*?

In all cases, our results were encouraging: The 100,000th-cheapest solution is only 0.054% more expensive than the 1st-cheapest solution, so considering k-best solutions is not prohibitively expensive for the buyer in this real procurement auction. Furthermore the k-best solutions are remarkably *diverse* in terms of how they apportion the buyer's expenditure across sellers, and moreover when we add randomly-generated volume discounts to the bids, this measure of diversity improves. When we consider the bi-criteria optimization problem in which the buyer wishes to minimize expenditure and also spread expenditure as evenly as possible across sellers, we find that the Pareto frontier of undominated solutions is small enough to admit consideration by a human decision-maker. Finally, our experiments show that our method can assign prices to bundles of side constraints, e.g., constraints on both the number of sellers included in a solution and the uniformity of expenditure across sellers.

8 Extreme Value Statistics

Our experiments have shown empirically that for a real auction, the k-cheapest solutions have very similar expenditure [2]. In this section we examine the same issue theoretically, by examining the important question of the probability, in the face of randomly distributed bids, of the cost of the k-cheapest solutions relative to the absolute cheapest. The question we therefore address is the likely tradeoff between cost and diversity in solutions.

For items $i = 1, \ldots, I$, let X_i denote a random variable representing the total cost for item i; For $Q = 1$, X_i denotes a random selection of supplier j for all units of item i with total cost p_{ij}. Let $Y = X_1 + X_2 + \ldots + X_I$, then Y denotes the total procurement cost to obtain all the items with desired number of units. Let $a_i \equiv \min X_i$, $b_i \equiv \max X_i$, then $0 < a_i \leq b_i$ for each i, and $Y_{\min} = a_1 + a_2 + \ldots + a_I, \quad Y_{\max} = b_1 + b_2 + \ldots + b_I$. Here $Y_{\min}$ denotes the minimum cost to purchase all items with desired number of units, and it is the optimal solution for other solutions to compare with. We say a solution Y is *ϵ-approximately optimal* if $Y \leq (1+\epsilon)Y_{\min}$. Here we abuse the notation and use Y to denote both the solution and its corresponding total cost.

Assumption 1 *$b_i \leq 2a_i$ for all items i.*

Given that bids are normally competitive, it is reasonable to assume that the highest unit-price for each item is at most twice as expensive as the lowest unit-price for each item. This implies that $Y_{\max} \leq 2Y_{\min}$. Suppose that items are sorted according to their minimum cost, i.e., $a_1 \leq a_2 \leq \ldots \leq a_I$, then

$$\frac{\sum_{i=1}^{\delta}(X_i - a_i)}{Y_{\min}} \leq \frac{\sum_{i=1}^{\delta} a_i}{\sum_{i=1}^{I} a_i} \leq \frac{\delta}{I}.$$

Pick δ such that $\delta \leq \epsilon I$, then any supplier selected for the first δ items together with minimum cost for items $i > \delta$ consist of a solution with total cost at most $(1+\epsilon)Y_{\min}$. Recall that $R = R(S,Q)$ is defined to be the total number of solutions for each item; for the special case of $Q = 1$, R is equal to S, the total number of suppliers.[1] The total number of ϵ-approximately optimal solutions is $R^{\delta} = R^{\epsilon I} = 2^{\epsilon I \log R}$.

Assumption 2 *There is an aggressive new entrant who matches the minimum price for a significant fraction of all the items.*

Suppose that new entrant j_0 matches the min-price for a fraction f_0 of all the items. For each of these items, there are at least two choices of suppliers with min-cost, thus the total number of min-cost solutions grows by a factor of $2^{f_0 I}$. Combining results using Assumptions 1 and 2, we obtain:

Theorem 2. *There are at least $R^{\epsilon I} 2^{f_0 I} = 2^{\epsilon I \log R + f_0 I}$ ϵ-approximately optimal solutions assuming that for each item the max-cost bid is at most twice the min-cost one, and a new entrant matches the min-cost bid of existing suppliers for a fraction f_0 of all items.*

9 Related Work

Decision support in auctions is an important problem in practice and has inspired much research, primarily on *preference elicitation* and *scenario navigation*.

Preference elicitation techniques typically represent a decision maker's preferences as a latent utility function with a specified functional form and unknown coefficients, and then repeatedly query the decision maker to refine estimates of these coefficients (e.g., by asking her to choose between two alternatives). Preference elicitation is applicable to auctions [14], and can preserve privacy and shorten bids [15] and aid uncertain decision makers [16]. However, most approaches place strong restrictions on the mathematical form of the utility function and may require auction participants to reply to exponentially many queries. Furthermore, revealed preferences may be intransitive. Our approach does not suffer from any of these difficulties.

[1] The number of solutions for item i is $\Theta(S^Q)$, thus much larger than S when $Q > 1$.

