ABSTRACT

HSIAO, PING-LIN. Visualization for Combinatorial Auctions. (Under the direction of Dr. Christopher G. Healey.)

Visualization converts raw data into meaningful graphical images that allow users to rapidly identify and explore values, trends, and patterns in their datasets. Visualization techniques often apply spatial relationships to abstract data to represent the relationships between data elements in a meaningful way. Our specific interest in this thesis is visualizing combinatorial datasets that contain all subsets of a collection of base elements. The combinatorial datasets we study encapsulate results from combinatorial auctions where multiple items are sold simultaneously to multiple bidders. Different bidding strategies and the relationships between them are revealed in our visualizations.

We propose a new 2D scheme for concisely visualizing combinatorial datasets. The visualization displays concentric rings composed of arcs, with each base element subset mapped to a single arc. Equal sized subsets are placed on a common ring. The outermost ring contains subsets of size one. Interior rings contain larger subsets. The rings are positioned to try to overlap common base elements as much as possible. This allows viewers to search a local region of the visualization to study the behavior of a given base element (i.e., a given item offered within the combinatorial auction).

Additional visual features, including motion, color, and texture are applied to represent auction attributes like the identity of a bidder, which bids win in a particular stage of the auction, and so on. Our visualizations provide viewers with an efficient and effective way to observe how an auction progresses.

Visualization for Combinatorial Auctions

by

PING-LIN HSIAO

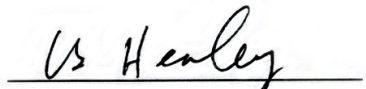
A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER SCIENCE

Raleigh, North Carolina

2006

APPROVED BY:



Chair of Advisory Committee

BIOGRAPHY

Ping-Lin Hsiao was born May 20, 1979 in Taipei City, Taiwan. He received a Bachelor of Science degree in Computer Science and Engineering from Yuan Ze University, Chung-Li, Taoyuan, Taiwan, after which he was in Alternative Service for two years. Ping-Lin is currently enrolled in the computer science masters program at North Carolina State University. He will continue his studies at North Carolina State University in the computer science doctoral program.

ACKNOWLEDGEMENTS

I want to thank many people for helping while I was writing this thesis. First, I would like to thank my chair committee member Dr. Christopher Healey who provides his professional advice on the visualization and this thesis. I also want to thank the other two committee members, Dr. Peter Wurman and Dr. Alan Tharp, for spending their time reading my work.

There are a number of my peers I also need to thank. A special thanks goes to Brent and Laura who spend a lot of time looking at some of my drafts and give me positive feedback and suggestions as they still have their own work to do. Many thanks go to Sarat who shares the computer with me to run programs for days. Thanks also goes to the rest of members in the Knowledge Discovery Laboratory, Thomas, Loyd, and Amit, who have all supported me during my time at NCSU.

Finally, I want to thank my family for their strong faith in my abilities and support me attaining higher education in the United States.

Contents

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**
1.1 Introduction to Visualization . . . 1
1.2 Problems . . . 2
1.2.1 Combinatorial Datasets . . . 3
1.3 Project Design . . . 5
1.4 Thesis Organization . . . 6

2 Visualization **7**
2.1 Visualization History . . . 7
2.2 How Visualization Helps . . . 8
2.3 Categories of Visualization . . . 10
2.4 Information Visualization . . . 11
2.4.1 Overview + Detail . . . 12
2.4.2 Focus + Context . . . 16

3 Combinatorial Auction **21**
3.1 Combinatorial Auction . . . 21
3.2 Iterative Combinatorial Auctions . . . 22

4 Method **28**
4.1 Layout . . . 28
4.1.1 Rings and Arcs . . . 29
4.1.2 Layout Problem . . . 30
4.1.3 Layout Algorithm . . . 31

5 Practical Application **38**
5.1 Testbed . . . 38
5.1.1 Bundle Prices . . . 39
5.1.2 Agents' Allocation of Attentions . . . 40

5.1.3 Competitive Allocations 41
5.1.4 Times of Inflection Points 42

6 Discussion and Future Work **45**

Bibliography **48**

List of Figures

1.1 The Hasse diagram and the Venn diagram 4

2.1 The tree diagram and corresponding tree-maps 13
2.2 The geographic information for centenial campus of NCSU in googlemap . . . 15
2.3 fisyeye 17
2.4 Hyperbolic Browser 18
2.5 The layout of cone tree and cam tree 19

3.1 four buyers' prices for three objects 23
3.2 Bundle prices over time 25
3.3 Changes for competitive allocations 26
3.4 information of six of ten rounds in a combinatorial auction 27

4.1 The layout showing the family of subsets of {A,B,C,D,E} 29
4.2 A completely concise layout and an incomplete one 30
4.3 The possible mappings between subsets and arcs in each ring 33
4.4 Comparison for our visualization to the optimal one 35
4.5 Compare our visualization and the brute-force's one 36

5.1 Execution snapshots 43
5.1 Execution snapshot continued 44

List of Tables

4.1 Comparisons of required computations for combinatorial datasets from $\{A, B, C\}$, $\{A, B, C, D\}$, and $\{A, B, C, D, E\}$. 34

Chapter 1

Introduction

1.1 Introduction to Visualization

With more powerful computing tools, scientists are able to gather more detailed data than ever in their research, making efficient analysis even more challenging for scientific communities. Visualization holds the promise to help a great deal with this problem. Visualization is a process of transforming numerical data into graphical representations. It applies vision research to practical problems of data analysis, just as engineers apply physics and electricity to practical problems of building cars and machines. Visualizations can provide user interactions within a graphical environment where scientists are able to rapidly switch observations between a high level global overview and detailed local views that display more information about individual data elements. This is a great benefit in situations where fast analysis is critical. Scientists in astronomy, molecular biology, and medicine have confirmed the value of visualization in their research [GFG+94]. Other uses of visualization include military, geographic information

system, land and satellite weather information, and fluid flow applications. Carefully designed visualizations take advantage of the strengths of the low-level human visual system, which can identify certain visual features rapidly and accurately [Tri85, Wol94]. For example, we can design visualizations by selecting salient visual features based on what is known about human perception, then mapping attributes of a dataset to these visual features.

This process of visualization can be formulated as follows. A dataset *D* contains *n* individual elements where $D = \{e_1, e_2, ..., e_n\}$. In a multidimensional dataset, each data element has a set of attributes $A = \{A_1, A_2, ..., A_m\}$, $m \geq 2$. That is, each data element e_i in *D* encodes m values, one for each attribute. Thus, $e_i = \{a_{i,1}, a_{i,2}, ..., a_{i,m}\}$, where $a_{i,j} \in A_j$. To transfer the dataset into a visual form, a mapping between attributes and visual features can be chosen. The mapping is defined as $M = (V, \Phi)$, where $V = \{V_1, V_2, ..., V_m\}$ is a set of *m* selected visual features, and $\Phi_j : A_j \rightarrow V_j$ defines how to map the attribute values in A_j to the displayable values of V_j [Hea01].

1.2 Problems

A good visualization is an effective selection of *M*, which produces images that support rapid, accurate, and effortless exploration and analysis. Critical factors for achieving this goal are the choices of visual features and the choices of mapping functions to map attributes to visual features. The values of n and m, the size of the datasets and the number of of attributes, as well as the nature of the relationships between the elements, affect the difficulty of this goal. For a large dataset, it is important to build a concise and easily explored structure to represent

all the data. For data with many attributes, the main task is to choose an appropriate mapping function that harnesses the strengths of human perception and produces accurate images.

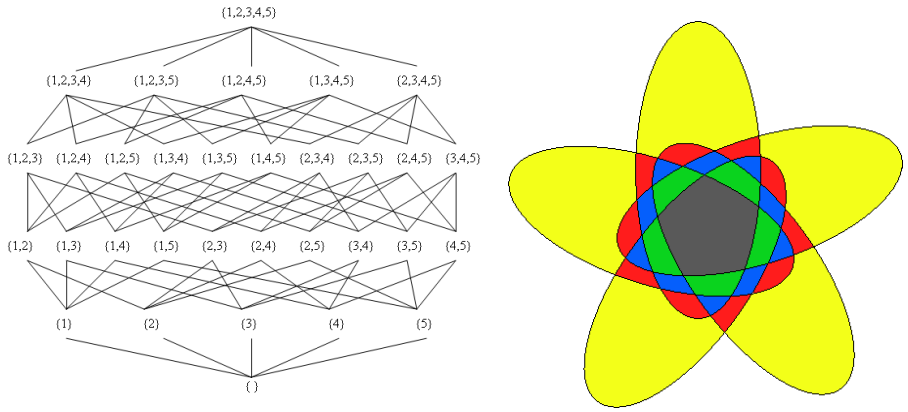
1.2.1 Combinatorial Datasets

A combinatorial dataset is an example of a dataset which often combines a large amount of data. Combinatorial datasets are used in scientific areas, such as statistics, e-commerce, and biology. Such a dataset is the set of subsets of an original data sequence. Its cardinality is determined by the cardinality of the sequence. For a dataset D of size n, a combinatorial dataset, C, is the set of all D_i for $0 \leq i \leq n$, where $D_i \subseteq D$. If the original dataset has *n* elements, the cardinality of the combinatorial dataset will be $\sum_{k=0}^{n} \binom{n}{k}$. By the binomial theorem, $\sum_{k=0}^{n} \binom{n}{k} r^k = (1+r)^n$, so $r = 1$ gives $\sum_{k=0}^{n} \binom{n}{k} = 2^n$. The following characteristics present challenges for creating a good visualization for a combinatorial dataset.

1. The cardinality of C grows exponentially as n increases, i.e., $|C| = \sum_{k=0}^{n} \binom{n}{k} = 2^n$.

2. The exponential growth of the cardinality also corresponds to more complex relationships between subsets.

3. Since the number of subsets of size i is $|S_i| = \binom{n}{i}$, the cardinality of the subset is not constant as i increases. It is not even monotonically increasing or decreasing. For $0 \leq i < \lceil \frac{n}{2} \rceil$, $|S_i|$ is increasing, and for $\lceil \frac{n}{2} \rceil \leq i < n$, $|S_i|$ is decreasing. This feature makes traditional hierarchical visualization techniques such as cone trees difficult to apply.



(a) (b)

Figure 1.1: (a) The Hasse diagram of the family of subsets of $\{1,2,3,4,5\}$. (b) The Venn diagram of the same set as in (a).

Two types of diagrams commonly used to represent combinatorial datasets are *Hasse diagrams* [MCG05] and *Venn diagrams* [Ven80]. Figure 1.1a illustrates the Hasse diagram of the family of subsets of $\{1, 2, 3, 4, 5\}$. Same sized subsets are drawn on the same row with size n on top going down to size 0 on the bottom. For each $0 \leq i < n$, each set of size i emits edges to its supersets of size $i + 1$. In other words, each edge extends from a set to a superset, representing a belongs to relationship between consecutive levels. One disadvantage of a Hasse diagram is that as the number of nodes increases, the number of edges can become too large to be tracked by a viewer, making it difficult to understand the subset relationships. Figure 1.1b shows the Venn diagram for the same combinatorial dataset. No edges are drawn; rather, the Venn diagram uses regions to represent subsets and the position of a region to represent relationships between subsets, making better use of space than Hasse diagrams. Different colors are important to help viewers maintain visual separations of nodes in different levels. In this example, yellow, red, and blue represent subsets of size one, two, and three, respectively.

In the Venn diagram, however, the area for each subset is relatively non-uniform in size. The area in the core and areas near the outside are much larger than the intermediate areas. Again, as n increases regions will shrink in size, possibly to a point that makes them impractical for certain applications.

