Consideration Set Generation in Commerce Search

Sayan Bhattacharya[*]
Dept. of Computer Science
Duke University
bsayan@cs.duke.edu

Sreenivas Gollapudi
Microsoft Search Labs
Microsoft Research
sreenig@microsoft.com

Kamesh Munagala[†]
Dept. of Computer Science
Duke University
kamesh@cs.duke.edu

ABSTRACT

In commerce search, the set of products returned by a search engine often forms the basis for all user interactions leading up to a potential transaction on the web. Such a set of products is known as the *consideration set*. In this study, we consider the problem of generating consideration set of products in commerce search so as to maximize user satisfaction. One of the key features of commerce search that we exploit in our study is the association of a set of *important attributes* with the products and a set of *specified attributes* with the user queries. Those important attributes not used in the query are treated as *unspecified*. The attribute space admits a natural definition of user satisfaction via user preferences on the attributes and their values, *viz.* require that the surfaced products be close to the *specified* attribute values in the query, and diverse with respect to the *unspecified* attributes. We model this as a general *Max-Sum Dispersion* problem wherein we are given a set of n nodes in a metric space and the objective is to select a subset of nodes with total cost at most a given budget, and maximize the sum of the pairwise distances between the selected nodes. In our setting, each node denotes a product, the *cost* of a node being inversely proportional to its relevance with respect to specified attributes. The *distance* between two nodes quantifies the diversity with respect to the unspecified attributes. The problem is NP-hard and a 2-approximation was previously known *only* when all the nodes have unit cost.

In our setting, we do not make any assumptions on the cost. We label this problem as the *General Max-Sum Dispersion* problem. We give the *first* constant factor approximation algorithm for this problem, achieving an approximation ratio of 2. Further, we perform extensive empirical analysis on real-world data to show the effectiveness of our algorithm.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

[*]Supported by NSF award CCF-0745761. Work done while the author was an intern at Microsoft Search Labs.

[†]Supported by an Alfred P. Sloan Research Fellowship, and by NSF via CAREER award CCF-0745761 and grant CCF-1008065.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

General Terms

Theory, Experimentation, Performance

Keywords

Approximation Algorithms, Facility Dispersion, Relevance, Novelty

1. INTRODUCTION

Commerce search is gaining significance as more users spend increasing amounts of time searching for products on web. Typically, the shopping verticals of commercial search engines or web portals maintain a catalog (or index) of products, and surface a *consideration set* of products from the catalog that best match the user query. This is akin to the result set the user sees in traditional web search. Unlike web search, the main goal of a consideration set is to aid the user in completing her transaction. Figure 1[1] underscores the importance of a consideration set in commerce search where the entropy of clickthrough rates in the top 10 positions is very high. In fact, it is close to the normalized maximum value of 1.0. This suggests that the user is more likely to browse (or click) multiple results in different positions for commerce queries. Another fundamental difference between commerce search and web search is the presence of structure in both data and queries. Typically the catalog is composed of products in a well-structured or semi-structured format and/or other rich metadata. For example, a digital camera is associated with *attribute* information such as `brand`, `model`, `color`, etc. In fact, this key aspect of the data can be exploited in commerce search. On the query side, commercial search queries are good examples of (partially) structured queries wherein a set of attribute values relevant to the product being searched can potentially be extracted from the query string. For example, a user query such as 10mp Nikon Digital Camera denotes the user's intent to see digital cameras from a particular manufacturer (`Nikon`) having certain desired features (`10 megapixel resolution`).

An important characteristic of the queries in commerce search is that the user does not always associate a commercial query with her intent to buy a product *immediately*. Often, she issues many queries with different keywords (attributes of a product) and their combinations before she decides to buy a product. Further, many of these exploratory queries are often imprecise and incomplete in expressing the

[1]The data for this figure was obtained from mining three months of query and click data from the search log of a commercial search engine.

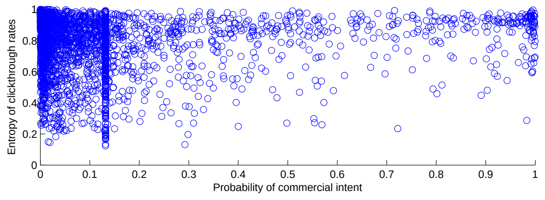

Figure 1: The correlation between the likelihood of commercial intent in a query and the entropy of clickthrough rates in the top 10 positions

user's information need. This arises because of two reasons - 1) the user often does not have the domain knowledge. For example, she may not be aware of the fact that Samsung only makes 55" lcd tvs and not 52" lcd tvs, or for that matter there are no 50" lcd tvs made by any manufacturer; and 2) not all keywords (attributes) that describe a product are used in a search query. The number of keywords typically depends on the intent of the user such as how close is she to buying a product. Let us elaborate with an example. A user looking for digital cameras of a certain resolution might start her search session with *10mp digital cameras* and then work her way toward issuing more specific queries to the search engine such as *Nikon Coolpix 10mp digital camera* or browsing on the result page to the product she is interested in. Note that in both these queries, a large fraction of the product attributes are still not specified. Continuing with the example, there are multiple products in the Coolpix series with 10 megapixel resolution - the *S3000* and *S4000* - with different colors and price points. To substantiate our point, we define the notions of *specified* and *unspecified* attributes that we will use in our study to characterize the space of relevant products for a user query. In our example, the specified attributes are `brand`, `model line`, and `resolution`, while the unspecified attributes correspond to `color`, `optical zoom`, and `model number`.

The above user query model motivates the development of new techniques for generating effective *consideration set* of products in response to user queries. Before we summarize our techniques for effective consideration set generation, we need to introduce another important aspect of the user query model that we assume and exploit in our study.

Though the query acts as a point of reference to compute the closeness of a product with respect to the specified attributes, there is nothing similar to deal with the unspecified attributes. User preferences over different attributes will help us fill this void. A user preference function is typically defined over the attribute value space for any given attribute. Even though the user does not explicitly and exactly specify her information need through the user query, she will value the consideration set more if the search engine surface products with attribute values that are close to or exceed her preferences. Again, consider the example of the digital camera. If the S3000 has (nearly) the same attribute values as the S4000 but comes with a lower cost, then surfacing the lower priced camera could be considered a good search result by the user. Extending this to multiple attributes each with different preference functions makes the problem challenging.

