

**Applications of Auction Algorithms to Complex Problems with Constraints**

by

John N. Dukellis

B.S. Computer Science and Engineering, B.S. Management Science  
Massachusetts Institute of Technology, 1999

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND  
COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF

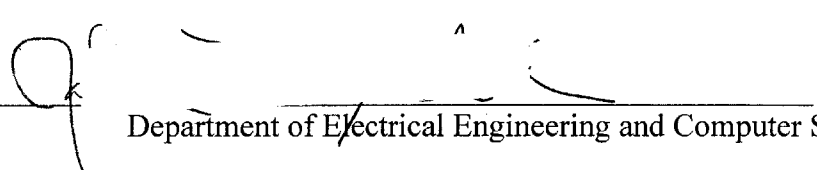
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER  
SCIENCE  
AT THE  
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2000


© 2000 John N. Dukellis. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and  
electronic copies of this thesis document in whole or in part.

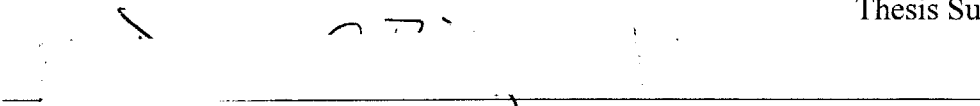
Signature of Author

  
Department of Electrical Engineering and Computer Science

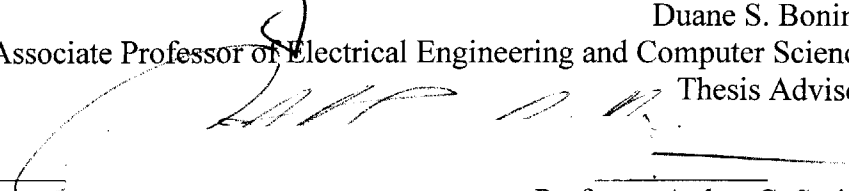
Certified by

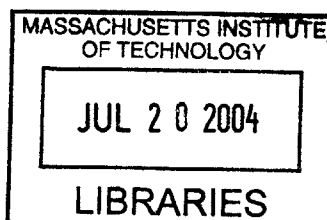
  
Dr. Owen Deutsch  
Charles Stark Draper Laboratory  
Thesis Supervisor

Certified by

  
Duane S. Boning  
Associate Professor of Electrical Engineering and Computer Science  
Thesis Advisor

Accepted by

  
Professor Arthur C. Smith  
Chairman, Departmental Graduate Committee



**BARKER**

# **Applications of Auctions Algorithms to Complex Problems with Constraints**

by

John N. Dukellis

Submitted to the Department of Electrical Engineering and Computer Science on March 24, 2000, in partial fulfillment of the requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

## **Abstract**

---

Linear and nonlinear assignment problems are addressed by the use of auction algorithms. The application of auction to the standard linear assignment problem is reviewed. The extension to nonlinear problems is introduced and illustrated with two examples. Techniques that are employed for model reduction include discretization, classification, and imposition of assignment constraints. The tradeoff between solution speed and optimality for the nonlinear problem is analyzed and demonstrated for the sample problem.

---

Technical Supervisor: Dr. Owen Deutsch  
Title: Senior Member of the Technical Staff

Thesis Advisor: Duane S. Boning  
Title: Associate Professor of Electrical Engineering and Computer Science

## ACKNOWLEDGMENT

March 24, 2000

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Contract No. N68936-99-C-0169 for the NAWC/ONR.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Those deserving my appreciation in the development of my perspective and thesis include Mary Jane Dukellis for her unconditional support, Dr. Bertsekas for his outstanding progress in the field of auction, Dr. Deutsch for his incessant enthusiasm toward the topic and my work and for fueling ideas through discussions, Professor Boning for his expeditious and pragmatic remarks, Simo Kamppari for his influence in pursuing integrity to the fullest through my Masters experience, Michaela Fradin for her sheer positivity and understanding, David White for his exemplary and inspiring work ethic, Ray Conley and Sean Warnick for substantially increasing my motivation in complementary ways, Howard Man for his respect and for enhancing my experience through Sloan and its social environment, Carolyn and Elisabeth for their shared happiness, Michael Bos for his impact on my thinking, Anne Hunter for her unparalleled help through the process and for her devotion to the students of EECS at MIT, Lisa and Niki Dukellis for their contribution to my pride, and Bruce, Stephanie, and Jack Mitchener for their elevation of my desire to succeed. Through the process, I have grown at a rate that unquestionably provides me with a personal happiness and sense of accomplishment, surpassing any notion I had prior to entering graduate school. I look back and am utterly thankful for this precious opportunity and pledge to make possible the power of the experience to others deserving.

---

John N. Dukellis

# Applications of Auction Algorithms to Complex Problems with Constraints

## 1 Introduction

## 2 Auction Algorithms—A Theoretical Overview

2.1 Motivation

2.2 Formulation

2.3 Auction Process Overview—Forward Auctions

2.4 Complimentary Slackness and  $\epsilon$ -Complimentary Slackness

2.5  $\epsilon$ -Scaling and Price Wars

2.6 Similarity Classes and Price Wars

2.6.1 Auctions with Similar Objects

2.6.2 Auctions with Similar Bidders

2.7 Reverse Auctions

2.8 Combined Forward-Reverse Auctions

2.9 Asymmetric Assignment

2.9.1 A Modified Asymmetric Forward-Reverse Auction Algorithm

2.9.2 Relaxing the Constraint of Bidder Assignment

2.10 Multi-Assignment Problems

2.10.1 With All Bidders Assigned Constraint

2.10.2 Linear Combinatorial Auctions—Relaxing the All Bidders Assigned Constraint

2.11 Non-Linear Combinatorial Optimization Using Auctions

## 3 Distributor Problem

3.1 Problem Description

3.2 Constraints

3.3 Complexity

3.4 Reducing and Formulating the Problem

3.4.1 Bucketing

3.4.2 Determining Computational Requirement

3.5 Setting Up the Auctions

## 4 Extension of Distributor Problem to Nonlinear Bidder Behavior

4.1 Problem Description

4.2 Nonlinear Object Behavior

4.3 Nonlinear Bidder Behavior—Single Object-Type Constraint per Bidder Group

4.4 Nonlinear Bidder Behavior—Single Object-Type per Bidder

4.5 Nonlinear Bidder Behavior Based on Items and Not Bidders

4.6 The Makings of a Good Nonlinear Bidder Group

## 5 Conclusion

## Appendices

Derivation of Forward Auction Run-Time

## Bibliography

## 1 Introduction

This thesis addresses a special class of problems known as assignment problems. Assignment problems take two groups of items and match each item in the first group with a unique item in the second group, or perhaps leave items purposefully unassigned. Many examples of the problem exist in our everyday lives. Take, for example, a hospital emergency room where there are a limited number of doctors and nurses who must help patients with different ailments. Specific doctors are assigned to certain patients based on their proficiencies and the time constraints. For any individual doctor-patient assignment, there is a utility value based on how well the doctor can treat the patient. A surgeon mapped to an asthma sufferer would likely yield a low utility score, while a surgeon working on a patient with a severe laceration would produce a high utility score. At any given moment, there is an optimal assignment of doctors working with patients, where optimal is defined as the maximal aggregate utility over all assignments.

Another example is a distributor with customers and a limited number of items. The distributor has to decide how best to divide the items among the customers so that everyone is best served. This example will be more fully explored in the chapter three.

In either case, there exists a best way to pair all the items, suggesting that there is some overall assignment that is at least as good as, and potentially better than, all other assignments. In the linear case, the total utility (or “goodness”) is calculated by simply adding up the utility of each pairing to reach a total utility score. Some particular overall assignment yields the highest total utility score, even though each item might not be assigned its optimal partner. This type of linear problem is solved proficiently by auction, a special method that reaches a guaranteed optimal solution in polynomial time.

Another, more complicated type of problem exists where the total utility score depends not only on individual pairing scores, but also depends on combinations of pairings. An example of this occurs with software and a computer, where each item is valuable to a consumer only as long as both are purchased. This is an example of nonlinear behavior and requires an exhaustive test of all assignment permutations in order to guarantee optimality. However, in many of these problems, patterns exist which models can take advantage of to reduce computational time. By making simple assumptions, much of the computation can be avoided. Though the solutions are not guaranteed to be optimal outside of the assumptions, good solutions can be produced within a given time constraint.

In this case, the degree to which modeling simplifications are made clearly represents a tradeoff in accuracy versus computational time. The right assumptions can lead to a “good enough” solution in a given amount of time. Given these assumptions, auction can be utilized to yield an optimal result for the modeled problem. This thesis examines problems of this nature and modeling techniques that can reduce the computational requirement to an acceptable level through an increasing level of model simplification.

The structure of this thesis begins in Chapter 2 with a theoretical and practical overview of auction theory. This chapter is a restatement of the reigning auction theory developed primarily by Bertsekas and is designed to prime the reader on the subject. Additionally, it is intended to develop insight into the power of auction and how basic assumptions can lead to fast solutions.

Following the theoretical overview, two problems are given to illustrate the use of auctions in the nonlinear problems described above. These problems and their solution techniques are intended to serve as models for using auction effectively under difficult scenarios, with special attention given to reducing time requirements. The first problem is discussed in Chapter 3 and tackles the issue of nonlinear object combinations. The second problem is presented in Chapter 4, delving further into complexity by also allowing nonlinear bidder combinations. Careful use of assumptions will be shown to reduce intractable problems to levels that can produce accurate results quickly.

The application of auction to a new space of problems has become possible through both theoretical and technological advancement and promises to be used widely in the future. Particularly, the scaling ability of auction due to its polynomial time operation lends powerful potential to its use. Additionally, the flexibility of determining the level of optimality through its approximation characteristic allows application to problems with computational time constraints to guarantee solutions within specific optimality boundaries. This feature undoubtedly enhances the attractiveness of auction.

For a textbook approach to auction, as well as the original derivations, the reader is referred to Bertsekas in [4] and [6].

## 2 Auction Algorithms—A Theoretical Overview

### 2.1 Motivation

Assignment problems, where each item from one list is paired with a distinct item from a second list to yield an overall optimality over all pairs, have become commonplace in today's world. Moreover, the confluence of increasing computational power and larger available memory, both with reduced prices, has enlarged the sphere of problems that can be solved. Specifically, the scale of problems which can be solved in reasonable time has grown tremendously, allowing many new applications to benefit from analysis and implementation of assignment algorithms.

Traditionally, linear assignment problems, where the overall optimality is determined by summing the utility scores of each pairing to yield an overall utility score, had been solved using the simplex method (primal descent) or the primal-dual method (dual ascent). These methods were originally proposed by [12] and [14], respectively; see [4] and [10] for discussion. However, the worst case times for the simplex method is exponential,  $O(c^n)$  where  $c$  is a constant and  $n$  is the number of items in the problem. The dual ascent run-time can be shown to be pseudopolynomial. The auction method, a departure from primal-dual methods, is an approximate dual ascent method that has been shown to solve an  $n \times m$  problem in pseudopolynomial  $O(n^2mc)$  time [6]. Utilizing specialized scaling techniques, the required time can even be reduced to polynomial  $O(nm \log[c*n])$ ; see [1], [2] and [6]. Additionally, despite its typecast of "approximate dual ascent," the solutions are optimal to the degree desired, in which known bounds can be placed on the solution prior to running the algorithm.



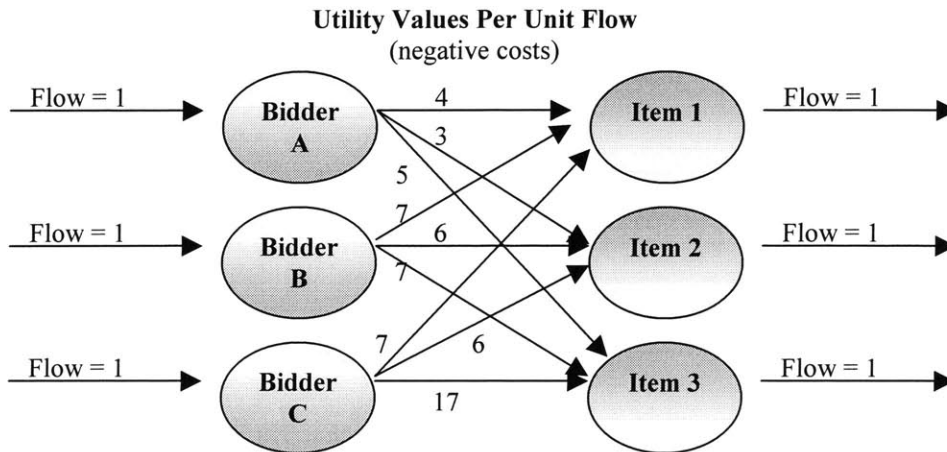
The polynomial growth of the solution technique allows practical application to significantly larger problems than its counterparts. It allows more flexibility in the modeling of problems to capture a larger portion of the relevant information given the computational and time constraints. Solutions are improved in two ways. First, the sheer scale of the problems can be increased to handle larger problems in far less time than its counterparts. Secondly, because the auctions are solved more quickly, they can be used as components of larger problems with significant time reduction [13]. One such method to be explored utilizes a series of different auctions to solve a nonlinear problem.

The crux of this thesis examines the application of auction algorithms into a larger space of problems. This chapter restates the current state of auction theory for the convenience of the reader. Techniques of modeling will be discussed that view the tradeoff of computation versus accuracy. Additionally, the auction methodology carries an intuitive interpretation grounded in everyday economics, particularly in the concept of competitive bidding. To the extent that this process is understood, new problems in disparate areas can benefit from the application of auction. It is hoped that this thesis will serve as a primer for auction application to new problems.

## **2.2 Formulation**

The formulation of an auction problem includes one set of entities (bidders) bidding among its members for the items in another set (objects). The set of bidders that can bid for an object  $j$  is denoted  $B(j)$ . Likewise, the set of objects that can be bid on by bidder  $i$  is denoted  $A(i)$ . The problem can be interchanged, such that the bidders become objects and objects become bidders, yielding the dual problem. The fundamental idea is

to achieve the globally optimal pairing of bidders  $B$  to objects  $A$ , as defined in Figure 2, through sequential bidding. This solution, in the traditional symmetric auction (where the number of objects equals the number of bidders), matches each bidder with a unique object such that the sum of the utilities is highest. The problem can be set up as a traditional network flow problem as in Figure 1.



**Figure 1**

Assignment problems can be viewed as maximum network flow problems (or, equivalently, as a minimum network flow problems with the negative utility values). Flow through each bidder must equal 1 and through each item must equal 1.

The network flow problem can likewise be presented as a maximization problem summing the flows and their costs, given the corresponding constraints (Figure 2). The constraints consist of each bidder necessarily being assigned one unique object and each object necessarily being assigned one unique bidder. While the flow of every arc in the network flow diagram is either zero ( $i$  and  $j$  not paired with each other) or one ( $i$  and  $j$  paired with each other), as in the actual auction, it can be modeled as a continuous value between 0 and 1 because the value will always tend to one extreme or the other. This allows mathematical assumptions regarding convexity to hold in the proof of auctions.

$$\text{Maximize } \sum_{i=1}^N \sum_{j=1}^N a(i,j)f(i,j)$$

$a(i,j)$  = Utility of pairing bidder  $i$  and object  $j$   
 $f(i,j)$  = 1 if  $i$  and  $j$  paired together, 0 otherwise (an indicator function)

$$\sum_{j \in A(i)} f(i,j) = 1 \quad \forall i=1, \dots, n \text{ (bidders)} \quad \begin{array}{l} \text{Outflow} = 1 \\ \text{Each bidder assigned once} \end{array}$$

$$\sum_{i \in B(j)} f(i,j) = 1 \quad \forall j=1, \dots, n \text{ (objects)} \quad \begin{array}{l} \text{Inflow} = 1 \\ \text{Each item assigned once} \end{array}$$

**Figure 2**

Assignment problems can be viewed as maximization, where the utilities of each assignment are summed. In the simplest formulation, each bidder and object are constrained to have exactly one assigned partner.

This formulation represents the traditional symmetric problem. Other variations, including the asymmetric case and differing pairing constraints, are addressed later in the chapter, as well as in the actual applications.

### 2.3 Auction Process Overview

Before the auction begins, an initial price must be designated for each object. This price will be used by bidders to compare the price of an object versus the utility it will provide them and will allow them to select the most profitable object. Additionally, the auction must start with a set of assignments such that every assignment is currently best for the particular bidder (known as the complementary slackness condition). Complementary slackness is often observed trivially by starting with no assignments, allowing initial prices to be unconstrained. Prices can begin at any level because the auction is finding the assignment that maximizes the overall absolute utility. Throughout

the auction, prices are used only as a relative indicator among the bidders and objects. Over the course of the auction, the prices will reach their eventual equilibrium with each other and the auction will terminate. Hence, the three auctions indicated in Figure 3 will result in the same outcome of pairings (given identical utilities for all case), though the initial prices will affect the speed and prices at which the auction reaches its conclusion.

Object Number	Auction 1 Initial Price	Auction 2 Initial Price	Auction 3 Initial Price
1	5	50	0
2	20	20	0
3	30	-5	0

**Figure 3**

Three auctions are run with different initial prices. However, because the utilities are identical, the auctions should terminate achieving an overall utility within the same bounds. The initial prices, therefore, do not play a role in determining overall utility. They may, however, impact the speed at which the solution is reached.

In the standard naïve forward auction, an unassigned bidder is selected at each stage to evaluate his options, given the current prices, and place a bid on the best value. Value  $v$  is determined by subtracting the price  $p_j$  of the object  $j$  from the utility  $a_{ij}$  received from the pairing  $ij$ . The value of matching object  $j$  with person  $i$  is

$$v_{ij} = a_{ij} - p_j.$$

Values can take on positive and negative levels, again only presenting a relative comparison among bidders and objects. The condition of the bidder choosing the best possible choice is referred to as “complementary slackness” and will be discussed in Section 2.4.

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{ a_{ik} - p_k \} \quad \forall (i,j) \in S \text{ (set of pairings)}$$

The bid is an amount equal to the price of the object he is bidding for plus the difference in value between the best and second best objects. The intuition is that a bidder will remain loyal to the best object until the point at which he sees more value in moving to the next best. For all the prices up to that point, he will continue to bid for the same object. To eliminate the tedious bidding cycles with small bid increments, the bidder places a bid at the level to which he would bid for the next object. Therefore, he finds the object  $j$  with the most value

$$j = \arg \max_{j \in A(i)} \{ a_{ij} - p_j \}$$

and then solves for the bidding increment  $\Upsilon_i$ , equal to the difference in value between the best value  $v_i$  and the next best value  $\omega_i$ . The bid is then computed as the price of object  $j$  plus the bidding increment.

