# An Efficient Heuristic Bundle Search Algorithm for Buyers in Electronic Markets

Linli He and Thomas R. Ioerger
Department of Computer Science
Texas A&M University, College Station, TX77843-3112
{linli, ioerger}@cs.tamu.edu

## ABSTRACT

*In the age of electronic commerce, with low-cost information access, it has been recognized that a bundle search in a combinatorial trade is very valuable for buyers. Optimal travel package search is one of the most prominent examples of bundle search, allowing buyers to exploit various discounts through business partnerships among sellers. There are many varieties of bundle search problem. In general, these problems are varieties of the NP-hard Knapsack problem. Developing heuristic algorithms to find close optimal results is a promising approach to solving those problems with polynomial or even linear computational complexity. In this paper, we address a bundle search problem for buyers in electronic markets. There exist multiple sellers in a market. Sellers offer widely varying prices for identical items and various discount ratios for different purchasing costs. We propose a heuristic bundle search algorithm, Maximal Gain Bundle Search (MGBS) algorithm, to solve this problem. The time complexity of MGBS algorithm is O(NM) in the worst case. The experiments show that the execution time of the MGBS algorithm is not sensitive to an increase of the number of bundle goods and sellers. The simulation results also show that the MGBS algorithm can produce a nearly optimal bundle purchase for buyers.*

## Keywords

Heuristic search, Buyer strategy, Bundle search, Discount ratio

## 1. INTRODUCTION

In electronic commerce, the distance among producers, wholesalers, distributors, retailers, and consumers has nearly disappeared [1]. One of the most popular areas of research is electronic combinatorial commerce, where combinations of goods and services are traded and efficiently allocated. For instance, electronic combinatorial auctions are becoming more and more popular [2,3,4]. There are many more choices faced by all parties involved in electronic commerce than in traditional commerce. As a result, the relationship between suppliers and customers is undergoing revolutionary changes.

Buyers vary a great deal in the quantity of goods they purchase, in customer service requirements, in income, in time constraints and along many other aspects. Different purchasing goals can cause widely varying production and transaction costs. Suppliers have their "buyer selection" strategies to enhance profitability [5]. In electronic markets, quickly differentiating the supplier's marketing strategy based on the difference of purchasing goals from various buyers plays a key role in improving suppliers' competitive capabilities.

On the other hand, because of the price differentiation, buyers can build corresponding purchasing strategies to minimize their purchase cost. In traditional markets, the high cost of accessing product information, makes it impractical for buyers to build such purchasing strategies. Only suppliers employed market analysis and intelligence to extract the buyer surplus [6]. However, in the age of electronic commerce, buyers also can access product information easily and inexpensively through the Internet. For suppliers, bundling large numbers of goods can be surprisingly profitable [7]. Conversely, buyers can build a corresponding bundled purchasing strategy to obtain a better discount. A well-known example of building such a purchasing strategy for buyers is to form buyer coalitions (buyers' clubs) to gain the discount for buying a large numbers of goods in the electronic market [1,8,9]. This strategy addresses the situation where different buyers want to buy small numbers of items. So these buyers can form a buyer coalition to enlarge the quantity of items in each transaction and get a greater discount.

There is another very interesting strategy called "bundle search" [6], which addresses the situation where a buyer needs to buy multiple goods as a bundle. The partnerships among suppliers, result in different bundles having different discounts. Typical applications are travel packaging, software, PC peripherals, etc. The problem is for the buyers to find the optimal bundle that causes minimum cost. Unfortunately, from an algorithmic perspective, this problem carries a high computational cost. In its general form, the scored bundle search is an NP-hard knapsack problem [6].

In electronic markets, there are many kinds of bundle search, which are different from travel package search. For example, many retailers like Amazon, always provide free shipping when the total amount of the purchase is over a certain threshold. Other retailers, like Express, provide a coupon for the customer's next purchase. Also, even for an identical item, the prices from different retailers vary widely. Hence, if a buyer has a long shopping list, finding an optimal combination of retailers that results in spending a minimal amount of money is a different bundle search problem from the travel package search. Instead of considering the discount from the partnerships among suppliers, the buyer tries to obtain a greater discount from retailers as well as get minimal price for each item on the shopping list. Solving this problem is not only valuable for individual consuming buyers, but also meaningful for industrial buyers when they have multiple manufacturers from which to select.