Scenario navigation typically employs mixed-integer program solvers to find price-optimal solutions under different constraints. This approach requires the buyer in a procurement auction to specify a different set of constraints for each scenario—a potentially tedious exercise. By contrast, in its simplest form our method does not require explicit modeling of side constraints.

The close relationship between combinatorial auction/exchange WDPs and generalized knapsack problems is described in [4], which furthermore exploits this connection to develop a general multi-unit combinatorial exchange WDP solver that offers attractive computational properties. Specifically, the time and memory required are pseudo-polynomial (indeed, linear) in all problem parameters *except* that they are exponential in the number of good types. Our present contribution exploits the special properties of procurement auctions to achieve good scalability in terms of *all* problem-size parameters.

Eppstein [7] surveys k-shortest paths problems and algorithms, and indeed applies his shortest paths algorithm to the solution of the 0-1 knapsack problem, generating what can be seen as a special case of our single-item graph. Encoding constraints in graphs so that a k-shortest paths algorithm generates only satisfying paths has also been explored. Villeneuve and Desaulniers [10] describe an approach based on string-matching algorithms; as noted above, this method is not suitable for our problem. Coutinho-Rodrigues et al. employ k-shortest paths computations with interactive elicitation queries to explore the Pareto frontier in bi-criteria optimization [17].

To the best of our knowledge, ours is the first systematic method of generating k-best solutions to auction WDPs. An early paper described an inefficient solution generation algorithm that required exponential time and memory [2]. The present paper makes the overall method more practical by greatly improving the computational efficiency of solution generation.

10 Conclusions

This paper has described an efficient method for computing k-cheapest solutions to procurement auction WDPs. It supports multi-sourcing, volume discounts and surcharges, and it scales pseudo-polynomially (in fact at most quadratically) with respect to all problem size parameters. Furthermore, the constrained solutions graph can accommodate many useful global hard constraints with only a modest increase in computational complexity. We have presented analytical results on the number of "reasonably cheap" solutions, complementing previous empirical results addressing the same issue. Finally, we have shown that k-safest solutions may be computed using the same framework as k-cheapest solutions, and we have presented an algorithm for computing solutions on the Pareto frontier of the bi-criteria cost/risk problem. Taken together, the contributions of this paper enable a promising approach to decision support for practical procurement auctions.

Acknowledgments

We thank Kemal Guler for early support and encouragement. Alex Zhang and Claudio Bartolini also provided valuable feedback.

References

1. Andersson, A., Tenhunen, M., Ygge, F.: Integer programming for combinatorial auction winner determination. In: Proc. 4th Int'l Conf. on Multi-Agent Systems (ICMAS). (July 2000) 39–46
2. Kelly, T., Byde, A.: Generating k-best solutions to auction winner determination problems. ACM SIGecom Exchanges **6**(1) (2006) 23–34 Presents an inefficient generation algorithm and extensive experimental results; see [18] and the present paper for an efficient algorithm.
3. Rothkopf, M.H., Pekec, A., Harstad, R.M.: Computationally manageable combinatorial auctions. Management Science **44**(8) (August 1998) 1131–1147
4. Kelly, T.: Generalized knapsack solvers for multi-unit combinatorial auctions. In: Proc. Agent Mediated E-Commerce (AMEC VI). (July 2004) Also available as HP Labs TR HPL-2004-21 and Springer LNAI 3435.
5. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
6. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall (1993)
7. Eppstein, D.: Finding the k shortest paths. SIAM Journal on Computing **28**(2) (1998) 652–673
8. Beckett, J.: The business of bidding: Reinventing auctions for better results (September 2005) `http://www.hpl.hp.com/news/2005/jul-sep/auctions.html`.
9. Dijkstra, E.W.: A note on two problems in connection with graphs. Numerische Mathematik **1** (1959) 83–89
10. Villeneuve, D., Desaulniers, G.: The shortest path problem with forbidden paths. European Journal of Operations Research **165** (2005) 97–107
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979)
12. Henig, M.: The shortest path problem with two objective functions. European Journal of Operational Research **25**(2) (1985) 281–291
13. Warburton, A.: Approximation of Pareto optima in multiple-objective, shortest-path problems. Operations Research **35**(1) (1987) 70–79
14. Cramton, P., Shoham, Y., Steinberg, R., eds.: Chapter 10. In: Combinatorial Auctions. MIT Press (January 2006)
15. Lahaie, S.M., Parkes, D.C.: Applying learning algorithms to preference elicitation. In: Proc. ACM E-Commerce Conf. (May 2004) 180–188
16. Boutilier, C., Sandholm, T., Shields, R.: Eliciting bid-taker non-price preferences in (combinatorial) auctions. In: Proc. AAAI. (2004)
17. Coutinho-Rodrigues, J., Climaco, J., Current, J.: An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. Computers & Operations Research **26** (1999) 789–798
18. Byde, A., Kelly, T.: Efficiently generating k-best solutions for procurement auctions. Technical Report HPL-2006-145, HP Labs (October 2006) Presents a more efficient generation algorithm than that of [2].