1.3 Project Design

This thesis proposes a new 2D scheme for visualizing multidimensional combinatorial datasets. The thesis goals are as follows:

1. Design a visualization structure applicable to multi-dimensional combinatorial datasets.

2. Provide an efficient and effective visual environment for rapid exploration and analysis.

We chose to maintain the concentric layout of the Venn diagram to preserve the useful spatial relationships it provides. A concentric ring structure is used. Subsets of the same size are distributed in a common ring. Each ring is separated into fixed-width arcs to position its subsets. We try to map subsets with common data elements to arcs that overlap across neighboring rings as much as possible to maintain the relationships between subsets of different sizes. Visual features of each arc are varied to represent subset attributes.

A proxy bidding combinatorial auction, which uses combinatorial datasets to represent its state, acted as a practical testbed for our techniques. Combinatorial auctions are those auctions in which bidders can put bids on combinations of items, rather than just individual items. The combinations of items are called "bundles." There are multiple choices of assigning bundles

to bidders as different bidders bid on common bundles. The auctioneer tries to find the one which maximizes the sum of the bids for all items (called the surplus). Of course, any single item can not be assigned to more than one bidder. In proxy bidding combinatorial auctions, a buyer provides a maximum price for each possible bundle to an agent and lets the agent act on his behalf. In this environment, the proxy agent tries to maximize his profit, which is the difference between the price he pays the auctioneer for a bundle, versus the price his bidder has agreed to pay. Related research in combinatorial auctions includes bidding strategies and winner determination.

In our testbed, individual bundles are mapped to arcs, and the space inside the arcs is used to visualize additional information about the agents bidding on that bundle. Visual features, such as color, texture, transparency and blur, are used to represent important data values within each bundle. The entire auction process is recorded in discrete steps. Animations are used to visualize the transitions between the steps.

1.4 Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 describes relevant research in visualization and human perception. Chapter 3 provides some background about the testbed, the proxy bidding combinatorial auction. Chapter 4 discusses our algorithm to reduce the search space and generate a well-organized layout. In Chapter 5, the details of applying visual features to combinatorial auctions are discussed. Chapter 6 presents the conclusion and future work.

Chapter 2

Visualization

This chapter provides some background about visualization. It includes a short history and two principal areas of visualization. It discusses research involved with these areas and provides explanations of the problems they address.

2.1 Visualization History

Visualization converts raw data into meaningful graphical representations which allow users to more easily understand and rapidly become aware of trends in the data. Playfair (1786), who developed the classic plotting methods, seems to be the first person to use abstract properties such as lines and drawings to represent data [PWS05]. In 1977, Tukey's *Exploratory Data Analysis* emphasized how to use graphing techniques to produce pictures that provide a rapid insight into data. The techniques included spotting outliers, discriminating clusters, and examining relationships [Tuk77]. Cleveland and McGill's *Dynamic Graphics for Statistics* in-

troduced new applications of visualization in statistics. They emphasized the use of dynamic methods which provide interactions between pictures and users [CM88]. Visualization was recognized as an important field of science in a 1987 NSF report, *Visualization in Scientific Computing* [MDB87]. Later, in 1990, the first IEEE Visualization Conference was held in San Francisco, California. At that time, visualization was a useful tool for managing the huge amounts of data sent back from satellites. It was also used with scientific computer simulations to produce visual results for researchers. Although visualization often focuses on physically based data, collected from experiments or simulations, researchers have shown that nonphysical information — such as business information, statistical data, and abstract concepts — can also benefit from a visual representation. The demand for visualizing abstract information is becoming stronger because today's world contains a great deal of complex abstract data. Unfortunately, existing visualization techniques for physically based data do not always translate easily to abstract data. Unlike data from physical objects, there is no explicit spatial mappings between visual structures and the data. This problem motivates extensions to visualization techniques for handling different types of abstract data [MRC91].

2.2 How Visualization Helps

As defined by Card, Mackinlay, and Shneiderman, visualization is the use of computer supported, interactive, visual representations of data to amplify cognition [CMS99]. Cognition can be defined as the acquisition or use of knowledge. They proposed six major ways in which visualization may possibly amplify cognition.

1. *Increased Resources:*

   Some visual attributes can be processed in parallel as opposed to text, which must be scanned sequentially. Offloading tasks from the cognition system to the perception system by using visualization reduces the load for humans, similar to expanding the working memory available for solving a problem [Nor93]. Visualization is a compact representation. It can store massive quantities of information in a compressed and quickly accessible form. A concise representation makes it easier for people to perceive multiple visual features at once.

2. *Reduced Search:*

   Visualization reduces the search time for information by grouping related information together. This allows users to more rapidly locate interesting regions. It can then provide more information about a particular region with additional zooms or pop up views.

3. *Enhanced Recognition of Patterns:*

   Patterns can be revealed by clustering data elements in a visualization. Data with a similar organization can be viewed as a pattern. Through clustering, patterns are observed more easily because of the higher frequency of similar characteristics.

4. *Perceptual Inference:*

   The inference process creates new knowledge by linking from the current representation to the experiences a person is familiar with. Visualization uses a number of perceptual inferences which are intuitive for humans [LS87]. The cold-to-hot scale is one example.

People recognize blue as a lower value because blue is often associated with the low end of a temperature scale. The same is true with red, which is often associated with the high end. The experience for people from different cultures or living environments must be considered when defining inferences, however.

5. *Perceptual Monitoring:*

   There are many ways to direct users' attention towards specific data, e.g. using motions or textures. There are also ways to obscure or hide unrelated data from the user, e.g. making certain areas blurred or transparent. By using these methods, visualization can allow for the monitoring of a large number of potential events. As the events change their visual states, they can be observed quickly.

6. *Manipulable Medium:*

   Unlike static forms or tables, users can explore a parameter-value space by using different actions, including selection, filter and navigation. These interactions can assist users build helpful mental models of the data.

2.3 Categories of Visualization

Visualization is often classified into two categories: *scientific visualization*, which focuses on physically based data and *information visualization*, which focuses on abstract data. As mentioned in the previous section, physically based data is gathered from physical objects or scientific experiments. Visualizations often use the inherent physical properties of these results

to build meaningful layouts. On the other hand, there is often no associated physical structure for abstract data, thus failing to provide a visualization with an intuitive starting point.

This classification does not imply that scientific and information visualization are two disjoint areas. Researchers in scientific visualization sometimes use techniques from information visualization [RTM+03]. Similarly, some scientific visualization techniques work in an information visualization environment. By imposing geometries onto abstract datasets one can build a visualization using techniques from scientific visualization. For example, Treinish used a transparent curved surface to represent ozone concentration in the atmosphere based on a 3D earth model [Tre94].

The use of an appropriate visualization technique is mainly determined by the task and the features of data. For example, visualizations of physically based data often add abstract structures to provide a better understanding of things that can not be seen, like air flows. Information visualizations can embed abstract datasets onto familiar physical structures to improve presentation and comprehension. Both categories of visualization have a common goal: to provide better communication between humans and information.

Although scientific and information visualization share many techniques, the research in this thesis is mostly related to information visualization.

2.4 Information Visualization

Techniques in information visualization usually deal with large datasets. One major challenge is to fit as much information into a single view as possible without overwhelming users. Two

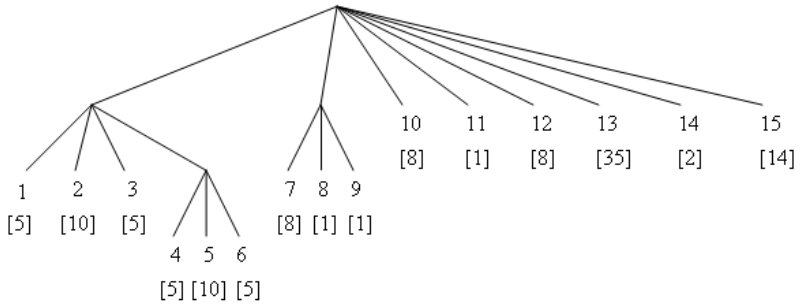
major groups of techniques designed around this challenge are *overview + detail* and *focus + context*.

2.4.1 Overview + Detail

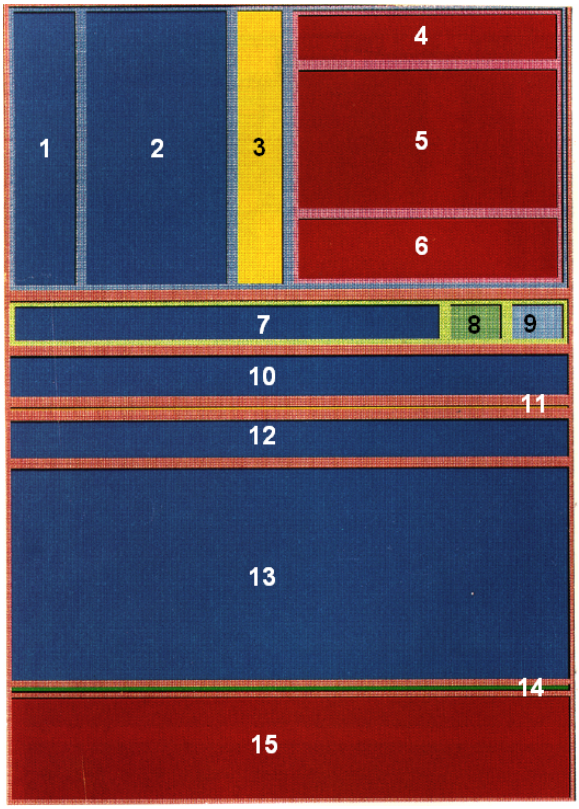
Overview + detail techniques provide multiple views to users, often including one overview and one or more detail views. Overviews are important because users can easily see the overall structure of the data and quickly become aware of significant variances between different regions. If users want to study details about a region of interest, detail views are generated to provide this information. These two types of views can be shown either simultaneously in different parts of the screen (space multiplexing), or individually at different time (time multiplexing) [CMS99]. In space multiplexing, multiple views share the same screen space. This method usually involves a trade-off between available space and the number of views. In time multiplexing, the system must provide mechanisms for users to easily switch between different views while minimizing loss of context. One of the biggest issues for overview + detail is whether searching different parts of the screen and switching between views reduces users' performance.

Tree-maps

A *Tree-map* is a well-known overview + detail space multiplexing system [Shn92]. It was developed to represent hierarchical data from a tree diagram with a more compact visual form of multiple embedded rectangles. The tree-map recursively separates a rectangular screen space



(a)



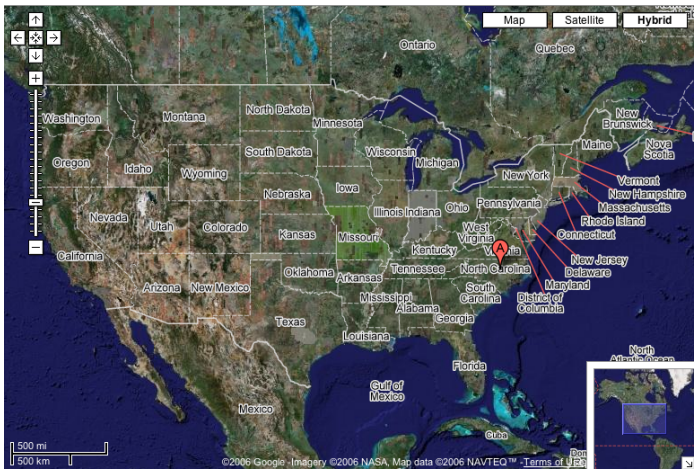
(b)

Figure 2.1: (a) A tree diagram. Numbers in brackets under nodes represent the nodes' weights. (b) A tree-maps for the tree in (a). The number in each area is the node number in tree (a).

