Top-k diversity queries over objects embedded in a low-dimensional vector space aim to retrieve the best \( k \) objects that are both relevant to given user's criteria and well distributed over a designated region. An interesting case is provided by spatial Web objects, which are produced in great quantity by location-based services that let users attach content to places and are found also in domains like trip planning, news analysis, and real estate. In this paper we present a technique for addressing such queries that, unlike existing methods for diversified top-\( k \) queries, does not require accessing and scanning all relevant objects in order to find the best \( k \) results. Our Space Partitioning and Probing (SPP) algorithm works by progressively exploring the vector space, while keeping track of the already seen objects and of their relevance and position. The goal is to provide a good quality result set in terms of both relevance and diversity. We assess quality by using as a baseline the result set computed by MMR, one of the most popular diversification algorithms, while minimizing the number of accessed objects. In order to do so, SPP exploits score-based and distance-based access methods, which are available, e.g., in most geo-referenced Web data sources. Experiments with both synthetic and real data show that SPP produces results that are relevant and spatially well distributed, while significantly reducing the number of accessed objects and incurring a very low computational overhead.

Categories and Subject Descriptors: H.2.4 [Database Management]: Systems
General Terms: Algorithms, Design, Experimentation, Performance
Additional Key Words and Phrases: Diversification, Scoring, Top-k, Ranking, Aggregation

1. INTRODUCTION

Geo-referenced data are becoming increasingly available on the Web, especially after the advent of location-based services, whereby users can create content attached to places. Web spatial objects are also found in real estate directories, local news aggregators, image sharing sites, and travel services. Current approaches to querying geo-referenced data have focused mostly on finding relevant objects near a given location [Chen et al. 2006; Cong et al. 2009]. The spatial dimension has also been used to re-rank result sets, so as to present Web objects sorted by both relevance and distribution in space [van Kreveld et al. 2005; Tang and Sanderson 2010].

Examples of queries that require the coverage of a region through spatial scattering of results can be found in several domains: \( i \) a user moving to a new city wants an overview of real estate offers that meet some relevance criteria (e.g., price, square meters, etc.) and cover most neighborhoods; \( ii \) a user spending the night in town wants
suggestions of venues where friends check in frequently and that are well distributed
over town; iii) a journalist making a countrywide research on breaking news wants a
list of local news ranked by topical relevance from local news services that represent
well the whole country; iv) a tourist deciding the ski station to choose for the day wants
a preview of snow conditions at various places as visible in recent user-generated pho-
tos of the region.

In this paper, we address the problem of answering top-\( k \) diversity queries over online
data sources covering a region of interest. We assume that objects are represented in
a vector space and can be fetched through interfaces, common for Web data sources,
granting sorted access either by relevance (e.g., an object property or the degree of
match with the query) or by distance from a given point. Our objective is to improve the
performance of diversified query processing by accessing only a small number of objects
that guarantee to find the best result set in terms of both relevance and diversity. This
is in contrast with classical diversification techniques, which access all the objects first,
and then choose the best subset of diversified objects. As an example, consider a real
estate query: a sample search in a commercial service for flats in London between
£200,000 and £300,000 returned 60,000+ results; if the user wants to browse just a
few dozens of them in diverse neighborhoods, we should access and present a number
of objects proportional to the user’s wishes, scattered throughout the London region,
without accessing all the 60,000+ relevant flats. In addition, we rely solely on the
presence of sorted access methods based on relevance and distance, without requiring
the knowledge of the specific indexing structures being used, as these typically reside
on remote third-party services.

Top-\( k \) diversity queries over a vector space require a mix of techniques from top-\( k \)
query processing and result diversification. As in top-\( k \) query processing [Marian et al.
2004; Schnaitter and Polyzotis 2008], the cost model of access methods requires min-
imizing the number of fetched objects. To this end, a threshold (upper bound) on the
value of an objective function that quantifies both relevance and diversity is main-
tained by means of a so-called bounding scheme. As more candidate objects are ac-
cessed via probe queries, the upper bound decreases until the guarantee is reached
that no unseen object can lead to a value of the objective function better than the one
determined by the already retrieved objects. However, unlike in top-\( k \) query process-
ing, the need of ensuring diversification of results entails that the objective function
cannot be computed on individual objects, because the diversity measure (e.g., spatial
scatter) must consider sets of objects at once.

In addition, existing diversification methods [Carbonell and Goldstein 1998; Golla-
pudi and Sharma 2009] are not directly applicable in our context, since they construct
a solution incrementally, by comparing all the relevant objects (i.e., the results of the
user query) with the objects that have been top-ranked so far, thus materializing and
scanning all of them several times [Carbonell and Goldstein 1998]. This is not feasi-
ble for data sets involving thousands of objects or more, especially in scenarios such as
those considered in this paper, where online queries are performed with handheld
devices.

We propose a novel approach, which integrates the notion of probe queries into the
framework of result diversification, providing on-the-fly construction of top-\( k \) result
sets that are both relevant and diverse, by using only sorted access methods and with-
out fetching all relevant objects. Our approach works as follows. The top-\( k \) set is built
incrementally, as in [Carbonell and Goldstein 1998], adding each time the object that
maximizes an objective function based on both relevance and diversity. Instead of vis-
it ing all objects, at each step, we probe the vector space by issuing distance-based
queries at suitable points, called probing locations, that are likely to lie close to the
best objects. We may alternatively use score-based access to retrieve objects with high
relevance. Based on the retrieved objects, we maintain an upper bound on the value of the objective function that can be attained by using the unseen objects. The currently best object is added to the result set when the value of the objective function determined by its inclusion is at least as high as the upper bound. Note that the choice of probing location and the proper selection of score-based and distance-based access (akin to a pulling strategy [Schnaitter and Polyzotis 2008]) exploits the geometry of the vector space. The proposed approach introduces efficiency without compromising the quality of diversification wrt. the best known general-purpose diversification algorithms.

