Chapter 18

A Test Suite for Combinatorial Auctions

Kevin Leyton-Brown and Yoav Shoham

Many researchers have proposed algorithms for determining the winners of general combinatorial auctions, with encouraging results. (Some of these algorithms are described in Chapter 14 of this book.) This line of research has given rise to another problem, however. In order to evaluate—and thus to improve—such algorithms, it is necessary to use some sort of test data. Unfortunately, there is little data recording the behavior of real bidders upon which such test data may be built; furthermore, even as such data becomes available, algorithmic testing will require benchmarks which permit arbitrary scaling in the problem size. It is thus necessary to generate generate artificial data that is representative of the sort of scenarios one is likely to encounter.

This chapter describes such a test suite.

18.1 Past work on testing CA algorithms

18.1.1 Experiments with human subjects

One approach to experimental work on combinatorial auctions has used human subjects. These experiments assign valuation functions to subjects, then have them participate in auctions using various mechanisms (Banks et al. 1989; Ledyard et al. 1997; DeMartini et al. 1998). Such tests can be useful for understanding how real people bid under different auction mechanisms; however, they are less suitable for evaluating the mechanisms' computational characteristics. In particular, this sort of test is only as good as the subjects' valuation functions, which in the above papers were hand-crafted. As a result, this technique does not easily permit arbitrary scaling of the problem size, a feature that is important for characterizing an algorithm's performance. In addition, this method relies on relatively naive subjects to behave rationally given their valuation functions, which may be unreasonable when subjects are faced with complex and unfamiliar mechanisms.

18.1.2 Particular problems

A parallel line of research has examined particular problems to which CA's seem well suited. For example, researchers have considered auctions for real estate (Quan 1994), the right to use railroad tracks (Brewer and Plott 1996), pollution rights (Ledyard and Szakaly 1994), airport time slot allocation (Rassenti et al. 1982) and distributed scheduling of machine time (Wellman et al. 1998). Most of these papers do not suggest holding an unrestricted general CA, presumably because of the computational obstacles. Instead, they tend to discuss alternative mechanisms that are tailored to the particular problem. None of them proposes a method of generating test data, nor does any of them describe how the problem's difficulty scales with the number of bids and goods. However, they still remain useful to researchers interested in general CA's because they give specific descriptions of problem domains to which CA's may be applied.

18.1.3 Artificial distributions

A number of researchers have proposed algorithms for determining the winners of general CA's. In the absence of test suites, some have suggested novel bid generation techniques, parameterized by number of bids and goods (Sandholm 1999; Fujishima et al. 1999; Boutilier et al. 1999; de Vries and Vohra 2003). (Other researchers have used one or more of these distributions, e.g., (Parkes 1999; Sandholm et al. 2001), while still others have refrained from testing their algorithms altogether, e.g., (Nisan 2000; Lehmann et al. 1999).) Parameterization represents a step forward, making it possible to describe performance with respect to the problem size. However, there are several ways in which each of these bid generation techniques falls short of realism, concerning the selection of which goods and how many goods to request in a bundle, what price to offer for the bundle, and which bids to combine in an XOR'ed set. More fundamentally, however, all of these approaches suffer from failing to model bidders explicitly, and from attempting to represent an economic situation with an non-economic model.

Which goods

First, each of the distributions for generating test data discussed above has the property that all bundles of the same size are equally likely to be requested. This assumption is clearly violated in almost any real-world auction: most of the time, certain goods (for which "natural" complementarities exist) will be more likely to appear together than others.

Number of goods

Likewise, each of the distributions for generating test data determines the number of goods in a bundle completely independently from determining *which* goods appear in the bundle. While this assumption may appear more reasonable, there are many domains in which the expected number of items in a bundle will be related to which goods it contains. To give an example, in an electronics domain people buying computers might tend to make long combinatorial bids, requesting monitors, printers, etc., while those buying refrigerators might tend to make short bids.)

Price

Next, there are problems with the schemes for generating price offers used by all four techniques. Prices cannot make an easy distribution hard (consider the tractable cases discussed in Chapter 13); however, if prices are not chosen carefully then an otherwise hard distribution can become computationally easy.

In Sandholm (1999) prices are drawn randomly from either $[0, 1]$ or from $[0, g]$, where g is the number of goods requested. The first method is clearly unreasonable (and computationally trivial) since price is unrelated to the number of goods in a bid—note that one bid for a large bundle and another for a small subset of the same bundle will have the same expected price. The second method is better, but has the disadvantage that mean and range are parameterized by the same variable.

In Boutilier et al. (1999) prices of bids are distributed normally with mean 16 and standard deviation 3, giving rise to the same problem as the $[0, 1]$ case above.

In Fujishima et al. (1999) prices are drawn from $[g(1-d), g(1+d)]$, $d = 0.5$. While this scheme avoids the problems described above, prices are simply additive in g and are unrelated to *which* goods are requested in a bundle, both strong and often unrealistic assumptions.

More fundamentally, Anderson et al. (2000) note a critical pricing problem that arises in several of the schemes discussed above. As the number of bids to be generated becomes large, a given short bid will be drawn much more frequently than a given long bid. Since the highest-priced bid for a bundle dominates all other bids for the same bundle, short bids end up being much more competitive. Indeed, it is pointed out that for extremely large numbers of bids a good approximation to the optimal solution is simply to take the best singleton bid for each good. One solution to this problem is to guarantee that only the first bid for each bundle will be retained. However, this solution has the drawback that it is unrealistic: different real bidders *are* likely to place bids on some of the same bundles.

The best way of addressing this problem is to make bundle prices superadditive in the number of goods they request—indeed, the fact that bidders' valuations satisfy this property is often a motivation for holding a combinatorial auction in the first place. This approach is taken by de Vries and Vohra (2003), who make the price for a bid a quadratic function of the prices of bids for subsets. (However, this pricing scheme makes it difficult to control the amount of the increase in price as a function of bundle length.) Where appropriate, the distributions presented in this chapter will include a pricing scheme that may be configured to be superadditive or subadditive in bundle length, parameterized to control how rapidly the price offered increases or decreases as a function of bundle length.

XOR bids

Finally, while most of the bid-generation techniques discussed above permit bidders to submit sets of bids XOR'ed together, they have no way of generating meaningful sets of such bids. As a consequence the computational impact of XOR'ed bids has been very difficult to characterize.

18.2 Generating realistic bids

While the lack of standardized, realistic test cases does not make it impossible to evaluate or compare algorithms, it does make it difficult to know what magnitude of real-world problems each algorithm is capable of solving, or what features of real-world problems each algorithm is capable of exploiting. This second ambiguity is particularly troubling: it is likely that algorithms would be designed *differently* if they took the features of more realistic[1] bidding into account. (As we will see, Chapter 19 discusses one such algorithm-design approach.)

18.2.1 Prices, price offers and valuations

The term “price” has traditionally been used by researchers constructing artificial distributions to describe the amount offered for a bundle. However, this term really refers to the amount a bidder is made to pay for a bundle, which is of course mechanism-specific and often not the same as the amount offered. The distributions described in this chapter aim to model bidders’ valuations, which are of course mechanism-independent. For consistency with past literature we will continue to use the term *price offer* to refer to the numeric portion of a bid; this may be understood as referring to bids placed in the VCG mechanism, in which it is a dominant strategy for bidders to make price offers equal to their true valuations (see Chapter 1). Researchers wanting to model bidding behavior in other mechanisms can still use our distributions by transforming the generated valuation according to bidders’ equilibrium strategies in the given mechanism.