Contributions of this study

In this study, we propose methods for generating the consideration set in commerce search that exploit *both* the specified and unspecified attributes and user preferences to maximize user satisfaction. We observe that each product can be represented as an attribute vector and each query as a list of attribute values. It is natural to require that the surfaced products be close to the specified attribute values in the query, and diverse with respect to the unspecified attributes. This motivates the following Max-Sum Dispersion problem. We are given n nodes in a metric space, each node denoting a product and *cost* of a node being inversely proportional to its relevance with respect to specified attributes in the query. The distance between two nodes (with respect to unspecified attributes) captures the notion of novelty or coverage. The objective is to select a set of nodes with total cost not more than a given budget, and maximize the sum of the pairwise distances between the selected nodes. This problem is known to be NP-hard, and no constant factor approximation was known previously. We give the first 2-approximation algorithm for this problem along with some implementation results. We note that our formulation is general enough to admit attributes and their values with associated importance scores. These scores admit a natural preference a user might have over an attribute (value) with respect to the other attributes (values). In fact, we incorporate the relative importance of attributes and their values in our experiments and show that our methods extend to these settings as well.

2. RELATED WORK

The work on diversification of search results has looked into similar objectives as ours where the likelihood of the user finding *at least* one result relevant in the result set forms the basis of the objective function. One of the early influential work on diversification is that of Maximal Marginal Relevance (MMR) presented by Carbonell and Goldstein in [5]. In their work, a trade-off between novelty (a measure of diversity) and the relevance of search results is made explicit through the use of two similarity functions, one measuring the similarity among documents, and the other the similarity between document and query. A parameter controls the degree of trade-off.

Vee *et al* [15] study diversification in the context of structured database with applications to online shopping. In their work, items can be represented as a set of features or attributes, and the objective is to select a set of items that are as diverse as possible according to a lexicographical ordering of attributes. In contrast, we propose an algorithm that admits general distance functions that satisfy the metric property and user preferences over attribute values in addition to attribute importance. Further, we provide strong theoretical guarantees about the performance of our algorithm.

Zhai and Lafferty [18] formalize the notion that it is in general insufficient to simply return a set of relevant results and that the correlations among the results are also important. Specifically, they propose a risk minimization framework for information retrieval that allows a user to define an arbitrary *loss* function over the set of returned documents. Bookstein [4] and Chen and Karger [7] both consider information retrieval in the context of ambiguous queries. The basic idea in these works is that documents should be selected sequentially according to the probability of the document being relevant *conditioned* on the documents that come before. Radlinski, Kleinberg, and Joachims [11] propose a learning algorithm to compute an ordering of search results from a diverse set of orderings. By iterating through all documents in each of the positions while holding fixed the documents in the other positions, they attempt to learn a "best" ranking of documents using user clicks. Agrawal *et al.* [1] take a different approach that makes use of a taxonomy for classifying queries and documents, and creates a diverse set of results in accordance to this taxonomy.

In approximation algorithms literature, all previous works on Max-Sum Dispersion considered the *Uniform-cost* version (when each node has unit cost) of the problem. Another closely related problem is that of Max-Min Dispersion where the objective is to maximize the minimum pairwise distance of the selected nodes subject to a budget constraint. Both the Uniform-cost Max-Sum and Uniform-cost Max-Min Dispersion problems are NP-hard [8, 17]. Ravi, Rosenkrantz, and Tayi [12] give a 2-approximation for Uniform-cost Max-min dispersion, and shows that it is NP-hard to obtain an approximation ratio better than 2. For Uniform-cost Max-sum dispersion, they provide an efficient heuristic with approximation guarantee of 4, Hassin *et. al.* [9] give a different algorithm achieving an approximation ratio of 2. Birnbaum *et. al.* [3] show, using factor-revealing LPs, that the efficient heuristic proposed in [12] has in fact an approximation ratio of 2 for Uniform-cost Max-sum dispersion. Finally, Rosenkrantz, Tayi, and Ravi [13] consider the Max-min dispersion problem and give a 2-approximation for the general case, along with other extensions.

We conclude this section by mentioning that there is an extensive literature on *multiattribute utility theory* and its applications in commerce search that is closely related to our work. The interested reader is referred to the article by Wallenius *et. al.* [16] for an excellent survey of this field.

3. PROBLEM FORMULATION

First, we observe that each product can be characterized as an attribute vector, and given a query, attributes may be classified as *specified* or *unspecified* (with respect to the query). Moreover, an attribute is either *monotone* or *single peaked*. For example, the price and resolution of a camera are monotone attributes. Everyone wants a cheap camera with high resolution. On the other hand, attributes like color are single-peaked. We assume that a user wants a consideration set of products that are *close* to the specified attribute values, and at the same time *diverse* with respect to the unspecified attributes.

Therefore we can model the above mentioned situation as follows. We are given an undirected complete graph $G = (V, E)$, a weight function $d : E \rightarrow \mathbf{R}$, and a cost function $c : V \rightarrow \mathbf{R}$. Each node represents a product, the weight of an edge captures the distance between the corresponding products with respect to unspecified attributes, and the cost of a node captures the distance of the corresponding product from the specified attribute values. Throughout the rest of the paper, we will use the terms *weight of an edge* (x, y) and *distance between nodes* x, y interchangeably. Our objective is to select a set of nodes S with small total cost $c(S) = \sum_{v\in S} c(v)$ and high *dispersion* $\mathrm{Disp}(S)$. The function $\mathrm{Disp}(S)$ captures how far away the nodes in S are from each other.

Two most natural measures of dispersion are *Max-min Dispersion* and *Max-sum Dispersion*. Max-min dispersion of a set of nodes S is defined as $\min_{u,v\in S}\{d(u, v)\}$, that is, the minimum pairwise distance between the nodes. On the other hand, Max-sum dispersion of a set S is defined as $\sum_{x,y\in S} d(x, y)$, that is, the sum of the pairwise distances of the nodes. Intuitively, dispersion of a consideration set should increase as more products are added to the set. However, we note that the Max-min dispersion function is not monotone: Consider two sets $S \subset S'$. It may happen that Max-min dispersion of S is strictly greater than that of S'. Since Max-sum dispersion does not suffer from this potential drawback, we define $\mathrm{Disp}(S)$ to be the Max-sum dispersion of S.