Bidding increment:	$\Upsilon_i = v_i - \omega_i$
Best value:	$v_i = \max_{j \in A(i)} \{ a_{ij} - p_j \}$
Second best value:	$\omega_i = \max_{k \in A(i), k \neq j} \{ a_{ik} - p_k \}$
Bid:	$\text{Bid}_{ij} = p_j + \Upsilon_i$

If a bidder selects an unassigned object, he becomes assigned to that object at the initial price and the next round begins. If a bidder selects an assigned object, he replaces the formerly assigned bidder. The formerly assigned bidder becomes unassigned and will have to bid in a later round for another (or possibly the same) object. Two methods exist in the bidding stage for determining the bids and objects. In the Gauss-Seidel method,

any unassigned bidder can be chosen arbitrarily and undergo the bidding process. A second method, the Jacobian method, evaluates all unassigned bidders and calculates the bids for all of them. The best bids for each object are then selected and processed simultaneously. The trade-off occurs in whether the calculations of finding the highest bidder each round (in the Jacobian version) are more expensive than additional bidding rounds required because a suboptimal object was assigned one or more times in a previous round. The Gauss-Seidel method can potentially avoid continuously suboptimal bids by rotating or randomly selecting which bidders are selected, rather than searching through a static preordered list of bidders for the first unassigned bidder. In the static preordered search, two early bidders on the list may slowly, alternately drive the price of an object up, while a third bidder would price the object far above the other two but not be reached for some time because of the ordering.

The auction terminates when all bidders are assigned. At this point, no bidder can increase his utility by bidding for another object, because prices have either risen to or stayed at the level they were at when the bidder placed his best bid on the object to which he is currently assigned. A simple example is given in Figure 4 to demonstrate the procedures of a naïve auction.

## Illustrative Example of Naive Auction

Figure 4

### Utility Values of Assignments

The utility values of each possible assignment are given in the matrix on the right, corresponding to the diagram in Figure 1.

		Objects		
		1	2	3
<b>Bidders</b>	<b>A</b>	4	3	5
	<b>B</b>	7	6	7
	<b>C</b>	7	6	17

The initial conditions to this auction are that there are no assignments and that the prices of objects 1, 2, and 3, respectively, are 5, 20, and 30.

Round 1, shown below, begins with these conditions and (1) selects a bidder who is unassigned (A in this case). The profit of pairing the bidder with each object is then examined in (2). Given the starting prices, the highest utility for the bidder occurs with object 1, where the profit is equal to -1 (3). The second best object is 2, with a profit of -17 (4). A bid is then constructed by adding the profit difference of the best and second best (5) to the price of the best object, which yields a bid of 21 for object 1 (6). The reason this equals the bid is because the bidder would continue to bid for the best object until it passed that price, at which point it would bid for the second best object. Because the overall utility is computed by adding the utilities, the absolute price is actually irrelevant to the overall score, meaning that the importance is in which assignment is made, not the price. Rather than continue to bid a small increment, the entire difference is added to the bid so the next round can occur.

The assignments are made (6), reflecting bidder A's new assignment to object 1, and the price of the newly bid object is updated. The auction then checks to see if there are any bidders still unassigned, which is true, and hence continues to the next round.

Step #	Auction Round #				Assignment (by Object)			Assignment (by Bidder)			
1)	1	Bidder	A			1	2	3	A	B	C
		Objects	1	2	3	-	-	-	-	-	-
		Starting Prices	5	20	30						
		Utilities	4	3	5						
2)		Profit	-1	-17	-25						
		(Profit = Utility - Price)									
3)		Best Object, Profit	1	-1							
4)		2nd Best Object, Profit	2	-17							
5)		Difference	16								
6)		Object Bid On, Bid	1	21		<b>Ending Assignment</b>			<b>Ending Assignment</b>		
		(Bid = Start Price + Difference)				A	-	-	1	-	-
7)		Ending Prices	21	20	30						

Round 2, shown below, begins with the ending prices and assignments of round 1. An unassigned bidder B is selected as the bidder in round 2 (1). Again, the profits for each assignment are computed, using the utility table and current prices (2). Object 1 and 2 appear equally attractive, each with profits of -14 (3), so the bidder randomly selects object 2 (3). The difference is computed to be 0 (5) and the bid is calculated to be the price of object 2 plus the difference of 0, equal to 20 (6). The assignments are updated, noting bidder B is now assigned to object 2 (6), at a price of 20 (7).

Step #	Auction Round #				Assignment (by Object)			Assignment (by Bidder)			
1)	2	Bidder	B			1	2	3	A	B	C
		Objects	1	2	3	A	-	-	1	-	-
		Starting Prices	21	20	30						
		Utilities	7	6	7						
2)		Profit	-14	-14	-23						
		(Profit = Utility - Price)									
3)		Best Object, Profit	2	-14							
4)		2nd Best Object, Profit	1	-14							
5)		Difference	0								
6)		Object Bid On, Bid	2	20		<b>Ending Assignment</b>			<b>Ending Assignment</b>		
		(Bid = Start Price + Difference)				A	B	-	1	2	-
7)		Ending Prices	21	20	30						

Round 3 begins with unassigned bidder C (1). The most attractive objects to bidder C is object 3 with a profit of -13 (3). The next most attractive is object 1 or 2, each with a profit of -14 (4). The bidder therefore chooses object 3, bidding 31 (6), which is equal to the starting price of 30 plus the profit difference of 1 (5). The assignments are updated (6), along with the price update of object 3 (7).

Step #	Auction Round # 3			Assignment (by Object)			Assignment (by Bidder)			
1)	Bidder C			1	2	3	A	B	C	
	Objects	1	2	3	A	B	-	1	2	-
	Starting Prices	21	20	30						
	Utilities	7	6	17						
2)	Profit	-14	-14	-13						
	(Profit = Utility - Price)									
3)	Best Object, Profit	3	-13							
4)	2nd Best Object, Profit	1	-14							
5)	Difference	1								
6)	Object Bid On, Bid	3	31							
	(Bid = Start Price + Difference)									
7)	Ending Prices	21	20	31						
					Ending Assignment			Ending Assignment		
					A	B	C	1	2	3

Since all bidders have been assigned, the auction terminates. The ending assignments and utilities are

A1	4
B2	6
C3	17
<b>Total</b>	<b>27</b>



## 2.4 Complementary Slackness and $\epsilon$ -Complementary Slackness

The complementary slackness (CS) condition requires that the value received by the bidder for the object (utility minus price) is the highest possible given all current object prices. Hence, *a bidder would never choose a less than optimal object at any given stage.*

$$\text{Complementary Slackness: } a_{ij} - p_j \geq \max_{k \in A(i)} \{ a_{ik} - p_k \} \quad \forall (i,j) \in S$$

The auction presented above requires that a bidder bid at least the current price for an object, given the bidding increment is always non-negative. However, it is easy to construct a scenario in which three bidders desiring the same object at the same level will enter into an endless cycle, each subsequently bidding for the object at the same price. Because of this, the auction presented above is referred to as the naïve auction. The method for fixing this problem is to require each successive bid to be more than the previous, so that eventually one of the bidders will find it more attractive to bid for a different object. Figure 5 demonstrates a naïve auction that never terminates.

# Illustrative Example of Non-terminating Naïve Auction

Figure 5

Utility Values of Assignments  
Objects

The utility values of each possible assignment are given in the matrix on the right. The utility of C3 has changed from 17 in Figure 4 to 4 here.

		1	2	3
Bidders	A	4	3	5
	B	7	6	7
	C	7	6	4

The initial conditions to this auction are that there are no assignments and that the prices of objects 1, 2, and 3, respectively, are 5, 20, and 30.

Round 1 proceeds identically to that in Figure 4.

Step #	Auction Round #	1			Assignment (by Object)	Assignment (by Bidder)
1)	Bidder	A			1    2    3	A    B    C
	Objects	1	2	3	-    -    -	-    -    -
	Starting Prices	5	20	30		
	Utilities	4	3	5		
2)	Profit	-1	-17	-25		
	(Profit = Utility - Price)					
3)	Best Object, Profit	1	-1			
4)	2nd Best Object, Profit	2	-17			
5)	Difference	16				
6)	Object Bid On, Bid (Bid = Start Price + Difference)	1	21		<b>Ending Assignment</b> A    -    -	<b>Ending Assignment</b> 1    -    -
7)	Ending Prices	21	20	30		

Round 2 begins with the ending prices and assignments of round 1. An unassigned bidder B is selected as the bidder in round 2 (1). Again, the profits for each assignment are computed, using the utility table and current prices (2). Object 1 and 2 appear equally attractive, each with profits of -14 (3), so the bidder randomly selects object 1 (3). The difference is computed to be 0 (5) and the bid is calculated to be the price of object 1 plus the difference of 0, equal to 21 (6). The assignments are updated, noting bidder B is now assigned to object 1 (6), at a price of 21 (7), and bidder A is becomes unassigned.

Step #	Auction Round #	2			Assignment (by Object)	Assignment (by Bidder)
1)	Bidder	B			1    2    3	A    B    C
	Objects	1	2	3	A    -    -	1    -    -
	Starting Prices	21	20	30		
	Utilities	7	6	7		
2)	Profit	-14	-14	-23		
	(Profit = Utility - Price)					
3)	Best Object, Profit	1	-14			
5)	Difference	0				
6)	Object Bid On, Bid (Bid = Start Price + Difference)	1	21		<b>Ending Assignment</b> B    -    -	<b>Ending Assignment</b> -    1    -
7)	Ending Prices	21	20	30		

Round 3 begins with unassigned bidder C (1). The most attractive objects to bidder C are objects 1 and 2. The bidder therefore chooses either object, being indifferent, which in this case he bids on object 2 (3). However, there is no price increment (5), but again a bid equal to the current price, this time of object 2, of 20 (6). The assignments are updated (6) and the prices are kept the same.

Step #	Auction Round #	3			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	C			1	2	3	A	B	C
	Objects	1	2	3	B	-	-	-	1	-
	Starting Prices	21	20	30						
	Utilities	7	6	4						
2)	Profit	-14	-14	-26						
	(Profit = Utility - Price)									
3)	Best Object, Profit	2	-14							
4)	2nd Best Object, Profit	1	-14							
5)	Difference	0								
6)	Object Bid On, Bid	2	20		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference)				B	C	-	-	1	2
7)	Ending Prices	21	20	30						

Round 4 starts with bidder A computing his profits, noticing that he is indifferent to objects 1 and 2. He therefore chooses either object, but the new bid is exactly the same as the old bid. He nevertheless replaces the old owner and the next round continues with which ever bidder he ousted this round. The problem is that both bidders he is ousting will view the two objects indifferently also and the prices will never be incremented. Hence, the auction will continue for an infinite number of rounds, with no bidder ever bidding on object 3.

Step #	Auction Round #	4			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	A			1	2	3	A	B	C
	Objects	1	2	3	B	C	-	-	1	2
	Starting Prices	21	20	30						
	Utilities	4	3	5						
2)	Profit	-17	-17	-25						
	(Profit = Utility - Price)									
3)	Best Object, Profit	1	-17							
4)	2nd Best Object, Profit	2	-17							
5)	Difference	0								
6)	Object Bid On, Bid	1	21		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference)				A	C	-	1	-	2
7)	Ending Prices	21	20	30						

The actual auction algorithm used in practice incorporates a required increment in each bid of value  $\epsilon$ , in addition to the bid determined in the naïve auction.

Bidding increment: 
$$\gamma_i = v_i - \omega_i + \epsilon$$

This precludes the possibility of infinite cycling among an equally valued object by two bidders and guarantees the auction will finish in a fixed amount of time. The requirement forces each bid to increase the bidding amount by a minimum of  $\epsilon$ . Hence, eventually a bidder will bid on a new object because the price has increased enough to make another object a more profitable choice. While this deviates from the original CS condition, it is replaced by an  $\epsilon$ -CS condition, which states that a bidder's selection is within  $\epsilon$  of optimal.

$\epsilon$ -Complementary Slackness: 
$$a_{ij} - p_j \geq \max_{k \in A(i)} \{ a_{ik} - p_k \} - \epsilon \quad \forall (i,j) \in S$$

With each bidder within  $\epsilon$  of being optimal, the total deviation is at most  $\epsilon$  times the number of bidders. Therefore, by insuring that  $\epsilon^* n$  is less than the utility granularity, the solution will be optimal. Assuming the utility granularity is to the integer,  $\epsilon$  must be less than  $1/n$ . This allows  $n$  different bidders to sequentially bid for the object, if it is in fact the optimal choice, before the price increases to the next level of value. If each bidder is therefore within  $\epsilon$  of optimal, all bidders together must be within  $\epsilon^* n$ , which is less than 1. Since it is impossible to be suboptimal by less than one if utilities are integer (the minimum suboptimality would be at least one), the solution must be optimal. Figure

6 presents an auction using  $\epsilon$ -complementary slackness, which guarantees a solution within  $n * \epsilon$  of optimality.

A practical implementation sets  $\epsilon$  equal to one and each utility equal to  $(n+1) * (\text{Integer Utility})$ , increasing utility granularity to the requisite level and requiring only integers to be used in the program. This is identical to the previous argument, but multiplying all terms by  $n+1$ . Figure 7 modifies the auction in Figure 6 by multiplying the integer utilities by  $n+1$ , yielding the optimal solution.

An important issue in using  $\epsilon$  is that there can at most be  $\lceil \max_{(i,j)} |a_{ij} - p_{j(\text{initial})}| / \epsilon \rceil$  bids for each object, or  $O(c)$ , where  $c$  is a constant. By multiplying the utilities by  $n$  as above, there can be at most  $n$  times this amount of bids for each object. With  $n$  objects, total possible number of bids in the auction to the number of objects  $n$  times the number of bids per object, a total maximum of  $n^2 * \lceil \max_{(i,j)} |a_{ij} - p_{j(\text{initial})}| / \epsilon \rceil$ , or  $O(n^2 c)$ . Each round requires  $O(n)$  operations to find the best bid, examining the profit of pairing the object with each of  $n$  items. Hence, the total running time of the auction becomes  $O(n^3 \lceil \max_{(i,j)} |a_{ij} - p_{j(\text{initial})}| / \epsilon \rceil)$ , or essentially  $O(n^3 c)$ , which is pseudopolynomial, where  $c$  is a constant.

## Illustrative Example of Modified Auction (optimal within $n \cdot \epsilon = 3$ )

Figure 6

Everything in this example is identical to Figure 5 except that every bid is increased by an additional value epsilon equal to 1.

		Utility Values of Assignments		
		Objects		
		1	2	3
Bidders	A	4	3	5
	B	7	6	7
	C	7	6	4

Round 1 begins identically, choosing a bidder (1), finding the profits (2), and calculating the difference (5) of the best (3) and second best (4) objects. However, the bid is now equal to the starting price plus the difference plus epsilon (epsilon = 1). Hence, the assignment is identical to last time (6), but the bid (6) and ending prices (7) are higher.

Step #	Auction Round #	1			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	A			1	2	3	A	B	C
	Objects	1	2	3	-	-	-	-	-	-
	Starting Prices	5	20	30						
	Utilities	4	3	5						
2)	Profit	-1	-17	-25						
	(Profit = Utility - Price)									
3)	Best Object, Profit	1	-1							
4)	2nd Best Object, Profit	2	-17							
5)	Difference	16								
6)	Object Bid On, Bid	1	17		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)									
7)	Ending Prices	22	20	30	A	-	-	1	-	-

Round 2 proceeds as usual. The difference here is 1, so the bid is equal to the start price of 20 plus the difference of 1 plus 1 for epsilon. Hence the bid is 22 for object 2.

Step #	Auction Round #	2			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	B			1	2	3	A	B	C
	Objects	1	2	3	A	-	-	1	-	-
	Starting Prices	22	20	30						
	Utilities	7	6	7						
2)	Profit	-15	-14	-23						
	(Profit = Utility - Price)									
3)	Best Object, Profit	2	-14							
4)	2nd Best Object, Profit	1	-15							
5)	Difference	1								
6)	Object Bid On, Bid	2	22		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)									
7)	Ending Prices	22	22	30	A	B	-	1	2	-

Step #	Auction Round #	3			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	C			1	2	3	A	B	C
	Objects	1	2	3	A	B	-	1	2	-
	Starting Prices	22	22	30						
	Utilities	7	6	4						
2)	Profit	-15	-16	-26						
	(Profit = Utility - Price)									
3)	Best Object, Profit	1	-15							
4)	2nd Best Object, Profit	2	-16							
5)	Difference	1								
6)	Object Bid On, Bid	1	24		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)									
7)	Ending Prices	24	22	30	C	B	-	-	2	1

<b>Step #</b>	<b>Auction Round #</b>	<b>4</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>A</b>			<b>1</b> <b>2</b> <b>3</b>		<b>A</b> <b>B</b> <b>C</b>
	<b>Objects</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>C</b> <b>B</b> <b>-</b>		<b>-</b> <b>2</b> <b>1</b>
	<b>Starting Prices</b>	<b>24</b>	<b>22</b>	<b>30</b>			
	<b>Utilities</b>	<b>4</b>	<b>3</b>	<b>5</b>			
2)	<b>Profit</b>	<b>-20</b>	<b>-19</b>	<b>-25</b>			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	<b>2</b>	<b>-19</b>				
4)	<b>2nd Best Object, Profit</b>	<b>1</b>	<b>-20</b>				
5)	<b>Difference</b>	<b>1</b>					
6)	<b>Object Bid On, Bid</b>	<b>2</b>	<b>24</b>		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	<b>24</b>	<b>24</b>	<b>30</b>	<b>C</b> <b>A</b> <b>-</b>		<b>2</b> <b>-</b> <b>1</b>

<b>Step #</b>	<b>Auction Round #</b>	<b>5</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>B</b>			<b>1</b> <b>2</b> <b>3</b>		<b>A</b> <b>B</b> <b>C</b>
	<b>Objects</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>C</b> <b>A</b> <b>-</b>		<b>2</b> <b>-</b> <b>1</b>
	<b>Starting Prices</b>	<b>24</b>	<b>24</b>	<b>30</b>			
	<b>Utilities</b>	<b>7</b>	<b>6</b>	<b>7</b>			
2)	<b>Profit</b>	<b>-17</b>	<b>-18</b>	<b>-23</b>			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	<b>1</b>	<b>-17</b>				
4)	<b>2nd Best Object, Profit</b>	<b>2</b>	<b>-18</b>				
5)	<b>Difference</b>	<b>1</b>					
6)	<b>Object Bid On, Bid</b>	<b>1</b>	<b>26</b>		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	<b>26</b>	<b>24</b>	<b>30</b>	<b>B</b> <b>C</b> <b>-</b>		<b>-</b> <b>1</b> <b>2</b>

<b>Step #</b>	<b>Auction Round #</b>	<b>6</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>A</b>			<b>1</b> <b>2</b> <b>3</b>		<b>A</b> <b>B</b> <b>C</b>
	<b>Objects</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>B</b> <b>C</b> <b>-</b>		<b>-</b> <b>1</b> <b>2</b>
	<b>Starting Prices</b>	<b>26</b>	<b>24</b>	<b>30</b>			
	<b>Utilities</b>	<b>4</b>	<b>3</b>	<b>5</b>			
2)	<b>Profit</b>	<b>-22</b>	<b>-21</b>	<b>-25</b>			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	<b>2</b>	<b>-21</b>				
4)	<b>2nd Best Object, Profit</b>	<b>1</b>	<b>-22</b>				
5)	<b>Difference</b>	<b>1</b>					
6)	<b>Object Bid On, Bid</b>	<b>2</b>	<b>26</b>		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	<b>26</b>	<b>26</b>	<b>30</b>	<b>B</b> <b>A</b> <b>-</b>		<b>2</b> <b>1</b> <b>-</b>

