Composite Retrieval of Diverse and Complementary Bundles

Sihem Amer-Yahia, Francesco Bonchi, Carlos Castillo,
Esteban Feuerstein, Isabel Mendez-Diaz, and Paula Zabala

Abstract—Users are often faced with the problem of finding complementary items that together achieve a single common goal (e.g., a starter kit for a novice astronomer, a collection of question/answers related to low-carb nutrition, a set of places to visit on holidays).

In this article, we argue that for some application scenarios returning *item bundles* is more appropriate than ranked lists. Thus we define *composite retrieval* as the problem of finding k bundles of complementary items. Beyond complementarity of items, the bundles must be valid w.r.t. a given budget, and the answer set of k bundles must exhibit diversity.

We formally define the problem and characterize its complexity. We prove that the problem in its general form is $\mathbf{NP}$-hard and that also the special cases in which each bundle is formed by only one item, or only one bundle is sought, are hard. Our characterization however suggests us how to adopt a two-phase approach (*Produce-and-Choose*, or PAC) in which we first produce many valid bundles, and then we choose k among them. For the first phase we devise two ad-hoc clustering algorithms, while for the second phase we adapt heuristics with approximation guarantees.

We also devise another approach which is based on first finding a k-clustering and then selecting a valid bundle from each of the produced clusters (*Cluster-and-Pick*, or CAP).

We compare experimentally the proposed methods on a large real-world database of user-generated restaurant reviews from Yahoo! Local, exploring their performance under a variety of settings. Our experiments show that when diversity is highly important, CAP is the best option, while when diversity is less important, a PAC approach constructing bundles around randomly chosen pivots, is better.

Index Terms—Composite retrieval, Maximum Edge Subgraph, complementarity, diversity.

1 Introduction

Online search has become a daily activity and a source of a variety of valuable information, from the finest granularity such as finding the address of a specific restaurant, to more complex tasks like looking for accessories compatible with an iPhone or planning a trip. The latter typically involves running multiple search queries to gather information about different places, reading online reviews to find out about hotels, and checking geographic proximity of places to visit. We refer to this information seeking activity as composite retrieval and propose to organize results into item bundles that together constitute an improved exploratory experience over ranked lists.

As a first step towards composite retrieval definition, we need to formalize intuitive desirable properties of item bundles. We distinguish between properties of each bundle in the answer and properties of the answer as a whole. Given the wide applicability of composite retrieval, we propose to first explore these properties in a few application scenarios, then we motivate the need for a framework to study composite retrieval complexity and develop efficient algorithms.

1.1 Composite Retrieval Scenarios

Consider the case of a user selecting the restaurants to try during a visit to a new city. In this scenario, the user has a limited budget which might be either financial, or simply the number of nights spent in the city. Moreover the user prefers suggested restaurants to serve different cuisines. In this setting the validity of a bundle of restaurants is given by the budget constraint and the complementarity of the restaurants in the bundle w.r.t. the cuisine they serve. Other restaurant attributes could be used for defining valid bundles. For example, instead of cuisines, different dress codes (e.g., casual, business casual, formal) could distinguish restaurants in a single bundle.

Moreover, in order to provide meaningful bundles, restaurants forming each bundle must be compatible, e.g., close geographically, or liked by similar reviewers. The degree of compatibility of the items forming a bundle defines the quality of the bundle. Intuitively, in the case geographic distance is used, the closer restaurants are from each other, the higher the quality of the bundle they belong to. Similarly, when common reviewers are used as the quality of a bundle, the higher the overlap in similar reviewers between restaurants in the same bundle, the higher

- *S. Amer-Yahia is with CNRS - LIG, Grenoble, France. E-mail:* Sihem.Amer-Yahia@imag.fr.
- *F. Bonchi is with Yahoo! Research Barcelona, Spain. E-mail:* bonchi@yahoo-inc.com.
- *C. Castillo is with Qatar Computing Research Institute, Doha. E-mail:* chato@acm.org.
- *E. Feuerstein, I. Mendez-Diaz, and P. Zabala are with Universidad de Buenos Aires, Argentina. E-mail:* {efeuerst,pzabala,imendez}@dc.uba.ar.

the quality of that bundle. Finally, bundles forming an answer set can be generated to cover different various geographic areas, or different reviewers segments, thereby producing an answer set of bundles with diversity.

In general, browsing and searching social media, e.g., a photo-sharing site, or a question-answering portal, might benefit from composite retrieval. Given a query or while browsing a category, diverse bundles of complementary resources might provide a better experience for the user. Complementarity of resources forming a bundle can be expressed e.g., using topical sub-categories in order to ensure that no two resources in the same bundle are on the same sub-category. Budget can simply be the number of resources forming a bundle. Compatibility of two resources can be expressed by their similarity, e.g., the Jaccard coefficient of their tags, or the overlap of the users that expressed interest for the resources – thereby bundling together resources from the same user community. In this last case, answer set diversity is achieved by finding multiple bundles each of which representing the opinion of a different user population.

Other applications include online shopping where products forming the same bundle have different types (e.g., a telescope, a carrying bag, a lens kit, an astronomy book), their total price is within budget, and they are compatible according to their manufacturers. Answer diversity in this case could be enforced by including offerings from different manufacturers.

City tours can also be represented as a bundle with one item per attraction (e.g., museum, monument, market, park), budget is the total time required to visit all items in a bundle, and compatibility is an inverse of geographic distance. Therefore, answer diversity would reflect attractions in different areas in a city.

Finally, team building for problem solving is another application scenarios for composite retrieval. Here we want to find experts on a topic (e.g., safety standards experts) to form a team. The team needs to contain complementary members with different roles – e.g., a manager, an engineer, a lawyer – with an upper limit on the team size. Compatibility should reflect experts who worked together in the past (e.g., those co-mentioned in a document, are colleagues in the same department, have collaborated on problem solving, etc). This use case highlights the subtle difference between the need for each bundle to contain members with different skills on one hand, and the requirement that those members have worked together.

1.2 Article Contributions and Roadmap

We define and study the problem of retrieving k diverse bundles of complementary items under a per-bundle budget.

The above examples show the wide applicability of composite retrieval beyond traditional information retrieval and the variety of complementarity and budget constraints that could be used. It is important to note that bundles may be built using the most relevant items to a query thereby making traditional relevance orthogonal to bundle construction. That allows us to define the quality of a bundle, i.e., its score, as a function of pair-wise similarities between its items. As in traditional retrieval, we aim to retrieve highly scoring and also *diverse* bundles. The quality of a collection of k bundles is given by a weighted combination of the quality of each bundle and inter-bundle diversity.

In Section 2 we formally define the problem and we provide two different proofs of **NP**-hardness, highlighting two different aspects of the complexity of the problem. Both proofs reduce the well known **NP**-complete MAXIMUM EDGE SUBGRAPH problem to ours.

Given that our problem is **NP**-hard, we turn our attention to approximation algorithms. Following the hint of our **NP**-hardness proofs, in Section 3 we describe a two-phase approach (*Produce-and-Choose*, or PAC) in which we first produce many valid bundles, and then we choose k among them. For the choosing phase we show an approximation-preserving reduction from MAXIMUM EDGE SUBGRAPH which enables us to adopt heuristics that have been developed in the literature for that problem.

For the task of producing good bundles we observe the similarity between the objective function of our problem, and that of clustering. Following this observation, in Section 4 we devise two ad-hoc clustering algorithms: the first one based on constrained hierarchical clustering, and the second one inspired by *k-nn* clustering.

In Section 5 we introduce a different approach which is based on first finding a k-clustering and then selecting a valid bundle from each of the produced clusters and in Section 6 we present an integer programming formulation of our problem, which may be used to obtain the exact solution.