Recall that we want to select a set of nodes with small cost and large dispersion. One possible way to formulate the problem is to maximize $\mathrm{Disp(S)} - \lambda \times c(S)$, where λ is some predefined constant. Note that the problem can then be described as a LP relaxation.

$$\text{Maximize} \sum_{u,v\in V} d(u, v)x_{uv} - \lambda \sum_{v\in V} c(v)y_v \qquad \text{LP}$$

$$x_{uv} \le \min\{y_u, y_v\}\ \forall u, v \in V \qquad (1)$$

$$0 \le x_{uv}, y_u, y_v \le 1\ \forall u, v \in V \qquad (2)$$

Each y_v is an indicator variable denoting whether or not the node v has been included in the consideration set S. Furthermore, the variable x_{uv} is set to 1 if and only if $y_u = y_v = 1$. Let $\{\tilde{x}_{uv}, \tilde{y}_u, \tilde{y}_v\}$ denote the optimal solution to the above LP. Pick an α uniformly at random from the interval $[0, 1]$. For each node $v \in V$, include v in the consideration set iff $\alpha \le \tilde{y}_v$. It is easy to check that this rounding scheme preserves the optimal objective value in expectation. Hence the optimal solution can be found in polynomial time. However, this formulation suffers from a drawback. It may so happen that the total cost of the selected nodes is too high, and this will defeat the intuition that the surfaced products should be close to the specified attributes. The best way to get around this difficulty is to maximize $\mathrm{Disp}(S)$ subject to the constraint that $c(S)$ is at most some predefined budget B. It will be termed as the Max-Sum dispersion problem, and the main result of our paper is a 2-approximation algorithm for this setting, provided the edge weights satisfy triangle inequality. For completeness, we note that without triangle inequality, the problem may be hard to approximate: When all distances are either 0 or 1 and all nodes have unit cost, the problem is equivalent to finding the B-densest subgraph, that is, finding a set of B nodes such that the induced subgraph has maximum number of edges, and the best known approximation ratio is $O(|V|^{1/4})$ [2].

While the max-sum dispersion formulation addresses the issue of potentially unbounded cost of the consideration set, it has its own drawback related to the quality of unspecified attribute values of the products it surfaces. Consider the query `brand` = *Nikon*; `Color` = *black*; `Category` = *digital camera*. Note that resolution of the camera is an unspecified attribute, and our algorithm will ensure that the products are as diverse as possible with respect to resolution. However, resolution is also a monotone attribute: We should be surfacing only high resolution cameras. Consider two different cameras x and y with different resolutions. The distance

```
UniformGreedy($V, B$)
if $B = 0$ then
    Output $\emptyset$;
end
if $B = 1$ then
    Output any node $x \in V$;
end
Find an edge $(x, y) \in \text{argmax}\,\{d(u, v) : u, v \in V\}$;
$B' \leftarrow B - 2$;
$V' \leftarrow V \setminus \{x, y\}$;
Output $\{x, y\} \cup$ UniformGreedy($V', B'$);
```

Algorithm 1: Greedy algorithm with uniform costs

between the two, $d(x, y)$ does not capture the monotonicity. Still we can remedy the situation by considering a new distance function $d'(x, y) = d(x, y) + w(x) + w(y)$, where $w(x)$ (resp. $w(y)$) denotes the *importance* of the resolution value of x (resp. y). In this case, a camera with higher resolution has higher importance. We note that this works even with multiple monotone attributes. If the underlying distance $d(u, v)$ is a metric, then $d'(u, v)$ is also a metric and therefore all our theoretical guarantees hold for $d'(u, v)$ as well.

Another potential drawback of the max-sum dispersion objective is that the algorithm may select a product that is far away from the *natural* consideration set. In such a situation, however, the outlier will have a high cost, and the algorithm will quickly exhaust the budget if too many such outliers are selected.

4. THE ALGORITHM

Hassin *et. al.* [9] proposed a simple 2-approximation for Uniform-cost Max-Sum dispersion (see Algorithm 1). The procedure UniformGreedy takes the set of nodes V and a budget B as inputs, finds the edge (x, y) with highest weight, outputs its two endpoints, and calls itself recursively with $V \setminus \{x, y\}$ and a budget of $B - 2$. In this section, we extend their result to give a 2- approximation for general Max-Sum dispersion where different nodes can have different costs.

4.1 Our Approach

First, consider the special case when the cost function takes only small number of different values. In other words, suppose there is a partition of the set of nodes V into k *buckets* $V_0, \ldots V_{k-1}$, such that all nodes in the same bucket have same costs. Let $\vec{w} = (w_0, \ldots, w_{k-1})$ be the *demand* vector whose i^{th} component denotes the number of nodes from bucket V_i selected by the optimal solution. We can *guess* the vector $\vec{w}$ in time $O(n^k)$, where $n = |V|$, and it is a polynomial in n as long as k is a constant. In Section 4.2, we describe an extension to UniformGreedy that finds a subset of nodes S subject to the constraint induced by vector $\vec{w}$. Furthermore, we show (see Theorem 4.1) that it is a 2-approximation to the optimal solution. The proof of Theorem 4.1 closely follows the argument in [9].

Next, consider the general situation when all the cost values can potentially be different from each other. Fix some $\epsilon > 0$, and suppose we are allowed to overshoot the budget by a factor of at most $(1 + O(\epsilon))$. Without any loss of generality, the maximum node cost is bounded from above by B, the budget. Further, the minimum node cost is bounded from below by $\epsilon B/n$, since we can include in our solution all the nodes below this threshold with a total additional cost of ϵB. Now discretize the range from $\epsilon B/n$ to B in powers of $(1+\epsilon)$, getting L levels, and round every cost value down to the nearest level below it. Clearly, $(\epsilon B/n)(1 + \epsilon)^L = B$, implying $L = (\log n - \log \epsilon)/\log(1 + \epsilon) \leq (\log n)/\epsilon^2$, for sufficiently small ϵ. Let $\tilde{c}(x)$ denote the rounded cost value of node x, for all $x \in V$. Thus, the set of all nodes has been effectively partitioned into $(\log n)/\epsilon^2$ buckets, and we can guess how many nodes from each bucket is selected by the optimal solution in $O(n^{(\log n)/\epsilon^2})$ time. We can now run the greedy algorithm from Section 4.2 to get a 2 approximation. Furthermore, note that for any subset of nodes S, if $\tilde{c}(S) = \sum_{v \in S} \tilde{c}(v) \leq B$, then $c(S) = \sum_{v \in S} c(v) \leq (1 + \epsilon)B$. In Section 4.3, we proceed to discretize the cost values even further in such a way that brings down the running time to a polynomial in n while preserving the approximation factor.

```
Greedy($V, \vec{w}$)
if $w_i = 0$ for all $i \in \{0, \ldots, k-1\}$ then
    Output $\emptyset$;
end
if $w_l = 1$ for some $l$ and $w_i = 0$ for all $i \neq l$ then
    Output any node $x_l \in V_l$;
end
Let $(x_i, x_j)$ be the feasible edge with maximum weight;
Let $x_i \in V_i$, and $x_j \in V_j$ ;
if $i = j$ then
    $w'_i \leftarrow w_i - 2$;
    $V'_i \leftarrow V_i \setminus \{x_i, x_j\}$;
else
    $w'_i \leftarrow w_i - 1$;
    $w'_j \leftarrow w_j - 1$;
    $V'_i \leftarrow V_i \setminus \{x_i\}$;
    $V'_j \leftarrow V_j \setminus \{x_j\}$;
end
for all $t \neq i, j$ do
    $V'_t = V_t$ ;
    $w'_t = w_t$ ;
end
$V' = V'_0 \cup \ldots \cup V'_{k-1}$. ;
$\vec{w'} = (w'_0, \ldots, w'_{k-1})$. ;
Output $\{x_i, x_j\} \cup$ Greedy($V', \vec{w'}$);
```

Algorithm 2: Greedy Algorithm for small number of distinct costs

4.2 Small Number of Distinct Cost Values

We begin with a simple definition.