<b>Step #</b>	<b>Auction Round #</b>	<b>7</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>C</b>			<b>1</b> <b>2</b> <b>3</b>		<b>A</b> <b>B</b> <b>C</b>
	<b>Objects</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>B</b> <b>A</b> <b>-</b>		<b>2</b> <b>1</b> <b>-</b>
	<b>Starting Prices</b>	<b>26</b>	<b>26</b>	<b>30</b>			
	<b>Utilities</b>	<b>7</b>	<b>6</b>	<b>4</b>			
2)	<b>Profit</b>	<b>-19</b>	<b>-20</b>	<b>-26</b>			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	<b>1</b>	<b>-19</b>				
4)	<b>2nd Best Object, Profit</b>	<b>2</b>	<b>-20</b>				
5)	<b>Difference</b>	<b>1</b>					
6)	<b>Object Bid On, Bid</b>	<b>1</b>	<b>28</b>		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	<b>28</b>	<b>26</b>	<b>30</b>	<b>C</b> <b>A</b> <b>-</b>		<b>2</b> <b>-</b> <b>1</b>

<b>Step #</b>	<b>Auction Round #</b>	<b>8</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>B</b>			<b>1</b> <b>2</b> <b>3</b>		<b>A</b> <b>B</b> <b>C</b>
	<b>Objects</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>C</b> <b>A</b> <b>-</b>		<b>2</b> <b>-</b> <b>1</b>
	<b>Starting Prices</b>	<b>28</b>	<b>26</b>	<b>30</b>			
	<b>Utilities</b>	<b>7</b>	<b>6</b>	<b>7</b>			
2)	<b>Profit</b>	<b>-21</b>	<b>-20</b>	<b>-23</b>			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	<b>2</b>	<b>-20</b>				
4)	<b>2nd Best Object, Profit</b>	<b>1</b>	<b>-21</b>				
5)	<b>Difference</b>	<b>1</b>					
6)	<b>Object Bid On, Bid</b>	<b>2</b>	<b>28</b>		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	<b>28</b>	<b>28</b>	<b>30</b>	<b>C</b> <b>B</b> <b>-</b>		<b>-</b> <b>2</b> <b>1</b>

Step #	Auction Round # 9				Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder A				1	2	3	A	B	C
	Objects	1	2	3	C	B	-	-	2	1
	Starting Prices	28	28	30						
	Utilities	4	3	5						
2)	Profit	-24	-25	-25						
	(Profit = Utility - Price)									
3)	Best Object, Profit	1	-24							
4)	2nd Best Object, Profit	2	-25							
5)	Difference	1								
6)	Object Bid On, Bid	1	30		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)				A	B	-	1	2	-
7)	Ending Prices	30	28	30						

Step #	Auction Round # 10				Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder C				1	2	3	A	B	C
	Objects	1	2	3	A	B	-	1	2	-
	Starting Prices	30	28	30						
	Utilities	7	6	4						
2)	Profit	-23	-22	-26						
	(Profit = Utility - Price)									
3)	Best Object, Profit	2	-22							
4)	2nd Best Object, Profit	1	-23							
5)	Difference	1								
6)	Object Bid On, Bid	2	30		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)				A	C	-	1	-	2
7)	Ending Prices	30	30	30						

Step #	Auction Round # 11				Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder B				1	2	3	A	B	C
	Objects	1	2	3	A	C	-	1	-	2
	Starting Prices	30	30	30						
	Utilities	7	6	7						
2)	Profit	-23	-24	-23						
	(Profit = Utility - Price)									
3)	Best Object, Profit	3	-23							
4)	2nd Best Object, Profit	1	-23							
5)	Difference	0								
6)	Object Bid On, Bid	3	31		Ending Assignment			Ending Assignment		
	(Bid = Start Price + Difference + epsilon)				A	C	B	1	3	2
7)	Ending Prices	30	30	31						

The auction terminates. However, note that the assignments yield a score of

A1	4
C2	7
B3	6
<b>Total</b>	<b>17</b>

while a higher scoring assignment exists in the form

A3	5
B2	6
C1	7
<b>Total</b>	<b>18</b>

However, the total score is within  $n * \epsilon = 3 * 1 = 3$  of being optimal. Because the utility values are integer, an assignment that is not optimal will be a minimum of 1 away (and perhaps 2 or some other integer) from the optimal score. Hence, by reducing  $n * \epsilon$  to be less than 1, the solution will be guaranteed to be optimal. Setting epsilon equal to  $3/4$  will allow this to happen. An alternative way to achieve this is to increase the discrete differences in the utility values so that the minimum discreteness is greater than 3. By this argument, the minimum amount a suboptimal assignment can be from optimal is equal to the minimum discrete difference. This allows us to set epsilon equal to whatever value desired (1 in this case) and multiply utilities by a certain value to achieve the discrete level required. Because  $n * \epsilon = 3$  in this case, multiplying the integer utilities (which are currently at a discrete level of 1) by anything more than 3 will create a discrete level greater than 3 and achieve optimality. The added benefit of this is that using integer values for utilities and epsilon allows the entire implementation to occur using integers on a computer.



## Illustrative Example of Modified Auction (Completely Optimal)

Figure 7

This auction modifies the utility matrix so the final assignment is guaranteed to be optimal. To achieve this, the utility table is multiplied by 4 and the same auction is run. It does not matter whether or not the prices are adjusted as well, though it may affect the time until solution.

### New Utility Values of Assignments

		Objects		
		1	2	3
Bidders	A	16	12	20
	B	28	24	28
	C	28	24	16

### Old Utility Values of Assignments

		Objects		
		1	2	3
Bidders	A	4	3	5
	B	7	6	7
	C	7	6	4

Step #	Auction Round #	Bidder	Objects			Assignment (by Object)			Assignment (by Bidder)					
1)	1	A	1	2	3	1	2	3	A	B	C			
			Starting Prices	5	20	30	-	-	-	-	-	-		
			Utilities	16	12	20								
2)			Profit	11	-8	-10								
			(Profit = Utility - Price)											
3)			Best Object, Profit	1	11									
4)			2nd Best Object, Profit	2	-8									
5)			Difference	19										
6)			Object Bid On, Bid	1	25									
			(Bid = Start Price + Difference + epsilon)											
7)			Ending Prices	25	20	30	Ending Assignment	A	-	-	Ending Assignment	1	-	-

Round 2 proceeds as usual. The difference here is 1, so the bid is equal to the start price of 20 plus the difference of 1 plus 1 for epsilon. Hence the bid is 22 for object 2.

Step #	Auction Round #	Bidder	Objects			Assignment (by Object)			Assignment (by Bidder)					
1)	2	B	1	2	3	1	2	3	A	B	C			
			Starting Prices	25	20	30	A	-	-	1	-	-		
			Utilities	28	24	28								
2)			Profit	3	4	-2								
			(Profit = Utility - Price)											
3)			Best Object, Profit	2	4									
4)			2nd Best Object, Profit	1	3									
5)			Difference	1										
6)			Object Bid On, Bid	2	22		Ending Assignment	A	B	-	Ending Assignment	1	2	-
			(Bid = Start Price + Difference + epsilon)											
7)			Ending Prices	25	22	30								

Step #	Auction Round #	Bidder	Objects			Assignment (by Object)			Assignment (by Bidder)					
1)	3	C	1	2	3	1	2	3	A	B	C			
			Starting Prices	25	22	30	A	B	-	1	2	-		
			Utilities	28	24	16								
2)			Profit	3	2	-14								
			(Profit = Utility - Price)											
3)			Best Object, Profit	1	3									
4)			2nd Best Object, Profit	2	2									
5)			Difference	1										
6)			Object Bid On, Bid	1	24		Ending Assignment	C	B	-	Ending Assignment	-	2	1
			(Bid = Start Price + Difference + epsilon)											
7)			Ending Prices	25	24	30								

<b>Step #</b>	<b>Auction Round #</b>	<b>4</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>A</b>			1    2    3		A    B    C
	<b>Objects</b>	1	2	3	C    B    -		-    2    1
	<b>Starting Prices</b>	25	24	30			
	<b>Utilities</b>	16	12	20			
2)	<b>Profit</b>	-9	-12	-10			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	1	-9				
4)	<b>2nd Best Object, Profit</b>	3	-10				
5)	<b>Difference</b>	1					
6)	<b>Object Bid On, Bid</b>	1	27		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	27	24	30	A    B    -		1    2    -

<b>Step #</b>	<b>Auction Round #</b>	<b>5</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>C</b>			1    2    3		A    B    C
	<b>Objects</b>	1	2	3	A    B    -		1    2    -
	<b>Starting Prices</b>	27	24	30			
	<b>Utilities</b>	28	24	16			
2)	<b>Profit</b>	1	0	-14			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	1	1				
4)	<b>2nd Best Object, Profit</b>	2	0				
5)	<b>Difference</b>	1					
6)	<b>Object Bid On, Bid</b>	1	29		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	29	24	30	C    B    -		-    2    1

<b>Step #</b>	<b>Auction Round #</b>	<b>6</b>			<b>Assignment (by Object)</b>		<b>Assignment (by Bidder)</b>
1)	<b>Bidder</b>	<b>A</b>			1    2    3		A    B    C
	<b>Objects</b>	1	2	3	C    B    -		-    2    1
	<b>Starting Prices</b>	29	24	30			
	<b>Utilities</b>	16	12	20			
2)	<b>Profit</b>	-13	-12	-10			
	(Profit = Utility - Price)						
3)	<b>Best Object, Profit</b>	3	-10				
4)	<b>2nd Best Object, Profit</b>	2	-12				
5)	<b>Difference</b>	2					
6)	<b>Object Bid On, Bid</b>	3	33		<b>Ending Assignment</b>		<b>Ending Assignment</b>
	(Bid = Start Price + Difference + epsilon)						
7)	<b>Ending Prices</b>	29	24	33	C    B    A		3    2    1

The auction terminates. Note the optimal total utility score of 18.

A3	5
B2	6
C1	7
<b>Total</b>	<b>18</b>

The reason the auction terminated more quickly was because the initial prices were closer to their relative equilibrium values and the increments were larger due to the larger discrete values in the utility table.

## 2.5 $\epsilon$ -Scaling and Price Wars

The initial prices set for the objects can significantly affect the length of the auction. Often, one or more particular objects will be of similar value to each other, yet far away from their relative equilibrium prices given the initial prices of other objects. Bidders may see similar value in the objects and increase the price by low increments, requiring a very large number of bidding rounds to occur before the prices reach the desired levels. The slow, incremental increase in an object price, referred to as a price war, can be reduced by a procedure known as  $\epsilon$ -scaling. Essentially, this requires early bidding rounds to be in much larger increments, while prices have large movements to make. The auction procedure then takes the prices obtained from the large  $\epsilon$  rounds as initial starting prices in subsequent auctions, where  $\epsilon$  is decreased, eventually to the desired level.

$\epsilon$  is lowered over the course of bidding cycles, typically by a factor of 10 (or a more appropriate constant) per auction. This is economically equivalent to only allowing large bid increments in the beginning of an auction, reducing the total number of bids received. As an example, imagine two objects are worth \$10,000 to three bidders and a third object is worth \$9,999. However, the initial prices of two of the items are set to \$0, while the third is set at \$10,000. All three will compete for the two items priced at \$0, raising the price as little as possible because their next most valuable item is at the same level. In fact, this competition will proceed until the level of the third item is reached, at which point it becomes more valuable for a bidder to bid for the third item, at which point the auction will terminate.

If the final price of the items will ultimately sell for \$10,000, minimum increments of \$1 will require 10,000 rounds to raise the two objects to the level of the third, whereas requiring the initial auction to increment at least \$1,000 per bid until the auction is over will require only 10 rounds. Of course, this causes the auction to only be within  $3 * \$1,000$  of optimality. But the process continues with a second auction that starts with no initial assignments, but the prices achieved by the previous auction, which requires far less bidding rounds. While the first auction will only be within  $n * \$1,000$  of optimality (where  $\epsilon = 1,000$ ), the next auction will be within  $n * (\text{minimum increment level})$  of optimality. As the auction proceeds,  $\epsilon$  is reduced to an acceptable level of guaranteed optimality. The reduction of epsilon by factors contributes to shrinking a portion of the theoretical bound to a logarithmic factor. In the example auction, if the second auction set  $\epsilon = \$100$ , a third with  $\epsilon = \$10$ , and a final with  $\epsilon = \$1$ , a total of 22 rounds would have been required, as opposed to the non  $\epsilon$ -scaling version requiring 10,003 rounds. This scenario is illustrated in Figure 8.

While not all savings will be as dramatic as in this case, there is always a theoretically bounded logarithmic decrease in the number of rounds required when  $\epsilon$ -scaling by factors is used. Attention has been focused on the optimal method of reducing epsilon, yet it appears that this method changes on a case by case basis, depending on the probability and frequency of price wars, as well as worst-case requirements. These scaling methods are lumped into a category called *adaptive scaling* [6].

It is important that complementary slackness be observed in the subsequent auctions; merely reducing epsilon in further rounds can violate conditions required.

Typically this is observed trivially by beginning each successive auction in the series with no assignments.

## Illustrative Example of Epsilon Scaling (e-scaling)

Figure 8

The auction on the left is run without e-scaling. The utility values of all objects in this example are very close, resulting in final prices that are likewise very close. However, the initial starting prices for objects 1, 2, and 3 are 0, 0, and 10,000, respectively. Therefore, the prices of objects 1 and 2 must be raised until it is more advantageous for object 3 to receive a bid at a price over 10,000. Because there are two objects already near their equilibrium value, the increments are small (typically  $2 * \epsilon$ ) for each bid round. This results in the auction needing 10,003 rounds.

The auction on the right is run with e-scaling. The initial epsilon value is 1000, giving a result that is accurate to  $n * \epsilon = 3000$ . The resulting prices are saved when the first auction terminates and the next auction is run beginning with those prices and a reduced epsilon (the reductions are by a factor of 10 in each auction), but starts with no assignments. Hence, the second auction has an epsilon equal to 100. The process is repeated until the desired epsilon is reached. Because the objects are so far from their relative equilibrium, using the large epsilon allows the prices to converge more quickly, with increments of 1000 versus the non e-scaling's increments of 1. The total number of rounds required is 22, versus the 10,003 required in the non e-scaling case. Using e-scaling, the auction is solved in polynomial time because the order depends on the log of the utility value and can grow according to bit representation.

The charts below are read in the following way. The initial conditions are specified first, denoting the starting prices of the objects. Each round of the auction then shows which bidder has been selected to bid, the object for which he will bid (the most profitable), the bid he is willing to make (old price + profit difference of best two objects + epsilon), and the revised, post-round assignments. The pre-round assignments are simply the previous round's assignments. The non e-scaling version consists of a single auction that terminates with its solution, while the e-scaling runs several auctions. Each successive e-scaling auction begins with the resultant prices of the previous auction as its new initial prices.

Utility Values				
	Objects	1	2	3
Bidders				
A		10,000	10,000	9,999
B		10,000	10,000	9,999
C		10,000	10,000	9,999

Figure 8 (page 2)

Auction without e-scaling

Auction with logarithmic e-scaling (reduction by 10x each auction)

AUCTION 1		Epsilon = 1		Owners of Each Object		
Round #	Bidder	Object Bid On	New Price	1	2	3
Initial	-	1	0	-	-	-
Conditions	-	2	0	-	-	-
	-	3	10,000	-	-	-
1	A	1	1	A	-	-
2	B	2	2	A	B	-
3	C	1	3	C	B	-
4	A	2	4	C	A	-
5	B	1	5	B	A	-
6	C	2	6	B	C	-
7	A	1	7	A	C	-
8	B	2	8	A	B	-
9	C	1	9	C	B	-
10	A	2	10	C	A	-
11	B	1	11	B	A	-
12	C	2	12	B	C	-
13	A	1	13	A	C	-
14	B	2	14	A	B	-
...	...	...	...	...	...	...
9999	C	1	9,999	C	B	-
10000	A	2	10,000	C	A	-
10001	B	1	10,001	B	A	-
10002	C	2	10,002	B	C	-
10003	A	3	10,001	B	C	A
				(final assignments)		
<b>10003 ROUNDS</b>						
Final Utility	Assignments		Utility Values			
	B1		10,000			
	A3		9,999			
	C2		10,000			
<b>Total</b>			<b>29,999</b>			
Optimality within 3*epsilon = 3						

AUCTION 1		Epsilon = 1000		Owners of Each Object		
Round #	Bidder	Object Bid On	New Price	1	2	3
Initial	-	1	0	-	-	-
Conditions	-	2	0	-	-	-
	-	3	10,000	-	-	-
1	A	1	1,000	A	-	-
2	B	2	2,000	A	B	-
3	C	1	3,000	C	B	-
4	A	2	4,000	C	A	-
5	B	1	5,000	B	A	-
6	C	2	6,000	B	C	-
7	A	1	7,000	A	C	-
8	B	2	8,000	A	B	-
9	C	1	9,000	C	B	-
10	A	2	10,000	C	A	-
11	B	1	11,000	B	A	-
12	C	2	11,000	B	C	-
13	A	3	11,000	B	C	A
<b>AUCTION 2 Epsilon = 100</b>						
Initial	-	1	11,000	-	-	-
Conditions	-	2	11,000	-	-	-
	-	3	11,000	-	-	-
14	A	1	11,100	A	-	-
15	B	2	11,100	A	B	-
16	C	3	11,100	A	B	C
<b>AUCTION 3 Epsilon = 10</b>						
Initial	-	1	11,100	-	-	-
Conditions	-	2	11,100	-	-	-
	-	3	11,100	-	-	-
17	A	1	11,110	A	-	-
18	B	2	11,110	A	B	-
19	C	3	11,110	A	B	C
<b>AUCTION 4 Epsilon = 1</b>						
Initial	-	1	11,110	-	-	-
Conditions	-	2	11,110	-	-	-
	-	3	11,110	-	-	-
20	A	1	11,111	A	-	-
21	B	2	11,111	A	B	-
22	C	3	11,111	A	B	C
				(final assignments)		
<b>22 ROUNDS</b>						
Final Utility	Assignments		Utility Values			
	A1		10,000			
	B2		10,000			
	C3		9,999			
<b>Total</b>			<b>29,999</b>			
Optimality within 3*epsilon = 3						

## 2.6 Similarity Classes and Price Wars

### 2.6.1 Auctions with Similar Objects

Often times a group of items will have identical utilities to all bidders and are said to comprise a *similarity class*, as illustrated in Figure 9.