The main contributions of the paper are as follows:
(1) Bounded diversification with sorted access methods is introduced for the first time and formally defined.
(2) The Pull/Bound Maximum Marginal Relevance (PBMMR) family of algorithms is illustrated, which exploits spatial probing locations and the adaptive selection of score-based and distance-based access to reduce the number of fetched objects.
(3) An instance of PBMMR, called Space Partitioning and Probing (SPP), is presented, whose pulling strategy uses a tight upper bound, thus ensuring the optimal selection of the access method used to fetch the objects.
(4) SPP is shown to attain the same diversification quality and exactly the same output as MMR [Carbonell and Goldstein 1998], the most popular result diversification algorithm, but accessing only a fraction of the objects. For example, in a 2-dimensional synthetic data set characterized by uniform distribution, only 4% of 100000 objects were accessed to retrieve the top-10 results. In a real data set comprising more than 65000 properties in London, only 1% of the objects were accessed to retrieve the top-10 real estate results. Overall, this is a substantial gain over past work on spatial scatter queries [van Kreveld et al. 2005], which access all objects.
(5) We consider the practically relevant scenario of batched access, in which distance-based access returns a set of objects instead of just a single object, and present an alternative version of SPP to deal with that case, which produces almost the same output as MMR and SPP, but accessing an even smaller number of objects. Indeed, only 0.2% and 0.4% of the objects were accessed, respectively, in the synthetic and real data set mentioned above, assuming that prior knowledge on the object distribution is available.

The rest of the paper is organized as follows. In Section 2, we formalize the notion of diversification problem and introduce the baseline MMR algorithm [Carbonell and Goldstein 1998]. In Section 3 we introduce bounded diversification and show that it is conveniently addressed by the PBMMR family of algorithms. In Section 4, such a family is instantiated to the concrete SPP algorithm, equipped with a specific bounding scheme and pulling strategy. We adapt SPP to cover the case of batched access in Section 5. An extensive experimental evaluation of our techniques is presented in Section 6, while related work is discussed in Section 7. Finally, in Section 8 we conclude.

2. PRELIMINARIES

Consider a universe of objects $\Omega$, where each object $o \in \Omega$ is associated with a real-valued $d$-dimensional feature vector $x(o) \in \mathbb{R}^d$, denoting the characteristics of the object.

The diversity of two objects can be expressed by a measure $\delta: \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}^+$, where the value 0 denotes maximum similarity, i.e., minimum diversity. Let $q$ be a query that selects a finite set $\mathcal{O} \subseteq \Omega$ of objects. The relevance of an object $o \in \mathcal{O}$ to $q$ can be represented by a score $S_q(o) \in \mathbb{R}$. Let $N = |\mathcal{O}|$ denote the cardinality of the set $\mathcal{O}$, and let $\mathcal{O}_K \subseteq \mathcal{O}$ be a subset of $K$ objects that are selected, e.g., to be presented to the user. We
are interested in identifying a subset \( O_K \) that contains objects that are both relevant to the query and diverse.

The diversification problem, i.e., computing the best subset \( O_K^* \) of \( O \), can be expressed as follows [Gollapudi and Sharma 2009].

**Definition 2.1 (diversification).** A diversification problem is a tuple \( \langle O, S_q, \delta, K, F \rangle \), where

- \( O \) is a finite set of objects; each object \( o \in O \) has a feature vector \( \mathbf{x}(o) \) in \( \mathbb{R}^d \). \( O \)
  might be the result of a user query \( q \), which also determines the relevance \( S_q \) of the objects in \( O \).
- \( S_q : O \rightarrow \mathbb{R} \) is a score function on the objects in \( O \).
- \( \delta : O \times O \rightarrow \mathbb{R}^+ \) is a distance function between pairs of objects in \( O \).
- \( K \) is a positive integer such that \( K \leq |O| \).
- \( F : 2^O \times S_q \times \delta \rightarrow \mathbb{R} \) is an objective function that associates each set \( \mathcal{O}_I \subseteq O \) with a real number, based on \( S_q \) and \( \delta \), i.e., taking into account both relevance and diversity.

Solving a diversification problem entails finding the set of objects:

\[
O_K^* = \arg\max_{O_K \subseteq O, |O_K| = K} F(O_K; S_q(\cdot), \delta(\cdot, \cdot))
\]  

(1)

In the following, we assume scores \( S_q(o) \) to be finite and then normalized in the \([0, 1]\) range, where 1 indicates the highest relevance. We also assume that the diversity measure is the Euclidean distance \( \delta(o_u, o_v) = \| \mathbf{x}(o_u) - \mathbf{x}(o_v) \|_2 \) between the objects’ feature vectors, where \( \| \cdot \|_2 \) denotes the \( l_2 \) norm.

The exact solution of a diversification problem is a set \( O_K^* \subseteq O \) defined as in (1), where score ties are resolved using a tie-breaking criterion. An algorithm that maps a problem \( \langle O, S_q, \delta, K, F \rangle \) to the optimal solution \( O_K^* \) upon termination is said to be correct. Solving Equation (1) exactly has been shown to be NP-hard [Gollapudi and Sharma 2009] for various objective functions. Hence, several approximate greedy algorithms have been proposed and validated experimentally. Among these, \( \text{MMR} \) (Maximum Marginal Relevance) [Carbonell and Goldstein 1998] is one of the most popular. The original paper [Carbonell and Goldstein 1998] does not explicitly provide a formulation for the \( \text{MMR} \) objective function. In this paper, we propose to adopt the following objective function:

\[
F(O_K) = \frac{1 - \lambda}{K} \sum_{o \in O_K} S_q(o) + \lambda \min_{o_u, o_v \in O_K} \delta(o_u, o_v),
\]

(2)

where \( \lambda \) is a parameter in \([0, 1]\) specifying the trade-off between relevance and diversity. NP-hardness can be shown by reduction from the minimum \( k \)-center problem [Gonzalez 1985], posing \( \lambda = 1 \).

