The Winner Determination Model and Computation for Linear Arrangement of Booth Auction

Puchit Sariddichainunta* and Krung Sinapiromsaran*

Abstract

The winner determination problem (WDP) for a single object auction is a relatively easy problem to solve using the greedy algorithm. However, a booth auction is one of the nonidentical multiple-object auctions which is NP-hard problem. Formulation of the winner determination model for a linear arrangement of a multiple-object auction is explained in this study – the integer linear programming model. Moreover, this research improves the polynomial time complexity algorithm from the previous study of allocation in geometry-based structure. Finally, the comparison of a running time exhibits the advantage of our proposed algorithm. The simulation results are discussed.

Keywords: multi-object auction, winner determination problem, integer programming model, polynomial time algorithm style.

1. Introduction

Auction is a well-known process to determine allocation of some scarce objects which are highly demanded by many agents [1]. In the general auction, there is only one owner, so-called auctioneer, and several agents, so-called bidders to participate in the event. The auctioneer imposes justifiable competition rule such as a price submission method and a payment method for bidders. Next, bidders submit their competitive price according to their willingness to pay. Finally, the auctioneer evaluates those offers and determines the most advantageous bidder to be the winner of the auction. This kind of optimization problem is known as the winner determination problem (WDP).

WDP is one part of research in combinatorial auction [2]. WDP focuses on the termination process of auction – allocation for bidders which is an optimal decision for auctioneer. The other part is the analysis of strategic behaviors among bidder. Since they could submit the price which is lower than their estimation, the auctioneer would lose some benefit from this manner. Therefore, the auctioneer has to design a mechanism to create sufficient incentive for bidders to tell their truthful estimation for price submission. However, in this paper, we treat the strategic behavior as given before computation in WDP. Indeed, we are interested just in improving the solution time for the optimal bidding solution.

The difficulty of WDP depends on the number and the type of objects in the auction, which is the effect of the bidding process and payment rule [1]. First, the object in the auction is the major source of complexity to determine the winner in the auction. The bidding strategy for a single object is simple, but WDP becomes the combinatorial problems when bidders have more options to bid their targets. In addition, if the object is indivisible, then the solution must be integer. As well, the case of non-identical multiple objects imposes computational burden on the auctioneer in searching for the optimal solution. In sum, WDP is an NP-hard problem because its decision variable sparsely growing in number of bidders and combination of options.

2. Problem Background

Our problem domain is a linear arrangement of a booth auction. There is a big space in a hall; the layout of the hall has been planed and divided into blocks in a certain size. We specify our problem into two cases: a single line and double line. As in Fig. 1, the booths are set up in a single line consecutively, or likely in a double line which back side of each booth locates back-to-back.

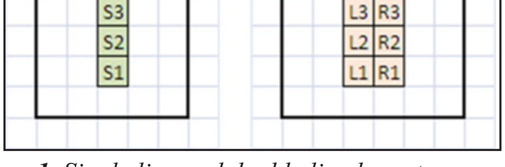

Figure 1 *Single line and double line layout, respectively.*

* *Department of Mathematics and Computer Science, Chulalongkorn University.*

The bidding options are followed by the layout of booth. This situation is defined as linear arrangement. Bidders can offer one price for one bundle of blocks to rent the consecutive blocks only; e.g., bundle S2 and S3, bundle L1, L2 and L3. Unconnected or non-rectangular blocks are not permitted; e.g., a bundle with S4 and S2, a bundle with R3, R2 and L2. In the case of single line, if we have n row blocks, the number of possible bundle is $n(n+1)/2$. An example of the booth of three blocks is displayed in Fig. 2 that there are 6 possible bidding options. In the case of double line, the number of possible bundle is $3n(n+1)/2$, which provides rectangular blocks. These combinations are from two times of single line case plus crossing line case. Therefore, the option in this combinatorial auction does not really grow exponentially. This is a special structure of the linear arrangement in our problem domain.

