

**IMPROVED OPPORTUNITY COST ALGORITHM FOR CARRIER  
SELECTION IN COMBINATORIAL AUCTIONS**

A Thesis

Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
Requirements for the degree of  
Master of Science in Industrial Engineering

in

The Department of Industrial and Manufacturing Systems Engineering

by

Viswanath Uma Gnanasekaran  
B.E. , Anna University, Chennai, India, 2000  
August 2004

## **Acknowledgements**

I would like to express my sincere gratitude to the members of my thesis committee for their constant guidance and support. I am particularly indebted to my major professor, Dr. Gerald Knapp, for his guidance, support and constant encouragement during the course of this research. I am grateful to Dr. Thomas Ray and Dr. Baba Sarker for serving on my committee and for their suggestions and encouragement.

Finally, I am grateful to my parents and my sister for their constant support and encouragement without which this research would not have been possible. I would also like to thank all my friends, especially all the students working in the Systems Integration Lab for their help when needed.

# Table of Contents

Acknowledgements.....	ii
List of Tables .....	v
List of Figures.....	v
Abstract.....	vii
Chapter 1. Introduction.....	1
1.1 Supply Chain.....	1
1.2 Supply Chain Management.....	4
1.3 Transportation in Supply Chain.....	5
1.4 Auctions .....	6
1.4.1 English Auctions.....	7
1.4.2 Dutch Auctions .....	8
1.4.3 First-Price, Sealed Bid Auctions.....	8
1.4.4 Vickery Auctions .....	9
1.4.5 Double Auctions .....	10
1.5 Combinatorial Auctions.....	10
1.6 Combinatorial Auctions for Transportation Service Procurement .....	13
Chapter 2. Problem Statement .....	15
Chapter 3. Literature Review.....	18
3.1. Opportunity Cost Algorithm.....	21
3.2. Local Ratio Opportunity Cost Algorithm .....	22
Chapter 4. Methodology .....	24
4.1 Opportunity Cost Algorithm.....	24
4.2 Effects of Node Ordering.....	25
4.3. Effect of Graph Topology on Opportunity Cost Algorithm .....	27
4.3.1. All Nodes Have Zero or One Predecessor and Successor .....	27
4.3.2 One Node With Two Successors .....	30
4.3.3. One Node With Two Predecessors .....	33
4.3.4. One Node With Two Successors; One Node With Two Predecessors.....	35
4.3.5. Double Counting in Opportunity Cost Algorithm .....	38
4.3.6. Effect Of Node Removal .....	40
4.4. Recursive Recalculation in OPCOST (OPCOST-R).....	41
4.5. Maximum Total Revenue Using OPCOST (MTR).....	45
Chapter 5. Results.....	47
5.1. Problem Dataset.....	47
5.2. Types of Distribution in Dataset.....	47
5.2.1 Random.....	48
5.2.2 Weighted Random .....	48

5.2.3 Uniform.....	48
5.2.4 Decay .....	49
5.3 Experimental Results .....	49
5.3.1 Random Distribution Results.....	49
5.3.2 Weighted Random Results.....	50
5.3.3 Uniform Distribution Results.....	53
5.3.4 Decay Distribution Results .....	53
5.3.5. Results On Bigger Problems.....	59
Chapter 6. Analysis.....	61
6.1. Solution Quality With Respect to Optimal .....	61
6.2. Solution Quality On Distributions .....	64
6.3. Execution Time.....	66
Chapter 7. Conclusions .....	71
References.....	72
Vita.....	75

## List of Tables

Table 1: Random Distribution Class Problem Results For OPCOST-R.....	51
Table 2: Random Distribution Class Problem Results For MTR .....	51
Table 3: Weighted Random Distribution Class Problem Results For OPCOST-R.....	52
Table 4: Uniform distribution Class Problem Results For OPCOST-R .....	54
Table 5: Decay Distribution Class Problem Results For OPCOST-R.....	55
Table 6: Decay Distribution Class Problem Results For OPCOST-R (contd.) .....	56
Table 7: Decay Distribution Class Problem Results For MTR.....	57
Table 8: Decay Distribution Class Problem Results For MTR (contd.).....	58
Table 9: Results For Bigger Problems – Solution Quality .....	60
Table 10: Results For Bigger Problems – Execution Time .....	60
Table 11: Comparison With The Optimal Solution.....	63

## List of Figures

Figure 1: A Simple Supply Chain Model .....	3
Figure 2: A More Complex Supply Chain.....	3
Figure 3: Example Bid Graph.....	26
Figure 4: All Nodes Have Zero or One Predecessors and Successors.....	29
Figure 5: One Node With Two Successors.....	31
Figure 6: One Node With Two Predecessors.....	34
Figure 7: One Node With Two Successors; One Node With Two Predecessors. ....	36
Figure 8: Bid Graph With Double Counting.....	39
Figure 9: Opportunity Cost Recalculation Example.....	43
Figure 10: Solution Quality By Problem Size .....	62
Figure 11: Solution Quality By Distribution Class.....	65
Figure 12: Execution Time vs. Number Of Bids (OPCOST, OPCOST-R and MTR) .....	68
Figure 13: Execution Time vs. Number of Bids (OPCOST and OPCOST-R).....	69
Figure 14: Execution Time vs. Number Of Items.....	70

## **Abstract**

Transportation costs constitute up to thirty percent of the total costs involved in a supply chain. Outsourcing the transportation service requirements to third party logistics providers have been widely adopted, as they are economically more rational than owning and operating a service. Transportation service procurement has been traditionally done through an auctioning process where the auctioneer (shipper) auctions lanes (distinct delivery routes) to bidders (carriers). Individual lanes were being auctioned separately disallowing the carriers to express complements and substitutes. Using combinatorial auctions mechanism to auction all available lanes together would allow the carriers to take advantage of the lane bundles, their existing service schedule, probability of securing other lanes and available capacity to offer services at lower rates and be more competitive. The winners of the auction are the set of non-overlapping bids that minimize the cost for the shippers. The winner determination problem to be solved in determining the optimal allocation of the services in such kind of combinatorial auctions is a NP-hard problem. Many heuristics like approximate linear programming, stochastic local search have proposed to find an approximate solution to the problem in a reasonable amount of time. Akcoglu et al [22] developed the opportunity cost algorithm using the “local ratio technique” to compute a greedy solution to the problem. A recalculation modification to the opportunity cost algorithm has been formulated where opportunity costs are recalculated every time for the set of remaining bids after eliminating the bid chosen to be a part of the winning solution and its conflicts have eliminated. Another method that formulates the winning solution based on the maximum total revenue values calculated for each bid using the opportunity cost algorithm has also been researched.

## **Chapter 1. Introduction**

As businesses strive to reduce costs to remain on top of the competition, one of the major areas that has attracted a lot of attention is supply chain systems and especially transportation. Techniques such as combinatorial auctions have recently been used to increase the efficiency of the transportation procurement process. This research introduces some modifications to the opportunity cost algorithm formulated by Akcoglu et al [22] to improve the solutions for the winner determination problem in combinatorial auctions. A maximum total revenue (MTR) method that uses the opportunity cost algorithm has also been formulated to produce better quality solutions.

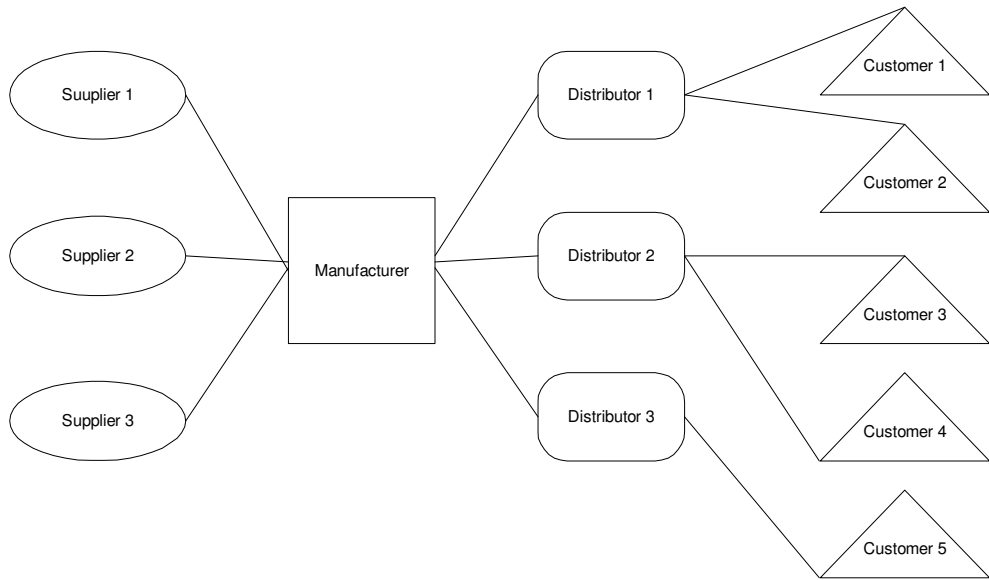
### **1.1 Supply Chain**

A supply chain is a network that is comprised of the production and distribution facilities that procure raw materials and transform these materials into intermediate and finished goods and distribute the finished goods to customers [1]. The example supply chain depicted in Figure 1 consists of suppliers, a manufacturer, distributors and customers. The supplier supplies raw materials required for the production process to the manufacturer. The manufacturer processes the raw material and has the finished goods ready for the distributors to ship them to the customers when and where they want it.

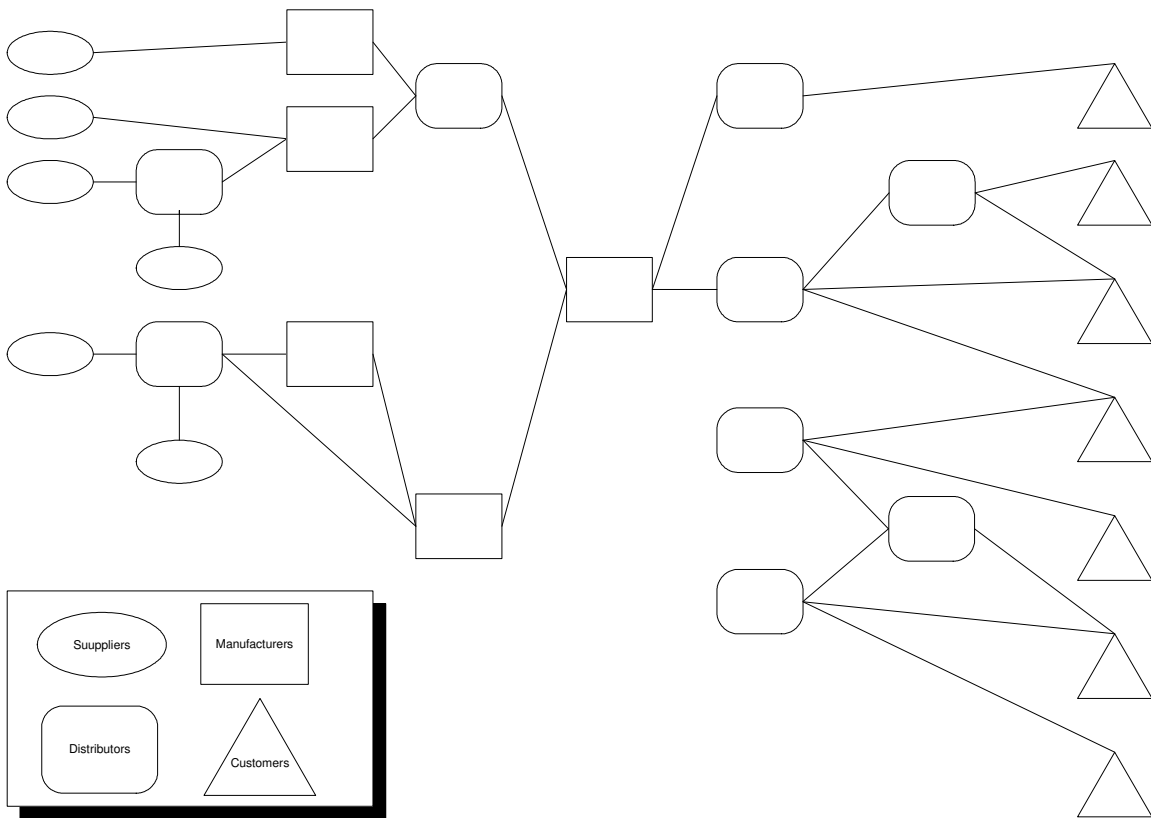
The term supply chain may also be applied to a large company with several sites often located in different countries. Coordinating materials and information flows for such a company in an efficient manner is a formidable task. However, the resultant network will be a simple supply chain, similar to the network shown in Figure 1. Supply chain may include many different organizations working together to meet the customer demand by coordinating material, information and financial flow between themselves.

Coordinating the activities of the different organizations involved in the supply chain is a complex process and is critical to the successful operation of the supply chain.

Supply chain networks get more complicated as the number of participants in the network grows and the flow of materials and information between them become more interwoven. Suppliers by themselves can form a large network and the transfer of the raw materials from the suppliers to the manufacturer is a logistical problem of its own. The manufacturing process in the supply chain network need not be a single step as shown in Figure 1. The manufacturer can process the raw material in stages to produce an intermediate product. The intermediate products require transportation from one manufacturer to another. The production process can thus consist of a network of various manufacturers. Once the product has been processed to completion, the product gets shipped to the various distributors in the supply chain. The distributor then ships the final product to the final destination, the customers. The distribution process may involve various distributors contributing to complete the delivery of the products to the customers. All distributors may not be able to carry out delivery to all the customers in their various geographically dispersed locations. So, they have to utilize the distribution network of their distribution partners who may be able to reach those customer locations to complete delivery. Distribution centers are used to make the transfer of goods possible between distributors. The distributors own warehouses and distribution centers in various locations to aid the process. Thus supply chain in the real world may be considerably more complex than the supply chain network shown in the Figure 1. A real world supply chain might look more like the network shown in Figure 2.



**Figure 1: A Simple Supply Chain Model**



**Figure 2: A More Complex Supply Chain**

## **1.2 Supply Chain Management**

Supply chain management's objective is to integrate and manage the sourcing, flow and control of materials and information between the various participants in the supply chain network in order to strike a balance between the conflicting goals of high customer service, low inventory management and low unit cost.

Supply chain management takes a systems approach to view the whole supply chain network as a single entity, rather than as individual organizations, integrating the various participants to manage the total flow of goods from the supplier to the customer [1]. All participants in the supply chain network directly and indirectly affect the performance of all other members in the network, thus affecting the overall performance of the total network. It seeks to synchronize the various intra-firm and inter-firm operations to reduce the total cost involved and deliver the products to the customers in a timely manner. By working together, all the participants in the supply chain enhance revenue by reducing inventories, lowering operating costs, increasing product availability, increasing resource utilization as well as by raising the standards of customer satisfaction. The efficiency and the response time of the entire supply chain is one of the keys to its success because getting the products to the customer at the right time is an integral part of customer satisfaction.

The six key elements to a supply chain are production, supply, inventory, location, transportation and information [1]. Decisions involving each of these factors have a ripple effect and travel through the entire supply chain to eventually dictate the efficiency of the supply chain.

### **1.3 Transportation in Supply Chain**

Transportation of goods is a major part of the supply chain and contributes to a significant portion of the costs of operation of the supply chain. Every movement of goods involves decisions on the kind of transport mode to be used and who carries out the transportation of the goods. The transportation can be done either by the supplier of the goods or the receiver of the goods, or the transportation can be outsourced to third party logistic providers. Transportation also takes place within each site, but these movements are classified under material handling issues and are handled by the production planning process. The transportation phase of the supply chain is concerned only with the transportation between suppliers, manufacturers, and customers and intermediaries such as warehouses or distributors. Physical distribution is not only a significant cost for most businesses, it also has a direct and significant impact on its competitiveness through speed, reliability and its availability. As customers expect faster and quicker delivery, the transportation in the supply chain has to adapt to meet those demands while trying to reduce the overall cost of operation of the supply chain.