Definition 4.1. *Suppose the set of nodes V has been partitioned into k buckets $V_0 \ldots V_{k-1}$. Furthermore, let $\vec{w} = (w_0, \ldots, w_{k-1})$ denote the demand vector. An edge $(x_i, x_j) \in E$, where $x_i \in V_i, x_j \in V_j$ is* feasible *if 1) $i = j$ and $w_i \geq 2$, or 2) $i \neq j$ and $w_i, w_j \geq 1$.*

For a given partition of V into buckets $V_0 \ldots V_{k-1}$, and demand vector $\vec{w} = (w_0, \ldots, w_{k-1})$, let $\text{OPT}(V, \vec{w})$ denote the subset of nodes with maximum dispersion that includes exactly w_i nodes from bucket V_i, for all $i \in \{0, \ldots, k-1\}$.

Theorem 4.1. *Algorithm* Greedy *is a 2-approximation to OPT$(V, \vec{w})$.*

We prove Theorem 4.1 by induction on the demand vector $\vec{w}$. Suppose the claim holds for any demand vector strictly dominated by $\vec{w}$. To get some intuition behind the proof, consider the first edge (x_i, x_j) selected by GREEDY, and suppose both its endpoints are also included in $\text{OPT}(V, \vec{w})$. Now consider some other node u (resp. v) selected by $\text{OPT}(V, \vec{w})$ (resp. $\text{GREEDY}(V, \vec{w})$). Since (x_i, x_j) has maximum weight amongst the feasible edges, $d(u, x_i) + d(u, x_j) \le 2d(x_i, x_j)$. Since the edge weights satisfy triangle inequality, $d(v, x_i) + d(v, x_j) \ge d(x_i, x_j)$. Thus, we have $d(u, x_i) + d(u, x_j) \le 2(d(v, x_i) + d(v, x_j))$, and we can use induction hypothesis on the set $V' = V \setminus \{x_i, x_j\}$ and the adjusted demand vector $\vec{w'}$ (see Algorithm 2) to get the desired 2-approximation. The complete proof is described in appendix.

4.3 Extending to General Cost Functions

The complete procedure is outlined in Algorithm 3. Let V_0 denote the set of nodes with cost at most $\epsilon B/n$. As explained in Section 4.1, we can output all the nodes in V_0. Now discretize the range from $\epsilon B/n$ to B in powers of $(1+\epsilon)$, getting L levels, and round every cost value down to the nearest level below it. Let $\tilde{c}(x)$ denote the rounded cost value of node x, for all $x \in V \setminus V_0$. It results in the set $V \setminus V_0$ being partitioned into L buckets $V_1 \dots V_L$, where $L = (\log n)/\epsilon^2$. Let w_l denote the number of nodes selected from bucket V_l in the optimal solution. Define $\tilde{c}_l = \tilde{c}(x)$ for any node $x \in V_l$. Let $w_l \tilde{c}_l \in \left[t_l(\epsilon^3 B/\log n), (t_l + 1)(\epsilon^3 B/\log n)\right]$, for some integer $t_l \in \{0, \dots, (\log n)/\epsilon^3 - 1\}$. Under these circumstances, we will round the total cost of the nodes selected from bucket V_l down to $t_l(\epsilon^3 B/\log n)$. In other words, we are pretending that the contribution from each bucket is an integral multiple of $\epsilon^3 B/\log n$. We may underestimate the contribution from each bucket by at most $\epsilon^3 B/\log n$. Since there are $(\log n)/\epsilon^2$ buckets, we can overshoot the budget by at most $((\log n)/\epsilon^2) \times (\epsilon^3 B/\log n) = \epsilon B$. Thus, we get the following Lemma (complete proof appears in appendix).

LEMMA 4.2. *The set of nodes returned by the modified greedy algorithm has a total cost of at most $(1 + O(\epsilon))B$, for sufficiently small $\epsilon > 0$.*

Note that we need to *guess* $L = (\log n)/\epsilon^2$ integers $t_1 \dots t_L$ such that each $t_l \in \{0, \dots, (\log n)/\epsilon^3 - 1\}$, and $\sum_l t_l = (\log n)/\epsilon^3$. It was shown in [6] that there are only polynomially many tuples satisfying the above mentioned condition. For the sake of completeness, we prove the statement of Lemma 4.3 in appendix.

LEMMA 4.3. *[6] For any fixed $\epsilon > 0$, the number of different L-tuples $(t_1, \dots, t_L)$ is polynomial in n whenever $L = (\log n)/\epsilon^2$, $\sum_{l=1}^{L} t_l = (\log n)/\epsilon^3$, and each t_l is a non-negative integer.*

We can thus guess the correct demand vector in polynomial time and run GREEDY to get a 2-approximation.

LEMMA 4.4. *Algorithm* MODIFIEDGREEDY *gives a* 2-*approximation to the optimal solution.*

PROOF. Suppose the optimal solution satisfying the budget constraint includes exactly k_l nodes from bucket V_l. MODIFIEDGREEDY guesses in polytime a demand vector $\vec{w'} = (w'_0, \dots, w'_L)$ such that $w'_l \ge k_l$, for each $l \in \{0, \dots, L\}$. MODIFIEDGREEDY returns the set $\text{GREEDY}(V, \vec{w'})$, and Theorem 4.1 implies that it is a 2-approximation to $\text{OPT}(V, \vec{w'})$.

```
MODIFIEDGREEDY$(V, B)$
$V_0 \leftarrow \{v \in V \mid c(v) \le \epsilon B/n\}$;
$V' \leftarrow V \setminus V_0$;
Partition $V'$ into $L$ buckets $V_1, \dots, V_L$ such that for
each $l \in \{1, \dots, L\}$, $V_l \leftarrow$
$\{v \in V' \mid (\epsilon B/n)(1+\epsilon)^{l-1} < c(v) \le (\epsilon B/n)(1+\epsilon)^l\}$;
for $l \in \{1, \dots, L\}$ do
    $\tilde{c}_l \leftarrow (\epsilon B/n)(1+\epsilon)^{l-1}$;
    for $v \in V_l$ do
        $\tilde{c}(v) = \tilde{c}_l$;
    end
end
For each $l \in \{1, \dots, L\}$, guess
$t_l \in \{0, 1, \dots, (\log n)/\epsilon^3 - 1\}$ s.t. $\sum_{l=1}^{L} t_l = (\log n)/\epsilon^3$;
for $l \in \{1, \dots, L\}$ do
    Select the largest integer $w_l$ s.t.
    $w_l \tilde{c}_l \le (t_l + 1)(\epsilon^3 B/\log n)$;
    $w'_l \leftarrow \min(w_l, |V_l|)$;
end
Let $w'_0 = |V_0|$;
Let $\vec{w'} \leftarrow \{w'_0, w'_2, \dots, w'_L\}$;
$S \leftarrow$ GREEDY$(V = V_0 \cup V_1 \cup \dots \cup V_L, \vec{w'})$;
OUTPUT $S$
```

Algorithm 3: Modified Greedy Algorithm For General Cost Functions

Since the function $\text{Disp}(S)$ is monotone in S, dispersion of the set $\text{OPT}(V, \vec{w'})$ is at least the optimal objective value subject to the budget constraint. The lemma follows. □

Combining the previous lemmas, we get the main result of this section.