In Section 7 we compare experimentally the proposed methods on a large database of user-generated restaurant reviews from Yahoo! Local. We assess both quality and efficiency. We show that our heuristics produce good results according to our optimization objective, comparing them with the results of the LP implementation within a Branch-and-Cut framework. Our main finding is that the performance of these methods depends basically on a parameter controlling the trade-off between the average score of the bundles and the diversity of the set of bundles. When diversity is highly important, we obtained the best performance using algorithms based on creating a global clustering of the items first, and then choosing bundles that respect those clustering boundaries. When diversity is less important, we show that "local" methods that

construct good bundles around randomly chosen pivots produce better results.

Finally in Section 8 we summarize related work, while in Section 9 we conclude.

2 Composite Retrieval: Definition and Complexity

In this section we develop the formal data model and problem statement of composite retrieval, then we prove its hardness.

2.1 Data Model

We are given a set of items $\mathcal{I}$. Each item in $\mathcal{I}$ is uniquely identified and has a set of attributes. We assume a *similarity* value $s(u, v)$ for each pair of items $(u, v) \in \mathcal{I} \times \mathcal{I}$. The similarity values $s(u, v)$ may be provided explicitly in the input, or computed implicitly from the representation of the items. For instance, if items are documents, $s(u, v)$ may be defined as the cosine between the vectors that represent the documents u and v; if items are restaurants, $s(u, v)$ may be defined as the fraction of reviewers that like both u and v; etc. Hereinafter, we simply consider the values $s(u, v)$ as input to our problem and we do not make any assumption on how to obtain those values.

Sometimes it is useful to think of our input as a complete, undirected and weighted graph $G = (\mathcal{I}, E, s)$, where each edge (u, v) has a weight $s(u, v)$. For sake of clarity of presentation, we consider that the similarity function s takes values in the interval $[0, 1]$ (this is always achievable by normalization).

For convenience we also refer to the *distance function* between item pairs, defined as $d(u, v) = 1 - s(u, v)$, that also takes values in the interval $[0, 1]$ (although by definition we cannot warrant that $d(u, v)$ defined this way is really a metric, i.e. it obeys triangular inequality).

2.2 Problem Statement

Our goal is to retrieve a set of bundles $\mathcal{S} = \{S_1, \ldots, S_k\}$, where a bundle $S_i \in 2^{\mathcal{I}}$ is a set of items that satisfy constraints of *complementarity* and *budget* as expressed in the following definition.

Definition 1 (Valid Bundle)**.** *Given a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, a bundle $S \in 2^{\mathcal{I}}$ is said to be* valid *iff it satisfies the following two constraints.*

- ***Complementarity:*** *given a property α of the items (e.g., an attribute), no two items in $S_i \in \mathcal{S}$ exhibit the same value for that property: i.e., $\forall u, v \in S_i, u.\alpha \neq v.\alpha$.*
- ***Budget:*** *given a set-valued non-negative and monotone function $f : 2^V \rightarrow \mathbb{R}^+$, and given a budget threshold β, we require that $\forall S_i \in \mathcal{S}, f(S_i) \leq \beta$. Typical examples of budget are simply the number of items forming a bundle or an upper-bound on the sum of the costs of items forming the bundle, given a cost attribute.*

For example, in the case of restaurants, the property α could be cuisine type (e.g., Chinese, American) or geographic area (e.g., East Village, Greenwich Village, Lower East side) or a combination thereof. The budget function f could be the sum of the average price of a meal at all restaurants forming the bundle. In a travel application, items are specific attractions, α their type (e.g., Museum, Park), and f can be the time required to visit each place. We are now ready to define the problem of composite retrieval.

Problem 1 (Composite Retrieval)**.** *Given a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, a pair-wise similarity function $s(u, v)$ for each $(u, v) \in \mathcal{I} \times \mathcal{I}$, a complementarity attribute α, a budget function $f : 2^{\mathcal{I}} \rightarrow \mathbb{R}^+$, a budget threshold β, and an integer k, find a set $\mathcal{S} = \{S_1, \ldots, S_k\}$ of valid bundles that maximizes:*

$$\sum_{1 \leq i \leq k} \sum_{u,v \in S_i} \gamma \; s(u, v) + \sum_{1 \leq i < j \leq k} (1-\gamma)(1 - \max_{u \in S_i, v \in S_j} s(u, v))$$

where γ is a user-defined scaling parameter.

The objective function resembles a typical clustering objective, where the total quality of the clustering is expressed as a weighted combination of the quality of single clusters (which in turn is defined as their intra-cluster cohesion) and inter-cluster separation. The latter can be defined as the minimum distance between an item in one bundle and an item in another one. Intra-cluster cohesion reflects cluster quality as a function of the similarity or cohesion between items forming the cluster. Inter-cluster separation reflects answer diversity. Unlike standard clustering, our problem does not seek a total partitioning of items, instead it aims at finding k good groups, that might potentially be small as they are bounded by the budget constraint. Therefore, some items in $\mathcal{I}$ might not belong to any bundle or belong to more than one. It is worth noting that our problem definition, by summing over all elements in a bundle, favors larger bundles: as large as possible given the budget constraint.

The complementarity requirement, requiring no more than one single element of a given kind to belong to a bundle, can be seen as a set of many *cannot-link* constraints typical to constrained clustering [5]. In particular, given the complementarity property α, each item *cannot-link* with all the other items in $\mathcal{I}$ that have the same value for α. These observations will be used later in Section 4 to devise algorithms for composite retrieval.

Finally, it is important to note that bundles are built using the most relevant items to a query thereby making traditional relevance orthogonal to bundle construction. That allows us to define the quality of a bundle, i.e., its score, as a function of pair-wise compatibilities between its items. similarly to

traditional retrieval, we aim to retrieve highly scoring and also *diverse* bundles. The quality of a collection of k bundles is given by a weighted combination of the quality of each bundle and inter-bundle diversity.

2.3 Problem Complexity

Not surprisingly our problem is hard. We provide two different proofs of **NP**-hardness, where the first one highlights the complexity of the first argument of the objective function, and the second one the complexity of the second argument. Both proofs reduce the well known **NP**-complete problem MAXIMUM EDGE SUBGRAPH (also known as *Dense k-subgraph*, or *Heavy subgraph*) to our problem. The MAXIMUM EDGE SUBGRAPH problem requires to find a set of k nodes, such that the induced subgraph has maximum sum of edge weights.[1]

Theorem 1. *Composite retrieval is* **NP***-hard.*

Proof: We prove the hardness by reducing MAXIMUM EDGE SUBGRAPH to our problem. Given an instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH, consisting of a graph $G = (V, E)$, a weight function $w : E \to \mathbb{N}$, and an integer $k \leq |V|$, we create an instance $\mathcal{J}$ of composite retrieval as follows. For each node $v \in V$ we create an item $v \in \mathcal{I}$. Moreover we give the same value of complementarity attribute α to all items $v \in \mathcal{I}$, so that no bundle of size larger than one qualifies. Similarly we set the budget function to be the cardinality of a bundle and the budget threshold to be 1. The required number of bundles is k. Finally, for each pair of items $u, v \in \mathcal{I}$, we set $s(u, v) = 1 - w(u, v)$ if $(u, v) \in E$, and $s(u, v) = 1$ otherwise. The reduction can clearly be carried out in polynomial time. Under that reduction, the left summation in the objective function of composite retrieval is null, as each bundle is formed only by one item. Therefore, the objective becomes to maximize

$$(1 - \gamma) \sum_{i \leq j \leq k} (1 - \max_{u \in S_i, v \in S_j} s(u, v))$$

The term $(1 - \gamma)$ can be removed. Moreover we can identify each bundle with the unique element forming it. Thus, we can rewrite the objective function as

$$\sum_{i \leq j \leq k} w(S_i, S_j).$$

Therefore a set $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq V$ is a solution in instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH *iff* it is a solution in instance $\mathcal{J}$ of composite retrieval. This was to be proven. □

In order to further characterize the complexity of the problem, we provide another proof of **NP**-hardness. While the first proof exploits a reduction where each bundle can contain at most a single item, the next proof uses a reduction where a unique bundle is sought.