18.2.2 The combinatorial auction test suite

In this chapter we present the Combinatorial Auction Test Suite (CATS), a set of distributions that attempt to model realistic bidding behavior. This suite is grounded in previous research on specific applications of combinatorial auctions, as described in Section 18.1.1 above. At the same time, all of our distributions are parameterized by number of goods and bids, facilitating the study of algorithm performance. This suite represents a move beyond previous work on modeling bidding in combinatorial auctions because we provide an economic motivation for both the contents and the valuation of a bundle, deriving them from basic bidder preferences. In particular, in each of our distributions:

- Certain goods are more likely to appear together than others.
- The number of goods appearing in the bundle is often related to which goods appear in the bundle.
- Valuations are related to which goods appear in the bundle. Where appropriate, valuations can be configured to be subadditive, additive or superadditive in the number of goods requested.
- Sets of XOR'ed bids are constructed in meaningful ways, on a per-bidder basis.

The CATS suite also contains a legacy section including all bid generation techniques described above, so that new algorithms may easily be compared to previously-published results. More information on the test suite, including executable versions of our distributions for Linux and Windows, may be found on the CATS website (CATS Website 2000).

In Section 18.3, below, we present distributions based on five real-world situations. For most of our distributions, the mechanism for generating bids involves first building a graph representing economically-motivated relationships between goods, and then using the graph to generate sets of substitutable bids, each of which requests bundles of complementary goods. Of the five real-world situations we model, the first three concern complementarity based on adjacency in a graph, while the final two concern complementarity based on correlation in time. Our first example (Section 18.3.1) models shipping, rail and bandwidth auctions. Goods are represented as edges in a nearly planar graph, with agents submitting an XOR'ed set of bids for paths connecting two nodes. Our second example (Section 18.3.2) models an auction of real estate, or more generally of any goods over which two-dimensional adjacency is the basis of complementarity. Again the relationship between goods is represented by a graph, in this case strictly planar. In Section 18.3.3 we relax the planarity assumption from the previous example in order to model arbitrary complementarities between discrete goods such as electronics parts or collectables. Our fourth example (Section 18.3.4) concerns the matching of time-slots for a fixed number of different goods; this case applies to airline take-off and landing rights auctions. In Section 18.3.5 we discuss the generation of bids for a distributed job-shop scheduling domain, and also its application to power generation auctions. Finally, in Section 18.3.7, we provide a legacy suite of bid generation techniques, including all those discussed in Section 18.1.3 above.

In the descriptions of the distributions that follow, let $rand(a, b)$ represent a real number drawn uniformly from $[a, b]$. Let $rand\_int(a, b)$ represent a random integer drawn uniformly from the same interval. With respect to a given graph, let $e(x, y)$ represent the proposition that an edge exists between nodes x and y. All of the distributions are parameterized by the number of goods (num_goods) and number of bids (num_bids).

18.3 CATS in detail

18.3.1 Paths in space

There are many real-world problems that involve bidding on paths in space. Generally, this class may be characterized as the problem of purchasing a connection between two points. Examples include truck routes (Sandholm 1993), natural gas pipeline networks (Rassenti et al. 1994), network bandwidth allocation, and the right to use railway tracks (Brewer and Plott 1996).[2] In particular, spatial path problems consist of a set of points and accessibility relations between them. Although the distribution we propose may be configured to model bidding in any of the above domains, we will use the railway domain as our motivating example since it is both intuitive and well-understood.

More formally, we will represent this domain by a graph in which each node represents a location on a plane, and an edge represents a connection between locations. The goods on auction are therefore the edges of the graph, and bids request a set of edges that form a path between two nodes. We assume that no bidder desires more than one path connecting the same two nodes, although the bidder may value each path differently.

Building the graph

The first step in modeling bidding behavior for this problem is determining the graph of spatial and connective relationships between cities. One approach would be to use an actual railroad map, which has the advantage that the resulting graph would be unarguably realistic. However, it would be difficult to find a set of real-world maps that could be said to exhibit a similar sort of connectivity and would encompass substantial variation in the number of cities. Since scaling the size of input data is of great importance to the testing of new CA algorithms, we have chosen to generate such graphs randomly. Figure 18.1 shows a representative example of a graph generated using our technique.

We begin with *num_cities* nodes randomly placed on a plane. We add

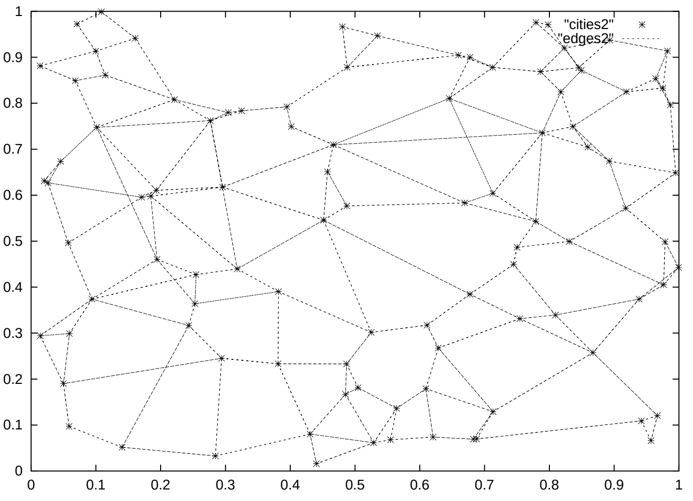

Figure 18.1: Sample “Railroad” Graph

edges to this graph, G, starting by connecting each node to a fixed number of its nearest neighbors. Next, we iteratively consider random pairs of nodes and examine the shortest path connecting them, if any. To compare, we also compute various alternative paths that would require one or more edges to be added to the graph, given a penalty proportional to distance for adding new edges. (We do this by considering a complete graph C, an augmentation of G with the new edges weighted to reflect the distance penalty.) If the shortest path involves new edges—despite the penalty—then the new edges (without penalty) are added to G, and replace the existing edges in C. This process models our simplifying assumption that there will exist uniform demand for shipping between any pair of cities, though of course it does not mimic the way new links would actually be added to a rail network. The

```
Let $num\_cities = num\_goods \div edge\_density \times 1.05$
Randomly place nodes (cities) on a unit box
Connect each node to its $initial\_connections$ nearest neighbors
While $num\_edges < num\_cities \times edge\_density$
    $C = G$
    For every pair of nodes $n_1, n_2 \in G$ where $\neg e(n_1, n_2)$
        Add an edge to $C$ of length
         $building\_penalty \cdot Euclidean\_distance(n_1, n_2)$
    End For
    Choose two nodes at random, and find the shortest path
     between them in $C$
    If shortest path uses edges that do not exist in $G$
        For every such pair of nodes $n_1, n_2 \in G$ add an edge to $G$
         with length $Euclidean\_distance(n_1, n_2)$
    End If
End For
```

Figure 18.2: Paths in Space: Graph-Building Technique

process continues until slightly more edges have been created than the number of goods in the auction being modeled. (This is achieved by the "1.05" in the first line of Figure 18.2.) The reason more edges than are necessary are created is that some edges will not ultimately appear in bids.

Our technique produces slightly non-planar graphs—graphs on a plane in which edges occasionally cross. We consider this to be reasonable, as the same phenomenon may be observed in real-world rail lines, highways, network wiring, etc. Of course, determining the "reasonableness" of a graph is a subjective task unless quantitative metrics are used to assess quality.

Generating bids

Once we have constructed a map of cities and the connectivity between them, we must next use this map to generate bids. We propose a method which generates a set of substitutable bids from a hypothetical agent's point of view. We start with the value to an agent for shipping from one city to another and with a shipping cost which we make equal to the Euclidean distance along the path. We then place XOR bids on all paths for which the agent has positive utility. The path's value is random, in (parameterized) proportion to the Euclidean distance between the chosen cities, and with a minimum value of this distance. (Bidders with smaller values would never be able to place bids.)