Unfortunately, finding the optimal result for this bound search problem is computationally intractable even for a small number of goods and sellers. This problem is even more complicated than the traditional Knapsack problem, because whenever you put one more item in the knapsack, the values of items that are already in the knapsack may change according to the supplier of the new item, because of the different discount policies from different sellers. The development of heuristic algorithms is a promising approach to solving this problem with polynomial or even linear computational complexity [6].

In this paper, we propose an efficient heuristic algorithm to solve this problem. We totally agree with one statement in [6], "Exploiting the structure of bundle search is key to reduce computational complexity". Our heuristic search rules are based on our observations of the special characteristics of this problem.
The rest of this paper is organized as follows: Section 2 gives a simple example and a formal definition of the problem addressed in this paper. Section 3 describes the heuristic algorithm proposed to solve this problem. Before a detailed algorithm description, we discuss the heuristic search rules that are applied for the algorithm. Section 4 analyzes the complexity and correctness the algorithm. Section 5 presents the experimental evaluation. Finally, Section 6 presents our conclusions and discusses future research directions and possible ways to improve this work further.

# 2. BUNDLE SEARCH PROBLEM

This section gives a formal definition of the bundle search problem addressed in this paper. To demonstrate the difficulty of the problem, we give a simple example.

## 2.1 An Example

At the beginning of each semester, for every college student, purchasing textbooks is a big expenditure. Thanks to the electronic market, it is possible to get books much cheaper than buying them from campus bookstores. Amazon provides information on some used textbooks with very attractive prices. Half is also a popular place for textbook shopping. Ordinarily, the prices for the same textbook vary widely with different electronic bookstores, seasons, book conditions, and so on. Of course, if a customer buys more books in one store, he or she may get free shipping, a mail-in rebate, or other kinds of coupons for future purchases. How does one select sellers for textbooks with minimal cost? Intuitively, the optimal solution could be obtained by enumerating all possible combinations of retailers, calculating the costs and picking the one that causes minimal cost. This is a very simple algorithm. Unfortunately, the computational cost is very high.

**Table 1: Textbook Shopping Example**

| | Retail0 | Retail1 | Retail2 | Retail3 | Retail4 |
|---|---|---|---|---|---|
| Book0 | $90.00 | $95.00 | $98.00 | Empty | $88.00 |
| Book1 | $35.00 | Empty | $30.00 | $39.00 | Empty |
| Book2 | Empty | $48.00 | $50.00 | $58.00 | $45.00 |
| Book3 | $80.00 | Empty | Empty | $75.00 | Empty |

Table 1 shows an example for this case. "Empty" means the retailer does not have this book in stock. There are 4x3x4x2 = 96 possible combinations of retailers to buy these four books. If there were no "Empty" cases, there would be $5^4$ = 625 possible combinations. If there are 10 books and 10 suppliers, the total number of possible combinations of retailers is $10^{10}$.
Customers may not pay attention to this optimization problem in the traditional market because the complete goods information may not be available to them. In an electronic market, it is not expensive at all for buyers to obtain the goods information. A formal definition of this problem follows.

## 2.2 Problem Formalization

**Problem Definition:**

*Given a set of goods that a buyer needs to purchase $G = \{G_0, G_1, \ldots, G_{m-1}\}$. There is a set of sellers $S = \{S_0, S_1, \ldots, S_{n-1}\}$ who can supply some or all goods in G. Each seller $S_i$ (i = 0, 1, .., n-1) has its own discount function $f_i$: $C_i \rightarrow D_i$. $C_i$ is the cost of each purchase and $D_i$ is the corresponding discount for $C_i$. Also there is a retail price vector $P_i = (P_{i0}, P_{i1}, \ldots P_{i, m-1})$ for each seller $S_i$. If seller $S_i$ has no item $G_j$ available, $P_{ij}$ = 0. The question is how to find an optimal combination of sellers from S that can provide minimal cost of purchasing all goods in G efficiently.*

To illustrate this problem, we borrow the idea of the linear graph in [6]. Figure 1 shows how to represent this bundle search problem in a linear graph. We use the example in Table 1. $G_i$ represents book[i] and $S_j$ represents retailer[j]. There are 4 layers of nodes. Each layer refers to the corresponding item. The nodes of each layer represent the sellers who can supply the goods. We use term "layer" instead of "stage" in [6] because there is no order among different goods in our case. Namely, the index of each layer is just an identifier of each goods. We define a "purchase path", which is formed by a node sequence in the linear graph. It starts at layer $G_0$ and ends at layer $G_3$ and includes only one node from each layer. The total cost of this purchase path is the total amount of cost spent on all sellers in the path. The bundle search problem defined above turns out to be a problem of finding a purchase path in the linear graph, the cost of which is minimum.

