조합 경매에서의 최적 분배를 위한 빠른 알고리즘

(A Rapid Algorithm for Optimal Allocation in Combinatorial Auctions)

송 진 우 † 양 성 봉 ††
(Jin-Woo Song) (Sung-Bong Yang)

요 약 조합 경매에서는 구매자들이 원하는 상품들을 조합해서 입찰할 수 있다. 그러나 판매자의 이익을 최대로 하는 구매자들, 즉 조합 경매의 승자를 결정하는 문제는 NP-complete이다. 적절한 시간 내에 승자를 찾기 위해서 분기 한정법(branch-and-bound)을 사용할 때, 구매자들의 비드(bid)들 중에서 어떤 비드를 선택해서 분기할 것인가를 결정해야 한다. 이 때, 비드를 선택하는 휴리스틱이 분기 한정법의 성능을 결정하는 중요한 역할을 한다.

본 논문에서는 조합경매의 승자를 결정하기 위해서 분기 한정법과 Linear Programming(LP)를 사용하는 알고리즘을 설계하고, 분기할 비드를 선택하기 위하여 분기할 비드와 충돌하는 비드들을 동시에 고려하는 비드 선택 휴리스틱을 제안한다. 그리고 최대 한계치를 재사용하는 경우를 찾아내서 알고리즘의 수행 시간을 줄였다.

알고리즘의 수행 성능을 평가하기 위해서 다섯 가지의 데이타 분포에 대한 실험 결과를 이전 논문들과 비교했다. 제시한 휴리스틱을 사용한 알고리즘은 두 가지 데이타 분포에서는 더 빠른 성능을 보였고 나머지 세 분포에서는 비슷한 성능을 보였다.

키워드 : 전자 상거래, 조합 경매, 분기 한정법, 선형 계획법

Abstract In combinatorial auctions buyers may bid for arbitrary combinations of goods. But determining the winners of combinatorial auctions who maximize the profit of a seller is known to be NP-complete. A branch-and-bound method can be one of practical algorithm for winner determination. However, bid selection heuristics play a very important role in the efficiency of a branch-and-bound method.

In this paper, we designed and implemented an algorithm which used a branch-and-bound method and Linear Programming for winner determination in combinatorial auctions. We propose new bid selection heuristics which consider a branching bid and conflicting bids simultaneously to select a branching bid in the algorithm. In addition, upper bounds are reused to reduce the running time in specific cases.

We evaluated the performance of the algorithm by experiments with five data distributions and compared our method with others. The algorithm using heuristics showed a superior efficiency in two data distributions and a similar efficiency in three distributions.

Key words : Electronic commerce, Combinatorial auction, Branch and Bound, Linear programming

1. 서 론

경매(auction)는 판매자의 상품들을 구매자들이 가격 경쟁을 통해서 상품들을 구매하는 시장의 역할을 한다. 요즈음에는 인터넷 경매가 보편화되면서 다양한 경매 이론에 대한 연구가 컴퓨터 분야에서 진행되고 있으며, 또한 자원을 효율적으로 분배하는 경매 방식이 전자상거래에도 적용이 되고 있다.

경매에서 거래되는 상품들 중에는 상보성(complementarity)이 존재하는 경우가 있다. 예를 들어 상품 a,

† 학생회원 : 연세대학교 컴퓨터과학과
fantaros@cs.yonsei.ac.kr
†† 비 회 원 : 연세대학교 컴퓨터과학과 교수
yang@cs.yonsei.ac.kr
논문접수 : 2002년 9월 18일
심사완료 : 2003년 5월 21일

b의 가치가 각각 $p(a)$, $p(b)$이고, a와 b를 하나의 상품으로 묶었을 때의 가치가 $p(a,b)$이다. 이때 $p(a,b) > p(a) + p(b)$라면 상품 a와 b사이에는 상보성이 있다고 한다. 이러한 상보성을 경매에 반영하기 위해서는 여러 개의 상품을 조합해서 입찰을 할 수 있는 조합 경매(combinatorial auction)가 적합하다[1-4]. 조합 경매는 현실적으로 여러 분야에 응용이 된다. FCC(Federal Communications Commission) 스펙트럼 경매와 airport time slot 경매[5]같은 예를 들 수 있다.

그러나 조합 경매에서 판매자의 최대 이익을 보장하는 구매자를 선택하는 문제는 NP-complete로 최악의 경우 계산 복잡도가 지수시간이 되어 많은 수행 시간이 걸린다[1]. 그래서 이 문제를 풀기 위해서 소요되는 수행 시간을 줄이기 위한 연구가 많이 진행되고 있다. Andersson[6]은 조합 경매에서 승자를 결정하기 위한 문제를 Mixed Integer Programming(MIP)으로 변환시킬 수 있음을 보이고, MIP 문제를 푸는데 있어서 상용 프로그램 중에서 가장 좋은 성능을 보이는 것으로 알려진 ILOG의 CPLEX를 사용한 결과를 이전의 다른 결과들과 비교하여 CPLEX의 성능이 우수함을 보였다. CABOB[7]은 분기 한정법을 사용하여 최적값을 보장하는 한편, LP로 최대 한계치(upper bound)를 구하고 여러 가지의 성능 향상을 위한 기법들을 사용한 알고리즘이다. CABOB의 성능을 평가하기 위해서 동일한 데이타들에 대한 CPLEX의 수행 시간을 비교했고, 그 결과 CABOB은 일부 데이타 분포에서는 보다 빠른 수행 시간을 보였으나 그 차이가 크지 않았고 나머지 데이타 분포에서는 CPLEX보다 상당히 좋지 않은 수행 시간을 나타냈다.

본 논문에서는 CABOB과 마찬가지로 LP와 분기 한정법을 사용한 최적 분배 알고리즘으로 조합 경매의 최적해를 구하고자 한다. 알고리즘의 성능을 향상시키기 위하여 기존의 분기하는 비드를 선택하는 방법들과는 다른 새로운 선택 휴리스틱을 제안하였고 최대 한계치를 재사용하는 경우를 통해 LP의 계산 회수를 줄이는 방법을 사용했다. 제안된 휴리스틱과 최대 한계치 재사용 기법이 사용된 알고리즘의 성능을 평가하기 위해서 CPLEX의 결과와 비교하였다.