Next, the rational behavior of the auctioneer is discussed. The rational auctioneer has to select the offers from many bidders to maximize his benefit. In order to obtain the optimal allocation in this combinatorial auction, it is necessary to compare each offer more carefully than single object auction. The greedy algorithms would not be appropriate. For instance, the situation of three-block single line has price offer from each options –S1, S2, S3, S1∪S2, S2∪S3, S1∪S2∪S3 – as the following price offers in Fig. 2. There are bidders A and B compete in the auction. If the auctioneer begins with the biggest bundle S1∪S2∪S3, he obtains bid value 7. Next step, the auctioneer compares result with partition [S1, S2∪S3]; the revenue becomes 1+4=5. This new partition returns less than the former; therefore, the greedy algorithm terminates with the result [S1∪S2∪S3]. However, the partition [S1, S2, S3] provides better solution which the bid result is 1+5+2=8. Hence, the greedy algorithm might not give the optimal solution for the auctioneer.

Bundle	A	B	Max
S1	1	1	1
S2	3	5	5
S3	2	1	2
S1∪S2	4	5	5
S2∪S3	4	3	4
S1∪S2∪S3	7	6	7

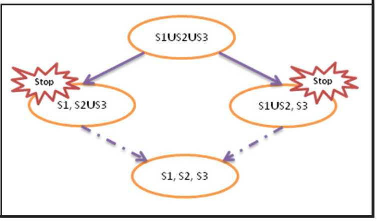

Figure 2 *Bidding in combinatorial case which greedy algorithm cannot provide optimal solution.*

Indeed, this research employs other methods to better solve the winner determination problem in a linear arrangement of booth auction; i.e., integer programming approach and dynamic programming approach. Their specific details are explains in the methodology section.

3. Literature Review

Combinatorial auction has been studied in a theoretical aspect and an algorithmic aspect. The former focuses on auction design and the latter on efficient algorithms. Auction design is the study for effective auction rules. However, the effectiveness of the rules would be lessened if the computation is excessively costly and impractical in reality [2]. Therefore, scholars consider on the specific problem domains observed in reality, and propose manageable solving method. This section briefly reviews some literatures on WDP for some specific problems, such as, internet ad-slot auction and geometric allocation.

Geometric allocation is our main concern in this paper. Rothkopf et al [3] explicitly identify this type of problem domain. They begin with the computability with limitation on permitted combinatorial bids; e.g., nested structures, cardinality-based and geometry-based structure. Nested structures are that only one type of combination is able to bid together, while the cardinality-based structures illustrate more than one type of group. The geometry-based structure is the most relevant to our problem which binding biding options depending on their adjacency as explained in the problem background. According to those structures, they propose the polynomial time algorithms to solve and prove the optimality in auctions.

In addition, Tennenoltz [4] further investigates the tractable case for combinatorial auction. He proves polynomial solutions for a combinatorial network auctions, various sub-additive combinatorial auctions, and some restricted forms of multiple-objects auctions. The allocation of objects in geometry-based structure could be one of the restricted forms. In his proofs for polynomial complexities solutions, he elaborates b-matching techniques in graph algorithms to identify those tractable combinatorial auctions [5]. Even though there are no implementation results in this work, the computationally tractable in polynomial complexity are guaranteed for combinatorial auction in geometry-based structure.

Internet ad slot auction is another important research target actively conducted by researchers not only in academics but in private research institutes, Yahoo and Google. Internet ad slot auction is similar to booth auction that bidders want the best position for high rate of visits. Regarding to bidders' pricing for different position and budget constraint, Feldman et al [6] set up an allocation rule for advertisement slot and algorithms to solve WDP. Since the unit of marginal benefit to acquire from ad slot in this auction is number of clicks, the nature of auction object is divisible when consider the allocation as proportion occupied each slot. Moreover, bidder is not able to request for specific ad slot position, while he could obtain just the clicks which redistribute after each winners being determined. Thus, the bidders' specific option in position arrangement has not yet been studied for the internet ad slot auction.

Among those special structures in combinatorial auction, this paper focuses on the linear arrangement of a multiple-object auction. The research problem is that the auctioneer arranges the auction to distribute blocks lined on a row and bidders can inquire for one block or several consecutive blocks without separation. Basically, this structure of problem is solvable in integer programming, and reducible to linear programming formulation. This mathematical programming method is compared with our modified algorithm of Rothopf et al [3] for single line. Moreover, we successfully extend our algorithm to the double line case of booth auction.

The structure of this paper after introduction is as the following: methodologies, experimental results, and conclusion.

4. Methodology

In this section, our main solving method for WDP is explained: integer programming and dynamic programming approach. Initially, integer programming is very flexible to solve WDP [7]; however, the computation complexity is the main concerns of this method. That the bidding option grows rapidly and the number of decision variables increase sharply incurs computation cost. On the other hand, dynamic programming is an alternative to avoid expensive calculation [3,4]. While the efficient calculation is guaranteed, the con is the limitation to solve for a specific domain of problem, unlike integer programming for general WDP. The explanations of both methodologies are followed as below.