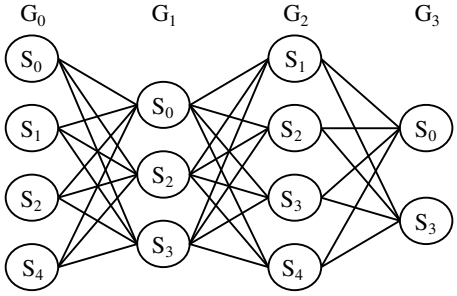


**Figure 1: Illustration of the Bundle Search Problem**

The bundle search problem we address in this paper is different from the travel package search problem. For the travel package problem, the discount comes from the partnership between different sellers (e.g. airlines, car rental companies, hotels etc.). The discount function in the bundle search problem in this paper is built independently for each seller. Partnerships may exist among different sellers. The discount ratio from partnership is fixed.

The difficulty of our bundle search problem mainly comes from the various discount ratios. Different sellers have their own discount policies. Even with a single seller, the discount ratios could vary widely with different amounts of purchase. The general assumption is that the greater the amounts purchased each time, the greater discount that buyers obtain. However, the discount ratio function is not necessarily monotonic increasing. For example, J.C.Penney often provides $10 off for purchases over $50, $15 off for purchases over $75, and $35 off for purchases over $150. If one spends $65, the discount is still $10. The discount ratio at $65 (15%) is lower than the one at $50 (20%). Figure 2 shows the corresponding discount ratio function.

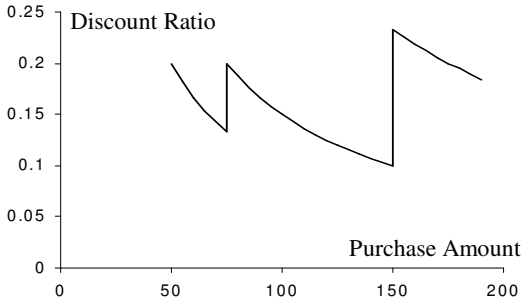


**Figure 2: The Discount Ratio Function**

# 3. ALGORITHM

There are a number of characteristics of our bundle search problem that could help to reduce the complexity of this problem and also obtain a nearly optimal result. These characteristics are the basis of heuristic search rules used in our heuristic algorithm.

## 3.1 Heuristic Search Rules

The problem of bundle search comes from the general economic case, the more one spends with a single seller, the more discount one gets from that seller. Without this basis, the bundle search is not a problem. Buyers only need to sort all prices for each item from different sellers and choose the lowest price. On the other hand, whenever a bundle discount exists, the sum of the minimal prices for all goods a buyer wants to buy becomes the upper bound of the bundle search. If the total cost of any single combination of sellers for a bundle purchase is larger than the sum of the corresponding minimal prices, those combinations of sellers should be discarded, since that bundle is not valuable. This observation becomes the first heuristic search criterion: for a bundle purchase, if the maximal discount from each seller cannot make the total cost of the bundle less than the sum of the corresponding minimal prices, then the bundle search fails. We just need to make a purchasing plan according to the minimal price of each item.