Operating a transportation service that is capable of handling such flexible and changing requirements is expensive for most businesses to handle. For them, contracting their transportation requirements to third party logistics providers makes the most economic sense and helps them concentrate on their core business areas. Third party logistics providers offer a highly competitive physical distribution system, which can handle shipments of varying sizes with tight delivery deadlines. Reasons to outsource the transportation requirements extend beyond economic benefits. It eliminates the need for a

huge initial capital investment. Other reasons to outsource transportation may include the following.

- **Control.** Owning a transport service that is not being utilized to its fullest efficiency is a liability and an overhead on company resources. Logistics companies offer good control without the burden of inflexibility of your own vehicles.
- **Service.** Outsourcing transportation requirements allows businesses to concentrate more on their core competency areas and make their products better.
- **Flexibility.** Owning and operating a transport service requires a constant demand. Varying demands on the system may lead to inefficiencies and decrease in quality of service. Third-party logistics providers are well equipped to handle varying demands and can execute orders of any size within any time period.
- **Management.** Managing a self owned transportation service become complex as the demands on the system increases. It not only requires maintenance of the system but also the recruitment and training of personnel to operate it. Transportation backed by an inefficient management renders the transportation operation uncompetitive against specialized and better-managed logistic providers. The results of such a transport system are reflected in the unit price the customer pays for the service.

#### **1.4 Auctions**

An auction is an excellent mechanism to determine the value of goods that do not have a determined market value and to distribute them to those who value them most highly. An auction offers the advantage of simplicity in determining market-based prices. It is simple

and efficient and ensures that sellers receive the best available value for their goods or services. The price of goods are set by the bidders and not the seller, thus as the demand for the goods increase so does the price the offered by the bidders increasing the revenue of the sellers.

There are different ways to classify auctions. There are open auctions as well as sealed-bid auctions. There are one-sided auctions where only bids are permitted, but not "asks", where a seller puts up a price for the goods to be matched by the bidders. In a double auction bids and asks take place at the same time. There are auctions where the price ascends and auctions where the price drops at regular intervals. Vickrey established the basic taxonomy of auctions [2] based upon the order in which prices are quoted and the manner in which bids are tendered. He established four major auction types: English, Dutch, First-Price sealed-bid, and Vickrey (uniform second-price).

#### **1.4.1 English Auctions**

The English auction, also known as the open-outcry auction or the ascending price auction is the most commonly used. Milgrom [3] defines the English auction in the following manner. "Here the auctioneer begins with the lowest acceptable price--the reserve price-- and proceeds to solicit successively higher bids from the customers until no one will increase the bid. The item is 'knocked down' (sold) to the highest bidder." If the reserve price is not met for some goods, the auctioneer can opt to hold on to the item. The items can be auctioned after some time when the auctioneer feels that it might bring him the reserve price or an even better price. In some cases, the auctioneer can opt to start the auctions without setting a reserve price, mainly in order to thwart colluding bidders. The manner in which the auctions are held varies widely.

Competition is at its highest in the English auctions. Winner's curse (paying more for an item than its value) is a discouraging factor in this type of auction because inexperienced participants bid up the price.

#### **1.4.2 Dutch Auctions**

Dutch auctions are also commonly known as decreasing price auctions or the uniform second-price auction. The Dutch auctions use the open bid format rather than the sealed bid format. In these auctions the starting price is set to an extremely high price. The auctioneer calls out the set starting price and looks out for any bidders ready to take up on the offer. If no bidders are ready to take the goods at that price then the auctioneer progressively lowers the price till he can find bidders willing to take up the items at that price. In the case where multiple items are being auctioned, more bidders put out bids as the price keeps lowering. As a result of which the winners of the initial auctions pay a heavier price than the bidders that take the latter units in the auction. And the price at which the items are bought gets lower as the auction progresses for the multiple items. The basic concept behind the Dutch auctions is that the longer the item remains unsold the lower the price fetched by the item.

#### **1.4.3 First-Price, Sealed Bid Auctions**

The primary characteristics of these auctions are that the bids are sealed and the participating bidders are unaware of the prices bid by the other bidders unlike the open bidding methods of the English and the Dutch auctions. In these auctions, the bidders submit their prices in complete ignorance of their competitor's prices. Generally, the sealed bid auctions have two phases. During the first phase, the auctioneer accepts sealed bids from the various bidders till a preset time deadline is reached. The second phase

involves resolution of the submitted bids, that is, the auctioneer opens up the received bids and determines the winner of the auction. Again there exists a distinction between single and multiple bid auctions. The name 'First-price' comes from the fact that the goods are awarded to the bidder with the highest price. In the case of multiple item auctions, it is called 'discriminatory' because not all the winners pay the same price. In the multiple unit auctions, all the received bids are sorted in a descending order of their price and items are allocated in order from the highest bidder until the items are exhausted. The downside in this regard is that the winners end up paying different prices for the same item.

#### **1.4.4 Vickery Auctions**

Vickery auctions are otherwise called uniform second-price auctions. Vickery auctions are also sealed bid auctions and the bidders are ignorant of their competitors' prices. But the distinguishing factor is that the winners of the Vickery auctions do not pay the amount they bid but instead the amount of the highest losing bid. On first sight, the Vickery auctions seem to be very seller unfriendly because of the fact that the seller settles for price that is less than the highest offered price. But deeper analysis shows that it is not the truth. In the discriminatory First-price sealed bid auctions, the bidders fear the fact that they might end up paying more for the same goods than somebody else. So the bidders tend to adjust their bids downwards. This kind of thinking across the bidders eventually brings down the price. On the other hand, the Vickery auctions encourage the bidders to upsize their bids. Because higher bids do not mean that the bidder is paying an uncompetitive price but the bidder just pays the price of the highest losing bid that is

closer to the market value. Thus the price that the winner pays is not entirely depended on his actions but solely on the bid prices of the competitors.

#### **1.4.5 Double Auctions**

In double auctions, both the seller and the buyer submit bids, that is, the auction is not one sided and is more interactive. These are used to create a demand profile. The bids are arranged in order from highest to lowest. From the profiles that have been generated, the maximum quantity exchanged can be determined by matching selling offers (starting with lowest price and moving up) with demand bids (starting with highest price and moving down). This format allows buyers to make offers and sellers to accept those offers at any particular moment. A "continuous double auction" is one in which many individual transactions are carried on at a single moment and trading does not stop as each auction is concluded.

#### **1.5 Combinatorial Auctions**

In the previous auction formats items are auctioned separately either sequentially or in parallel. The bidders are forced to bid on each item separately and speculate on the value of each item individually. But the actual value of the item depends on what other items the bidder receives in the auction. The bidders associate a price with a specific collection of goods; associating a value with the individual items can be problematic. This requires the bidder to look ahead and base the price decisions of its bids on its expectations to win other items in future auctions. Even after forecasting and predicting the expected course of the auctions, there remains an uncertainty factor due to the absence of complete information about the other bidders participating in the auction. This leads to inefficient

allocations of items where the bidders do not win their required combination of products and as a result the bidders may value it at a price less than what they had paid for it.

The inefficiencies in the allocations resulting from the sequential and parallel auction mechanisms can be overcome by various techniques. The bidders can be allowed to retract their bids when they do not get the combinations they needed. These items can then be auctioned again or the item can be allocated to the bidder who ended up second. But if the winning price of the second auction was less than the price of the first then the difference in the amount has to be paid by the retractor as a penalty. Another approach that has been practiced is to sell the option for retracting upfront. Also an aftermarket can be setup where the bidders exchange items among them after the auction has ended. This approach can undo some of the inefficiencies in the allocation. But to reach an optimal allocation among the bidders there might have to be impossibly large number of exchanges between the bidders [4].

Combinatorial auctions can be used to overcome deficiencies of the single item sequential or parallel auctions. In combinatorial auctions, instead of selling items individually, the seller allows bidders to bid on collections or bundles of items. The bidder is allowed to express complements and substitutes between items being auctioned. This allows the bidders to express their requirements completely without room for speculation and avoid the risk of obtaining incomplete bundles. The bidders do not have to interpolate the outcome of other auctions with complementary and substitutable items when valuing prices of items or bundles of items.

Combinatorial auctions have been successfully used in various fields to replace sequential parallel auctions to improve revenues or reduce costs. Volvo conducted a

combinatorial auction in February 2001 for the procurement of wooden packaging material [25]. Volvo auctioned the contracts for 600 commodities aggregated in 14 segments. The result of the combinatorial auction helped Volvo to reduce its total cost from \$180 million to \$172.9 million Swedish Kronors. The saving of \$7.1 million is a substantial amount considering that Volvo just had to change its auctioning mechanism. The auction results also reduced the number of winning suppliers from 15 to 6, which on its own resulted in cost savings in terms of reduced administrative and overhead costs. The efficiency and the benefits resulting from combinatorial auctions attracted Federal Communications Commission to switch its auction mechanism to combinatorial auctions in June 2002 [26]. Other proposals to use combinatorial auctions for resource allocation of airport takeoff and landing time slots [15] and for the telecommunications industry [16] have been suggested.

The combinatorial auction mechanism works based on the assumption that the participating bidders are able to express their requirement exactly with out any compromise and can submit any number of bids as required. In case of auctions with small number of items, bids containing complementary and supplementary items can be expressed with the bidders submitting bids with the list of items they require and an attached value. The auctioneer has to allocate items to bidders so that the items in the allocated bids do not conflict. But as the number of items being auctioned increases and the requirements of the bidders become more complex, the number of bids that the bidder has to submit in order to completely and exactly express his requirements becomes larger in size. Allowing the bidders to submit their requirements in the form of a computer

algorithm or program that completely express the requirement can solve this problem of exponential number of bids.

## **1.6 Combinatorial Auctions for Transportation Service Procurement**

As discussed in Section 1.2, transportation of goods is a major factor in supply chain management. The transportation of the goods in many networks contributes up to thirty percent of the total supply chain costs. In many cases, it is more economical for the shippers to outsource their transportation to various third party logistic providers than to operate a transportation service of their own. In the past, when the shippers needed to obtain transportation services for a set of distinctive delivery routes (called lanes) with different origins and destinations or delivery schedules, they would send out a request for quotes to all of their carrier partners. When the shippers need to transport goods or materials, they send out a request for quotes to all of their carrier partners. The carriers study the request and come up with a price that they would require to satisfy the request. The carriers submit their bid amount to the shippers who, after receiving replies from all the carriers, evaluates the best quotes in order to reduce its own transportation costs. This process adopted by the shippers is a simple sealed-bid auction process to determine the carriers and in turn optimize the shippers' costs. This system may be able to achieve economies of scale for carriers. However, it ignores the economies of complements and substitutes that are inherent in transportation operations. In some cases, the shippers would negotiate a price for bundles of lanes with each carrier one at a time.

Carriers tend to have different valuations for different combinations of lanes depending on whether the lanes in the bundle are complementary or substitutable. The cost of operating lanes exhibits interdependencies. Valuations for bundles may differ by a

substantial amount between carriers depending on their available capacity, previously scheduled deliveries and the possibility of obtaining another complementary lane from other shippers. The carriers can use their existing capacity in half-truck and empty movements to fit the shipper's needs and be able to offer the service at a considerably lower price than other carriers. Caplice studied these interdependencies [11] and observed that the traditional procurement process does not properly make use of this property.

With the help of new software and information technology systems, it has become possible for the shippers to make all lanes available for bidding to all the carriers at the same time. This enables the carriers to bid on any combinations of available lanes that fit well with its existing capacity and schedule thus fully utilizing the complements and substitutes existing between all the lanes available for bidding. This process of introducing many lanes for auctioning simultaneously can be carried out efficiently using combinatorial auction mechanisms. Shippers can use combinatorial auction mechanisms to auction the lanes and make significant cost savings [12] using the synergies of sets of lanes. Ledyard et al [12] recorded the procurement of trucking services by Sears Logistics Services involving 854 different lanes with a service cost of approximately \$190 million per year. Sears Logistics Services switched from the traditional trucking service acquisition process to using a multi-round combinatorial reverse auction in which participating carriers were pre-selected to guarantee service levels. In each round a tentative winning price is announced and the auction continues until the terminating rule is satisfied. Using this variation of the combinatorial process, Sears Logistics Services reported a 13% savings, which reduced its transportation procurement cost by \$25 million per year.

## Chapter 2. Problem Statement

In this research, we study the single-unit transportation service procurement problem using the traditional request-for-quote-and-negotiation process and the effects of conducting the process using combinatorial auction mechanisms. In this problem, the shipper requires transport for a set of lanes. Each of the lanes in the requirement has only a single unit demand; in other words, all of the lanes in the set are unique. If the shipper has a demand of multiple units for the same lane, then the shipper has to represent the multiple requirements as separate single unit requirement and uniquely identify them. The shipper receives bids from all the interested bidders and has a set of bids with conflicting lanes possibly multiple bids from individual carriers. The shipper has to examine all the bids received and pick a winning combination of bids. The winning combination must minimize the total cost incurred by the shipper while satisfying the demand for all the lanes. In any outcome of the combinatorial auction, the winning combination must be disjoint set of lanes that is no single lane can be present in more than one winning combination of bids or in other words no single should be allocated to more than one carrier.

Winner determination in sequential and parallel auction mechanisms is as easy as picking the bid with the largest quoted price for each item separately. Picking the winner of such auctions is linear in time. The computing time is linear with the number of bids received for all of the items on auction. That is, if  $m$  is the number of items that are being auctioned and  $n$  the number of bidders bidding on those items, then the time for computation is equal to  $m*n$ .

Unfortunately in combinatorial auctions, the winner determination is not as simple as in the case of sequential and parallel auctions. The bidders are allowed to submit any number of bids, on any combinations of bundles and the items in the bundles can overlap. That is, we have a set of all the items up for bid  $M$  and a collection of bids  $B$  that are subsets of  $M$  with non-negative weights. Finding the smallest total weight independent set is equivalent to a Set Packing Problem (SPP) problem. Obtaining the optimal solution in such cases would involve checking all possible combinations that are possible with the collections of bids  $B$  that are submitted. The number of solutions that need to be checked in that would be equal to  $2^{|B|}$ , which would be impossible to solve optimally for increasingly large number of bids. Although the optimal solution for cases with small number of bids may be obtainable in a reasonable amount of time as the number of bids increases, the time required for computation explodes. The winner determination problem belongs to a class of problems called NP-hard. No polynomial time algorithms have yet been suggested for problems that belong to the NP-hard class. Such problems are solved by either relaxing some of the requirements of the problem definition or by deploying heuristics that find a winning solution that is close to the optimal solution in a reasonable amount of time.

Many heuristics have been developed to tackle the problem of winner determination in combinatorial auctions. Various models of the problem have been used to handle the complexity of the problem. Section 3 describes some of the history and methods that have been developed in the past to solve this problem. One of the heuristics developed to handle the winner determination problem is the Opportunity Cost (OPCOST) algorithm developed by Akcoglu et al [22]. The OPCOST algorithm gives a

heuristic solution to the problem in a reasonable amount of time. It is proved to produce better solutions than the greedy algorithm for an acceptable increase in the amount of computations.

The objective of this research is to study the OPCOST algorithm and to improve upon the quality of solution produced by the OPCOST algorithm. The effect of the number of bids and the number of items on the solution quality will also be studied. Both the solution quality and the time of execution for OPCOST and modifications will be compared together and studied.