Algorithm 1 illustrates the details of \( \text{MMR} \), which incrementally constructs \( O_K \) by adding, at each step, an object that is both relevant and diverse from the already selected objects. The initial object is chosen according to some initialization strategy \( IS \); typically, the object that maximizes \( S_q(\cdot) \) is selected. The added object \( o^* \) (line 3 of Algorithm 1) maximizes the diversity-weighted score \( \sigma \), defined as:

\[
\sigma(o; O_K) = (1 - \lambda)S_q(o) + \lambda \min_{o' \in O_K} \delta(o, o')
\]

(3)

The algorithm proposed for optimizing the \( \text{MMR} \) objective function assumes that all the \( N \) objects relevant to the query are retrieved and re-ranked so as to select the top \( K \) diversified elements. Therefore, this algorithm exhibits a complexity that depends on \( N \). The complexity of the loop at line 2 of Algorithm 1 is \( O(K^2N) \), since each of
ALGORITHM 1: MMR algorithm

Input: Set of objects \( O \); result size \( K \)
Output: Selection \( O_K \) from \( O \)

Parameters: Initialization Strategy \( IS \)

1. \( O_K := \{ IS \cdot \text{initialObject} () \} \);
2. \( \text{while } (|O_K| < K) \)
3. \( o^* := \arg \max_{o \in O \setminus O_K} \left\{ (1 - \lambda) S_q (o) + \lambda \min_{o' \in O_K} \delta (o, o') \right\} \);
4. \( O_K := O_K \cup \{ o^* \} \);
5. \( \text{return } O_K \);

the \( K \) iterations requires \( |O_K| \cdot N \) evaluations of measure \( \delta \). If the initialization step at line 1 selects the object that maximizes \( S_q () \), then this simply requires to scan all objects in \( O \). The resulting overall time complexity of MMR is \( O(K^2 N) \).

3. BOUNDED DIVERSIFICATION WITH SORTED ACCESS METHODS

The diversification problem addressed in this paper assumes that the feature vectors of the objects are contained in a finite bounded region and that the objects are retrieved using only sorted access methods. The goal is to compute a relevant and diversified result set, by accessing only a small fraction of the relevant objects through the available access methods. We consider two kinds of sorted access methods for fetching the objects:

- Score-based access. The set \( O \) of relevant objects is accessed sequentially in decreasing order of \( S_q () \), i.e., of relevance to the query.

- Distance-based access. The set \( O \) is accessed sequentially in increasing order of \( \delta (\cdot, v) \), where \( v \) is an arbitrary vector in \( \mathbb{R}^d \) called probing location. For example, objects are retrieved by geographical distance wrt. a given point. Techniques for the efficient implementation of distance-based access are widely deployed, based on indexing structures that enable fast nearest neighbors computation in time sub-linear in the size of \( O \), i.e., \( O(\log N) \) in the average case [Berchtold et al. 1998].

Both kinds of access are largely available in Web data sources, as witnessed by a recent rise in geo-localization services. For example, in the UK alone, there are more than 17,000 real estate services, the majority of which provides forms of distance-based access (e.g., flats neighboring a given address) as well as score-based access (e.g., increasing price) [Furche et al. 2011].

Bounded diversification with sorted access methods is a constrained version of the diversification problem introduced by Definition 2.1:

Definition 3.1 (bounded diversification). A bounded diversification problem (with sorted access methods) is a tuple \( (O, S_q, \delta, K, F, U) \), where

- \( O, S_q, \delta, K, F \) are the same as in Definition 2.1.
- \( U \subset \mathbb{R}^d \) is a bounded convex polyhedron such that \( x(o) \in U \) for every object \( o \in O \).
- The objects in \( O \) can be accessed only by score-based access or by distance-based access.

Again, the exact solution of a bounded diversification problem is a set \( O_K^* \subset O \) defined as in (1), where score ties are resolved using a tie-breaking criterion, and an algorithm is correct if it maps a problem \( (O, S_q, \delta, K, F, U) \) to the optimal solution \( O_K^* \) upon termination.

In much the same way as diversification problems, bounded diversification problems are also NP-hard, as can be shown by a straightforward reduction from diversification to bounded diversification. Therefore, bounded diversification must also be
tackled with heuristic approaches; in turn, this requires evaluating the quality of the output. As the focus of this paper is to apply diversified top-\(K\) query processing in contexts where accessing all relevant objects is impractical, we compare bounded diversification algorithms with “classical” (non-bounded) diversification algorithms. Our goal is to assess how close the former ones can get to the result of the latter ones, without naively accessing and reordering all relevant objects. For illustration, we use as a baseline the objective function and object selection principle of the classical MMR diversification algorithm. To perform the comparison, we proceed as follows: we fix by design the output to be exactly the same as MMR and then investigate the existence and properties of bounded diversification algorithms fulfilling such a requirement. To this aim, we introduce and study the class of MMR-correct algorithms, defined as those deterministic algorithms that comply with the specifications of Definition 3.1 and that output the same result as MMR, i.e., \(O_{MMR}^K\).

To set the ground for the investigation of MMR-correctness, Algorithm 2 introduces the Pull/Bound MMR (PBMMR) template, which provides a general framework for bounded diversification that incrementally builds the solution by adding, one by one, objects to the result set. PBMMR is adapted from the Pull/Bound Rank Join (PBRJ) template originally introduced for the rank join problem in [Schnaitter and Polyzotis 2008].

As MMR, PBMMR starts by selecting the initial object according to an initialization strategy IS (line 1). Throughout the algorithm, several variables are used to maintain the current state of the execution. The essential control variables are initialized at line 2: i) the subset \(D\) of region \(\mathcal{U}\) that has been already explored (and thus discarded as no longer useful for retrieving objects), which is initially set to \(\emptyset\); ii) the highest score possible for unseen objects \(S_q^{\text{last}}\), which is initially set to the maximum value \(1\); iii) the set of currently retrieved objects \(P\), which is initially set to the object selected at line 1.