2장에서는 조합 경매의 문제를 해결하기 위한 관련 연구에 대해 알아보고, 3장에서는 조합 경매의 정의에 대해 알아본다. 4장에서는 최적해를 찾기 위한 알고리즘과 제안하는 분기 선택 휴리스틱 및 최대 한계치 재사용 기법에 대해 기술한다. 5장에서는 실험 결과를 분석하고 6장에서는 결론과 향후 과제를 제시한다.

2. 관련연구

여러 종류의 상품들이 경매에서 취급될 때는 가능한 상품 조합들에 대해 구매자가 가치 평가를 할 수 있는 조합 경매가 바람직하고, 조합 경매를 통해서 판매자는 최대의 이익을 얻을 수가 있다. 그러나 상품의 종류가 많아지고 구매자의 수도 크게 증가할 경우, 판매자의 이득을 최대화시키기 위하여 구매자들에게 상품들을 최적으로 할당하는 문제는 계산적 어려움(computational difficulty)이 존재한다. 이 문제를 해결하기 위해서 크게 두 가지 접근법이 있다. 첫 번째는 최적해를 찾는 시간을 줄이기 위한 휴리스틱들에 대한 연구이고 두 번째는 최적해를 찾지는 않지만 최적해에 근사한 값을 빠른 시간에 찾기 위한 휴리스틱들에 대한 연구[8]이다. 본 논문에서는 수행한 연구는 첫 번째에 해당하며 다음 2.1과 2.2는 이에 대한 최근의 관련 연구들이다.

2.1 MIP(Mixed Integer Programming)

조합경매의 승자를 결정하는 문제는 MIP로 수식화될 수 있다[6]. Andersson은 MIP를 풀어주는 상용 프로그램인 CPLEX 6.5를 사용해서 여러 데이타 분포에 대해서 기존의 실험 결과들과 비교를 했다. 그 결과 CPLEX는 기존의 조합 경매의 승자를 결정하기 위해서 제시된 다른 알고리즘들에 비해서 좋은 성능을 보였고, 또 특정한 데이타 분포뿐만 아니라 일반적인 데이타에 대해서도 좋은 성능을 나타냈다.

2.2 CABOB(Combinatorial Auction Branch On Bids)

CABOB[7]은 Sandholm이 제시한 알고리즘으로 기본적으로 깊이 우선 탐색 분기 한정법(depth-first-branch-and-bound)으로 조합경매의 승자 결정을 한다. CABOB에서도 좋은 최대 한계치를 빠른 시간에 구하기 위해서 CPLEX의 LP를 사용했다.

다음은 CABOB에서 제시된 분기할 비드를 선택하는 휴리스틱들이다.

p_j:구매자 j가 제시한 가격

$|s_j|$:구매자 j가 입찰한 상품의 개수

s_j:구매자 j가 입찰한 상품들의 집합

x_j: LP에서 구해지는 구매자 j의 계수값

휴리스틱들 중에서는 실험적인 경험을 통해서 OB(One Bids)와 NSS(Normalized Shadow Surplus)를 혼합한 휴리스틱을 사용했을 때 가장 좋은 결과를 얻을 수 있었다. CABOB의 실험에서는 여러 데이타 분포에 대한 실험 결과를 CPLEX 7.0의 결과와 비교했다. CABOB은 일부 데이타 분포에서 더 좋은 결과를 보였으나 그 차이가 크지 않고 그 외의 데이타 분포에서는 CPLEX보다 상당히 떨어지는 결과를 냈다.

▶ NBP(Normalized Bid Price) : $w_j=(p_j/|s_j|^a)$. 상품의 평균 값을 계산하여 가장 높은 값을 가지는 비드로 분기하는 방법이다.

▶ NSS(Normalized Shadow Surplus) : $w_j=(p_j-\sum_{i\in s_j} y_i/(\sum_{i\in s_j} y_i)^a)$. NBP는 각 상품의 가치를 동일하게 보고 계산을 하지만 입찰자들에 따라서 상품의 가치는 틀리기 마련이다. NSS는 LP의 다른 형태인 DUAL 문제에서 구해지는 가격인 y_i를 상품의 가치로 생각하고 입찰자가 제시한 가격과의 차이를 정규화하여 가장 큰 값을 가지는 비드로 분기한다.

▶ BGN(Bid Graph Neighbors) : 충돌하는 비드가 가장 많은 비드, 즉 비드를 그래프화 했을 때 이웃의 비드들이 가장 많은 비드로 분기하는 방법이다.

▶ NI(Number of Items) : 가장 많은 상품을 가지는 비드로 분기하는 방법이다.

▶ OB(One Bids) : x 값이 1에 가장 근접한 비드로 분기하는 방법이다.

▶ FB(Fractional Bids) : x 값이 0.5에 근접한 비드로 분기하는 방법이다. 이 방법은 x 값이 0이나 1처럼 확실한 비드보다는 중간 값을 가지는 비드를 우선적으로 선택해서 탐색함으로써 불확실성을 먼저 제거해나가는 전략이다.

3. 조합 경매(Combinatorial Auction)

3.1 정의

조합 경매는 입찰자들이 다수의 상품들을 하나의 꾸러미(package)로 묶어서 입찰할 수 있는 경매다. 따라서 상품들간의 상보성을 표현할 수 있다. 판매자가 팔고자 하는 상품들에 대해서 조합 경매를 실시하면 입찰자들은 각자가 원하는 꾸러미와 꾸러미에 대한 입찰 가격을 제시한다. 판매자는 자신의 이득을 최대로 하는 입찰자들을 선택하고, 선택된 입찰자들이 승자가 된다. 판매자의 이득은 승자들이 제시한 입찰 가격들의 합이 된다.

n명의 입찰자가 m개의 상품에 대해 입찰을 할 때,

· 입찰자 집합 : $B = \{b_1, b_2, ..., b_n\}$

· 상품 집합 : $S = \{1, 2, ..., m\}$

· 입찰자 b_i의 입찰: $b_i=(s_i, p_i)$ (단, s_i는 공집합을 제외한 S의 부분집합. p_i는 s_i에 대한 입찰가격)

3.2 승자 결정(Winner Determination)

