Algorithmic Operations Research

Branch-and-Cut Algorithms for Winner Determination in Discount Auctions

S. Kameshwaran, Lyès Benyoucef and Xiaolan Xie

Volume 2, Number 2, Summer 2007

URI: https://id.erudit.org/iderudit/aor2_2art04

See table of contents

Publisher(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (digital)

Explore this journal

Cite this article

Kameshwaran, S., Benyoucef, L. & Xie, X. (2007). Branch-and-Cut Algorithms for Winner Determination in Discount Auctions. *Algorithmic Operations Research*, *2*(2), 111–128.

Article abstract

Discount auction is a procurement mechanism for buying M indivisible heterogeneous items. The bidders are suppliers and a bid consists of two entities: individual cost for each of the items and a non-decreasing discount function defined over the number of items. The winner determination problem faced by the buyer is to determine the winning suppliers and their corresponding winning items that minimizes the total procurement cost, subject to the supply, demand, and discount constraints. We show that this problem is N P-hard upon reduction from the set covering problem. An integer programming formulation is presented and valid inequalities are derived, which serve as cuts to the linear relaxation. A collection of branch-and-cut algorithms are developed with different cut addition techniques and branching strategies. The performance of the proposed algorithms for different problem types are studied with extensive computational experiments.

Algorithmic Operations Research Vol.2 (2007) 112–128

Branch-and-Cut Algorithms for Winner Determination in Discount Auctions

S. Kameshwaran [a]

[a]Indian School of Business, Hyderabad 500032, India

Lyès Benyoucef [b]

[b]INRIA Lorraine, ISGMP Bat: A, Ile du Saulcy, Metz 57000 France

Xiaolan Xie [c]

[c]ENSM de Saint-Etienne, 158, cours Fauriel, 42023 Saint-Etienne, France

Abstract

Discount auction is a procurement mechanism for buying M indivisible heterogeneous items. The bidders are suppliers and a bid consists of two entities: individual cost for each of the items and a non-decreasing discount function defined over the number of items. The winner determination problem faced by the buyer is to determine the winning suppliers and their corresponding winning items that minimizes the total procurement cost, subject to the supply, demand, and discount constraints. We show that this problem is $\mathcal{NP}$-hard upon reduction from the set covering problem. An integer programming formulation is presented and valid inequalities are derived, which serve as cuts to the linear relaxation. A collection of branch-and-cut algorithms are developed with different cut addition techniques and branching strategies. The performance of the proposed algorithms for different problem types are studied with extensive computational experiments.

Key words: Discount auctions, integer programming, linear relaxation, valid inequalities, branch-and-cut, transportation problem.

1. Introduction

Procurement is the process by which a company obtains materials and services necessary for its manufacturing and/or operations. The advent of Internet and Internet-based technologies have led to new and innovative auction mechanisms for procurement. With the Internet technologies enabling an interactive front end for human interaction and back end computers that can support complex computations, the research in e-procurement is focused on auction mechanisms, bidding languages, and winner determination techniques to make the process computationally and economically efficient. This has led to new generation of procurement techniques: *volume-discount* [9,13], *combinatorial* [12,15], and *multi-attribute* [2,4,21]. For a more general review of auction techniques in e-commerce and e-procurement, see [5,11].

Email: S. Kameshwaran [Kameshwaran_S@isb.edu], Lyès Benyoucef [lyes.benyoucef@loria.fr], Xiaolan Xie [xie@emse.fr].

The auction mechanism used for e-procurement primarily depends on the type and number of items procured. For an industrial procurement of large quantity of a single good (like raw material), volume discount auctions [13,22,23] are appropriate candidates. The bid submitted by a supplier is a cost *function* defined over the quantity. The function, in essence, can capture the discount offered by the supplier based on the quantity procured. Combinatorial auctions (CA) [8], are useful for procuring a set of heterogeneous, but related items. CA allows package bidding, that is, quoting a single cost for a bundle (subset) of items. In this way, the bidders can capture the *complementarity* or *substitutability* existing among the items in a bundle. For M items, a bidder could thus possibly submit $2^M - 1$ combinatorial bids, one for each of the possible bundles. The volume discount and combinatorial auctions have led to several profitable industrial procurements [3,12,15].

An alternate auction mechanism called as *discount auctions* (DA) for procuring heterogeneous items was proposed in [19]. DA is applicable in scenarios where

the items do not exhibit complementarity or substitutability. Consider the procurement of office supplies: *stationary*, *computers*, and *furniture*. A supplier has positive cost for each of the items and his profits may not change substantially by selling them separately or together. If he cannot sell the furniture in this auction, he can sell it elsewhere. This is not the case with items that exhibit complementarity. Bundles of items that cannot be split and items that cannot be bundled together are not uncommon in scenarios with complementarity and substitutability, respectively. Given that such conditions do not exist, the supplier is not concerned about *which* bundle of items, but rather about *how much* worth of items he can sell. The supplier has positive cost for each of the items, and in order to promote sales he gives incentive to the buyer by providing discounts on the number of items procured. The discount bid consists of two parts: (1) individual cost for each of the items and (2) discounts for different number of items (for example, if three items are bought then the discount is 5%, for four items 6%, etc.). The difference between DA and CA is obvious: the costs in the CA are defined over the subsets of the items whereas the discounts in the DA are defined over the cardinality of the subsets of the items.

In this paper, we develop exact branch-and-cut algorithms for the winner determination problem of DA. The winner determination problem faced by the buyer is to choose a set of winning bids and a set of winning items for each of the winning bids, such that all demanded items are procured at minimal (total) cost subject to the supply, demand, and discount constraints. DA can also be seen as an instance of CA with only substitutability and no complementarity. The cost of a combinatorial bid for a subset is the sum of the cost of the constituent items that is discounted based on the cardinality of the subset. Thus one can apply the well studied winner determination algorithms of CA [24,27] for DA. However, this is not favorable for the following reasons. Firstly, the total number of combinatorial bids is rarely its upper limit $2^M - 1$. Many of the CA algorithms exploit this property. But for an instance of CA obtained from DA, it is in the order of $O(2^M)$. Thus, the algorithms have to be applied to problems with the worst case size that is rarely encountered in practice. Secondly, DA is only a special case of CA and has many exploitable properties that CA lack. For example, in DA the buyer knows the range in which the optimal price is in for each item, which is not the case in CA. Hence, we develop algorithms for DA that exploit the computationally favorable structures in the problem. The preliminary versions of this work have been presented in [18–20]. This paper extends and complements the above with new branch-and-cut algorithms, branching techniques, primal heuristic, and extensive numerical experiments with various combinations of cut algorithms and branching rules.

The reminder of the paper is organized as follows. The winner determination problem of discount auctions is described and its complexity is studied in Section 2. An integer programming formulation is proposed and its linear relaxation is analyzed in Section 3. Further, the valid inequalities that can serve as violated cuts are identified. Section 4 presents the branch-and-cut algorithms with several cut addition techniques. A novel branching technique called as branch-on-price is proposed in Section 5. A primal heuristic exploiting the network structure of the problem is proposed in Section 6 to obtain incumbent solutions. Extensive computational experiments with varying problem sets were conducted for the proposed algorithms and their results are discussed in Section 7. Section 8 concludes the paper.

2. Discount auctions

The buyer is interested in procuring M different items. Each of the items is indivisible, *i.e.* it can be supplied by only one supplier. An *item* need not refer to a single unit. It can be *a computer* or *a computer and printer* or *hundred computers*, but it cannot be split and supplied by multiple suppliers. Let there be N suppliers. An item is denoted by index m and a supplier by index j. Each supplier can submit only one discount bid and hence the index j is used to denote both the supplier and his bid. The *discount bid* j (*i.e.* from supplier j) consists of two parts: (1) cost Q_j^m for each item m and (2) non-decreasing discount θ_j^i for i $(= 1, \ldots, M)$ number of items. The bid can be compactly expressed as an ordered pair of M-tuples: $((Q_j^1, \ldots, Q_j^m, \ldots, Q_j^M), (\theta_j^1, \ldots, \theta_j^i, \ldots, \theta_j^M))$. Note that m denotes a particular item and i denotes number of items. If the buyer procures items 2, 4, and 7 from bid j, then the cost of procurement is $(1-\theta_j^3)(Q_j^2+Q_j^4+Q_j^7)$. As three items 2, 4, and 7 were procured, the total cost was discounted by θ_j^3. All the Q_j^m are positive (possibly infinite for an unavailable item) and the θ_j^i are non decreasing over i (the discount cannot decrease with the number of items bought). The winner determination problem (WDP) faced by the buyer is to choose a set of winning bids and a set of winning items for each of the winning bids, such that all demanded items

are procured at total minimal cost, subject to the supply, demand, and discount constraints. This problem is $\mathcal{NP}$-hard.

Theorem 1 *The WDP of the discount auctions is $\mathcal{NP}$-hard.*

PROOF. We prove the hardness of the WDP by showing that the decision version of the WDP is $\mathcal{NP}$-complete upon reduction from the minimum set cover.