Proof: We again prove hardness by reducing MAXIMUM EDGE SUBGRAPH to our problem, but using a different reduction. Given an instance $\mathcal{L}$ of MAXIMUM EDGE SUBGRAPH, consisting of a graph $G = (V, E)$, a weight function $w : E \to \mathbb{N}$, and an integer $k \leq |V|$, we create an instance $\mathcal{J}$ of composite retrieval as follows. For each node $v \in V$ we create an item $v \in \mathcal{I}$. The parameter k of our problem is set to 1, i.e, we ask for a single bundle. We now assign a *different* value of the complementarity attribute α to all items $v \in \mathcal{I}$, so that any two items can be part of the same bundle. Similarly to the previous proof, we set budget function to be the cardinality of a bundle, but now the budget threshold is set to be k, i.e., we ask for a single bundle of size k. Finally, for each pair of items $u, v \in \mathcal{I}$, we set $s(u, v) = w(u, v)$ if $(u, v) \in E$. Clearly, this reduction can be carried out in polynomial time.

With this reduction, the *right* summation in the objective function of composite retrieval is null, as there is just one bundle in the answer. The objective function to maximize is

$$\sum_{1 \leq i \leq 1} \sum_{u, v \in S_i} \gamma s(u, v)$$

or simply

$$\sum_{u, v \in S_1} \gamma s(u, v) = \sum_{u, v \in S_1} \gamma w(u, v),$$

which is exactly the same as maximizing the objective function of MAXIMUM EDGE SUBGRAPH. This was to be proven. □

Given that our problem is **NP**-hard, we turn our attention to approximation algorithms. Going in that direction, it might be interesting to see what is known about the MAXIMUM EDGE SUBGRAPH that we used in our reduction. By the nature of our reductions (in both of them the number of nodes and edges remain equal, as well as the edges' weights), we conclude that composite retrieval is not more approximable than MAXIMUM EDGE SUBGRAPH.

The problem cannot be approximated within constant factors (i.e. it does not have a PTAS) unless **NP** has subexponential time algorithms. Moreover, it is believed that it is hard to approximate within a polylogarithmic factor. In the following, k denotes the parameter of the MAXIMUM EDGE SUBGRAPH problem. Asahiro *et al.* [4] propose a simple heuristic that consists in repeatedly removing a vertex with the minimum weighted-degree in the currently remaining graph, until exactly k vertices are left. The approximation ratio is $O(n/k)$. Feige *et al.* [12] and very recently, Bhaskara *et al.* [6] give more complex heuristics improving the approximation ratios up to $O(n^{1/4} + \epsilon)$. In the significant particular case in

1. http://www.nada.kth.se/~viggo/wwwcompendium/node46.html

which the weights obey the triangular inequality, a 2-approximation exists [16].

Note that, even if the max-dense-subgraph (i.e. where no bound on the size of the subgraph is given) is polinomialy solvable [14], our double NP-hardness proof shows that even if we removed the budget of complementarity constraints the problem would still NP-hard, as the complexity would be attached to the sub-problem of finding the most diverse set of k bundles.

Also note that, as it may be seen from the above **NP**-hardness proofs, even the particular problem of finding just one (the best) bundle is hard, whenever there are budget constraints. However, we mention some particular cases that could appear in our context that can be solved in polynomial time. These are the cases in which the bundle size is bounded by some constant. That happens, for example, whenever the number of different values of the complementarity attribute α is bounded by a constant, so in $O(n^{|\alpha|})$ time we can solve the problem by considering all possible item combinations that fulfill the coverage constraints. A similar approach can be taken when the cardinality of the bundles is limited by a constant upper-bound. In all these cases a brute force approach could be feasible. The same holds whenever k, the number of bundles requested, is small (say, 2 or 3).

3 Produce Bundles and Choose

In the previous section, we showed the complexity of our problem by means of two different reductions from Maximum Edge Subgraph. The first of the two reductions suggests a possible approach for composite retrieval. In fact, if we generate candidate bundles and we consider each candidate bundle as a node of a *bundle-graph*, where inter-bundle distances are the edge weights, then we are again in front of the Maximum Edge Subgraph problem.

This suggests that composite retrieval can be solved by generating a set of candidate bundles and then selecting the best possible subset. We call this approach *Produce-and-Choose* (Algorithm 1).

Algorithm 1 `Produce − and − Choose`

Input: $\mathcal{I}$, α, f, β, k, γ as in Problem 1
Output: A set $\mathcal{S}$ of k valid bundles.

1: $Cand \leftarrow$ `produce_bundles`$(\mathcal{I}, \alpha, f, \beta)$;
2: $G \leftarrow$ `build_bundle_graph`$(Cand)$;
3: **return** `ChooseBundles`(k, γ, G);

If the set of candidates $Cand$ covers all possible bundles we can obtain the exact solution, but we know that this cannot be done in polynomial time. Thus we tackle our problem by generating proper subsets of candidate bundles and trying to smartly select among them.

In the rest of this section we focus on the Choose phase (algorithm `ChooseBundles`) and show that we can exploit the known results for Maximum Edge Subgraph, discussed in the previous section, while preserving approximation guarantees. For this, we need to define an intermediate problem, in which both edges and nodes are weighted, and where we need to find a maximum weight (considering edges AND nodes) k-node induced subgraph.

Formally, the Maximum Edge-Node Subgraph problem is the following: the input consists in a graph $G = (V, E)$, a weight function $\psi : E \to \mathbb{N}$, a weight function $\omega : V \to \mathbb{N}$, an integer $k \leq |V|$ and a real value $\gamma \in [0, 1]$. The output is a set $V' \subseteq V$ such that $|V'| = k$ that maximizes the total weight of the edges and the nodes in the subgraph induced by V', denoted $G' = (V', E')$, weighted by the parameter γ, as follows:

$$\gamma \sum_{u \in V'} \omega(u) + (1 - \gamma) \sum_{(u,v) \in E'} \psi(u, v).$$

It is straightforward to see the direct relation between our PAC approach to composite retrieval (i.e., first produce bundles and then choose those maximizing the objective function), and the Maximum Edge-Node Subgraph problem defined above. In particular bundles are nodes, quality of a bundle (i.e., cohesion) is the node's weight, and inter-bundle distances are edge weights. We next show an approximation-preserving reduction from instances of Maximum Edge-Node Subgraph to Maximum Edge Subgraph that will allow us to apply any of the approximated algorithms for the Maximum Edge Subgraph problem and maintain the same approximation guarantee.