\(\)\(^{1}\)Henceforth, we use superscripts to qualify the top-\(K\) result set \(O_K\) with the name of the algorithm that produced it.

---

**ALGORITHM 2: PBMMR\((K, U)\)**

Input: Result size \(K\); bounding region \(U\)
Output: Selection \(O_K\) from the objects enclosed in \(U\)

Main variables: Set \(P\) of retrieved objects; discarded region \(D\); last score \(S_q^{\text{last}}\); upper bound \(\tau\); top diversity-weighted score \(\sigma^*\); top object \(o^*\)

Parameters: Initialization Strategy IS; Pulling Strategy PS; Bounding Scheme BS

(1) \(O_K := \{\text{IS.initObject}()\}\);
(2) \(D := \emptyset; S_q^{\text{last}} := 1; P := O_K\);
(3) while \(|O_K| < K\) do
    (4) \(\tau := \infty; \sigma^* = -\infty\);
    (5) if \((P \setminus O_K \neq \emptyset)\) then \(o^* = \arg\max_{o \in P \setminus O_K} \sigma(o; O_K)\); \(\sigma^* = \sigma(o^*; O_K)\);
    (6) while \((\sigma^* < \tau \text{ and } D \subset \mathcal{U})\) do
        (7) \(m := \text{PS.chooseAccessMethod}()\);
        (8) \(o := m\text{.getNextObject}()\);
        (9) \(P := P \cup \{o\}\);
        (10) if \((\sigma(o; O_K) > \sigma^*)\) then \(\sigma^* = \sigma(o; O_K)\); \(o^* := o\);
        (11) \((D, S_q^{\text{last}}) := m\text{.updateDiscardedRegionAndScore}(D, S_q^{\text{last}})\);
        (12) \(\tau := \text{BS.updateBound}(P, D, S_q^{\text{last}})\);
    (13) \(O_K := O_K \cup \{o^*\}\);
    (14) return \(O_K\);
Then, the algorithm performs $K - 1$ iterations to determine the remaining $K - 1$ objects (line 3). Within each iteration, an upper bound $\tau$ on the diversity-weighted score of the unseen objects and the current best diversity-weighted score $\sigma^*$ are initialized (line 4). If there are objects retrieved during a previous iteration that have not been selected (i.e., in $P \setminus O_K$), the one with the highest diversity-weighted score $\sigma$ (as defined in (3)) is marked as the current best candidate object $o^*$ and $\sigma^*$ is updated accordingly (line 5). The loop at line 6 performs an exploration step to update $o^*$ and $\tau$ until the guarantee is reached that its diversity-weighted score $\sigma^*$ is at least as high as the upper bound $\tau$, or the region $\mathcal{U}$ has been entirely explored. At each step of the exploration, the chooseAccessMethod function (line 7) of a given pulling strategy PS decides the access method $m$ to use for retrieving the next object (score-based or distance-based). In the latter case, the function also decides which probing location to use, i.e., from which point in the vector space to start returning objects in increasing order of distance. An implementation of chooseAccessMethod, adopted by the SPP algorithm, is shown in Algorithm 3 and discussed in Section 4.1. As soon as an object is accessed (line 8), it is added to the set $P$ of currently retrieved objects (line 9), and, if its diversity-weighted score is greater than the current best value $\sigma^*$ (line 10), it is set as the current candidate object $o^*$ and $\sigma^*$ is updated accordingly. A distance-based access might enlarge the explored region of space, while a score-based access might lower the value of the highest score possible for unseen objects; therefore $D$ and $S^\text{last}_q$ are updated (line 11), based on the access made. Finally, the updateBound function (line 12) of a given bounding scheme BS computes an upper bound on the diversity-weighted score that can be achieved by unseen objects (i.e., those in $O \setminus P$).

The main similarity between the PBMMR and PBRJ templates lies in the use of an external cycle of $K$ iterations for determining the result set by maintaining a buffer of visited objects ($P$) and a threshold for early termination ($\tau$). Yet, PBMMR has a more complex structure, due to a more complex geometry and to a richer set of available access kinds. This entails very different implementations, in the two templates, of the policies for accessing the next object and for updating the threshold. In particular, one of the main differences between PBMMR and PBRJ lies in the notion of 'data source', which, in PBRJ, is a data source proper, while, in PBMMR, it simply amounts to a different perspective on the same data set through a different probing location or a score-based access. The dynamic nature of such data sources requires proper handling; for this reason, PBMMR includes an internal cycle for accessing objects, while the external cycle is used for determining the probing locations. Among the other differences, we mention the fact that $\mathcal{U}$ the result set of PBRJ contains $n$-tuples of objects (where $n$ is the number of data sources), not simply objects returned by data sources; ii) once an object is selected in the result set of PBMMR it never goes out, while, in PBRJ, a currently best join result can be beaten by other join results and be discarded from the result set; iii) PBMMR requires special bookkeeping for the discarded region due to the geometry of the problem, which is not needed in PBRJ.

Any instance of PBMMR is MMR-correct as long as we assume that updateBound actually returns an upper bound on the diversity-weighted score, and that chooseAccessMethod provides a way to eventually access all unseen objects.