Definition 1. [DAuc]
INSTANCE: Set of goods $G = \{1, \ldots, M\}$, set of discount bids $J = \{1, \ldots, N\}$, where a discount bid $j \equiv ((Q_j^1, \ldots, Q_j^M), (\theta_j^1, \ldots, \theta_j^M))$ with $Q_j^m \geq 0\ \forall m \in G$ and $0 \leq \theta_j^i \leq \theta_j^{i+1} \leq 1, 1 \leq i < M$, and a goal $K \geq 0$.
QUESTION: Does there exist a winning set $J' \subseteq J$, which defines a partition $P = \{B_j : B_j \subseteq G, j \in J'\}$ of G, such that the total cost of procurement $\sum_{j \in J'} (1 - \theta_j^{|B_j|}) \sum_{m \in B_j} Q_j^m \leq K$?

Definition 2. [SCov]
INSTANCE: Collection C of subsets of finite set F, positive weight $w_R\ \forall R \in C$, and a goal $H \geq 0$.
QUESTION: Does there exist a cover $C' \subseteq C$ for F such that $\sum_{R \in C'} w_R \leq H$?

The minimum set cover [SCov] is $\mathcal{NP}$-complete [14]. First we note that [DAuc] is in $\mathcal{NP}$: given a winning set $J' \subseteq J$, one can verify whether it defines a partition and the procurement cost is less than K in polynomial time. Let an instance of [SCov] be given. We construct an instance of [DAuc] in the following way:

- $G = F$, $|J| = |C|$
- Create a bid j for each of the subset $R \in C$ as follows:

$$Q_j^m = \left\{ \begin{array}{l} w_R \text{ if } m \in R \\ \infty \quad \text{otherwise} \end{array} \right\}, \quad \forall m$$

$$\theta_j^i = \frac{i-1}{i},\ 1 \leq i \leq M$$

- $K = H$

The above reduction can be clearly done in polynomial time. We now show that the reduction is valid by showing that an instance of [SCov] is a *yes* iff its reduction [DAuc] is a *yes* instance.
($\Leftarrow$) Let there exist a yes instance of [DAuc] that was generated from an instance of [SCov] using the above reduction. Then $J' \subseteq J$ defines a partition of G with procurement cost $\leq K$. A cover C' for [SCov] can be constructed as follows. For every $j \in J'$, include the corresponding subset R in C'. Note that $B_j \subseteq R$ as $m \notin R$ implies $Q_j^m = \infty$. The cost of procurement from winning bid j is given by

$$\begin{aligned}(1 - \theta_j^{|B_j|}) \sum_{m \in B_j} Q_j^m &= \left(\frac{1}{|B_j|}\right) |B_j| w_R \\ &= w_R\end{aligned}$$

Thus the cost of procurement from each bid is equal to the weight of the corresponding subset in C'. Since the winning bids partition G, the collection C' covers F with cost $\leq K = H$.
($\Rightarrow$) Let there exist a yes instance for [SCov] with cover C'. The solution to [DAuc] can be constructed as follows. For every subset $R \in C'$, include its corresponding bid j in J'. Since C' covers F, J' also covers G. If an item is supplied by more than one supplier then it can be removed from its respective suppliers except one. Note that removing an item from bid will not change the cost because of the assumed discount and cost structure. Hence, we have a partition of G that satisfies the goal.

3. Integer programming formulation and linear relaxation

Without the discount function, the WDP can be solved in $O(NM)$ time (choose the minimum bidder for each of the items). Due to the discount function, the cost of an item bought from a bid depends on the total number of items bought from that bid. Hence, the formulation should also take into account the total number of items bought from a bid. If i items are bought from a bid j, then the cost of an item m is $(1 - \theta_j^i) Q_j^m$. With two different decision variables to choose *an* item m and the *number* of items i from bid j, the cost of an item would be a nonlinear function. To linearize the objective function, we define the *effective cost* of an item m if i items are bought from bid j as

$$p_j^{im} = (1 - \theta_j^i) Q_j^m \tag{1}$$

The WDP can be restated as choosing the items with minimal sum of effective costs subject to the demand and discount constraints.

3.1. *Integer programming formulation*

Following is an integer programming (IP) formulation for the WDP.

$$\min \sum_j \sum_i \sum_m p_j^{im} w_j^{im} \text{ subject to} \quad (2)$$

$$\sum_i v_j^i \leq 1 \ \forall j \quad (3)$$

$$\sum_m w_j^{im} = i v_j^i \ \ \forall i, j \quad (4)$$

$$\sum_j \sum_i w_j^{im} = 1 \ \ \forall m \quad (5)$$

$$w_j^{im}, v_j^i \in \{0, 1\} \ \ \forall j, i, m \quad (6)$$

In the remainder of this paper, the notation $\forall j$ denotes $j = 1, \ldots, N$. Similarly, $\forall i$ and $\forall m$ denote $i = 1, \ldots, M$ and $m = 1, \ldots, M$, respectively. If the binary decision variable $w_j^{im} = 1$, then item m is bought from bid j with effective cost p_j^{im} and $w_j^{im} = 0$ denotes otherwise. If binary variable $v_j^i = 1$, then i items are bought from j. The constraint corresponding to j in (3) is a multiple choice constraint for variables v_j^i in i that determines the number of items i bought from j (supply constraint). If $\sum_i v_j^i = 0$, then no items are chosen from j and if $v_j^i = 1$, i items are bought from j. Constraints (4) ensure that the items chosen from j are consistent with their effective cost: if i items are chosen, then they have the cost with discount for i items (discount constraints). Constraints (5) ensure that every item is procured from only one supplier (demand constraints).

3.2. *Linear relaxation*

The linear programming (LP) relaxation provides a lower bound on the optimal cost, which is useful in pruning the search space in branch-and-bound algorithms. We study the LP relaxation by investigating its dual.

Let $\{\gamma_j\}$, $\{\lambda_j^i\}$, and $\{\beta^m\}$ be the dual variables corresponding to the constraints (3), (4), and (5), respectively. Dual variables for the primal bounds $\{v_j^i \leq 1\}$ and $\{w_j^{im} \leq 1\}$ are not required as they are implied by (3) and (5), respectively. The dual of the linear relaxation is given by:

$$\max \sum_m \beta^m - \sum_j \gamma_j \text{ subject to} \quad (7)$$

$$\beta^m + \lambda_j^i \leq p_j^{im} \ \ \forall j, i, m \quad (8)$$

$$-\gamma_j - i\lambda_j^i \leq 0 \ \ \forall i, j \quad (9)$$

$$\gamma_j \geq 0 \quad \forall j \quad (10)$$

A feasible solution to the above problem can be easily obtained. The nonnegative variables γ_j have negative coefficients in the objective function and hence can be equated to zero. The variables λ_j^i can be eliminated by equating to zero. Thus $\beta^m = \min_{j,i}\{p_j^{im}\}$. The θ_j^i are non-decreasing over i by assumption. Hence the p_j^{im} are non-increasing and therefore $\beta^m = \min_j\{p_j^{Mm}\}$. From this dual solution one can easily construct a feasible solution to the linear relaxation such that both are optimal solutions to their respective problems.

Proposition 1 *$\{\gamma_j = 0, \lambda_j^i = 0, \beta^m = \min_j\{p_j^{Mm}\}\}$ is the optimal dual solution and $v_j^i = 0$ for $i < M$, $v_j^M = \frac{\sum_m w_j^{Mm}}{M}$, where*

$$w_{j'}^{Mm} = \left\{ \begin{array}{l} 1 \text{ if } j' = \min\{\arg\min_j\{p_j^{Mm}\}\} \\ 0 \text{ otherwise} \end{array} \right\} \quad (11)$$

is the optimal solution to the linear relaxation.

It can be easily seen that the solutions are feasible to their respective problems. Moreover, they both have the same objective value. Hence by strong duality theorem, they are optimal solutions. The LP relaxation considers only the maximum discounted cost (with discount for M items) and allocates the items based on this minimum cost. As a consequence, it violates the discount constraint by procuring less number of items from a supplier but with a maximum discount for M items. It is worth noting that only variables $\{v_j^i\}$ may take non integer values.

The linear relaxation problem can be solved in $O(MN)$ by taking the minimum of p_j^{Mm} over j, for each m. If the $\{v_j^M\}$ are fractional, then it is not an optimal solution to the WDP. One can easily construct a feasible integer solution $(\{\overline{v}_j^m\}, \{\overline{w}_j^{im}\})$ to the WDP from the fractional $(\{v_j^M\}, \{w_j^{Mm}\})$.