조합 경매에서 판매자의 이득을 최대로 하는 승자들은 다음과 같다.

$$Max \sum_{i=1}^{n} x_i p_i \quad s.t. \quad \sum_{j\in s_i} x_i \le 1, \ \forall j \in S \qquad (1)$$

$$x_i \in 0,1, \ \text{for each } i = 1,2,\ldots,n.$$

(x_i가 1이면 승자, 0이면 패자)

3.3 LP와 최대 한계치

상품이 승자들에게 분할될 수 있다면 LP식은 식 (1)에서 $x_i \in 0,1$이 $x_i \in \overrightarrow{R^+}$로 바뀐 다음 식과 같다.

$$Max \sum_{i=1}^{n} x_i p_i \quad s.t. \quad \sum_{j\in s_i} x_i \le 1, \ \forall j \in S \qquad (2)$$

$$x_i \in \overrightarrow{R^+}, \ \text{for each } i = 1,2,\ldots,n.$$

식 (2)에서 판매자의 이득을 최대로 하는 함수가 목적 함수(object function)로 설정이 되고, 모든 상품들의 수량은 한 개이기 때문에 분할되어 판매된 상품들의 부분들의 합이 1을 넘지 않는 조건이 제약식(constraint)으로 설정된다. 위의 LP식에서 x_i는 각 구매자들에게 분할되어 할당되는 i번째 상품의 양으로 0과 1사이의 실수로 결정된다.

분기 한정법에서는 최대 한계치를 이용해 탐색 공간을 가지치기(prune)하면서 최적의 해를 찾아나간다. 이 때 LP는 상품들을 분할해서 할당하기 때문에 승자 결정을 직접적으로 할 수 없으나 분기 한정법에서 사용될 수 있는 좋은 최대 한계치를 제공할 수 있고, 이는 최적의 해를 찾는 시간을 줄이는데 영향을 준다[3].

4. 최적 분배 알고리즘과 선택 휴리스틱

판매자의 최대 이익을 보장하기 위해 논문에서 설계된 최적 분배 알고리즘은 분기 한정법에 LP를 혼합한 알고리즘으로 판매자의 최대 이익을 보장하기 위해서 상품들을 구매자들에게 최적으로 분배한다. 다음 그림 1에 승자 결정을 위한 최적 분배 알고리즘이 제시되어 있다. 최적 분배 알고리즘은 기본적으로 우선순위 큐를 사용하여 BFS(best-first-search)를 구현하는 분기 한정법이며 모든 탐색 공간에 대해서 탐색을 진행하되 남은 비드(*remaining bid*)의 수가 적은 부분 탐색 공간에 대해서는 DFS(depth-first-search)를 한다. 우선순위 큐에서 꺼낸 노드(*node*) v에는 현재 선택된 비드들과 그렇지 않고 남아있는 비드들의 정보들이 저장되어 있으며, *Bound*(v)는 LP를 사용하여 v의 최대 한계치를 구하는 것을 의미한다. 최대 한계치가 현재까지 구한 최적값(*optimal value*)보다 크다면 분기를 하여 탐색을 더 해봐야 한다. 이때 남은 비드의 수가 전체 비드의 수의 10%보다 작다면 DFS로 남은 비드들을 탐색하며 그렇지 않다면 BFS로 탐색을 한다. 그리고 10%는 실험을 통해서 적절하다고 얻은 수치이다. 분기를 하기 전에 v가 Integer Case에 해당되면 v에 대해서는 더 이상 탐색할 필요가 없다. 그렇지 않으면 남은 비드들 중에서 비드 선택 휴리스틱을 사용해 새로운 비드를 선택하고 v의 자식 노드 $u(u_1,u_2)$를 만든다. u_1은 선택한 비드를 v의 해집합에 포함시키는 경우고 u_2는 포함시키지 않는 경우로서 각 노드가 대해서 최적값을 갱신할 수 있다면

```
최적 분배 알고리즘()
{
 PriorityQueue PQ;
 Node u,v;
 Check Complete Case;        //Special Case//
 Bound(v);                   //Compute upper bound by LP//
 Insert(PQ, v);
 while(PQ is not empty){
 Remove(PQ, v);
 if(v's bound is better than optimal value) then
    if(Remaining bids are less than 10% of total bids) then
       Explore remaining bids by DFS;
       Update optimal value;
    else
       Check Integer Case;                    //Special Case//
                     Select a bid to explore by heuristic;    //Bid Selection Heuristic//
       for(Each child u of v){                 /Include(exclude) selected bid to(from) v//
            if(Value(u) is better than optimal value) then
               Update optimal value;
            if(Bound(u) is better than optimal value) then
               Insert(PQ, u);
       }
 }
}
```

그림 1 승자결정을 위한 최적 분배알고리즘

최적값을 갱신한다. 그리고 각 노드의 최대 한계치를 구해서 계속 탐색을 할지 아니면 가지치기할지를 결정한다.

4장의 나머지 내용에서는 분기할 비드를 선택하기 위해서 사용한 휴리스틱에 대해서 자세히 알아보고 최대 한계치를 재사용하는 경우와 그 외에 알고리즘에 사용된 기법들을 설명한다.

4.1 비드 선택 휴리스틱(Bid Selection Heuristics)

본 논문에서 사용한 분기 한정법에서는 선택이 되지 않고 남아있는 비드들 중에서 최적의 해에 포함되는 비드라고 생각되는 비드를 선택하기 위해서 비드 선택 휴리스틱을 이용한다. 기존에 사용되어온 선택 휴리스틱들은 LP의 계수값을 이용하거나 구매자들이 제시하는 정보(입찰 가격, 상품의 개수)를 사용하고, 또는 각각의 비드를 노드로 표시하고 동일한 상품을 입찰한 비드들을 연결선(edge)으로 연결하여 그래프로 표현했을 때, 그래프의 연결선의 수를 이용한 휴리스틱들이 제시되었다.

조합 경매의 특성상 승자가 되는 비드들은 서로 중복되는 상품을 가질 수 없고, 이는 비드 그래프에서 연결된 노드들은 동시에 선택이 될 수 없다는 것을 의미한다. 제안된 다음 휴리스틱들은 LP의 계수값을 비드를 선택하기 위한 정보로 사용하며 조합 경매의 특성을 반영하기 위해서 서로 충돌하는 비드(비드 그래프에서 연결된 노드)들의 LP 계수값을 동시에 고려한다.