**Theorem 3.2.** PBMMR is MMR-correct.

**Proof.** We prove the claim by induction. In the base case ($K = 1$), both PBMMR and MMR return \{IS, initialObject\}. Assume now, by inductive hypothesis, that PBMMR($K - 1, \mathcal{U}$) = $O^\text{MMR}_{K-1}$. At the $K$-th step, MMR selects the object $o^*$ maximizing the diversity-weighted score wrt. the previous selection $O^\text{MMR}_{K-1}$ among all the objects in $O \setminus O^\text{MMR}_{K-1}$ (line 3 of Algorithm 1). Similarly, PBMMR selects $o^*$ according to the same
Theorem 3.2 shows that the algorithms of the PBMMR criterion, but only among the objects in \( P \setminus O^\text{K-1}_{\text{MMR}} \) (line 5 of Algorithm 2), and then also among extra objects retrieved in the loop of line 6. However, if the value \( \tau \) returned by \( BS\.update\text{Bound} \) is an upper bound on the diversity-weighted score \( \sigma \), then none of the objects unseen by PBMMR can be the one that maximizes \( \sigma \). When we exit the loop, either the current best diversity-weighted score \( \sigma^* \) exceeds \( \tau \) or there are no more objects, therefore the selected object \( o^* \) is necessarily the same between MMR and PBMMR, and thus \( O^\text{K-1}_{\text{MMR}} = O^\text{K-1}_{\text{PBMMR}} \).

Theorem 3.2 shows that the algorithms of the PBMMR family produce exactly the same output as the baseline diversification algorithm MMR, thereby achieving exactly the same quality of result as MMR. In addition, the intermediate result sets iteratively computed by PBMMR are identical to those computed by MMR.

The next step is to characterize the performance of PBMMR in terms of reduction of the number of accessed objects. To this end, we introduce the notions of tight bounding scheme and instance optimality.

First, we introduce the notion of execution state of an algorithm of the PBMMR family when applied to a problem \( I = \langle O, S_q, \delta, K, F, U \rangle \). With this, we can precisely characterize the best value of the objective function that can be achieved from a given state. Among the \( P \subseteq O \) objects extracted at a certain point of the execution, the set of objects \( O_t \subseteq P \), with \( |O_t| = \ell \leq K \), retained as part of the solution is called a selection; \( O_t \) is final if \( \ell = K \), running otherwise. Let \( v_1, \ldots, v_V \) be the probing locations used by the algorithm, i.e., the points from which distance-based accesses are made. If the last object retrieved by distance-based access from a probing location \( v_u \) is at a distance \( r_u \) from \( v_u \), it is known that no unseen object may occur inside the open hypersphere \( \Sigma_u \) centered at \( v_u \) with radius \( r_u \), where \( \Sigma_u = \{ x \in \mathbb{R}^d | \delta(\mathbb{x}, v_u) < r_u \} \). We define the union of the hyperspheres centered at the probing locations

\[
D = \bigcup_{u=1}^{V} \Sigma_u
\]

as the discarded region using distance-based access.

The last score \( S^\text{last}_q \) is the score of the last object fetched by score-based access, or 1 if no object has been retrieved in this way. Finally, a state for \( I \) is a tuple \( \langle O_t, P, D, S^\text{last}_q \rangle \); the state is running if \( O_t \) is running, final otherwise.

We now introduce the notion of tight bounding scheme. This allows us to compute, based on the running state, an upper bound on the diversity-weighted score that is the smallest possible (“tight”), in the sense that it can be achieved in some hypothetical continuation of the instance being explored, i.e., it corresponds to some assignment of the admissible locations and the scores of the unseen objects.

**Definition 3.3 (tightness).** Let \( I = \langle O, S_q, \delta, K, F, U \rangle \) be a bounded diversification problem and \( \gamma = \langle O_t, P, D, S^\text{last}_q \rangle \) a running state for \( I \).

- A continuation of \( I \) wrt. \( \gamma \) is a problem \( \langle O', S_q, \delta, K, F, U \rangle \) such that \( P \subseteq O' \) and, for every unseen object \( o \in O' \setminus P \), we have \( x(o) \in U \setminus D \) and \( S_q(o) \leq S^\text{last}_q \). Also, \( \sigma(o; O_t) \) is called a potential outcome of \( I \) wrt. \( \gamma \).

- A bounding scheme \( BS \) is tight if, for every problem \( I \) and running state \( \gamma = \langle O_t, P, D, S^\text{last}_q \rangle \) for \( I \), function \( BS\.update\text{Bound}(P, D, S^\text{last}_q) \) always returns a potential outcome of \( I \) wrt. \( \gamma \).

In order to characterize the performance of bounded diversification algorithms, which depends on the number of objects accessed to return the top-\( k \) results, we adapt
to our context the \( \text{sumDepths} \) cost metric and the notion of instance optimality, which are common in top-\( k \) query answering [Ilyas et al. 2008].

**Definition 3.4** (\( \text{sumDepths} \)). Let \( \text{depth}(A, I, i) \) indicate:

- the number of objects retrieved by algorithm \( A \) on problem \( I \) using distance-based access from probing location \( v_i \), for \( i > 0 \);
- the number of objects retrieved by \( A \) on \( I \) using score-based access, for \( i = 0 \).

Given the probing locations \( v_1, \ldots, v_V \) used by algorithm \( A \) on problem \( I \), we define \( \text{sumDepths}(A, I) \) as \( \sum_{i=0}^{V} \text{depth}(A, I, i) \).