(1) $\delta_j = \sum_m w_j^{Mm}$, $\forall j$; $\overline{v}_j^m = v_j^m$, $\forall j, m$; $\overline{w}_j^{im} = w_j^{im}$, $\forall j, i, m$;

(2) $\forall j$ **do**:

  (a) **if** $(v_j^M > 0)$ $\overline{v}_j^{\delta_j} \leftarrow 1$, $\overline{v}_j^M \leftarrow 0$;

  (b) $\forall m$ **if** $(w_j^{Mm} = 1)$ $\overline{w}_j^{\delta_j, m} \leftarrow 1$, $\overline{w}_j^{Mm} \leftarrow 0$;

3.3. *Valid inequalities as cuts*

A *valid inequality* for an IP problem is an inequality that is satisfied by all the feasible solutions. Valid inequalities that are not part of a formulation are essentially redundant constraints. However, they may serve as *cuts* if they are not satisfied by all feasible solutions of the LP relaxation [29]. A cut that is not satisfied by an optimal solution of the LP relaxation is called as a *violated cut*. Addition of a violated cut to the LP relaxation tightens it and provides a better bound. The formulation is changed in such a way that the LP feasible region becomes smaller but the IP feasible region is unaffected. Identification of violated cuts, adding them to the LP relaxation, and resolving them to find better bounds, can be iterated till no violated cuts can be found.

The optimal LP solution to the IP formulation had binary values for $\{w_j^{im}\}$ and fractional values for $\{v_j^i\}$. Hence the following valid inequalities serve as cuts to the LP relaxation:

$$w_j^{im} \leq v_j^i \ \ \forall j, i, m \tag{12}$$

They are obviously valid for the IP formulation and they exclude the optimal solution of the original LP relaxation. Hence, the bounds obtained with the inclusion of cuts can be expected to be tighter than the original formulation. The above family of cuts were generated by studying the LP relaxation solution. This does not involve solving separation problems and hence no algorithmic efforts involved. Further the size of the family of cuts is polynomial: $O(NM^2)$. However, these cuts are not facet defining and hence not strong enough to obtain the integer hull of the feasible set.

The optimal solution of the LP relaxation with cuts satisfies the following properties:

(1) If $v_j^i > 0$, then number of non-zero w_j^{im}s are greater than or equal to i.
(2) The $\{w_j^{im}\}$ take binary values only if $\{v_j^i\}$ are binary.

The first property is a direct consequence of the valid inequalities (12) and the constraint set (4). The second property follows from the first.

4. Branch-and-Cut

Branch-and-cut [25] is a generalization of branch-and-bound (B&B), which includes the cut routine to identify and add violated cuts. B&B is an exact intelligent enumerative technique that attempts to avoid enumerating a large portion of the feasible integer solutions [6]. It is a widely used approach for solving discrete optimization, combinatorial optimization, and integer programming problems in general. The B&B approach first partitions the overall set of feasible solutions into two or more sets and as the algorithm proceeds the set is partitioned into many simpler and smaller sets, which are explored for the optimal solution. Each such set is represented algebraically by a *candidate problem* (CP). A typical iteration of B&B consists of:

- **Selection/Removal** of a CP from the list of CPs
- Determining the **lower bound** of the selected CP
- **Fathoming** or **pruning**, if possible, the selected CP
- Determining and updating the **incumbent** solution, if possible
- **Branching strategy**: If the CP is not fathomed, branching creates subproblems which are added to the list of CPs

The algorithm first starts with the original problem as the only CP in the list, considering the entire feasible set of solutions. As the algorithm proceeds, numerous CPs are added to the list, each containing a set of feasible solutions. The CPs partition the search space and at every iteration, a prospective CP is chosen to search for the optimal solution. The CP though containing less number of solutions than the original problem, could still be hard to solve, and hence a easily solvable lower bounding technique is applied to obtain a good lower bound on the objective value. This lower bound is for the solutions in that particular CP. If a feasible solution had been obtained so far in the algorithm, it can be used to prune a CP with lower bound greater than the cost of the known solution. A CP can be fathomed (removed from further search) if the best solution in that CP is found. If not fathomed, it is then split into smaller CPs and added to the list of CPs. As the algorithm proceeds, the best known feasible solution is maintained and when the list of CPs become empty, the best known feasible solution is the optimal solution.

Although the B&B technique is easy to understand, the implementation for a particular problem is a non-trivial task [6] requiring:

- An efficient lower bounding technique that can be solved with less computational efforts and also guarantee a tight lower bound
- Efficient data structures for handling the rather complicated book-keeping of the list of CPs,
- Clever strategies for selecting promising CPs, and
- Branching strategies that could effectively prune the enumeration tree.

The generation of tighter bounds with addition of violated cuts helps in pruning the B&B nodes, thereby reducing the search space. In branch-and-cut (B&C), the LP relaxation is repeatedly solved with addition of new cuts at a B&B node. For a detailed exposure, see [7,26]. Cuts can be added to the B&B tree in various ways, leading to different algorithms.

Cut-and-branch (C&B) is a B&C variant where a family of cuts are added to the formulation and B&B is applied to the modified formulation. This technique is useful if generation of cuts are easier and are known a priori (as is the case in our problem). It is advantageous as the cuts added are valid throughout the tree and no further cuts are required to be added. Usually the number of cuts added are very large and many of them may not be useful. With large number of constraints, the time taken to solve the LP relaxation can increase considerably. An alternate approach is to add the cuts that are only violated by the current LP solution. In this way, cuts are progressively added. There are two possible approaches here: B&C-*global*, in which cuts added are valid throughout the search tree and B&C-*local*, where the cuts added are valid only to the subtree of the current node.

Following cut addition techniques are considered in this paper:

Cut-and-Branch (C&B) : All the NM^2 cuts ($w_j^{im} \leq v_j^i$, $\forall j, i, m$) are added to the IP formulation at the root node.

Branch-and-Cut Global-w (B&C-Gbl-w) : After each simplex iteration (that solves the LP relaxation), only those variables that violate the inequality $w_j^{im} \leq v_j^i$ induce the corresponding cuts and are added globally to all the nodes in the search tree.

Branch-and-Cut Global-V (B&C-Gbl-V) : After each simplex iteration, if an w_j^{im} violates the inequality $w_j^{im} \leq v_j^i$ then cuts for all items for that j and i are added as global cuts.

Branch-and-Cut Local-w (B&C-Loc-w) : Same as **B&C-Gbl-w**, except that the cuts are local cuts.

Branch-and-Cut Local-V (B&C-Loc-V) : Same as **B&C-Gbl-V**, except that the cuts are local cuts.

5. Branching techniques

The conventional branching technique is *variable dichotomy*. If the LP relaxation provides a solution with non-integer values for integer variables, then one such variable is chosen and CPs are created by imposing bounds on the variable. According to the properties of the optimal LP solution with cuts, either all variables are binary (optimal to the IP) or many variables are fractional. The variable dichotomy branching is to chose a particular fractional v_j^i or w_j^{im} to create two CPs by imposing the variable to equal to 0 and 1, respectively. The branching on a v_j^i is more generic than that on a w_j^{im}. The former splits the solution space based on the number of items supplied by a bid, whereas the latter is more specific about an item, supplied by a particular bid that supplies a certain number of other items. In this paper, we propose a novel heuristic branching technique called as *branch-on-price* (BoP) to create candidate problems by branching on the price of an item, which is fractionally supplied by more than one supplier.

Let w_j^{im} be fractional. Due to the constraint (5), there exist at least one another $w_{j'}^{i'm}$ with $i \neq i'$, which is fractional. Let β^m be the price of the item m as dictated by the LP solution:

$$\beta^m = \sum_j \sum_i p_j^{im} w_j^{im} \tag{13}$$

We create two CPs, CP- and CP+, by branching on the above price. The CP- is created by adding the following constraints:

$$w_j^{im} = 0 \text{ if } p_j^{im} \geq \beta^m, \ \forall j, i, m \tag{14}$$

The CP+ is created by adding constraints

$$w_j^{im} = 0 \text{ if } p_j^{im} < \beta^m, \ \forall j, i, m \tag{15}$$

The two CPs partition the IP feasible solution space. The optimal LP solution (which is infeasible) does not belong to the solution space of the relaxations of the either of the CPs. To facilitate this branching, we represent a CP by using bounds on the prices of each of the items.

The CP is compactly represented by using an allowable price range $[\underline{\beta}^m, \overline{\beta}^m)$ for each item m. Algebraically, this is achieved by imposing the following bounds in the IP formulation: $w_j^{im} = 0$ if p_j^{im} is outside the above range. The initial CP contains all the solutions and hence $\underline{\beta}^m = \min_j\{p_j^{Mm}\}$ and $\overline{\beta}^m = \max_j\{p_j^{1m}\} + \epsilon$, for some $\epsilon < 0$. If an item m is chosen for branching with price β^m, then CP- has $[\underline{\beta}^m, \beta^m)$ and CP+ has $[\beta^m, \overline{\beta}^m)$, as the respective price range for m. This is illustrated in Figure 1. Note that the price range of other items will remain the same as that of the parent CP.