### Chapter 3. Literature Review

While the winner determination is a NP-Hard problem, it has been well formulated. The winner determination problem has been formulated before as a dynamic programming problem [6], integer programming problem [6, 7, 9, 18] and a weighted set packing problem [8, 13]. De Vries and Vohra [14] gave their formulation of the problem and reviewed earlier approaches used to solve the problem. Rothkopf et al [13] modeled the problem as a maximum-weight set packing problem and studied the structure of the problem. Sandholm [6] modeled the winner determination as a dynamic programming problem and formulated an IDA\* method to determine the optimal solution to the problem. The approach makes use of the sparseness of the search space in the problem and structures the branch and bound search so as to consider every possible winning combination and selects the best of all the possible winning combinations. The branching in the IDA\* algorithm is based on the set of items present in each bid. The maximum average price of the items in the set is used to provide a bounding on the search space. Later, Sandholm [17] formulated another improved method that branches based on the bids and not on the items comprising the bids. Both algorithms formulated by Sandholm find the optimal winning combination among all the possible winning combinations. The execution time for auctions with thousand or more items becomes high. Jones and Koehler [21] formulated the combinatorial auction winner determination problem in a television advertisement time slot auction as a constraint satisfaction problem and developed heuristics to find the approximate optimal winning combination in an efficient manner.

Many heuristics have been implemented to find a near optimal solution without taking too much processing time. Nissan and Zurel [9] formulated the problem as an integer-programming problem. They proposed an approximate linear programming method to order the bids based on their desirability. A greedy algorithm was used to determine the winning combination from the set of the ordered bids. The approximate solution derived from this method had an approximation error of less than 1% compared to the solutions obtained by De Vries and Vohra and the processing time required was less than a minute for large auctions with 1000 items and 100,000 bids on a Pentium 450MHZ machine. The heuristic although assumes that all the items in the problem have to be allocated and this reduces the problem from a NP – Hard to a NP – Complete problem. Sakurai et al [10] proposed another method applying limited discrepancy search to Sandholm’s IDA\* algorithm to determine the approximate winning combination. The limited discrepancy search limits the search to the most promising regions of the combinatorial tree. The algorithm is able to find up to a 95% approximate solution in about 1% of the time required for the IDA\* algorithm. Gonen and Lehmann [18] and Leyton-Brown et al [19] suggested improved functions for obtaining tighter bounds in the IDA\* search tree and limit the search to the most desirable solution areas and other heuristics to find out the winning combination. Hoos and Boutilier [8] used stochastic local search techniques to solve the winner determination problem and showed that the performance and the solution quality were very impressive.

Akcoçlu, et al [22] proposed a general framework to exploit the topological structures of the bids to determine the winning bids with a good approximation ratio in linear time. They modeled the winner determination problem as a maximum weight

independent set problem and have come up with the “Opportunity Cost” and the “Local Ratio Opportunity Cost” algorithms to obtain an approximate solution to the problem. Both the algorithms return the same approximation of the optimal solution. The set of all submitted bids  $B$  can be formulated into a conflict bid graph  $G$  where the bids are the nodes and an edge occurs between any two bids that share the same product. The winning solution is one where no two bids of the solution are connected by an edge and the optimal solution is the maximum weight solution among all the winning solutions. The opportunity cost algorithm is different from the greedy algorithm because of the fact that they take into account the cost of excluding previously considered neighbors of a chosen node. Since this accounting requires propagating information only between neighbors, it increases the running time by at most a small constant factor, and yet in many cases produces a great improvement in the approximation ratio.

The quality of the approximation depends on the local structure of the ordered input graph  $G$ . For each node  $v$  in  $G$ , all successors of  $v$  (adjacent nodes that appear later in the ordering) are examined. The maximum size of any independent set among  $v$  and its successors is called the directed local independence number at  $v$  and is represented by  $\beta(v)$ , which is the maximum size of any independent set among  $v$  and its successors. The maximum value of  $\beta(v)$  over all nodes in the graph will be written as  $\beta(G)$  or just  $\beta$  and is the directed local independence number of  $G$ . The opportunity cost algorithms approximate a maximum weight independent set to within a factor of  $\beta$ . By comparison, the greedy algorithm approximates a maximum weight independent set within a ratio of the maximum size of any independent subset of both the predecessors and the successors of any node, which in general can be much larger than  $\beta$ . Finding  $\beta$  for any undirected

bid graph is again a NP-hard problem on its own. So the bound on the approximation ratio even though provable is not easily obtained. The approximation ratio can vary for each problem depending upon the structure and size of the problem.

### 3.1. Opportunity Cost Algorithm

This section explains the opportunity cost algorithm proposed by Akcoglu et al [22]. Let  $B$  be the set of all the bids that were submitted in the auction. The bid set  $B$  is formulated into a directed bid graph  $G$ ;  $E$  represents the set of edges in  $G$ . The notation  $u \rightarrow v$  is used if  $uv \in E$  and  $u$  is called a predecessor of  $v$  and  $v$  a successor of  $u$ . The set of all predecessors of  $u$  will be written as  $P(u)$  and the set of all successors as  $S(u)$ .

Given a directed acyclic graph  $G_0 = (B_0, E_0)$  with weights  $weight(v)$  for each  $v$  in  $B_0$ , the opportunity cost algorithm proceeds in two stages. The opportunity cost method  $OPCOST(G_0)$  is given below.

- OC1 Traverse the nodes according to the topological order of  $G_0$ . Compute  $value(u)$  for each node  $u$ .  $value(u)$  represents an estimate of the gain we expect by including  $u$  in the independent set. The  $value(u)$  is computed by taking the weight of  $u$  and subtracting an opportunity cost which consists of the values of earlier positive-value nodes that conflict with  $u$ .

$$value(u) = weight(u) - \sum_{v \rightarrow u} \max(0, value(v))$$

- OC2 In the reverse topological order, add any node with non-negative value to the desired independent set  $V$  and discard its predecessors.

$$select(u) = [value(u) \geq 0] \wedge \forall v \in S(u) : \neg select(v)$$

The output of the above algorithm is the set  $V$  defined as all  $u$  for which  $\text{select}(u)$  is true. Also the output set  $V$  is an independent set.

The opportunity cost algorithm calculates  $\text{value}(v)$  for each node  $v$  in time proportional to its in-degree i.e. its number of predecessors and  $\text{select}(v)$  for each node in time proportional to its out-degree i.e. its number of successors. Thus the total time for execution of the Opportunity cost algorithm is  $O(|B_0| + |E_0|)$ . The opportunity cost algorithm approximates the solution problem of the winner determination in combinatorial auctions to within a factor of the directed local independence number of the bid graph  $G$  [22],  $\beta$ .

### 3.2. Local Ratio Opportunity Cost Algorithm

The local ratio technique was originally proposed and developed by Bar-Yehuda and Even [23] and later extended by Bar-Noy et al [24]. Local ratio technique can be used to recursively find approximate solutions to optimization problems over vectors in  $\mathfrak{R}^n$ , subject to a set of feasibility constraints.

Given a directed acyclic graph  $G_0 = (B_0, E_0)$  with weights  $\text{weight}(v)$  for each  $v$  in  $B_0$ , pass  $(G_0, \text{weight}(\cdot))$  to the following recursive procedure. The procedure takes a graph  $G$  and a weight function  $w$  and proceeds as below

- LR1 Delete all nodes in  $G$  with non-positive weight. Let this new graph be  $G_2$ .
- LR2 If  $G_2$  has no nodes, return the empty set.
- LR3 Otherwise, select a node  $u$  with no predecessors in  $G_2$ , and decompose the weight function  $w$  as  $w = w_1 + w_2$ , where

$$w_1(v) = \begin{cases} w(u), & \text{if } v \in \{u\} \cup P(u), \\ 0, & \text{otherwise} \end{cases}$$

And  $w_2 = w - w_1$ .

- LR4 Solve the problem recursively using  $(G_2, w_2)$  as input. Let  $V_2$  be the approximation to a maximum weight independent set returned by this recursive call.
- LR5 If  $V_2 \cup \{u\}$  is an independent set, return  $V = V_2 \cup \{u\}$ . Otherwise, return  $V = V_2$ .

## Chapter 4. Methodology

This research focuses on the problem of winner determination of combinatorial auctions as discussed in Section 2. The problem is proven to be NP-hard and the optimal solution cannot generally be computed in linear time. Many efforts have been made towards developing heuristics that produce a solution of acceptable quality in a reasonable amount of time. The existing approaches are studied and extended to develop a new method to formulate an algorithm that produces better quality solutions.

### 4.1 Opportunity Cost Algorithm

The opportunity cost algorithm distinguishes itself from the greedy algorithm by taking into consideration the cost of excluding previously considered neighbors of a given bid in the given directed acyclic graph. The opportunity cost algorithm only increases the run time by a small factor since only the neighbors of any node are considered and yet results in a considerable improvement in the approximation ratio of the resultant solution.

Consider the example illustrated in Figure 3. The nodes in the bid graph represent the bids and the weights the corresponding bid amounts. The nodes have been ordered in an ascending order by weights. The edges in the graph represent the conflicts between the bids i.e. bids that share a common item cannot both be in the winning solution and thus conflict with each other. The edges always extend from the lower weight bid to the higher weight bid and thus are directed. The following steps show the opportunity value calculations for each of the node in the bid graph.

$$v_1 = w_1 = 5;$$

$$v_2 = w_2 - \max(0, v_1) = 10 - 5 = 5;$$

$$v_3 = w_3 = 15;$$

$$v_4 = w_4 - \max(0, v_1) - \max(0, v_2) - \max(0, v_3) = 27 - 5 - 5 - 15 = 2;$$

$$v_5 = w_5 - \max(0, v_4) - \max(0, v_3) = 30 - 2 - 15 = 13;$$

$$v_6 = w_6 - \max(0, v_5) - \max(0, v_2) = 35 - 13 - 5 = 17;$$

The winning set is initialized to null. The nodes are traversed in the reverse order and the bids that have a positive opportunity cost value and do not conflict with the current winning set are combined to form the winning solution. Thus the winning solution for the example shown in Figure 3 is {6, 4}. The total revenue from the winning solution is 62, which is also the maximum possible revenue from the bid set.

#### **4.2 Effects of Node Ordering**

The performance of the opportunity cost algorithm is strongly related to the order in which the nodes are processed. The processing order of the nodes affects the value of  $\beta(u)$  for each node  $u$ . That is, the topological order of the directed acyclic graph  $G_0$  directly affects the quality of the solution. The node ordering used to form the directed acyclic graph  $G_0$  from the undirected graph  $G$  needs to be paid considerable attention.

It has been proven [22] that for every undirected bid graph  $G$  there exists a topological orientation  $G'$  that will produce the maximum independent set of  $G$  with the opportunity cost algorithm. It has also been shown that formulating the orientation  $G'$  by itself is a problem that is equivalent to the maximum independent set problem. Therefore any bid graph  $G$  can be arranged in an orientation  $G'$  to produce the optimal solution to the winner determination problem. Since computing the orientation  $G'$  by itself is a maximum independent set problem the bid graph  $G$  must be oriented in the best order possible so as to obtain small approximation ratios.

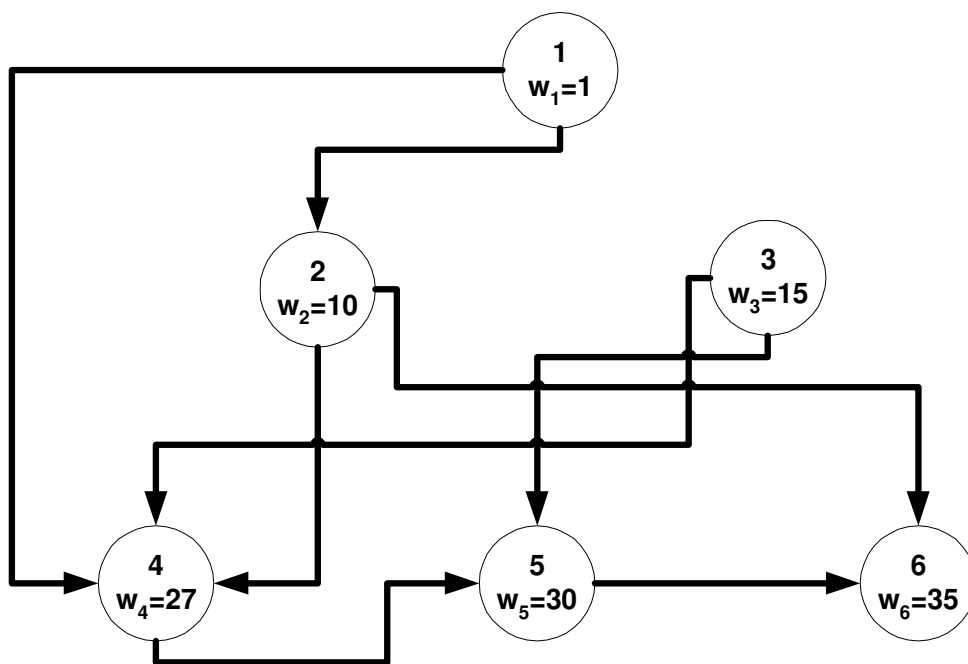


Figure 3: Example Bid Graph

In [22], it has also been proved that when  $G$  consists of all distinct value bids, then orienting  $G$  in order of decreasing weights causes the opportunity cost algorithm to return the same solution as the greedy algorithm.

### **4.3. Effect of Graph Topology on Opportunity Cost Algorithm**

This section studies the effect of different graph topologies on the opportunity cost algorithm in an effort to identify and isolate the cases where the opportunity cost algorithm fails to produce the maximum independent set. The weights of all the nodes in the bid graph are assumed to be positive since the bid amount for any bid in an auction cannot be negative. All the following cases of the bid graphs orient the bids in an increasing order by weights. Edges always extend from the lower weight bid to the higher weight bid. So in a conflict the bid with the lower weight is the predecessor and the higher weight bid is the successor. As a result the lowest weight bid does not have any predecessors and the highest weight bid does not have any successors. This kind of formulation provides us with a directed acyclic graph with all nodes having a positive weight value that is the bid amount. This topology or the structure of the bid graph has great influence on the quality of the solution produced by the opportunity cost algorithm. The following sections study certain structures of the bid graph and their effect on the OPCOST solution.

#### **4.3.1. All Nodes Have Zero or One Predecessor and Successor**

Let the number of bids or nodes in the graph be  $n$  and the number of edges be  $e$ . Since no node can have more than one predecessor or successor, we have that  $e < n$ . Also we the opportunity cost (OPCOST) value for all of the bids is always positive since the

predecessor of any bid always has a bid value less than itself. Thus the bid graph can look as shown in Figure 4.

**Theorem 1:** OPCOST always produces the optimal solution for bid graphs where all nodes have zero or one predecessor and successor.

**Proof:** In cases, where there are even number of bids the maximum independent set that can be formed is

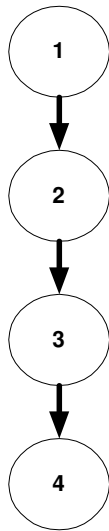
$$w_n + w_{n-2} + \dots + w_2 \geq w_{n-1} + w_{n-3} + \dots + w_1$$

Because  $w_n \geq w_{n-1}$ ,  $w_{n-2} \geq w_{n-3}$  and so on. And the even numbered bids always have a higher weight than the odd numbered values. So, the collective value of all the even numbered bids that do not conflict with each other is more than the collective value of the odd numbered bids. Substituting any even numbered with an odd numbered bid or deleting any bid only reduces the total value. Similarly, with odd number of bids the maximum independent set that can be formed is:

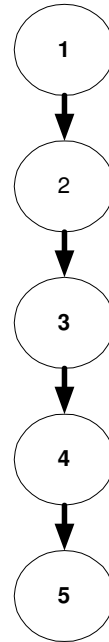
$$w_n + w_{n-2} + \dots + w_1 \geq w_{n-1} + w_{n-3} + \dots + w_2$$

where  $w_n \Rightarrow \text{weight}(n)$ . And the odd numbered bids always have a higher weight than the even numbered bids. Therefore, we can infer that  $w_n$ , the highest weighted bid, is always in the optimal solution. When there are two or more disjoint paths in the bid graph, the optimal set for each separate path is computed as above and then combined (union operation) to obtain the optimal solution set for the entire bid graph.