A. Integer programming

The winner determination problem is based on the assignment problem. In this paper, we use binary decision variables to indicate the optimal solution and allocation constrains. There is a finite set of bidders N, with n bidders, and a finite set of indivisible objects, G, with m row blocks. Each bidder $i \in N$ has a non-negative, integer valuation for each bundle of objects $S \subseteq G$ denoted by $b_i(S) \in N_0$. The binary decision variables are defined by $x_i(S) \in \{0, 1\}$; $x_i(S) = 1$ means that the bundle S is allocate to bidder i and otherwise

(IP1)
$$\max \sum_{i=1}^{n} \sum_{S \subset G} b_i(s) x_i(s)$$

$$\sum_{i=1}^{n} \sum_{S \subset G, S} x_i(s) \le 1 \qquad \text{for all } j \in G$$

$$x_i(s) \in \{0, 1\}.$$

WDP is formulated in a binary integer programming as in (IP1). The meaning of each line is straightforward. The objective function is to maximize revenue for each set, and the constraints are to prevent duplicated allocation of each object. Moreover, one bidder is able to get more than one bundle if the solution is still feasible. Specifically, this WDP is in the OR bidding language defined in [2]. We explicitly illustrate the matrix A_i as the coefficient matrix in the constraint for bidder i, and the sparse structure of coefficient is observed.

For simplicity, we assume the size of to G be 4 to demonstrate the structure of coefficient matrix. The possible bundle is illustrated as $s = [a, b]$: a is the begin block position and b is the end block position. For example in Fig. 1, [S2, S2] means only one single block at S2 in single line case. For double line case, [L1, R3] is a rectangular six blocks from L1 to R3, and [L2, L4] is a bundle of three blocks on the left side from L2 to L4 consecutively. Thus, we can represent all possible options as index of column in coefficient matrix.

Single line case

Let m be the number of row blocks, $m = |G|$.

$$\begin{array}{c} \begin{array}{cccccccccc} \scriptstyle S1S1 & \scriptstyle S1S2 & \scriptstyle S1S3 & \scriptstyle S1S4 & \scriptstyle S2S2 & \scriptstyle S2S3 & \scriptstyle S2S4 & \scriptstyle S3S3 & \scriptstyle S3S4 & \scriptstyle S4S4 \end{array} \Leftarrow S = [a, b] \\ A_i = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{array}{l} \text{for all } i \text{ bidder} \\ A_i \sim m \times 0.5m(m+1) \end{array} \end{array}$$

$$A_i = [A_1 \ldots A_i \ldots A_n]$$

Double line case

$$\begin{bmatrix} A_i & 0 & A_i \\ 0 & A_i & A_i \end{bmatrix} \text{for all } i \text{ bidder} \quad \tilde{A}_i \sim 2m \times 1.5m(m+1)$$

$$\tilde{A} = [\tilde{A}_1 \ldots \tilde{A}_i \ldots \tilde{A}_n]$$

According to the special structure of these coefficient matrices, total unimodular property are claimed by Schrijver [8] and Ahuja et al [5]. For the sake of integer value in right hand side and total unimodularity, the integer formulation (IP1) is reducible to (LP1) which becomes a linear programming which yields integer outcome, in this case just 1 or 0. Hence, it is eligible to employ just linear programming, (LP1) and (LP2), instead of integer programming.

(LP1)

$$\max \sum_{i=1}^{n} b_i x_i$$

$$\sum_{i=1}^{n} A_i x_i \le 1$$

$$x_i \ge 0$$

(LP2)

$$\max \sum_{i=1}^{n} \tilde{b}_i \tilde{x}_i$$

$$\sum_{i=1}^{n} \tilde{A}_i \tilde{x}_i \le 1$$

$$\tilde{x}_i \ge 0$$

B. Dynamic programming

A dynamic programming can be used to solve the optimization problem. We exert a square matrix and its index to construct data structure which is suitable to work for dynamic programming. The idea is influenced by the setup for single line of Rothkopf et al in algorithm 3 of [3], called RPH's method. Their algorithm is rewritten in Pseudocode 1 and 2. RPH's complexity is $O(n^2)$ for a single line with fixed start, say [1, n]. Pseudocode 1 demonstrates this method; specifically, it has $\sum_{k=1}^{n-1} k = n(n-1)/2$ comparison works.