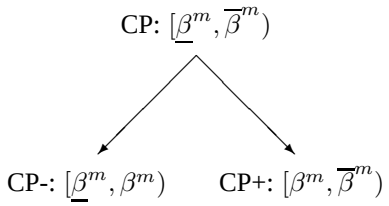

Fig. 1. Branching Strategy of BoP

The above branching scheme imposes many variables to be zero, across several bids, rather than just fixing one variable to 0 or 1, as in the variable dichotomy branching. Further, such a branching is more meaningful in terms of the WDP. The β_m can be considered as the price of the item m and the BoP algorithm is searching for the optimal price from the set of $\{p_j^{im}\}$, subject to the discount and demand constraints. Violation of the discount and the demand constraints leads β^m to be a convex combination of some of the prices from the set $\{p_j^{im}\}$. The proposed branching scheme partitions the set such that the same convex combination cannot be encountered again, thereby removing the violation in the constraints. In this way, one can expect that the algorithm will converge quickly towards the optimal prices. It is worth noting that even though the β^m is a real number and thus the branching could be infinitely divisible, the possible optimal values it can take is NM and hence the number of branches is finite.

If there are more than one item which have fractional allocations, the algorithm has to choose one item to branch. Let β^m be the price of item m defined by the convex combination (13) and B^m be the set of bids that partially supply item m. The item m' to branch on is chosen by one of the following rules:

$$\mathbf{BoP1}: \; m' = \arg\min_m \{\beta^m - \underline{\beta}^m : \beta^m > \underline{\beta}^m, |B^m| > 1\} \quad (16)$$

$$\mathbf{BoP2}: \; m' = \arg\min_m \{\overline{\beta}^m - \beta^m : \beta^m > \underline{\beta}^m, |B^m| > 1\} \quad (17)$$

$$\mathbf{BoP3}: \; m' = \arg\max_m \{|B^m| : \beta^m > \underline{\beta}^m \, |B^m| > 1\} \quad (18)$$

BoP1 chooses the item with price closest to its lower bound, BoP2 chooses the item with price closest to its upper bound, and BoP3 chooses the item with maximum number of allocated bids. In all the three rules, the branching item is chosen such that its price is strictly greater than its lower bound. According to the rules of the creation of CPs, branching on an item with $\beta^m = \underline{\beta}^m$ will create an infeasible CP- with range $[\beta^m, \beta^m)$ and CP+ with range $[\beta^m, \overline{\beta}^m)$, which is same as its parent CP. To avoid an infinite loop, only items $\beta^m > \underline{\beta}^m$ are considered. However, a pathological case is encountered when all items having their prices equal to their respective lower bounds. In this situation, we chose an item m randomly and create only CP+ with range $[\beta^m + \epsilon, \overline{\beta}^m)$. The $\epsilon > 0$ is chosen such that it is small enough to exclude just the price β^m. Note that there is no CP- created and with this new CP+ creation, it is possible that a feasible IP solution with prices $\{\beta^m\}$ might be excluded in the search, thus not guaranteeing optimality.

Not Strictly Convex

Let there exist a non-integer solution such that $\beta^m = \underline{\beta}^m$ if $|B^m| > 1$. The cost of this solution is $\sum_m \beta^m$ and it is possible that there exists an integer solution with the same cost. Consider the LP solution shown in Figure 2 as a transportation network. A link between a bid and an item denotes the allocation. The *(Supply)* denotes the number of items that the bid *should* supply. The option is to either accept the bid with a supply of three items or reject the bid entirely. Note that modifying the supply will result in change of discount and hence in the change of solution cost. The solution shown in the figure is infeasible as each item is supplied by more than one bid ($w_j^{3,m} = 1/3$, $\forall j, m$). However, if allocation from any two bids are removed then it is a feasible IP solution with the same cost as that of the LP solution. It can be easily seen that the LP solution in Figure 3 ($w_j^{2,m} = 1/2$, $\forall j, m$) has no feasible IP solution. Thus when a LP solution is encountered with no item to branch on, it is required to find a feasible IP solution with the same cost or prove that no such solution exists.

Let $\mathcal{M}$ be the set of items with $|B^m| > 1$ and $\beta^m = \underline{\beta}^m$. Note that the rest of the items will have $|B^m| = 1$ and hence satisfy the integrality constraints. Let I_j be the set of items that are being supplied by bid j with allowed supply in the range $[\underline{S}_j, \overline{S}_j]$. The allowed supply range is determined by the discounts at which the current partial allocation is made. If $\underline{S}_j < \overline{S}_j$ then

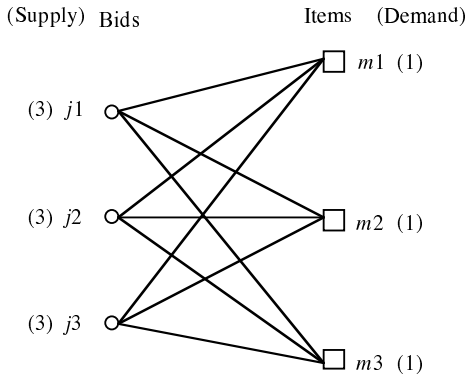

Fig. 2. With feasible IP solution

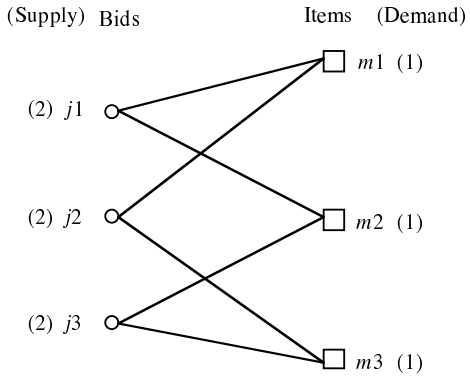

Fig. 3. With no feasible IP solution

$\theta_j^{\underline{S}_j} = \theta_j^{\overline{S}_j}$. Let $\mathcal{J}$ be the set of bids that supply items partially. Given such a solution, one has to find a feasible IP solution with the following properties:

- $|B^m| = 1,\ m \in \mathcal{M}$
- $|I_j| \in \{0\} \cup [\underline{S}_j, \overline{S}_j],\ j \in \mathcal{J}$

The second property ensures that either the bid supplies in the allowable range such that the cost of the solution is not altered or it is rejected. This is a *capacitated facility location problem* [10], where the bids are the facilities that can be opened or closed and if opened their capacity is in the allowable supply range. Though the generic facility location problem is $\mathcal{NP}$-hard, we present here a recursive search algorithm exploiting the price structure in the LP solution.

Consider an item $m' \in \mathcal{M}$ and let $j' \in B^{m'}$ be chosen to supply this item. This allocation results in the following sequence of allocations:

(1) Other bids in $B^{m'}$ have to be removed: $B^{m'} \leftarrow B^{m'} \setminus \{j\},\ j \neq j' \in B^{m'}$

(2) Item m' is removed from corresponding I_j: $I_j \leftarrow I_j \setminus \{m'\}$

(3) If removal of m' from I_j violates the supply constraints, then it is removed:

$$|I_j| < \underline{S}_j \Rightarrow B^m \leftarrow B^m \setminus \{m\},\ m \in I_j$$
$$\Rightarrow I_j \leftarrow \emptyset$$

The above sequence will result in one of the following cases:

(1) A feasible IP solution.

(2) A infeasible IP solution with any of $|B^m| = 0$.

(3) At least there exists one m such that $|B^m| > 1$.

If a feasible solution is encountered then the search can be stopped. If an infeasible solution is encountered, then m' cannot be supplied by j' and hence j' can be removed. In this case, the next bid from $B^{m'}$ is considered for allocation. For case 3, a bid from B^m is chosen and allocated to m and the search proceeds iteratively. If all bids of $B^{m'}$ were allocated and no feasible solution was found, then there exists no IP solution. This is a depth first search that can be implemented using a recursive algorithm.

The BoP branching technique is similar to special ordered set (SOS) branching [1], which fixes a set of variables as against only one variable in the case of variable dichotomy. The constraint corresponding to item m in (5) is a SOS type 1 constraint, where only one of the binary variables in $\{w_j^{im}\}_{j,i}$ can be non-zero. SOS branching is implemented in commercial optimization packages by assigning unique weights to each of the concerned variables and the weighted average LP solution value of these variables is used as a reference in branching to create mutually exclusive sets of variables [17]. In BoP, the effective price p_j^{im} is used as the weight for the variable w_j^{im} and the non-uniqueness of weights leads to the above pathological case of not strictly convex price.

6. Primal heuristic

In the generic B&B, an incumbent solution is found when the LP relaxation yields a feasible IP solution. Primal heuristic is used to find an incumbent solution from the LP relaxation. This helps in pruning the search tree, as incumbent solutions provide upper bounds. We develop a primal heuristic by exploiting the embedded network structure in the WDP. This ensures an incumbent solution at every node and thus makes the B&C algorithms *any time algorithms* that can be stopped before convergence with an available feasible solution.