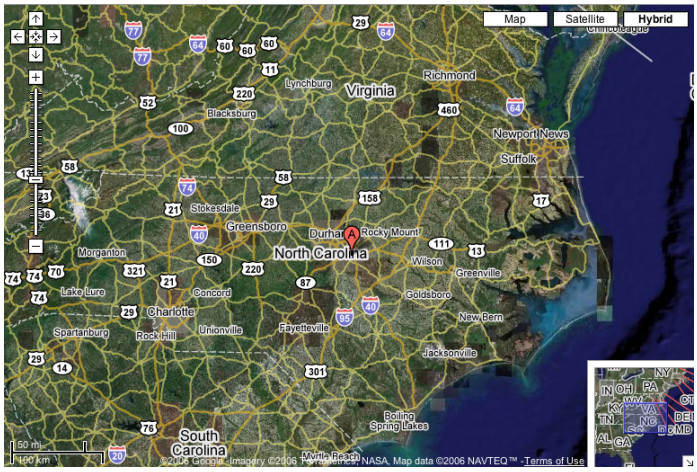
into smaller rectangles in both the horizontal and vertical directions. Each separation creates rectangles which represent nodes located on the same level of the tree digram. If a node has child nodes, its rectangle will be divided recursively until no children remain. Figure 2.1a is a tree diagram representing a file system with 15 files. Figure 2.1b shows its corresponding tree-maps. Each rectangular region is mapped to a leaf node. The area of this region is proportional to its node's weight. In this example, the tree-map is initially separated into eight blocks in the vertical direction according to level two in the tree diagram. If a leaf node is reached, the created block is assigned to that node, otherwise, it is continuously separated into sub-blocks in the perpendicular direction. For example, nodes 10 to 15 are assigned to blocks after the first separation, and nodes 1, 2, and 3 are assigned to blocks established after the second separation. Later revisions of tree-maps added a zoom-in functionality. These tree-maps allowed users to select an individual block to expand and fill the whole screen with more detailed information [Shn06].

Electronic maps

A time multiplexing example of overview + detail is the application of an electronic map, such as Google Maps [GOO06]. Users can choose how close they want to be to a target and indirectly determine how much information will be shown. Figure 2.2 shows the geographic location of Centenial Campus in NCSU with multiple views in Google Maps. In Figure 2.2a, the viewer is farthest from the target. At this distance, the entire continental United States can be seen. In Figure 2.2b, the viewer is closer to the target. Detail information about cities and highways in North Carolina is shown in this view. Figure 2.2c is the most detailed view



(a)



(b)

(c)

Figure 2.2: Different views for Centenial Campus in NCSU. The red balloon with a letter 'A' represents its location. (a) A view covering entire USA. (b) A view covering North Carolina State. (c) A local view containing detailed information.

providing local streets and building information. Although shorter distance from the target produce a more detailed view, some global information is lost. Switching focus between views with different degrees of detail may require additional time for users to reorient themselves.

2.4.2 Focus + Context

Another technique of information visualization is focus + context. Focus is the view which has detailed information about some part of the region being visualized. It is analogous to the detail view in overview + detail. Likewise, context is analogous to the overview.

Focus + context differs from overview + detail in that it often combines a detail view with the overview. The advantage is that changing the contents of the detail view does not affect the perception of relevant context. This reduces the time for reorienting during a switch between different views. To emphasize the focus information and reduce context regions' demand on screen space, multiple methods selectively reduce information in the contextual area [CMS99]. The issue for focus + context is whether the visual representation is still effective for the task at hand after the application of these reduction techniques. The following algorithms are concerned with one possible method for generating focus + context: distortion. Distortion selectively changes the number of pixels assigned to objects on the screen, usually through the application of a mathematical filter or change in the underlying geometric space.

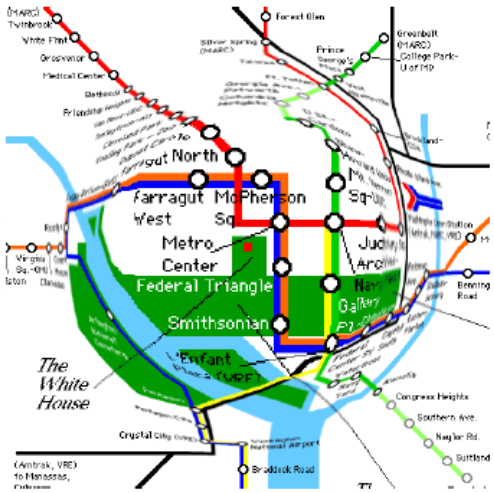
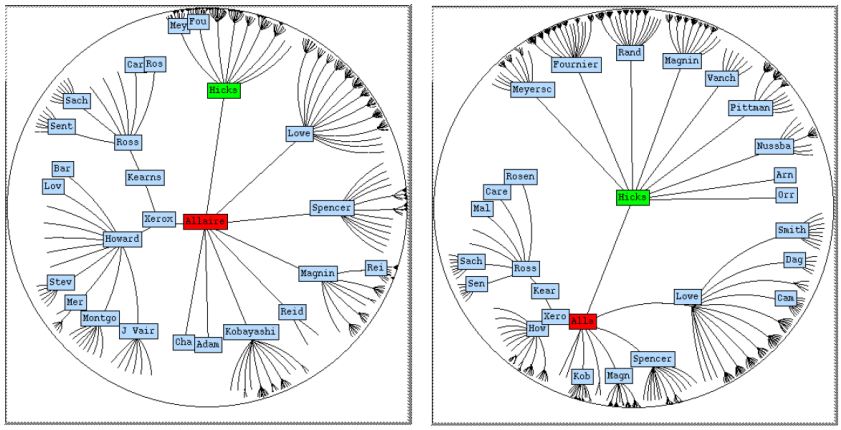


Figure 2.3: A Fisheye len used on a subway system in Washington. D.C.

Fisheye Views

Fisheye views are a technique used to generate a small display of a large structure [Fur86]. There are two levels of detail displayed simultaneously. The overview (context) is a *low level of detail* view. The detail view (focus) modifies the covered area like a magnifying lens into a *high level of detail* (Figure 2.3). This technique provides smooth transitions between overview and detail by maintaining the spatial orientation of the overview. Multiple focuses are allowed, and all of them can be moved by user interaction.

A Degree Of Interest (DOI) function assigns each point in a structure to be visualized a number indicating how interesting the point is to the viewers. In fisheye views, this function is decomposed into two parts: $DOI(x|y) = i_a(x) - D(x, y)$, where $i_a(x)$ is an *a priori* number assigned to x, $D(x, y)$ is the distance between x and y, and $DOI(x|y)$ is the degree of interest for x given the current focus point y. Interest values decrease with increased distance, i.e. information farther from the focus must be more interesting for it to be shown. In the final



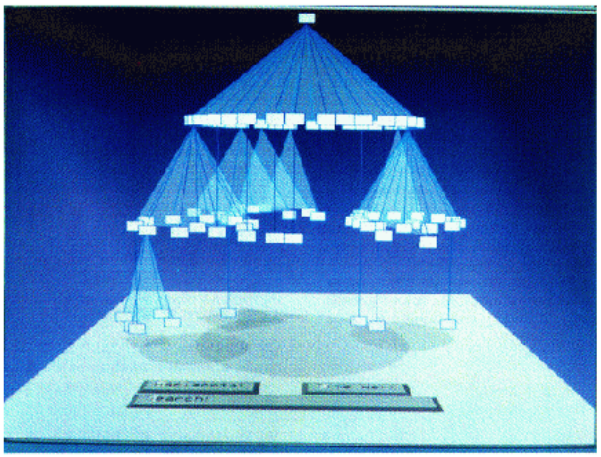
(a) (b)

Figure 2.4: (a) A hyperbolic browser showing the relations in a family with the focus on the red node "Allaire." (b) The focus node changes from "Allaire" to the green node "Hicks."

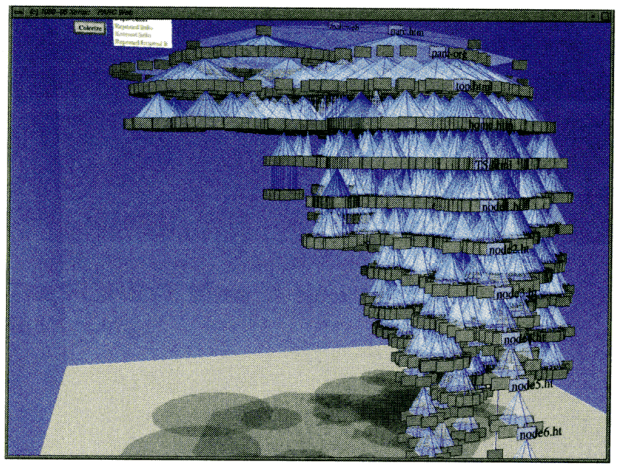
view, x is displayed if and only if its degree of interest is above some threshold k.

Hyperbolic Browser

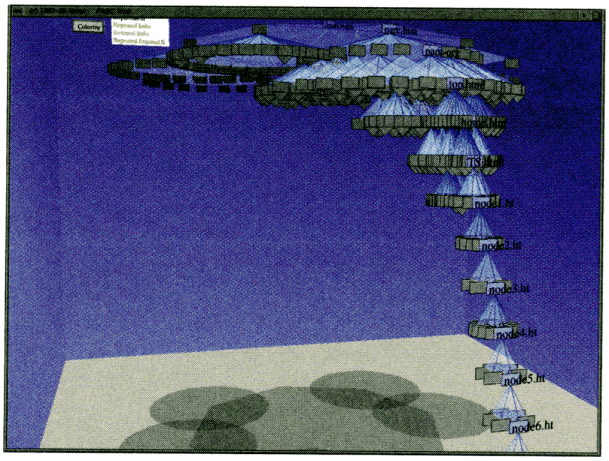
A *hyperbolic browser* is another focus + context technique used to visualize and manipulate large tree hierarchies. In a hyperbolic browser, a tree structure is embedded on a hyperbolic plane, and the plane is mapped into a display region. The chosen mapping provides a distorted fisheye view which supports a smooth display of focus and context (Figure 2.4). Users can click or drag on the display region to bring other nodes into focus with a smoothly animated transition. The amount of space available to a node is determined by its distance in the tree from the node that currently holds focus. The longer the distance, the less space it is assigned. Figure 2.4a shows the focus on a red node with its neighbors visible. In Figure 2.4b a new green node is moving into focus with a new set of neighbors. Note how some of the old information has been allocated less screen space as it moves to the periphery.



(a)



(b)



(c)

Figure 2.5: (a) A cone tree with shadows added to help users understand the 3d shape of the structure. (b) A cone tree of a 10,000-node website. (c) A filtered cone tree from (b) using a degree of interest algorithm.

Cone Trees

A cone tree is another technique from focus + context [RMC91]. Figure 2.5a shows a snapshot of a cone tree. Cone trees are designed for manipulating hierarchical information in a 3d space using both depth and perspective projection. The hierarchical relationships are represented with semi-transparent cones. The heights for cones are fixed while the radiuses of the cones' bases decrease as one moves down the hierarchy. The parent nodes of categories are placed at the top of the cones, and sub nodes are placed along the peripheral of the cone's base. There are also shadows projected onto the bottom plane to help users understand relational structures between cones. After the selections of nodes of interest, the cones rotate about their center y-axis separately to bring the interesting nodes to the front. The interesting nodes are bigger relative to the remaining nodes and may be partially obstructed because of the perspective view. This is how cone trees allocate more pixels for nodes with a higher degree of interest.

Filtering is another method used to reduce contextual information. The technique can be combined with distortion. Figure 2.5b shows a cone tree of a 10,000-node website. Figure 2.5c shows the same tree but with some branches filtered by a degree of interest algorithm.