Consider an instance of Maximum Edge-Node Subgraph: $G = (V, E), \psi(u, v), \omega(u), k$ and γ. We transform it to an instance of the Maximum Edge Subgraph where $G = (V, E)$ and k are the same, and the edges weight function is:

$$w(u, v) = \frac{\gamma}{2(k - 1)}(\omega(u) + \omega(v)) + (1 - \gamma)\psi(u, v).$$

We can see that a solution V' (and its induced subgraph $G' = (V', E')$) maximizing the instance of Maximum Edge Subgraph is also maximizing the corresponding instance of Maximum Edge-Node Subgraph:

$$\sum_{(u,v)\in E'} w(u,v) =$$
$$\frac{\gamma}{2(k-1)} \sum_{(u,v)\in E'} (\omega(u)+\omega(v)) + (1-\gamma) \sum_{(u,v)\in E'} \psi(u,v) =$$
$$\frac{\gamma}{2(k-1)}((k-1) \sum_{u\in V'} \omega(u) + (k-1) \sum_{v\in V'} \omega(v)) +$$
$$+ (1-\gamma) \sum_{(u,v)\in E'} \psi(u,v) =$$
$$\frac{\gamma}{2(k-1)} 2(k-1) \sum_{u\in V'} \omega(u) + (1-\gamma) \sum_{(u,v)\in E'} \psi(u,v) =$$
$$\gamma \sum_{u\in V'} \omega(u) + (1-\gamma) \sum_{(u,v)\in E'} \psi(u,v)$$

The best approximation guarantee for MAXIMUM EDGE SUBGRAPH is given for the case that edge weights, i.e., inter-bundle distance, is a metric [16]. Unfortunately, in our problem where inter-bundle distance is defined as $(1-\max_{u\in S_i, v\in S_j} s(u,v))$, or equivalently as the minimum distance $d(u,v) = 1 - s(u,v)$, triangular inequality does not hold. This can be seen through the following counter-example.

Consider the case in which the items $\mathcal{I}$ represent restaurants and $s(u,v)$ is defined as the Jaccard coefficient of reviewers that wrote a positive review on u and v.

Consider now three bundles A, B and C. It is possible that a restaurant $b \in B$ has exactly the same reviewers of a restaurant $a \in A$, and another restaurant $b' \in B$ has exactly the same reviewers of a restaurant $c \in C$, while there does not exist a restaurant in A having exactly the same reviewers of a restaurant in C. In this setting the distance between bundles A and C would be larger than zero, while distances between A and B, and between B and C would be both null, i.e, triangular inequality does not hold. Thus, we cannot borrow the 2-approximation of [16], and will need to refer to some of the heuristics that have been proposed in [4], [12], [6]. The first of them may be applied to the weighted version of the problem, while the other two were developed for the unweighted version and can be extended to the weighted version incurring an additional $O(\log n)$ factor in the approximation ratio [12].

This is done by first scaling edge weights to n^2 and then applying a procedure that consists in dividing the edges into $2\log n$ buckets according to their weights, then solving separately the $2\log n$ instances of the unweighted problem obtained by including only the edges in each bucket, and finally selecting the best of the $2\log n$ solutions obtained.

Among the three heuristics just mentioned, we choose to implement the one in [4], due to its simplicity and applicability to the weighted case.

Algorithm 2 provides the pseudocode for the Choose phase of our method. It receives in input a set of valid bundles according to Definition 1. This set of bundles is represented as a complete weighted graph, where nodes are bundles. Nodes and edges are weighted with intra-bundle cohesion and inter-bundle separation respectively.

Algorithm 2 `ChooseBundles`

Input: k, γ, and the bundle weighted graph $G = (V,E)$ where $\forall S \in V : \omega(S) = \sum_{u,v\in S} s(u,v)$, and $\forall (S_i,S_j) \in E : \psi(S_i,S_j) = 1 - \max_{u\in S_i, v\in S_j} s(u,v)$.
Output: A set $\mathcal{S}$ of k valid bundles.
1: Define $w(u,v) = \frac{\gamma}{2(k-1)}(\omega(u)+\omega(v)) + (1-\gamma)\psi(u,v)$.
2: $\mathcal{S} \leftarrow V$
3: **while** $\mathcal{S} > k$ **do**
4: $u \leftarrow \text{argmin}_{[u\in\mathcal{S}]} \sum_{v\in\mathcal{S}} w(u,v)$;
5: Remove u from $\mathcal{S}$
6: **return** $\mathcal{S}$

Given this graph and the parameters γ and k, this is an instance of MAXIMUM EDGE-NODE SUBGRAPH that can be reduced to an instance of MAXIMUM EDGE SUBGRAPH by setting the edges' weight as in line 1. Then we can apply the heuristic of [4] greedily removing at each iteration the vertex with minimum weighted-degree (lines 4-5) in the currently remaining graph, until exactly k vertices are left (line 3).

4 CREATING GOOD BUNDLES

In this section we devise methods for the `produce_bundles` phase of our method. As we discussed in Section 2, our objective function resembles a typical clustering objective function, where the total quality of the clustering is expressed as a weighted combination of the quality of single clusters and the inter-cluster separation. Additionally, complementarity within each bundle can be enforced by means of a set of *cannot-link* constraints.

Following these observation we propose two alternative algorithms for the task of producing a set of good valid bundles $Cand$, with given cardinality $|Cand| = c \gg k$. The first method is based on constrained hierarchical clustering, while the second takes inspiration from *k-nn* clustering.

4.1 Constrained Clustering

The first option we consider is to perform a constrained hierarchical agglomerative clustering [10] (`C-HAC`) using $d(u,v) = 1 - s(u,v)$ as distance among items u and v.

Similarly to a standard hierarchical agglomerative clustering algorithm, `C-HAC` starts with a number of clusters equal to the number of input elements, and then iteratively merges the closest clusters until a stop condition (e.g. number of clusters) is met.

The idea of `C-HAC` is to never merge two clusters S_1, S_2 if the resulting bundle $S_1 \cup S_2$ is not valid,

Algorithm 3 C−HAC

Input: $\mathcal{I}$, α, f, β, and number of bundles c
Output: a set of c valid candidate bundles

1: $Cand \leftarrow \cup_{i \in \mathcal{I}} \{i\}$
2: **while** $|Cand| > c$ **do**
3: $bestscore \leftarrow -\infty$
4: $bestcandidate \leftarrow \emptyset$
5: **for** (each $S_i \in Cand$) **do**
6: **for** (each $S_j \in Cand; S_j \neq S_i$) **do**
7: **if** (validMerge$(S_i, S_j, \alpha, f, \beta)$) **then** // $S_i \cup S_j$ form a valid bundle
8: **if** ($score(S_i \cup S_j) > bestscore$) **then**
9: $bestscore \leftarrow score(S_i \cup S_j)$
10: $bestcandidate \leftarrow \{S_i, S_j\}$
11: **if** ($bestcandidate = \emptyset$) **then** // no more merging is possible
12: **break**
13: $Cand \leftarrow Cand - S \quad \forall S \in bestcandidate$
14: $Cand \leftarrow clusters \cup bestcandidate$
15: **return** $Cand$

i.e., if it does not honor budget or complementarity constraints. This is described in Algorithm 3.

The verification of whether two bundles can be merged into a valid bundle (validMerge in line 7) can be made faster by observing that if $S_1 \cup S_2$ is not a valid bundle, then $(S_1 \cup T) \cup (S_2 \cup V)$ is not a valid bundle either for any $T, V \subseteq \mathcal{I}$. This means that a graph of "incompatibility" (connecting clusters that cannot be merged) can be kept, and after every merge in the algorithm, this graph can be updated simply by lumping the nodes corresponding to the merged clusters and keeping all their non-redundant edges.

Recall that in this algorithm each cluster is always a valid bundle, so when it reaches the stopping condition $|Cand| = c$ we can immediately return $Cand$ without the need for any further check.

Other clustering methods can be adapted to incorporate budget and complementarity constraints during the process of creating clusters, e.g. [21] describes a constrained version of k-means that follows similar principles: elements are associated to their closer centroid as long as they can form a valid bundle with the other elements already associated to that centroid.

In this article we adopt C−HAC as a representative of this family of algorithms, and leave the study of other constrained clustering approaches to future work. We could also try to find exact solutions, using mathematical programming tools, as proposed, for instance, in [19], [1], but in this article we focus on efficient heuristic algorithms.

4.2 Bundles One-by-One

The second method for producing a good set of candidate bundles is inspired by *k-nn* clustering. At each step an item is chosen as pivot, and a valid bundle is built around that pivot. If the bundle generated has a good internal cohesion it is kept, otherwise it is discarded. We call this method BOBO (Bundles One-by-One) and we report its pseudo-code in Algorithm 4.