#### Similar Objects

Figure 9

Utility Values

	Objects	1	2	3
Bidders				
A		10,000	10,000	9,999
B		10,000	10,000	9,999
C		10,000	10,000	9,999

Objects 1 and 2 are in a similarity class because

$$A_1 = A_2$$

$$B_1 = B_2$$

$$C_1 = C_2$$

*Similar objects are represented by identical columns.*

Formally, the utility of each object-bidder pair  $a_{ij}$  is identical for every object  $j$  in the similarity class:

$$a_{ij} = a_{ij'} \quad \forall i \in B(j) \text{ where } j \text{ and } j' \text{ are in a similarity class}$$

When competing for these objects, bidders will bid a very small increment (the minimum) for an item in a similarity class because the next optimal object bid would be for another object in the class at the same price. The next bid will then be the minimum increment for the next item in the class. Bidders will continue alternating among items in the similarity class until eventually a level is reached where a bidder chooses an item outside the class, referred to as the *contention threshold* or *minimum departure price*.



Until the contention threshold is reached, bidders will be engaged in a price war, raising bids as slowly as possible.

Price wars due to similarity classes can be circumvented by keeping track of similarity classes and contention thresholds. Every object  $j$  in a similarity class has its own contention threshold  $\rho_j$ . Unassigned objects carry a contention threshold of the initial price. Assigned objects have a contention threshold equal to  $\epsilon$  plus the price to which it can be raised under its current assignment before an item in a different class becomes equally valuable. If all contention thresholds of objects in a similarity class are higher than the prices of the objects, the prices can all be raised to the minimum contention threshold, avoiding the price war.

Therefore, the price  $p_j$  of every object  $j$  of a similarity class  $M(j)$  is the same, equal to the minimum contention threshold  $\rho$  over all objects in the class.

$$p_j = \min_{k \in M(j)} \rho_k$$

The previous example auction (demonstrating  $\epsilon$ -scaling) also contains similarity classes which can be exploited for efficiency. An example is presented in Figure 10 that demonstrates the use of similarity classes. Figure 11 then compares the similarity class auction with the standard auction, verifying a reduction in the number of rounds from 10,003 to 3, a clear and substantial improvement.

## Illustrative Example of Object Similarity Classes

Figure 10

		Utility Values		
Objects		1	2	3
<b>Bidders</b>				
<b>A</b>		10,000	10,000	9,999
<b>B</b>		10,000	10,000	9,999
<b>C</b>		10,000	10,000	9,999

Contention Thresholds (CTs) start at initial prices for unassigned objects. The CTs replace the former role of prices. First, an unassigned bidder is selected in (1), bidder A, who determines his best choice by finding his best profit (3), which now is equal to the utility minus the CT, rather than the actual price. The most profitable object is selected in (4) and the next most profitable in a different class is selected in (6). The new CT is now computed by calculating the price at which the best object would need to reach in order for the bidder to want a different class object. This is computed by taking the current price of the object, adding the difference in profits, and finally adding an epsilon. This becomes the CT of the best object (10), though the prices are the minimum CT of their class, which is still 0 in this case. The object is assigned to the bidder and the next round begins.

Step #	Auction Round #	1			Assignment (by Object)			Assignment (by Bidder)			
		1	2	3	1	2	3	A	B	C	
1)	<b>Bidder</b>	<b>A</b>									
	<b>Objects</b>	1	2	3							
	<b>Starting Prices</b>	0	0	10000	-	-	-	-	-	-	
2)	<b>Starting CTs</b>	0	0	10000							
	<b>Utilities</b>	10000	10000	9999							
3)	<b>Profit</b>	10000	10000	-1							
	<i>(Profit = Utility - CT)</i>										
4)	<b>Best Object, Profit</b>	1	10000								
5)	<b>Min Contention Threshold Object in Class, CT Value</b>	1	0								
	<b>Best Object Outside of Class, Profit</b>	3	-1								
6)	<b>Price of Best Object at which Object in (6) is chosen + epsilon</b>	<i>Price of Best Object + (4) - (6) + 1</i>		10002							
7)											
8)	<b>Object Bid On, Bid</b>	1	0		<b>Ending Assignment</b>			<b>Ending Assignment</b>			
	<i>(Bid = Min Contention Threshold within Class—Object 1 or 2's value of 0 + epsilon)</i>										
9)	<b>Ending Prices</b>	0	0	10000	<b>A</b>	-	-	<b>1</b>	-	-	
	<i>(Price - Min CT of Class, = Object 2's 0)</i>										
10)	<b>Ending CTs</b>	10002	0	10000							
		<i>(7) - Price of this object at which object 3 (outside of class) is chosen</i>		<i>Still unassigned</i>							

Unassigned bidder B is selected (2) and his profits options are examined (3). His best object is clearly object 2, and the new CT of the object is calculated to be the price of 2 plus the profit difference of object 2 and the best object outside of the class, plus epsilon, which is equal to  $0 + (10000 - -1) + 1 = 10002$  (7). This becomes the new CT for object 2 (10), and the prices of the class are now recomputed to be the minimum CT of the class, which is now 10002 (9). Object 2 is assigned to B (8) and the next round continues.

Step #	Auction Round #	2			Assignment (by Object)			Assignment (by Bidder)		
		1	2	3	1	2	3	A	B	C
1)	<b>Bidder</b>	<b>B</b>								
	<b>Objects</b>	1	2	3						
	<b>Starting Prices</b>	0	0	10000	A	-	-	1	-	-
2)	<b>Starting CTs</b>	10002	0	10000						
	<b>Utilities</b>	10000	10000	9999						
3)	<b>Profit</b>	-2	10000	-1						
	<i>(Profit = Utility - CT)</i>									
4)	<b>Best Object, Profit</b>	2	10000							
5)	<b>Min Contention Threshold Object in Class, CT Value</b>	2	0							
	<b>Best Object Outside of Class, Profit</b>	3	-1							
6)	<b>Price of Best Object at which Object in (6) is chosen + epsilon</b>	<i>Price of Best Object + (4) - (6) + 1</i>		10002						
7)										
8)	<b>Object Bid On, Bid</b>	2	10002		<b>Ending Assignment</b>			<b>Ending Assignment</b>		
	<i>(Bid = Min Contention Threshold within Class—Object 2's value of 0 + epsilon)</i>									
9)	<b>Ending Prices</b>	10002	10002	10000	<b>A</b>	<b>B</b>	-	<b>1</b>	<b>2</b>	-
	<i>(Price - Min CT of Class, = Object 1 or 2's 10002)</i>									
10)	<b>Ending CTs</b>	10002	10002	10000						
		<i>(7) - Price of this object at which object 3 (outside of class) is chosen</i>		<i>Still unassigned</i>						

Round 3 takes the only remaining unassigned bidder C (1) and again computes the profits (2), choosing object 3 as the best choice. The CT is now computed by taking the price of object 3 plus the profit difference plus epsilon (7). The assignment is made in (8) and the price of the class of C is now recomputed. Trivially, the price of C is always equal to its CT. Since all objects are assigned, the auction terminates.

Step #	Auction Round #	3			Assignment (by Object)			Assignment (by Bidder)		
1)	Bidder	C			1	2	3	A	B	C
	Objects	1	2	3	A	B	-	1	2	-
	Starting Prices	10002	10002	10000						
2)	Starting CTs	10002	10002	10000						
	Utilities	10000	10000	9999						
3)	Profit	-2	-2	-1						
	<i>(Profit = Utility - CT)</i>									
4)	Best Object.Profit	3	-1							
5)	Min Contention Threshold Object in Class, CT Value	3	10000							
6)	Best Object Outside of Class, Profit	1	-2							
7)	Price of Best Object at which Object in (6) is chosen + epsilon	<i>Price of Best Object + (4) - (6) + 1</i>		10002						
8)	Object Bid On, Bid	3	10002		Ending Assignment			Ending Assignment		
	<i>(Bid = Min Contention Threshold within Class--10002)</i>									
9)	Ending Prices	10002	10002	10002	A	B	C	1	2	3
	<i>(Price - Min CT of Class, = Object 3 CT)</i>									
10)	Ending CTs	10002	10002	10002						
		<i>Price of this object at which object 3 (outside of class) is chosen</i>		<i>(7) - Price of this object at which object outside of</i>						

## Comparative Example of Object Similarity Classes

Figure 11

The charts below are read in the following way. The initial conditions are specified first, denoting the starting prices of the objects. Each round of the auction then shows which bidder has been selected to bid, the object for which he will bid (the most profitable), the bid he is willing to make (old price + profit difference of best two objects + epsilon), and the revised, post-round assignments. The pre-round assignments are simply the previous round's assignments. The non-similarity class auction (at left) engages in a price war between objects 1 and 2, slowly incrementing each object until eventually object 3 is selected. The similarity class auction (at right), on the other hand, allows objects A and B to instantly increase their price to a level where an object outside the class would be selected. This avoids the price war and terminates the auction in 3 rounds. Detailed rounds are given for the similarity class auction in the prior figure.

**Utility Values**

	Objects	1	2	3
<b>Bidders</b>				
<b>A</b>		10,000	10,000	9,999
<b>B</b>		10,000	10,000	9,999
<b>C</b>		10,000	10,000	9,999

**Auction Without Similarity Classes**

**Auction With Similarity Classes**  
(objects 1 and 2 in same class)

<b>AUCTION 1</b>				<b>Owners of Each Object</b>		
Round #	Bidder	Object	New Price	1	2	3
<b>Initial</b>	-	1	0	-	-	-
<b>Conditions</b>	-	2	0	-	-	-
	-	3	10,000	-	-	-
1	A	1	1	A	-	-
2	B	2	2	A	B	-
3	C	1	3	C	B	-
4	A	2	4	C	A	-
5	B	1	5	B	A	-
6	C	2	6	B	C	-
7	A	1	7	A	C	-
8	B	2	8	A	B	-
9	C	1	9	C	B	-
10	A	2	10	C	A	-
11	B	1	11	B	A	-
12	C	2	12	B	C	-
13	A	1	13	A	C	-
14	B	2	14	A	B	-
...	...	...	...	...	...	...
9999	C	1	9,999	C	B	-
10000	A	2	10,000	C	A	-
10001	B	1	10,001	B	A	-
10002	C	2	10,002	B	C	-
10003	A	3	10,001	B	C	A
(final assignments)						
<b>10003 ROUNDS</b>						
<b>Final Utility</b>	<b>Assignments Utility Values</b>					
		<b>B1</b>	10,000			
		<b>A3</b>	9,999			
		<b>C2</b>	10,000			
<b>Total</b>	<b>29,999</b>					
Optimality within 3*epsilon = 3						

<b>AUCTION 1</b>				<b>Owners of Each Object</b>		
Round #	Bidder	Object	CT	1	2	3
<b>Initial</b>	-	1	0	-	-	-
<b>Conditions</b>	-	2	0	-	-	-
	-	3	10,000	-	-	-
1	A	1	10,002	A	-	-
2	B	2	10,002	A	B	-
3	C	1	10,002	A	B	C
(final assignments)						
<b>3 ROUNDS</b>						
<b>Final Utility</b>	<b>Assignments Utility Values</b>					
		<b>A1</b>	10,000			
		<b>B2</b>	10,000			
		<b>C3</b>	9,999			
<b>Total</b>	<b>29,999</b>					
Optimality within 3*epsilon = 3						

### 2.6.2 Auctions with Similar Bidders

An auction with similar bidders can also be handled to avoid price wars and expedite the process [4] [6]. The intuition here is that if one bidder is not assigned, then all  $k$  bidders of that similarity class bid a common amount set to the  $(k+1)^{st}$  bid for the first  $k$  most profitable items. Essentially, bidders are automatically raising their bids to the minimum contention threshold of the class.

#### Similar Bidders

Figure 12

Utility Values

	Objects	1	2	3
Bidders				
A		10,000	10,000	9,999
B		10,000	10,000	9,999
C		10,000	10,000	9,999
D		9,999	9,999	9,999

Persons A, B, and C are in a similarity class because

$$A_1 = B_1 = C_1$$

$$A_2 = B_2 = C_2$$

$$A_3 = B_3 = C_3$$

*Similar Persons are represented by identical rows.*

### 2.7 Reverse Auctions

Reverse auctions switch the bidding perspective such that the objects now lower their prices, trying to extract as high a price as possible from the bidders [4] [6] [11]. To obtain a bidder, the unassigned object must drop its price low enough to draw a bidder away from some other assignment or to attract an initial bidder. Note here that once a bidder has been assigned, he can at worst be reassigned to another, and will never again be unassigned. Hence, the idea of irreversible progress is made when a new person becomes assigned. Reverse auctions are equivalent to forward auctions and are subject to

the same issues of complementary slackness,  $\epsilon$ -CS, price wars, and similarity classes, which can all be handled in analogous ways as its forward counterpart.

Formally, the bid in a reverse auction for an unassigned person is computed in the following way, analogous to the forward case. A key point is that price  $p$  is exchanged with profit  $\pi$  as the perspective has reversed. An unassigned object  $j$  looks for its best person  $i$  and the corresponding value  $\beta$ .

$$\text{Best person } i: \quad i_j = \arg \max_{i \in B(j)} \{ a_{ij} - \pi_i \}$$

$$\text{Best value:} \quad \beta_j = \max_{i \in B(j)} \{ a_{ij} - \pi_i \}$$

Additionally, the value of the second best person  $k$  is computed.

$$\text{Second best value:} \quad \omega_j = \max_{k \in B(j), k \neq i} \{ a_{kj} - \pi_k \}$$

The bid by the object is to the best person  $i$  and increases  $\pi_i$  by  $(\beta_j - \omega_j + \epsilon)$ . The price is determined by the relation

$$p_j + \pi_i = a_{ij} \quad \forall (i,j) \in S$$

for all assigned pairs  $ij$ . If a different object had been assigned to the person earlier, it becomes unassigned and reenters the bidding process. Additionally, complementary slackness holds in the following way:

$$\text{Complementary Slackness:} \quad a_{ij} - \pi_i \geq \max_{k \in B(j)} \{ a_{kj} - \pi_k \} \quad \forall (i,j) \in S$$

## 2.8 Combined Forward-Reverse Auctions

A notably efficient method in practice occurs through the implementation of forward-reverse combination [6] [7] [8]. A typical forward-reverse auction begins with a forward auction of several iterations, at least until one new object becomes assigned.

Both profit and price are tracked, under the relation

$$p_j + \pi_i = a_{ij} \quad \forall (i,j) \in S$$

Next a reverse auction iterates until an additional bidder becomes assigned. The process then cycles until no bidders are left. At each stage of the auction, the fact that an additional bidder becomes assigned represents irreversible progress. Therefore, the auction will be guaranteed to terminate after at most  $N/2$  cycles, with each cycle consisting of a series of forward iterations followed by a series of reverse iterations.

The reason for the efficiency stems from receiving the benefits of both forward and reverse auctions simultaneously. While auctions that initially have all items start at the same price typically exhibit forward-only behavior, auctions beginning with differing prices and values can greatly benefit from the ability to approach the problem from both sides simultaneously.

## 2.9 Asymmetric Assignment

Asymmetric assignment problems are those that have an unequal number of bidders  $i$  and objects  $j$  [4] [6]. Two major methods have emerged in solving such systems. The first involves a conversion to the symmetric case by introducing a number

of artificial objects with very high price or bidders with very low value for a match (or any other arbitrary value that is identical to every bidder of the artificial object or to every object of the artificial bidder). This results in the least attractive bidders or objects being paired with the artificial counterparts, which are separated from the results in the end.

While inheriting the positive qualities of symmetric solution techniques, it can dramatically increase the size of the problem. The benefit from this technique is that it accommodates both sided asymmetric cases where  $i \geq j$  and  $i \leq j$ .

To avoid price wars, it is best to set the value of the artificial objects to a number less than the lowest value of any real object. This way, the artificial objects will be selected last, rather than competitively bid for until another object becomes more lucrative. This issue can also be skirted by using a similarity class for all artificial objects and bidders.

A second technique requires that  $i \leq j$  and pursues a forward auction until all objects have been assigned. This is followed by a reverse auction that reduces all unassigned object prices below a threshold price of the lowest assigned object (trivially observed in the symmetric case). Once this last constraint is met, the solution is deemed optimal. The conditions required for optimality are as follows:

$$\epsilon\text{-Complementary Slackness: } p_j + \pi_i \geq a_{ij} - \epsilon \quad \forall (i,j) \in A$$

$$\text{Price-Profit Parity: } p_j + \pi_i = a_{ij} \quad \forall (i,j) \in S$$

$$\text{Assigned objects above threshold: } p_j \leq \min_{k \text{ assigned under } S} p_k \quad \forall j \text{ unassigned under } S$$



### 2.9.1 A Modified Asymmetric Forward-Reverse Auction Algorithm

The following algorithm was developed in [8], to which the reader is referred for additional reference. Two types of iterations comprise the algorithm. The forward iteration acts only when there are unassigned persons, and the reverse iteration occurs only when there are unassigned objects with a price above a threshold  $\lambda$ . Three conditions to be met for optimality are described in the previous section. Each iteration begins with a set of assignments that satisfies  $\epsilon$ -CS, Price-Profit Parity, and an additional constraint that requires every assigned object to have a price of at least  $\lambda$ . Initially, the algorithm can begin with no assignments,  $\lambda$  and  $p_j$  to be arbitrary, and  $\pi_i$  to be at least  $(a_{ij} p_j - \epsilon)$ .

In the forward iteration, an unassigned person  $i$  finds his best object  $j$  as usual. However, the bid price and profit now become

$$p_j = \max\{ \lambda, a_{ij} - \omega_i + \epsilon \}$$

$$\pi_i = \omega_i - \epsilon$$

If  $\lambda \leq a_{ij} - \omega_i + \epsilon$ , the new pair  $(i,j)$  is added to the assignment set and the old pair involving object  $j$ , if any, is removed. Otherwise, no new assignments are made and only the profits and prices are updated.

As noted above, the reverse iteration takes an unassigned object  $j$  with a  $p_j > \lambda$  and finds its best person  $i$ . The best price  $\beta_j$  and second best price  $\omega_j$  are found, as usual, but the algorithm strays at this point. Two cases dictate the next action. In the first case, the

Case 1:  $\beta_j \geq \lambda + \epsilon$

$$p_j = \max\{ \lambda, \omega_i - \epsilon \}$$

$$\pi_i = a_{ij} - \max\{\lambda, \omega_i - \varepsilon\}$$

The new pair  $(i,j)$  is added to the set and any old pair involving person  $i$  is removed.

Case 2:  $\beta_j < \lambda + \varepsilon$

$$p_j = \beta_j - \varepsilon$$

No assignment occurs here, but the price is dropped. If there are now more than  $n - m$  objects with a price less than  $\lambda$ , then  $\lambda$  is set to the next value below  $\lambda$  such that the number of objects below the new  $\lambda$  is equal to  $n - m$  (and possibly greater than if there is more than one object at that exact level).

$\lambda$  can only be reduced in the reverse algorithm and is unaffected in the forward iteration. The reverse algorithm may also end up not assigning an object. However, the price of it is reduced and the object will not be able to bid until  $\lambda$  is reduced.