THEOREM 4.5. MODIFIEDGREEDY *runs in polytime for any fixed sufficiently small $\epsilon > 0$, overshoots the budget by a factor of at most $(1 + O(\epsilon))$, and is a* 2-*approximation to the optimal solution.*

5. EXPERIMENTS

In this section, we do an empirical evaluation of our model and algorithms described in the earlier sections. We primarily evaluated the performance of our algorithm, MODIFIEDGREEDY along two dimensions - the quality of results, and the diversity of the products with respect to the unspecified attributes in the query.

5.1 Setup

We used to two different ranking functions to compare our algorithms against. In the first setup, we used a prototype commerce search engine serving results from an index of about 30 million products taken from the Bing shopping catalog[2]. We use the technique described in [14] to annotate the query with attribute information. For the second ranking function, we used the shopping vertical of a commercial search engine. In both the cases, we compared the quality of the results produced by our algorithms with those produced by the respective ranking functions. In order to make the comparison, we need attribute information related the products in the consideration set in each case. In the prototype

[2]http://shopping.bing.com

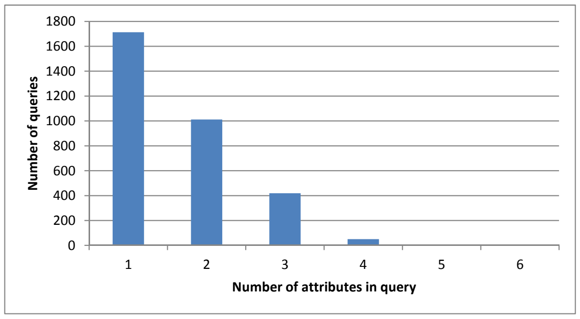

Figure 2: The histogram of queries with number of extracted attributes

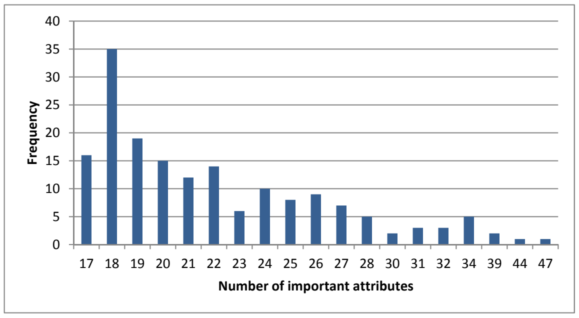

Figure 3: The histogram of number of categories with a given number of attributes

search engine, the search index was generated using the full text descriptions and the structured attribute information of the products. Thus, the consideration set produced by the baseline structured ranking function is already enriched with attribute information. In the case of the shopping vertical, we extracted the attributes of the products in the consideration set from the product pages surfaced by the search engine and we mapped the attributes into the space of important attributes used in our prototype commerce search engine.

As our query set we used a random sample of 3200 queries from the shopping vertical of Bing. We extracted the attribute value pairs for these queries. Naturally, our annotator extracted structured information to different degree from these queries. Figure 2 shows the histogram of queries with number of structured attributes extracted from them. Further, we categorized these queries using a Naiv̈e Bayes classifier and found them to be spread across 173 categories in our product taxonomy. Another important statistic is that the average length of the query in our test set was 4.4 terms. In all the experiments, we considered the *important* attributes of a category, i.e., attributes that are frequently asked by the user or occur in a large fraction of the products[3] in the index. In our case, we set the selectivity threshold to 0.75. This yielded us the histogram shown in Figure 3 for the 173 categories we consider in our experiments. This gives us an idea of the number of unspecified attributes that our algorithm has to deal with.

[3]also known as selectivity of the attribute

5.2 Structured Ranking and Distance Functions

In the prototype search engine, we used a ranking function that exploits structure in both queries and products. It is a simple combination of both structure and text. It exploits structure where it's present (in the query) and falls back to using textual relevance when no structure could be extracted from the query. The relevance with respect to the text is based on the well-known IR features like *BM25F*, *Proximity* of query terms in the product description, and *fractional match* of query terms. The key measure that is used in computing the structural relevance is the distance between corresponding attribute values. We denote the distance between any two attribute values u and v for a given attribute i as $d_i(u, v)$. Specifically, we use the likelihood of an average user preferring v to u as the similarity[4] between u and v. This measure can be applied to both categorical and numeric attributes. For numeric attributes, we define $d_i(u, v) = min(1.0, \frac{|u-v|}{u})$. For categorical attributes, we adopt the method described in [10] which is based extracting user preferences for attribute values from browse trails on the web. We note that this distance function does not satisfy all the properties of a metric.

Next, we describe the application of the distance function in our experiments. We begin with the baseline ranking function. For the specified attributes, the relevance of the product with respect to the query is computed using the L_1-norm of the distance between the corresponding attribute values, i.e., $c(p) = d(q, p) = \sum_i d_i(u, v)$ where u and v are respectively the values of attribute i in the query q and product p. We use this scoring function along with the textual relevance function as the baseline ranking function. We compare the performance of ModifiedGreedy against this function.

Handling Attribute Dependencies

Though we don't explicitly deal with attribute dependencies in this study, our algorithms admit more complex distance functions that incorporate attribute dependencies. For example, the algorithms proposed in [10] can be used to learn user preferences over a set of attribute values. To keep the problem tractable, this computation could be limited to sets of popular attributes and their values.

5.3 Implementation of ModifiedGreedy

We now move on to the implementation of ModifiedGreedy. We will describe how we use the baseline ranking function and the distance between unspecified attributes to implement ModifiedGreedy. We begin with user preferences. Note that here we did not incorporate any user preferences in the baseline structured ranking function. To do so, we need to modify the distance function as

$$d_i^p(u, v) = \begin{cases} 0 & \text{if } v \geq u \\ d_i(u, v) & \text{otherwise} \end{cases}$$

for an attribute i that is monotone-upward, i.e., we don't penalize products with attribute value v greater than the corresponding attribute value u in the query. A similar function is defined for monotone-downward functions as well. For single-peaked attributes, the distance stays unmodified.