BOBO starts with an empty set of candidate bundles (line 1), and considers each item as a possible pivot (line 2). At each iteration an item is picked from the set $Pivots$. Line 4 determines the way in which pivots are chosen. We can pick pivots uniformly at random among the candidates pivots, or with a certain bias to particular candidate pivots, or do a more sophisticated approach such as picking at each step the furthest pivot to the previous one. Once a pivot is selected we build a bundle S around it. This is done by the routine pick_bundle described in Algorithm 5. The routine greedily keeps picking the closest element to the pivot ω (line 3), as far as the complementarity constraint (line 4) and the budget constraint (line 5) are satisfied.

Algorithm 4 BOBO

Input: $\mathcal{I}$, α, f, β, minimum bundle score μ, and number of bundles c
Output: a set of c valid candidate bundles

1: $Cand \leftarrow \emptyset$
2: $Pivots \leftarrow \mathcal{I}$
3: **while** $Pivots \neq \emptyset$ **and** $|Cand| < c$ **do**
4: $\omega \leftarrow$ pick an element from $Pivots$
5: $\mathcal{I} \leftarrow \mathcal{I} \setminus \{\omega\}$
6: $S \leftarrow$ pickBundle$(\omega, \mathcal{I}, \alpha, f, \beta)$
7: **if** $score(S) \geq \mu$ **then**
8: $\mathcal{I} \leftarrow \mathcal{I} \setminus S$
9: $Pivots \leftarrow Pivots \setminus S$
10: $Cand \leftarrow Cand \cup \{S\}$
11: **else**
12: $Pivots \leftarrow Pivots \setminus \{\omega\}$
13: **return** $Cand$

Other greedy objectives for pick_bundle can be used. For instance, instead of choosing the element i that maximizes $s(i, \omega)$ one can pick the element that maximizes $s(i, j)$ for $j \in s$, or the one that maximizes $\sum_{j \in s} s(i, j)$. We experimented with these more complex objectives and the results were not substantially better than with the simpler one we report, while the running times were definitively worse.

Let us go back to BOBO's main loop. Once a candidate bundle is created we check (line 7) whether its internal cohesion (called $score$ in the pseudocode) is larger than an input given threshold μ. More precisely $score(S) = \sum_{u,v \in S} s(u, v)$. If also this check is passed then the bundle enters in $Cand$ and its elements are removed from $\mathcal{I}$ and $Pivots$ so that they are no longer used. Instead if the bundle S has a score lower than μ then it is discarded. In both cases the pivot ω is removed from $Pivots$ so that it is no longer considered.

Algorithm 5 `pick_bundle`

Input: pivot ω, set of items $\mathcal{I}$, parameters α, f, β
1: $s = \{\omega\}$; $covered = \{\omega.C\}$; $active \leftarrow \mathcal{I} \setminus \{\omega\}$; $finish$ =false
2: **while** not $finish$ **do**
3: $i \leftarrow \text{argmax}_{[i \in active]}\ s(i, \omega)$
4: **if** $i.\alpha \notin covered$ **then**
5: **if** $f(s \cup \{i\} \leq \beta)$ **then**
6: $s \leftarrow s + i$; $covered \leftarrow covered \cup \{i.\alpha\}$
7: **else**
8: $finish =$ true
9: $active \leftarrow active \setminus \{i\}$
10: **return** s

It should be noted that the algorithm might end without having produced c valid bundles due to a too restrictive μ. In this case we might keep the $cand$ set we have produced, if its size is $\gg k$, or we can re-run `BOBO` with a smaller μ.

5 CLUSTER-AND-PICK

We next introduce a totally different method, suggested by the observation (that we already made in Section 2) that the objective function of composite retrieval has various similarities with a clustering problem.

Therefore, we define a new approach based on clustering. In a first phase items are clustered based on their compatibilities, to form k clusters with good internal cohesion and external separation. This can be done by means of any standard clustering algorithm. Then there is a second phase where we pick a good bundle from each cluster (subroutine `BestBundle`, Algorithm 7, which in turn calls subroutine `PickBundle`, Algorithm 5). We refer to this method as `CAP` (Cluster-And-Pick) and provide its pseudo-algorithm below.

Algorithm 6 `CAP`: Cluster-And-Pick

Input: $\mathcal{I}$, α, f, β,k;
Output: A set $\mathcal{S}$ of k valid bundles.
1: $clusters \leftarrow clustering(I, k)$
2: $\mathcal{S} \leftarrow \emptyset$
3: **for** each $cluster \in clusters$ **do**
4: $\mathcal{S} \leftarrow \mathcal{S} \cup \text{bestBundle}(cluster, \alpha, f, \beta)$;
5: **return** $\mathcal{S}$

Algorithm 7 `BestBundle` Routine

Input: set of items $\mathcal{C}$, α, f, β;
Output: one valid bundle
1: $best \leftarrow 0$
2: **for each** $\omega \in \mathcal{C}$ **do**
3: $s \leftarrow$ `pickBundle`$(\omega, \mathcal{C}, \alpha, f, \beta)$
4: **if** score$(s) > best$ **then**
5: $best \leftarrow s$
6: **return** $best$

6 INTEGER PROGRAMMING

Like many combinatorial problems, COMPOSITE RETRIEVAL can be tackled with an integer programming approach. Let us consider the following variables:

$$x_{uj} = \begin{cases} 1 & \text{if } u \in S_j \\ 0 & \text{otherwise} \end{cases} \qquad u \in \mathcal{I}, j = 1, \ldots, k$$

$$y_{uvj} = \begin{cases} 1 & \text{if } u \in S_j, v \in S_j \\ 0 & \text{otherwise} \end{cases} \qquad u, v \in \mathcal{I}, u < v$$

$$z_{ij} = \max_{u \in S_i, v \in S_j} s(u, v) \qquad i, j = 1, \ldots, k \quad i < j$$

Using these definitions, the objective function of our problem may be rewritten as:

$$\gamma \sum_{j=1}^{k} \sum_{\substack{u,v \in S_j \\ u<v}} s(u, v) + (1 - \gamma) + \sum_{1 \leq i < j \leq k} (1 - \max_{u \in S_i, v \in S_j} s(u, v)) =$$

$$\gamma \sum_{j=1}^{k} \sum_{\substack{u,v \in \mathcal{I} \\ u<v}} s(u, v) y_{uvj} - (1 - \gamma) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} z_{ij} + (1 - \gamma) \frac{k(k-1)}{2}$$

Then the problem can be formulated as maximizing this function subject to:

$$\sum_{u \in \mathcal{I}} c_u x_{uj} \leq \beta \quad j = 1, \ldots, k \tag{1}$$

$$\sum_{\substack{u \in \mathcal{I} \\ u.\alpha = a}} x_{uj} \leq 1 \quad j = 1, \ldots, k \quad \forall a \text{ possible value of } \alpha \tag{2}$$

$$y_{uvj} \leq \frac{x_{uj} + x_{vj}}{2} \quad j = 1, \ldots, k \qquad u, v \in \mathcal{I} \tag{3}$$

$$z_{ij} \geq s(u, v)(x_{ui} + x_{vj} - 1) \quad 1 \leq i < j \leq k \; u, v \in \mathcal{I} \tag{4}$$

$$x_{uj} \in \{0, 1\} \qquad y_{uvj} \in \{0, 1\} \qquad 0 \leq z(jj') \leq 1$$

Constraints (1) assert that the budget does not exceed the threshold for each bundle. Constraints (2) guarantee complementarity (two items with the same value for property α may not be assigned to the same bundle). Constraints (3) ensure that y_{uvj} is equal to 0 if items u, v are not in the same bundle j. Finally, constraints (4) give to z_{ij} the maximum similarity between items in bundles i and j.