## 2.9.2 Relaxing the Constraint of Bidder Assignment

Another interesting variation occurs when it is optimal to have one or more bidders not assigned to any object [6]. This can be set up by creating a number of artificial objects that have a utility value set at a threshold level for each bidder. Additionally, the values must be standardized such that it is more beneficial to pair a group among the threshold value (easiest when set to zero to represent positive as good and negative as not preferred) while values below the threshold level will choose artificial pairings. Figure 13 demonstrates the use of artificial objects in an auction of three bidders and three real objects. Figure 14 presents the formulation of the assignment problem when objects and bidders are allowed to remain unassigned.

**Allowing Bidders to Remain Unassigned**  
Utility Values

	Objects	1	2	3	4	5	6
Bidders							
1		10,000	10,000	-242	0	0	0
2		10,000	10,000	-564	0	0	0
3		10,000	10,000	-738	0	0	0

**Figure 13**

Objects 4, 5, and 6 are artificial objects with a threshold price of 0. Therefore, object 3 is displaced by "no object" (object 4). However, objects 1 and 2 are still assigned because of their superior utility to the artificial variables.

$$\text{Maximize } \sum \sum a(i,j)f(i,j)$$

$a(i,j)$  = Utility of pairing bidder  $i$  and object  $j$

$f(i,j)$  = 1 if  $i$  and  $j$  paired together, 0 otherwise (an indicator function)

$$\sum f(i,j) = 0 \text{ or } 1 \quad \forall i=1, \dots, m \text{ (bidders)} \quad \begin{array}{l} \text{Outflow} = 0 \text{ or } 1 \\ \text{Each bidder possibly assigned once} \end{array}$$

$$\sum_{i \in A(i)} \sum_{j \in B(j)} f(i,j) = 0 \text{ or } 1 \quad \forall j=1, \dots, n \text{ (items)} \quad \begin{array}{l} \text{Inflow} = 0 \text{ or } 1 \\ \text{Each item possibly assigned once} \end{array}$$

**Figure 14**

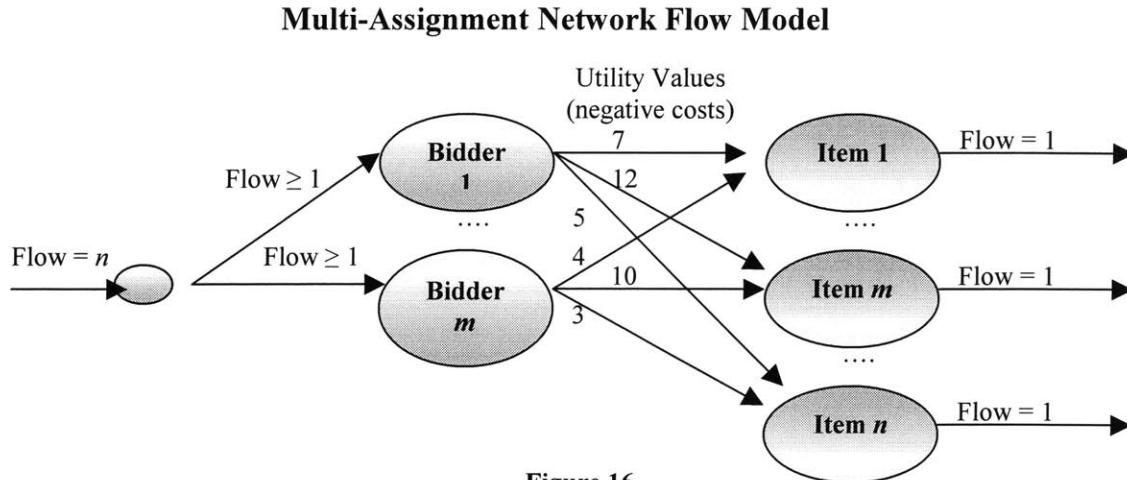
**Formulation of Assignment Problem without Required Assignment of Each Object**

By allowing bidders to remain unassigned, inflow and outflow to each bidder and item can be either 0 or 1. Auctions can solve this problem type by introducing artificial variables to bring the constraints back in line with Figure 2, where the sum must equal 1.  $n$  items becomes  $n + m$  items,

## 2.10 Multi-Assignment Problems

Multi-assignment problems are a variation of standard auctions in which each person can be assigned more than one object, yet each bidder is assigned at least one

object [4]. Additionally, every object is assigned to some bidder. The network model can be visualized in Figure 16.



**Figure 16**

Multi-assignment problems are represented in a network flow model where each item must have a flow of 1, while each bidder is constrained have a flow of at least one. Flow from bidder  $i$  to item  $j$   $f(i,j)$  is constrained such that  $0 \leq f(i,j) \leq 1$ .

Formally, the problem is formulated as in Figure 17 below.

$$\text{Maximize } \sum \sum a(i,j) f(i,j)$$

$a(i,j)$  = Utility of pairing bidder  $i$  and object  $j$

$f(i,j)$  = 1 if  $i$  and  $j$  paired together, 0 otherwise (an indicator function)

$$\sum_{j \in A(i)} f(i,j) \geq 1 \quad \forall i=1, \dots, m \text{ (bidders)} \quad \begin{array}{l} \text{Outflow} \geq 1 \\ \text{Each bidder assigned to at least one object} \end{array}$$

$$\sum_{i \in B(j)} f(i,j) = 1 \quad \forall j=1, \dots, n \text{ (items)} \quad \begin{array}{l} \text{Inflow} = 1 \\ \text{Each item assigned once} \end{array}$$

**Figure 17**

Formal maximization definition of a multi-assignment problem.

By transposing the problem so that bidders become objects, the problem of assigning at least one bidder to each object can be solved, where each bidder is assigned once and each object has at least one bidder assigned to it.

The method for solving the multi-assignment problems can be broken into two parts. Initially, the standard forward auction proceeds to until every bidder has one object. The second part consists of a modified reverse auction to assign the remaining objects. The conditions that need to be met for the multi-assignment to be optimal are as follows:

$$\varepsilon\text{-Complementary Slackness: } p_j + \pi_i \geq a_{ij} - \varepsilon \quad \forall (i,j) \in A$$

$$\text{Price-Profit Parity: } p_j + \pi_i = a_{ij} \quad \forall (i,j) \in S$$

$$\text{Assigned objects above threshold: } \pi_i = \max_{k \text{ assigned under } S} \pi_k \quad \forall j \text{ unassigned under } S$$

A constant  $\lambda$  will be used to determine a profit threshold for new persons.

Initially,

$$\lambda = \max_{i=1, \dots, m} \pi_i$$

At this point, a reverse auction is run by taking an unassigned object  $j$  and reducing its price low enough to attract a bidder  $i$ . However, the old object only displaced from bidder  $i$  if the profit of the assignment is equal to  $\lambda$ . The stages are outlined below:

$$\text{Best person } i: \quad i_j = \arg \max \{ a_{ij} - \pi_i \}$$

$$i \in B(j)$$

Best value: 
$$\beta_j = \max_{i \in B(j)} \{ a_{ij} - \pi_i \}$$

Second best value: 
$$\omega_j = \max_{k \in B(j), k \neq i} \{ a_{kj} - \pi_k \}$$

$$\delta = \min[\lambda - \pi_j, (\beta_j - \omega_j + \epsilon)]$$

The pair  $(i, j)$  becomes part of the assigned set  $S$ . The price of  $j$  and profit of  $i$  are set as follows:

$$p_j = \beta_j - \lambda$$

$$\pi_i = \pi_i + \lambda$$

If  $\delta > 0$ , the former object assigned to  $i$  becomes unassigned. Otherwise, the two objects now share the bidder. The auction ends when all objects are assigned.

### 2.10.2 Linear Combinatorial Optimization

The above multi-assignment auction is an instance of linear combinatorial optimization. Linear combinatorial optimization occurs when the values of objects maintain their pairing values regardless of whether groups of objects are assigned to bidders or just the object by itself. The pairing values of groups in this case add linearly. If the requirement of each bidder needing to receive at least one object is relaxed, a new problem is formed. While this problem can be assigned by introducing a set of artificial objects equal to the number of real objects less one with values equal to the threshold level of indifference, the problem can be solved more easily with other methods. Particularly, the problem can intuitively be thought of as the one sided view from each object about which person would be the best fit, regardless of the other objects. From

this view, the problem deteriorates into a mere search for the maximum pairing of each object over all its potential bidders will yield the optimal assignment in  $n*m$  operations total.

### **2.11 Nonlinear Combinatorial Optimization Using Auction**

Often times a bidder will value a group of products more than the sum of the individual values, such as purchasing a computer and software, where neither has value without the other. Conversely, a bidder may value his first car tremendously, but the second is unnecessary. In this case, the combination is worth far less than the sum of values of the individual cars. These nonlinear behaviors complicate the process of finding a solution because auction algorithms are only capable of solving linear models. However, auction can be used to reduce part of the required time by separating the problem into linear and non-linear sections, and applying the auction to the linear segments. Examples of this technique are examined in Chapters 3 and 4.

### 3 Distributor Problem

A scenario is constructed in which a distributor has received a number of objects. The distributor has a number of customers, as well, and needs to distribute the objects optimally among the customers. This scenario commonly occurs when a stockbroker receives a limited number of shares of companies undergoing an initial public offering (IPO). All the shares are priced identically, so it does not matter monetarily to which clients he sells the shares. However, unique customers will value receiving shares differently, perhaps leaving the broker if they do not receive certain amounts or shifting the amount of money they have under management to match their happiness. Assigning all shares to one client is probably not optimal because it alienates the broker's other customers. However, because customers are not valued equally, dividing the shares among them evenly would not reflect better treatment to better customers. While each customer would like to receive as much as possible (given a customer's investment constraints), an optimal assignment exists that maximizes the overall happiness of the broker's customers. In this example, stock shares would be thought of as objects and customers would be bidders. This problem addresses the following issues:

#### **Reducing Complexity:**

- *Exploiting the fact many objects are identical (members of a class) both in reducing auction time and number of required auctions.*
- *Discretizing objects into minimum-sized multiple-object "buckets"*
- *Limiting each bidder assignment to one type of object*

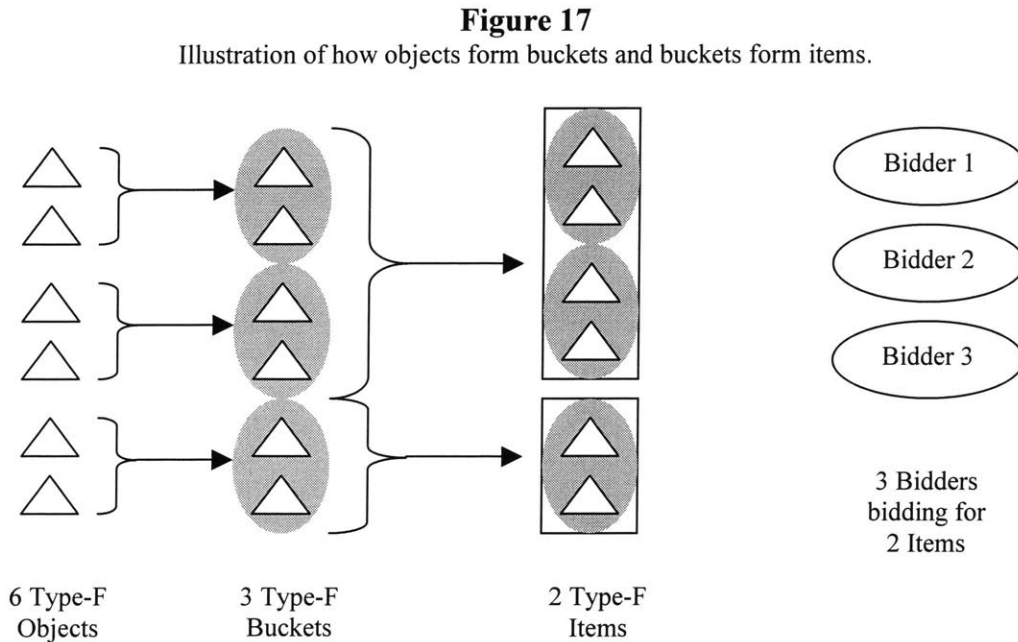
#### **Allowing Variation:**

- *Addressing nonlinearity in object assignments*
- *Setting up the possibility of not pairing (i.e. not using) a bidder*
- *Setting up the possibility of not pairing (i.e. not using) an object*



## Terminology

For clarity and reference, the terminology used throughout this example is described here. Auctions are composed of bidders that bid for items. Items are buckets or groups of buckets, which are a minimum allowable amount of a type of object to be assigned to a single bidder. In this example, type-F buckets are equal to two objects.



### 3.1 Problem Description

In this problem, the distributor receives six types of objects {A, B, C, D, E, F}. The number received of each object varies. The distributor has 10 unique bidder types to whom he can distribute objects {Q, R, S, T, U, V, W, X, Y, Z}. The number and types of objects and bidders are displayed in the Figure 18.

**Figure 18**

Types and numbers of bidders involved in the distributor problem.

Object Type		# Objects	Bidder Type		# Bidders
	A	20		Q	1
	B	16		R	1
	C	40		S	1
	D	7		T	1
	E	5		U	1
	F	6		V	1
<b>Totals</b>	<b>6 Types</b>	<b>94</b>		W	1
				X	1
				Y	1
				Z	1
			<b>Totals</b>	<b>10 Types</b>	<b>10 Bidders</b>

The distributor is capable of delivering either one type of object or no object to each bidder. For example, a Q-type bidder can receive five A-type objects, seven D-type objects, 16 B-type objects, or any available number of a single type of object. A Q-type bidder can also receive nothing, or even receive the same type of object as the R-type bidder provided there are enough objects to satisfy both bidders. The distributor need not distribute all the objects.

The utilities of each bidder sum linearly to yield the overall utility and are independent of each other (i.e. the scores of bidder Q and bidder R add linearly to contribute to the overall score). However, the utilities of each bidder vary nonlinearly with the number of objects assigned to it, where to bidder Q, five type-A objects are not necessarily worth one-half the score of ten type-A objects. A sample section of the overall utility matrix is examined in Figure 19. The figure shows each different possible pairing and the corresponding value of any bidder assigned to a type-A item. A similar matrix exists for each different type of object.

**Utility Values for Delivering Type-A Objects**

<b># Objects</b>		<b>5</b>	<b>10</b>	<b>15</b>	<b>20</b>
<b>Bidders</b>	<b>Q</b>	2	10	16	20
	<b>R</b>	-3	5	5	-10
	<b>S</b>	6	5	4	3
	<b>T</b>	5	6	7	8
	<b>U</b>	10	5	0	-10
	<b>V</b>	7	-40	-41	-42
	<b>W</b>	4	5	6	7
	<b>X</b>	5	10	11	12
	<b>Y</b>	7	6	5	4
	<b>Z</b>	-1	-2	-3	20

**Figure 19**

Each value in the chart is the individual utility of pairing the bidder with the specified number of type-A objects.

At first glance, the problem looks potentially approachable by auction as it is an assignment problem. Conversely, because the pairs behave nonlinearly, the question arises as to whether auction can really be utilized in finding the solution. The optimal assignment can, of course, be computed using complete enumeration of the possibilities. The computation required, though, grows too large. Single auctions are capable of solving a large number these possibilities quickly, and enumerating a number of different auctions allows a very large space to be solved quickly. This occurs most obviously when several types of objects, each consisting of a number of objects, must be permuted across many bidders. A single auction is required for each unique a object combination, whereas the number of permutations for that object combination that can be assigned distinctly among bidders grows at a factorial rate. While both the single auction and the

complete set of permutations will solve the problem, the auction produces the result in polynomial time, opposed to the NP-complete permutation complexity.

The important idea is that applying auction to certain parts of the solution reduce the solution time. The constraints, complexities, and reduced models are discussed in the following section to demonstrate how auction can be implemented to extract the optimal assignment.

### **3.2 Constraints**

The major constraints are as follows

- Each bidder receives at most one type of object, though the number objects of that type may vary.
- An object can be assigned to at most one bidder.
- Each bidder need not be assigned.
- Each object need not be assigned.

### **3.3 Complexity**

The major complexity in this model lies in the nonlinear utilities relating to the number of objects chosen from a type. Therefore, each different combination of pairings must be evaluated, which grows to be an enormous number. However, because each bidder is limited to receiving only one type of object, the problem size diminishes greatly. The problem increases mildly in size because objects and bidders have the freedom to not be assigned. Complexity is handled through object classification, bucketing objects into minimum-sized items, constraining each bidder to one type of object, and

### **3.4 Reducing and Formulating the Problem**

#### **3.4.1 Bucketing**

To reduce the size of the problem, a major assumption must be made in how objects will be delivered, specifically limiting the number of ways the objects can be assigned to bidders. Assuming the objects in a type are identical and interchangeable, with twenty of the A-type objects, the number of different possible combinations of A-type objects for this example would be 635. Not assuming they are identical and interchangeable, the number of possible combinations are on the order of  $(c+1)^n$ , where  $c$  is the number of bidders and  $n$  is the number of objects. If all objects in the problem were considered unique, the number of potential combinations would be equal to  $11^{94}$ , or  $7.8E+97$ , a number too large to investigate in real-time. Intuitively, this can be thought of as each object going to either one specific bidder or no bidder, a total of  $c+1$  choices for each of the  $n$  objects. By having identical object types, the number of unique permutations decreases because many of the permutations are now viewed as indistinguishable. However, with each unique configuration of objects permuted to cover all possibilities among the ten bidders, the number of permutations grows to roughly  $\sim[6.5E+12] * [15!/(15-10)!]$ , which is on the order of  $1.2E+23$ , a significant drop from  $7.8E+97$  yet still computationally demanding. The first term is the number of configurations and the second a rough estimate of the permutations among the bidders for each configuration. Here, buckets will be introduced to reduce the size of the problem.

Buckets are just groups of objects that cannot be separated further. They are analogous to cases of products that must be purchased as an entire case and cannot be sold separately. Buckets can also be grouped together, enabling bidders to purchase several buckets at once. In this example, the A-type objects will be divided into buckets of size five. Where 20 objects existed, four now represent. Buckets can exist in a variety

of sways, such as in sizes one, two, four and eight or in sizes one, five, and 15 (this example uses a single size for each object type). The power lies in constructing buckets in sizes that both reduce the number of possibilities and yet still capture likely optimal assignments. Essentially, this approximates the maximum utility by solving a courser problem, allowing a reduction of size of the problem.

If the solution is found to include the A-type objects, the model can be rerun with a finer level of buckets while excluding objects that were not assigned in the former assignment. While optimality is *not* guaranteed in this case, far better time constraints can be met.

### **3.4.2 Determining Computational Requirement**

The proposed model seeks to use auction to find the optimal assignment. Because objects within their own class can be grouped and behave nonlinearly, however, the problem becomes combinatorial. It is impossible for an auction to compare a bidder receiving all four buckets versus four bidders receiving one bucket each because each auction relies on linearity (where  $aF(x) = F(ax)$ ) to reach its solution. Nonlinearity allows local maxima to exist that potentially lead to globally suboptimal results. Therefore, a structured method of using auction involves creating a series of auctions. Each auction is composed of a different enumeration of items and will yield an overall optimal score for that configuration, in turn allowing comparison among different configurations. Correct use of this method requires running every item configuration possibility as an auction to capture each nonlinear utility.