The OPCOST values of all the bids in the graph are positive since predecessors always have a lesser weight than the successors and no bid has more than one predecessor. Since all the bids have a positive value, the OPCOST algorithm selects the highest weighted bid to be a part of the winning solution. Thus the set of winning bids



**Even Number of bids**



**Odd Number of bids**

**Figure 4: All Nodes Have Zero or One Predecessors and Successors.**

from OPCOST is either the collection of all the even numbered bids for graphs with an even number of bids or vice versa. Therefore in both cases, the OPCOST algorithm always produces the optimal solution for bid graphs where all nodes have zero or one predecessor and successor.

### 4.3.2 One Node With Two Successors

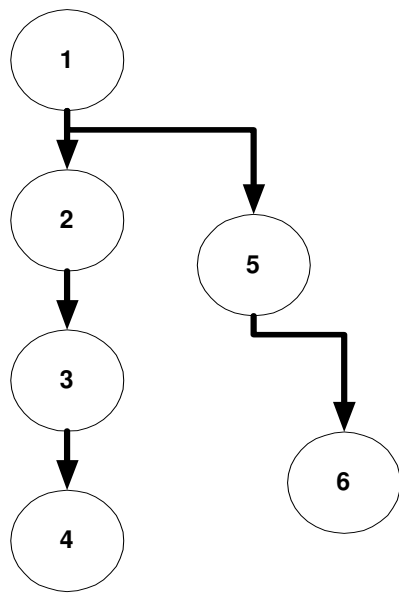
In this case, all the nodes in the bid graph have zero or one predecessor except for one node with two successors. The bid graph can look the example shown in Figure 5.

**Theorem 2:** OPCOST always produces the optimal solution for bid graphs where all nodes have zero or one predecessor and successor except for one node with two successors.

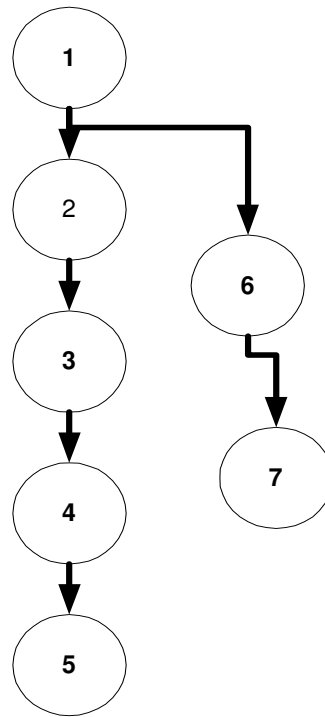
**Proof:** Since no bid in the graph has more than one predecessor, the opportunity cost (OPCOST) value for all of the bids in the graph is always positive. Assume that the bid or node with two successors is the lowest weighted bid.

In cases, where there are even number of bids  $n$ , it can be observed that one path in the graph is odd and the other even. Assume that the odd path has  $i$  bids and the even path has  $j$  bids both including the lowest bid. Therefore, we have  $i + j - 1 = n$ . Consider the odd path, from Section 4.3.1, we have that the maximum independent set in this path is  $\{B_i, B_{i-2}, \dots, B_1\}$ . Now consider the even path, again from Section 4.3.1, we have that the maximum independent set from this path is  $\{B_j, B_{j-2}, \dots, B_2\}$ . It is evident that the maximum independent sets derived above do not conflict and the optimal solution for such a bid graph is  $\{\{B_i, B_{i-2}, \dots, B_1\}, \{B_j, B_{j-2}, \dots, B_2\}\}$ .

In cases where there is an odd number of bids  $n$ , it can be observed that both paths in the graph are either odd or even. Assume that one of the paths has  $i$  bids and then the



**Even Number of bids**



**Odd Number of bids**

**Figure 5: One Node With Two Successors**

other path has  $j$ . Therefore, again we have  $i + j - 1 = n$ . When both the paths are even, then the maximum independent sets for the two paths are  $\{B_i, B_{i-2}, \dots, B_2\}$  and  $\{B_j, B_{j-2}, \dots, B_2\}$ . Since both the sets do not have any bids in common, they can be combined to obtain the optimal solution for the entire graph. When both the paths are odd, then the maximum independent sets for the two paths are  $\{B_i, B_{i-2}, \dots, B_1\}$  and  $\{B_j, B_{j-2}, \dots, B_1\}$ . The maximum independent sets derived above conflict with each other and so cannot be combined together as a whole to form the optimal solution. The possible winning solutions are

$$\{\{B_i, B_{i-2}, \dots, B_{i-((i-1)/2)}\}, \{B_j, B_{j-2}, \dots, B_1\}\}$$

or

$$\{\{B_i, B_{i-2}, \dots, B_1\}, \{B_j, B_{j-2}, \dots, B_{j-((j-1)/2)}\}\} \quad (\text{Eq. 1})$$

$$\{\{B_{i-1}, B_{i-3}, \dots, B_{i-1-((i-1)/2)}\}, \{B_j, B_{j-2}, \dots, B_1\}\} \quad (\text{Eq. 2})$$

$$\{\{B_i, B_{i-2}, \dots, B_1\}, \{B_{j-1}, B_{j-3}, \dots, B_{j-1-((j-1)/2)}\}\} \quad (\text{Eq. 3})$$

$$\{\{B_{i-1}, B_{i-3}, \dots, B_{i-1-((i-1)/2)}\}, \{B_{j-1}, B_{j-3}, \dots, B_{j-1-((j-1)/2)}\}\} \quad (\text{Eq. 4})$$

Since the predecessors always have a lower weight than that of the successor bid, we have the following.

$$\{B_i, B_{i-2}, \dots, B_{i-((i-1)/2)}\} > \{B_{i-1}, B_{i-3}, \dots, B_{i-1-((i-1)/2)}\} \quad (\text{Eq. 5})$$

$$\{B_j, B_{j-2}, \dots, B_{j-((j-1)/2)}\} > \{B_{j-1}, B_{j-3}, \dots, B_{j-1-((j-1)/2)}\} \quad (\text{Eq. 6})$$

We can conclude from Equations 5 and 6 that the total value of the bids in Equation 1 is greater than that of the bids in Equation 2, 3 and 4. Therefore the union of the maximal independent sets from the two odd paths gives us the optimal solution.

Now we have that the union of the maximal independent set of the two paths gives us the optimal solution for graphs where all nodes have zero or one predecessors

and successors except for one node with two successors. Since both  $B_i$  and  $B_j$  are in the optimal solution and one of them is the highest weighted bid, we have that the highest weighted bid is in the optimal solution.

The OPCOST values of all the bids in the graph are positive since predecessors always have a lesser weight than the successors and no bid has more than one predecessor. Since all the bids have a positive value, the OPCOST algorithm selects the highest weighted bid in each path and its non-conflicts to be a part of the winning solution. Therefore the OPCOST algorithm always produces the optimal solution for bid graphs where all nodes have zero or one predecessor and successor.

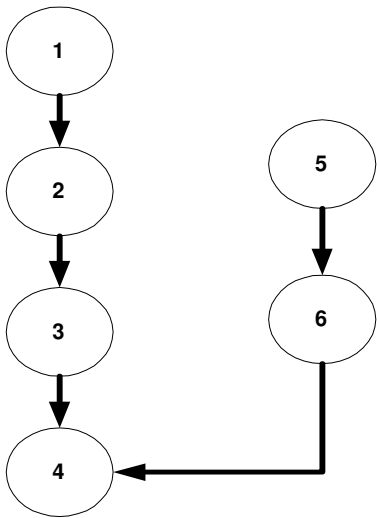
It can be proven using section 4.3.1 that the node with two successors can occur anywhere in the bid graph with even or odd number of nodes but still the conclusions derived above will hold true.

### **4.3.3. One Node With Two Predecessors**

In this case, all the nodes in the bid graph have zero or one predecessor except for one node with two successors. The bid graph can look the example shown in Figure 6.

**Theorem 3:** OPCOST always produces the optimal solution for bid graphs where all nodes have zero or one predecessor and successor except for one node with two predecessors.

**Proof:** Since all nodes except one have only one predecessor, it is evident that the opportunity cost (OPCOST) value for all of the bids in the graph except for the bid with two predecessors is always positive. Assume that the bid or node with two predecessors is the highest weighted bid. The bid graph looks as shown in Figure 6.



Even Number of bids

**Figure 6: One Node With Two Predecessors.**

Consider the two paths in the graph with no nodes ignoring the node with two predecessors and evaluate the OPCOST for the bids on the two paths as before. Let the number of nodes on the two paths be  $i$  and  $j$  respectively. There are two possible winning solutions, one with the node with two predecessors  $n$  as part of the solution and the other without. When  $n$  is part of the winning solution then the winning solution set is

$$\{ \{B_n\}, \{B_{i-1}, B_{i-3}, \dots\}, \{B_{j-1}, B_{j-3}, \dots\} \} \quad (\text{Eq. 7})$$

When  $n$  is not a part of the winning solution then the winning solution set is

$$\{ \{B_i, B_{i-2}, \dots\}, \{B_j, B_{j-2}, \dots\} \} \quad (\text{Eq.8})$$

By opportunity cost principle we have the following

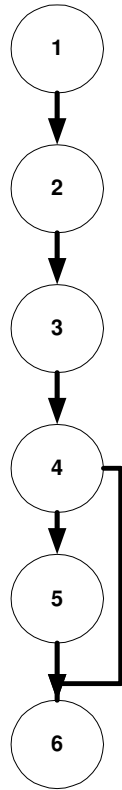
$$\text{value}(i) = \{w_i + w_{i-1} + \dots\} - \{w_{i-1} + w_{i-3} + \dots\} \quad (\text{Eq. 9})$$

$$\text{value}(j) = \{w_j + w_{j-1} + \dots\} - \{w_{j-1} + w_{j-3} + \dots\} \quad (\text{Eq. 10})$$

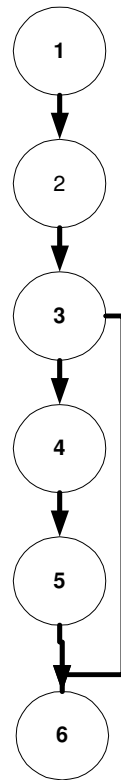
The optimal solution is the set with higher total weight of the sets from Equation 7 and 8. According to the OPCOST algorithm, if  $w_n > \text{value}(i) + \text{value}(j)$  then bid  $B_n$  is in the winning solution. If  $\text{value}(i) + \text{value}(j) > w_n$  then bids  $B_i$  and  $B_j$  are in the winning solution and bid  $B_n$  is not. Therefore, the OPCOST algorithm produces the optimal solution for bid graphs where all nodes have zero or one predecessors except for one node with two predecessors.

#### **4.3.4. One Node With Two Successors; One Node With Two Predecessors**

The scenario dictates that a conflict exists between the node with two successors and the node with two predecessors. Let us assume that the bid with two predecessors is the highest weighted bid. The resultant bid graph looks as shown in Figure 7.



**Example 1**



**Example 2**

**Figure 7: One Node With Two Successors; One Node With Two Predecessors.**

**Theorem 4:** The optimal solution for bid graphs where all nodes have zero or one predecessor and successor except for one node with two predecessors and another with two successors always includes the bid with the highest weight.

**Proof:** Ignoring the conflict between bid with two successors and the node with two predecessors in a graph with odd number of bids we have

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i} + w_{n-i-2} + \dots + w_1 \geq w_{n-1} + w_{n-3} + \dots + w_{n-i+1} + w_{n-i-1} + \dots + w_2 \quad (\text{Eq. 11})$$

Now, when we consider the excluded or ignored conflict, the term  $w_{n-i}$  gets eliminated from the left hand side of the above equation

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i-2} + \dots + w_1 \quad (\text{Eq. 12})$$

and the right hand side of the equation remains the same

$$w_{n-1} + w_{n-3} + \dots + w_{n-i+1} + w_{n-i-1} + \dots + w_2 \quad (\text{Eq. 13})$$

No definite conclusion can be made about the resulting two sides of the above Equation 11. Another solution set can be formed from Equation 12 and Equation 13 as below

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i-1} + \dots + w_2 \quad (\text{Eq. 14})$$

It is evident that Equation 14 > Equation 12 and Equation 14 > Equation 13. Thus Equation 14 is the new maximum independent set.

With even number of bids,

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i} + w_{n-i-2} + \dots + w_2 \geq w_{n-1} + w_{n-3} + \dots + w_{n-i+1} + w_{n-i-1} + \dots + w_1 \quad (\text{Eq. 15})$$

Now, when we consider the excluded or ignored conflict, the term  $w_{n-i}$  gets eliminated from the left hand side of the above equation

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i-2} + \dots + w_2 \quad (\text{Eq. 16})$$

and the right hand side of the equation remains the same

$$w_{n-1} + w_{n-3} + \dots + w_{n-i+1} + w_{n-i-1} + \dots + w_1 \quad (\text{Eq. 17})$$

No definite conclusion can be made about the resulting two sides of the above Equation 15. Another solution set can be formed from Equation 16 and Equation 17 as below

$$w_n + w_{n-2} + \dots + w_{n-i+2} + w_{n-i-1} + \dots + w_1 \quad (\text{Eq .18})$$

It is evident that Equation 18 > Equation 16 and Equation 18 > Equation 17. Thus Equation 18 is the new maximum independent set. Also, it can be observed from Equation 14 and Equation 18 that the highest weighted bid  $B_n$  is always in the optimal solution.

#### 4.3.5. Double Counting in Opportunity Cost Algorithm

The OPCOST algorithm does not always produce the optimal solution for graphs where all nodes have zero or one predecessors and successors except for one bid with two successors and another with two predecessors. The OPCOST produces the optimal solution in graphs where there is odd number of bids between the node with two successors and the node with two predecessors. When there is even number of bids between the node with two predecessors and the node with two successors, the OPCOST algorithm outputs a less than optimal solution. The example illustrated in Figure 8 is used to demonstrate one of the simple scenarios where the Opportunity Cost algorithm deviates from the optimal.