Chapter 3

Combinatorial Auction

The purpose of this chapter is to provide a brief overview of our practical application testbed. We begin by describing general ideas about combinatorial auctions, including what constitutes a combinatorial auction, its advantages, and where it is used. Finally, we discuss issues specific to our test datasets containing results from an iterative proxy combinatorial auction.

3.1 Combinatorial Auction

Combinatorial auctions are auctions where one or more items are sold simultaneously. Bidders can place bids on bundles of items rather than bidding separately on individual items. This ability to bid on bundles makes combinatorial optimization and mathematical programming relevant. Thus, combinatorial auctions are studied in economics, operations research, and computer science [SR05, ZN01, ONH03].

Combinatorial auctions allow bidders to fully express their preferences, especially when

items are complementary. A complement is a set of items that has a greater utility than the sum of the utility for each individual item. For example, a pair of shoes is more valuable than a left shoe or a right shoe alone. The value of an individual item may differ between bidders if it belongs to a complement and a bidder already possesses other items from that complement.

Recently, combinatorial auctions have been employed in a variety of industries: truckload transportation and bus routing, airport arrival and departure scheduling, and allocating radio spectrums for communications services [RSB82, Cra02]. In each case, the motivation for using a combinatorial auction is the complementary nature of the desired resources. For example, a trucker's cost for handling shipping to a new destination depends on what his original routes are. If the new destination is on his existing path, the cost for shipping goods there will be relatively lower than shipping to a location not on the path.

3.2 Iterative Combinatorial Auctions

An iterative combinatorial auction is one type of combinatorial auction mechanism. The auction proceeds in rounds. At the beginning of each round, the bidders submit their offers for preferred bundles. The auctioneer receives all the bids and solves a *Winning Determination Problem* (WDP). The auctioneer announces the winners and winning prices to end the round. The bidders then submit their offers for the next round. An iterative process allows bidders to provide provisional prices and to then adjust their allocations according to other bidders' offers.

	A	B	AB	C	AC	BC	ABC
Buyer 1	10	3	18	2	18	10	20
Buyer 2	4	9	15	3	12	18	20
Buyer 3	1	3	11	9	16	17	25
Buyer 4	7	7	16	7	16	16	20

Figure 3.1: An example of a SPCA with four buyers bidding on combinations of three objects. The integers represent the highest prices that buyers are willing to pay for each bundle.

The testbed combinatorial auction application we are visualizing accepts bids on bundles and generates non-linear bundle prices, i.e., the bundle prices may not be the sum of the individual items' prices. It runs as an iterative proxy combinatorial auction where proxy agents act in place of the actual bidders. The buyer submits to his proxy agent the values that he is willing to pay for each bundle before the auction begins. The agent bids on behalf of the buyer by following their values as in Figure 3.1. The agent's profit for each bundle is the difference between the price at which he wins a bundle and the value provided by the buyer. The profit is greater than or equal to zero if the agent wins a bundle. If an agent does not win a bundle, the profit for that bundle will be zero. All agents try to maximize profit.

The time for each round is very short (e.g. less than one second). During a round, an agent may either increase his bid on his preferred bundle or pass. Each agent has a demand set containing the agent's most desired bundles. The demand set contains the bundles which maximize the agent's profit based on the bundles' current prices in the auction. As the auction progresses and bundle prices change, an agent's demand set will also change. his profit and it

may change over rounds. Although an agent may have multiple bundles in the demand set, he can only improve his bid on one bundle in each round. An agent improves his bid by adding a fixed amount to the bundle's current price. Therefore, in a new round, an agent can continue bidding on the last bid-on bundle, start bidding on different bundle in the demand set, add a new bundle to his demand set and bid on it, or pass for the round.

After receiving all bids, the auctioneer may have multiple competitive allocations. A competitive allocation is a mapping between winning agents and bundles that maximizes the sum of the bid prices (i.e. the auctioneer's surplus). If multiple competitive allocations (i.e. allocations with the same surplus) exist, the auctioneer chooses one at random as the winning allocation. Set notation is used to define competitive allocations. Positions in the set correspond to an ordered list of agents. Bundles appearing at a position i implies agent i has currently won that bundle. For example, assume four agents bidding on three items A, B, and C. In the first few rounds, each agent bids on the bundle $\{A, B, C\}$. Therefore, the possible competitive allocations in these rounds are $\{ABC, -, -, -\}$, $\{-, ABC, -, -\}$, $\{-, -, ABC, -\}$, and $\{-, -, -, ABC\}$, where agent one, two, three, or four wins $\{A, B, C\}$, respectively. Since the bid prices from all four agents are the same, the auctioneer would randomly choose one of the four allocations as the winning one. Thus, all the agents are equally likely to be chosen as a winner. After the bundle's price increases, agents with relatively lower maximum values would stop bidding on the entire bundle and turn to bid on smaller bundles in order to maximize their profit. As the smaller bundles increase their prices, they start entering competitive allocations. Figure 3.3 shows the change of competitive allocations and their prices over time.

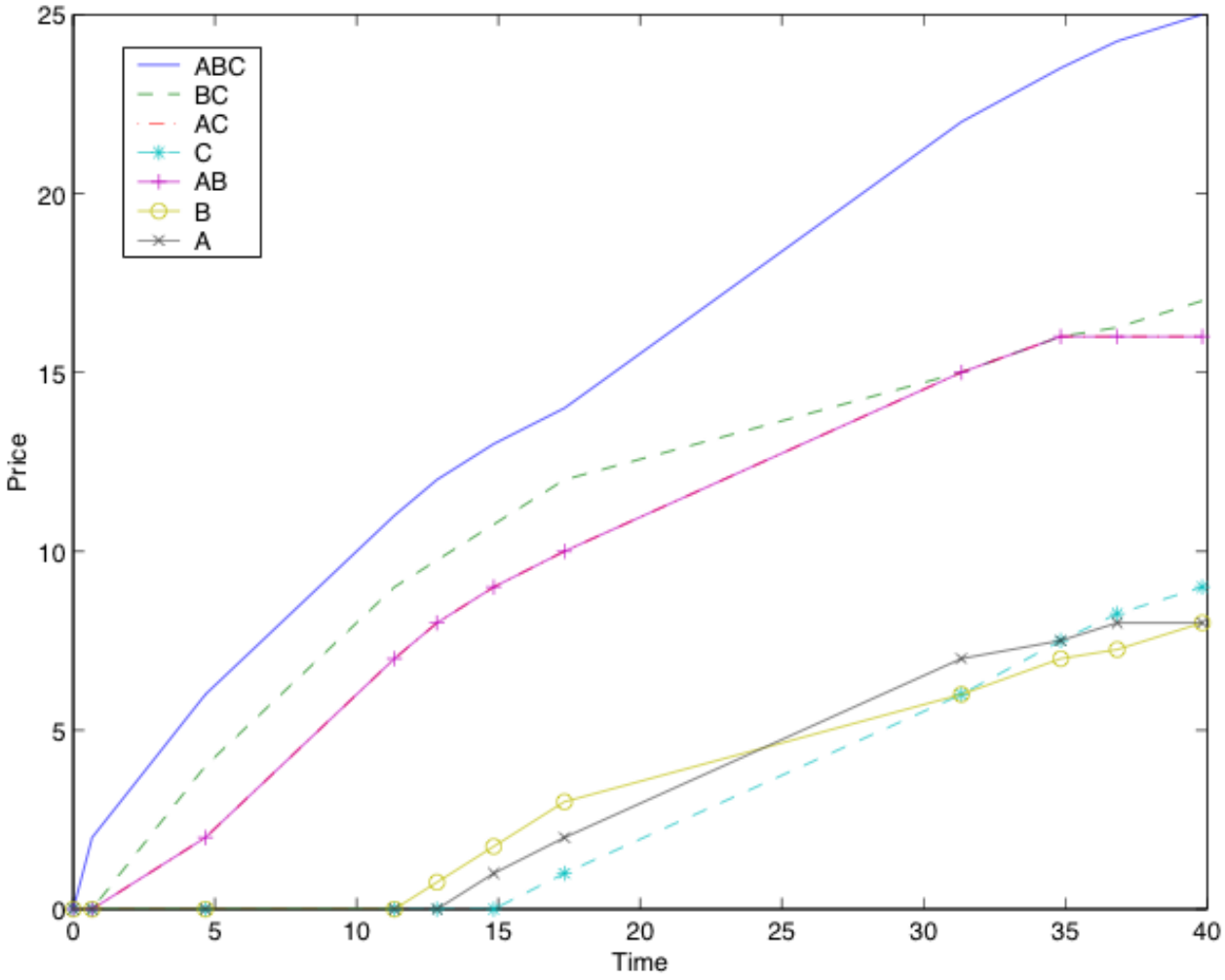


Figure 3.2: The bundle prices over time when proxy agents bid based on Figure 3.1.

To analyze the bidding behaviors in an auction, auction information is recorded in terms of significant events, called inflection points. Inflection points are rounds where one of three events happens: a new bundle enters into an agent's demand set, the competitive allocations change, or an agent withdraws as an active bidder. Figure 3.4 shows a table with detailed information about six inflection points. The symbols on the lines in Figure 3.3 indicate diagrammatically the different inflection points.

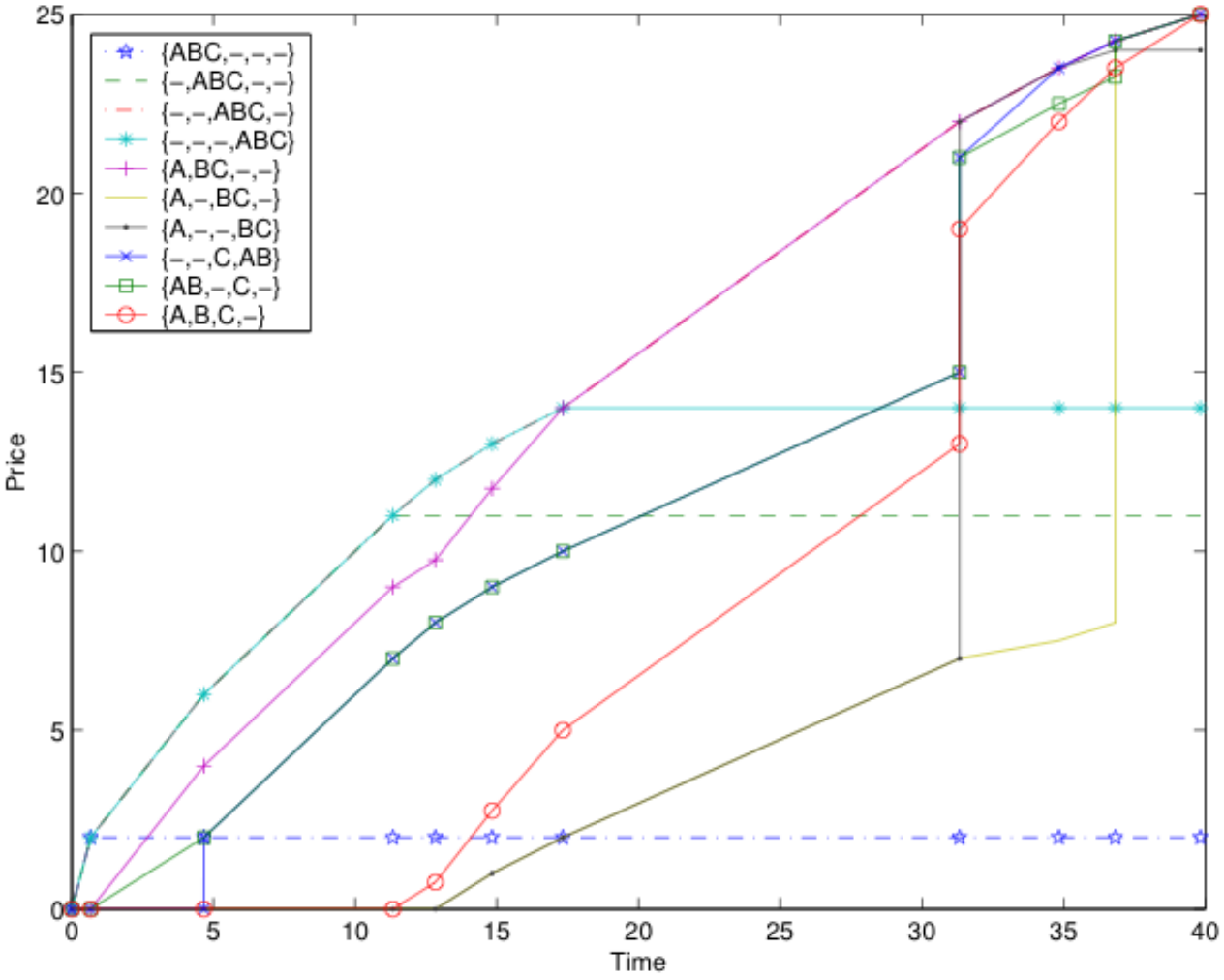


Figure 3.3: The changes of the competitive allocations for the example in Figure 3.1