The method which is suitable to our problem is the intervals on the line. They interpreted as the intervals on the circle of which computational complexity is $O(n^3)$. The reason is simple that it repeats each block to start again by renumerating the objects. The first round is [1, n], then the second round is [*2, n+1*] which *n+1 refer to the first block*, and so on. After that, we bring those n outcomes to contest for the most valuable solution. This method is displayed in Pseudocode 2. Precisely, the count is $n^2(n-1)/2 + n$. The former part is to repeat the previous algorithm n time, and the latter is to compare the results of each round.

```
Pseudocode 1: Fixed start
step 0  Input  $p(i, j)$ for all $i, j$
step 1  Set    $w(1) = p(1,1)$. Set $r = 2$.
step 2  Set    $w(r) = p(1,r)$.
step 3  For    $i = 2$ to $r$
        If     $w(i-1) + p(1, r) > w(r)$
        Then   $w(r) = w(i - 1) + p(1, r)$
step 4  If $r < n$, then set $r = r + 1$ and go to step 2
    Otherwise, terminate with optimal revenue $w(n)$.
```

On the other hand, our modified methods for single line case and double line case are in Pseudocode no.3 and no.4 respectively. Firstly, the single line case has computational complexity equivalent to $O(n^3)$. Precisely, the work is counted as $\sum_{k=1}^{n-1} k(n-k) = n(n-1)(n+1)/6$ which is less than the case of algorithm in Pseudocode 2.

```
Pseudocode 2: Intervals on the circle
step 0  Input  $p(i, j)$ for all $i, j$
step 1  For $k = 1: n$
   1.1  Set  $w(k) = p(k, k)$. Set $r = k + 1$.
   1.2  Set  $w(r) = p(k, r)$.
   1.3  For  $i = k + 1$ to $r$
        If   $w(i - k) + p(k, r) > w(r)$
        Then $w(r) = w(i - k) + p(k, r)$
   1.4  If $r < n+1- k$, then set $r = r+1$ and go to step 1.2
        Otherwise, get the kth round optimality $\mathbf{w}(\mathbf{n}+1-k)$
step 2  $w(\text{opt}) = \text{Max}\{ w(n),..., w(2n-1)\}$
```

Our algorithm works more efficiently since we reap the benefit of data structure more effectively than the RPH's method in Pseudocode 2. Further that, our approach can easily apply to double line case which has complexity in $O(n^3)$ as well. The calculation burden is obviously three time of the single line case, since it works repetitively for the left, right and crossing line. Thus, we improve RPH's method our approach in dynamic programming is effective for booth auction.

In dynamic programming, the data structure for RPH's method and our method is two-dimension array. There is a little difference for the algorithm in pseudo code 3 because of the repetitively and elongation. The algorithm of RPH will reorder that the first block connect to the end and the second block becomes a new fixed start. However, the basic idea of operation is to pick up input value from the matrix to searching for the optimal solution. Optimality in dynamic programming is proved by mathematical induction as regarded in [9].

To implement each algorithm, the bidding values are necessary to rearrange in descendent order. The most valuable bid in each option becomes the first input in square matrix of the algorithms. For a single line case, only one matrix is sufficient to keep the highest bid for every option. However, for a double line case, it is necessary to utilize three matrices for valuation inputs; i.e., left, right and crossing line. These three matrices represent the best price of each option.