[4]$sim_i(u, v) = 1 - d_i(u, v)$

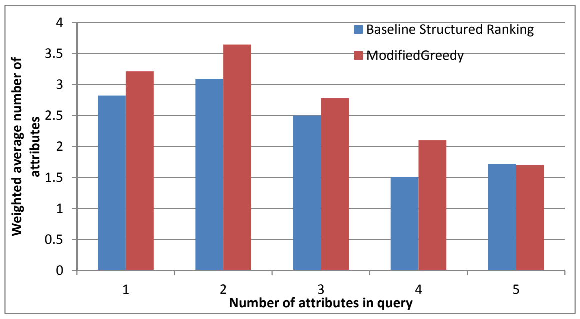

Figure 4: The weighted average number of different attributes for queries grouped by the number of specified attributes in the query

We use the modified distance function along with the textual relevance function used in the baseline structured ranking function to generate the *filter set* of products. Note that the filter set is a intermediate set of relevant products on which we run ModifiedGreedy to generate the consideration set for the user. Typically, the size of the filter set is much larger (e.g., 300) than the size of the consideration set (e.g., 10). Note that the above distance function does not capture monotonicity. To circumvent this problem, we incorporate relative importance of attributes and their values into the distance function as described in Section 3. The computation of importance values is not the focus of this study and we therefore assume that the importance values are pre-computed and available to our algorithm.

Selection of the vector $\vec{w}$

We also made a few changes to ModifiedGreedy to make it run efficiently in the prototype search engine. The key step is the guessing of the integers t_i for $i = 1, 2, \cdots, L$. For each bucket we toss a coin with bias $1 - \frac{c_i}{C}$ where C is the total cost of all the buckets. Constructing $\vec{w}$ this way has the desired property of picking elements from buckets with smaller costs and thereby resulting in products which are closer to the user query (in terms of specified attributes).

5.4 Comparison with the Baseline Structured Ranking Function

We ran a set of experiments to compare the performance of ModifiedGreedy against the baseline ranking function described in the previous section. In all these experiments, we used the top 10 results surfaced for each query in our query set. Along with the basic attribute information such as name of the product, manufacturer and price, we extracted information associated with all the important attributes of each product. In the first experiment, we compared the amount of important yet diverse information surfaced by both the algorithms. To do so, we compute the number of different attributes surfaced in the result set weighted by their importance. Figure 4 illustrates the weighted average number of attributes for different number of specified attributes in the query. As the figure shows, the more general the query (corresponds to less number of attributes), the better ModifiedGreedy performs with respect to the baseline. As the queries get more specific (i.e., have more attributes), both the algorithms perform equally well.

Next, we compare the relevance of the consideration set

Number of	Baseline			ModifiedGreedy		
attributes	Min	Max	Avg	Min	Max	Avg
1	0.217	0.411	0.346	0.266	0.407	0.341
2	0.050	0.161	0.104	0.064	0.189	0.123
3	0.116	0.275	0.196	0.122	0.231	0.170
4	0.142	0.418	0.263	0.143	0.332	0.209
5	0.205	0.431	0.328	0.205	0.270	0.136

Table 1: Relevance of the consideration set with respect to the specified attributes of the best, worst, and average document in the set, averaged over all queries in test set

in both cases by computing the minimum, maximum, and the average distance of the products from the query over the specified attributes. These three measures give us a sense of how close is the consideration set to the query in terms of the best, worst and the average result in the set. Table 1 highlights the relative performance of both the algorithms. Again, we compare the corresponding values by bucketing the queries into buckets based on the number of attributes extracted from the query.

As the results show, ModifiedGreedy indeed performs as well as the baseline with respect to the nearness of the surfaced products to the query in terms of the specified attributes, i.e., the attributes recognized in the query. This observation seems to hold across all range of queries, from specific to general.

Finally, we performed a qualitative evaluation by conducting a user study using the Amazon Mechanical Turk platform[5]. To compare the effectiveness of the consideration set in both cases, we selected two criteria to measure, *viz.*, the best product for each query and the usefulness of the unspecified attribute values in the consideration set in helping the user in her decision to buy the product. As part of the evaluation, we sampled 240 queries and presented the human judge the top 10 results along with a budget (equal to the price of the largest product in the consideration set) and the unspecified attributes associated with the query and asked the judge to select the product she is most likely to buy and chose the unspecified attributes (if any) that aided in her decision. We presented each query to 11 judges.

To get the best product, we simply selected the product with the largest number of votes among the judges. Ties were broken randomly. We then asked the judges to pick the better product among the two best products (one for each algorithm). This gives us a measure of the effectiveness of each consideration set in surfacing the product the user is more likely to buy. In fact, the judges found the best product surfaced by ModifiedGreedy bettered the best product surfaced by the baseline algorithm 58.6% of the queries. With respect to the second criteria, we also presented the judges with the best unspecified attribute (again one for each algorithm) and asked them to choose the better one. Again, the judges concluded something similar. ModifiedGreedy was surfacing more relevant information related to the unspecified attributes compared to the baseline. Specifically, it did better in 67.8% of the queries. Coincidentally, in the first evaluation, the judges did not find the unspecified attributes helpful in around 21.7% of the queries while the corresponding fraction for the consideration sets generated by ModifiedGreedy was 10.6%. This suggests that the

[5]https://www.mturk.com/mturk/welcome

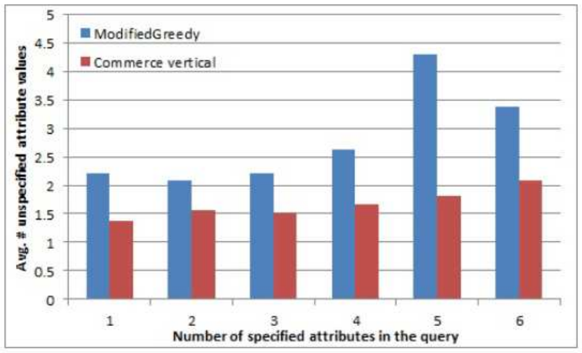

Figure 5: The average number of unspecified attribute values in the consideration set generated by ModifiedGreedy and the shopping vertical of a commercial search engine

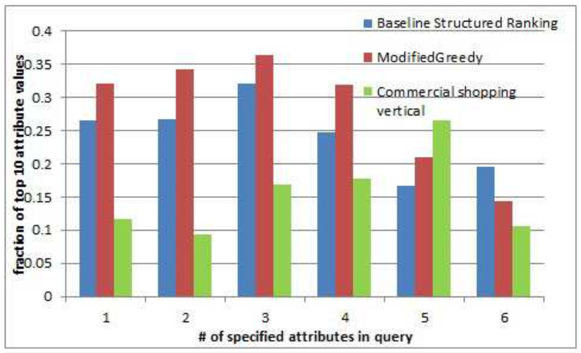

Figure 6: The fraction of the top-10 attribute values covered by the consideration set. This fraction is computed as a average over all the unspecified attributes

consideration set generated by our algorithm not only surfaces relevant results to the user's query but also surfaces more helpful information not directly associated with the specified attributes in the query.