Step 1, $t = 0$ — Prices: 0 0 0 0 0 0 0

$\boldsymbol{f}\|V(\boldsymbol{f}) = 0$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
*{ABC, –, –, –}	1	ABC							0.75	0.25
*{–, ABC, –, –}	2	ABC							0.75	0.25
*{–, –, ABC, –}	3	ABC							0.75	0.25
*{–, –, –, ABC}	4	ABC							0.75	0.25
		θ_b	0	0	0	0	0	0	3	

Step 2, $t = 2/3$ — Prices: 0 0 0 0 0 0 2

$\boldsymbol{f}\|V(\boldsymbol{f}) = 2$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
{ABC, –, –, –}	1	AB, AC, ABC			0.5		0.5			
*{–, ABC, –, –}	2	BC, ABC						1	0	
*{–, –, ABC, –}	3	ABC							0.5	0.5
*{–, –, –, ABC}	4	ABC							0.5	0.5
		θ_b	0	0	0.5	0	0.5	1	1	

⋮ ⋮ ⋮

Step 7, $t = 17\frac{1}{3}$ — Prices: 2 3 10 1 10 12 14

$\boldsymbol{f}\|V(\boldsymbol{f}) = 14$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
*{A, BC, –, –}	1	A, AB, AC	5/14		0		1/14			4/7
{–, ABC, –, –}	2	B, BC, ABC		3/14				3/14		4/7
*{–, –, ABC, –}	3	ABC							4/7	3/7
{–, –, –, ABC}	4	AB, C, AC, ABC			5/14	5/14	4/14			
		θ_b	5/14	3/14	5/14	5/14	5/14	3/14	4/7	

Step 8, $t = 31\frac{1}{3}$ — Prices: 7 6 15 6 15 15 22

$\boldsymbol{f}\|V(\boldsymbol{f}) = 22$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
*{A, BC, –, –}	1	A, AB, AC	1/7							6/7
*{A, –, –, BC}	2	B, BC		1/7				1/7		5/7
*{–, –, ABC, –}	3	C, ABC				3/7			3/7	1/7
	4	B, AB, C, AC, BC		1/7	2/7		2/7	1/7		1/7
		θ_b	1/7	2/7	2/7	3/7	2/7	2/7	3/7	

Step 9, $t = 34\frac{5}{6}$ — Prices: 7.5 7 16 7.5 16 16 23.5

$\boldsymbol{f}\|V(\boldsymbol{f}) = 23.5$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
*{A, BC, –, –}	1	A	1/4							3/4
{A, –, –, BC}	2	B, BC		1/8				1/8		3/4
*{–, –, ABC, –}	3	C, ABC				3/8			3/8	1/4
*{–, –, C, AB}	4	∅								
		θ_b	1/4	1/8	0	3/8	0	1/8	3/8	

Step 10, $t = 36\frac{5}{6}$ — Prices: 8 7.25 16 8.25 16 16.25 24.25

$\boldsymbol{f}\|V(\boldsymbol{f}) = 24.25$	Agent	$\hat{D}_i$	A	B	AB	C	AC	BC	ABC	pass
*{A, BC, –, –}	1	A, AB, AC	0		0		0			1
*{A, –, BC, –}	2	B, BC		1/4				1/4		1/2
*{–, –, ABC, –}	3	C, BC, ABC				1/4		0	1/4	1/2
*{–, –, C, AB}	4	∅								
*{AB, –, C, –}		θ_b	0	1/4	0	1/4	0	1/4	1/4	

Step 11, $t = 39\frac{5}{6}$ — Prices: 8 8 16 9 16 17 25

Agent 3 stops bidding and either {A, BC, –, –} or {A, B, C, –} wins.

Figure 3.4: Information of six of ten round-result for the example in Figure 3.1.

Chapter 4

Method

This chapter describes the implementation details of our algorithms to generate visualizations of a combinatorial dataset. Considering the special characteristics of combinatorial auctions mentioned in Chapters 1 and 3, we try to organize the data elements in an effective way. For proxy bidding combinatorial auctions, the system uses multiple visual features to handle the multiple attributes in the auction data. These visual features include *color*, *motion*, *transparency*, *blur*, and *texture*.

4.1 Layout

The foundation of this visualization technique is a collection of multiple equal width concentric rings separated by a fixed distance. Beginning from a given degree, each ring is divided into multiple arcs. Figure 4.1 shows an example layout for a combinatorial dataset D with elements $\{A, B, C, D, E\}$. Subsets from D are distributed onto individual arcs. The blue tag on each arc

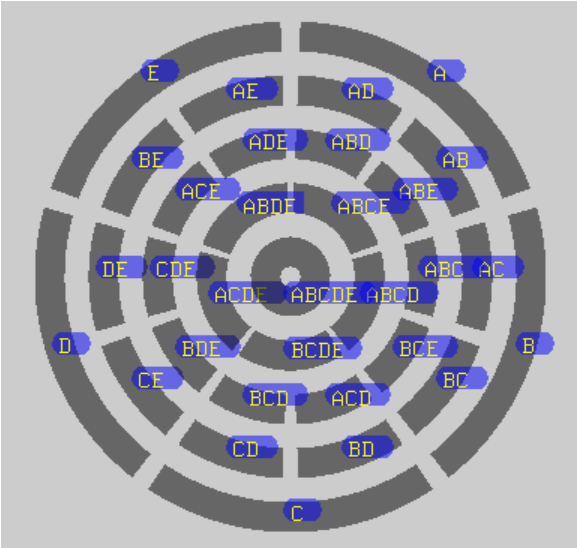
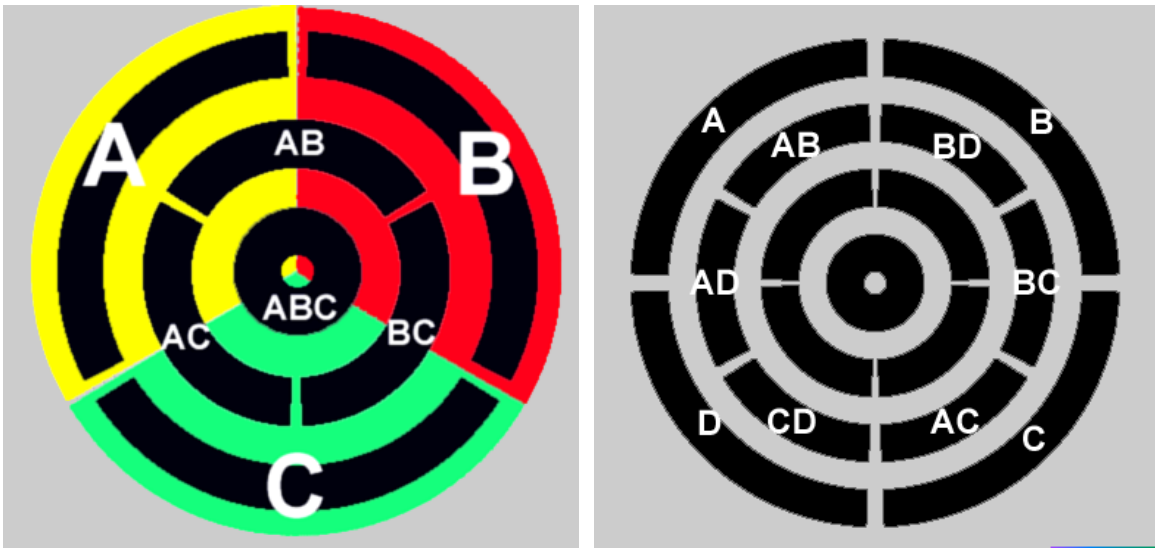


Figure 4.1: A concentric ring layout for the family of subsets of {A,B,C,D,E}.

in the example figure names its corresponding subset. The core center ring is always a whole circle that maps to the subset containing all of D's elements.

4.1.1 Rings and Arcs

The visualization's rings and arcs within each ring serve to group the dataset's subsets in a logical manner. All subsets of equal size are distributed within a common ring. The mapping relationship of subsets to arcs is *one-to-one* and *onto*, meaning distinct subsets map to distinct arcs, and for each arc, there is at least one subset mapped to it. The outermost ring contains subsets of size one, the first interior ring contains subsets of size two, and so on until the largest subset is reached. Therefore, the number of rings equals the number of elements in D.



(a) (b)

Figure 4.2: (a) A completely concise layout for the family of subsets of $\{A, B, C\}$; (b) An incomplete, concise form for the subsets of $\{A, B, C, D\}$.

4.1.2 Layout Problem

Each arc in a ring defines a pie-shaped region, or sector, by connecting both of the arcs' end points to the center of the circle. For example, the outermost ring subdivides space into n sections, where $n = |D|$. To organize the layout, we will try to arrange the arcs so that for each of the outermost arcs, a sector spanning the arc intersects with the arc of every superset containing that arc's element. We call a layout that achieves this a completely concise arrangement. Figure 4.2a shows a completely concise layout for the combinatorial dataset $\{A, B, C\}$. The yellow, red, and green areas indicate the sectors for $\{A\}$, $\{B\}$, and $\{C\}$, respectively. $\{A, B\}$ maps to an arc that overlaps with the sectors for $\{A\}$ and $\{B\}$. Similarly, $\{B, C\}$ maps to an arc that overlaps the sectors for $\{B\}$ and $\{C\}$, and $\{A, C\}$ maps to an arc that overlaps with the sectors for $\{A\}$ and $\{C\}$. $\{A, B, C\}$ maps to the core circle which overlaps all three sectors.

Unfortunately, a completely concise form does not exist when the dataset's size is greater than three. To see this, consider the outer ring's layout for a dataset, $\{x_i, ..., x_n\}$, $n \geq 4$. Any arc has exactly two neighbors. x_1, for example, has neighbors x_n and x_2. By definition, the first interior ring has more arcs than the outer ring, so its arcs are shorter than arcs on the outer ring. This implies that any arc in an interior ring can overlap at most two neighboring sectors on the outer ring. Any subset involving x_1 and a non-neighboring element (e.g. $\{x_1, x_3\}$) therefore cannot be mapped to an arc on the interior ring that overlaps both $\{x_1\}$'s and $\{x_3\}$'s sectors. Thus no completely concise mapping exists. Figure 4.2b shows the layout for the combinatorial dataset $\{A, B, C, D\}$. In the first inner ring, subsets $\{A, D\}$ and $\{B, C\}$ are mapped to arcs in completely concise locations, but $\{A, B\}$, $\{A, C\}$ and $\{B, D\}$, and $\{C, D\}$ are not. As we have shown, there is no configuration of this ring that creates a completely concise layout. Thus, our techniques must approximate a concise layout as closely as possible. Our method for measuring conciseness in described in the next section.