The number of total configurations is arrived at by first determining the number of unique combinations of buckets within a type of object. Each combination consists of items, where each item is composed of inseparable buckets. For example, each type-A bucket represents five type-A objects, and each bidder can be assigned between zero and four buckets (assuming not more than four buckets are assigned in total). One specific combination consists of four items, where each item is a separate bucket. Another group consists of an item of three buckets (as one inseparable item) and an item of a single bucket. The total number of combinations that can be constructed for each type is illustrated in Figure 20.

# of Buckets	# Unique Combinations	Specific Configurations of Items						
1	1	1						
2	2	2	1,1					
3	3	3	2,1	1,1,1				
4	5	4	3,1	2,2	2,1,1	1,1,1,1		
5	7	5	4,1	3,2	3,1,1	2,2,1	2,1,1,1	1,1,1,1,1
...	...							
20	635	20	19,1	18,2	18,1,1	...	...	...

**Figure 20**

Unique combinations are illustrated, demonstrating the ways in which a number of identical buckets can be divided uniquely. For example, 3 buckets can be uniquely configured in 3 distinct ways. The first configuration consists of an item containing all 3 buckets. The second configuration is composed of an item of 2 buckets and an item of 1 bucket. The third configuration occurs as 3 separate items, each a single bucket.

From an auction perspective, each item of a specific group can be viewed as a single item available in an auction. For example, in the four-bucket case, five different auctions could be constructed. The first auction would have only a single item of the type, which is composed of all four of the buckets. The second auction would consist of two items, specifically an item containing three buckets and an item containing one

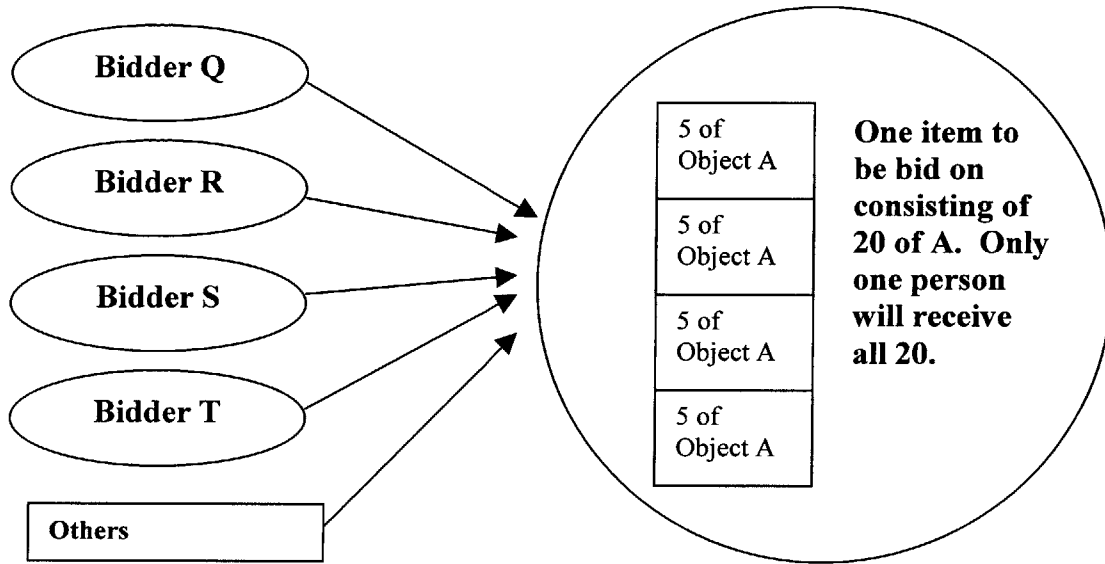
bucket. When bidders bid for objects in the first auction, they can only bid for the four-bucket item. Likewise, in the second auction, bidders can bid for either the three-bucket item or the one-bucket item, with these being the only two items of type-A for which any bidder can bid. These two auctions are shown in Figure 21.



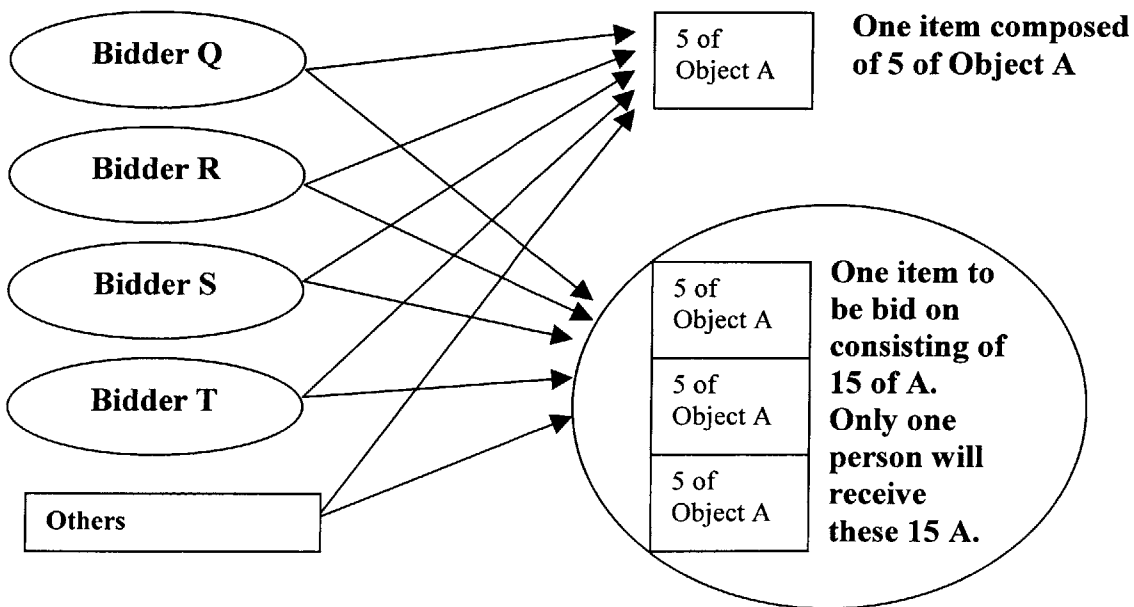
**Figure 21**

Two different auction configurations are shown. The first consists of the bidders bidding on an item configuration of one item. The second consists of the bidders competing for a different distribution of the same objects, but this time as two separate items.

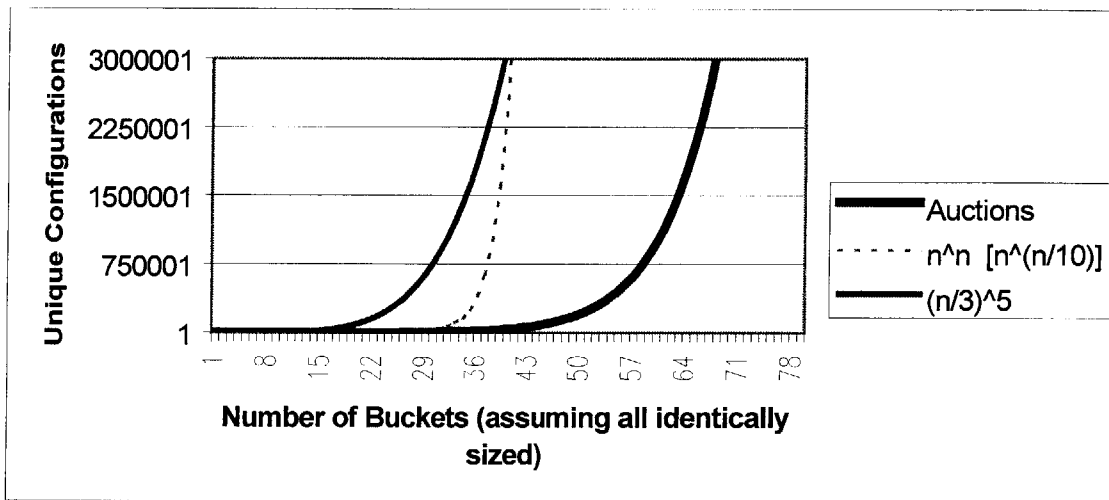
**Auction 1—Bidders are bidding for one item, a group of 20 of object A.**



**Auction 2—Bidders are bidding for a different configuration of items. The two items available are a 5-Object-A and a 15-Object-A. A bidder can be assigned one or the other, but not both.**



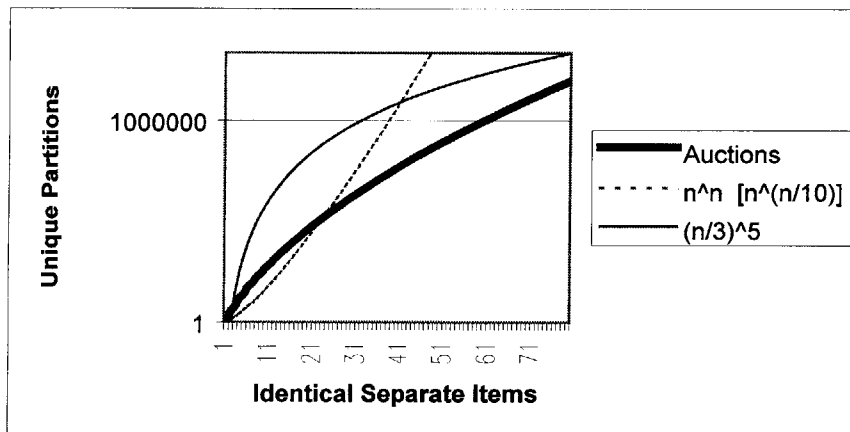
The total number of different auctions that can be set up for each class is exactly equal to the total number of configurations. The number of configurations was at first thought to be NP-complete, but upon closer investigations reveals itself to behave, at least at a level of up to 79 items, better than exponentially. The following charts demonstrate the growth of the number of configurations that can be uniquely formed (assuming items within the configuration are identical and indistinguishable). Figure 22 shows the number of unique configurations versus polynomial and NP-complete functions.



**Figure 22**

A comparative graph of the order of growth of the number of groups formed by  $n$  identical items.

Figure 23 illustrates that the growth of unique configurations behaves better than  $O(n^n)$ , easily discernable by examining the log behavior. However, it also appears to grow faster than the polynomial time functions. It appears to asymptotically approach an exponential growth rate  $O(c^n)$  and can be shown to be upper bounded by  $O(2^n)$ , empirically demonstrated in Figure 24.



**Figure 23**

A comparative graph of the order of growth of the number of groups formed by  $n$  identical items, plotted on a logarithmic y-axis.

**Figure 24**

Examining the empirical growth of the number of unique auction configurations (Auctions) produced by a given number of buckets is compared to an exponential function  $2^n$ , where  $n$  corresponds to buckets. The decreasing derivative value of auctions with respect to buckets suggests that the function is upper bounded by an exponential function that maintains a constant derivative as  $n$  grows.

Buckets (n)	$d(2^n)/d(n)$	$d(\text{Auctions})/d(\text{Buckets})$		Buckets (n)	$d(2^n)/d(n)$	$d(\text{Auctions})/d(\text{Buckets})$
2	2	2.000		24	2	1.266
3	2	1.500		25	2	1.238
4	2	1.667		26	2	1.243
5	2	1.400		27	2	1.236
6	2	1.571		28	2	1.245
7	2	1.364		29	2	1.227
8	2	1.467		30	2	1.234
9	2	1.318		31	2	1.208
10	2	1.448		32	2	1.228
11	2	1.310		33	2	1.210
12	2	1.364		34	2	1.221
13	2	1.333		35	2	1.216
14	2	1.340		36	2	1.203
15	2	1.284		37	2	1.193
16	2	1.343		38	2	1.207
17	2	1.281		39	2	1.203
18	2	1.294		40	2	1.204
19	2	1.282		41	2	1.185
20	2	1.293		42	2	1.199
21	2	1.249		43	2	1.181
22	2	1.276		44	2	1.193
23	2	1.254		45	2	1.185

The total number of auctions that can be set up over all classes is upper bounded by the product of the number of groups in each class. The total number of auctions is guaranteed by the lower bound  $\Omega(n^{ct})$  where  $n$  is the number of items in a type,  $t$  is the number of types, and  $c$  is a constant.

The bucket size, number of buckets, number of unique configurations, and auction sizes for the proposed model are given in Figure 25.

	<b>Object Type</b>	<b>#</b>	<b>Bucket Size</b>		<b># of Buckets</b>	<b># Unique Bucketed Item Combinations</b>	<b># Unique Non-bucketed Item Combinations</b>
	A	20	5		4	5	635
	B	16	8		2	2	231
	C	40	10		4	5	38566
	D	7	7		1	1	15
	E	5	1		5	7	7
	F	6	2		3	3	11
<b>Totals</b>	<b>6 Types</b>	<b>94</b>			<b>19</b>	<b>1050 Auctions (<math>\prod</math> Unique Combos)</b>	<b>6.5339E+12</b>

**Figure 25**

Each object type is discretized into a bucket size of at least one object, reducing the number of items that can compete in an auction. The number of unique combinations of the discretized buckets is then shown to be far fewer (by several factors) than the original level.

For example, object type-C is discretized in 10-object buckets, creating four total buckets that can be uniquely distributed in five ways, as opposed to the 38,566 ways that 40 identical items can be uniquely distributed. The last line totals the number of auctions required, which is a product of the number of unique distributions for each type.

Note the significant reduction in the number of auctions required due to the use of buckets, from 6.5339E+12 to 1,050, a factor of 6.2E + 9. Where  $t$  is the number of object types and  $g_i$  is the number of combinations for a particular type  $i$ , the number of auctions (i.e. different configurations) required becomes

$$\# \text{ Configurations} = \prod_{i=1}^t (g_i)$$

While the order of growth is still NP-complete, the larger terms are first to be reduced and have the most significant impact. After all the auctions are run, the total utilities of each auction are compared and the highest is selected as optimal.

The reason the upper bound, rather than exact growth order, is mentioned for the auction count is that some of the auctions in the original case will be identical, such as the case with object type C. Imagine two auctions, the first with eleven items of size-two buckets and the remaining items as 18 single buckets. The second has twelve items of size-two buckets and sixteen items single buckets. Because there are only ten bidders, the asymmetric auction will appear the same to the bidders. In each auction, every bidder has the right to choose whichever item it prefers, still leaving both types of items available for the rest. This occurs whenever the number of items available is equal to at least  $3e + 2$ , where  $e$  is the number of items. This occurrence can be exploited in auctions with few bidders and many objects of similar types by limiting the number of items of a particular type and size that participate in the auction to the number of total bidders in each auction.

### **3.5 Setting Up the Auctions**

This solution technique requires a number of auctions be set up, run, and compared. In every auction, the same ten bidders will hold constant. However, the number of items and actual items bid for by the bidders will be unique in every auction. An asymmetric auction will be used to assign bidders to items. This will allow objects to remain unassigned if optimal.

Prior to running the auctions, a table listing the utilities of all the possible bidder-item pairs is required. Not all utilities will be used in each auction, but over the course of running all auctions, each utility will be used at least once. Attention should be given to the resources needed to compute the utilities for the auctions. The finer the grain in number of items and buckets, the larger the necessary computation set both to determine the utilities and to solve the auctions. However, the bound of optimality is compromised as the grain becomes coarser.

Figure 26 shows the utilities of different items assigned to each bidder. Note the nonlinear nature of the utilities (Utility[5A]  $\neq$   $\frac{1}{2}$ \* Utility[10A] for ever bidder).

**Bidder-Item Pairing Utilities**

ITEMS

	# Objects	5	10	15	20	8	16	10	20	30	40	7	1	2	3	4	5	2	4	6
	Type->	A	A	A	A	B	B	C	C	C	C	D	E	E	E	E	E	F	F	F
Bidder																				
Q		2	10	16	20	-3	4	-3	-8	0	5	-2	2	-1	-5	-16	-2	1	1	-6
R		-3	5	5	-10	-3	-5	4	7	9	4	-4	7	-2	-2	0	-18	-3	2	-4
S		6	5	4	3	-3	6	5	0	0	15	-6	6	3	0	-1	2	-7	-2	0
T		5	6	7	8	-3	-5	6	0	-3	-40	8	9	-1	-5	7	12	-7	1	-4
U		10	5	0	-10	-3	-5	-3	3	9	-30	-8	4	-6	2	-8	-10	8	4	6
V		7	-40	-41	-42	-3	-5	-9	0	1	-40	9	-2	-4	0	-2	7	6	-4	2
W		4	5	6	7	-3	-5	-2	1	-5	16	9	11	-2	-14	11	12	-5	7	0
X		5	10	11	12	-3	-5	0	-3	-9	10	3	-2	3	11	12	-11	-6	2	-3
Y		7	6	5	4	-3	-5	3	4	4	8	3	-2	-6	-3	0	4	-8	1	3
Z		-1	-2	-3	20	-3	12	0	6	-10	-12	14	0	-4	3	-5	0	0	6	-1

**Figure 26**

The utility values of pairing a particular bidder with a particular number of objects of a certain type.

As noted in the constraints, neither every bidder nor every object need be assigned. The bidder issue is addressed by introducing artificial objects that have a utility

of zero for every bidder. The number of artificial objects is equal to the number of real bidders so that each bidder has the option of not choosing an object (choosing the artificial object instead). In this case, ten artificial objects are introduced to the auction (object type G).

The option for objects to be either assigned or unassigned is addressed by the combination of the artificial objects existing along with the utilization of the asymmetric auction. This allows, in the extreme case, every bidder to be assigned to an artificial object and for no objects to be paired. However, if it is more valuable to assign a real object, the assignment to the real object will occur, leaving an artificial object unassigned.

The algorithm is summarized as follows:

**Assignment Algorithm**

*One object type per bidder, no constraint of number of objects*

*Objects and bidders can be assigned or unassigned*

*Order =  $O(c^{mt}(m+n)n^2a)$*

*$g$  = # unique combinations of a single type =  $O(c^m)$*

*$t$  = # of types of objects*

*$n$  = number of bidders*

*$m$  = number of objects*

Set utility\_max = 0

For each Combination of Items over All Item Types:  $O(c^{mt})$

{  
Run an Auction with the Specific Items and Artificial Objects  $O((m+n)n^2a)$   
and compute utility.

If utility > utility\_max, set utility\_max = utility and save assignments.  $O(1)$

}

The advantage of this algorithm over a fully enumerated case lies in the use of auctions where objects of a similar type exist. The larger the number of objects in the class and the larger the number of bidders, the more computational effort saved by using



auction. Fully enumerating every case for each bidder is an NP-complete problem, replacing the auction component of polynomial order  $O((m+n)n^2a)$  with an NP-order  $O([n+1]^{[m]})$ . Because this term is multiplicative in the overall algorithm, the time savings are substantial.

Additionally, bucketing objects into groups allows the outside loop of the algorithm to scale better. Though the theoretical bound is the same, which is  $O(n^n)$ , the actual values of  $n$  are shrunk to a desirable level. For example, by bucketing object A into four buckets, each containing five of object A, there are only five different ways for the objects to be distributed among bidders. With 20 individual objects, there are 635 combinations. The number of combinations of object types are multiplied together to reach the total number of auctions to be run. Hence, a problem with two times the number of groups in each type increases the number of auctions by a factor of  $2^{(\# \text{ of types})}$ , or generically by a factor of  $[(\text{groups now}/\text{groups before})^{(\# \text{ of types})}]$ . Because the growth of groups is exponential with the number of buckets, the number of auctions required grows tremendously with the number of buckets, with the previous term in the exponent.