6.1. *Exploitable network structure*

The WDP of DA can be considered as a transportation network with N supply nodes (bids) and M demand nodes (one for each item). Each supply node has a supply of M units and each demand node has a unit demand. The embedded transport structure is shown in Figure 4. A flow in the network connecting node j to node m indicate that bid j is supplying item m. Due to the unit demand at each demand node, the flow in any given arc will be at most one. The complicating feature of the model is the cost $c(j,m)$ of the flow in the arc (j,m), which is the function of number of units supplied from node j. Note that this is different from the conventional nonlinear cost network models, where the cost will vary based on the flow through the arc, whereas in this case the cost varies on the total flow from the supply node. Let δ_j be the total supply from node j in a solution. Then the cost of the flow in an arc (j,m) is given by $c(j,m) = (1-\theta_j^{\delta_j})Q_j^m$. The solution is feasible only if the total supply $\sum_j \delta_j = M$. It is worth noting that for a given feasible supply, determining the optimal flow is a transportation problem. Thus the problem can be solved without the integer restrictions on the flow. In terms of the IP formulation, if the binary variables $\{v_j^i\}$ are fixed in a feasible way, the $\{w_j^{im}\}$ can be easily obtained by solving a transportation problem. Once the $\{v_j^i\}$ are fixed, the discounts are known and hence the problem is easy. The binary variables $\{w_j^{im}\}$ can indeed be relaxed to take continuous values, as there will always exist an optimal solution with integer values.

6.2. *Incumbent solution from the LP solution*

Let w_j^{im} be the optimal LP solution. If all are binary, then it is a feasible solution to the IP. The following heuristic constructs a feasible solution from the fractional LP solution.

(1) (Initialize) $S_j = 0, \forall j$
(2) **do** $\forall m$: $k = \arg_j \max_{j,i}\{w_j^{im}\}$; $S_k \leftarrow S_k + 1$;
(3) Construct a transportation network with winning bids as sources and items as sinks. The source corresponding to winning bid j has a supply of $S_j > 0$ and each sink has a unit demand. The cost of flow from j to m is $p_j^{S_j,m}$. Let x_j^m be the optimal flow. Assign $V_j^{S_j} = 1$ and $w_j^{S_j,m} = x_j^m$.

The winning bid for an item m is chosen as the one with the largest w_j^{im} value. This is used to determine the number of winning items S_j for each winning bid.

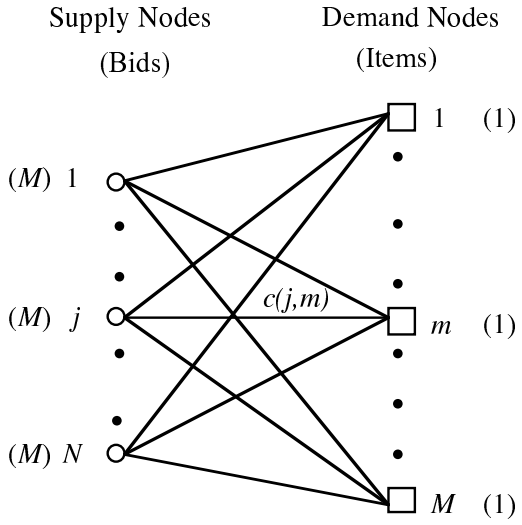

Fig. 4. Embedded network structure

This is in turn used to determine the winning items with the consistent discount prices $p_j^{S_j,m}$. This will provide a better IP solution than directly rounding the largest w_j^{im} to 1. Using this heuristic, an incumbent solution is obtained whenever a new CP is created and the best known solution is updated and stored.

7. Computational experiments

In this section, we present the results of the extensive computational experiments conducted using the various proposed algorithms across various problem types.

7.1. *Discount auction instance generator*

An instance generator called as *discount auction instance generator* (DAIG) was created to randomly generate different types of problem instances. The intention is to study the effects of the varying discount and cost structures on the computational requirements of the problem. A problem instance is defined by M, N, $\{Q_j^m\}$, and $\{\theta_j^i\}$. Given M and N, the cost and the discounts can be generated in many ways. Firstly, there are two kinds of cost differences: relative market costs across the items and for each item, relative bid costs quoted by the suppliers. We assume a normalized market cost rc^m for each item m, which is uniformly distributed in range $[\underline{rc}, 1]$ with $0 < \underline{rc} < 1$. The $\underline{rc}$ is an input parameter and rc^m are randomly chosen in the above range. At least one item is chosen to have value 1 and another $\underline{rc}$. The $\underline{rc}$ is the minimum relative cost in

the portfolio of items that are being procured. For example, to model the scenario where the procured items have a maximum of 15% relative difference in the cost, $\underline{rc} = 0.85$.

For any given item, the bid price quoted by the suppliers vary. The minimum cost quoted for an item m is captured using a parameter mc^m. The $\underline{mc}$ is the input parameter and mc^m are randomly chosen in range $[\underline{mc}, 1)$. The individual cost of m for bid j in chosen in the following way:

$$Q_j^m = \text{Random}[mc^m, 1] \ \times \ rc^m \tag{19}$$

The above costs are chosen such that at least one bid has rc^m and one has the minimum cost $mc^m \times rc^m$. As rc^m denotes the market value, it is maximum price quoted that can be quoted by the suppliers.

For generating the discount functions, a discount range is input to the DAIG. For example, if the input discount range is $[\underline{\theta}, \overline{\theta}]$, then all the maximum discounts $\{\theta_j^M\}$ are chosen randomly in this range. There is no discount for one item: $\theta_j^1 = 0$. The intermediate values are chosen according to type of the discount function. The DAIG currently supports following types of discount functions:

(1) *Linear*: $\theta_j^i = (i-1) \times \frac{\theta_j^M}{M-1}$
(2) *Marginally Decreasing*: $\theta_j^i = -\frac{\theta_j^M}{(M-1)^2} \times (M-i)^2 + \theta_j^M$
(3) *Marginally Increasing*: $\theta_j^i = \frac{\theta_j^M}{(M-1)^2} \times (i-1)^2$
(4) *Step*
(5) *Arbitrary*
(6) *Random*: The discount type for a bid j is chosen randomly from one of the above.

All functions are strictly increasing, except for the type *Step*. The *Arbitrary* is strictly increasing without any notable structure like the preceding types. The experiments were carried out on a Windows XP based PC equipped with a 2.8GHz Intel P4 processor with 1GB RAM. The algorithms were coded in Java, and for the model building and solving of LP relaxations and transportation problems in primal heuristics, ILOG Concert Technology of CPLEX 10.0 [16] was used.

7.2. *LP experiments*

The first set of experiments were conducted to study the tightness of the LP relaxation with cuts. The performance criterion is the duality gap, calculated as follows:

$$\text{Duality Gap } (\%) = \frac{\text{Optimal Value } - \text{ Relaxed Value}}{\text{Optimal Value}} \times 100 \tag{20}$$

Extensive experiments were conducted by varying the problem parameters shown in Table 1. The $\underline{rc}$ and $\underline{mc}$ were varied from low to high values. The discounts $[\underline{\theta}, \overline{\theta}]$ were chosen from three different sets of values representing close range, medium range, and high range. M and N were chosen to in two sets to study the effect of varying N for the same M and vice versa. All the six discount types currently supported by DAIG were tested.

Table 1

Parameters and values for LP experimentation

Parameter	Values
$\underline{rc}$	{0.2, 0.4, 0.6, 0.8}
$\underline{mc}$	{0.05, 0.1, ..., 0.95}
$[\underline{\theta}, \overline{\theta}]$	{[0.1, 0.2], [0.2, 0.3], ..., [0.8, 0.9]}
	{[0.1, 0.5], [0.5, 0.9]}
	{[0.1, 0.9]}
M, N	{10}, {10, 25, 50, 75, 100}
	{5, 10, 15, 20}, {30}
Discount types	Linear, marginally decreasing, marginally increasing, step, arbitrary, random

Fifty problem instances were created for each of the different combination of values of the parameters and the average duality gap for the LP relaxation with and without the cuts were calculated. Following are the main inferences from the experimentation:

(1) The LP relaxation with cuts gives significantly tighter bounds. Problem instances with duality gap as high as 70% had a duality gap of less than 1% with cuts. Indeed for all the problem instances considered, the average duality gap was less than 1% with the addition of cuts.
(2) The parameter $\underline{rc}$ had no effect on the duality gap. Thus items with varying costs or similar costs have no influence on the duality gap.
(3) The discount types did not affect the duality gap significantly for LP relaxation without cuts. However, with the addition of cuts, the discount types had the maximum average duality gap in the following increasing order: arbitrary (0.005%), marginally increasing (0.03%), linear (0.04%), random (0.3%), step (0.6%), and