**Heuristic Search Rule 1: Maximal Bundle**

*If the cost of a bundle with the maximal discount from every available seller is larger than the sum of the corresponding minimal prices for the goods in the bundle, then it is not necessary to continue the bundle search.*

We call the bundle purchase from one seller which provides the maximal discount the "Maximal Bundle". If there are multiple sellers whose maximal bundle cost less than the sum of the corresponding minimal prices, we choose the seller who has the "Maximal Gain Ratio". To define this term, we need to define the gain of each bundle purchase from one seller. In this paper, the gain of each bundle purchase is not defined by the amount of the discount. If the prices provided by a seller are too high, even if it gives a large discount, the purchase cost could still be very high. So, the gain of each bundle purchase is defined to be the difference between the final cost of this bundle purchase and the sum of the corresponding minimal prices. The gain ratio is defined to be the ratio of the gain of a bundle purchase to the sum of the corresponding minimal prices; so the maximal gain ratio is the ratio of the gain of the maximal bundle purchase to the sum of the corresponding minimal prices. Picking the seller with the best "Maximal Gain Ratio" as our purchasing candidate is our second heuristic search rule.

**Heuristic Search Rule2: Maximal Gain Ratio**

*If the costs of Maximal Bundles from multiple sellers are less than the sums of the corresponding minimal prices, we pick the seller with the greatest "Maximal Gain Ratio" as the candidate seller.*

The third heuristic search rule comes from the possibility that the discount ratio could be non-monotonic increasing. Through the inverse function of the discount function, we can find the minimal cost to get the same amount of discount from one seller. Based on this minimal cost, we search for the cheapest bundle purchase with same amount of discount from this seller, and leave the other goods for another round of searching. This rule provides a method to refine the search results already obtained from the two rules above. The heuristic goal here is to achieve a higher discount ratio for each partial bundle purchase. According to this rule, in Rule 2, the maximal gain ratio is calculated for the sub-bundle with the highest discount ratio.

**Heuristic Search Rule 3: Bundle Regression**

*Before calculating the maximal gain ratio, the maximal bundle for each seller should be refined to the minimal bundle from the seller with same amount of discount as the maximal bundle.*

As a summary, all of the heuristic search rules above are based on the idea that, in order to find a better bundle, buyers search for lower prices as well as higher discount ratio. These heuristic search rules are the basis upon which we build our heuristic algorithm. We call it the Maximal Gain Bundle Search (MGBS) algorithm.

## 3.2 Maximal Gain Bundle Search Algorithm

Before we give the complete algorithm, we need to give some formal mathematical definitions for the terms in the heuristic search rules above, based on our formal problem definition.

The first term we need to define is the minimal price vector $P_{\min} = ( P_{\min}^0 , P_{\min}^1 , ..., P_{\min}^{m-1} )$, for a bundle purchase $G = \{G_0, G_1, ..., G_{m-1}\}$. $P_{\min}^i$ is the minimal price provided by the sellers in layer $i$ in the linear graph.

Suppose there is a bundle of sellers, $Sb = (Sb_0, Sb_1, ..., Sb_k)$ for a bundle of goods, $Gb = (Gb_0, Gb_1, ..., Gb_k)$. The gain g(Gb, Sb) of the bundle of Sb is defined by the following equation:

$$g(Gb, Sb) = \frac{\sum P_{\min}^{Gb} - (\sum P_{Sb}^{Gb} - \sum D_{Sb}^{Gb})}{\sum P_{\min}^{Gb}}$$

$\sum P_{Sb}^{Gb}$ denotes the sum of the prices of all goods in Gb of the sellers in $Sb$. $\sum P_{\min}^{Gb}$ denotes the sum of the minimal prices to purchase all goods in $Gb$. $\sum D_{Sb}^{Gb}$ denotes the sum of the discounts obtained from all of the sellers in $Sb$ for purchasing goods in $Gb$ as a bundle.The inverse function of the discount function of each seller $S_i$ is defined by $f_i^{-1} : D_i \rightarrow C_i^{\min}$. We use two matrices to represent the input data, the price matrix and the of item $G_i$ from seller $S_j$. If $S_j$ cannot provide $G_i$, $P_{ij}$ is equal to 0. providing matrix. The index of rows and columns of these two matrices represents the identified number of goods and sellers respectively. In the price matrix $M_p$, entry $P_{ij}$ represents the price.

**Maximal Gain Bundle Search (MGBS) Algorithm**
**Input:** *Mp, Mr, Empty Vector PV, PC = 0;*
**Output:** 1. Purchasing Plan Vector *PV*;
2. Cost of this purchasing plan *PC*.