In the example illustrated in Figure 9,

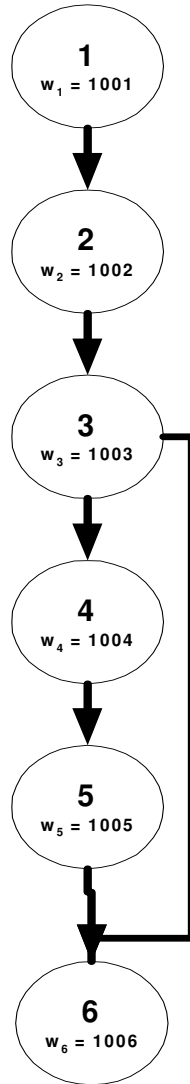
$$v_1 = w_1 = 1001;$$

$$v_2 = w_2 - \max(0, v_1) = w_2 - w_1 = 1;$$

$$v_3 = w_3 - \max(0, v_2) = w_3 - w_2 + w_1 = 1002;$$

$$v_4 = w_4 - \max(0, v_3) = w_4 - w_3 + w_2 - w_1 = 2;$$

$$v_5 = w_5 - \max(0, v_4) = w_5 - w_4 + w_3 - w_2 + w_1 = 1003;$$



**Figure 8: Bid Graph With Double Counting**

$$v_6 = w_6 - \max(0, v_5) - \max(0, v_3) = w_6 - w_5 + w_4 - w_3 + w_2 - w_1 - w_3 + w_2 - w_1 = -999;$$

Where  $v_n \Rightarrow \text{value}(n)$ . Since  $v_6$  is negative the opportunity cost algorithm skips  $B_6$  and selects  $\{B_5, B_3, B_1\}$  to be the winning solution. But it has been proved in Section 4.3.4 that for the above problem the highest weighted bid is always in the solution. Observing the equation to calculate  $v_6$ , we see that the weights of  $B_3$  and  $B_1$  have been subtracted twice in the same equation while the weight of  $B_2$  has been added twice. In this case, the double counting of the weight values of certain nodes had lead to the non-optimality of the independent set generated by the opportunity cost algorithm. As the problem size grows and the conflict graph gets more interwoven, the double counting phenomenon actually helps the OPCOST algorithm reach better solutions. But eliminating double counting does not lead to a better solution all the time. The double counting phenomenon either helps or defeats the approximation ratio achieved by the opportunity cost algorithm depending on the problem structure or the topology of the bid graph. When an algorithm eliminating the double counting was run on the sample dataset described in Section 5, only a few problems resulted in a better solution than the OPCOST but for most problems the results obtained were much worse than the OPCOST results.

#### **4.3.6. Effect Of Node Removal**

After the opportunity cost is calculated for each bid in the graph, the OPCOST algorithm uses the OPCOST values for each bid as a decision variable. All the bids with negative values are not considered and only bids with positive values are evaluated. The first positive bid in the descending order of bid amount is included to be a part of the winning solution. The list is then traversed down and all positive value bids that do not conflict with the current winning solution set are added to be a part of the winning solution. The

result is a set of independent bids that form the winning solution. As each bid gets added to the winning solution, their conflicts are eliminated out of contention, that is, they cannot be a part of the winning solution anymore. But the OPCOST algorithm ignores the fact and uses the OPCOST values that are influenced by the eliminated bids to decide whether a bid is part of the winning solution or not. Removing the bid that is selected to be part of the winning solution and its conflicts from the entire bid list reduces the number of bids that are still under consideration. Recalculating the OPCOST values for the remaining bids updates the values to reflect only the current list of bids and eliminates the effect of the discarded bids. This updated OPCOST value serves as a better decision variable than the old values and improves the quality of the winning solution considerably. The reduced number of bids on which the OPCOST algorithm is executed as the winning bids are selected, results in smaller graphs and reduces chances of double counting errors.

#### **4.4. Recursive Recalculation in OPCOST (OPCOST-R)**

This research introduces one of the suggested modifications to the opportunity cost algorithm. The recursive recalculation method involves rerunning the opportunity cost algorithm on the remaining bids after the first bid is selected to be a part of the solution set and eliminating itself and its predecessors from the bid graph. Although the recursive recalculation involves more calculations, the algorithm is still polynomial in time. The total time of execution for the opportunity cost algorithm has been shown in Section 4.1 to be  $O(|B_0| + |E_0|)$ . Therefore, the total time of execution for the opportunity cost algorithm with recalculation is less than  $O(|B_0| * (|B_0| + |E_0|))$ . In practice the time for execution is never this high because the selected bid and its conflicts get discarded after

each selection. This reduces the number of bids in contention with each selection. The conflict density of the graph also has a great influence on the execution time. The time for execution decreases as the conflict density of the graph increases. The algorithm for the recursive recalculation proceeds as listed below.

Given a directed acyclic graph  $G_0 = (B_0, E_0)$  with weights  $weight(v)$  for each  $B_0$ , the opportunity cost algorithm with recalculation is initialized with an empty set to be the winning independent set  $V$  and proceeds as follows:

ROC1 If the input graph  $G_0$  has no nodes, go to step ROC 6.

ROC2 Traverse the nodes according to the topological order of  $G_0$ . Compute  $value(u)$  for each node  $u$ .  $value(u)$  represents an estimate of the gain we expect by including  $u$  in the independent set. The  $value(u)$  is computed by taking the weight of  $u$  and subtracting an opportunity cost which consists of the values of earlier positive-value nodes that conflict with  $u$ .

$$value(u) = weight(u) - \sum_{v \rightarrow u} \max(0, value(v))$$

ROC3 Processing the nodes in the reverse order, add the first node  $u$  with a non-negative value to the set  $V$  i.e.,  $V = V \cup \{u\}$

ROC4 Delete  $\{u\}$  and all predecessors of  $\{u\}$  from the bid graph  $G$  resulting in a new bid graph  $G_1$  i.e.  $G_1 = G_0 - \{u\} - P(u)$ .

ROC5 Go to step ROC1 with  $G_1$  as the input graph.

ROC6 Return  $V$ .

Consider the example illustrated in Figure 9. The bid graph shown in the example has the same structure as the example shown in Figure 3. But the bid amount associated

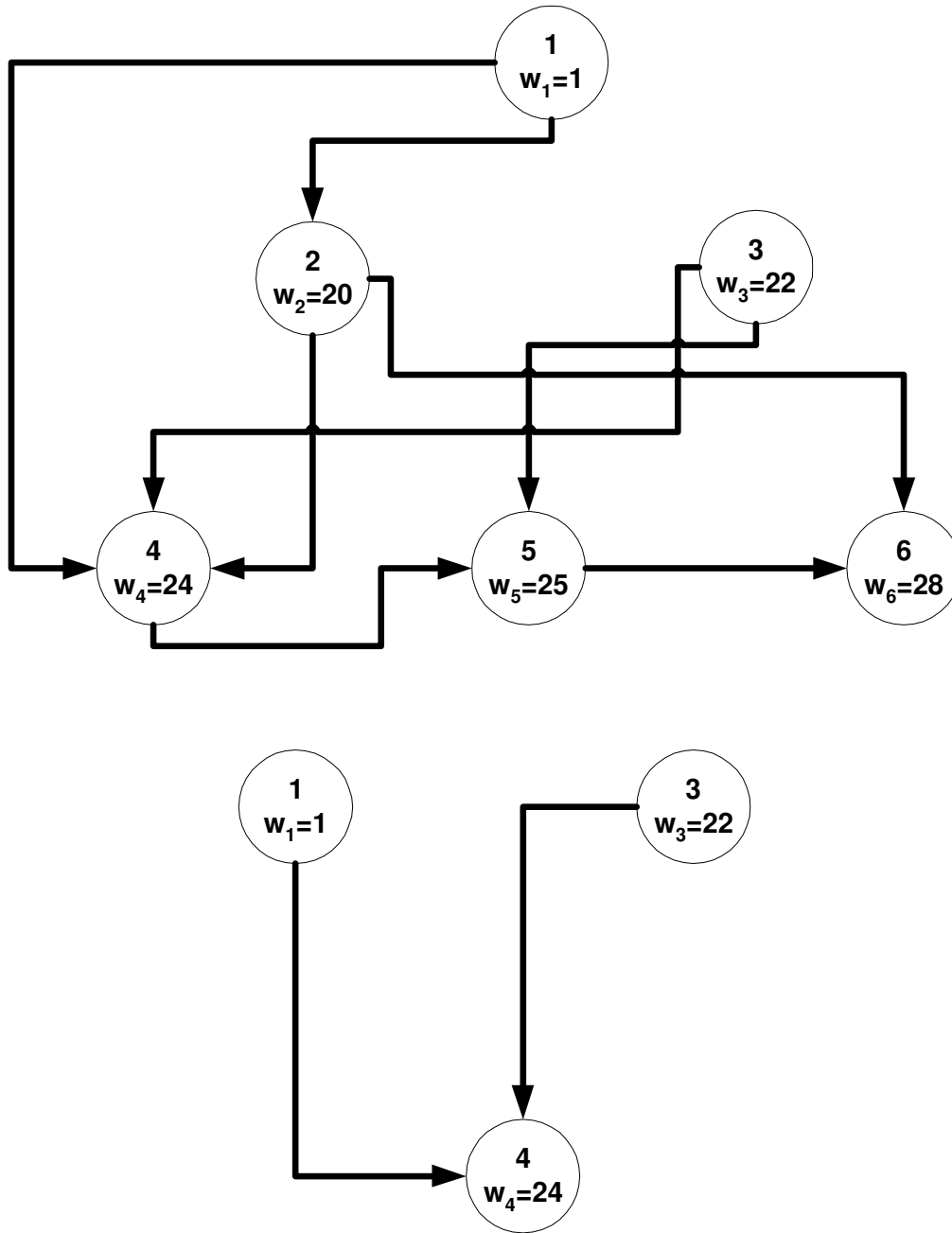


Figure 9: Opportunity Cost Recalculation Example

with each bid is different from the example in Figure 3. The opportunity cost calculation for bid is shown below.

$$v_1 = w_1 = 1;$$

$$v_2 = w_2 - v_1 = 20 - 1 = 19;$$

$$v_3 = w_3 = 22;$$

$$v_4 = w_4 - v_1 - v_2 - v_3 = 24 - 1 - 19 - 22 = -18;$$

$$v_5 = w_5 - v_4 - v_3 = 25 - 22 = 3;$$

$$v_6 = w_6 - v_5 - v_2 = 28 - 3 - 19 = 6;$$

The winning solution according to OPCOST algorithm is  $\{6, 3, 1\}$ . The total revenue of the OPCOST winning solution is 51. But the maximum total revenue that can be obtained from the problem in Figure 9 is 52 from the winning set  $\{6, 4\}$  and not 51 from the set  $\{6, 3, 1\}$ . Applying the OPCOST-R algorithm on the same problem we get the same opportunity cost values for all the bids as the OPCOST before any bid is chosen to be in the winning set. So, again by OPCOST-R, the bid 6 is added to winning set. But once the bid 6 is added to the winning solution, the OPCOST-R recalculates the opportunity cost values for the remaining bids in the problem excluding bid 6 and its conflicts. The remaining bid graph after choosing bid 6 is shown in Figure 9. Now the new opportunity cost values for the remaining bids is calculated as shown below.

$$v_1 = w_1 = 1;$$

$$v_3 = w_3 = 22;$$

$$v_4 = w_4 - v_1 - v_3 = 24 - 22 - 1 = 1;$$

Since bid 4 has a positive value, it is selected to be a part of the winning solution. Adding bid 4 to the winning solution and eliminating its conflicts leaves no remaining bids; thus

the winning set by the OPCOST-R algorithm is  $\{6, 4\}$  with a total revenue of 52 which is the same as the maximum total revenue for the problem and also better than the result produced by the OPCOST algorithm.

#### **4.5. Maximum Total Revenue Using OPCOST (MTR)**

This section introduces another method considered for calculating the winning bids in a combinatorial auction. The new method uses the opportunity cost algorithm to find out the maximum independent set for each node in the bid graph that includes the current node itself. The MTR algorithm is based on the principle that the opportunity cost algorithm delivers better results for smaller graphs. The approximation ratio of the OPCOST algorithm is based on  $\beta$ . Thus reducing the size of the graph on which the OPCOST algorithm operates also reduces  $\beta$  thereby increasing the accuracy of the OPCOST algorithm. The MTR algorithm uses the OPCOST algorithm to find out the maximal independent set among the non-conflicts for each bid. The size of the graph for only the non-conflicts of each bid is less than that of the entire graph and thereby the resultant set is closer to the optimal. If OPCOST calculates the optimal solution to the maximal independent set among the non-conflicts for each bid, then MTR will produce an optimal solution. This increased accuracy in calculating the MTR for each bid results in better decision variable values when selecting the winning solution. The bid graph  $G$  in this case is just an undirected graph with no specific orientation. The set of all the conflicting bids to the bid  $u$  is represented as  $C(u)$  and the set of all non-conflicting bids to the bid  $u$  is represented as  $C'(u)$ . The maximal independent set for each node including the node itself is obtained by applying the opportunity cost algorithm to only the non-conflict bids of the current node  $C'(u)$ . The value  $value(u)$  for each node now becomes

the total weight of the maximum independent set including  $u$ . Once the  $\text{value}(u)$  for all the nodes have been calculated, then the node with the highest  $\text{value}(u)$  is selected to be part of the winning solution. The selected bid and its conflicts are then eliminated from the bid graph and the  $\text{value}(u)$  for each node calculated again. The process is repeated till the bid graph  $G$  is empty. Although this method significantly increases the total time of execution, it may be justified by the improved solution quality.

Given an undirected graph  $G = (B, E)$  with weights  $\text{weight}(v)$  for each  $B$ , the method is initialized with an empty set to be the winning independent set  $V$  and proceeds as follows:

- NOC1 If the input graph  $G$  has no nodes, go to step NOC 6.
- NOC2 Traverse the nodes according to the topological order of  $G_0$ . Compute  $\text{value}(u)$  for each node  $u$ .  $\text{value}(u)$  represents an estimate of the total revenue we expect by including  $u$  in the independent set. The set of non-conflicting bids for  $u$  given by  $C'(u)$  is oriented in the order of increasing weights to obtain the directed acyclic graph  $C_0'(u)$ . The  $\text{value}(u)$  is computed by adding the weight of  $u$  to the maximum independent set returned by  $\text{OPCOST}(C_0'(u))$ .
- NOC3 Add the node  $u$  with the highest  $\text{value}(u)$  to the set  $V$  i.e.,  $V = V \cup \{u\}$
- NOC4 Delete  $\{u\}$  and all conflicts of  $\{u\}$  i.e.,  $C(u)$  from the bid graph  $G$  resulting in a new bid graph  $G_1$  i.e.  $G_1 = G - \{u\} - C(u)$ .
- NOC5 Go to step NOC1 with  $G_1$  as the input graph.
- NOC6 Return  $V$ .

## **Chapter 5. Results**

As a part of this research, a number of experiments have been performed on a C# implementation of the algorithms. The original opportunity cost algorithm and the suggested improvements have been implemented and the solution quality and the execution time have been recorded to compare the performance of the variations. The algorithms were tested on a large data set provided by De Vries and Vohra [14].

### **5.1. Problem Dataset**

The problem dataset provided by De Vries and Vohra uses five different distributions, where each distribution consists of sequence of problem sizes, varying both the number of bids as well as the number of items. The problem size is defined as the total number of items in bids, which is different from the product of the number of items with the number of bids. Many replications or problems are available for each combination of the number of items and the number of bids. The results for the algorithms have individually reported for each of the distributions separately since the type of distribution can affect the time of execution and the solution quality.

More problems with increasing number of bids were formed from various combinations of the random distribution datasets provided by De Vries and Vohra. These bigger problems had the number of bids increasing in steps of thousands from 2000 to 8000 to study the performance of all the discussed algorithms with respect to solution quality and time of execution.

### **5.2. Types of Distribution in Dataset**

The problem dataset is characterized by four different classes of distributions. The number of items and the number of bids and their values were chosen based on the

distribution. The distribution type and the variations of the problem instances are explained in the following sections. The results for problems within each distribution type have been recorded and compared for the different algorithms. The effects of the number of items and the number of bids in a problem on the solution quality and the time of execution are also plotted and studied.

### **5.2.1 Random**

For each bid, the number of items is randomly selected from  $1 \dots m$ . The selected number of items is randomly picked without replacement. The bid values are randomly selected from  $[0, 1]$ . The dataset contains 240 problems of this type. The values in Table 1 show the combinations of the number of items and the number of bids present in the dataset along with the number of problem instances for each combination.

### **5.2.2 Weighted Random**

For each bid, the number of items is randomly selected from  $1 \dots m$ . The selected number of items is randomly picked without replacement. The bid values are randomly selected from  $[0, \text{number of items in bid}]$ . The dataset contains 320 problems of this type. The values in Table 3 show the combinations of the number of items, the number of bids and the number of items in each bid present in the dataset along with the number of problem instances for each combination.