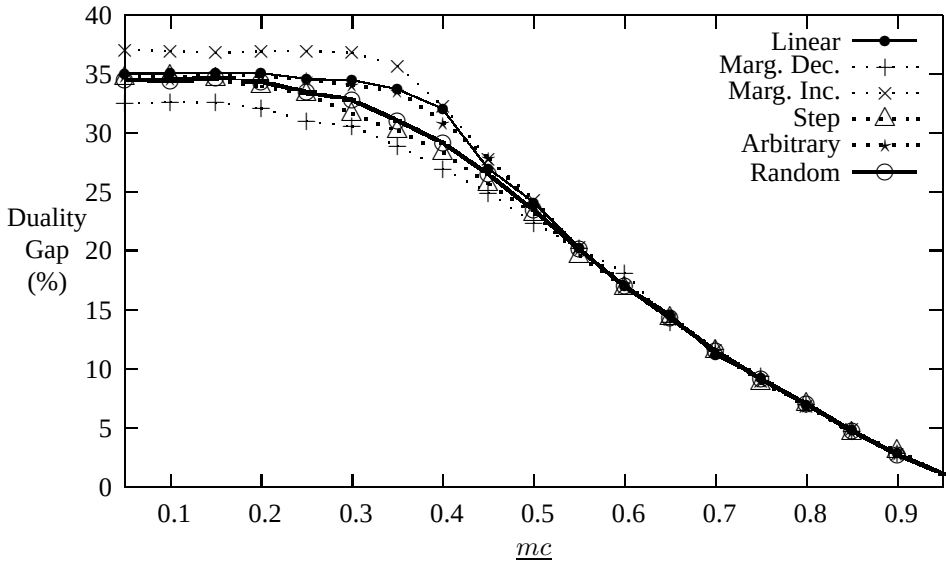

Fig. 5. LP relaxation without cuts: $N = 50,\ M = 15,\ \underline{rc} = 0.5,\ [\underline{\theta}, \overline{\theta}] = [0.3, 0.4]$

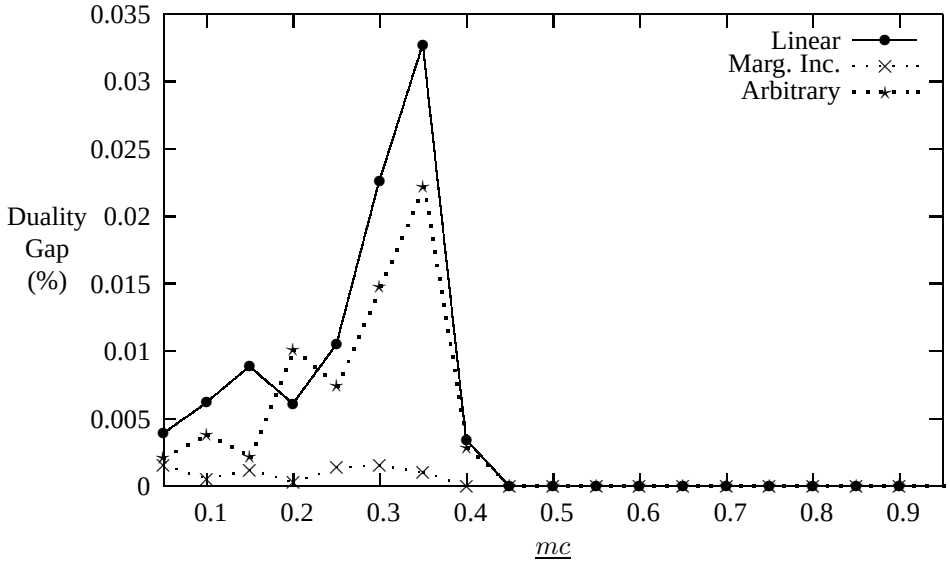

Fig. 6. LP relaxation with cuts: $N = 50,\ M = 15,\ \underline{rc} = 0.5,\ [\underline{\theta}, \overline{\theta}] = [0.3, 0.4]$

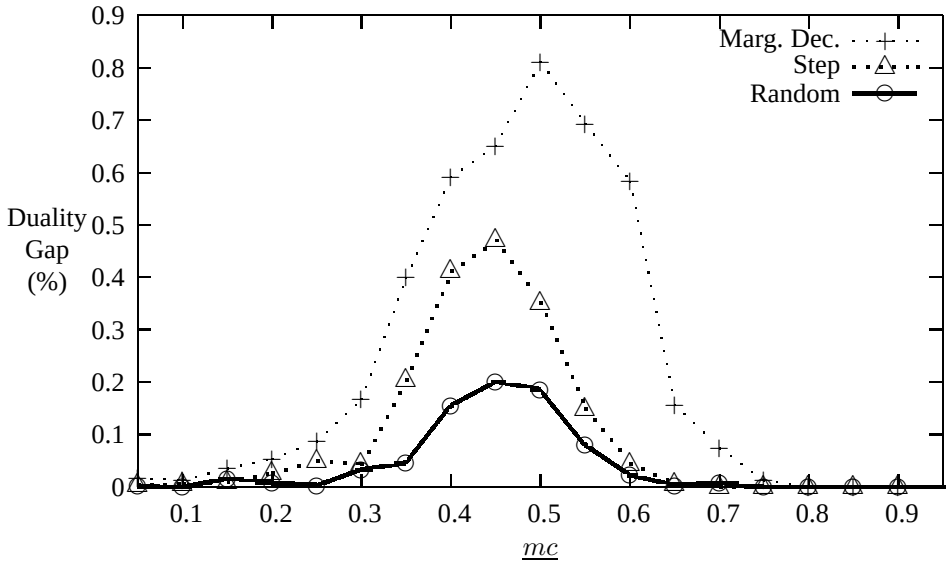

Fig. 7. LP relaxation without cuts: $N = 50$, $M = 15$, $\underline{rc} = 0.5$, $[\underline{\theta}, \overline{\theta}] = [0.3, 0.4]$

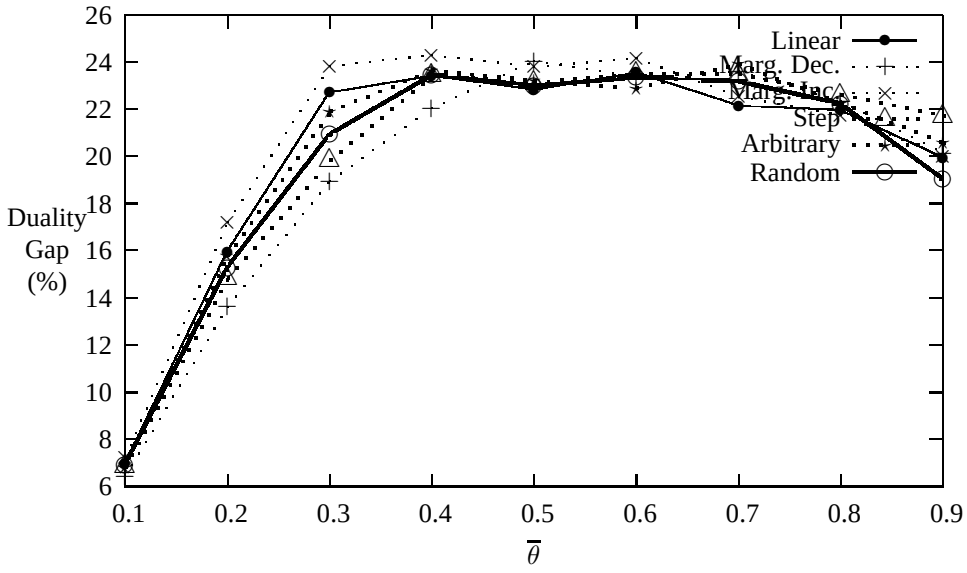

Fig. 8. LP relaxation without cuts: $N = 50$, $M = 15$, $\underline{mc} = 0.5$, $\underline{rc} = 0.5$, $\underline{\theta} = \overline{\theta} - 0.1$

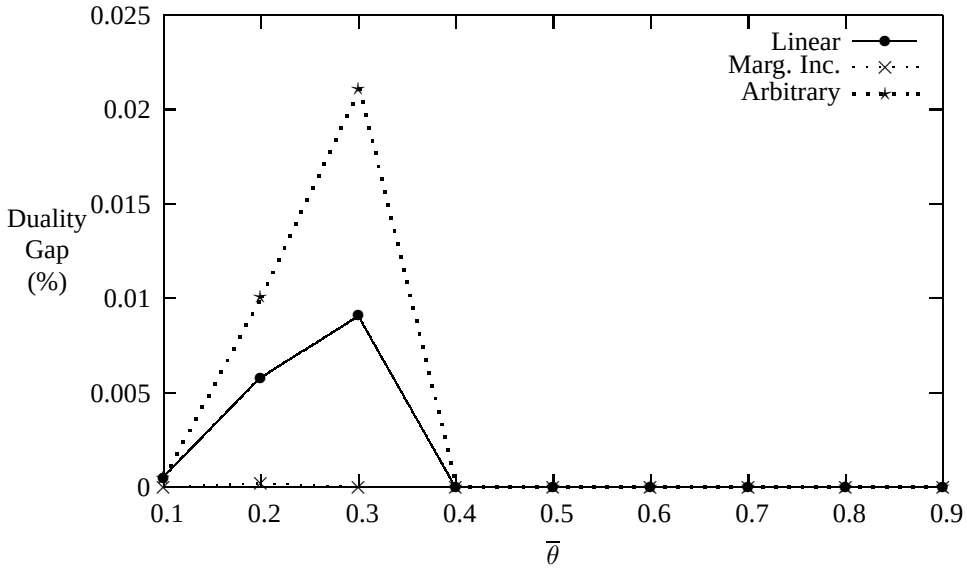

Fig. 9. LP relaxation with cuts: $N = 50$, $M = 15$, $\underline{mc} = 0.5$, $\underline{rc} = 0.5$, $\underline{\theta} = \overline{\theta} - 0.1$