**Begin MGBS** (*Mp, Mr*) {
1. Get information about good set *G* and seller set *S* from Mr and Mp;
2. Size of *PV* = Size of *G*;
3. Calculate the minimal price Vector $P_{\min}$ and get the corresponding seller Vector $SV_{\min}$ of $P_{\min}$;
4. Calculate the maximal cost *MaxCost* of this purchase $\sum P_{\min}^{i}$ - *sum of possible discount*;
5. For each seller $S_j$, {
6. Compute the sum of its maximal bundle for G by get the sum of each columns of *Mp*;
7. Calculate its discount *MaxDiscount[j]* by calling its own discount function;
8. Calculate its exact bundle cost *BundleCost[j]*;
}
9. For all *BundleCost[j]*, {
10. If **all** of *BundleCost[j]*s are larger than the corresponding *MaxSubCost[j]* $=\sum P_{\min}^{j}$ - *sum of possible discount;*
11. Then { *PV* = $SV_{\min}$, *PC* = *MaxCost*;
12. Return *PV, PC*;}
13. Else {
14. For each *BundleCost[j]*, {
15. Calculate the corresponding minimal cost *MinBundleCost[j]* with discount *MaxDisicount[j]* by calling its inverse discount function;
16. Compute the minimal good bundle vector *MinBundle[j]* = FindMinBundle (*MinBundleCost[j], j, Mp*)
17. Calculate the gain ratio $g(MinBundle[j], S_j) = (MaxSubCost[j]-BundleCost[j])/MaxSubCost[j]$;
}
18. Pick up *MinBundle[k]* of seller $S_k$ with maximal $g(MinBundle[k], S_k)$ among all $g(MinBundle[j], S_j)$;
19. Add $S_k$ to the corresponding positions in Vector PV according to the goods in *MinBundle[k]*;
20. *PC* = *PC* + the cost of *MinBundle[k]*;
21. Set the entries of the corresponding rows of this *MinBundle[k]* to *0* in *Mp* and *Mr*;
}
}
22. If all entries in *Mr* are *0* {
23. Return *PV, PC*;
}
24. Else {
25. Go back to line *5* with new *Mp and Mr*;
}

**End MGBS** (*Mp, Mr*).

**Procedure FindMinBundle** (MinBundleCost, j, Mp) {
1. Fetch the column j in Mp to be the good vector $GV_j$ of seller $S_j$;
2. Sort $GV_j$ according to the price increasingly;
3. s = size of $GV_j$;
4. While (s>0) {
5. Remove the last element in $GV_j$;
6. If the sum of prices in $GV_j$ < MinBundleCost,
7. then add the removed element back to $GV_j$;
8. s = s – 1 ; }
9. return $GV_j$;}

**Figure 3: Maximal Gain Bundle Search Algorithm**

**Input:**

$Mp$ =

| | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| $G_0$ | 90 | 95 | 98 | 0 | 88 |
| $G_1$ | 30 | 0 | 35 | 39 | 0 |
| $G_2$ | 0 | 48 | 50 | 58 | 45 |
| $G_3$ | 80 | 0 | 0 | 75 | 0 |

$Mr$ =

| | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| $G_0$ | 1 | 1 | 1 | 0 | 1 |
| $G_1$ | 1 | 0 | 1 | 1 | 0 |
| $G_2$ | 0 | 1 | 1 | 1 | 1 |
| $G_3$ | 1 | 0 | 0 | 1 | 0 |

**Discount function** $f_i$ **:**

Discount[i ] =
- 10, if the cost per purchase is larger than 50;
- 20, if the cost per purchase is larger than 100;
- 35, if the cost per purchase is larger than 150;
- 20% off the cost per purchase, if the cost per purchase is larger than 200;

**Step 1**

In this case, assume all sellers use the same discount policy

$P_{\min}$ =

| $G_0$ | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| 88 | 30 | 45 | 75 |

$SV_{\min}$ =

| $G_0$ | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| $S_4$ | $S_0$ | $S_4$ | $S_3$ |

*MaxCost* = 218

**Step 2**

*MaxDiscount* =

| $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| 40 | 20 | 35 | 35 | 20 |

| | | | | |
|---|---|---|---|---|
| $G_0$ | $G_0$ | $G_0$ | $G_1$ | $G_0$ |
| $G_1$ | $G_3$ | $G_1$ | $G_2$ | $G_1$ |
| $G_3$ | | $G_2$ | $G_3$ | |