4.1.3 Layout Algorithm

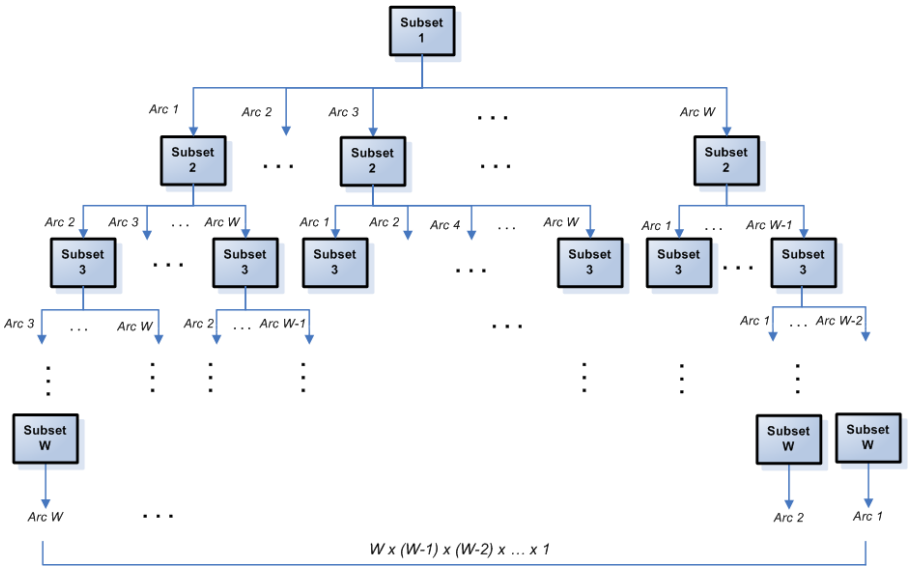
We generate the required arcs in each ring using a scoring system to evaluate how close a candidate mapping is to a completely concise form. A subset is assigned a score when mapped to an arc. To calculate a subset's score, we first extract its elements. For each element, we count the number of arc offsets within is ring it must move to reach an arc covered by the element's sector. The direction of the movement can be either clockwise or counterclockwise. The shortest distance is the score assigned to this element. A subset's score is simply the sum

of all its elements' scores. The ring's overall score is computed by summing all of its subsets' scores. A lower score represents a more concise form.

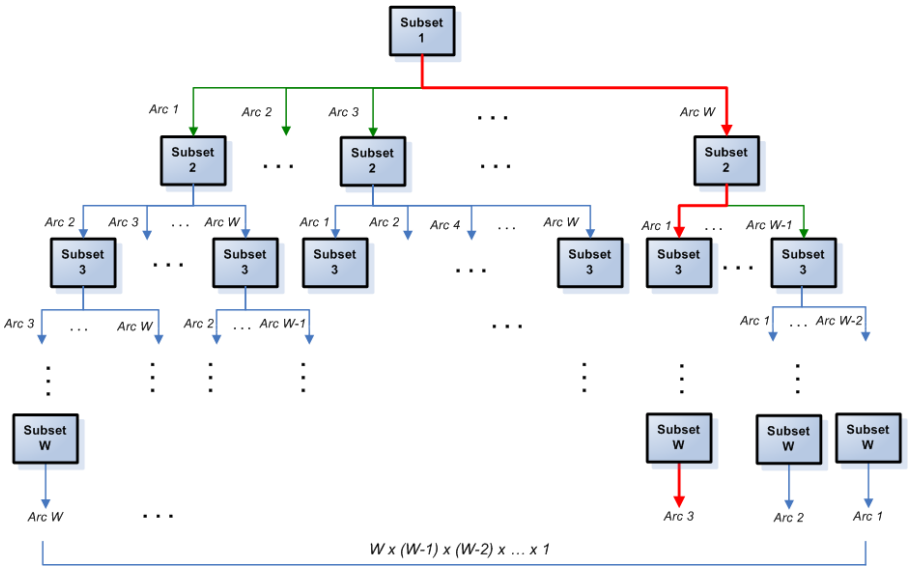
For example, consider subset $\{A, C\}$ in Figure 4.2b. Element C from the subset falls within $\{C\}$'s sector, so it receives a score of zero. Element A does not fall within $\{A\}$'s sector, however. $\{A, C\}$ would need to walk two clockwise arcs or three counter clockwise arcs to enter $\{A\}$'s sector. The score for element A is therefore two, and the overall score for $\{A, C\}$ is $2 + 0 = 2$. This same procedure is applied to all the subsets and the resulting scores are summed to generate an overall score for the entire layout.

Given w subsets to be mapped to w arcs in a ring, there are $w!$ possible layouts. Figure 4.3a shows a tree diagram enumerating each possible layout. One way to find the lowest score mapping is to traverse the tree completely. Each subset maintains the scores for all possible arcs, which requires total w^2 subset scores. To find out the lowest scored path, $w! - 1$ comparisons are needed (comparing the $w!$ distinct paths). This brute-force method guarantees the best mapping, but it is usually not feasible due to long computation times and large memory requirements.

To improve efficiency, we make a trade-off between time and accuracy. First, we pick a subset, and map it to the arc where it has the minimum score. Next, we pick the second subset and perform the same search on the remaining arcs. We continue iteratively until all subsets are mapped to an arc. This method explores only a small part of the tree. Figure 4.3b shows an example of paths being explored. In the same case of w arcs and w subsets, only $w + (w-1) + (w-2) + ... + 1 = \frac{w^2+w}{2}$ subset scores are needed. The number of comparisons



(a)



(b)

Figure 4.3: (a) A tree of all possible mapping relations between w subsets and w arcs; (b) rather than traversing the complete tree, we iteratively assign a subset to an arc, then search among the remaining arcs for a position for the next subset, continuing until each subset is assigned to an arc; The green lines indicate the arcs examined in each round, and the red lines represent the chosen arcs.

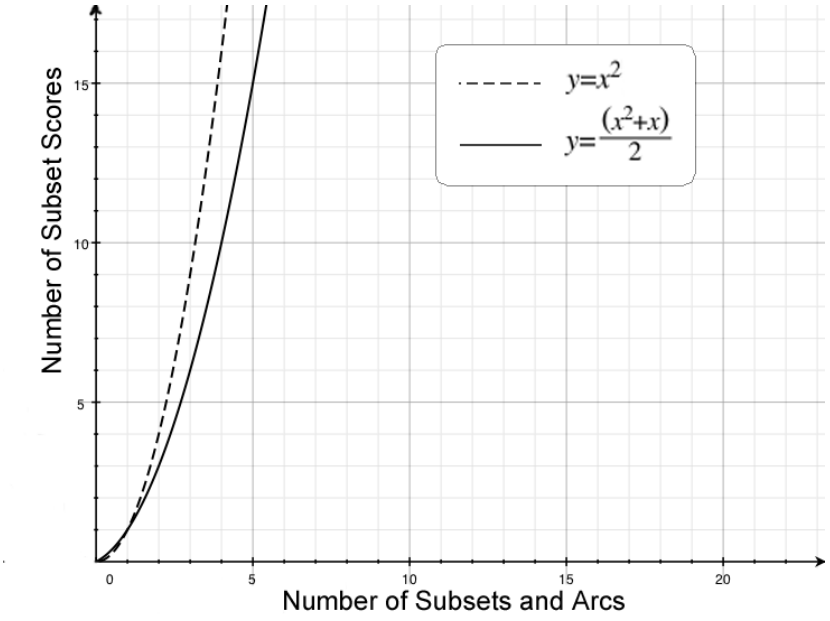
Table 4.1: Comparisons of required computations for combinatorial datasets from $\{A, B, C\}$, $\{A, B, C, D\}$, and $\{A, B, C, D, E\}$.

	Our Algorithm			Brute-force		
	Score Allocation	Comparions made	Best Score	Score Allocation	Comparions made	Best Score
$\{A, B, C\}$	6	3	**3**	9	6	**3**
$\{A, B, C, D\}$	31	29	**14**	52	742	**14**
$\{A, B, C, D, E\}$	125	100	**81**	225	7,257,717	**80**

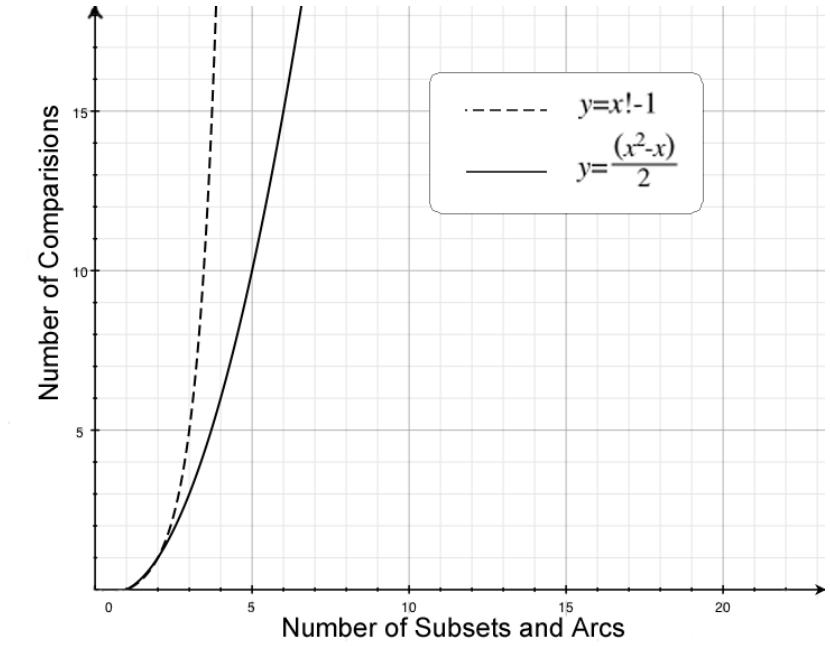
is $(w-1)+(w-2)+...+1 = \frac{w^2-w}{2}$ which happen in each level represented by green lines in Figure 4.3b.

Figure 4.4a and Figure 4.4b compare the equations for the required subset scores and score comparisons between our algorithm and the brute-force method. From these two pictures, it is easy to see that our algorithm reduces computation significantly when the number of subsets grows. Table 4.1 shows the required subset scores, required number of score comparisons, and the resulting best scores in our algorithm and the brute-force method. The first column contains the sets where the test combinatorial datasets are generated. The scores in the table are the sum of scores from different rings in each case. Take $\{A, B, C, D, E\}$ for example, its layout has five rings. It's scores are the sum of the scores from the first, second, and third interior rings. Those rings have $\binom{5}{2}$ = 10, $\binom{5}{3}$ = 10, and $\binom{5}{4}$ = 5 arcs respectively. The required subset scores and comparisons in our algorithm are $\sum_{w=10,10,5} \frac{w^2+w}{2}$ and $\sum_{w=10,10,5} \frac{w^2-w}{2}$, respectively. In the brute-force method, they are $\sum_{w=10,10,5} w^2$ and $\sum_{w=10,10,5}(w! - 1)$.