N: 구매자의 수.
$B = b_1, b_2, \ldots, b_N$: 구매자들의 집합.
$X = x_1, x_2, \ldots, x_N$: LP의 계수값들의 집합.

$b_i (1 \le i, j \le N)$에 대해서,

$$c_{ij} = \begin{cases} 0, & i=j \text{ 또는 } b_i\text{와 } b_j\text{가 서로 다른 상품을 입찰한 경우.} \\ 1, & b_i\text{와 } b_j\text{가 동일한 상품을 한 개 이상 입찰한 경우.} \end{cases}$$

• Conflict Bids Sum(CBS) : 임의의 b_i에 대해서 x_i가 0.5이상이면 b_i와 충돌하는 비드들의 x 값들의 합을 계산해서 CBS_i를 구하고 x_i가 0.5미만이면 x_i를 CBS_i로 한다. 그리고 가장 높은 CBS_i 값을 가지는 비드가 선택된다.

$$CBS_i = \begin{cases} \sum_{j=1}^{N} (x_j \times c_{ij}) & , (x_i \ge 0.5) \\ x_i & , (x_i < 0.5) \end{cases} \quad (3)$$

$$CBS = \max(CBS_i)\ (i = 1,2,\cdots N.\ \text{단, } b_i\text{는 } \textit{remaining bid}) \quad (4)$$

CBS는 x 값이 0.5인 것을 기준으로 그 이상의 값을 가지는 비드들은 동일한 가치를 가지는 것으로 가정하고 이와 충돌하는 비드들의 x 값들을 합해서 이를 우선순위를 결정하는 정보로 사용한다. 따라서 자신의 x 값이 높더라도 충돌하는 비드들의 x 값이 모두 작다면 선택이 될 가능성이 낮아진다.

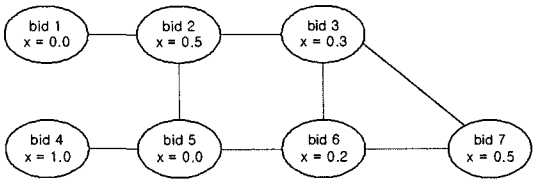

그림 2 Bid Graph Example

결국 *CBS*는 자신의 x 값도 높으면서 충돌하는 비드들의 x 값도 높은 비드를 선택하는 성격을 가진다. x 값이 1이나 0처럼 편향된 비드들보다 0.5와 같이 중간값을 가지는 비드들을 우선적으로 선택하고 x 값이 어중간한 비드들을 먼저 탐색해버림으로써 비드들의 x 값이 0이나 1로 결정되는 Integer Case에 빨리 도달되도록 기대할 수 있다.

그림 2의 그래프와 같이 비드들의 정보가 그래프로 표현되었을 때 각 비드의 *CBS*값을 식 (3)에 따라 계산해보면,

$$CBS_7(x_7 \geq 0.5) = x_3 + x_6 = 0.3 + 0.2 = 0.5$$

로서 *bid 7*이 0.5로 가장 크기 때문에 분기할 비드로서 선택된다.

• Sum of Adjacent Sum(*SAS*) : 임의의 b_i에 대해서 b_i와 충돌하는 모든 비드들의 x 값을 더하고, 충돌하는 비드의 수만큼 b_i의 x_i 값을 더한다. 그리고 가장 높은 SAS_i 값을 가지는 비드가 선택된다.

$$SAS_i = \sum_{j=1}^{N} (x_i + x_j \times c_{ij}) \quad (5)$$

$$SAS = \max(SAS_i)(i = 1,2,\cdots N.\ \text{단, } b_i\text{는 } remaining\ bid) \quad (6)$$

그림 2의 그래프로부터 분기할 비드를 *SAS*를 이용해서 구해보면,

$$SAS_3 = x_3 + x_2 + x_3 + x_6 + x_3 + x_7$$
$$= 0.3 + 0.5 + 0.3 + 0.2 + 0.3 + 0.5 = 2.1$$

으로 *bid 3*이 2.1로 가장 크기 때문에 선택된다. 이처럼 *SAS*는 자신의 x 값도 어느 정도 크면서 자신과 충돌하는 비드의 수가 많고 또 그들의 x 값의 합이 큰 비드를 선택하는 특성을 가진다. 결국 *SAS*는 LP의 계수값이 표시된 비드 그래프에서 볼 때, x 값이 큰 비드들이 많이 연결된 비드 중에서도 자신의 x 값이 큰 비드를 분기할 비드로 선택하는 경향을 나타낸다.

• Sum of Adjacent Difference(*SAD*) : 임의의 b_i와 충돌하는 비드들의 x 값의 차이를 모두 합한다. 그리고 가장 높은 SAD_i 값을 가지는 비드가 선택된다.

$$SAD_i = \sum_{j=1}^{N} (x_i - x_j \times c_{ij}) \quad (7)$$

$$SAD = \max(SAD_i)(i = 1,2,\cdots N.\ \text{단, } \text{는 } remaining\ bid) \quad (8)$$

*SAD*를 그림 2에 적용해보면,

$$SAD_2 = x_2 - x_1 + x_2 - x_3 + x_2 - x_5$$
$$= 0.5 - 0.0 + 0.5 - 0.3 + 0.5 - 0.0 = 1.2$$

으로 *bid 2*가 1.2로 가장 크다. *SAS*처럼 자신과 충돌하는 비드의 수가 많을수록 자신의 x 값을 더하는 횟수가 증가하는 점은 같다. 그러나 *SAS*와는 다르게 충돌하는 비드의 x 값과의 차이를 계산하기 때문에 자신의 x 값이 작거나 연결된 비드의 x 값이 상대적으로 높다면 *SAD*값은 낮아진다. 즉, *SAD*는 연결된 비드가 많으면서 자신의 x 값과 연결된 비드들의 x 값의 차이가 큰 비드를 분기할 비드로 선택한다.