We aim to generate bids over a desired number of goods; however, in this distribution the number of goods (edges in the graph) is not a parameter which can be set directly. Thus we must do some extra work to ensure that we hit our target.

First, there are some generated edges that we choose to remove. Some edges are useful only for shipping directly between the two cities they connect. These edges are somewhat unrealistic; also, because they will only be selected for singleton bids, they will not increase the size of the search space. A similar argument can be made about any small disconnected component of the graph: these goods would be better modeled as a separate auction, and contribute very little to the difficulty of the winner determination problem. At some point in the bid generation process—usually before we have generated all of the bids—the total number of goods requested across all bids will meet our target. (Recall that we started out with more goods than we want to generate.) At this point we check for edges that are used only in singleton bids or isolated groups of bids, and delete those bids. Once we reach the target number of goods without deleting any bids, we delete all goods that are uninvolved in the bids we have generated so far, and continue with bid generation.

Second, it is possible that we will reach our target number of bids *without* making use of the target number of goods. In this case, we must generate a new graph, increasing the number of cities in order to increase the expected number of different goods used as a fraction of bids generated.

Note that this distribution, and indeed all others presented in this chapter, may generate slightly more than *num_bids* bids. This occurs because we only check to see whether we have generated enough bids after we have generated an entire XOR set of bids. In our experience CA optimization algorithms tend not to be highly sensitive to the number of bids, so we judged it more important to build economically sensible sets of substitutable bids. If generating a precise number of bids *is* important, it is a simple matter to remove an appropriate number of bids after generation is complete.

```
$finished = true$
Do
    While $num\_generated\_bids < num\_bids$
        Randomly choose two nodes, $n_1$ and $n_2$
        $d = rand(1, shipping\_cost\_factor)$
        $cost = Euclidean\_distance(city_1, city_2)$
        $value = d \cdot Euclidean\_distance(city_1, city_2)$
        Make XOR bids of $value - cost$ on every path from $city_1$ to
          $city_2$ having $cost < value$
        If there are more than $max\_bid\_set\_size$ such paths, bid
          on the $max\_bid\_set\_size$ paths that maximize $value - cost$.
        If number of goods receiving bids $\geq num\_goods$
            remove isolated singleton bids and isolated bid
              groups
            remove from the city map all edges that do not
              participate in any bid
        End If
    End While
    If number of goods receiving bids $< num\_goods$
        delete all bids
        delete graph
        $num\_cities = num\_cities + 1$
        run graph generation
        $finished = false$
    End While
While $\neg finished$
```

Figure 18.3: Paths in Space: Bid-Generation Technique

CATS default parameter values: $initial\_connections = 2$, $building\_penalty = 1.7$, $shipping\_cost\_factor = 1.5$, $max\_bid\_set\_size = 5$.

Multi-unit extensions: bandwidth allocation, commodity flow

This model may also be used to generate realistic data for multi-unit CA problems such as network bandwidth allocation and general commodity flow. The graph may be created as above, but with a number of units (capacity) assigned to each edge. Likewise, the bidding technique remains unchanged except for the assignment of a number of units to each bid.

18.3.2 Proximity in space

There is a second broad class of real-world problems in which complementarity arises from adjacency in two-dimensional space. An intuitive example is the sale of adjacent pieces of real estate (Quan 1994). Another example is drilling rights, where it is much cheaper for an oil company to drill in adjacent lots than in lots that are far apart. In this section, we first propose a graph-generation mechanism that builds a model of adjacency between goods, and then describe a technique for generating realistic bids on these goods. Note that in this section nodes of the graph represent the goods on auction, while edges represent the adjacency relationship.

```
Place nodes at integer vertices $(i, j)$ in a plane, where
 $1 \leq i, j \leq \lceil \sqrt(num\_goods) \rceil$
For each node $n$
    If $n$ is on the edge of the map
        Connect $n$ to as many hv-neighbors as possible
    Else
        If $rand(0,1) \leq three\_prob$
            Connect $n$ to a random set of three of its four
             hv-neighbors
        Else
            Connect $n$ to all four of its hv-neighbors
        While $rand(0,1) \leq additional\_neighbor$
            Connect $g$ to one of its d-neighbors, provided that
             the new diagonal edge will not cross another
             diagonal edge
        End While
End For
```

Figure 18.4: Proximity in Space: Graph-Building Technique

Building the graph

There are a number of ways we could build an adjacency graph. The simplest would be to place all the goods (locations, nodes) in a grid, and connect each to its four neighbors. We propose a slightly more complex method in order to permit a variable number of neighbors per node (corresponding, for example, to non-rectangular pieces of real estate). As above we place all goods on a grid, but with some probability we omit a connection between goods that would otherwise represent vertical or horizontal adjacency, and with some probability we introduce a connection representing diagonal adjacency. (We call horizontally- or vertically-adjacent nodes *hv-neighbors* and diagonally-adjacent nodes *d-neighbors*.)

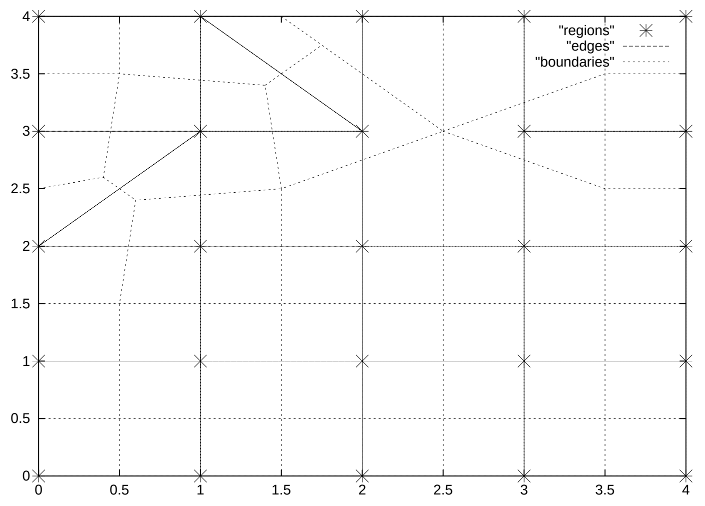

Figure 18.5: Sample Real Estate Graph

Figure 18.5 shows a sample real estate graph, generated by the technique described in Figure 18.4. Nodes of the graph are shown as asterisks, while edges are represented by solid lines. The dashed lines show a set of property boundaries that would be represented by this graph. Note that one node falls inside each piece of property, and that two pieces of property border each other iff their nodes share an edge.

Generating bids

To model realistic bidding behavior, we generate a set of common values for each good, and private values for each good for each bidder. The common value represents the appraised or expected resale value of each individual good. The private value represents the amount that a given bidder values that good, as an offset to the common value (e.g., a private value of 0 for a good represents agreement with the common value). We use these private valuations to determine both a value for a given bid and the likelihood that a bidder will request a bundle including that good. There are two additional components to each bidder's preferences: a minimum total common value for which the bidder is prepared to bid, and a budget. The former reflects the idea that a bidder may only wish to acquire goods of a certain recognized value. The latter reflects the fact that a bidder may not be able to afford every bundle that is of interest to him.

To generate bids, we start with a random good, chosen with probability weighted by a bidder's preferences. Next, we determine whether another good should be added by drawing a value uniformly from [0,1], and adding another good if this value is smaller than a threshold. This is equivalent to drawing the number of goods in a bid from a decay distribution.[3] We must now decide which good to add. Ordinarily, we select only goods from the set of nodes bordering the goods in B. However, we assign a small probability to "jumping" to a new node of the graph: in this case we add a new good selected uniformly at random from the set of goods, without the requirement that it be adjacent to a good in the current bundle B . This permits bundles requesting unconnected regions of the graph: for example, a hotel company may only wish to build in a city if it can acquire land for two hotels on opposite sides of the city.