In Pseudocode 3 and 4, WDP is characterized and recursively defined easily by two-dimension array. Step 1 informs the stage of computation. Next, step 2 characterizes the maximum value referred to a related value from the previous stage. Subsequently, it leads to the maximum value in the final stage. In another word, the final result depends on comparison of their substitutable pair that each component also relies on the relevant pair backwardly. Those previous comparison results are put conveniently in the callable memory in our data structure. In double line case, the sequence is more complex. There are more moving to compare the

```
Pseudocode 3: Single line algorithm
step 0  Given square matrix array size $n \times n$, say B ~ $n \times n$
step 1  step 1  Let $k = 0$ be the diagonal line in the matrix;
        $k$+1 is the consecutive lower diagonal line.
step 2  For    $k \geq 1$
        For    $j = 1$ to $n$
               $w = j + k$
            If   $w \leq n$
               For $k$ = $j$ to ($w$-1)
                    If      B($w, j$) < B($k, j$) + B($w, k$+1)
                    Then   B($w, j$) = B($k, j$) + B($w, k$+1)
step 3     Terminate when $k = n$.
```

```
Pseudocode 4: Double line algorithm
step 0  Run single column algorithm for the left and right
        column and keep result in the square matrix L
        and R respectively.
        Given a square matrix C for cross-side options value
step 1  Let $k = 0$ be the diagonal line in the matrix; $k$+1
        is the consecutive lower diagonal line.
step 2  For $k \geq 0$
        For    $j = 1$ to $n$
                 $w = j + k$
               If $w = j$
                 If $C(j, j)$ < $L(j, j) + R(j,j)$
                 Then      $C(j, j)$ = $L(j, j) + R(j, j)$
               If $w \leq k$
                 For $k$ = $j$ to ($w$-1)
                      If $C(w, j) < C(k, j) + C(w, k$+1)
                     Then $C(w,j) = C(k, j) + C(w, k$+1)
step 3  Terminate when $k = n$.
```

option crossing between left and right line; however, the main idea, to compare bidding price from the lowest single level first and consecutively move to the biggest bundle later, is unchanged.

5. Experimental Results

The experimental environments, parameter setting and simulation results are described in this section. First, the computer for simulation experiments has the following specification: CPU is Intel Core 2 Quad Processor 2.83 GHz with RAM 2 GB. The operating system is Windows XP service pack 3. Linear programming solver and the other algorithmic codes are implemented by Matlab.

Next, bidding values vector for each option are generated randomly by Matlab internal pseudorandom generating command. To pick up simulated value, we also maintain two assumptions. One is additive bundle assumption that the more combining blocks the greater valuation, the other is that the nearer the gate the more expected benefit. For the case of n bidders, vector of random number is converted to the ceiling integer number to represent each individual evaluation. Even though the various parameters setting are adjustable, the comparison results in Tab. I and Tab. II are restricted to the case of 10 row blocks for the sake of limited space. The simulation round is one thousand.

The time measurement is recorded from their actual jobs. The solver for linear programming is the simplex method, and the time is measured soon after the optimal solution revealed. For dynamic programming, the running time in preprocess, to select the best offer for each option, is included and sum with the computation time of comparison process. The average time from the 1,000 simulation experiments, the case of 10 row blocks auction, is indicated in Tab. I and Tab. II, corresponding with total number of bidders in the auction. We have two parts of simulation as the following: single line case and double line case.

A. Single line case

The experimental results of single line case is obvious that the dynamic programming approach, RPH's and author's method, are further effective than linear programming approach. Moreover, the average of operation time in our method is significantly less than RPH's method. This simulation result consequently supported to our computational thought in the methodology. The line plot also shows the advantage of our method over linear programming and RPH's method.

Table 1 *Average running times of each algorithm for single line case, unit: micro seconds.*

No. of bidders	1	2	3	4	5	6	7	8	9	10
Linear Prog	4330	4680	5233	5753	6241	6837	7531	8281	9205	10327
RPH	74	92	98	109	119	130	144	157	173	193
Authors	56	65	66	68	70	71	74	77	79	82

Single line case

Average time in log scale

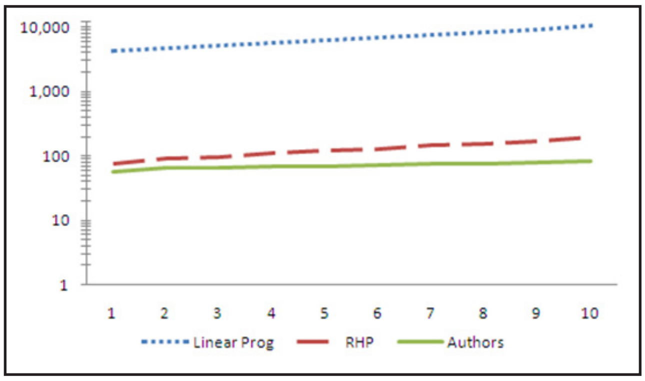

Figure 3 *The comparison results of average time among linear programming, RPH and our method. The y-axis is running time in log scale and the x-axis is the number of bidders.*

B. Double line case

In double line case, the result is consistent to the single line case. Hereafter, the dynamic programming approach hereafter refers to the authors' method only since RPH's method is not adaptable to double line case. The box plots displayed in Fig. 4 expands the whole picture of simulation. The vertical axis is running time in micro seconds and the horizontal axis is the total number of bidders in the auction. It is obvious that running time monotonically increase when the number of bidders grows up for both algorithms. The dynamic programming shows many outliers, which the further analysis might be necessary.