Our algorithm reduces computation in both stages, especially the number of comparisons as the size of the dataset increases. For $\{A, B, C, D, E\}$, our algorithm reduces 7,257,717 comparisons to only 100. Note also that our algorithm's best scores are very close to the optimal



(a)



(b)

Figure 4.4: (a) The equations for required subsets scores in our algorithm and in the brute-force method; (b) The equations for required score comparisons in our algorithm and in the brute-force method

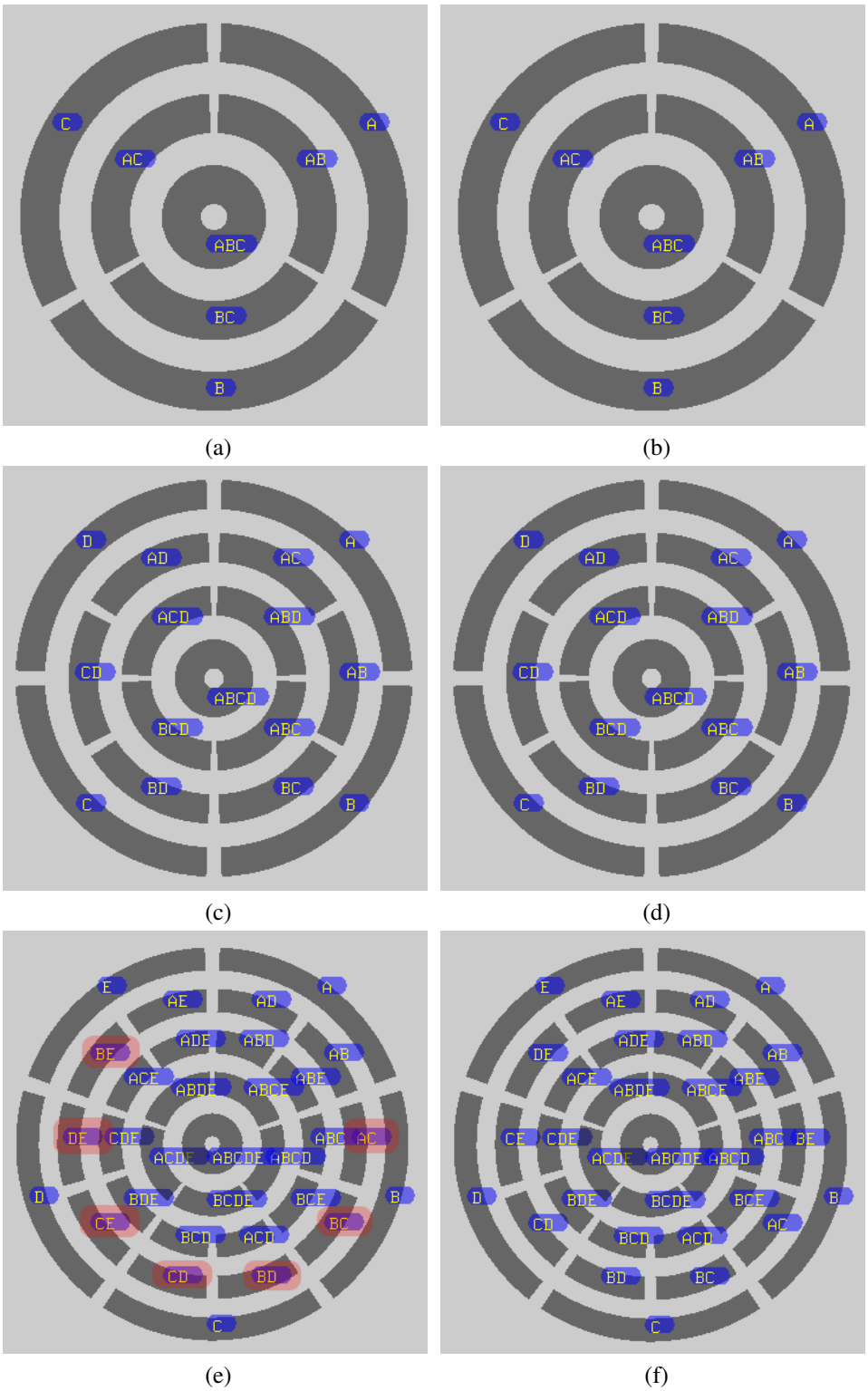


Figure 4.5: The visualizations generated by our algorithm and the brute-force method. Our visualizations are in the left side. The three pairs results are for combinatorial datasets from $\{A, B, C\}$, $\{A, B, C, D\}$, and $\{A, B, C, D, E\}$, respectivly. Red shields indicate different placements from the right visualization.

score from the brute-force method. Figure 4.5 shows both our technique's and the brute-force method's vsualizations for combinatorial datasets generated from $\{A, B, C\}$, $\{A, B, C, D\}$, and $\{A, B, C, D, E\}$. Our visualizations are on the left. The results are the same for $\{A, B, C\}$ and $\{A, B, C, D\}$. For $\{A, B, C, D, E\}$, there are seven different placements marked by a red shield in Figure 4.5e. However, the resulting scores, 81 and 80 respectively, are very close.

Chapter 5

Practical Application

This chapter talks about the details of applying our technique to iterative proxy combinatorial auctions. The discussion is focused on how we visualize the additional data attributes using the visual features color, texture, and motion.

5.1 Testbed

The testbed for our technique was a proxy agent combinatorial auction generated in Dr. Peter Wurman's e-commerce laboratory [Wur04]. In the example dataset we studied, four buyers bid on combinations of three objects, A, B, and C. Each buyer has exactly one agent bidding for him in the auction. Before the auction begins, buyers provide the agents with the highest values they are willing to pay for each possible bundle. Figure 3.1 (Chapter 3) shows the values used in this testbed.

Figure 3.4 shows the original tabular representation for the combinatorial auction results. It

contains bidding information from six of the auction's ten time steps. Each step is an inflection point. Recall, inflection points are times when one of some events occurs. The table includes bundle prices at the designated time, all potentially competitive allocations (with those that remain competitive in the next round with an asterisk), each agent's potential demand set, and each agent's allocation of attention.

These attribute values are the source of our input data. We focused on four attributes: bundle prices, each agent's allocation of attention, competitive allocations, and the times of inflection points. Figure 5.1 shows ten snapshots corresponding to Figure 3.4 from our animated visualization.

5.1.1 Bundle Prices

Two visual features, color and blur, are used to visualize bundle prices. The minimum and maximum prices for each bundle are known before the auction begins, which allows us to parameterize a bundle's price into the range [0,1]. The bundle's color is selected by mapping the parameterized value onto a rainbow scale (on the bottom-right in each snapshot). A constantly increasing bundle price results with an arc's changing color during animations.

The rainbow colors are selected in the CIE LUV model, with corresponding colors guaranteed to have a constant, monotonically increasing luminance within the monitor's gamut. The result is a set of colors selected from a perceptually balanced color model with equal perceptual distance. This feature helps viewers compare ratios between different bundles based on their colors.

Blur is applied to bundles with no new bids to filter them out and give viewers motivation to focus attention on updated bundles. Blur is a method of highlighting suggested by Kosara et al. (2002). He showed that blurring less relevant data can make more important information stand out [KMH02]. He called this technique *semantic depth of field*, similar to the method used in photography to focus on a target subject and imply other details through an out-of-focus background.

5.1.2 Agents' Allocation of Attentions

Four subregions are embedded within each arc in the visualization representing the four proxy agents who could be bidding on a given bundle. The embedded regions use color and transparency to represent four possible states of an agent's bidding status:

1. ***White*** means that the bidder is a candidate to win this bundle.

2. ***Grey*** means that the bidder is interested in this bundle and he is increasing his bid.

3. ***Transparent*** means that the bidder has no interest and does not bid on this bundle.

4. ***Dark Green*** means that the bidder is interested in this bundle, but he does not increase his bid.

We tried to visualize each agent's numeric allocation of attention. To obtain a representative value, each agent's allocation of attention for a bundle is divided by the total attention from all agents for that bundle. This value represents how strong an agent's attention is relative to all the other agents who are interested in this bundle. This number is mapped to the rainbow scale,

similar to the mapping between bundle prices and colors. These colors vary over time to reflect changes in the agents' allocation of attention. Unfortunately, this resulted in a complex scene with many different colored regions of varying sizes. Viewers had difficulties managing this amount of dynamic information, so we decided to reduce the scale for allocation of attention to one of four static states.

5.1.3 Competitive Allocations

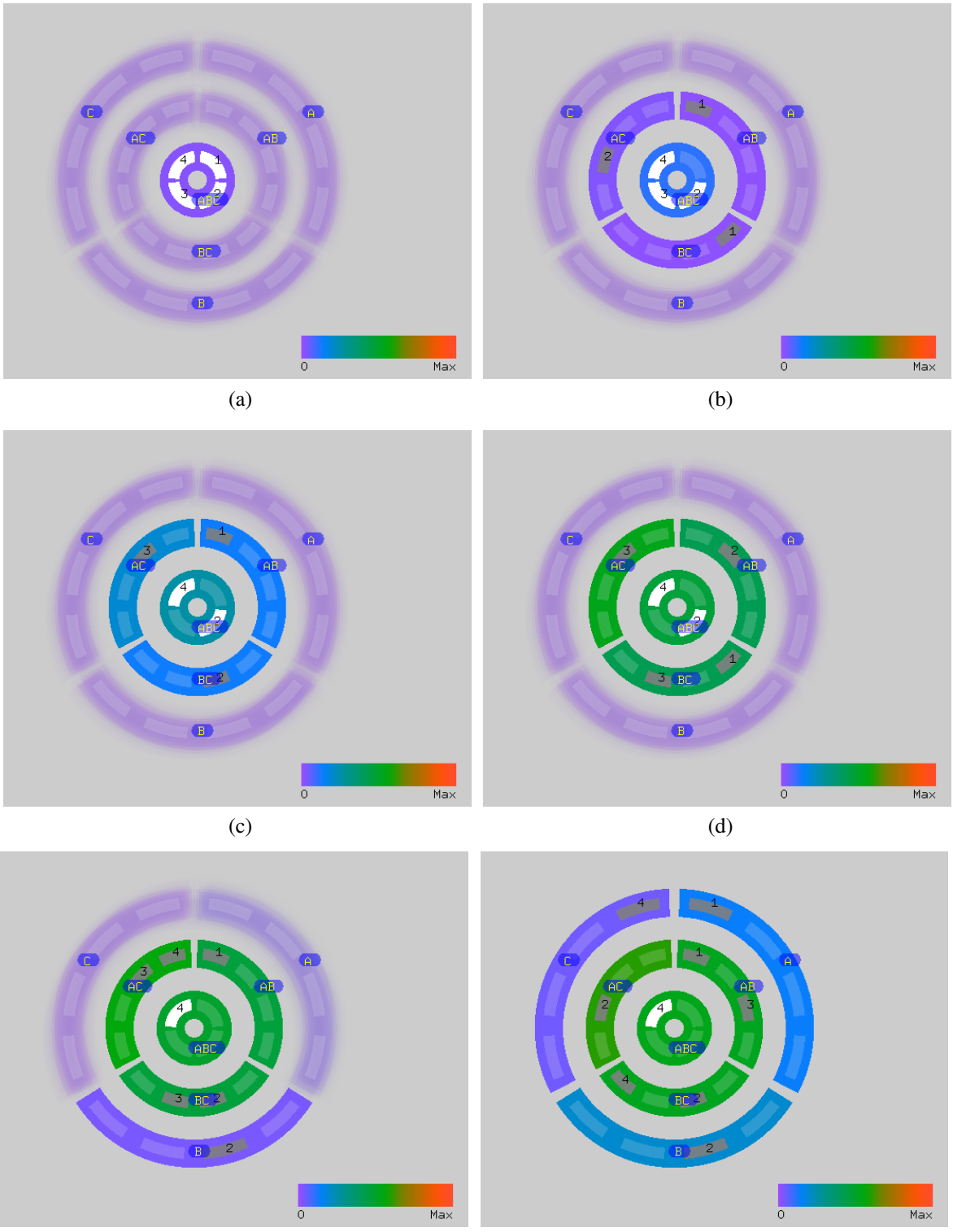
We visualize all competitive allocations that could be chosen by the auctioneer at a given inflection point. Potentially winning agents in each competitive allocation are displayed as white embedded regions within an arc, as described in the previous section. Collections of winning bundles in a common competitive allocation are connected by checked curves.