In the case where we do not “jump”, the probability that some adjacent good n_1 will be added depends on how many edges n_1 shares with the current bundle, and on the bidder’s relative private valuations for n_1 and n_2. For example, if nodes n_1 and n_2 are each connected to B by one edge and the private valuation for n_1 is twice that for n_2, then the probability of adding n_1 to B, $p(n_1)$, is $2p(n_2)$. Further, if n_1 has 3 edges to nodes in B while n_2 is connected to B by only 1 edge, and the goods have equivalent private values, then $p(n_1) = 3p(n_2)$.

Once we have determined all the goods in a bundle we set the price offered for the bundle, which depends on the sum of common and private valuations for the goods in the bundle, and also includes a function that is superadditive (depending on our parameter settings) in the number of goods.[4] Finally, we generate additional bids that are substitutable for the original bid, imposing the constraint that each bid in the set must request at least one good from the original bid.

CATS default parameter values: $three\_prob = 1.0$, $additional\_neighbor = 0.2$, $max\_good\_value = 100$, $max\_substitutable\_bids = 5$, $additional\_location = 0.9$, $jump\_prob = 0.05$, $additivity = 0.2$, $deviation = 0.5$, $budget\_factor = 1.5$, $resale\_factor = 0.5$, and $S(n) = n^{1+additivity}$. Note that $additivity = 0$ gives additive bids, and $additivity < 0$ gives sub-additive bids.

```
For all $g, c(g) = rand(1, max\_good\_value)$
While $num\_generated\_bids < num\_bids$
    For each good, reset $p(g) =$
      $rand(-deviation \cdot max\_good\_value, deviation \cdot max\_good\_value)$
    $pn(g) = \frac{p(g)+deviation \cdot max\_good\_value}{2 \cdot deviation \cdot max\_good\_value}$
    Normalize $pn(g)$ so that $\sum_g pn(g) = 1$
    $B = \{\}$
    Choose a node $g$ at random, weighted by $pn()$, and add it to
      $B$
    While $rand(0,1) \leq additional\_location$
        Add_Good_to_Bundle($B$)
    $value(B) = \sum_{x \in B}(c(x) + p(x)) + S(|B|)$
    If $value(B) \leq 0$ on $B$, restart bundle generation for this
      bidder
    Bid $value(B)$ on $B$
    $budget = budget\_factor \cdot value(B)$
    $min\_resale\_value = resale\_factor \cdot \sum_{x \in B} c(x)$
    Construct substitutable bids.  For each good $g_i \in B$
        Initialize a new bundle, $B_i = \{g_i\}$
        While $|B_i| < |B|$
            Add_Good_to_Bundle($B_i$)
        Compute $c_i = \sum_{x \in B_i} c(x)$
    End For
    Make XOR bids on all $B_i$ where $0 \leq value(B) \leq budget$ and
      $c_i \geq min\_resale\_value$.
    If there are more than $max\_substitutable\_bids$ such bundles,
      bid on the $max\_substitutable\_bids$ bundles having the largest
      $value$
End While
```

Figure 18.6: Proximity in Space: Bid-Generation Technique

```
Routine Add_Good_to_Bundle(bundle $B$)
    If $rand(0,1) \leq jump\_prob$
        Add a good $g \notin B$ to $B$, chosen uniformly at random
    Else
        Compute $s = \sum_{x \notin B, y \in B, e(x,y)} pn(x)$
        Choose a random node $x \notin B$ from the distribution
          $\sum_{y \in B, e(x,y)} \frac{pn(x)}{s}$
        Add $x$ to $B$
    End If
End Routine
```

Figure 18.7: Proximity in Space: Add_Good_to_Bundle

Spectrum auctions

A related problem is the auction of radio spectrum, in which a government sells the right to use specific segments of spectrum in different geographical areas (Plott and Cason 1996; Ausubel et al. 1997). It is possible to approximate bidding behavior in spectrum auctions by making the assumption that all complementarity arises from spatial proximity. (This assumption would be violated, for example, if some bidders wanted to secure the right to broadcast at the same frequency in several adjacent areas.) In cases where this assumption is acceptable, our spatial proximity model is nearly sufficient for generating bidding distributions for spectrum auctions, given two modifications. First, in a spectrum auction each good may have multiple units (frequency bands) for sale. It is insufficient to model this as a multi-unit CA problem, however, if bidders have the constraint that they want the same frequency in each region.[5] Instead, the problem can be modeled with multiple distinct goods per node in the graph, and bids constructed so that all nodes added to a bundle belong to the same 'frequency'. With this method, it is also easy to incorporate other preferences, such as preferences for different types of goods. For instance, if two different types of frequency bands are being sold, one 5 megahertz wide and one 2.5 megahertz wide, an agent only wanting 5 megahertz bands could make substitutable bids for each such band in the set of regions desired (generating the bids so that the agent will acquire the same frequency in all the regions). Second, our current scheme for generating price offers may be inappropriate for the spectrum auction domain. Research indicates that while price offers will still tend to be superadditive, this superadditivity may be quadratic in the population of the region rather than exponential in the number of regions (Ausubel et al. 1997). A quadratic model may be better; see section 18.3.6. Finally, we should note some very recent work on modeling bidder behavior in spectrum auctions, albeit with restrictions to specific auction mechanisms under consideration by the FCC (Dunford et al. 2004; Porter et al. 2003).

18.3.3 Arbitrary relationships

Sometimes complementarities between goods will not be as universal as geographical adjacency, but some kind of regularity in the complementarity

```
Build a fully-connected graph with one node for each good
Label each edge from $n_1$ to $n_2$ with a weight $d(n_1, n_2) = rand(0, 1)$
```

Figure 18.8: Arbitrary Relationships: Graph-Building Technique

relationships between goods will still exist. Consider an auction of different, indivisible goods, e.g., for semiconductor parts, collectables, or distinct multi-unit goods such as the right to emit some quantity of different pollutants produced by an industrial process. In this section we describe a general way of modeling such arbitrary relationships.

Building the graph

We express the likelihood that a particular pair of goods will appear together in a bundle as being proportional to the weight of the appropriate edge of a fully-connected graph. That is, the weight of an edge between n_1 and n_2 is proportional to the probability that, having only n_1 in our bundle, we will add n_2. Weights are only proportional to probabilities because we must normalize them so that the sum of all weights from a given good sum to 1.

Generating bids

Our technique for modeling bidding is a generalization of the technique used for proximity in space (Section 18.3.2), applied to the complete graph constructed above. We choose a first good and then proceed to add goods

```
Routine Add_Good_to_Bundle(bundle $B$)
    Compute $s = \sum_{x \notin b, y \in B} d(x, y) \cdot pn(x)$
    Choose a random node $x \notin B$ from the distribution
      $\sum_{y \in B} d(x, y) \cdot \frac{pn(x)}{s}$
    Add $x$ to $B$
End Routine
```

Figure 18.9: Arbitrary Relationships: Add_Good_to_Bundle

one by one, with the probability of each new good being added depending on the current bundle. Unlike in the proximity in space distribution the graph is fully-connected here; thus there is no need for the "jumping" mechanism described above. The likelihood of adding a new good g to bundle B is proportional to $\sum_{y \in B} d(x, y) \cdot p_i(x)$. The first term $d(x, y)$ represents the likelihood (independent of a particular bidder) that goods x and y will appear in a bundle together; the second, $p_i(x)$, represents bidder i's private valuation of the good x. We implement this new mechanism by changing the routine *Add_Good_to_Bundle()*; otherwise, we use the bid-generation technique described in Figure 18.6 above.