4.2 최대 한계치 재사용(Upper Bound Reuse)

분기 한정법에서는 비드 선택 휴리스틱에 의해서 선택한 비드를 포함해서 분기하는 경우와 포함시키지 않고 분기하는 두 가지 경우로 나누어서 탐색을 한다. 이때 조합 경매의 특성상 각각의 경우에 LP를 사용하지 않고 이전에 구한 최대 한계치를 이용해도 되는 상황이 존재한다.

• 선택한 비드를 포함하는 경우: 휴리스틱으로 선택한 비드를 해집합에 포함시켜서 분기할 때, 남아있는 비드들의 최대 한계치를 구하기 위해서는 LP를 다시 계산해야 한다. 그러나 선택한 비드의 계수값이 1인 경우, 분기할 때의 최대 한계치는 분기하기 이전의 최대 한계치에서 선택한 비드의 가격을 뺀 값과 동일하다. 왜냐하면 선택한 비드와 충돌하는 비드들의 계수값은 모두 0이고, 충돌하는 비드들은 분기할 때의 최대 한계치를 구할 때 제외되기 때문이다. 따라서 이전의 최대 한계치를 재사용하면 남은 비드들의 최대 한계치를 구하는 시간을 줄일 수 있다.

• 선택한 비드를 포함시키지 않는 경우: 선택한 비드를 해집합에 포함시키지 않고 분기할 때는 선택된 비드와 충돌하는 비드들이 제외되지 않는다. 이때 선택한 비드의 계수값이 0이라면 남은 비드들의 최대 한계치는 분기하기 전의 최대 한계치와 동일하기 때문에 최대 한계치를 다시 계산할 필요가 없다[9].

4.3 BDS(Best-Depth-Search)

BFS는 최대 한계치가 큰 방향으로 탐색을 하지만 필요한 메모리 공간이 지수적으로 증가하므로 문제가 커질 경우 메모리 부족 문제가 발생한다. 그에 반해 DFS는 선형 공간(linear space)을 사용하는 이점이 있다[7].

하지만 DFS는 한계치에 의해 가지치기되지 않는 한 끝까지 탐색을 하기 때문에 BFS보다 더 많은 탐색을 할 수 있다. 그래서 본 논문에서는 BFS과 DFS을 혼합

한 형태의 탐색 방법인 BDS를 사용했다.

BDS는 처음에는 BFS을 사용해서 탐색을 하다가 남은 비드의 수가 한계치(*threshold* β = 총 비드 수의 10%) 이하가 되면 DFS를 사용해 남은 비드들을 탐색한다. DFS를 통해 끝까지 탐색을 하더라도 비드의 수가 적기 때문에 빠른 시간에 찾은 해가 최적인지 아닌지 판단할 수 있다. 문제의 크기가 큰 경우에도 탐색의 후반부에는 DFS를 사용하기 때문에 BFS를 사용하는 것보다는 메모리 문제에 있어 안정적인 탐색이 가능하다.

4.4 Integer Case 와 Complete Case

• Integer Case : 남아있는 비드들에 대해 LP를 사용해 최대 한계치를 구했을 때, 모든 비드들의 x 값이 0이나 1인 경우 Integer Programming의 해가 된다. 따라서 x가 1인 비드들이 최적의 해가 되므로 더 이상의 탐색이 필요 없다[7].

• Complete Case : 남은 비드들이 서로 충돌하는 경우 승자는 한 명밖에 될 수 없으므로, 가장 큰 가격을 제시한 입찰자가 선택된다[7]. 이러한 경우는 발생할 가능성이 적으므로 탐색을 시작하기 전에만 체크를 한다.

4.5 전처리(Preprocessing)

경매에 제출되는 비드들 중에는 승자 결정 알고리즘을 통하지 않고도 승자가 될 수 없는 비드들이 있을 수 있다. 예를 들어 동일한 상품 리스트에 대해서 입찰을 한 비드들 중에서 가장 높은 가격을 제시한 비드를 제외하고 모두 제거할 수 있다. 또 비드 a와 비드 b가 상품 집합을 $s(a)$와 $s(b)$, 그리고 입찰 가격은 $p(a)$와 $p(b)$로 제시했을 때, $s(a) \subset s(b)$이고 $p(a) > p(b)$라면 비드 b가 제거된다.

온라인 상으로 경매가 진행되고 비드들이 순차적으로 입력되는 경우, 입력되는 비드들을 전처리 하는 시간은 승자를 결정하기 위한 시간에서 제외될 수 있다.

5. 실험 및 결과

5.1 실험 환경 및 데이타

실험은 펜티엄IV-1.0G, 메모리 512M, 윈도우 2000 환경에서 이루어졌고 CPLEX는 7.0버전을 사용했다. 실험 데이타는 Sandholm이 제시한 5가지 데이타 분포(Random, Weighted Random, Decay, Uniform, Bound)[7]를 사용했다. Uniform을 제외한 각 데이타들의 상품의 수는 비드 수의 1/10이다. 모든 결과 그래프에서의 각 점들은 100개의 서로 다른 데이타에 대한 평균 값이다.

다음은 실험에 사용된 데이타 분포들로서 비드의 수가 n, 상품의 수가 m일 때,

• Random 분포 : 상품은 m개 중에서 랜덤하게 선택을 하고, 상품이 중복되더라도 다시 선택하지 않는다. 가격은 0과 1사이에서 선택하고 10000을 곱한다.

• Weighted Random 분포 : 상품은 m개 중에서 랜덤하게 선택을 하고, 상품이 중복되더라도 다시 선택하지 않는다. 가격은 0과 선택된 상품의 수의 사이에서 선택하고 10000을 곱한다.

• Decay 분포 : m개의 상품 중에서 하나를 랜덤하게 선택한다. 그리고 α의 확률로 새로운 상품을 반복해서 추가한다. 선택된 상품의 수가 m을 넘지 않고, 더 이상 추가가 안될 때까지 반복해서 추가한다.

• Uniform 분포 : 각 비드마다 동일한 수의 상품을 랜덤하게 선택한다. 가격은 0과 1사이에서 선택하고 10000을 곱한다.

• Bounded 분포 : 상품은 하위 한계와 상위 한계 사이에서 랜덤한 개수만큼 선택을 한다. 가격은 0과 선택된 상품의 수의 사이에서 선택하고 10000을 곱한다.