The \( \text{sumDepths} \) metric counts the total number of accesses to objects using score-based and distance-based access, and thus provides an indication of the amount of I/O performed by \( A \) to solve \( I \). Let \( \mathcal{A} \) be a class of deterministic, MMR-correct bounded diversification algorithms, and let \( \mathcal{T}^K \) be a class of problems of the form \( (\mathcal{O}, S_q, \delta, K, F, \mathcal{U}) \) satisfying Definition 3.1 for some given \( K \). We say that \( A \) is *instance-optimal* wrt. \( \mathcal{A} \) and \( \mathcal{T}^K \) for the \( \text{sumDepths} \) cost metric if there exist constants \( c_1 \) and \( c_2 \) such that \( \text{sumDepths}(A, I) \leq c_1 \cdot \text{sumDepths}(A', I) + c_2 \) for all \( A' \in \mathcal{A} \) and \( I \in \mathcal{T}^K \). In other words, instance optimality is a form of robustness of performance wrt. the underlying data, meaning that no other algorithm in \( \mathcal{A} \) can perform arbitrarily better within the problem instances in \( \mathcal{T}^K \). The behavior of an instance of the \( \text{PBMMR} \) template depends on the parameters of Algorithm 2: *i*) the initialization strategy \( \mathcal{IS} \), *ii*) the bounding scheme \( \mathcal{BS} \), *iii*) the pulling strategy \( \mathcal{PS} \), which includes the selection of the probing locations to be used. A common initialization strategy consists in selecting the object with highest score. In such a case, if score-based access is available, the object can be efficiently obtained as the first result by score-based access. If only distance-based access is available, one needs to explore the entire region \( \mathcal{U} \) to obtain all the objects in \( \mathcal{O} \) and sort them by score. In order to avoid such a costly step, one could pick a random location in \( \mathcal{U} \), and then select the first object returned by distance-based access from that location. In fact, it turns out that the impact of the initialization strategy on the quality of the result set is small, as shown experimentally in Section 6.

### 4. SPACE PARTITIONING AND PROBING

We now introduce *Space Partitioning and Probing* (SPP), an instance of the \( \text{PBMMR} \) template, and discuss its main components: in Section 4.1 we show how candidate probing locations are chosen, in Section 4.2 we illustrate the bounding scheme, and in Section 4.3 the pulling strategy.

#### 4.1. Probing locations

We start the illustration of the SPP algorithm by discussing the policy for determining the probing locations, i.e., the starting points used for distance-based access.

Ideally, each time a distance-based access is made, one should explore the region of space that grants the highest chances to retrieve the object with the best diversity-weighted score. To this end, at each of the \( K \) iterations of the algorithm (line 3), we fix the probing locations at the most promising points of the unexplored space. Then, we use these probing locations in the iterations of the inner loop (line 6), possibly querying the same location multiple times asking for the next object (and thus increasing the search radius). Such a strategy leverages the access methods offered by most spatial data sources, which use spatial indexes (e.g., kd-trees, R-trees) to support incremental nearest-neighbor queries.

At each of the \( K \) main iterations, the most promising probing locations are points that lie within the bounding region \( \mathcal{U} \) and are as far as possible from all the objects of the current selection \( \mathcal{O}_t \). Let \( \mathcal{X} = \{ x(o) \in \mathbb{R}^d | o \in \mathcal{O}_t \} \) denote the set of points corre-
Fig. 1. Voronoi diagram of Example 4.1.

Responding to the current selection $O_ℓ$. The bounding region can, e.g., be chosen to be a bounding rectangle:

$$\mathcal{U} = \{ \mathbf{x} \in \mathbb{R}^d | l_i \leq x_i \leq u_i, i = 1, \ldots, d \}$$

(5)

where the lower and upper bounds $l_i, u_i$, can be determined, e.g., based on prior knowledge of the application domain. Then, the probing locations can be defined as the local maxima\(^2\) of the function $f$ that expresses the distance of a point $\mathbf{x} \in \mathcal{U}$ from the closest object in the current selection:

$$f(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{X}} |\mathbf{x} - \mathbf{y}|$$

(6)

As shown in Theorem 4.3 below, an effective procedure for determining probing locations when $\mathcal{U}$ is a bounded polyhedron exploits the fact that the local maxima of $f(\mathbf{x})$ lie in a subset of the vertices of the bounded Voronoi diagram $\text{Vor}(\mathcal{X}, \mathcal{U})$ [de Berg et al. 2008] of the points $\mathcal{X}$ corresponding to the current selection $O_ℓ$.

$\text{Vor}(\mathcal{X}, \mathcal{U})$ is obtained by restricting the conventional Voronoi diagram $\text{Vor}(\mathcal{X})$ to the region $\mathcal{U}$, and its construction proceeds as follows. Let $\mathcal{C}_i$ denote the cell of $\text{Vor}(\mathcal{X})$ that corresponds to the point $\mathbf{x}_i$; let also $h(\mathbf{x}_i, \mathbf{x}_j)$ denote a half-space, which contains the points that are closer to $\mathbf{x}_i$ than to $\mathbf{x}_j$:

$$h(\mathbf{x}_i, \mathbf{x}_j) = \{ \mathbf{x} \in \mathbb{R}^d | \delta(\mathbf{x}, \mathbf{x}_i) < \delta(\mathbf{x}, \mathbf{x}_j) \}$$

(7)

\(^2\)A real-valued function $f(\mathbf{x})$ is said to have a local maximum at the point $\mathbf{x}^*$ if there exists some $\epsilon > 0$ such that $f(\mathbf{x}^*) \geq f(\mathbf{x})$ whenever $|\mathbf{x}^* - \mathbf{x}|_2 \leq \epsilon$. 

Fig. 2. Bounded Voronoi diagram with shading indicating distance from closest point (Example 4.2).

Hence, by the definition of Voronoi diagram:

$$C_i = \bigcap_{1 \leq j \leq \ell, j \neq i} h(x_i, x_j)$$  \hspace{1cm} (8)

Thus $C_i$ is the intersection of $\ell - 1$ half-spaces and, hence, a (possibly unbounded) open convex polyhedral region bounded by at most $\ell - 1$ faces. Let $P_i$ be $C_i$'s bounded restriction:

$$P_i = C_i \cap U$$  \hspace{1cm} (9)

$P_i$ is a bounded polyhedron, because it is the intersection between two polyhedra, one of which is bounded. Note that $P_i$ might differ from $C_i$, also when $C_i$ is bounded. Indeed, $C_i$ might potentially extend outside $U$. The bounded Voronoi diagram $\text{Vor}(X, U)$ is represented by the set of $\ell$ cells $P_i$, $i = 1, \ldots, \ell$.