CATS default parameter values: $max\_good\_value = 100$, $additional\_good = 0.9$, $max\_substitutable\_bids = 5$, $additivity = 0.2$, $deviation = 0.5$, $budget\_factor = 1.5$, $resale\_factor = 0.5$, and $S(n) = n^{1+additivity}$.

18.3.4 Temporal matching

We now consider real-world domains in which complementarity arises from a temporal relationship between goods. In this section we discuss matching problems, in which corresponding time slices must be secured on multiple resources. The general form of temporal matching includes m sets of resources, in which each bidder wants 1 time slice from each of $j \leq m$ sets subject to certain constraints on how the times may relate to one another (e.g., the time in set 2 must be at least two units later than the time in set 3). Here we concern ourselves with the problem in which $j = 2$, and model the problem of airport take-off and landing rights.

At present, the FAA allocates take-off and landing slots through an administrative process. However, there has been much discussion of using a combinatorial auction to perform this allocation, in which certain high-traffic airports would require airlines to purchase the right to take off or land during a given time slice. Rassenti et al. (1982) made the first study of auctions in this domain. The problem has been the topic for much other work, such as Grether et al. (1989) which includes detailed experiments and an excellent characterization of bidder behavior. The domain is also discussed in the foreword to this volume and in Chapter 20.

Single-good auctions are not adequate for this domain because an airline which buys the right for a plane to take off at one airport must also be guaranteed the right for the plane to land at its destination an appropriate amount of time later. Thus, complementarity exists between certain pairs of goods, where goods are the right to use the runway at a particular airport at a particular time. Substitutable bundles are different departure/arrival packages; therefore bundles will only be substitutable within certain limits.

Building the graph

Departing from our graph-generating approach above, we ground this example in the map of high-traffic US airports for which take-off and landing right auctions have been proposed (Grether et al. 1989). These are the four busiest airports in the United States: La Guardia International, Ronald Reagan Washington National, John F. Kennedy International, and O'Hare International. This map is shown in Figure 18.10. We chose not to use a random graph in this example because the number of bids and goods is dependent on the number of bidders and time slices at the given airports; it is not necessary to modify the number of *airports* in order to vary the problem size. Thus, $num\_cities = 4$ and $num\_times = \lfloor num\_goods/num\_cities \rfloor$.

Generating bids

Our bidding mechanism presumes that airlines have a certain tolerance for when a plane can take off and land ($early\_takeoff\_deviation$,

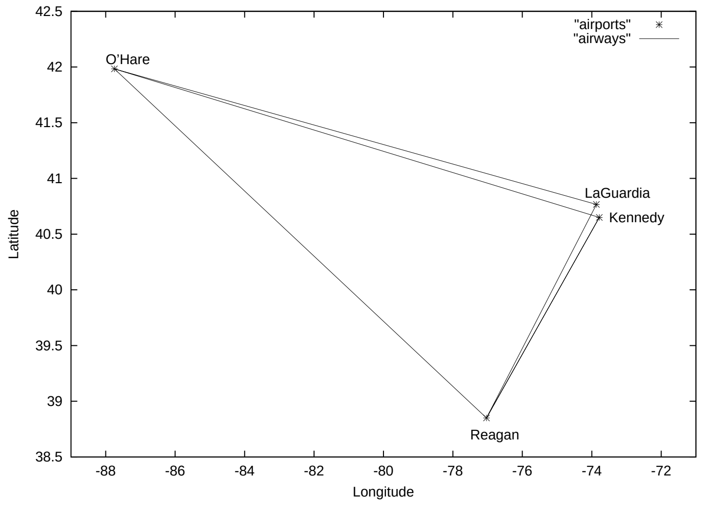

Figure 18.10: Map of Airport Locations

$late\_takeoff\_deviation$, $early\_land\_deviation$, $late\_land\_deviation$), as related to their most preferred take-off and landing times ($start\_time$, $start\_time + min\_flight\_length$). We generate bids for all bundles that fit these criteria. The value of a bundle is derived from a particular agent's utility function. We define a utility u_{max} for an agent, which corresponds to the utility the agent receives for flying from $city_1$ to $city_2$ if it receives the ideal takeoff and landing times. This utility depends on a common value for a time slot at the given airport, and deviates by a random amount. Next we construct a utility function which reduces u_{max} according to how late the plane will arrive, and how much the flight time deviates from optimal.

CATS default parameter values: $max\_airport\_value = 5$, $longest\_flight\_length = 10$, $deviation = 0.5$, $early\_takeoff\_deviation = 1$,

```
Set the average valuation for each city's airport:
  $cost(city) = rand(0, max\_airport\_value)$
Let $max\_l =$ length of longest distance between any two cities
While $num\_generated\_bids < num\_bids$
    Randomly select $city_1$ and $city_2$ where $e(city_1, city_2)$
    $l = distance(city_1, city_2)$
    $min\_flight\_length = round(longest\_flight\_length \cdot \frac{1}{max\_l})$
    $start\_time = rand\_int(1, num\_times - min\_flight\_length)$
    $dev = rand(1 - deviation, 1 + deviation)$
    Make substitutable (XOR) bids.  For $takeoff =$
      $max(1, start\_time - early\_takeoff\_deviation)$ to
      $min(num\_times, start\_time + late\_takeoff\_deviation)$
        For $land = takeoff + min\_flight\_length$ to $min(start\_time +$
          $min\_flight\_length + late\_land\_deviation, num\_times)$
            $amount\_late =$
              $min(land - (start\_time + min\_flight\_length), 0)$
            $delay = land - takeoff - min\_flight\_length$
            Bid $dev \cdot (cost(city_1) + cost(city_2)) \cdot delay\_coeff^{delay} \cdot$
              $amount\_late\_coeff^{amount\_late}$ for takeoff at time
              $takeoff$ at $city_1$ and landing at time $land$ at $city_2$
        End For
    End For
End While
```

Figure 18.11: Temporal Matching: Bid-Generation Technique

$late\_takeoff\_deviation = 2$, $early\_land\_deviation = 1$,

$late\_land\_deviation = 2$, $delay\_coeff = 0.9$, and $amount\_late\_coeff = 0.75$.

18.3.5 Temporal scheduling

Wellman et al. (1998) proposed distributed job-shop scheduling with one resource as a CA problem. We provide a distribution that mirrors this problem. While there exist many algorithms for solving job-shop scheduling problems, the distributed formulation of this problem places it in an economic context. Wellman *et al.* describe a factory conducting an auction for time-slices on some resource. Each bidder has a job requiring some amount of machine time, and a deadline by which the job must be completed. Some jobs may have additional, later deadlines which are less desirable to the bidder and so for which the bidder is willing to pay less.

Generating bids

In the CA formulation of this problem, each good represents a specific time slice. Two bids are substitutable if they constitute different possible schedules for the same job. We determine the number of deadlines for a given job according to a decay distribution, and then generate a set of substitutable bids satisfying the deadline constraints. Specifically, let the set of deadlines of a particular job be $d_1 < \cdots < d_n$ and the value of a job completed by d_1 be v_1, superadditive in the job length. We define the value of a job completed by deadline d_i as $v_i = v_1 \cdot \frac{d_1}{d_i}$, reflecting the intuition that the decrease in value for a later deadline is proportional to its 'lateness'. Like Wellman *et al.*, we assume that all jobs are eligible to be started in the first time slot. Our formulation of the problem differs in only one respect—we consider only allocations in which jobs receive continuous blocks of time. However, this constraint is not restrictive because for any arbitrary allocation of time slots to jobs there exists a new allocation in which each job receives a continuous block of time and no job finishes later than in the original allocation. (This may be achieved by numbering the winning bids in increasing order of scheduled end time, and then allocating continuous time-blocks to jobs in this order. Clearly no job will be rescheduled to finish later than its original scheduled time.) Note also that this problem cannot be translated to a trivial one-good multi-unit CA problem because jobs have different deadlines.