From the result in Tab. 2 and box plots, the average time operated in dynamic programming approach is far superior to linear programming approach. The first reason is the structure of coefficient matrix in linear programming bears too many feasible solutions. In addition, iteratively solving for linear programming approach is very costly because it has to update sparsely matrix too often [10].

Table 2 *Average running times of each algorithm for double line case, unit: micro seconds.*

No. of bidders	1	2	3	4	5	6	7	8	9	10
Linear Prog	16900	27630	32350	35240	39290	43530	48690	52610	56990	61880
Authors	100	110	109	111	114	117	121	124	121	125

Double line case:

Average time in log scale Dynamic programming

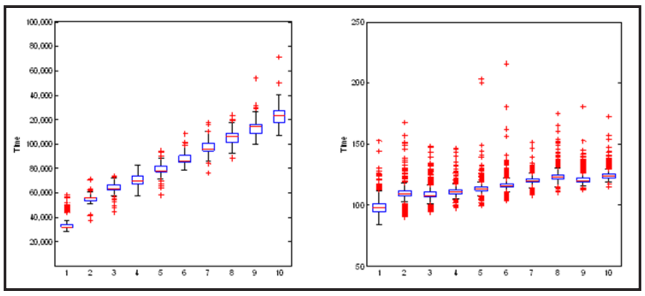

Figure 4 *The experimental results of linear programming and dynamic programming approach – the authors' method.*

6. Conclusion

Experimental results suggest that the dynamic programming approach is averagely more superior to linear programming approach in our setting environment – a booth auction in single line and double line. When changing the quantity of row blocks and number of bidders, the results would reinforce the inefficient trend of linear programming approach. While the sparse matrix structure is the root cause of linear programming approach, the two-dimension array data structure contributes realizes the recursive benefit of dynamic programming that brings about higher performance.

Actually, the real layout can utilize our method to determine winner in booth auction. Some modification techniques in programming are necessary for the booth allocation which has obstacles in the layout. The applications of this method would not probably be limited to the allocation of booth space, but the virtual space like the internet ad slot would be implied. The probabilistic argument of visitors is very important and should be defined before conduct further analysis.

7. References

[1] V. Krishna, *Auction Theory*, 2nd ed. Amsterdam: Elsevier, 2010.

[2] R. Müller, "Tractable Cases of the Winner Determination Problem," *in Combinatorial Auctions*, Y. Shoham and R. Steinberg P. Cramton, Ed. Cambridge, Massachusetts: The MIT Press, ch. 13, pp. 319-336, 2006.

[3] M.H. Rothkopf, A. Pekeč and R.M. Harstad, "Computationally Manageable Combinational Auction," *Management Science*, vol. 44, pp. 1131-1147, Aug. 1998.

[4] M. Tennenholtz, "Some Tractable Combinatorial Auctions," *in the Seventeenth National Conference on Artificial Intelligence*, Austin, pp. 98-103, 2000.

[5] T.L. Magnanti, J.B. Orlin R.K. Ahuja, Network flows : theory, algorithms, and applications. New Jersey: Prentice-Hall International, 1993.

[6] J. Feldman, S. Muthukrishnan, E. Nikolova and M. Pál, "A Truthful Mechanism for Offline Ad Slot Scheduling," *in Lecture Note in Computer Science*, B. Monien and U. Schroeder, Ed. Heidelberg: Springer-Verlag, vol. 4997, pp. 182-193, 2008.

[7] S. Bikhchandani and J. Ostroy, "From the Assignment Model to Combinatorial Auctions," *in Combinatorial Auctions*, Y. Shoham and R. Steinberg P. Cramton, Ed. Cambridge, Massachusetts: The MIT Press, ch. 8, pp. 189-214, 2006.

[8] A. Schrijver, *Theory of Linear and Integer Programming*. Chichester: John Willey & Sons, 1986.

[9] U. Manber, Introduction to Algorithms: A Creative Approach. Reading, Massachusett: Addison-Wesley, 1989.

[10] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali, *Linear Programming and Network Flows*, 3rd ed. New York: Willey, 2004.