**Example 4.1.** Figure 1 illustrates an example when $d = 2$ and $\ell = 5$ objects have already been determined within a bounding rectangle $U$. The corresponding points $x_1, \ldots, x_5$ (shown as discs whose sizes are proportional to the scores) define the Voronoi diagram $\text{Vor}(X)$. Note that $C_1$ is bounded, whereas all the other cells are unbounded.

With a slight abuse of terminology, we also use $\text{Vor}(X, U)$ to indicate the faces, edges and vertices of the Voronoi tessellation. Let $v_u$, $u = 1, \ldots, V$, denote a vertex of $\text{Vor}(X, U)$, and $e_{u,v}$ the edge connecting $v_u$ and $v_v$. The vertices may be of three kinds:

- Original vertices of the bounding region $U$.
- Original vertices of $\text{Vor}(X)$. Some of these vertices might be discarded, as they do not belong to $U$. 
Intersections between the edges of \( \text{Vor}(\mathcal{X}) \) and the boundary of \( \mathcal{U} \). Intersections might occur when an edge connects two vertices, one of which is outside \( \mathcal{U} \) (also including the case of the vertex at infinity).

**Example 4.2.** The bounded Voronoi diagram \( \text{Vor}(\mathcal{X}, \mathcal{U}) \) corresponding to the Voronoi diagram of Figure 1 is represented in Figure 2. Vertices \( v_1 \), \( v_2 \), \( v_3 \) and \( v_4 \) correspond to the original vertices of \( \mathcal{U} \). Out of the four vertices of \( \text{Vor}(\mathcal{X}) \), only two are retained (i.e., \( v_5 \) and \( v_6 \)), as the other ones are outside \( \mathcal{U} \). The remaining vertices (\( v_7 \) to \( v_{12} \)) are due to intersections between \( \text{Vor}(\mathcal{X}) \) and the edges of \( \mathcal{U} \). The shading indicates the distance from the closest point in \( \mathcal{X} \), where brighter indicates larger distance. Such a distance is maximized at the vertices, as confirmed by Theorem 4.3, below.

**Theorem 4.3.** Let \( x^* \in \mathcal{U} \) denote a local maximum of (6). Then, \( x^* \) is a vertex of \( \text{Vor}(\mathcal{X}, \mathcal{U}) \).

**Proof.** We prove the theorem by contradiction. Let us assume that \( x^* \) is not a vertex of \( \text{Vor}(\mathcal{X}, \mathcal{U}) \). Hence, there are two cases to be considered. 1) \( x^* \) is an interior point of one of the cells, e.g., \( P_i \). Therefore, the point in \( \mathcal{X} \) closest to \( x^* \) is \( x_i \). Consider the unit norm vector \( d \) corresponding to the direction of the ray originating in \( x_i \) going through \( x^* \). When moving along such a ray starting from \( x^* \), the distance from \( x_i \) increases in the neighborhood of \( x^* \), i.e., there exists \( \epsilon \) such that \( f(x^* + \epsilon d) > f(x^*) \). Then \( x^* \) cannot be a local maximum. Contradiction. 2) \( x^* \) is an interior point of an edge \( e \) separating two cells. Function \( f(x) \) is a continuous function defined for \( x \in e \), and it is also convex in \( e \), since: i) a point is a convex set; ii) the distance function from a convex set is convex; iii) the restriction of a convex function to a convex domain
ALGORITHM 3: PSPP.chooseAccessMethod()

Input (global variables from caller): bounding region \( U \);
selection \( O_K \) from the objects enclosed in \( U \);
set \( P \) of retrieved objects; discarded region \( D \); last score \( S^{last}_q \)

Output: Access method \( m \)

1. \( v_1, \ldots, v_V := \) vertices of \( \text{Vor}(X, U) \), where \( X = \{x(o) \in \mathbb{R}^d | o \in O_K \} \);
2. \text{if} (PSPP.byScoreIsPreferred()) \( m.\text{probLocation} := \text{null} \);
3. \text{else} \( m.\text{probLocation} := \text{PSPP.chooseProbLocation()} \);
4. \text{return} \( m \);

A tight upper bound on the attainable diversity weighted score can be found as follows

\[
\tau = (1 - \lambda)S^{last}_q + \lambda \max_{x \in Z} \min_{y \in X} |x - y|, \quad (10)
\]

which is a potential outcome induced by an object having the maximum score possible \( S^{last}_q \) and lying at the maximal minimum distance from the objects in the current selection \( O_l \).

Equation (10) in principle requires evaluating all points of the non-discarded region \( Z \). However, Theorem 4.4 provides an effective procedure for computing (10), which requires evaluating the upper bound only in a few selected points.
Theorem 4.4. The point $x^* \in Z$ that maximizes the minimum distance from all the points in $X$ is a vertex of the convex hull of $P_i \setminus D$, where $P_i$ is one of the cells of Vor($X, U$).

In order to prove Theorem 4.4, we introduce three lemmas.

Lemma 4.5. Given a line segment $e$ and a point $x \in \mathbb{R}^d$, the point $x^*$ of $e$ at maximum distance from $x$, is one of the endpoints.

Proof. Let $x'$ be the intersection between $e$ (or its continuation) and the straight line orthogonal to $e$ passing through $x$. Let $x_1$ and $x_2$ be the two endpoints of $e$. If $x^* = x_1$ and $x^*$ were a point of $e$ on the segment connecting $x_1$ and $x'$, then $\delta(x', x^*) < \delta(x', x_1)$. Furthermore, $\delta(x, x^*) = \sqrt{\delta(x, x')^2 + \delta(x', x^*)^2}$ and $\delta(x, x_1) = \sqrt{\delta(x, x')^2 + \delta(x', x_1)^2}$, therefore $\delta(x, x^*) < \delta(x, x_1)$. A similar argument applies to the segment connecting $x_2$ and $x'$. Therefore $x^*$ must be one of the endpoints. \hfill $\Box$

Lemma 4.6. Given a closed bounded polyhedron $C$ and a point $x \in C$, the point $y^*$ of $C$, at maximum distance from $x$, is one of the vertices of $C$.