5.5 Comparison with a Commercial Shopping Vertical

We ran the same set of 3200 shopping queries against a shopping vertical of a commercial search engine and scrapped the results of each query. Beyond basic knowledge of the ranking function of the search engine having access to the attributes values of the products, we did not know anything else about the ranking function itself. We simply treated it as a black-box ranking function.

To measure the amount of diversity in the consideration set with respect to the unspecified attributes, we aggregated the attribute values for each product in the consideration set over all queries, again grouping the queries according to the number of specified attributes in them. Figure 5 shows that the average number of unspecified attribute values surfaced by ModifiedGreedy is larger than the corresponding number for the commercial shopping vertical. Interestingly, we did not observe a similar difference when we compared ModifiedGreedy with the baseline structured ranking function.

We also performed a user study to compare the relevance of the consideration sets produced by the shopping vertical and ModifiedGreedy. As in the experiment in the last section, we compared the best products surfaced in the each of the consideration sets. Again, the best product of ModifiedGreedy was chosen 60% of the time over the best result produced by the shopping vertical. The difference is more pronounced in the case of the best unspecified attribute (i.e., the unspecified attribute chosen by the user to be most helpful in their decision to buy the product) in both cases. In fact, 78% of the users preferred the the unspecified attribute surfaced by ModifiedGreedy to be more helpful than the corresponding unspecified attribute in the consideration set in the shopping vertical.

5.6 Coverage of User Intents

In the experiments described in the previous section, we focused on coverage of important unspecified attributes where we measured the importance using the product data in the index. Further, we did not consider any difference in importance of the attribute values themselves. In this section, we describe a set of experiments, where we measured the number of intents covered by the consideration sets. Here, we define an intent to be the set of attribute values associated with a query. For example, when the user issues a query *17 inch laptops*, one of her intents could be to buy a *$700 17 inch dell latitude laptop*. Another could be to buy a *$500 17 inch acer laptop*. Thus, her intent could be defined as a distribution over the attribute values for a given product.

Bed type (Importance)	Bed Type (Importance)
Platform (0.23)	Trundle (0.07)
Bunk (0.19)	Kids (0.04)
Teen (0.17)	Sleigh (0.03)
Toddler (0.11)	Canopy (0.03)
Loft (0.11)	Adjustable (0.01)

Table 2: Relative importance of attribute values for the attribute `bed type` in the category `beds`

In this experiment, we don't explicitly compute this distribution. Instead we approximate it by computing the top attribute values for each of the important attributes the user is likely to consider in her search. Here we assume independence between attributes and hence we treat the likelihood of the user picking one attribute value independent of her choice of values for other attributes. We adopt the definition of attribute value importance described in [10] wherein the importance is measured in terms of the user's likelihood of switching the value during the course of her browsing the web for the product. Thus, an important attribute value in the query is likely to exist in the product she eventually buys. For completeness, we include an example here. Table 2 shows the important attribute values for `bed type` in the category `home furnishings|furniture|beds`.

Thus, for each consideration set generated for a query in our test set, we compute the weighted sum of the attribute values covered by the set using each value's importance as its weight. Again, to be able to compare both structured ranking functions with the shopping vertical, we mapped the attribute values extracted from the product pages to the corresponding attribute values used in the search index of our prototype search engine. This mapping was done using the Jaccard similarity[6] measure and picking the tar-

[6]Given two sets A and B, $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$

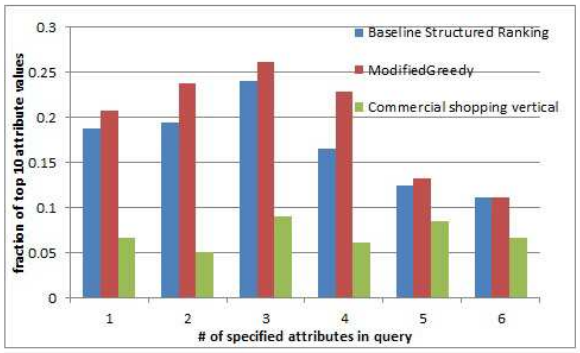

Figure 7: The fraction of the top-10 attribute values covered by the consideration set. This fraction is computed as a weighted average over all the unspecified attributes

get attribute value with the highest similarity score above a threshold (0.5 in our case). In another experiment, we further weigh the contribution of attribute values of an attribute with the importance of the attribute. Figures 7 and 6 show the relative performance of all baseline structured ranking function, ModifiedGreedy, and the shopping vertical for the respective settings of with and without attribute importance.

As both the figures show, ModifiedGreedy surfaces more useful attribute values to the user in the consideration set compared to both the baseline structured ranking function and the shopping vertical in the case when the contribution of attribute values of an attribute are weighted by the importance of the attribute. In the other case where all attributes are treated uniformly, ModifiedGreedy outperforms the other algorithms when the queries are more general.

6. CONCLUSIONS

In this study, we propose an algorithm for generating the consideration set in commerce search that exploits *both* the specified and unspecified attributes and user preferences to maximize user satisfaction. Specifically, we show that this setting admits a natural notion of user satisfaction that requires that the surfaced products be close to the specified attribute values in the query, and diverse with respect to the unspecified attributes which we solve using a max-sum dispersion formulation. This problem is known to be NP-hard, and no constant factor approximation was known previously. We give the first 2-approximation algorithm for this problem along with extensive empirical analysis involving both real-world data and user studies.

The directions for future work include studies on the effect of different demand vectors and budgets on the quality of the final solution.

7. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.

[2] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $(^{1/4})$ approximation for densest -subgraph. In *STOC*, pages 201–210, 2010.

[3] B. E. Birnbaum and K. J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing lps. *Algorithmica*, 55(1):42–59, 2009.

[4] A. Bookstein. Information retrieval: A sequential learning process. *Journal of the American Society for Information Sciences (ASIS)*, 34(5):331–342, 1983.

[5] J. G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[6] C. Chekuri and S. Khanna. A ptas for the multiple knapsack problem. In *SODA*, pages 213–222, 2000.

[7] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, pages 429–436, 2006.

[8] E. Erkut. The discrete p-dispersion problem. *Eur. J. Opnl. Res.*, 46:48–60, 1990.

[9] R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Oper. Res. Lett.*, 21(3):133–137, 1997.

[10] D. Panigrahi and S. Gollapudi. Result enrichment in commerce search using browse trails. In *WSDM*, 2011.