CATS default parameter values: $deviation = 0.5$, $prob\_additional\_deadline = 0.9$, $additivity = 0.2$, and $max\_length = 10$. Note that we propose a constant maximum job length, because the length of time a job requires should not depend on the amount of time the auctioneer makes available.

```
While $num\_generated\_bids < num\_bids$
    $l = rand\_int(1, max\_length)$
    $d_1 = rand\_int(l, num\_goods)$
    $dev = rand(1 - deviation, 1 + deviation)$
    $cur\_max\_deadline = 0$
    $new\_d = d_1$
    To generate substitutable (XOR) bids.  Do
        Make bids with price offered $= dev \cdot l^{1+additivity} \cdot d_1/new\_d$
          for all blocks $[start, end]$ where $start \geq 1$, $end \leq new\_d$,
          $end > cur\_max\_deadline$, $end - start = l$
        $cur\_max\_deadline = new\_d$
        $new\_d = rand\_int(cur\_max\_deadline + 1, num\_goods)$
    While $rand(0, 1) \leq prob\_additional\_deadline$
End While
```

Figure 18.12: Temporal Scheduling: Bid-Generation Technique

18.3.6 Legacy distributions

To aid researchers designing new CA algorithms by facilitating comparison with previous work, CATS includes the ability to generate bids according to all previous published test distributions of which we are aware, subject to the requirement that the distributions be able to generate arbitrary numbers of goods and bids. Each of these distributions may be seen as answering to three questions:

- What number of goods should be requested in a bundle?
- Which goods should be requested?
- What price should be offered for the bundle?

We begin by describing different techniques for answering each of these three questions, and then show how they have been combined in previously published test distributions.

Number of goods

Uniform: Uniformly distributed on $[1, num\_goods]$

Normal: Normally distributed with $\mu = \mu\_goods$ and $\sigma = \sigma\_goods$

Constant: Fixed at $constant\_goods$

Decay: Starting with 1, repeatedly increment the size of the bundle until $rand(0, 1)$ exceeds α

Binomial: Request n goods with probability $p^n(1-p)^{num\_goods-n}\binom{num\_goods}{n}$

Exponential: Request n goods with probability $C\exp(-n/q)$

Which goods

Random: Draw n goods uniformly at random from the set of all goods, without replacement[6]

Price offer

Fixed Random: Uniform on $[low\_fixed, hi\_fixed]$

Linear Random: Uniform on $[low\_linearly \cdot n, hi\_linearly \cdot n]$

Normal: Draw from a normal distribution with $\mu = \mu\_price$ and $\sigma = \sigma\_price$

Quadratic[7]**:** For each good k and each bidder i set the value $v_k^i = rand(0,1)$; then i's price offer for a set of goods S is $\sum_{k \in S} v_k^i + \sum_{k,q} v_k^i v_q^i$

18.3.7 Previously published distributions

The following is a list of the distributions used in all published tests of which we are aware. In each case we describe first the method used to choose the number of goods, followed by the method used to choose the price offer. In all cases the 'random' technique was used to determine which goods should be requested in a bundle. Each case is labeled with its corresponding CATS legacy suite number; very similar distributions are given similar numbers and identical distributions are given the same number.

[**L1**] *Sandholm*: Uniform, fixed random with $low\_fixed = 0$, $hi\_fixed = 1$

[**L1a**] *Anderson* et al.: Uniform, fixed random with $low\_fixed = 0$, $hi\_fixed = 1000$

[**L2**] *Sandholm*: Uniform, linearly random with $low\_linearly = 0$, $hi\_linearly = 1$

[**L2a**] *Anderson* et al.: Uniform, linearly random with $low\_linearly = 500$, $hi\_linearly = 1500$

[**L3**] *Sandholm*: Constant with $constant\_goods = 3$, fixed random with

$low\_fixed = 0$, $hi\_fixed = 1$

[L3] *deVries and Vohra*: Constant with $constant\_goods = 3$, fixed random with $low\_fixed = 0$, $hi\_fixed = 1$

[L4] *Sandholm*: Decay with $\alpha = 0.55$, linearly random with $low\_linearly = 0$, $hi\_linearly = 1$

[L4] *deVries and Vohra*: Decay with $\alpha = 0.55$, linearly random with $low\_linearly = 0$, $hi\_linearly = 1$

[L4a] *Anderson* et al.: Decay with $\alpha = 0.55$, linearly random with $low\_linearly = 1$, $hi\_linearly = 1000$

[L5] *Boutilier* et al.: Normal with $\mu\_goods = 4$ and $\sigma\_goods = 1$, normal with $\mu\_price = 16$ and $\sigma\_price = 3$

[L6] *Fujishima* et al.: Exponential with $q = 5$, linearly random with $low\_linearly = 0.5$, $hi\_linearly = 1.5$

[L6a] *Anderson* et al.: Exponential with $q = 5$, linearly random with $low\_linearly = 500$, $hi\_linearly = 1500$

[L7] *Fujishima* et al.: Binomial with $p = 0.2$, linearly random with $low\_linearly = 0.5$, $hi\_linearly = 1.5$

[L7a] *Anderson* et al.: Binomial with $p = 0.2$, linearly random with $low\_linearly = 500$, $hi\_linearly = 1500$

[L8] *deVries and Vohra*: Constant with $constant\_goods = 3$, quadratic

Parkes (1999) used many of the test sets described above (particularly those described by Sandholm and Boutilier *et al.*), but tested with fixed numbers of goods and bids rather than scaling these parameters.

Since the publication of Leyton-Brown et al. (2000), the CATS distributions have also been widely used, e.g., (Sandholm et al. 2001; Gonen and Lehmann 2001; Gonen and Lehmann 2000; Holte 2001; Schuurmans et al. 2001; Kastner et al. 2002; Zurel and Nisan 2000).

18.4 Tuning distributions

Our main goal in this work has been to generate realistic artificial test data, even if this data turns out not to be as computationally difficult as other, less realistic benchmarks. After all, combinatorial auction researchers should care more about optimizing WDP algorithm performance on realistic problems than on tackling arbitrary set packing problems that cannot easily be interpreted as combinatorial auctions. This should not make us entirely unconcerned with the hardness of our distributions, however. In this section we present evidence about how hard the CATS distributions really are, and show how to tune them so that the hardest possible instances are generated from a given distribution.

18.4.1 Removing dominated bids

For the WDP, it is known that problems become harder as the number of goods and bids increases.[8] For this reason, researchers have traditionally reported the performance of their WDP algorithms in terms of the number of bids and goods of the input instances. While it is easy to fix the number of goods, holding the number of bids constant is not as straightforward as it might seem. Most special-purpose algorithms make use of a polynomial-time preprocessing step which removes bids that are strictly dominated by one other bid. More precisely, bid i is dominated by bid j if the goods requested by i are a (non-strict) superset of the goods requested by j, and the price offer of i is smaller than or equal to the price offer of j. (This is similar in flavor to the use of arc-consistency as a preprocessing step for a CSP or weighted CSP problem.) It is thus possible for the size of problems given as input to the WDP algorithm to vary even if all generated instances had the same number of bids.