The total number of auctions in this example required by bucketing according to the given parameters is equal to the product of the number of groups for each type, equaling 1050. With the auctions ranging in size from 10 bidder by 16 objects to 10 bidders by 29 objects, the answer was computed in 0.77 seconds on a Pentium 166MHz machine with 32 megabytes of RAM. Not introducing artificial variables (forcing as many bidders as possible to be assigned) reduces the time to 0.45 seconds. Similarity classes were implemented for objects.

If desired, the bucket sizes could be altered at this stage to more accurately solve the problem, based on the results of the first series of auctions. This would require a utility matrix of the finer grain utility values for the object classes desired, while it may excuse other types of objects from the auction altogether, thereby keeping the computational requirements low. Further research into optimal methods of grouping objects over the course of subsequent auctions promises gains in efficiency.

The results of the current solution technique are very encouraging for future application.



**Auction 2**

The second auction illustrates a change in the Type-A object group. Where four separate items each of five A-type objects had existed, two of the items have been consolidated and will be delivered together, reducing the total number of objects by one and changing the available A-type objects to be bid for to two separate items of five A-type objects and a single ten A-type objects item.

**Type A modified**

Type	Objects in Group									
A	5	5	10							
B	8	8								
C	10	10	10	10						
D	7									
E	1	1	1	1	1					
F	2	2	2							
G	1	1	1	1	1	1	1	1	1	1

Auction Size = 10 X 28

The utility matrix is then constructed. Notice how there are only three A-type items that bidders can bid for now. The overall utility value of this auction is compared to the previous auction. If this auction's utility is higher, the assignments are saved as the current best and the utility value compared against future auctions. If the first auction had a higher utility, it would continue on as the current best and this auction configuration would not be saved.

**Utility Matrix Constructed for This Auction**

Bidder	Object->	# Objects										Artificial															
		A	A	A	B	B	C	C	C	C	D	E	E	E	E	E	F	F	F	G	G	G	G	G	G	G	G
Q		2	2	10	-3	-3	-3	-3	-3	-3	1	3	3	3	3	3	7	7	7	0	0	0	0	0	0	0	0
R		-3	-3	5	-3	-3	4	4	4	4	8	-8	-8	-8	-8	9	9	9	0	0	0	0	0	0	0	0	0
S		6	6	5	-3	-3	5	5	5	5	-2	6	6	6	6	6	-2	-2	-2	0	0	0	0	0	0	0	0
T		5	5	6	-3	-3	2	-4	0	11	-1	10	10	10	10	10	-10	-10	-10	0	0	0	0	0	0	0	0
U		10	10	5	-3	-3	-6	4	-1	-3	-6	3	3	3	3	3	-6	-6	-6	0	0	0	0	0	0	0	0
V		7	7	-40	-3	-3	4	15	-2	8	-8	4	4	4	4	4	-11	-11	-11	0	0	0	0	0	0	0	0
W		4	4	5	-3	-3	1	-2	-9	7	-1	7	7	7	7	7	-2	-2	-2	0	0	0	0	0	0	0	0
X		5	5	10	-3	-3	-4	-6	2	-12	10	-2	-2	-2	-2	-2	-2	-2	-2	0	0	0	0	0	0	0	0
Y		7	7	6	-3	-3	-12	2	1	-1	-2	-4	-4	-4	-4	9	9	9	0	0	0	0	0	0	0	0	0
Z		-1	-1	-2	-3	-3	0	0	-1	0	1	-1	-1	-1	-1	10	10	10	0	0	0	0	0	0	0	0	0

The third auction illustrates the final auction where all objects have been consolidated to their largest groups. In this case, all A-type objects are delivered as one item, of 20 A-type objects to one customer who wins the assignment. Note here that because object G behaves linearly, it never changes its offering of ten single G-type objects. This would likely be the last auction and all other 1064 auctions would have already occurred. At this point, the best utility of either this auction or the current best before the auction would be the overall best assignment and should be output as the optimal assignment.

**Auction 1050  
Largest Increments**

Type	Objects in Group									
A	20									
B	16									
C	40									
D	7									
E	5									
F	6									
G	1	1	1	1	1	1	1	1	1	1

**Utility Matrix Constructed for This Auction**

Auction Size = 10 X 16

Bidder	Object->	# Objects										Artificial			
		A	B	C	D	E	F	G	G	G	G	G	G	G	G
Q		20	4	5	1	-1	14	0	0	0	0	0	0	0	0
R		-10	-5	4	8	-3	-10	0	0	0	0	0	0	0	0
S		3	6	15	-2	1	-5	0	0	0	0	0	0	0	0
T		8	-5	-40	-1	0	-6	0	0	0	0	0	0	0	0
U		-10	-5	-30	-6	-5	4	0	0	0	0	0	0	0	0
V		-42	-5	-40	-8	2	-18	0	0	0	0	0	0	0	0
W		7	-5	16	-1	6	-12	0	0	0	0	0	0	0	0
X		12	-5	10	10	3	-2	0	0	0	0	0	0	0	0
Y		4	-5	8	-2	12	6	0	0	0	0	0	0	0	0
Z		20	12	-12	1	1	-10	0	0	0	0	0	0	0	0

## 4 Extension of Distributor Problem to Nonlinear Bidder Behavior

### 4.1 Problem Description

The distributor problem is now extended to account for nonlinear bidder interdependencies. An intuitive example appears in the stockbroker scenario where clients are aware of what other clients receive. Therefore, a more valuable client would be upset if he noticed a less valuable client receiving the same number of IPO shares, causing his utility to be dependent not only on the number of shares he receives, but depend on what others receive as well. Another example occurs when several clients require that they all receive shares or else they will all leave as a group. In this case, the stockbroker would decide whether to allocate shares to all members of the group or to allocate none to any member because allocating shares to a partial subset of the group would yield no future value to the broker. Instead, the broker may look to allocate shares only to clients not in the group.

In both of these examples, each combination of bidders that behaves nonlinearly must be examined individually for every object configuration. However, linearly behaving bidders can simply compete in a single auction for each object configuration, as in Chapter 3. A simple example of nonlinear bidding and object behavior is presented next to illustrate how one might handle a complex problem by still taking advantage of auction procedures. The example consists of three types of bidders and two types of objects. The bidding pool is comprised of one Q-type, one R-type, and two S-type bidders. There are two A-type objects and four B-type objects available for bidding. Figure 28 summarizes the available auction participants.

**Figure 28**

Bidders and objects to be assigned by type and number.

Bidder Types	# Bidders
Q	1
R	1
S	2

Object Types	# Objects
A	2
B	4

### 4.2 Nonlinear Object Behavior

Assuming the objects behave nonlinearly, auctions for each type of configuration must be constructed. As in the distributor problem of Chapter 3, each bidder is constrained such that he can receive either one type of object (though any number of objects of that item) or no object at all. The possible object configurations are listed in Figure 29.

**Figure 29**

The number of ways two A-type objects and four B-type objects can be uniquely distributed into same-type items is enumerated below. Ten possible configurations exist.

Configuration #	Items Available in Auction	# Items in Auction
1	A A B B B B	6
2	A A 2B B B	5
3	A A 2B 2B	4
4	A A 3B B	4
5	A A 4B	3
6	2A B B B B	5
7	2A 2B B B	4
8	2A 2B 2B	3
9	2A 3B B	3
10	2A 4B	2

For each object configuration, a separate auction must occur. While running a multi-assignment auction would seem a favorable alternative to enumerating the combinations, the nonlinear combinations of objects do not allow the multi-assignment auction approach to work correctly. A critical requirement is that the assignments are added linearly to obtain the overall score. In comparing two different configurations run through enumeration, configurations 1 and 10 would likely obtain different scores. Imagining huge incentives for single bidders each owning all of one type, in which configuration ten would likely produce the high score. However, the multi-assignment auction would not be able to account for the additional incentive provided by the owning all of the type and could lead to a suboptimal solution, perhaps distributing a single object to each bidder as in configuration one. This occurs because an auction uses a value that represents the score of the individual item only, independent of any others. Each configuration is then compared and the best score is recorded as the optimal assignment.

#### **4.3 Nonlinear Bidder Behavior—Single Object-Type Constraint per Bidder Group**

The notion of nonlinear bidder combinations is now explored. Assume that bidders of type-Q and type-R care about what the other receives. Therefore, the score is not just a linear sum, but also has an added score dependent on what the other receives. Figure 30 shows how combinations lead to nonlinear scores.

**Figure 30**

The complete enumeration of how two A-type objects and four B-type objects can be distributed among bidders Q and R. Note how Combinations 14 and 30, when added, do not equal the score of Combination 1, indicating a dependence of bidders Q and R on each other.

Combo #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Q	A	A	A	A	A	2A	2A	2A	2A	B	B	B	2B	A	2A	B	2B	3B	4B
R	A	B	2B	3B	4B	B	2B	3B	4B	B	2B	3B	2B	-	-	-	-	-	-
Score	11	7	3	5	2	6	7	8	4	10	9	20	18	4	2	5	2	4	6

Combo #	20	21	22	23	24	25	26	27	28	-	29	30	-	31	32	33	34	35	36
Q	Nothing	B	2B	3B	4B	B	2B	3B	4B		2B	3B		-	-	-	-	-	-
R	Nothing	A	A	A	A	2A	2A	2A	2A		B	B		A	2A	B	2B	3B	4B
Score	0	2	6	4	6	8	4	2	4		8	18		4	3	6	4	3	6

Because of the nonlinearity, the same procedure used for the initial distributor problem cannot be used. However, nonlinear bidders can be evaluated and then a representative linearly behaving bidder can participate in the auction. To achieve this linearly behaving bidder, different object combinations are evaluated among the various ways they can be assigned to the bidders. Each total combination of objects, including A, 2A, B, 2B, 3B, and 4B, is evaluated with bidders Q and R, and the best score is selected as the utility to be used in the auction. Figure 31 breaks up the combinations of Figure 30 into groups based on the total objects used, limiting the groups to only those confined to a single type of object. This is a more stringent constraint than previously, where bidders Q and R could receive different types of objects. Figure 32 shows which combinations are restricted from participating in the auction. Removal of the single object-type constraint is explored in Section 4.4. The assumption may lead to a suboptimal solution, but will shorten the time to reach a solution.



**Figure 31**

**Breakdown of Combinations by Objects Used (Single-type Object for Bidders)**

This chart enumerates the different ways that a single type of object can be divided among Bidders Q and R. For example, a single type-A object can be assigned to bidder Q, as in QA. Likewise, a single type-A object can be assigned to bidder R, as in RA. These are the two ways in which a single A-type object can be distributed among bidders Q and R. Two A-type object can be divided in three different ways. The first way is with bidders R and Q receiving a single A-type object (QA RA), the second way is with bidder Q receiving both (Q2A), and the third way is with bidder R receiving both (R2A). The best combo row signifies which distribution scores the highest given the number and type of object distributed. In the single A-type object case, either RA or QA scores 4. In the two A-type object case, the QA RA (one A-type object assigned to bidder R and one to bidder S) scores the highest (11) and will be chosen to represent the 2A item score in the auction.

Breakdown of Combinations by Objects Used (Single-type Object for Bidders)												
Total Objects	A	2A	B	2B	3B	4B	Total					
Assignment/Score	QA	4	QA RA	11	QB	5	QB RB	10	QB R2B	9	QB R3B	20
	RA	4	Q2A	2	RB	6	Q2B	2	R2B QB	8	Q2B R2B	18
			R2A	3			R2B	4	Q3B	4	Q3B RB	18
							R3B	3	Q4B	6	R4B	6
# Possible Assign.	2	3	2	3	4	5	19					
Best Assign./Score	QA	4	QA RA	11	RB	6	QB RB	10	QB R2B	9	QB R3B	20

**Figure 32**

**Combinations Not Included in Auction (Cross-type Combinations)**

Because of the single object-type constraint on the auction problem, if bidders Q and R are to participate as a single bidder, combinations in which they receive different types of objects are excluded from the auction. Therefore, the following combinations are not considered and could actually contain an optimal assignment. However, their exclusion decreases the size of the problem and may not affect the solution if a single type of item is likely to be allocated to the nonlinear bidder group (Q and R).

Combinations Not Included in Auction (Cross-type Combinations)																
Total Objects	AB	A2B	A3B	A4B	2AB	2A2B	2A3B	2A4B	Total							
Assignment/Score	QA RB	7	QA R2B	3	QA R3B	5	QA R4B	2	Q2A RB	6	Q2A R2B	7	Q2A R3B	8	Q2A R4B	4
	QB RA	2	Q2B RA	6	QB R3A	4	Q4B RA	6	QB R2A	8	Q2B R2A	4	Q3B R2A	2	Q4B R2A	4
# Possible Assign.	2	2	2	2	2	2	2	2	16							
Best Assign./Score	QA RB	7	Q2B RA	6	QA R3B	5	Q4B RA	6	QB R2A	8	Q2A R2B	7	Q2A R3B	8	Q2A R4B	4

**Figure 33**

**Utility Matrix for Bidders Q and R and Single-type Items**

The best scores from Figure 31 are used to create a single bidder entity representing bidders Q and R. The actual distribution of the objects to Q and R must be saved so that when the auction terminates with an assignment for the QR bidder, the individual objects can be assigned directly to Q or R.

	Items					
	A	2A	B	2B	3B	4B
QR	4	11	6	10	9	20

**Figure 34**

**Complete Utility Matrix for Bidders Q, R, and S**

The bidders for the auction include a single QR bidder along with two S-type bidders. Each S-type bidder has the same utility values and acts independently of all other items (linear behavior). Values will be extracted from this table when constructing different auction configurations.

Bidders	Items					
	A	2A	B	2B	3B	4B
QR	4	11	6	10	9	20
S	5	7	3	11	14	15

Figure 33 assembles the best utilities for each item assignment for the nonlinear bidding group. If the nonlinear bidding group were to receive the item, it would allocate the objects in the item in the best way, suggesting that the best score should be used for the auction. For example, if the QR bidder received an item consisting of two A-type objects, it would examine the different ways to distribute the two objects to Q and R and do so optimally. Figure 31 lists the three unique ways to divide the objects and concludes that distributing a single A-type object to each of bidder Q and bidder R results in the highest score, equaling 11. Therefore, the auction should let the score of 11 represent the value of a two A-type object assignment to bidder QR. The same procedure is applied to each object combination (A, 2A, B, 2B, 3B, 4B) to produce the utility row for each nonlinear bidding group (just QR in this example). The final utility matrix containing all values is then created in Figure 34 by entering a row for each unique linear bidding entity. While there are two S-type bidders, they are identical and can reference the same row of the utility matrix when constructing the auctions.

Auctions are then run for each item configuration, as shown in Figure 35. The ten unique configurations each use the same three bidders (QR, S, and S) and each generate a total utility score. The highest score corresponds to the optimal assignment, taken from the auction that produced that score.

**Figure 35**

The ten unique item configurations are each run as separate, asymmetric single-assignment auctions. The bidders this time, however, are a single QR bidder and two S-type bidders. Three artificial items are added to allow the bidders to remain unassigned. The shaded squares are the assignments producing the optimal score. Visually, there can be at most one shaded square per column, representing that the item in the column is assigned to the bidder in the shaded row. The total score is a simple linear sum of the shaded squared. The highest score is achieved in Auction 8, where QR is assigned to 2A, S to 2B, and the other S to 2B. Figure 32 holds that a 2A assignment to QR consists of a QA and an RA assignment. Hence, the final assignment is QA, RA, S-2B, and S-2B, for a score of 33.

Auction 1		Objects								
Bidders	A	A	B	B	B	B	Art.	Art.	Art.	
QR	4	4	6	6	6	6	0	0	0	
S	5	5	3	3	3	3	0	0	0	
S	5	5	3	3	3	3	0	0	0	
Score	76									

Auction 2		Objects						
Bidders	A	A	2B	B	B	Art.	Art.	
QR	4	4	10	6	6	0	0	
S	5	5	11	3	3	0	0	
S	5	5	11	3	3	0	0	
Score	22							

Auction 3		Objects						
Bidders	A	A	2B	2B	Art.	Art.	Art.	
QR	4	4	10	10	0	0	0	
S	5	5	11	11	0	0	0	
S	5	5	11	11	0	0	0	
Score	26							

Auction 4		Objects						
Bidders	A	A	3B	B	Art.	Art.	Art.	
QR	4	4	9	6	0	0	0	
S	5	5	14	3	0	0	0	
S	5	5	14	3	0	0	0	
Score	25							

Auction 5		Objects				
Bidders	A	A	4B	Art.	Art.	
QR	4	4	20	0	0	
S	5	5	15	0	0	
S	5	5	15	0	0	
Score	30					

Auction 6		Objects						
Bidders	2A	B	B	B	Art.	Art.	Art.	
QR	11	6	6	6	6	0	0	
S	7	3	3	3	3	0	0	
S	7	3	3	3	3	0	0	
Score	17							

Auction 7		Objects						
Bidders	2A	2B	B	B	Art.	Art.	Art.	
QR	11	10	6	6	0	0	0	
S	7	11	3	3	0	0	0	
S	7	11	3	3	0	0	0	
Score	25							

Auction 8		Objects					
Bidders	2A	2B	2B	Art.	Art.	Art.	
QR	11	10	10	0	0	0	
S	7	11	11	0	0	0	
S	7	11	11	0	0	0	
Score	33						

Auction 9		Objects					
Bidders	2A	3B	B	Art.	Art.	Art.	
QR	11	9	6	0	0	0	
S	7	14	3	0	0	0	
S	7	14	3	0	0	0	
Score	28						

Auction 10		Objects				
Bidders	2A	4B	Art.	Art.	Art.	
QR	11	20	0	0	0	
S	7	15	0	0	0	
S	7	15	0	0	0	
Score	27					

#### 4.4 Nonlinear Bidder Behavior—Single Object-Type per Bidder

Whereas the previous section only allowed a single nonlinear bidding group to receive a single type of object, this section permits each bidder in the nonlinear bidding group to receive its own type of item, welcoming the assignments of Figure 32 to be considered in the problem. Therefore, the assignment of an item composed of an A-type object and a B-type object can be assigned to the QR bidder, which can then point to a distribution that pairs the A-type object with bidder Q and the B-type object with bidder R. Note that the single-type item constraint still applies to each individual bidder, as described in the original distribution problem.

The methodology is to create more item configurations that enable each member of a nonlinear bidder group to be assigned a different type of item. With the QR bidder, there must be two types of items considered. If there were  $n$  bidders in the nonlinear bidder group and  $n + m$  items available, all combinations of  $n$  different types of items would need to be calculated. The combinations for the QR bidder are enumerated in Figure 32. The complete list of item configurations is enumerated in Figure 38. Each configuration is run as a separate asymmetric single-assignment auction, using values from Figure 37 and the three bidders QR, S, and S. The same procedure as in Section 4.3 is followed, finding the highest score of the 22 auctions and looking up the corresponding nonlinear distribution for the individual members of the nonlinear groups.