For solving the integer programming formulation we implemented a Branch-and-Cut algorithm using CPLEX 12.1. We added to the standard CPLEX algorithm a primal heuristic and valid cutting planes

specifically derivated for the problem. The heuristic is a simple fast greedy procedure that is applied on every node of the tree and is capable of finding good solutions in the first stages. The cutting planes, like $\sum_{v \in \mathcal{I} v.\alpha = a} y_{uvj} \leq x_{uj} \forall v \in \mathcal{I}$ and a possible value of α, are very useful to close the initial gap relaxation. These customized components have an important impact on the computational performance of the algorithm, but due to lack of space we cannot extend furtherly on this matter.

7 Experiments

In this section we test the effectiveness and efficiency of our algorithms on a real-world dataset.

7.1 Experimental framework

Dataset: Our experiments are conducted on a database extracted from a recent sample of Yahoo! Local[2] containing user-contributed restaurant reviews. We picked all cities for which there are at least 100 restaurants with reviews. This left us with 38,530 restaurants distributed over 149 US cities, with the largest cities being New York (over 2,000 restaurants) and Los Angeles (over 1,000 restaurants). Most cities in our sample have between 300 and 500 restaurants.

Each restaurant has one of the following 3 average meal prices: cheap ($10), moderate ($20) or expensive ($30). Each restaurant has multiple values for cuisine, drawn from 291 possible cuisine values, including *American*, *Italian*, *Cajun*, *Omelets*, *Contemporary*, *Sandwiches*, etc. The restaurant with the highest number of cuisine types has 34 cuisines ranging from Californian to Spanish.

Queries, attributes, and budget: Each query is the name of a city and the candidate items $\mathcal{I}$ are the restaurants in that city. For each city, we generate $k = 10$ bundles of recommended restaurants. The compatibility between two restaurants u, v is the number of reviewers that have given both restaurants a positive review. The complementarity attribute α is the cuisine type, meaning that on each bundle we do not want two restaurants having a cuisine type in common. The cost of each bundle is the sum of the costs of having a meal at each of the restaurants in the bundle. This must not exceed the budget β, which will take one of the following 3 values: $50, $100, and $200. The latter is often equivalent to an unlimited budget given the distribution of cuisine types and meal prices.

7.2 Implementation

Bundles One-by-One: We implemented the `BOBO` method described in Algorithm 4, varying the number of bundles that are generated before `ChooseBundles` is invoked to select the $k = 10$ that will be returned. `BOBO-1` is our baseline and it just picks 10 bundles at random (so `ChooseBundles` becomes trivial). We also implemented `BOBO-5`, that generates $5k = 50$ bundles, `BOBO-10`, that generates $10k = 100$ bundles, and `BOBO-E` (Exhaustive), that picks as pivots all of the items exhaustively and generates $c = |\mathcal{I}|$ candidate bundles with the routine `pick_bundle` described in Algorithm 5. In all but the exhaustive case, an item that has already been picked as a member of a bundle cannot be picked as a pivot, as specified by Algorithm 4. The minimum score of a bundle μ is determined by generating 5 bundles at random and then picking the median score.

Constrained clustering: The `C-HAC` method described in Algorithm 3 is implemented to stop at $k = 10$ clusters, or when no further merge operations can be executed. Typically, the latter happens much earlier. We experimented with generating more than k clusters and then using `ChooseBundles`; but the results are basically the same as the ones we report here, because anyway the algorithm generates a number of clusters larger than k, typically around $\frac{1}{3}|\mathcal{I}|$.

Cluster-and-pick: The `CAP` method described in Algorithm 6 is implemented using METIS [17] as the clustering method, with $k = 10$ clusters. Procedure `BestBundle` in Algorithm 7 is invoked once for each of the 10 clusters, returning one bundle per cluster.

Integer programming (`IP`): the branch-and-cut implementation described in Section 6 is run for a maximum time of 1 minute for each instance.

7.3 Results

We experimented with $\gamma \in \{0.1, 0.2, \ldots, 0.9\}$, evaluating the effectiveness of the different methods using the objective function defined in Section 3. Figure 1 shows the results for budgets of 50, 100, and 200, and $\gamma \in \{0.1, 0.3, 0.7, 0.9\}$. Results with intermediate values of γ lie between those we show here and are thus omitted. We observe the following:

- The best performance is in general obtained by `IP`. However, due to the high running times (see below), this algorithm may be seen more as a benchmark than as a really feasible approach.
- There is a large advantage of `BOBO-5` over `BOBO-1`: the baseline method `BOBO-1` performs significantly worse than the other methods for a wide range of settings.
- `CAP` and `C-HAC` are in general comparable: There is a difference of less than 3% (in terms of median) in favor of `C-HAC` for $\gamma = 0.1$ and for $\gamma = 0.3$ under a small budget of $\beta = 50$. In some cases ($\gamma = 0.9, \beta = 200$), `C-HAC` is better for up to 20% (in terms of median).

2. http://local.yahoo.com/

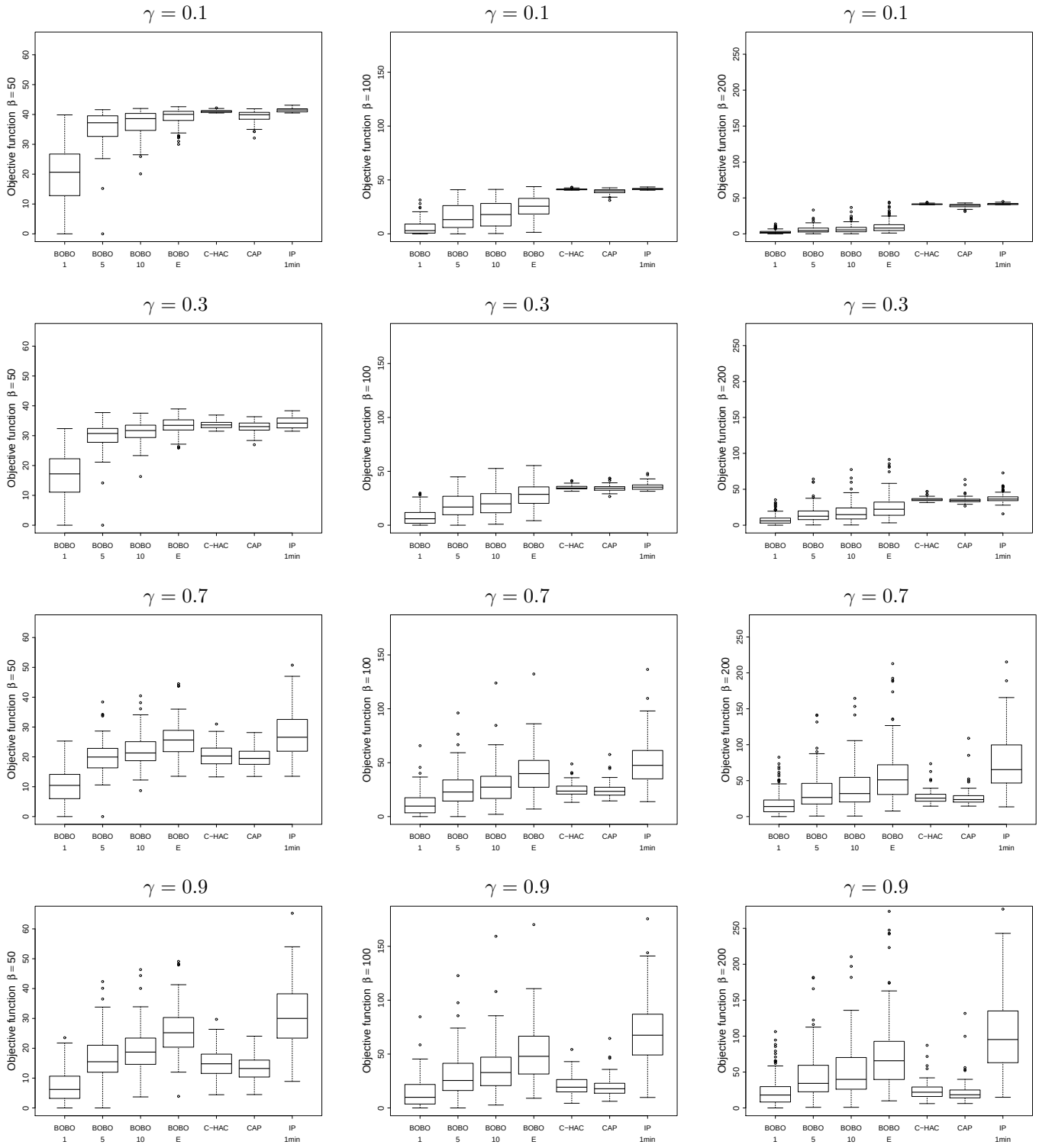
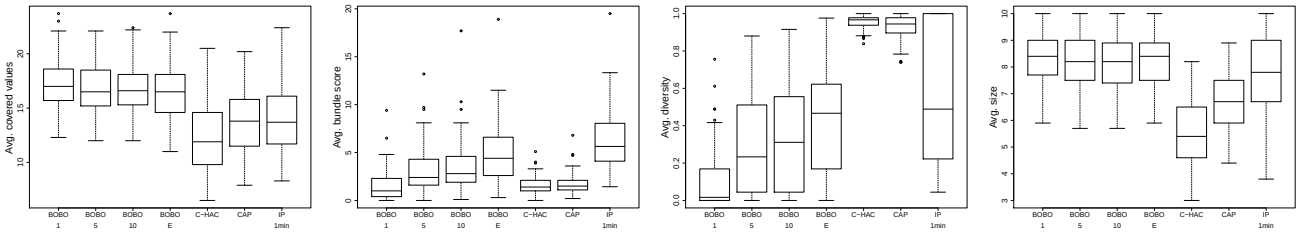