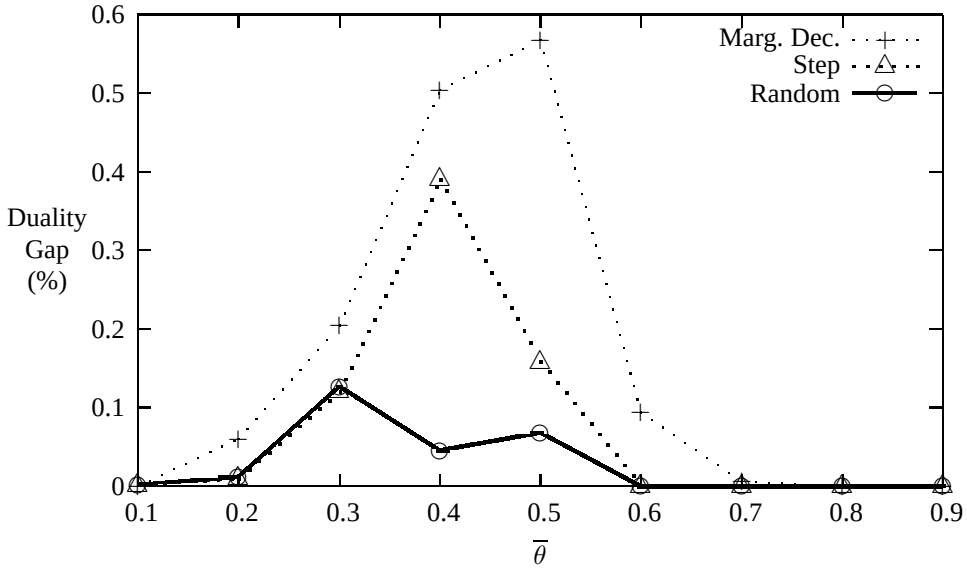

Fig. 10. LP relaxation with cuts: $N = 50$, $M = 15$, $\underline{mc} = 0.5$, $\underline{rc} = 0.5$, $\underline{\theta} = \overline{\theta} - 0.1$

marginally decreasing (0.9%). For most of the problem instances, linear, marginally increasing, and arbitrary discount types had zero duality gap.

(4) The parameter $\underline{mc}$ showed significant changes in the duality gap for both the LP relaxations with and without cuts (see figures 5 to 7). As mentioned above, the duality gap also depended on the discount types for relaxation with cuts.

(5) Significant changes in the duality gap were observed for discounts chosen in close range. As shown in figures 8 to 10, the duality gap increased with the increase of $\overline{\theta}$ for LP relaxation without cuts but showed a reverse behavior with cuts. Again, the gap depended on the discount type for relaxation with cuts.

(6) For $M = 10$, the average duality gap increased steadily with the increase of N till 50 and became negligible for $N = 75$ and almost zero for $N = 100$. Similar results were observed for varying M with fixed N. When the size of N becomes relatively larger than that of M, the duality gap is negligible as with more N, the possibility of *better* bids is high.

7.3. *Branch-and-Cut experiments*

To study the performance of the branch-and-cut, a collection of algorithms were created by combining different cut addition techniques and branching rules. The different branching techniques include the traditional variable dichotomy (Var-Dic) along with the three BoP techniques. With the above combinations of cut addition and branching techniques, twenty branch-and-cut algorithms were used for experimentation. We used best first search (BFS) as the search strategy, which explores the best CP from the current list of CPs. This reduces the search space, but it has to store all the unexplored CPs in memory. BFS is implemented by creating a *binary heap* that holds the list of CPs. At every iteration, the root of the heap, which is the CP with the least lower bound, is deleted from the queue and explored. If it is not fathomed or pruned, then two new CPs are generated and added to the queue. An incumbent solution is generated at each node from the LP solution using the proposed heuristic.

The performance parameters considered were: CPU time (in milliseconds), number of nodes explored, and number of simplex iterations. The entries in tables 2 to 6 are the triplet CPU time (ms), nodes explored, and simplex iterations. Tables 7 to 9 show the number of instances (out of 50) for which, each algorithm solved the WDP with the least solution time. With more than three hundred instances solved, following inferences were made:

- Problem instances with linear, marginally increasing, and arbitrary discount types required no branching and were solved at the root node itself. Even for problem instances with non-zero duality gap, the proposed primal heuristic found the optimal solution at the root node.
- For the rest of the discount types, the nodes explored and time taken were purely instance specific and was not much influenced by the parameter values. Tables 2 and 3 show the widely varying performance of two problem instances with the same set of parameter values.
- The *not strictly convex price* case was encountered only for the problems with step discount function.
- Every problem instance was solved using the commercial solver CPLEX using the IP formulation with cuts. CPLEX uses branch-and-cut with built pre-processing, cut generation routines, advanced branching techniques, and primal heuristics [17]. For most of the problem instances, the minimum time taken by the proposed branch-and-cut algorithms (shown in bold face in the tables) was less than that of CPLEX.
- Performance of algorithms with local cuts was poor for all the instances.
- In almost all the cases, BoP branching techniques (in particular BoP1 and BoP3) showed better performance than variable dichotomy.
- Cut-and-branch is computationally better than the other cut addition techniques, though in few cases global cuts showed superior performance.

8. Conclusions

In this work, we proposed a collection of branch-and-cut algorithms for the winner determination problem in discount auctions. The algorithms use valid inequalities that are added as cuts to the IP formulation. The valid inequalities that are added as cuts are not facet-defining. However, the size of the entire family of the cuts is polynomial (NM^2) and can be determined directly from the LP solution. Computational experiments showed that the duality gap is considerably less and hence the algorithm is many-folds faster than that without cuts. A novel branching technique called as BoP (*branch-on-price*) was proposed. It branches on the price of an item that is partially allocated to more than one supplier.

Table 2

CPU Time (ms), Nodes Explored, and Simplex Iterations for Marginally Decreasing

	Var-Dic	**BoP1**	**BoP2**	**BoP3**
C&B	69437, 12, 25	**14262, 1, 3**	21923, 2, 5	41865, 4, 9
B&C-Gbl-w	76225, 19, 593	19910, 1, 316	22985, 2, 351	36735, 4, 444
B&C-Gbl-V	149721, 28, 210	14844, 1, 107	18292, 2, 119	39076, 4, 148
B&C-Loc-w	428295, 21, 10308	34547, 1, 692	56266, 2, 1190	118037, 4, 2312
B&C-Loc-V	182156, 12, 2622	22655, 1, 269	41894, 2, 431	84364, 4, 860

$N = 75$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$, CPLEX Time = 38183

Table 3

CPU Time (ms), Nodes Explored, and Simplex Iterations for Marginally Decreasing

	Var-Dic	**BoP1**	**BoP2**	**BoP3**
C&B	273008, 63, 127	122619, 13, 27	193618, 17, 35	128974, 15, 31
B&C-Gbl-w	187098, 66, 673	**77276, 13, 431**	119650, 18, 513	82121, 15, 489
B&C-Gbl-V	249583, 66, 281	118355, 13, 158	182611, 18, 178	132086, 15, 166
B&C-Loc-w	1673004, 79, 40747	302033, 13, 6153	396731, 18, 8226	404872, 15, 6641
B&C-Loc-V	1313137, 79, 16811	250748, 13, 2818	341685, 17, 3735	299482, 15, 2918

$N = 75$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$, CPLEX Time = 34174

Table 4

CPU Time (ms), Nodes Explored, and Simplex Iterations for Step

	Var-Dic	**BoP1**	**BoP2**	**BoP3**
C&B	14171, 8, 17	8249, 1, 3	8281, 1, 3	**7484, 1, 3**
B&C-Gbl-w	11499, 4, 238	10499, 1, 220	10484, 1, 220	10874, 1, 251
B&C-Gbl-V	9577, 4, 98	9093, 1, 91	9062, 1, 91	7733, 1, 94
B&C-Loc-w	69401, 4, 1696	23545, 1, 539	23404, 1, 539	23795, 1, 550
B&C-Loc-V	133146, 10, 1784	18490, 1, 252	18396, 1, 252	19351, 1, 267

$N = 50$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$, CPLEX Time = 19904