### **5.2.3 Uniform**

For each bid, the number of items is randomly selected from  $1 \dots m$ . The selected number of items is randomly picked without replacement. The bid values are randomly selected from  $[0, 1]$ . The dataset contains 660 problems of this type. The values in Table 4 show the combinations of the number of items, the number of bids and the number of items in

each bid present in the dataset along with the number of problem instances for each combination.

#### **5.2.4 Decay**

For each bid, one random item is added. Then a new random item is repeatedly added with a chosen probability  $\alpha$  until an item isn't added or the bid contains all  $m$  items. The bid values are randomly selected from  $[0, \text{number of items in bid}]$ . The dataset contains 1040 problems of this type. The values in Table 5 show the combinations of the number of items, the number of bids and the decay value as a percentage value present in the dataset along with the number of problem instances for each combination.

### **5.3 Experimental Results**

This section produces the results of the opportunity cost algorithm, opportunity cost algorithm with recalculation and the Maximum Total Revenue algorithm for the dataset described above. All problems were solved using a computer running Microsoft Windows XP operating system and equipped with a Pentium 3.06GHZ processor with 1 GB of RAM. For each of the algorithms the result and the time of execution were recorded and compared.

#### **5.3.1 Random Distribution Results**

The OPCOST, OPCOST-R and MTR algorithms were run on all the 240 problem datasets available for the random distribution class. The OPCOST with recalculation performed consistently better than the OPCOST algorithm although occasionally the OPCOST solution was better than the OPCOST-R solution. The OPCOST-R algorithm produced better results for 199 problems out of the 240 and similar results for 18 problems. OPCOST did better than the OPCOST-R for the remaining 23 problems out of

the 240. Table 1 summarizes the results of the OPCOST and OPCOST-R to present a better understanding of the performances of the OPCOST-R algorithm. The table also presents the number of cases where the OPCOST-R was better, worse and the same as the OPCOST algorithm. The average increase in execution time for OPCOST-R was 4.07%. On the other hand, the algorithm produced solutions that were on an average 11.34% better than the OPCOST results.

The MTR algorithm produced better or equal results to the OPCOST algorithm for all the 240 problems in the random distribution dataset. MTR outputted better results than the OPCOST algorithm for 228 problems out of the 240 and produced the same results as the OPCOST for the remaining 12 cases. The MTR solutions were on an average 15.48% better than the OPCOST results. Table 2 summarizes the MTR results with respect the OPCOST results with varying number of items and bids.

### **5.3.2 Weighted Random Results**

The results generated by OPCOST, OPCOST-R and the MTR algorithms on all of the 320 weighted random distribution class problems were identical. The weighted random distribution problems have a structure similar to that of the random distribution but the weights of associated with each bid is proportional to the number of items in the bid. The results show that the solution quality depends not only on the structure of the problem but also on the weights or the bid amounts associated with each bid. The results produced by all algorithms were identical irrespective of the different combinations of the varying number of bids and items in the problem. Table 3 summarizes the OPCOST and OPCOST-R results for the weighted random distribution class of problems with varying number of items and bids.

**Table 1: Random Distribution Class Problem Results For OPCOST-R**

Items	Bids	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R	No. Of problems with better results OPCOST-R
100	1000	20	18.64	0	1	19
100	500	20	14.04	1	0	19
100	750	20	13.76	3	1	16
200	1000	20	11.89	2	0	18
200	500	20	14.35	1	1	18
200	750	20	8.32	4	0	16
300	1000	20	12.93	2	0	18
300	500	20	7.93	2	4	14
300	750	20	10.15	2	2	16
400	1000	20	9.28	5	1	14
400	500	20	5.55	0	7	13
400	750	20	9.31	1	1	18

**Table 2: Random Distribution Class Problem Results For MTR**

Items	Bids	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R	No. Of problems with better results OPCOST-R
100	1000	20	23.28	0	0	20
100	500	20	16.89	0	1	19
100	750	20	21.48	0	1	19
200	1000	20	14.73	0	0	20
200	500	20	18.14	0	0	20
200	750	20	12.24	0	2	18
300	1000	20	15.9	0	0	20
300	500	20	9.88	0	3	17
300	750	20	18.47	0	1	19
400	1000	20	13.18	0	1	19
400	500	20	9.52	0	3	17
400	750	20	12.09	0	0	20

**Table 3: Weighted Random Distribution Class Problem Results For OPCOST-R**

Items	Bids	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R
100	500	20	0	0	20
100	1000	20	0	0	20
100	1500	20	0	0	20
100	2000	20	0	0	20
200	500	20	0	0	20
200	1000	20	0	0	20
200	1500	20	0	0	20
200	2000	20	0	0	20
300	500	20	0	0	20
300	1000	20	0	0	20
300	1500	20	0	0	20
300	2000	20	0	0	20
400	500	20	0	0	20
400	1000	20	0	0	20
400	1500	20	0	0	20
400	2000	20	0	0	20

### **5.3.3 Uniform Distribution Results**

The results for all of the 660 problems of the uniform distribution class were identical for the OPCOST, OPCOST-R and the MTR algorithms. The results produced by all algorithms were identical irrespective of the different combinations of the varying number of bids and items in the problem. The uniform distribution problems have a smaller number of items in each bid when compared to the random distribution problems, leading it to reduce the number of conflicts that exist between the bids. The reduced number of conflicts causes all the algorithms to output the same results. Table 4 summarizes the OPCOST and OPCOST-R results for the uniform random distribution class of problems which remain the same in the case of MTR results for the problems.

### **5.3.4 Decay Distribution Results**

The dataset for the random distribution class consisted of 1040 problems. The OPCOST-R algorithm performed better than the OPCOST algorithm for 589 problems out of the 1040 and the results were identical for 404 problems. The results for the OPCOST algorithm were better than the OPCOST-R algorithm for 47 out of the 1040 problems. Table 5 and 6 summarizes the results of the OPCOST and OPCOST-R to present a better understanding of the performances of the OPCOST-R algorithms. The MTR algorithm on the other hand always produced equal or better results than the OPCOST algorithm for all the 1040 problems with varying number of items and bids. Table 7 and 8 summarizes the results of the MTR algorithm against the OPCOST results for the decay distribution class of problems. In decay distribution problems, the number of items in a bid increases with  $\alpha$  thus increasing the number of conflicts. It can be observed from the tables that for both the OPCOST-R and the MTR algorithm, the solution quality increases with  $\alpha$ .

**Table 4: Uniform distribution Class Problem Results For OPCOST-R**

Items	Bids	Alpha %	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R	No. Of problems with better results OPCOST-R
25	50	3	20	0	0	20	0
25	100	3	20	0	0	20	0
25	150	3	20	0	0	20	0
50	50	3	20	0	0	20	0
50	100	3	20	0	0	20	0
50	150	3	20	0	0	20	0
75	50	3	20	0	0	20	0
75	50	8	20	0	0	20	0
75	50	13	20	0	0	20	0
75	100	3	20	0	0	20	0
75	100	8	20	0	0	20	0
75	100	13	20	0	0	20	0
75	150	3	20	0	0	20	0
75	150	8	20	0	0	20	0
75	150	13	20	0	0	20	0
100	50	3	20	0	0	20	0
100	100	3	20	0	0	20	0
100	150	3	20	0	0	20	0
100	500	3	20	0	0	20	0
100	600	3	20	0	0	20	0
100	700	3	20	0	0	20	0
100	750	3	20	0	0	20	0
100	800	3	20	0	0	20	0
100	850	3	20	0	0	20	0
100	900	3	20	0	0	20	0
100	950	3	20	0	0	20	0
100	1000	3	20	0	0	20	0
100	1050	3	20	0	0	20	0
100	1100	3	20	0	0	20	0
100	1150	3	20	0	0	20	0
100	1200	3	20	0	0	20	0
100	1250	3	20	0	0	20	0
100	1300	3	20	0	0	20	0

**Table 5: Decay Distribution Class Problem Results For OPCOST-R**

Items	Bids	Alpha %	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R	No. Of problems with better results OPCOST-R
50	50	55	20	2.91	1	7	12
50	100	55	20	9.48	1	0	19
50	150	55	20	5.74	2	1	17
50	200	55	20	6.06	1	2	17
100	50	5	20	0	0	20	0
100	50	15	20	0.04	0	19	1
100	50	25	20	0.17	0	18	2
100	50	35	20	0.26	0	14	6
100	50	45	20	2.75	0	8	12
100	50	55	20	1.12	1	11	8
100	50	65	20	4.12	0	8	12
100	50	75	20	7.75	1	4	15
100	50	85	20	8.46	0	5	15
100	50	95	20	0	0	20	0
100	100	5	20	0.01	0	19	1
100	100	15	20	0.11	0	15	5
100	100	25	20	0.26	1	13	6
100	100	35	20	2.93	0	8	12
100	100	45	20	1.86	1	5	14
100	100	55	20	4.32	4	0	16
100	100	65	20	6.55	0	2	18
100	100	75	20	9.33	1	1	18
100	100	85	20	8.01	1	2	17
100	100	95	20	0	0	20	0
100	150	5	20	0.02	0	17	3
100	150	15	20	0.17	0	13	7
100	150	25	20	1.4	4	8	8
100	150	35	20	1.79	0	7	13
100	150	45	20	2.5	1	6	13
100	150	55	20	5.48	2	2	16
100	150	65	20	11.18	0	0	20
100	150	75	20	9.77	3	1	16
100	150	85	20	12.91	0	2	18
100	150	95	20	0	0	20	0
100	200	5	20	0.02	0	19	1
100	200	15	20	0.2	0	14	6
100	200	25	20	2.14	0	10	10
100	200	35	20	3.04	1	4	15
100	200	45	20	5.22	2	2	16

**Table 6: Decay Distribution Class Problem Results For OPCOST-R (contd.)**

Items	Bids	Alpha %	Replications	Average increase in solution quality OPCOST-R %	No. Of problems with worse results OPCOST-R	No. Of problems with identical results OPCOST-R	No. Of problems with better results OPCOST-R
100	200	55	20	3.79	3	0	17
100	200	65	20	9.91	4	0	16
100	200	75	20	14.18	0	0	20
100	200	85	20	12.3	0	1	19
100	200	95	20	0.07	0	18	2
150	50	55	20	2.05	1	13	6
150	100	55	20	2.49	1	4	15
150	150	55	20	5.54	1	1	18
150	200	55	20	5.99	3	2	15
200	50	55	20	1.71	2	5	13
200	100	55	20	2.37	1	9	10
200	150	55	20	3.67	1	3	16
200	200	55	20	5.59	2	1	17

**Table 7: Decay Distribution Class Problem Results For MTR**

Items	Bids	Alpha %	Replications	Average increase in solution quality MTR %	No. Of problems with worse results MTR	No. Of problems with identical results MTR	No. Of problems with better results MTR
50	50	55	20	4.22	0	7	13
50	100	55	20	13.67	0	0	20
50	150	55	20	9.41	0	1	19
50	200	55	20	10.18	0	1	19
100	50	5	20	0.08	0	19	1
100	50	15	20	0.05	0	18	2
100	50	25	20	0.17	0	17	3
100	50	35	20	0.89	0	11	9
100	50	45	20	3.25	0	8	12
100	50	55	20	1.51	0	10	10
100	50	65	20	5.17	0	7	13
100	50	75	20	8.98	0	1	19
100	50	85	20	12.77	0	3	17
100	50	95	20	0.03	0	19	1
100	100	5	20	0.01	0	19	1
100	100	15	20	0.22	0	14	6
100	100	25	20	0.35	0	14	6
100	100	35	20	3.28	0	6	14
100	100	45	20	4.96	0	4	16
100	100	55	20	5.68	1	0	19
100	100	65	20	9.19	0	1	19
100	100	75	20	14.98	0	1	19
100	100	85	20	16.15	0	2	18
100	100	95	20	0	0	20	0
100	150	5	20	0.03	0	16	4
100	150	15	20	0.21	0	12	8
100	150	25	20	2.32	0	10	10
100	150	35	20	2.58	0	6	14
100	150	45	20	3.75	0	4	16
100	150	55	20	8.73	0	1	19
100	150	65	20	15.13	0	0	20
100	150	75	20	13.39	0	1	19
100	150	85	20	23.37	0	1	19
100	150	95	20	0	0	20	0
100	200	5	20	0.09	0	17	3
100	200	15	20	0.29	0	12	8
100	200	25	20	2.96	0	8	12
100	200	35	20	4.59	0	2	18
100	200	45	20	8.74	0	1	19

**Table 8: Decay Distribution Class Problem Results For MTR (contd.)**

Items	Bids	Alpha %	Replications	Average increase in solution quality MTR %	No. Of problems with worse results MTR	No. Of problems with identical results MTR	No. Of problems with better results MTR
100	200	55	20	6.99	0	1	19
100	200	65	20	15.91	0	1	19
100	200	75	20	17.49	0	0	20
100	200	85	20	21.63	0	0	20
100	200	95	20	0.3	0	18	2
150	50	55	20	2.46	0	12	8
150	100	55	20	4.05	0	3	17
150	150	55	20	7.32	0	1	19
150	200	55	20	8.33	0	1	19
200	50	55	20	2.73	0	6	14
200	100	55	20	3.22	0	6	14
200	150	55	20	5.11	0	2	18
200	200	55	20	7.32	0	1	19

### **5.3.5. Results On Bigger Problems**

The dataset for the bigger problems belongs to the random distribution class and was formed by combining various problems of the random distribution class to obtain problems with larger number of bids. The problems have the number of bids in the problems ranging from 2000 to 8000 and increasing in steps of thousand. The number of items in the problem was also varied along with number of bids from 100 to 300 in steps of 100. The time of execution for the problems were recorded and compared both as a factor of the number of bids and the number of items. The time of calculation for OPCOST-R was observed to be not significantly higher than the OPCOST time for the larger problems and the rate of increase of the execution time remained similar to that of the OPCOST algorithm. But the OPCOST-R algorithm always produced much better results for larger problems than the smaller problems provided De Vries and Vohra where OPCOST-R sometimes produced worse results than OPCOST. The MTR took significantly more time than both the OPCOST and the OPCOST-R algorithms and the execution time increased rapidly with the number of bids in the problem but again the algorithm always produced significantly better results than OPCOST algorithm. Table 9 summarizes the solution quality of the OPCOST-R and MTR algorithm results compared to the OPCOST algorithm for the bigger problems dataset. It can be observed that the solution quality for both the MTR and the OPCOST-R compared to the OPCOST algorithm has significantly increased for the bigger problems. Table 10 summarizes the execution time for the OPCOST, OPCOST-R and MTR algorithm with varying number of bids and items in the problems.

**Table 9: Results For Bigger Problems – Solution Quality**

Average increase in solution quality % vs. OPCOST							
Number of Items							
		100		200		300	
		OPCOST-R	MTR	OPCOST-R	MTR	OPCOST-R	MTR
Number of Bids	1000	18.64	23.28	11.89	14.73	12.93	15.9
	2000	21.07	22.81	19.77	23.55	16.3	20.63
	3000	21.46	23.52	22.32	20.96	21.59	25.28
	4000	24.02	22.04	19.08	18.27	17.29	18.13
	5000	15.96	19.52	16.50	15.29	18.04	18.88
	6000	31.09	32.98	20.36	20.05	21.34	19.27
	7000	16.31	18.49	11.53	14.31	17.47	17.47

**Table 10: Results For Bigger Problems – Execution Time**

Average time for execution										
Number of Items										
		100			200			300		
		OPCOST	OPCOST-R	MTR	OPCOST	OPCOST-R	MTR	OPCOST	OPCOST-R	MTR
Number of Bids	1000	2.49	2.75	10.48	4.20	4.42	13.82	5.81	5.95	16.98
	2000	10.05	12.36	69.31	17.41	18.70	83.82	23.65	25.66	95.08
	3000	24.39	28.92	227.5	40.10	46.04	260.72	55.01	62.69	289.19
	4000	42.47	54.55	547.81	76.46	95.65	773.72	115.50	136.99	606.62
	5000	65.16	91.45	966	141.32	136.87	1028.24	187.06	196.07	1168.47
	6000	94.07	110.91	1673.66	179.31	230.42	1870.9	253.47	313.27	2044.92
	7000	129.06	190.90	2679.25	286.74	351.07	2859.54	351.75	434.92	2925.35
	8000	171.98	344.22	3976.31	280.89	319.02	4177.16	493.5	541.625	4386.67

## Chapter 6. Analysis

This section analyses the results of the OPCOST, OPCOST-R and the MTR algorithms on the various different datasets. Trends in solution quality and time for execution are observed with respect to the number of items and the number of bids in the problem.