Fig. 1. Objective function for (top to bottom) $\gamma \in \{0.1, 0.3, 0.7, 0.9\}$ and (left to right) $\beta \in \{50, 100, 200\}$. The scale of the y-axis is the same for the plots belonging to the same column.

- For small γ, `CAP` is better than `BOBO-5`; for $\gamma = 0.1, \beta = 200$, `CAP` its median is almost 9 times larger.
- For large γ, `BOBO-5` is better than `CAP`; up to 2 times larger for $\gamma = 0.9, \beta = 200$.

The performance of the methods highly depends on γ, i.e. on the relative importance given to the average score of the bundles and the diversity of the set of bundles.



(a) Covered attribute values (b) Bundle score (c) Diversity (d) Size

Fig. 2. Average number of covered complementarity attribute values, bundle score, diversity, and size of produced bundles for $\gamma = 0.5$, $\beta = 100$.

When diversity is highly important, the CAP approach that first creates a global clustering of the items obtains the best performance. On the other hand, methods that construct good bundles around randomly chosen pivots produce better results when diversity is less important in the objective function.

Running time. We ran our experiments on a mid-range Linux server (8×2.6GHz 64-bit Intel processors). Besides IP, that was run for 1 minute for each instance, in our setting the running time of our algorithms was comparable except for C-HAC, where the average running time is dominated by the larger problem instances. Something similar, but not as pronounced, is observed for BOBO-E. The average and median time for the different methods is shown on Table 1. The IP method is one order of magnitude slower, and the table excludes 94 problem instances (5.2% of them) where the IP formulation was unable to find a solution in one minute.

TABLE 1
Running time in seconds.

	Mean	Median
CAP	2.0	2.0
C-HAC	19.8	2.9
BOBO 1	1.7	1.7
BOBO 5	2.0	2.1
BOBO 10	2.3	2.3
BOBO E	6.4	2.8
IP - 1min	49	60

Comparison with the IP benchmark. Figure 3 shows the comparison between BOBO-E, the heuristic that performs best in most cases with a relatively high running time, and the IP implementation. We use a different mark for each β. As it may be seen, IP gives a better award in the majority of the cases, specially when the budget is large. However, observe that the average running time is of 49 seconds, that is, almost 8 times more than its counterpart. In those experiments, IP found the optimal solution in 387 out of 1815 instances ($\approx 20\%$ of the total) with an average running time of 11 seconds, showing a better performance for high budget and for high γ values. However, for 94

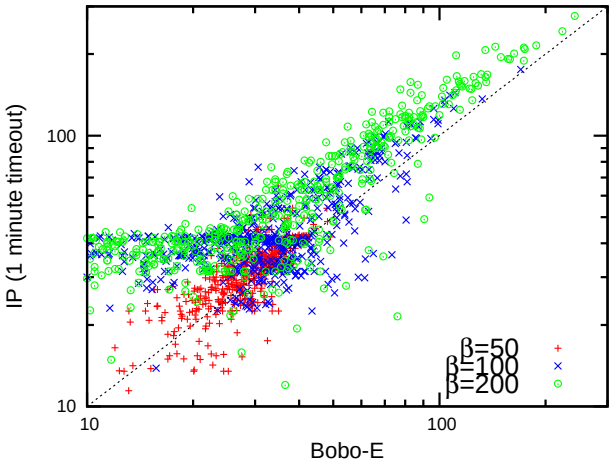


Fig. 3. Objective function for BOBO-E vs. IP (best seen in color). Each dot is a problem instance.

instances ($\approx 5\%$) it failed to produce any solution, this being more notorious for instances with high budget.

Complementarity, score, diversity, and size. To further understand the differences, we can look at some properties of the bundles: number of covered values of the complementarity attribute, bundle score, diversity, and size. A comparison of these quantities is shown in Figure 2 for $\gamma = 0.5$. We can observe that BOBO produces bundles covering more values of the complementarity attribute, and also having a higher degree of compatibility. Methods CAP and C-HAC produce more diverse bundles, which is expected given that they are both based on clustering. Finally, with respect to size, C-HAC produces small bundles, CAP medium-sized ones, and BOBO tends to produce larger bundles.

The Choose phase. To evaluate how important the Choose phase is, we compared the densest-subgraph heuristic ChooseBundles in Algorithm 2 with a simpler method, in which we simply pick the top $k = 10$ bundles by score. The results are shown in Figure 4. There are significant gains ranging from about 30% to almost 300% (in terms of the median of the objective function) of using the densest-subgraph heuristic.