[11] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *International Conference on Machine Learning (ICML)*, 2008. First presented at NIPS07 Workshop on Machine Learning for Web Search.

[12] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[13] D. J. Rosenkrantz, G. K. Tayi, and S. S. Ravi. Facility dispersion problems under capacity and cost constraints. *J. Comb. Optim.*, 4(1):7–33, 2000.

[14] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *SIGMOD*, pages 771–782, 2010.

[15] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, pages 228–236, 2008.

[16] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management Science*, 54(7):1336–1349, 2008.

[17] D. W. Wang and Y.-S. Kuo. A study on two geometric location problems. *Inf. Process. Lett.*, 28(6):281–286, 1988.

[18] C. Zhai and J. D. Lafferty. A risk minimzation framework for information retrieval. *Info. Processing and Management*, 42(1):31–55, 2006.

8. APPENDIX

8.1 Proof of Theorem 4.1

Proof. We use induction on the demand vector $\vec{w}$. For the base case, if exactly one component of $\vec{w}$ is 1 and every other component is zero, there is nothing more to prove. Suppose the claim is true for any demand vector that is strictly dominated by $\vec{w}$. The greedy algorithm selects the edge (x_i, x_j), $x_i \in V_i, x_j \in V_j$ before recursively calling itself on $(V', \vec{w'})$. Let S', O', S, O denote the set of nodes returned by GREEDY$(V', \vec{w'})$, OPT$(V', \vec{w'})$, GREEDY$(V, \vec{w})$, and OPT$(V, \vec{w})$ respectively. We introduce the notations y_i, y_j to denote two nodes depending on the following three cases.

- Case 1 If $x_i, x_j \in O$, let $y_i = x_i$ and $y_j = x_j$.
- Case 2 If $x_i \in O$ and $x_j \notin O$, let $y_i = x_i$ and let y_j be some node in the set $V_j \setminus \{x_i\}$ that belongs to O.
- Case 3 If $x_i \notin O$ and $x_j \notin O$, let y_i (resp. y_j) be some node in the set $V_i \setminus \{x_i, x_j\}$ (resp. $V_j \setminus \{x_i, x_j\}$) that belongs to O.

Note that

$$\begin{aligned}\text{Disp}(O) &= \text{Disp}(O \setminus \{y_i, y_j\}) + d(y_i, y_j) + \\ &\quad \sum_{u \in O \setminus \{y_i, y_j\}} (d(u, y_i) + d(u, y_j)) \qquad (3)\end{aligned}$$

Since $O \setminus \{y_i, y_j\}$ is a solution to the instance $(V', \vec{w'})$, Disp$(O \setminus \{y_i, y_j\}) \leq$ Disp(O'). By Induction hypothesis, Disp$(O') \leq 2$Disp(S'). Combining these two inequalities, we get

$$\text{Disp}(O \setminus \{y_i, y_j\}) \leq 2\text{Disp}(S') \qquad (4)$$

Consider any node $u \in O \setminus \{y_i, y_j\}$. Since both the edges (u, y_i) and (u, y_j) are feasible for $(V, \vec{w})$, and (x_i, x_j) is the feasible edge for $(V, \vec{w})$ with highest weight, we have $d(u, y_i) + d(u, y_j) \leq 2d(x_i, x_j)$. Now consider any node $v \in S'$. By triangle inequality, $d(v, x_i) + d(v, x_j) \geq d(x_i, x_j)$. Thus, $d(u, y_i) + d(u, y_j) \leq 2(d(v, x_i) + d(v, x_j))$. Since both the sets S' and $O \setminus \{x_i, x_j\}$ contain same number of nodes, we have

$$\sum_{u \in O \setminus \{x_i, x_j\}} (d(u, y_i) + d(u, y_j)) \leq 2 \sum_{v \in S'} (d(v, x_i) + d(v, x_j)) \qquad (5)$$

Combining Inequalities 3, 4, 5, and noting that $d(y_i, y_j) \leq d(x_i, x_j)$, we get

$$\begin{aligned}\text{Disp}(O) &\leq 2\text{Disp}(S') + d(x_i, x_j) \\ &\quad +2 \sum_{v \in S'} (d(v, x_i) + d(v, x_j)) \\ &\leq 2\text{Disp}(S)\end{aligned}$$

□

8.2 Proof of Lemma 4.3

PROOF. Let f denote the number of different L-tuples with $\sum_{l=1}^{L} t_l = \log n / \epsilon^3 = L/\epsilon$. Clearly,

$$\begin{aligned} f &= \binom{L/\epsilon + L - 1}{L - 1} \leq \binom{(1 + 1/\epsilon)L}{L - 1} \\ &\leq \frac{((1 + 1/\epsilon)L)^{L-1}}{(L-1)!} = O\left(((1 + 1/\epsilon)e)^{L-1}\right)\end{aligned}$$

The last equality follows by applying Stirling's formula on $(L-1)!$. Noting that $(1 + 1/\epsilon) \leq e^{1/\epsilon}$ and plugging in the value of L, we get

$$f = O\left(e^{(1+1/\epsilon)(\log n/\epsilon^2 - 1)}\right) = O\left(n^{O(1/\epsilon^3)}\right)$$

□

8.3 Proof of Lemma 4.2

PROOF. Since the set S contains exactly w'_l nodes from bucket V_l, $\tilde{c}(S \cap V_l) = w'_l \tilde{c}_l \leq (t_l + 1)(\epsilon^3 B / \log n)$, whenever $l \geq 1$. Therefore,

$$\begin{aligned}\tilde{c}(S \setminus V_0) &= \sum_{l=1}^{L} \tilde{c}(S \cap V_l) \leq \sum_{l=1}^{L} (t_l + 1)(\epsilon^3 B / \log n) \\ &= \left(L + \sum_{l=1}^{L} t_l\right) \epsilon^3 B / \log n = (1 + \epsilon)B\end{aligned}$$

Next, consider any bucket V_l, $l \in \{1, \ldots, L\}$, and note that for any node $v \in V_l$, $c(v) \leq (1 + \epsilon)\tilde{c}(v)$. Thus,

$$c(S \setminus V_0) \leq (1 + \epsilon)\tilde{c}(S \setminus V_0) \leq (1 + \epsilon)^2 B \leq (1 + 3\epsilon)B$$

Finally, each node in V_0 has a cost at most $\epsilon B/n$, and there can be at most n nodes in V_0. Thus, the total cost of the nodes in V_0 is at most ϵB. Combining this with the fact that $c(S \setminus V_0) \leq (1 + 3\epsilon)B$, we get

$$c(S) = c(S \setminus V_0) + c(V_0) \leq (1 + 3\epsilon)B + \epsilon B = (1 + 4\epsilon)B$$

□