*BundleCost* =

| $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| 160 | 123 | 148 | 137 | 113 |

*MaxSubCost* =

| $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| 183 | 133 | 163 | 152 | 123 |

**Step 3**

Current Bundles are all minimal bundles with these discounts

No any *BundleCost[j]* ≥ *MaxSubCost[j]*

*Gain Ratio* =

| $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| 0.13 | 0.075 | 0.09 | 0.09 | 0.08 |

**Step 4**

*PV* =

| $G_0$ | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| $S_0$ | | $S_0$ | $S_0$ |

*PC* = 163

*Mp* =

| | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| $G_0$ | 0 | 0 | 0 | 0 | 0 |
| $G_1$ | 0 | 0 | 0 | 0 | 0 |
| $G_2$ | 0 | 48 | 50 | 58 | 45 |
| | 0 | 0 | 0 | 0 | 0 |

*Mr* =

| | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| $G_0$ | 0 | 0 | 0 | 0 | 0 |
| $G_1$ | 0 | 0 | 0 | 0 | 0 |
| $G_2$ | 0 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 0 |

**Step 5**

Go back to Step 2 with new *Mp* and *Mr*

*PV* =

| $G_0$ | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| $S_0$ | $S_2$ | $S_0$ | $S_0$ |

*PC* = 200

**Figure 4: An Example Implementation by Using MGBS Algorithm**

In the providing matrix $M_r$, the value of each entry $R_{ij}$ only could be either "1" or "0". "1" means seller $S_j$ provides $G_i$. "0" means seller $S_j$ cannot provides $G_i$ for the current purchase goal. The providing matrix here is mainly a supportive matrix, which helps to allocate the identifiers of sellers and goods.

At the beginning of the MGBS algorithm, we need to calculate the minimal price vector $P_{min}$. Then we apply the heuristic search rule 1 (Maximal Bundle) to calculate the maximal bundle purchase from each seller based on its own discount function. If the gain of the maximal bundle for every seller is negative, we quit the bundle search and make the purchase decision based on the minimal price vector. Otherwise, we refine this bundle by applying the heuristic search rule 3 (Bundle Regression) and calculating the best discount ratio from each seller with the same discount according to the inverse discount function. After this, we pick up the seller with the maximal gain ratio to put the goods of this bundle from this seller into the purchase decision vector. Then, we set all entries related to these goods in the price matrix and proving matrix. Repeatedly run this algorithm until the purchase decision list is filled completely. Figure 3 shows the complete algorithm.

In Figure 4, we use the example in Section 2 to demonstrate this algorithm. Actually, the optimal result of this example is the same one that is obtained by the MGBS algorithm.

# 4. ALGORITHM ANALYSIS

The MGBS algorithm is extremely efficient and has very low complexity cost compared with other algorithms.

## 4.1 Complexity Analysis

The number of goods a buyer needs to purchase is *M*, and *N* sellers can provide some or all items in *G*. Since we use the maximal bundle search strategy, the rounds of the maximal bundle search must be less than M in the worst case. During each round, we need to calculate the costs, discounts, the sums of the corresponding minimal single price and the gain ratios of maximal bundles for all sellers. Those are *4N* times. Additionally, there are *N* computations for finding the minimal goods bundles for maximal discounts in the worst case. In other words, there are *CN* times of computation during each round, and *C* is a constant. The total time complexity of the MGBS algorithm is *CNM*, which is *O(NM)*, a very low time complexity compared with the time complexity $O(M^N)$ for the exhaustive search algorithm. In fact, as shown in the experimental evaluation, the worst case of MGBS algorithm rarely happens.

There is a procedure in the MGBS algorithm called *FindMinBundle*, which is used for bundle regression. One must find the minimal combination from a price vector. The sum of this minimal combination should be equal to or larger than a certain threshold. We use another heuristic algorithm to solve this problem. Since more expensive goods affect the amount of discount much more than cheaper goods (because of the assumption that the more one buys, the more discount one gets), the algorithm removes more expensive items in the prices vector. The bundle regression algorithm sorts the prices increasingly, and removes the most expensive one from the price vector. If doing so makes the sum less than the threshold, the algorithm adds the item back. Otherwise, it tests the second most expensive item, continuing until all items are tested. The sum of the remaining items in the price vector is the result The time complexity of this algorithm is *O(V)*, where *V* is the size of the price vector and $V \leq N$.