5.2 결과

각각의 분기 선택 휴리스틱 별로 최적 분배 알고리즘의 실험 결과를 CABOB과 CPLEX를 상대로 비교를 하려고 하였으나 CABOB의 소스 코드를 구할 수가 없었다. 그래서 논문을 참고하면서 직접 CABOB을 구현해보았으나 자세한 자료가 없어 CABOB과 똑같이 구현하기에는 한계가 있었다. 그래서 CPLEX 7.0을 사용하여 실험한 결과와 본 논문에서 설계한 알고리즘을 사용한 결과를 각 데이타 분포 별로 비교했으며 CABOB은 논문에서의 실험 결과를 참조했다.

참고로 CABOB 논문의 실험에서는 전처리를 하지 않은 입력 데이타를 CPLEX에 넣었기 때문에 전처리 시간이 CPLEX 시간에 포함되었고, CABOB 알고리즘을 사용한 결과에는 전처리 시간이 포함되지 않았다. 본 논문의 모든 실험에서는 전처리를 동일하게 거친 데이타들이 사용되었기 때문에 CPLEX의 수행 시간이 CABOB에서 실험한 CPLEX 결과보다 빨랐다.

표 1에서 볼 수 있듯이 Random분포와 Decay분포는 전처리에 의해서 제거되는 비드의 수가 많기 때문에 두 분포에서의 CPLEX 시간은 CABOB에서 실험한 CPLEX 시간보다 빠르다. 앞으로는 본 논문에서 실험한

표 1 평균 전처리 비율

분포	평균 전처리율(%)
Random	79.72
Weighted Random	12.27
Decay	61.92
Uniform	0.17
Bound	5.98

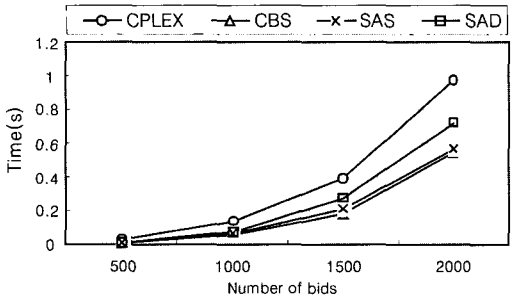

그림 3 Random 분포

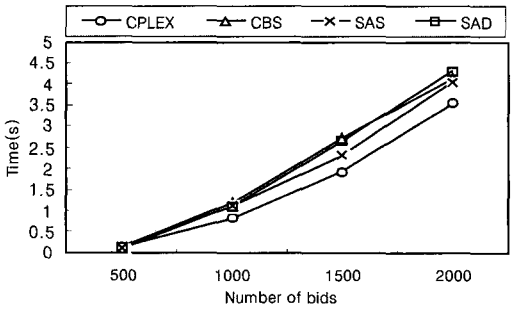

그림 4 Weighted Random 분포

CPLEX와 CABOB에서 실험한 CPLEX를 구분하기 위하여 후자를 CPLEX(CABOB)로 표현하기로 한다.

Random 분포(그림 3)에서는 제안한 휴리스틱을 사용한 알고리즘의 수행시간이 모두 CPLEX보다 좋은 결과를 보였다. 그리고 휴리스틱들 중에서는 *CBS*를 사용했을 경우 가장 빠른 결과를 얻었다. 비드의 수가 2000개일 때, CABOB은 3.5초 정도가 걸렸고 CPLEX(CABOB)는 6초 정도가 걸린 것으로 나와있다. CPLEX의 시간이 CPLEX(CABOB)보다 현저하게 빠른 것은 실험 환경이 다른 이유도 있지만 전처리가 이루어진 데이타가 사용되었기 때문이다.

Weighted Random 분포(그림 4)에서는 비드의 수가 2000일 때 *CBS*, *SAD*, *SAS*를 사용한 시간이 4.1초, 4.3초, 4.0초로 거의 비슷했고 CPLEX가 3.5초로 조금 빨랐다.

Weighted Random 분포의 경우 데이타들의 평균 전처리율이 12.27%로 낮은 편이기 때문에 비드의 수가 2000일 때 CPLEX(CABOB)은 3.5초 정도로 CPLEX의 결과와 비슷했고, CABOB은 4.3초 정도로 휴리스틱들의 결과와 비슷하다.

Decay 분포(그림 5)에서는 *SAD*와 *SAS*를 사용한 결과가 *CBS*를 사용한 결과와 CPLEX의 결과보다 비드의 수가 증가할수록 느려짐을 알 수 있다. Decay분포는 대부분의 데이타가 처음부터 Integer Case에 해당되기 때

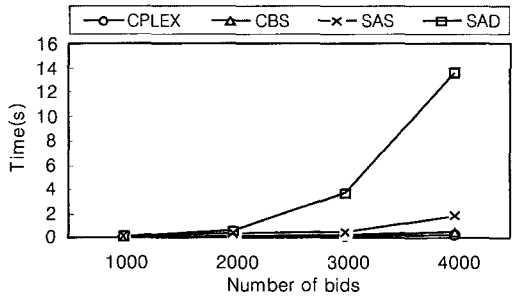

그림 5 Decay 분포(α = 0.45)

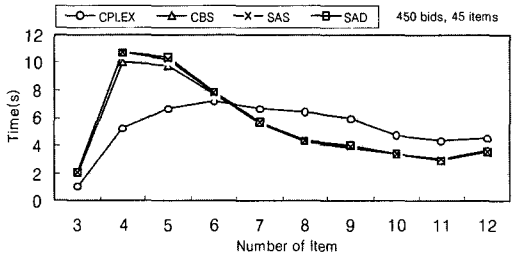

그림 6 Uniform 분포

문에 탐색을 하지 않고 승자를 찾을 수 있다. 그러나 그렇지 않은 데이타들에 대해서 *SAS*와 *SAD*는 비드의 x 값이 0.5로 동일한 비드들이 있을 때, 최적해에 포함되지 않는 비드를 선택하여 분기를 하였고 *CBS*보다 더 많은 탐색을 하여 수행 시간이 많이 걸렸다. 그러나 *CBS*는 이러한 경우 최적해에 포함되는 비드를 선택하여 탐색을 함으로써 빠른 수행 시간을 보이는 것을 볼 수 있었다. CABOB은 Decay 분포(α = 0.75)에서 CPLEX(CABOB)보다 좋은 결과를 나타냈지만, 비드의 수가 4000개일 때의 시간 차이가 0.22초 정도로 미비하다.