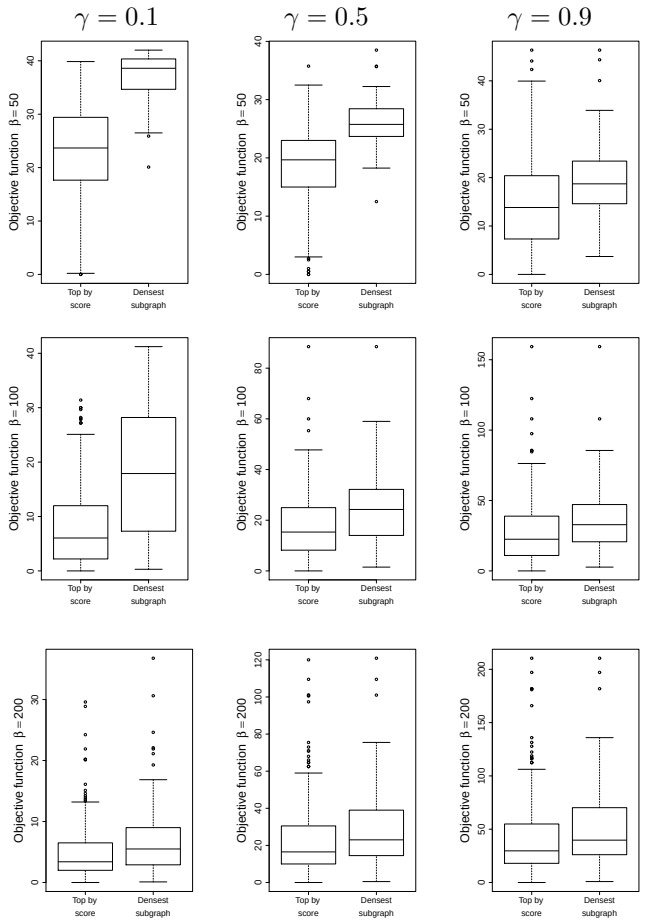


Fig. 4. Comparison of `BOBO-5` using two different methods for the Choose phase: the densest-subgraph heuristic `ChooseBundles`, or simply picking the ones with the highest score.

8 Related Work

Composite Retrieval: The notion of composite retrieval was proposed with different semantics in recent work [7], [3], [18], [9], [22]. CARD [7] is a framework for finding top-k recommendations of packages of products or services. A language similar to SQL is proposed to specify user requirements an the combination of atomic costs. Recommended packages are of fixed size. In [3], the problem is to query documents and build packages formed by multiple entities such as a city, a hotel or an airline. A package in that case is of fixed size (as a simple case of our notion of budget). In [18], the authors explore retrieving bundles of items in the form of a star (e.g., an iPhone and all its accessories). A bundle is subject to budget and item compatibility constraints (e.g., co-purchasing and co-browsing). In [9], the authors design a graph traversal algorithm that retrieves itineraries in a city. An itinerary is an ordering of a set of points of interest (e.g., landmarks in a city), subject to time constraints. The ordering imposes a different relationship between points of interest in the form of a chain, and makes the retrieval problem significantly different from the model developed in [18]. The solution relies on an adaptation of graph traversal for the orienteering problem. Finally, in [22], the authors explore returning approximate solutions to composite retrieval. The focus of the work is on using a Fagin-style algorithm for variable size bundles and proving its optimality.

None of these works accounts for diversity across bundles and formalizes retrieval as a clustering problem that accounts for intra-bundle compatibilities and complementarity as well as a general notion of budget that goes beyond bundle size.

Diversity: Our work is related to result diversification in Web search, database queries, and recommendations. Diversifying Web search results and recommendations aims to achieve a compromise between relevance and result heterogeneity. In [15], the authors adopt an axiomatic approach to diversity that aims to address user intent. They show that no diversification function can satisfy all axioms together and illustrate that with concrete examples. In [2], taxonomies are used to sample search results in order to reduce homogeneity. In the database context, Chen and Li [8] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [20], a hierarchical notion of diversity in databases is introduced, and efficient top-k processing algorithms are developed. In recommendations [23], [11], results are typically post-processed using pairwise item similarity in order to generate a list that achieves a balance between accuracy and diversity. For example, in the recommender systems world, the approach in [23] defines an intra-list similarity which relies on mapping items to taxonomies to determine topics or using item features such as author and genre. The method is based on an exhaustive post-processing algorithm which operates on a top-N list to compute the top-K results ($N > K$). In contrast, in [11], diversity is formulated as a set-coverage problem. These approaches do not retrieve item bundles under validity, diversity and compatibility constraints. Finally, [13] introduces diversity in the framework of sponsored search ads, proposing algorithms for the selection of ads that intend to increase heterogeneity while not significantly reducing revenue and maintaining an incentive for advertisers to keep their bids as high as possible. Heterogeneity is aimed at as a notion that spans various occurrences of the same query, and not just a single one.

9 Conclusions and Future Work

In many complex search and browsing applications returning item bundles is more appropriate than ranked lists. In this article, we present composite

retrieval as an alternative to ranked lists and formalize the problem as finding the k highest scoring item bundles. Our formulation ensures complementarity among items in the bundles and diversity between retrieved bundles.

We show that the problem is **NP**-hard and explore various heuristics, validating them on Yahoo! Local, a real-world database of user-generated restaurant reviews.

In our future work, we plan to explore personalized composite retrieval (e.g., retrieve item bundles that are most compatible with my interests, find the best item bundles including a specific item, find the best bundle compatible with a given item). These new problems bare similarities with item recommendation that account for a user profile, with the added flexibility of querying those recommendations in a stylized fashion. We conjecture that such queries will simplify retrieval complexity while raising an additional challenge of returning results as fast as possible.

REFERENCES

[1] D. A., B. K., and B. P.S. Using assignment constraints to avoid empty clusters in k-means clustering. In *Basu, S., Davidson, I., Wagstaff, K. (eds.) Constrained Clustering: Algorithms, Applications and Theory*, 2008.

[2] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. *World Wide Web*, 9(4):397–429, 2006.

[3] A. Angel, S. Chaudhuri, G. Das, and N. Koudas. Ranking objects based on relationships and fixed associations. In M. L. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold, editors, *EDBT*, volume 360 of *ACM International Conference Proceeding Series*, pages 910–921. ACM, 2009.

[4] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203 – 221, 2000.

[5] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 1 edition, 2008.

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

[7] A. Brodsky, S. M. Henshaw, and J. Whittle. Card: a decision-guidance framework and application for recommending composite alternatives. In P. Pu, D. G. Bridge, B. Mobasher, and F. Ricci, editors, *RecSys*, pages 171–178. ACM, 2008.

[8] Z. Chen and T. Li. Addressing diverse user preferences in sql-query-result navigation. In C. Y. Chan, B. C. Ooi, and A. Zhou, editors, *SIGMOD Conference*, pages 641–652. ACM, 2007.

[9] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In M. H. Chignell and E. Toms, editors, *HT*, pages 35–44. ACM, 2010.

[10] I. Davidson and S. S. Ravi. Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results. In *Knowledge Discovery in Databases: PKDD 2005*, Lecture Notes in Computer Science, pages 59–70, 2005.

[11] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, editors, *KDD*, pages 289–298. ACM, 2009.

[12] U. Feige, D. Peleg, and G. Kortsarz. The dense -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[13] E. Feuerstein, P. A. Heiber, J. Martínez-Viademonte, and R. A. Baeza-Yates. New stochastic algorithms for scheduling ads in sponsored search. In V. A. F. Almeida and R. A. Baeza-Yates, editors, *LA-WEB*, pages 22–31. IEEE Computer Society, 2007.

[14] A. V. Goldberg. Finding a maximum density subgraph. Technical report, Berkeley, CA, USA, 1984.

[15] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.

[16] R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21(3):133 – 137, 1997.

[17] G. Karypis and V. Kumar. *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*, 1995.

[18] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Constructing and exploring composite items. In A. K. Elmagarmid and D. Agrawal, editors, *SIGMOD Conference*, pages 843–854. ACM, 2010.

[19] B. Saglam. A mixed-integer programming approach to the clustering problem with an application in customer segmentation. Master's thesis, Koc University Graduate School of Sciences and Engineering, 2005.

[20] E. Vee, J. Shanmugasundaram, and S. Amer-Yahia. Efficient computation of diverse query results. *IEEE Data Eng. Bull.*, 32(4):57–64, 2009.

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained K-means Clustering with Background Knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[22] M. Xie, L. V. Lakshmanan, and P. T. Wood. Breaking out of the box of recommendations: From items to packages. In *RecSys*, 2010.

[23] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In A. Ellis and T. Hagino, editors, *WWW*, pages 22–32. ACM, 2005.