## 4.2 Correctness Analysis

As mentioned earlier, the upper bound of this algorithm is that the sum of minimal prices for each goods subtracts the possible discounts. Furthermore, since the MGBS algorithm tries to obtain the higher discount ratio rather than the higher amount of discount, the result is much better than the upper bound in general. The only case where the MGBS algorithm may not work properly is when the discount ration of a maximal bundle happens to be the local optimal point (i.e. Local Optima).

# 5. EXPERIMENTAL EVALUATION

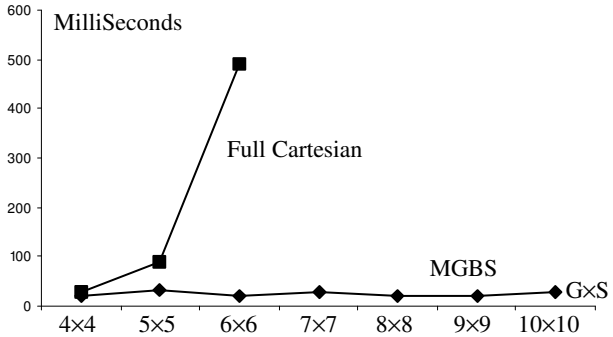


**Figure 5: The Execution Time of Different Algorithms**

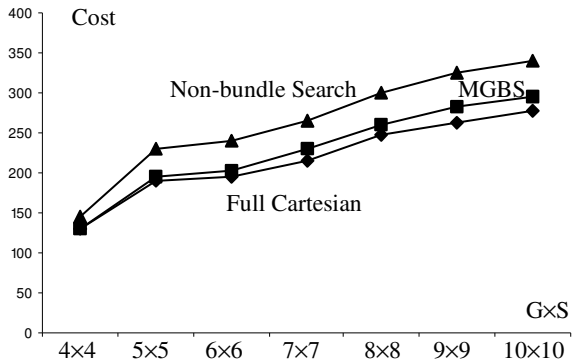


**Figure 6: The Results of Different Algorithms**

We implemented both the MGBS algorithm and the full Cartesian algorithm for our experiments. We used Java 1.4.1 to implement the algorithms and ran them under Windows 2000. The experiments ran on a Dell PC with 800MHz Pentium III CPU and 256MB memory. We used two-dimensional arrays to store the price matrix and the providing matrix. Another issue is the memory requirement. For MGBS algorithm, we reused variables for each subsequent round. For the full Cartesian algorithm, we used a recursive algorithm to only keep the information about the current evaluation and the current optimal purchasing path. Hence, in our implementations of each algorithm, memory access is not expensive at all.

Our test data constructed by $N \times N$ matrices, meaning there are *N* available sellers for this *N* goods bundle. We tested $4 \times 4$, $5 \times 5$, $6 \times 6$, $7 \times 7$, $8 \times 8$, $9 \times 9$, $10 \times 10$ matrices for both the MGBS algorithm and the full Cartesian algorithm. The MGBS algorithm is not sensitive to the increase of the number of goods and sellers at all. The results obtained from the MGBS algorithm are also

close to optimal results and much lower than the upper bounds. Figure 5 shows the execution time of different sizes of matrices. It is hard to put the execution times of $7 \times 7$, $8 \times 8$, $9 \times 9$, $10 \times 10$, because the magnitude is too high compared with the execution time of MGBS algorithm. Figure 6 shows the final results of different sizes of matrices. The results of the MGBS algorithm are close to the optimal results.

Currently, the discount function used in the experiment is the one in Figure 4. It is constructed based on one of the discount policies of J.C.Penney. We will test more discount policies and try to get some real data to test our algorithm in the future.