Uniform 분포(그림 6)의 경우, 제안한 휴리스틱들은 비슷한 성능을 보였다. 상품의 수가 3개에서 6개로 적은 경우 각 휴리스틱들을 사용한 알고리즘은 CPLEX보다 느렸지만 상품의 수가 7개 이상으로 많아질 경우에는 더 빠른 성능을 보였다. CABOB의 경우 CPLEX(CABOB)과 비교했을 때, 모든 경우의 상품의 수에서 CPLEX(CABOB)보다 항상 느린 결과를 보였으며 상품의 수가 12개일 때는 시간 차이가 14초나 될 정도로 안 좋은 성능을 보였다.

bound가 5에서 10인 Bounded-Low 분포(그림 7)에서는 휴리스틱들이 CPLEX보다 더 좋은 결과를 보였다. 휴리스틱들 중에서는 *SAS*와 *SAD*를 사용했을 때 가장 빠르고, *CBS*가 조금 느린 결과를 보였다. CABOB의 경우 Uniform 분포의 실험 결과처럼 CPLEX(CABOB)보다 항상 느린 결과를 보였고, 비드의 수가 400개인 경

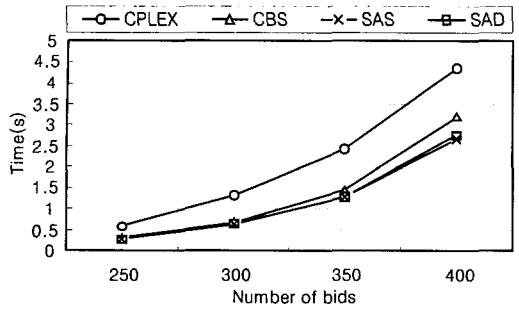

그림 7 Bounded-Low 분포

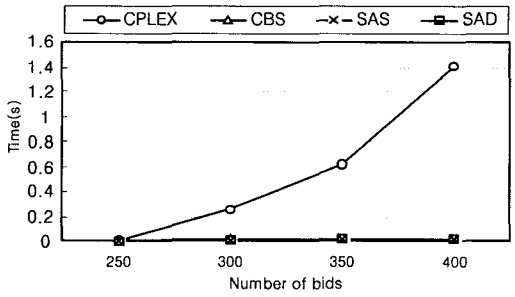

그림 8 Bounded-High 분포

우 CABOB은 약 9초, CPLEX (CABOB)은 5초 정도의 시간이 걸렸다.

bound가 20에서 25인 Bounded-High 분포(그림 8)의 경우 휴리스틱들은 시간이 거의 걸리지 않았으나 모든 데이타가 Complete Case에 속하기 때문에 실험 결과가 큰 의미를 가진다고 볼 수 없다.

각각의 비드 선택 휴리스틱을 사용한 알고리즘들은 대부분의 분포에서 서로 비슷한 수행 시간을 보였으나 Decay 분포에서는 *CBS*를 사용했을 경우가 *SAS*와 *SAD*를 사용했을 때보다 상당히 좋은 결과를 보였다. 이는 *CBS*가 Decay 분포에서의 x 값이 같은(*tie break*) 상황을 *SAS*나 *SAD*보다 잘 해결하기 때문이다.

*CBS*와 CPLEX를 비교해 봤을 때, *CBS*는 Random 분포와 Bounded 분포에서 CPLEX보다 좋은 성능을 보였고, Weighted Random 분포와 Decay 분포에서는 비슷한 성능을 나타냈다. Uniform 분포에서는 상품의 수가 많아질수록 *CBS*가 더 좋은 성능을 보였다.

CABOB은 CPLEX(CABOB)보다 Random 분포와 Bounded-High 분포에서만 좋은 성능을 보였을 뿐 다른 분포들에 대해서는 비슷하거나 상당히 떨어지는 결과를 보였다. 그러나 *CBS*를 사용한 알고리즘은 모든 분포에서 CPLEX와 비슷하거나 보다 좋은 결과를 보여 CABOB보다 실험 데이타 분포에 대해서 상대적으로 우수한 성능을 가짐을 알 수 있었다.