## Figure 36

### Utility Matrix for Bidders Q and R and Single-type Items

To allow each individual bidder (Q and R, separately) its own type of item, the best scores from both Figure 31 and Figure 32 are used to create a single bidder entity representing bidders Q and R, this time for all items of two types or less (all in this case).

	Items													
	A	2A	B	2B	3B	4B	AB	A2B	A3B	A4B	2AB	2A2B	2A3B	2A4B
QR	4	11	6	10	9	20	7	6	5	6	8	7	8	4

## Figure 37

### Complete Utility Matrix for Bidders Q, R, and S

The bidders for the auction include a single QR bidder along with two S-type bidders. However, bidder S is incapable of receiving two types of objects, such as AB or A2B. Therefore, the utility equals anything below that of an artificial variable. Accordingly, each combination of items will be broken out, so that A and B are separate items and can be bid on independently by bidder S. The only case in which the combination would matter, therefore, is when it is assigned to QR. Hence, no time is wasted in calculating redundant values for bidder S for multiple-type items.

Bidders	Single-type Items						Multiple-type Items							
	A	2A	B	2B	3B	4B	AB	A2B	A3B	A4B	2AB	2A2B	2A3B	2A4B
QR	4	11	6	10	9	20	7	6	5	6	8	7	8	4
S	5	7	3	11	14	15	-1	-1	-1	-1	-1	-1	-1	-1

*Values are -1 because they are impossible assignments. Therefore, an artificial variable will always be assigned instead.*

## Figure 38

### Complete Utility Matrix for Bidders Q, R, and S

The total number of configurations is now increased to allow the single QR bidding entity to be paired with a two-type item, such as AB or 2AB, as enumerated in configurations 11-22.

Configuration #	Items Available in Auction						# Items in Auction
1	A	A	B	B	B	B	6
2	A	A	2B	B	B		5
3	A	A	2B	2B			4
4	A	A	3B	B			4
5	A	A	4B				3
6	2A	B	B	B	B		5
7	2A	2B	B	B			4
8	2A	2B	2B				3
9	2A	3B	B				3
10	2A	4B					2
11	AB	A	B	B	B		5
12	AB	A	2B	B			4
13	AB	A	3B				3
14	2AB	B	B	B			4
15	2AB	2B	B				3
16	2AB	3B					2
17	A2B	A	B	B			4
18	A2B	A	2B				3
19	2A2B	B	B				3
20	2A2B	2B					2
21	2A3B	B					2
22	2A4B						1

#### **4.5 Nonlinear Bidder Behavior Based on Items and Not Bidders**

A special type of problem exists in which the nonlinear bidding behavior depends not on what objects other specific bidders receive, but rather depends on what objects have been assigned, regardless of who receives them. In this case, imagine a bidder that is nonlinear with many items, yet other items are linear with each other. First, enumerate all ways in which the objects can be distributed, similarly to Section 4.3, so that for every group of items, the best assignment can be found. Run auctions for each item configuration, where now the configurations consist not only of different ways in which all items are distributed, but also include partial distributions. This way, the nonlinear bidder can receive the items not included in the partial distribution and a total score for the unique distribution can be determined later by complete enumeration (as required for nonlinear problems). This score will equal the best assignment score of the partial distribution to the linear group coupled with the nonlinear bidder receiving the rest of the distribution (or a subset of it). This procedure can potentially be applied a successive number of times, essentially cascading the nonlinear groups such that the number of enumerations is reduced to allow auctions to handle intermediate, linear groupings. This method relies heavily on the structure of the problem and deserves future consideration.

#### **4.6 The Makings of a Good Nonlinear Bidder Group**

How one knows when to group objects into nonlinear groups is based on two primary factors. First, the fewer bidders involved in the nonlinear behavior, the better. Because enumeration is an NP-complete process, finding the optimal assignments for a large number of bidders will not occur quickly. Representing only the highly nonlinear

behavior by the group and allowing a margin of error for other bidders who are not included can give faster results, but at the cost of optimality, of course.

A second factor depends on how many object types exist. A small number of object types will not increase the number of different bidding configurations by much, as in the example given here needing a mere 12 more configurations. However, an example with a three bidder group and twenty different object types would require 1140 different ways for arranging the single multiple-type item. Multiplying this by the number of ways the remaining items can be grouped soon approaches a large computational requirement. While limiting the group to a single type of object, as in Chapter 4.3, will keep the number substantially lower, it will also compromise the optimality.

This last point implies that if a group of objects is likely to request an assignment of all the same type, then it makes sense to group them together, as the number of configurations will be kept much lower. While there is the cost of precomputing the best scores of the intra-group distribution, it may be worth the optimality increase.

The essence of a good nonlinear group lies in the ability to compartmentalize a nonlinear segment of the problem into a linearly behaving segment that can subsequently interact with the remainder of the problem. If this can be done accurately, the procedure can maintain precision as well as auction efficiency. If there are not a lot of bidders of the same type and many of the bidders depend on each other, auction may not be the best approach. As can be inferred from all the above assumptions, the power of auctions comes from linear behavior and from identical objects. The problems require at least some amount of linear behavior for auctions to exploit. Otherwise, the process essentially begins enumerating all the possibilities and may actually require more work



than pure enumeration from auction overhead. Figures 39 through 42 present an example that shows a process of identifying and compartmentalizing nonlinear groups of objects and bidders. In order for the process to occur, data must be given for individual and grouped assignments. These can be compared for linearity to see if the group utility matches the individual utility for different objects and bidders. The results of this comparison can then help the auctioneer more accurately determine which objects and bidders to treat as nonlinear pieces.

These results clearly indicate that the size of problems can be cut to computationally feasible sizes provided assumptions on assignments can be made.

## Example: Identifying Nonlinear Groupings

**Figure 39**

Independent Utilities (if only that assignment occurred)							
	1A	2A	1B	2B	1C	1D	
Q		2	6	8	2	6	3
R		2	8	8	6	2	2
S		6	9	9	8	9	3
T		5	4	9	4	6	8
U		6	7	7	7	3	7

In this example (Figures 39-42), the data of Figure 39 and Figure 40 are supplied. Figure 39 represents a single assignment of only one bidder to one object or to two of the same type of object. No other assignments were made when these values were computed.

**Figure 40**

Pairs of Assignments and the Corresponding Utilities							
	2A	2B	AC	AD	BC	BD	CD
QR	8	16	8	5	14	11	4
QS	9	17	12	9	17	12	6
QT	7	17	11	10	15	16	18
QU	8	15	12	9	13	15	5
RS	11	11	16	20	21	5	3
RT	24	30	0	18	16	4	2
RU	8	15	8	9	11	15	5
ST	2	4	5	3	8	4	13
SU	12	16	15	13	16	16	5
TU	11	16	12	14	13	16	1

Figure 40 presents data of multiple assignments, where two objects were assigned to a group of two bidders. We are not given the specific breakdown of how the individual objects were paired with the bidders, but only the collective utility.

**Figure 41**

Hypothetical Linear Pair Utilities Based on Independent Utilities							
	2A	2B	AC	AD	BC	BD	CD
QR	8	16	8	5	14	11	8
QS	9	17	12	9	17	12	12
QT	7	17	11	10	15	16	14
QU	8	15	12	9	13	15	13
RS	9	17	11	8	17	11	11
RT	8	17	8	10	14	16	10
RU	8	15	8	9	11	15	9
ST	11	18	14	14	18	17	17
SU	12	16	15	13	16	16	16
TU	11	16	12	14	13	16	13

Although we do not know the breakdown, we can compute the likely linear behavior of a group of objects assigned to a group of bidders based on the independent values. We will assume that a group of two objects is distributed optimally to the group of bidders, hence the group utility is the configuration yielding the MAX utility of independent assignments. For QR-2A, we look in Figure 39 and compare Q-2A (6), Q-A R-A (2+2 = 4), and R-2A (8). The MAX is R-2A and the utility value is inserted into the hypothetical linear pair utilities chart in Figure 41.

**Figure 42**

Difference in Actual Pair Utilities and Hypothetical Linear Pair Utilities  
Figure 42 = Figure 41 - Figure 40

	2A	2B	AC	AD	BC	BD	CD
QR	0	0	0	0	0	0	4
QS	0	0	0	0	0	0	6
QT	0	0	0	0	0	0	-4
QU	0	0	0	0	0	0	8
RS	-2	6	-5	-12	-4	6	8
RT	-16	-13	8	-8	-2	12	8
RU	0	0	0	0	0	0	4
ST	9	14	9	11	10	13	4
SU	0	0	0	0	0	0	11
TU	0	0	0	0	0	0	12

The actual utilities of pairs are now compared with the hypothetical linearized pair utilities by taking the difference, presented in Figure 42. A value of zero signifies that the utilities of the group assignment behaved exactly as the expected linear combination, whereas a nonzero value indicates nonlinear behavior in the grouping of either objects or utilities. Nonlinear objects will appear nonlinear to many bidders and nonlinear bidders will appear nonlinear to many objects.

We quickly notice by inspection in Figure 42 that RS, RT, and ST tend to deviate from the predicted linear group utility for all objects. We infer from this that bidders R, S, and T behave as a nonlinear group (among each other they are nonlinear, however, they appear to behave linearly with others as in QR, QS, QT, RU, etc.) and will separate them in a full assignment auction. We also notice that objects C and D have nonzero differences for all bidders, suggesting that they behave as a nonlinear object group (though they appear linear to the rest of the objects). We will create a single bidder to represent objects C and D in the auction.

## 5 Conclusion

Auctions present an opportunity for reducing the computational requirements of complex problems. Problems that have a particular structure, even if nonlinear, can benefit greatly by using auctions to circumvent a large number of unnecessary cases. The structures that are most viable for improvement contain groups of items that are identical, especially those that behave linearly. Additionally, problems with items that can be partitioned in a limited number of ways allow auctions to be reduced in size, as well as decrease the number of auctions required to reach optimality.

While nonlinear problems are still NP-hard, the scale of problems that can be solved grows tremendously by use of these techniques. This is reflected both by allowing larger numbers of items to participate in a given auction as well as in modeling problems to embrace a much higher degree of complexity. Large-scale auctions arrive at solutions very quickly by grouping the items into similarity classes. For nonlinear problems, auctions are run on all different possible configurations of items. Groups of identical items are exploited to readily reduce the number of auctions required by eliminating a number of configurations that are essentially identical. The coupling of a reduction in nonlinear possibilities with polynomial-time scalable auctions provides a powerful optimization technique for large and complex problems.

Efficient algorithms rely on breaking the problem into linear and nonlinear pieces. The more linear the model, the faster the solution can be reached. Approximating nonlinear behavior into larger discrete units can help narrow the range of a much larger problem, which can subsequently be solved at the desired precision. While optimality cannot be guaranteed, time constraints can be adhered to. Hence, the tradeoff boils down

to speed versus accuracy in creating a simplified model to represent a real-life complex model. The key lies in boosting accuracy as much as possible while keeping the time requirement low.

The specific techniques described here include bucketing, one-type object assignment limits, and expansion of only nonlinear segments. Bucketing simply allows the number of items within the auction to be reduced by taking a coarser view of the data. It essentially resizes the number of items into fewer items with larger pieces. While on the one hand it reduces combinatorial complexity and serves as an accurate guide in fairly linear environments, it can overlook drastic nonlinear anomalies, leading to a suboptimal conclusion. A possible extension of research would be in investigating smart techniques in adjusting bucket sizes. Additionally, this work has explored only same-size buckets, yet a mix of different bucket sizes could potentially lead to effective management of complexity within time bounds.

Single-type object assignment limits were imposed in the research to capture a specific type of domain-specific behavior. This limit assumes that a pattern exists in the assignment problem such that each bidder will only receive a single type of object. If this assumption holds, the solution avoids a good amount of computation. Though this technique could be used for purely linear models to reduce the size of the auctions, the impact is felt more strongly in the nonlinear case because it eliminates cross-type combinations. These can be easily handled linearly through multi-assignment, yet require exhaustive enumeration in the nonlinear case.

Expanding only nonlinear problem segments allows a complex, NP-complete problem to be reduced to linear elements, requiring only nonlinear segments to adhere to

the NP-complete requirement. After simplifying a segment, it can then participate in an auction that is solved in polynomial time. This is achieved by calculating and storing the best score and subassignment for a group of nonlinear bidders. Additionally, the degree to which constraints are relaxed can be determined by the specific problem and allow the number of auctions to decrease if certain patterns exist, such as one-type nonlinear groupings. Because the number of auctions in each segment more than add with each other to produce the total required, isolating and solving subproblems can avert a large number of potential combinations.

Auctions are very fast at handling linear behaviors. As such, any part of a problem that can be described linearly allows full exploitation of auctions. Allowing nonlinear pieces of the problem to be expanded to capture their nonlinear behaviors while keeping the rest of the model linear limits the growth of solution time. Because auctions can accommodate large problems quickly, the linear pieces can be fully and easily examined in every nonlinear scenario, especially through use of similarity classes. Though the auction size is larger in purely linear scenarios, the required number of auctions shrinks to overcompensate this.

Potential future areas of research that have come up through the investigation of this problem include the following topics. Bucketing can substantially reduce time, but how can nonlinearity be accounted for in the best selection of bucket sizes and the descent toward the smallest discrete levels? Even in completely linear models, the discreteness affects the optimal assignment because the granularity can be too large. Another area involves techniques for determining the margins of error when deciding a segment of a problem behaves nonlinearly. By limiting the actual number of members

that participate in the nonlinear group, other nonlinearities may not be accounted for in the auction. To what extent will this compromise the accuracy of the results? Departing from the general techniques described here, does a method exist in which a dynamic auction structure allow real-time perturbations of the existing assignment problem to be solved more quickly based on either predefined pricing or by precomputing “close” problems. This real-time procedure would explore auctions that are very similar, varying by an object or bidder, or by a different utility value for a particular entry that would be likely to occur. The premise here is that not all information is available to begin with, but that small amounts of information will be added to the existing assignment problem later, when time becomes a critical issue. Calculating “close” problems along the way can exploit redundant calculations, especially in nonlinear expansions, even though the available objects or bidders vary per problem. Therefore, savings can be made both in overall time by avoiding redundancies and at run-time by having precomputed results of probable variations that may occur.

The application of auction to large and complex problems promises to yield appealing results. Both the size and complexity of problems that can be modeled and simulated has grown tremendously and auctions will undoubtedly serve as a valuable optimization technique.

## Derivation of Auction Time and Space Complexity

See [4] and [6]

Assume  $n$  bidders,  $m$  objects

### Data Structures

Unassigned Bidders (stack)	$O(n)$
Bidder Assignments (vector)	$O(n)$
Utility Table	$O(nm)$
Object Assignments and Price for Object	$O(m)$

- **Space complexity =  $O(nm)$**

### Time Complexity

The minimum bidder increment is  $\epsilon = 1$

The starting prices for all objects are equal to the Min Utility.

- 1) The maximum number of bids for a object equals  $a/\epsilon = a = \text{Max Utility} - \text{Min Utility}$ .
- 2) To guarantee optimality such that  $n\epsilon < 1$ , multiply all utilities by  $n + 1$ , so  $a' = a * (n + 1)$ .  
(equivalently,  $\epsilon$  can be reduced such that it is  $< 1/n$ )
- 3) From the optimality guarantee, each object requires  $O(an)$  bids.
- 4) Since there are  $n$  people, there can be at most  $O(n^2 a)$  bids for all objects.
- 5) Each bid requires a search through the utility of each object pairing with that bidder  $O(m)$
- 6) Hence, the total number of operations is bounded by  $O(mn^2 a)$ .**

### Variations

- This assumes the use of a stack for unassigned bidders that can find an unassigned bidder in  $O(1)$ . Use of a list of people that is searched to see who is unassigned increases the order by a factor of  $n$  to  $O(mn^3 a)$ .
- Without the optimality guarantee, the solution is optimal within  $n\epsilon$ . The order is then  $O(mna)$  as each object only requires  $O(a)$  bids.
- Epsilon scaling decreases the number of bids to its log, from  $O(an)$  to  $O(\log(an))$ , and hence the total number of operations becomes  $O(mn \log(an))$
- This assumes using the Gauss-Seidel auction, where one bidder at a time bids. The Jacobian version checks all unassigned bidders simultaneously, requiring  $O(mn)$  bids, and assigning the best for each object from the group. This increases the overall bound to  $O(mn^3 a)$ . However, it decreases the total amount of bids made and the actual run-time depends on the utility landscape.

## Bibliography

- [1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., "Network Flows," Alfred P. Sloan School of Management, Massachusetts Institute Of Technology, Sloan Working Paper Number 2059-88, August 1988.
- [2] Bertsekas, D.P. "A Distributed Algorithm for the Assignment Problem," Massachusetts Institute of Technology Laboratory for Information and Decision Systems, Working Paper, 1979.
- [3] Bertsekas, D.P. "A New Algorithm for the Assignment Problem," *Mathematical Programming*, Vol. 21, pp. 152-171, 1981.
- [4] Bertsekas, D.P., Linear Network Optimization: Algorithms and Codes, Chapters 1-5, M.I.T. Press, Cambridge, Massachusetts, 1991.
- [5] Bertsekas, D.P., "Mathematical Equivalence of the Auction Algorithm for Assignment and the  $\epsilon$ -Relaxation (Preflow-Push) Method for Min Cost Flow," Massachusetts Institute of Technology Laboratory for Information and Decision Systems, Report P-2147, November 1992.
- [6] Bertsekas, D.P., "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," Massachusetts Institute of Technology Laboratory for Information and Decision Systems, Report P-2108, May 1992.
- [7] Bertsekas, D.P., and Castañon, D.A., "A Forward/Reverse Auction Algorithm for the Asymmetric Assignment Problem," ALPHATECH, Inc., Burlington, MA, April 1992.
- [8] Bertsekas, D.P., and Castañon, D.A., "A Forward/Reverse Auction Algorithm for Asymmetric Assignment Problems," Massachusetts Institute of Technology Laboratory for Information and Decision Systems, Report P-2159, January 1993.
- [9] Bertsekas, D.P., and Eckstein, J. "Dual Coordinate Step Methods for Linear Network Flow Problems," *Mathematical Programming*, Series B, 1988.
- [10] Bertsimas, D., and Tsitsiklis, J.N., Introduction to Linear Optimization, Chapter 7, Athena Scientific, Belmont, Massachusetts, 1997.
- [11] Castañon, D.A., "Reverse Auction Algorithms for Assignment Problems," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1992.
- [12] Dantzig, G.B., "Application of the Simplex Method to a Transportation Problem," T. C. Koopmans (ed.), *Activity Analysis of Production and Allocation*, Wiley, pp. 359-373, 1951



[13] Freling, R., Pinto Paixão, J.M., and Wagelmans, A.P.M., Models and Algorithms for Vehicle Scheduling.” To appear in Transportation Science, 2000.

[14] Kuhn, H.W., “The Hungarian Method for the Assignment Problem,” Naval Research Logistics Quarterly, Volume 2, pp. 83-97, 1955.