# 6. CONCLUSIONS AND FUTURE WORK

In this paper, we address a bundle search problem for buyers in electronic combinatorial trade. There are multiple available sellers for each item in the bundle. Each seller can provide multiple goods in the bundle. Sellers have widely varying prices for identical items and various discount ratios for different purchase costs. We propose a heuristic bundle search algorithm, Maximal Gain Bundle Search algorithm, to solve this problem. The time complexity of the MGBS algorithm is $O(NM)$ in the worst case. It is extremely efficient compared with the full Cartesian algorithm, which has a time complexity of $O(M^N)$. The upper bound of the MGBS algorithm is that the sum of the minimal prices of all goods subtracts the possible discount for the bundle. The simulation data show that the execution time of the MGBS algorithm is not sensitive to an increase of the number of goods and sellers. The simulation results also show that the MGBS algorithm can produce nearly optimal bundle purchases.

The bundle search problem in this paper is different from the travel package search problem in [6] because the discount comes from the total purchase cost for each seller instead of the partnership between sellers. Also in the travel package search problem, each seller only provides one kind of good in general. The discount ratio for partnership is normally fixed, but the discount ratio of the total purchase cost from one seller is non-monotonically increasing because the discount function of a seller is most likely not linear. Hence, the phase pruning strategy does not work well for our bundle search problem, because the optimal previous result may not be the optimal result after adding a new stage due to the non-monotonically increasing discount ratio, which is the same reason why dynamic programming [11] does not work well for this problem.

However, in real applications, the discount for a bundle search could come from both partnership among sellers and the total purchase amount from one seller. Developing a heuristic algorithm for a bundle search problem that considers both discount sources is a promising direction for future research.

The bundle search problem addressed in this paper could be extended to many complex bundle search problems for real applications in the future. For example, for industrial buyers, the quantity of each kind of good could be a large number. Is the MGBS algorithm able to handle this situation properly? Another interesting issue concerns time. In real electronic markets, many purchases happen during specific seasons or occasions, and sellers always have some seasonal sales. Buyers could adjust their purchase goals to utilize the seasonal sales. Buyers have to make purchasing decision before the sale deadline. Considering these time issues, making a good bundle purchase plan is a very interesting problem.

In electronic markets, the result of a bundle search for buyers can be the input of many other purchasing strategies. For instance, it can be an input for buyer coalition formation. According to our knowledge, currently, most buyer coalition formation work focused on purchasing one [8,9] or at most two [1] kinds of goods from only one seller. Bundle search for buyers provides a possibility to form buyer coalitions even if buyers have different target goods that may be available from different sellers.

Another promising application of bundle search for buyers is "Service Composition", which is the strategy of taking several component products or services, and bundling them together to meet the needs of a given customer [10]. The service composer needs to minimize the cost of the composition. In [10], the service composition is constructed by combinatorial auction. In more general cases, we can view a service composition as a bundle search for a buyer.

# 7. Acknowledgement

This work was supported in part by MURI grant #F49620-00-1-0326 from DoD and AFOSR.

# 8. REFERENCES

[1] Ye, Y., and Tu, Y., Dynamics of coalition formation in combinatorial trading, in proceedings of International Joint Conference of Artificial Intelligence (IJCAI), 625-630, 2003.

[2] Yokoo, M., Sakurai, Y., and Matsubara, S., Robust combinatorial auction protocol against false-name bids, in Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI), 110-116, 2000.

[3] Sandholm, T., Suri, S., Gilpin, A., and Levine, D., Winner Determination in Combinatorial Auction Generalizations, in Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), 69-76, 2002.

[4] Sandholm, T., Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence, 135, 1-54, 2002.

[5] Proter, M. E., Competitive Strategy, The Free Press, New York, 1980.

[6] Chang, Y., Li, C., and Smith, J. R., Searching dynamically bundled goods with pairwise relations, in proceedings of ACM Electronic Commerce, 135-143, 2003.

[7] Brynjolfsson, E., and Bakos, Y., Bundling information goods: pricing, profits and efficiency. Management Science, Dec. 1999.

[8] Li, C. and Sycara, K., Algorithm for combinatorial coalition formation and payoff division in an electronic marketplace, in proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), 120-127, 2002.

[9] Lerman, K. and Shehory, O., Coalition formation for largescale electronic markets, in proceedings of the International Conference on Multi-Aent Systems, 2000.

[10] Preist, C., Bartolini, C., and Byde, A., Agent-based service composition through simultaneous negotiation in forward and reverse auction, in proceedings of ACM Electronic Commerce, 55-63, 2003.

[11] Puterman, M. L., Dynamic Programming and Its Applications, Academic Press, 1978.