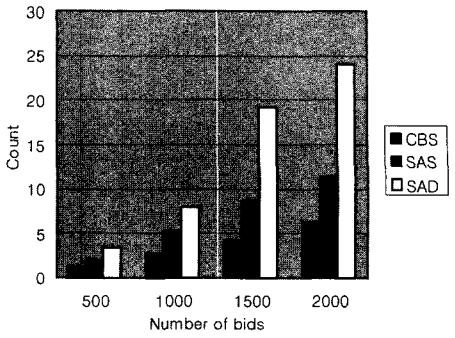

그림 9 Random 분포

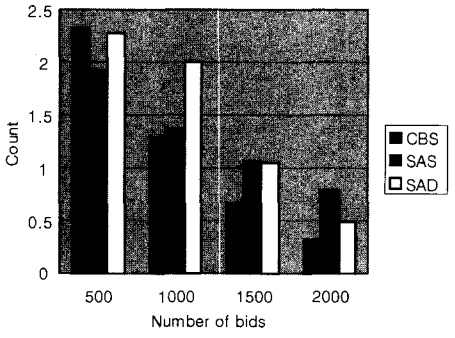

그림 10 Weighted Random 분포

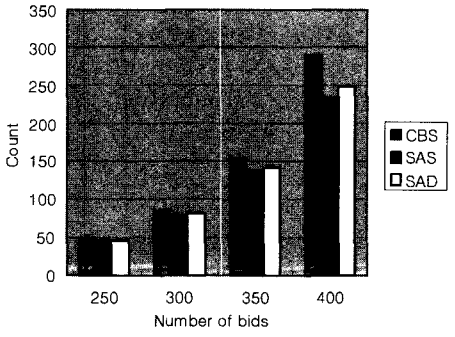

그림 11 Bounded-Low 분포

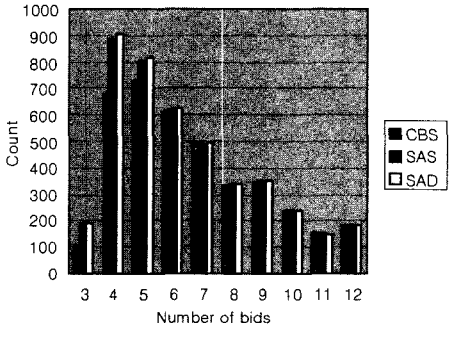

그림 12 Uniform 분포

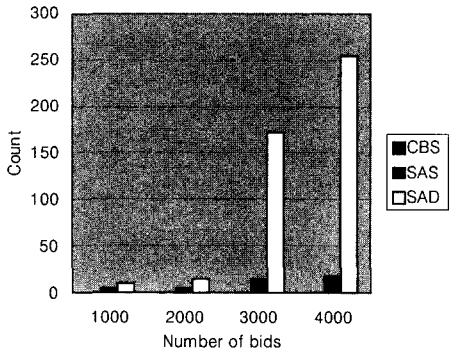

그림 13 Decay 분포

다음의 그래프들은 최적해를 찾을 때까지 각각의 비드 선택 휴리스틱에 의해서 비드가 선택된 평균 회수를 데이타 분포 별로 비교한 그림이다.

Random 분포(그림 9)에서 *CBS*는 비드의 크기가 2000인 경우에 평균 선택 회수가 6.23번으로 11.21, 24.09번을 기록한 *SAS*, *SAD*와 큰 차이를 보였다. Decay 분포(그림 13)에서 *CBS*는 선택 회수가 평균 1번 이내였으나 *SAS*와 *SAD*는 비드 수가 증가함에 따라 선택 회수도 크게 증가했고 이는 결과를 찾는 시간에도 반영이 되었다. 그리고 상품의 수가 적은 Uniform 분포(그림 12)에서도 *CBS*의 선택 회수가 적은 것으로 나타났다. 그 외의 데이타 분포들에서는 휴리스틱들이 비슷한 선택 회수를 보였고 Bound-High 분포의 경우는 데이타들이 Complete Case여서 탐색 회수가 모두 0이었다. 실험 결과 휴리스틱들 중에서는 자신의 x 값도 높으면서 충돌하는 비드들의 x 값도 높은 비드를 선택하는 *CBS*가 *SAS*, *SAD*보다 비드 선택을 잘 하여 탐색 회수를 줄이는 것을 알 수 있었다.

그리고 비드가 선택된 평균 회수와 실행 시간 결과를 비교했을 때, 전반적으로 휴리스틱에 의한 비드 선택 회수가 적을수록 최적해를 찾는 시간이 적게 걸리는 것을 알 수 있다.

6. 결 론

본 논문에서는 조합 경매에서 판매자의 최대 이익을 빨리 찾기 위해서 분기 한정법과 LP를 사용한 최적 분배 알고리즘을 설계하였고 최적해를 찾는 과정에서 최적해에 포함될 것 같은 비드를 잘 선택해서 분기를 하는 새로운 비드 선택 휴리스틱들을 제안했다. 그리고 최대 한계치를 재사용하여 최대 한계치를 구하는 시간을 줄임으로써 알고리즘의 성능을 향상시켰다.

탐색 회수가 적을수록 분기 한정법의 성능은 더 좋아질 수 있으므로 분기를 하기 위한 비드를 선택하는 휴리스틱은 매우 중요하다. 제안한 휴리스틱들은 이전의 방법들과는 달리 각 비드들의 x 값뿐만 아니라 비드와 충돌하는 비드들의 x 값들도 동시에 고려하는 특징을 가지고 있어서 탐색을 할 때 조합 경매의 특성을 반영할 수 있었다. 실험 결과 휴리스틱들 중에서는 *CBS*를 사용했을 때 *SAS*와 *SAD*를 사용했을 때보다 빠른 시간에 최적의 해를 찾아내고 평균 선택 회수의 비교 결과에서도 *CBS*가 *SAS*와 *SAD*에 비해 더 적은 선택을 하는 것을 알 수 있었다. CPLEX에 비해서 *CBS*를 사용한 알고리즘은 Random 분포와 Bounded 분포에서 보다 좋은 성능을 보였고 나머지 분포에서도 비슷한 성능을 보였다.

실험에 사용된 데이타 분포는 조합 경매에 대한 연구를 하는 기존 논문들에서 사용된 인위적 데이타이기 때문에 향후 과제로서는 실제 조합 경매에 사용되는 데이타 분포를 획득하고 그 특성에 대한 연구와 이에 맞는 휴리스틱을 개발하는 것이 필요하다.

참 고 문 헌

[1] M. H. Rothkopf, A. Pekec, And R. M. Harstad, "Computationally manageable combinatorial auctions," *Management Science*, Vol.44, No.8, pp. 1131-1147, 1995.

[2] Y. Fujishima, K. Leyton-Brown, And Y. Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches," *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 548-553, 1999.

[3] N. Nisan, "Bidding and allocation in combinatorial auctions," *In ACM Conference on Electronic Commerce*, pp. 1-12, 2000.

[4] S. de Vries, R. Vohra, "Combinatorial auctions: A Survey," *Draft*, 2000.

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

[6] A. Andersson, M. Tenhunen, and F. Ygge, "Integer programming for combinatorial auction winner determination," *In Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS)*, pp. 39 -46, 2000.

[7] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, "CABOB: A fast optimal algorithm for combinatorial auctions," *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1102-1108, 2001.

[8] E. Zurel and N. Nisan, "An efficient approximate allocation algorithm for combinatorial auctions," *In ACM Conference on Electronic Commerce*, pp.

125-136, 2001.
[9] R. Gonen and D. Lehmann, "Linear Programming helps solving large multi-unit combinatorial auctions," *Electronic Market Design Workshop*, July 11-13, 2001.

송 진 우
2001년 연세대학교 정보산업전공 졸업(공학사). 2003년 연세대학교 컴퓨터산업공학부 졸업(공학석사). 2003년~현재 연세대학교 컴퓨터산업공학부 박사과정. 관심분야는 Electronic Commerce, Grid Computing, P2P, Media Streaming

양 성 봉
정보과학회논문지 : 시스템 및 이론 제 30 권 제 4 호 참조