Table 5

CPU Time (ms), Nodes Explored, and Simplex Iterations for Random

	Var-Dic	**BoP1**	**BoP2**	**BoP3**
C&B	2048, 2, 5	1767, 1, 3	1766, 1, 3	**1720, 1, 3**
B&C-Gbl-w	3111, 3, 190	2830, 1, 177	2876, 1, 177	2783, 1, 176
B&C-Gbl-V	3126, 3, 81	2877, 1, 75	2892, 1, 75	2861, 1, 75
B&C-Loc-w	19026, 3, 1176	8082, 1, 460	8286, 1, 460	7754, 1, 458
B&C-Loc-V	18072, 3, 501	7832, 1, 206	7879, 1, 206	7520, 1, 202

$N = 50$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$, CPLEX Time = 2735

Table 6

CPU Time (ms), Nodes Explored, and Simplex Iterations for Random

	Var-Dic	BoP1	BoP2	BoP3
C&B	4422, 5, 11	17281, 7, 15	22563, 7, 15	**3516, 1, 3**
B&C-Gbl-w	4875, 5, 160	20093, 7, 296	15562, 7, 257	4672, 1, 152
B&C-Gbl-V	4547, 5, 96	18953, 7, 145	15125, 7, 121	4219, 1, 88
B&C-Loc-w	43234, 5, 1570	66891, 7, 2056	57422, 7, 2046	12937, 1, 424
B&C-Loc-V	41109, 5, 847	59500, 7, 1249	58281, 7, 1316	11391, 1, 228

$N = 75$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$, CPLEX Time = 8125

Table 7

Number of instances with least solution time for Marginally Decreasing

	Var-Dic	BoP1	BoP2	BoP3
C&B	7	6	4	4
B&C-Gbl-w	0	1	3	0
B&C-Gbl-V	6	**8**	4	7
B&C-Loc-w	0	0	0	0
B&C-Loc-V	0	0	0	0

$N = 50$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$
No. of instances for which CPLEX took the least time = 8

Table 8

Number of instances with least solution time for Step

	Var-Dic	BoP1	BoP2	BoP3
C&B	7	8	3	**12**
B&C-Gbl-w	1	0	1	0
B&C-Gbl-V	5	5	3	5
B&C-Loc-w	0	0	0	0
B&C-Loc-V	0	0	0	0

$N = 50$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$
No. of instances for which CPLEX took the least time = 7

The formulation also has a 0-1 knapsack structure (22) with GUB (generalized upper bound) constraints (21). In particular, the following set of constraints constitute a multiple choice knapsack structure.

$$\sum_i v_j^i \leq 1 \ \forall j \tag{21}$$

$$\sum_j \sum_i i v_j^i = M \tag{22}$$

Generation of valid cover inequalities for 0-1 knapsacks with GUB constraints were studied in [28]. The generation of cuts would involve solving the hard separation problem, either optimally or approximately. It is worth investigating the hardness of the separation problem for the above constraints and studying the trade-offs of investing resources in generation of the cover inequalities against the savings in the convergence of the algorithm.

Table 9

Number of instances with least solution time for Random

	Var-Dic	BoP1	BoP2	BoP3
C&B	5	1	7	6
B&C-Gbl-w	0	0	0	0
B&C-Gbl-V	9	**12**	7	3
B&C-Loc-w	0	0	0	0
B&C-Loc-V	0	0	0	0

$N = 50$, $M = 15$, $\underline{mc} = 0.4$, $\underline{rc} = 0.3$, $[\underline{\theta}, \overline{\theta}] = [0.2, 0.3]$
No. of instances for which CPLEX took the least time = 8

References

[1] E. M. L. Beale and J. J. H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10(1):52–69, 1976.

[2] D. R. Beil and L. M. Wein. An inverse-optimization-based auction mechanism to support a multiattribute RFQ process. *Management Science*, 49(11):1529–1545, 2003.

[3] M. Bichler, A. Davenport, G. Hohner, and J. Kalagnanam. Industrial procurement auctions. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auction*, chapter 23. MIT Press, 2006.

[4] M. Bichler, M. Kaukal, and A. Segev. Multi-attribute auctions for electronic procurement. In *Proceedings of the First IBM IAC Workshop on Internet*

Based Negotiation Technologies, pages 18–19, Yorktown Heights, NY, USA, 1999.

[5] T.S. Chandrashekar, Y. Narahari, Charlie Rosa, Devadatta Kulkarni, and Jeffrey Tew. Auction based mechanisms for electronic procurement. Technical report, Electronic Enterprises Laboratory, Department of Computer Science and Automation, Indian Institute of Science, July 2005.

[6] V. Chandru and M. R. Rao. Integer programming. In M. J. Atallah, editor, *Handbook of Algorithms and Theory of Computing*, pages 32.1–32.45. CRC Press, 1999.

[7] C. Cordier, H. Marchand, R. Laundy, and L.A. Wolsey. bc-opt: A branch-and-cut code for mixed integer programs. Papers 9778, Catholique de Louvain - Center for Operations Research and Economics, 1997. available at http://ideas.repec.org/p/fth/louvco/9778.html.

[8] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. MIT Press, Cambridge, 2005.

[9] A. Davenport and J. Kalagnanam. Price negotiations for procurement of direct inputs. Research Report RC 22078, IBM Research, Yorktown Heights, NJ, USA, 2001.

[10] Z. Drezner and H. W. Hamacher, editors. *Facility Location: Applications and theory*. Springer-Verlag, Berlin, 2002.

[11] W. Elmaghraby. Auctions and pricing in e-marketplaces. In David Simchi-Levi, S. David Wu, and Z. Max Shen, editors, *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, International Series in Operations Research and Management Science. Kluwer Academic Publishers, Norwell, MA, 2004.

[12] W. Elmaghraby and P. Keskinocak. Technology for transportation bidding at the home depot. In *Practice of Supply Chain Management: Where Theory and Practice Converge*. Kluwer Academic Publishers, 2003.

[13] M. Eso, S. Ghosh, J. Kalagnanam, and L. Ladanyi. Bid evaluation in procurement auctions with piece-wise linear supply curves. Research Report RC 22219, IBM Research, Yorktown Heights, NJ, USA, 2001.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, 1979.

[15] Gail Hohner, John Rich, Ed Ng, Grant Reid, Andrew J Davenport, Jayant R Kalagnanam, Soo Ho Lee, and Chae An. Combinatorial and quantity discount procurement auctions provide benefits to mars, incorporated and to its suppliers. *INTERFACES*, 33(1):23–35, 2003.

[16] ILOG CPLEX. *ILOG Concert Technology 1.1 User's Manual*, February 2001.

[17] ILOG CPLEX. *ILOG CPLEX 10.0 User's Manual*, January 2006.

Received 19 March, 2007; revised 24 May 2007; accepted 29 May 2007

[18] S. Kameshwaran and L. Benyoucef. Branch on price: A fast winner determination algorithm for discount auctions. In *Proceedings of the Second International Conference on Algorithmic Aspects in Information and Management (AAIM'06), June 20-22, Hong Kong (China)*, Lecture Notes in Computer Science 4041, pages 375–386. Springer, 2006.

[19] S. Kameshwaran, L. Benyoucef, and X. Xie. Discount auctions for procuring heterogeneous items. In *Proceedings of Seventh International Conference on Electronic Commerce (ICEC 2005), August 15-17, Xi'An (China)*, pages 244–249, 2005.

[20] S. Kameshwaran, L. Benyoucef, and X. Xie. Winner determination in discount auctions. In *Proceedings of Workshop on Internet and Network Economics (WINE 2005), December 15-17, Hong Kong (China)*, Lecture Notes in Computer Science 3828, pages 868 – 877. Springer, 2005.

[21] S. Kameshwaran and Y. Narahari. e-Procurement using goal programming. In *Proceedings of 4th International Conference on Electronic Commerce and Web Technologies - EC-Web 2003*, Lecture Notes in Computer Science 2738, pages 6 – 15. Springer, 2003.

[22] S. Kameshwaran and Y. Narahari. A Lagrangian heuristic for bid evaluation in e-Procurement auctions. In *Proceedings of 2005 IEEE International Conference on Automation Science and Engineering*, pages 220–225, 2005.

[23] A. Kothari, D. C. Parkes, and S. Suri. Approximately strategy proof and tractable multi-unit auctions. In *Proceedings of ACM Conference on Electronic Commerce (EC-03)*, 2003.

[24] D. Lehmann, R. Müller, and T. Sandholm. The winner determination problem. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auction*, chapter 12. MIT Press, 2006.

[25] J. E. Mitchell. Integer programming: Branch-and-cut algorithms. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization Vol. 2*, pages 519–525. Kluwer Academic Publishers, 2001.

[26] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P.M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press, 2002.

[27] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005.

[28] L. A. Wolsey. Valid inequalities for 0-1 knapsacks and MIPs with generalized upper bound constraints. *Discrete Applied Mathematics*, 29:251–261, 1988.

[29] L. A. Wolsey. *Integer Programming*. John Wiley and Sons, New York, 1998.