### 6.1. Solution Quality With Respect to Optimal

The difference between the optimal solution and the solution produced by the approximate algorithm is an important factor when evaluating the performance of the algorithm. This section compares the quality of the solution produced by the OPCOST, OPCOST-R and the MTR compared to the actual optimal solution of the problems. The optimal solution was obtained by a depth-first search on the problem. The problem dataset used for the depth- first search algorithm had problems containing 100 items and 50, 100 or 150 bids with 60 replications for each combination. Figure 10 shows the solution quality of the OPCOST, OPCOST-R and MTR algorithms with respect to the optimal solution on problems with 50, 100 and 150 bids. The X-axis is the various problem instance numbers ordered by size. The Y-axis is the percentage of the optimal solution outputted by the algorithms for the various problem instances. The X-axis is the problem number where the problems are ordered by size and the Y-axis is the percentage of the solution to the optimal solution. Table 11 summarizes the results of the algorithms with respect to the optimal solution. All algorithms produce acceptable quality solutions for the smaller problems compared to the optimal solution and also the variations in the solution quality are quite random. It can be observed from Figure 10 and Table 11 that as the number of bids in the problem increases, the solution quality of all three algorithms begins to degrade. The MTR algorithm produced optimal results for all except one

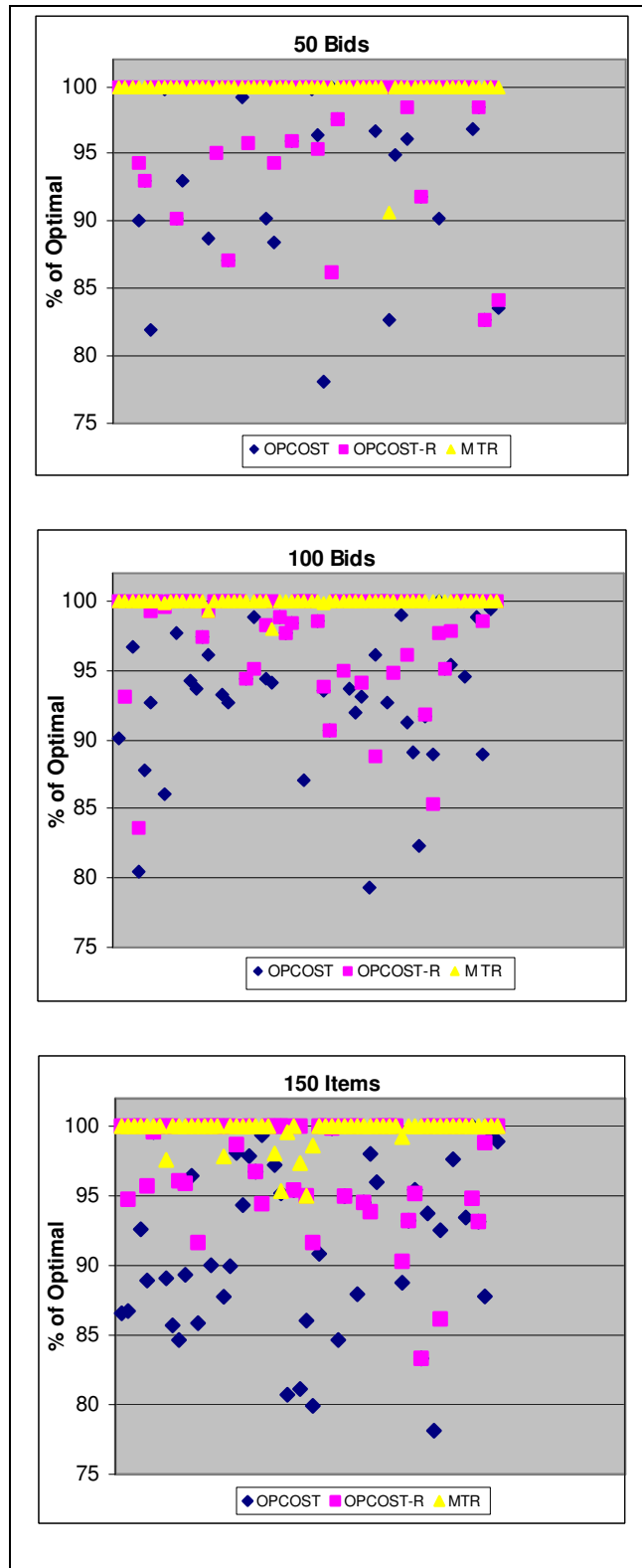


Figure 10: Solution Quality By Problem Size

**Table 11: Comparison With The Optimal Solution**

Average Percentile of the optimal solution			
	OPCOST	OPCOST-R	MTR
50 Items	96.38	98.008	99.84
100 Items	95.10	97.88	99.96
150 Items	93.29	97.73	99.64

problem with 50 bids and with 100 and 150 bids the MTR algorithm produced less than optimal results 5 and 11 times respectively out of 60 problems. The OPCOST-R produced less than optimal results 16, 26 and 26 times and the OPCOST 26, 43 and 47 times with 50, 100 and 150 items respectively. Thus it the OPCOST algorithm delivers close to optimal solutions for smaller problems and the solution quality begins to decrease as the size of the problem increases. The MTR remains close to optimal for smaller problems and the solution quality decreases as the size of the problem increases. Also MTR performs much better than OPCOST and OPCOST-R for the smaller problems. OPCOST-R produces better results than the OPCOST but is typically not as close to the optimal solution as MTR.

## **6.2. Solution Quality On Distributions**

The OPCOST-R and the MTR algorithms output the same results as the OPCOST algorithm for all of the uniform and weighted distribution sets with 660 and 320 problems respectively. But for the random and decay distributions the OPCOST-R and the MTR algorithms performed better then the OPCOST algorithms. For the random distribution dataset containing 240 problems of varying bids and items size the solution quality increased on an average by 11.34% for the OPCOST-R and by 15.48% for the MTR algorithm. In the case of the decay distribution dataset containing 1040 problems of varying items and bids size, the solution quality increases on an average by 3.99% and 6.04% for the OPCOST-R and MTR algorithms respectively. Figure 11 shows the performance of the OPCOST-R and the MTR algorithms against the OPCOST results. The X-axis is the problem number in increasing order and the Y-axis is the percentage of the solution produced by the algorithm to that of the OPCOST solution. The OPCOST-R

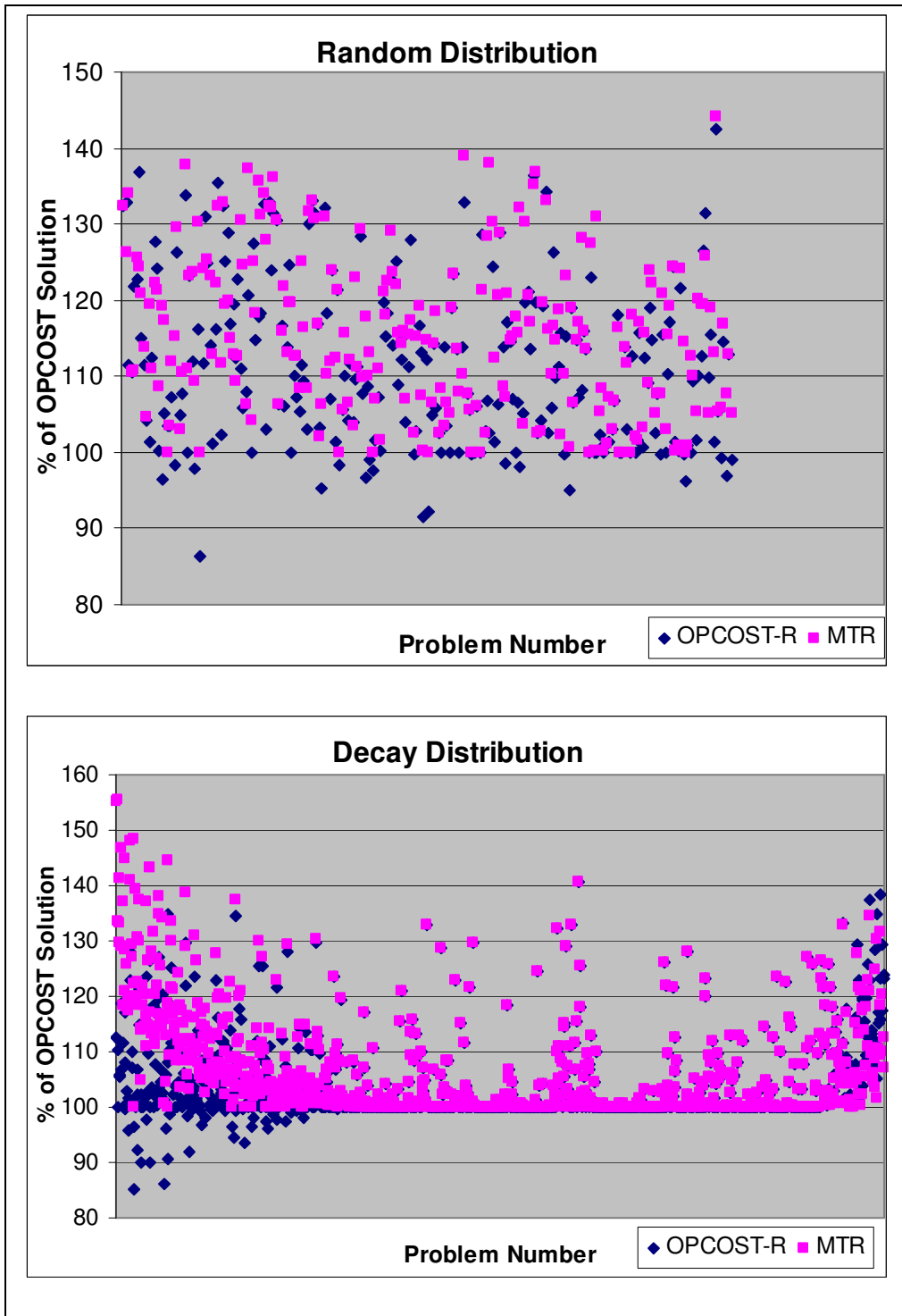


Figure 11: Solution Quality By Distribution Class

performed consistently better than the OPCOST algorithm on random distribution datasets for most problems but produced results lesser results than the OPCOST algorithm for 23 problems out of 240 and on the decay distribution dataset it produced less optimal results than the OPCOST algorithm for 46 problems out of 1040. It can be observed from Figure 11 although the OPCOST-R sometimes outputs lesser results than the OPCOST as the problem size gets larger OPCOST-R always performs better than OPCOST. In the cases of the uniform and the weighted distribution the OPCOST-R algorithm outputted the same results for all problems in the dataset. The MTR algorithm on the other hand always produced solutions that are better than the OPCOST algorithm for all distribution types including random and decay distributions.

### **6.3. Execution Time**

Figure 12 and 13 shows the execution time for OPCOST, OPCOST-R and the MTR against increasing number of bids and items in the problem respectively. In the figure the X-axis is the number of bids in the problem in increasing order and the Y-axis is the amount of time in seconds required for execution. The execution time for the algorithms is compared over the bigger problems dataset with 2000 to 8000 bids in each problem with various combinations of the number of items. As can be observed from Figure 12 and 13, the execution time for the MTR algorithm is much greater than the execution time for both the OPCOST and OPCOST-R and increases rapidly with increasing number of bids. The OPCOST-R algorithm does take longer time to execute than the OPCOST but only marginally higher than the OPCOST and increases similar to OPCOST with the number of bids in the problem. Both the OPCOST-R and the MTR algorithms although taking longer to execute than the OPCOST produce significantly better results than the

OPCOST algorithm in most problems, thus potentially justifying their increased time for execution. Figure 12 and 13 illustrate the rate of increase of the execution time with respect to the increasing number of bids for all the algorithms. For OPCOST and OPCOST-R the increase in execution time is close to linear with the number of bids. The increase in time for the MTR algorithm with increasing number of bids in the problem is shown in Figure 12. Thus the MTR algorithm, although producing consistently better results than the OPCOST algorithm, is limited in use for problems with large number of bids because of the rapid increase of the associated execution time. Also it can be observed that the quality of solution produced by the OPCOST-R with respect to the OPCOST solution becomes better with increasing number of bids. The execution time for the algorithms was also studied as a factor of the number of items in the problems. Figure 14 shows the average time for execution for OPCOST, OPCOST-R and MTR with increasing number of items. The X-axis in the graph is the number of items in the problem in increasing order and the Y-axis is the amount of time in seconds required to execute. The execution time in all three algorithms can be observed to be almost linearly increasing with the increasing number of items in the problem.