It is not obvious whether this domination procedure ought to remove many bids, or whether the relationship between the average number of non-dominated bids and total bids ought to vary substantially from one distribution to another, so we set out to measure this relationship. Figure 18.4.1 shows the number of non-dominated bids as a function of the total number of bids generated. In these experiments, with each line representing

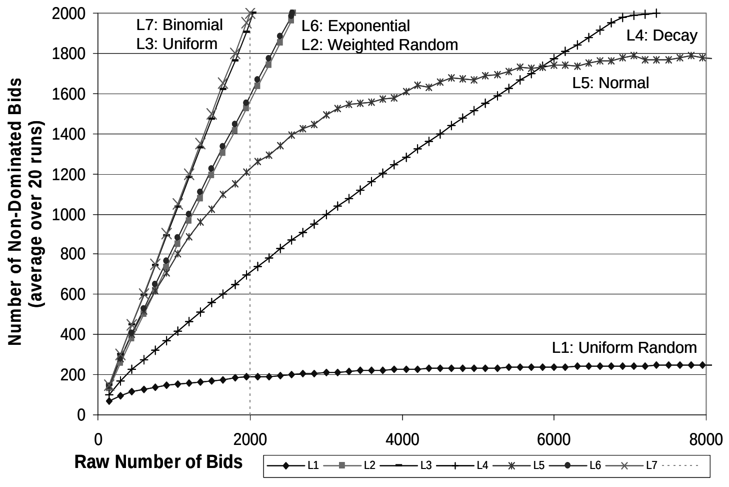

Figure 18.13: Non-Dominated Bids vs. Raw Bids

an average over 20 runs, bids were generated for an auction with 64 goods, and the program stopped after 2000 non-dominated bids had been made. We observe that some of the legacy distributions are considerably more likely than others to generate non-dominated bids; we do not show the CATS distributions in this graph as all five generated virtually no dominated bids.

Of course, many other polynomial-time preprocessing steps are possible, e.g., a check for bids that are dominated by a pair of other bids. Indeed, sophisticated solvers such as CPLEX employ many, much more complex preprocessing steps before initiating branch-and-bound search. Our own experience with algorithms for the WDP has suggested that other polynomial-time preprocessing steps offer much poorer performance in terms of the number of bids discarded in a given amount of time. In any case, the results above suggest that strict domination checking should not be disregarded, since distributions differ substantially in the ratio between the number of non-dominated bids and the raw number of bids.

For this reason, the CATS software has the ability to generate instances for all CATS and legacy distributions with a specified number of *non-dominated* bids: the software iteratively generates bids and removes dominated bids until the specified target is reached. Observe that if we want to be able to generate any given number of non-dominated bids then we will be unable to use the distributions L1 and L5, because they often fail to generate a target number of non-dominated bids even after millions of bids were created. As an aside, this observation helps to explain why L1 and L5 have been found empirically easy by other researchers, and suggest that they are a poor choice for computational benchmarking.

18.4.2 Sampling parameters

In our original paper on CATS (Leyton-Brown et al. 2000), we suggested default values for the parameters of each generator. These defaults represented reasonable, hand-tuned choices for the parameter values. However, the parameter space is large and the computational characteristics of the different CATS distributions vary substantially throughout this space. An alternative ensures that the whole parameter space is explored: reasonable ranges for each parameter are established, and then each time an

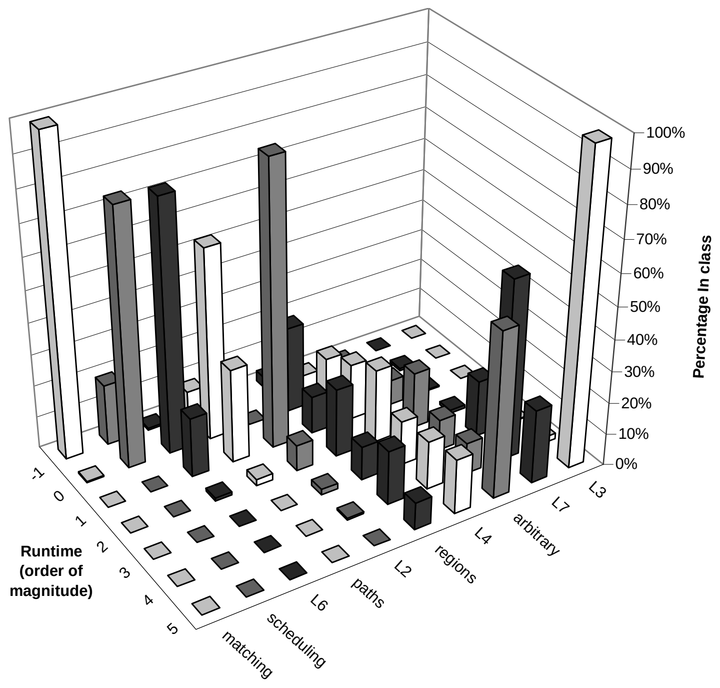

Figure 18.14: Gross Hardness

instance is generated, these ranges are sampled uniformly at random. Version 2.0 of the CATS software supports this sort of parameter sampling.

18.4.3 Making CATS harder

There has been discussion in the combinatorial auctions literature about whether CATS is computationally hard (see, e.g., (Gonen and Lehmann 2000; Sandholm et al. 2001)). We performed tests on both CATS and legacy distributions with ILOG's CPLEX 7.1 solver,[9] sampling parameters as described above. Figure 18.14 shows the results of 500 runs for each distribution on problems with 256 goods and 1000 non-dominated bids, indicating the number of instances with the same order-of-magnitude runtime—i.e., $\lfloor \log_{10}(\text{runtime}) \rfloor$. We ran these experiments on a cluster of Pentium III Xeon 550 Mhz machines with 4 GB of RAM each, and spent over a CPU-year gathering the data.

We can see that several of the CATS distributions are quite easy for CPLEX, and that others vary from easy to hard. It is interesting that most distributions had instances that varied in hardness by several orders of magnitude, despite the fact that all instances had the same problem size. This gives rise to the question of whether we can tune CATS so that in addition to generating "realistic" instances, it also generates the hardest possible instances. It turns out that the answer is yes: in Chapter 19 we show that even the easiest CATS distributions can be made orders of magnitude harder.

Our interest in generating the hardest possible instances notwithstanding, we should not be discouraged by the fact that some CATS distributions are computationally easy. On the contrary, this evidence suggests that realistic bidding patterns may often lead to much more tractable winner determination problems than the hardest unrealistic distributions such as "uniform" (L3). This is good news for those who hope to run practical combinatorial auctions.

18.5 Conclusion

In this chapter we introduced CATS, a test suite for combinatorial auction optimization algorithms. The distributions in CATS represent a step beyond earlier CA testing techniques because they are economically-motivated and model real-world problems. We hope that CATS will continue to facilitate the development and evaluation of new CA optimization algorithms.

Acknowledgments

This chapter is based on work first presented in Leyton-Brown et al. (2000) and Leyton-Brown et al. (2002). We would like to acknowledge Mark Pearson, who was involved in the creation of the original test suite and co-wrote Leyton-Brown et al. (2000); Galen Andrew, who has helped to maintain the test suite since the publication of Leyton-Brown et al. (2000) and was responsible for many of the enhancements described here; and Eugene Nudelman, who was our coauthor on Leyton-Brown et al. (2002).

Notes

[1]There does exist a body of previous work characterizing hard cases for weighted set packing, which is of course equivalent to the combinatorial auction problem. Real-world bidding is likely to exhibit various regularities, however, as discussed throughout this chapter. A data set designed to include the same

regularities may be more useful for predicting the performance of an algorithm in a real-world combinatorial auction.