To avoid interference, Ware recommends assigning new data to the least used graphical dimension in a visualization [War04]. Color is already used to represent bundle prices, so we use a texture on the curves to distinguish common winning bundles across multiple competitive allocations.

Different inflection points may have different competitive allocations, which means that the specific winning bundles are also different. Our visualization shows the competitive allocations for the current inflection point. As a new inflection point is fetched, current competitive allocations are updated simultaneously, with the curves being redrawn to connect new winning candidates.

5.1.4 Times of Inflection Points

The animations detail changes between inflection points. Before the animations begin, our visualization reads in all the information from the first two inflection points and assigns them to the current state and the next state, respectively. The difference in each bundle price between these two inflection points is divided by the difference of the two inflection points' times to get the price rate. When the animations begin, each bundle price increases in its rate while the other attributes' values from the current state are displayed. A local timer simulates the auction's progression. When the timer exceeds the time of the next state, it shifts the attribute values from current state to next state. It then reads in the information for the next inflection point, setting it as the new next state. After shifting states, each bundle's price rate is recomputed. Attribute values for the remaining attribute values are updated to reflect the new current state. Our visualization continues fetching inflection points until auction ends.

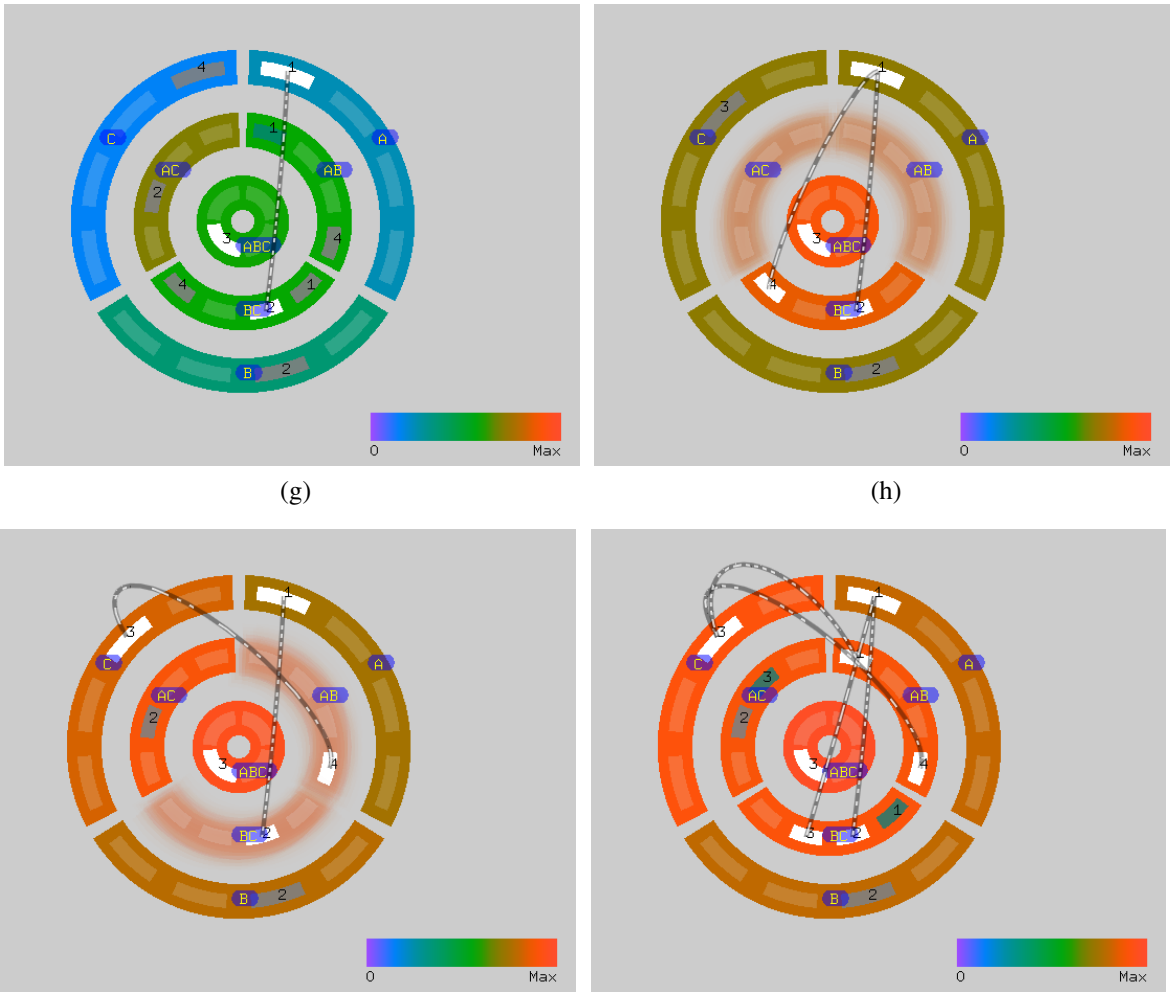


(a) (b)

(c) (d)

(e) (f)

Figure 5.1: The snapshots for each auction round using input of Figure 3.4



(g) (h)

(i) (j)

Figure 5.1: The snapshots for each auction round using input of Figure 3.4

Chapter 6

Discussion and Future Work

This thesis describes a new method for visualizing combinatorial auctions that use combinatorial datasets as their underlying data structures. Two diagrams, Hass diagrams and Venn diagrams, are the conventional methods used to represent these exponentially growing datasets. Both of them, however, are not designed for certain applications and have noticeable limitations. In Hass diagrams, the edges representing the subsets' relationships usually cross each other, producing a complex scene with many crowded lines. In Venn diagrams, there are no line problems, however, screen space is not proportionally allocated. Non-uniformly shrinking regions may be impractical for our application.

We present a concentric visualization to try to solve these issues by providing a well-organized structure. Our visualization uses one or more arcs in each ring to represent potential bundles of items. A well-organized visual representation will have bundles with common items overlapping their arc zones across rings. Instead of trying all the possible combinations

to locate the optimal organization, we developed an algorithm to search for high quality organizations while exploring only a small portion of the search space.

Color, texture, and motion, are applied to represent data attributes within the combinatorial auction. The data is divided across inflection points representing points in time where significant information changes. We record four important data attributes: the inflection point time, bundle prices, each agent's allocation of attention, and possible competitive allocations. Inflection point times trigger our visualization to fetch information for the next inflection point, and to update the static information being displayed to the viewer. Bundle prices are represented by colors selected from a rainbow scale. Agents' allocations of attention are reduced to one of four static states displayed with different colors and transparencies. Finally, potential winners in a competitive allocation are connected using textured curves. Common winner candidates across different competitive allocations are distinguished by curves with different textures.

From our visualizations, we observe a general bidding strategy. In Figure 5.1(a), there are no bids on any bundles other than the entire collection of items (the core, inner most ring). Beginning in Figure 5.1(b), some bidders attempt to bid on other bundles of size two. In Figure 5.1(e), the first bid on a bundle with a single item B appears. A simple interpretation of these images shows that at first, all of the agents try to win the bundle containing all available items. As this bundle's price increases rapidly, agents distribute their bids on other bundles with fewer items to avoid the highly competitive single bundle.

Our visualizations are currently limited to the four data attributes we choose to visualize. Visualizing additional information may produce a scene which is difficult to interpret. In the

future, we hope to include more data and to identify visual feature combinations to represent this data in an effective format.

Bibliography

[CM88] W. S. Cleveland and M. E. McGill, editors. *Dynamic Graphics for Statistics*. Wadsworth, Monterey, CA, 1988.

[CMS99] Stuart K. Card, Jock D. Mackinalay, and Ben Shneiderman. *Readings in Information Visualization : Using Vision to Think*. Morgan Kaufmann, San Francisco, 1999.

[Cra02] P. Cramton. Spectrum auctions, 2002.

[Fur86] George W. Furnas. Generalized fisheye views. In *Proceedings of CHI'86 ACM Conference on Human Factors and Computing Systems*, pages 16–23, New York, 1986. ACM.

[GFG+94] N.D. Gershon, R.M. Friedhoff, J. Grass, R. Langridge, H.P. Meinzer, and J.D. Pearlman. Is visualization really necessary? the role of visualization in science, engineering and medicine. *21st Annual Conference on Computer Graphics and Interactive Techniques*, 10(1):499–500, 1994.

[GOO06] GOOGLE.COM. Google local[online]. Website: http://map.google.com[Accessed 11 April 2006], 2006.

[Hea01] Christopher G. Healey. Formalizing artistic techniques and scientific visualization for painted renditions of complex information spaces. pages 371–376. IJCAI01, 2001.

[KMH02] Robert Kosara, Silvia Miksch, and Helwig Hauser. Focus + context taken literally. *j-IEEE-CGA*, 22(1):22–29, January 2002.

[LS87] J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[MCG05] Irith Ben-Arroyo Hartman Martin Charles Golumbic. *Graph Theory, Combinatorics and Algorithms*. Springer, 2005.

[MDB87] Bruce H. McCormick, Thomas A. Defanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6):1–14, 1987.

[MRC91] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of CHI'91, ACM Conference of Human Factors in Computing Systems*, pages 173–179, New York, April 1991. 1991.

[Nor93] Donald A. Norman. *Things that make us smart*. Addison-Wesley, 1993.

[ONH03] Chihiro Ono, Satoshi Nishiyama, and Hiroki Horiuchi. Designing a double-sided combinatorial auction system. In *Proceedings of the International Conference on Information and Knowledge Engineering 2003*, pages 445–451. CSREA Press, Bogart, GA 30622, United States, 2003.

[PWS05] William Playfair, Howard Wainer, and Ian Spence. *Playfair's Commercial and Political Atlas and Statistical Breviary*. Cambridge University Press, 2005.

[RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees : Animated 3d visualizations of hierarchical information. In *Proceedings of CHI'91, ACM Conference on Human Factors in Computing Systems*, pages 189–194, New York, 1991. ACM.

[RSB82] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[RTM+03] Theresa-Marie Rhyne, Melanie Tory, Tamara Munzner, Matt Ward, Chris Johnson, and David H. Laidlaw. Panel session: Information and scientific visualization: Separate but equal or happy together at last. In *Proceedings of IEEE Visualization'03*, 2003.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps : 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, January 1992.

[Shn06] Ben Shneiderman. Treemaps for space-constrained visualization of hierarchies[online]. Website: http://www.cs.umd.edu/hcil/treemap-history/[Accessed 02 May 2006], 2006.

[SR05] Jiongjiong Song and Amelia Regan. Approximation algorithms for the bid construction problem in combinatorial auctions for the procurement of freight transportation contracts. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 39(10):914–933, 2005.

[Tre94] L. Treinish. Ozone animation. 1994.

[Tri85] A. Triesman. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31:156–177, 1985.

[Tuk77] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[Ven80] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 9(9):1–18, 1980.

[War04] Colin Ware. *Information Visualization: Perception for Design*. MORGAN KAUFMANN, 2004.

[Wol94] Jeremy M. Wolfe. Guided search 2.0: A rerevised model of visual search. *Psychonomic Bulletion & Review*, 1(2):202–238, 1994.

[Wur04] *Computing Price Trajectories in Combinatorial Auctions with Proxy Bidding*. Electronic Commerce Research and Applications, 2004.

[ZN01] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions, 2001.