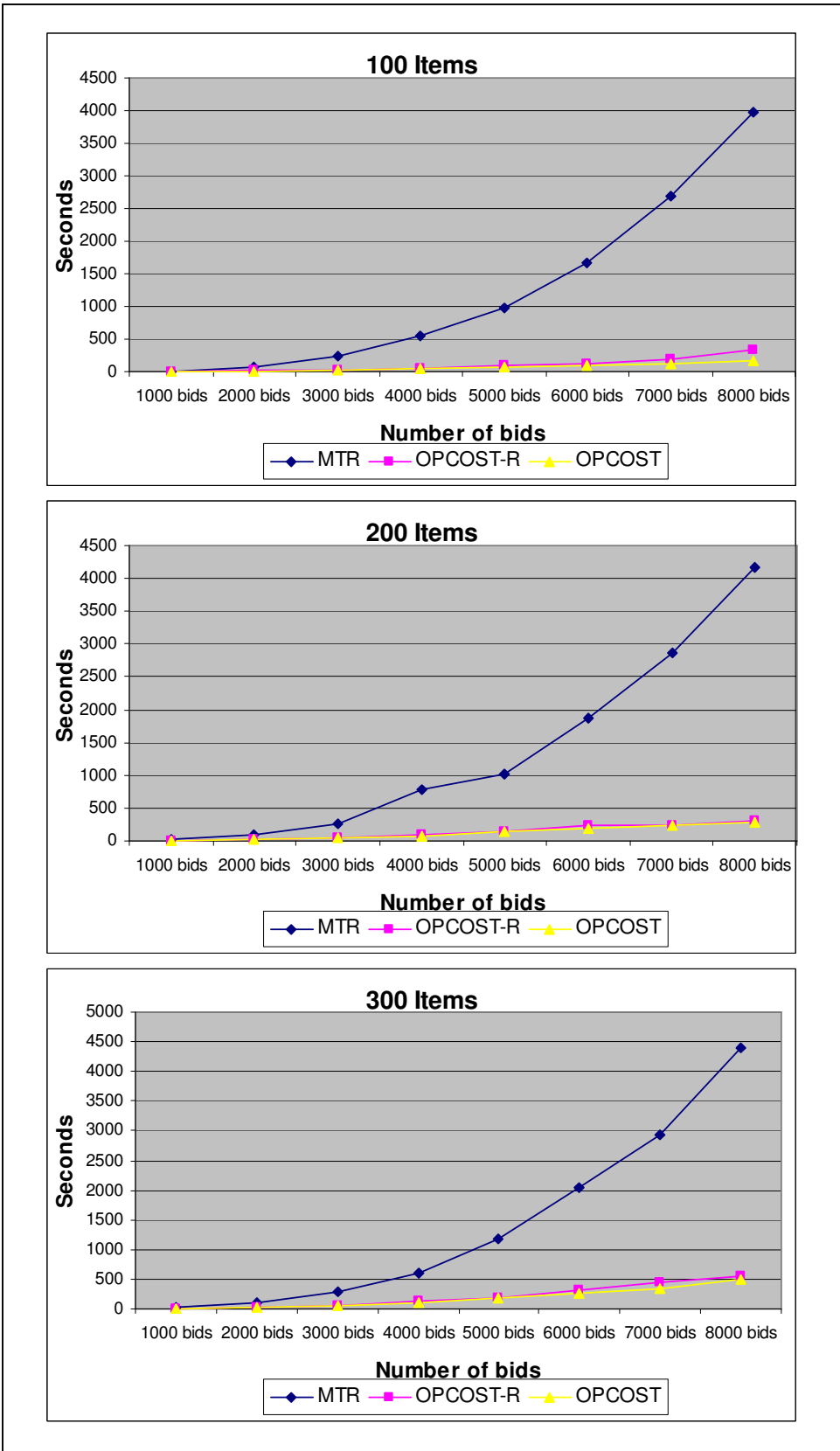


Figure 12: Execution Time vs. Number Of Bids (OPCOST, OPCOST-R and MTR)

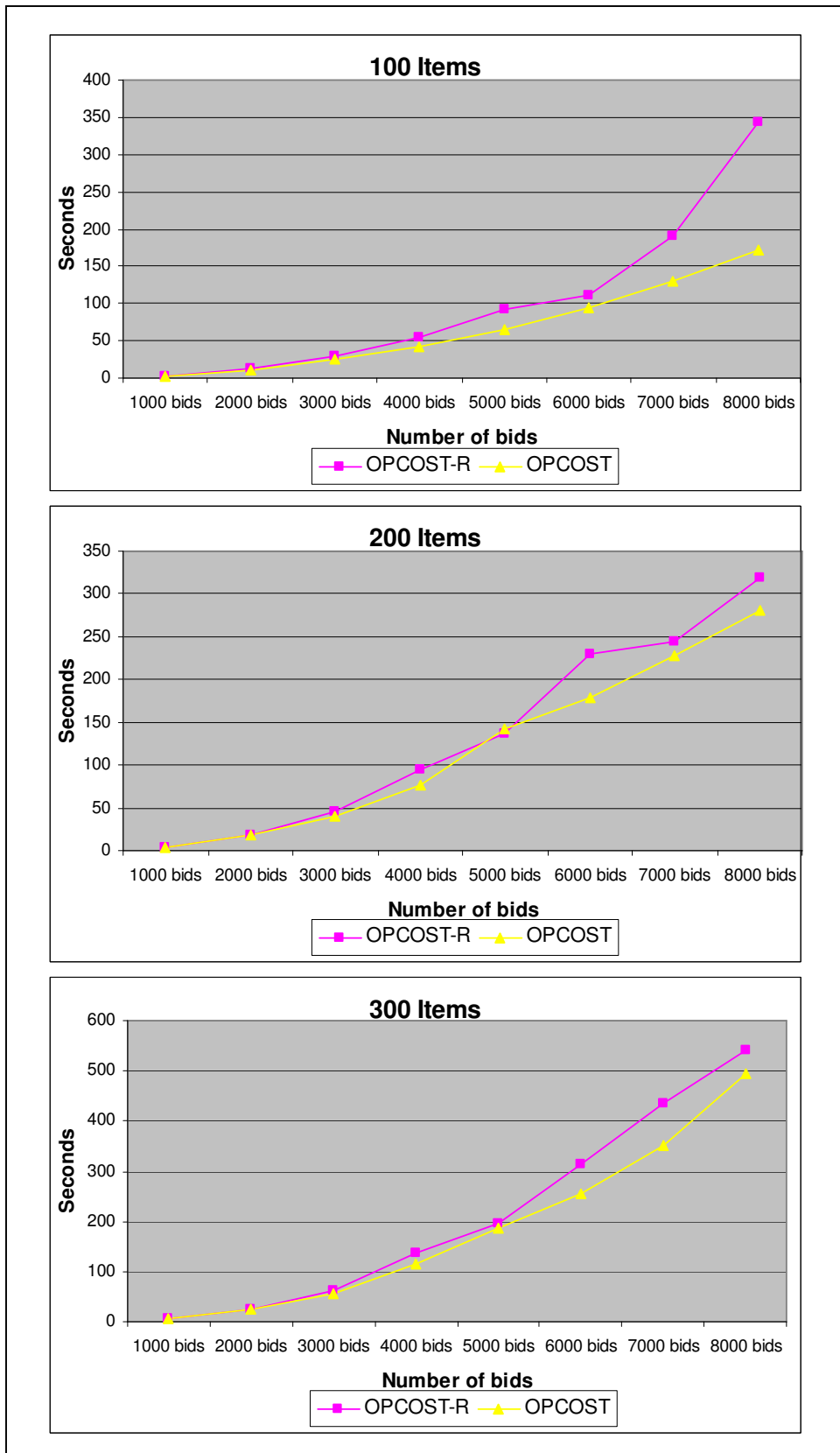


Figure 13: Execution Time vs. Number of Bids (OPCOST and OPCOST-R)

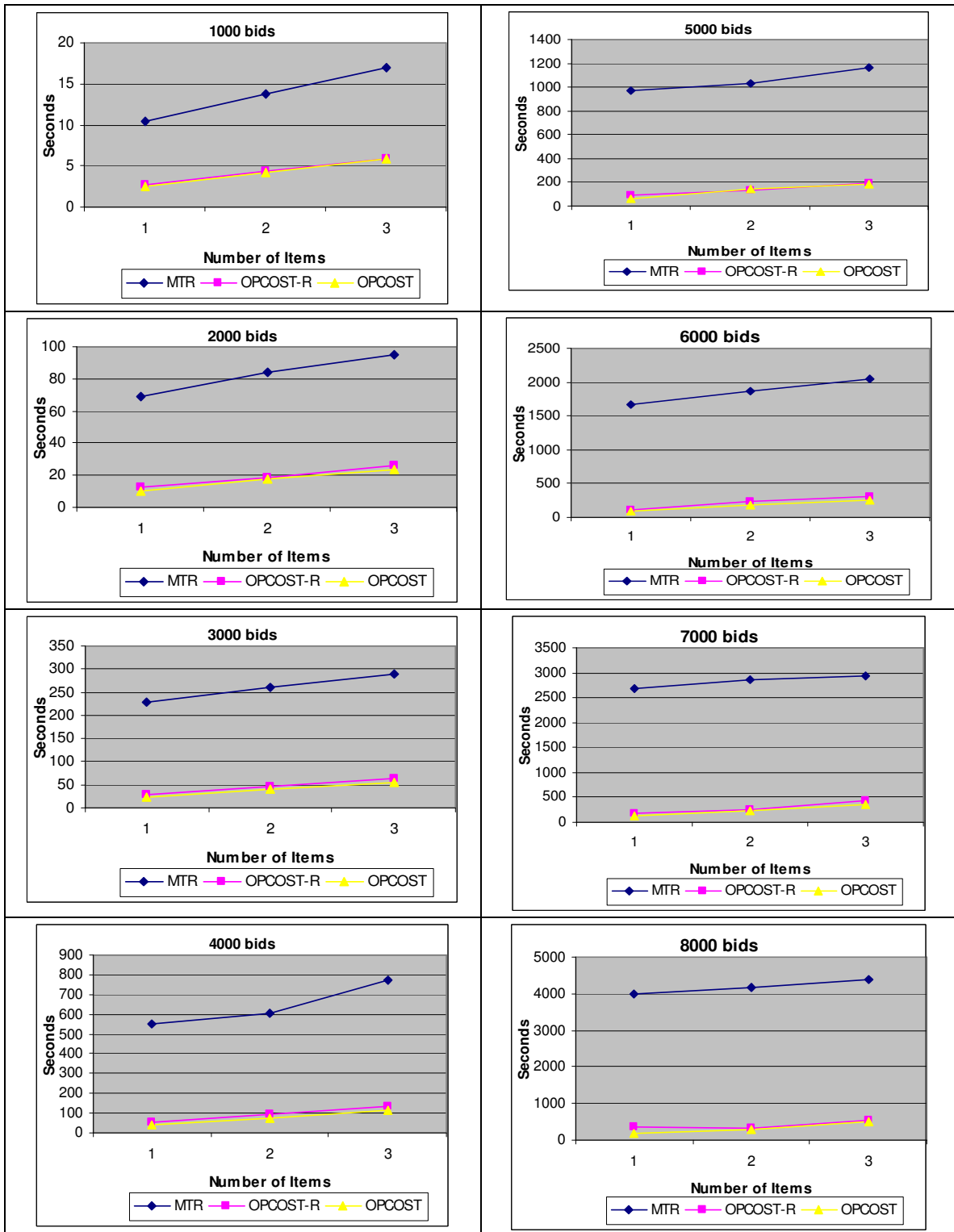


Figure 14: Execution Time vs. Number Of Items

## Chapter 7. Conclusions

This research studied the opportunity cost algorithm and the effects of various graph topologies and node ordering on the winning solution. The shortcomings of the opportunity cost algorithm were identified. Improvements to the opportunity cost algorithm in the form of recalculations in OPCOST-R algorithm were suggested and implemented which significantly improved the solution quality. The OPCOST-R improved the solution quality with a marginal increase in the execution time, which is well justified. The MTR algorithm was also proposed and implemented that significantly improved the solution quality. The execution time for the MTR algorithm increases rapidly with the problem size and must be used on problems of acceptable size. For significantly large problems the OPCOST-R algorithm, which produces significant increase in solution quality with a marginal increase in execution time must be used. The OPCOST-R and MTR when used in place of the OPCOST algorithm to determine the winning bidders in a combinatorial auction for carrier selection in supply chains will result in more desirable allocation resulting in cost savings for the shippers.

Further research can be done to study the effect of double counting and improved ways to eliminate it from the OPCOST algorithm. The solution quality and the execution of the OPCOST, OPCOST-R and the MTR on more problem datasets can be obtained and studied. The trends in the solution quality and execution time can also be studied as a factor of the ratio of the number of bids and the number of items.

## References

- 1) H.L. Lee, C.Billington, "Materials Management in Decentralized Supply Chains", Operations Research, vol. 41:5, 1993, pp. 835 - 47.
- 2) Vickrey, William, "Counterspeculation, auctions, and Competitive Sealed Tenders", Journal of Finance vol. 16, March, 1961, pp. 8 - 37.
- 3) P. Milgrom, "Auctions and Bidding: A Primer", Journal of Economic Perspectives, vol. 3, summer, 1989, pp. 3 - 22.
- 4) M.R. Andersson, T. Sandholm, "Time-quality tradeoffs in reallocation negotiation with combinatorial contract types", Proc. American Association for Artificial Intelligence-99, Orlando, FL, 1999, pp. 3-10.
- 5) R. Preston McAfee, J. McMillan, "Analyzing the airwaves auction", J. Economic Perspectives, vol. 10:1, 1996, pp.159 - 175.
- 6) T.Sandholm, "An algorithm for optimal winner determination in combinatorial auctions", International Joint Conference on Artificial Intelligence -99, pp.542 - 547, Stockholm, 1999.
- 7) N. Nisan, "Bidding and allocation in combinatorial auctions", In Association for Computing Machinery Conference on Electronic Commerce, pp. 1-12.
- 8) H. Hoos and C. Boutilier, "Solving combinatorial auctions using stochastic local search", In Proceedings of the Seventeenth National Conference on Artificial Intelligence, Austin, TX, 2000, pp.22 - 29.
- 9) E. Zurel, N. Nisan, "An efficient approximate allocation algorithm for combinatorial auctions", Electronic Commerce '01, October 14-17, 2001, Tampa, Florida, USA.

- 10) Y. Sakurai, M. Yokoo, Koji Kamei, "An efficient approximate algorithm for winner determination in combinatorial auctions", Electronic Commerce '00, October 17-20, 2000, Minneapolis, Minnesota.
- 11) C. Caplice, 1996, "An optimization based bidding process: A new framework for shipper-carrier relationship", Ph.D. Thesis, MIT.
- 12) J. Ledyard, M. Olson, D. Porter, J. Swanson and D. Torma, "The first use of a combined value auction for transportation services", Interfaces, 2002.
- 13) M. Rothkopf, A. Pekec, R.M. Harstad, "Computationally manageable combinatorial auctions", Management Science, vol. 44, No.8, 1998, pp. 1131 - 1147.
- 14) S. de Vries and R.V. Vohra, "Combinatorial Auction: A survey", INFORMS Journal on computing, 2001.
- 15) S.J. Rassenti, V.L. Smith, R.L. Buffin, "A combinatorial auction mechanism for airport time slot allocation", Bell Journal of Economics, vol. 13, 1982, pp. 402 - 417.
- 16) F. Kelly and R. Steinberg, "A combinatorial auction with multiple winners for universal service", Management Science, vol. 46, 2000, pp. 586 - 596.
- 17) T. Sandholm and S. Suri, "BOB: Improved winner determination in combinatorial auctions and generalizations", Artificial Intelligence, vol. 145, 2003, pp. 33 - 58.
- 18) Rica Gonen, Daniel Lehmann, "Optimal solutions for multi unit combinatorial auctions: Branch and bound heuristics", Electronic Commerce 2000, October 17 - 20.

- 19) Kevin Leyton-Brown, Yoav Shoham, Moshe Tennenholtz, “An algorithm for multi-unit combinatorial auctions”, American Association for Artificial Intelligence, 2000.
- 20) Holger H. Hoos and Craig Boutilier, “Solving combinatorial auctions using stochastic local search”, American Association for Artificial Intelligence, 2000.
- 21) Joni L. Jones and Gary J. Koehler, “An allocation heuristic for combinatorial auctions”, in review ,2001.
- 22) Karhan Akcoglu, James Aspnes, Bhaskar DasGupta and Ming-Yang Kao, “Opportunity Cost Algorithms for Combinatorial Auctions”, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) Technical Report 2000-27, 2000.
- 23) R.Bar-Yehuda and S.Even, “A local-ratio theorem for approximating the weighted vertex cover problem”, Annals of Discrete Mathematics, vol. 25, 1985, pp. 27 - 45.
- 24) A. Bar-Noy, R. Bar-Yehuda, A. Freund, J.S. Naor and B. Schieber, “A unified approach to approximating resource allocation and scheduling”, Proceedings of the 32<sup>nd</sup> Annual Association of Computing Machinery symposium on Theory of Computing, 2000, pp. 735 - 744.
- 25) Trade Extensions. <http://www.tradeextensions.com/press/volvopackcase.html>, February 2001.
- 26) Federal Communications Commission. <http://wireless.fcc.gov/auctions/31/>, April 2000.

## **Vita**

Viswanath “Vish” studied manufacturing engineering at Anna University, Chennai, India. He completed this program in 2000 with a bachelor of engineering degree. In 2004, he graduated from the Master of Science program in industrial engineering at Louisiana State University (LSU). During the course of his study at LSU, he worked as an intern with various startup and multi-national companies in Louisiana accruing a wealth of knowledge in business and real world experience. Vish will join First Responder Systems and Technology, one of the startup companies he had interned with earlier, as a systems analyst after his graduation in August 2004.