[2]Electric power distribution is a frequently discussed real world problem which is superficially similar to the problems discussed here. However, many of the complementarities in this domain arise from physical laws governing power flow in a network. Consideration of these laws becomes very complex in networks of interesting size. Also, because these laws are taken into account during the construction of power networks, the networks themselves are difficult to model using randomly generated graphs. For these reasons, we have not attempted to model this domain.

[3]We use Sandholm's (Sandholm 1999) term "decay" here, though the distribution goes by various names—for a description of the distribution please see Section 18.3.6. There are two reasons we use a decay distribution here. First, we expect that more bids will request small bundles than large bundles. Second, we require a distribution where the expected bundle size is relatively insensitive to changes in the total number of goods.

[4]Recall the discussion in Section 18.1.3 motivating the use of superadditive valuations.

[5]To see why this cannot be modeled as a multi-unit CA, consider an auction for three regions with two units each, and three bidders each wanting one unit of two goods. In the optimal allocation, b_1 gets 1 unit of g_1 and 1 unit of g_2, b_2 gets 1 unit of g_2 and 1 unit of g_3, and b_3 gets 1 unit of g_3 and 1 unit of g_1.

In this example there is no way of assigning frequencies to the units so that each bidder gets the same frequency in both regions.

[6]Although in principle the problem of *which* goods to request could be answered in many ways, all legacy distributions of which we are aware use the Random technique.

[7]de Vries and Vohra (2003) briefly describe a more general version of this price offer scheme, but do not describe how to set all the parameters (e.g., defining which goods are complementary); hence we do not include it here. Quadratic price offers may be particularly applicable to spectrum auctions; see Ausubel et al. (1997).

[8]Recall the exception discussed in Section 18.1.3: when distributions favor small bundles and lack sufficiently superadditive pricing then the problem generally becomes easier as the number of bids grows very large.

[9]CPLEX has become faster with every version released, and so newer versions of CPLEX will most likely exceed the performance reported here. We reran a subset of these problems using CPLEX 8.0, but found that the qualitative shape of the distribution was unchanged. Due to the investment of machine time that would have been required to regenerate Figure 18.14 we elected not to rerun the entire dataset, and report only our CPLEX 7.1 results here.

References

Anderson, Arne, Mattias Tenhunen and Fredrik Ygge (2000). Integer programming for combinatorial auction winner determination. *ICMAS* (pp. 39–46).

Ausubel, Lawrence M., Peter Cramton, R. Preston McAfee and John McMillan (1997). Synergies in wireless telephony: Evidence from the broadband PCS auctions. *Journal of Economics and Management Strategy*, *6*(3), 497–527.

Banks, Jeffrey S., John O. Ledyard and David P. Porter (1989). Allocating uncertain and unresponsive resources: An experimental approach. *RAND Journal of Economics*, *20*, 1–23.

Boutilier, Craig, Moisés Goldszmidt and Bikash Sabata (1999). Sequential auctions for the allocation of resources with complementarities. *IJCAI*.

Brewer, Paul J. and Charles R. Plott (1996). A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *International Journal of Industrial Organization*, *14*, 857–886.

CATS Website (2000). http://robotics.stanford.edu/CATS.

de Vries, Sven and Rakesh Vohra (2003). Combinatorial auctions: A survey. *INFORMS Journal on Computing*, *15*(3).

DeMartini, Christine, Anthony M. Kwasnica, John O. Ledyard and David Porter (1998). *A new and improved design for multi-object iterative auctions* (Technical Report Social Science Working Paper 1054). California Institute of Technology, Pasadena.

Dunford, Melissa, Karla Hoffman, Dinesh Menon, Rudy Sultana and Thomas Wilson (2004). Testing linear pricing algorithms for use in ascending combinatorial auctions. Working paper, George Mason University.

Fujishima, Yuzo, Kevin Leyton-Brown and Yoav Shoham (1999). Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*.

Gonen, Rica and Daniel Lehmann (2000). Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. *ACM Conference on Electronic Commerce*.

Gonen, Rica and Daniel Lehmann (2001). *Linear programming helps solving large multi-unit combinatorial auctions* (Technical Report TR-2001-8). Leibniz Center for Research in Computer Science.

Grether, David M., R. Mark Isaac and Charles R. Plott (1989). *The* *allocation of scarce resources: Experimental economics and the problem of allocating airport slots*. Boulder, CO: Westview Press.

Holte, Robert C. (2001). Combinatorial auctions, knapsack problems, and hill-climbing search. *Canadian Conference on AI.*

Kastner, Ryan, Christina Hsieh, Miodrag Potkonjak and Majid Sarrafzadeh (2002). On the sensitivity of incremental algorithms for combinatorial auctions. UCLA CS Tech. Report 020000.

Ledyard, John O., David Porter and Antonio Rangel (1997). Experiments testing multiobject allocation mechanisms. *Journal of Economics & Management Strategy*, *6*(3), 639–675.

Ledyard, John O. and Kristin Szakaly (1994). Designing organizations for trading pollution rights. *Journal of Economic Behavior and Organization*, *25*, 167–196.

Lehmann, Daniel, Liadan I. O'Callaghan and Yoav Shoham (1999). Truth revalation in rapid, approximately efficient combinatorial auctions. *ACM Conference on Electronic Commerce.*

Leyton-Brown, Kevin, Eugene Nudelman and Yoav Shoham (2002). Learning the empirical hardness of optimization problems: The case of combinatorial auctions. *CP.*

Leyton-Brown, Kevin, Mark Pearson and Yoav Shoham (2000). Towards a universal test suite for combinatorial auction algorithms. *ACM EC.*

Nisan, Noam (2000). Bidding and allocation in combinatorial auctions. *ACM Conference on Electronic Commerce.*

Parkes, David C. (1999). iBundle: An efficient ascending price bundle auction. *ACM Conference on Electronic Commerce.*

Plott, Charles R. and Timothy N. Cason (1996). EPA's new emissions trading mechanism: A laboratory evaluation. *Journal of Environmental Economics and Management*, *30*, 133–160.

Porter, David, Stephen Rassenti, Anil Roopnarine and Vernon Smith (2003). Combinatorial auction design. PNAS Early Edition, contributed by Vernon Smith.

Quan, Daniel C. (1994). Real estate auctions: A survey of theory and practice. *Journal of Real Estate Finance and Economics*, *9*, 23–49.

Rassenti, Stephen J., Stanley S. Reynolds and Vernon Smith (1994). Cotenancy and competition in an experimental auction market for natural gas pipeline networks. *Economic Theory*, *4*, 41–65.

Rassenti, Stephen J., Vernon Smith and Robert L. Bulfin (1982). A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, *13*, 402–417.

Sandholm, Tuomas (1993). An implementation of the contract net protocol based on marginal cost calculations. *Proceedings of AAAI-93* (pp. 256–262).

Sandholm, Tuomas (1999). An algorithm for optimal winner determination in combinatorial auctions. *IJCAI-99*.

Sandholm, Tuomas, Subhash Suri, Andrew Gilpin and David Levine (2001). CABOB: A fast optimal algorithm for combinatorial auctions. *IJCAI*.

Schuurmans, Dale, Finnegan Southey and Robert C. Holte (2001). The exponentiated subgradient algorithm for heuristic boolean programming. *IJCAI-01*.

Wellman, Michael P., William R. Walsh, Peter R. Wurman and Jeffrey K. MacKie-Mason (1998). Auction protocols for distributed scheduling. Games and Economic Behavior.

Zurel, Edo and Noam Nisan (2000). An efficient approximate allocation algorithm for combinatorial auctions. *ACM Conference on Electronic Commerce*.