Proof. Assume by contradiction that $y^*$ is an interior point of $C$. Call $y'$ the point at the intersection between the boundary of $C$ and the ray departing from $x$ to $y^*$. Clearly, $\delta(x, y') > \delta(x, y^*)$. Contradiction. Then $y^*$ must lie on the boundary of $C$. However, the boundary of $C$ is a set of segments, therefore, by Lemma 4.5, $y^*$ is necessarily one of the vertices of $C$. \hfill $\Box$

Lemma 4.7. Given a closed bounded polyhedron $C$, a set $D$ of hyperspheres, and a point $x \in C$, the point $y^*$ of $Z = C \setminus D$ at maximum distance from $x$ is one of the vertices of the convex hull of $Z$.

Proof. Let $Z'$ be the convex hull of $Z$. By definition of convex hull, $Z'$ is a closed bounded polyhedron. Therefore, by Lemma 4.6, the point $y'^*$ in $Z'$ at maximum distance from $x$ is necessarily a vertex of $Z'$. Moreover $i)$ $Z'$ contains all the points in $Z$, and $ii)$ the vertices of $Z'$ are also points in $Z$, therefore $y'^*$ coincides with $y^*$. \hfill $\Box$

Now we are able to prove Theorem 4.4.

Proof. Since $x^* \in Z = U \setminus D$, it necessarily lies in (at least) one closed cell $\overline{P}_i$, where $\overline{P}_i$ indicates the closure of cell $P_i$ of Vor($X, U$). Therefore $x_i$ is the point in $X$ closest to $x^*$ (possibly tied with other points, if $x^*$ lies in the intersection of more than one closed cell). By Lemma 4.7, since $\overline{P}_i$ is a closed bounded polyhedron, $D$ is a set of hyperspheres, and $x_i \in \overline{P}_i$, the point $x^*$ within $\overline{P}_i \setminus D$ that maximizes $\delta(x^*, x_i)$ must be a vertex of the convex hull of $\overline{P}_i \setminus D$, which is the same as the convex hull of $P_i \setminus D$. \hfill $\Box$

Thanks to Theorem 4.4, the value of the upper bound $\tau$ as of (10) can be computed by enumerating the cells of Vor($X, U$) and, for each cell diminished by $D$, by considering the vertices of its convex hull.

For illustration, we now detail a procedure for $d = 2$, based on the fact that, when $d = 2$, the point $x^*$ of Theorem 4.4 can only be $i)$ one of the vertices of Vor($X, U$), $ii)$ one of the intersections between the edges of Vor($X, U$) and the circumference of $\Sigma_u$, $u = 1, \ldots, V$, or $iii)$ one of the intersections between two such circumferences. Based on this observation, instead of enumerating the cells of Vor($X, U$), we equivalently enumerate each vertex $v_n$ of Vor($X, U$), and find the intersections of the circumference of $\Sigma_u$ with the edges or other circumferences.
Let $v_y$.

For each vertex $v_y$ of $\text{Vor}(\mathcal{X}, \mathcal{U})$ we can compute a local threshold $\tau_y$ that only considers the intersections due to the edges outgoing from $v_y$. The global upper bound is then given by

$$\tau = \max_{u=1, \ldots, V} \tau_u$$

(11)

Let $v_{uy_i}, i = 1, \ldots, p_u$ denote the $p_u$ vertices that are connected to $v_y$ through such edges. The $p_u$ intersection points $y_{uy_i}, i = 1, \ldots, p_u$, between the edges and the circumference $\Sigma_u$ are

$$y_{uy_i} = v_y + r_u \frac{v_{uy_i} - v_y}{|v_{uy_i} - v_y|}, \quad i = 1, \ldots, p_u$$

(12)

For each $y_{uy_i}$, the distance from the closest point in $\mathcal{X}$ is

$$\delta_{uy_i} = \min_{x \in \mathcal{X}} \delta(y_{uy_i}, x), \quad i = 1, \ldots, p_u$$

(13)

The local threshold $\tau_u$ is then given by

$$\tau_u = (1 - \lambda)S_{q}^{last} + \lambda \max_{i=1, \ldots, p_u} \delta_{uy_i}$$

(14)

**Example 4.8.** With reference to Figure 3, let us consider $v_5$. We have $p_5 = 3$ vertices (namely, $v_6$, $v_8$, and $v_9$) connected to $v_5$ through edges. The circumference $\Sigma_5$ centered in $v_5$ intersects such edges in three points: $y_{51}$, $y_{52}$, and $y_{53}$. By Theorem 4.4, we know that among these points we may find the point in $\mathcal{Z}$ at maximum minimum distance from all the points in $\mathcal{X}$, since $y_{51}$, $y_{52}$, and $y_{53}$ all lie on the convex hull of some cell of $\text{Vor}((\mathcal{X}, \mathcal{U}))$ diminished by $\mathcal{D}$. For illustration, Figure 4 zooms in on cell $\mathcal{C}_2$ associated with $x_2$ and shows the convex hull of $\mathcal{C}_2 \setminus \mathcal{D}$ in black, where $y_{51}$ and $y_{52}$ lie.

When the circumferences do intersect, computing $\tau_u$ requires considering these intersection points as well.

The appropriateness of the bounding scheme is expressed by the following result.

**Theorem 4.9.** The bounding scheme (10) is tight.

**Proof.** In order to verify Definition 3.3 for any given bounded diversification problem $I = (\mathcal{O}, S_q, \delta, K, F, \mathcal{U})$ and state $\gamma = (\mathcal{O}_\ell, P, D, S_q^{last})$, it suffices to exhibit a continuation $I' = (\mathcal{O}', S_q, \delta, K, F, \mathcal{U})$ of $I$ that includes an object $o$ in $\mathcal{O}' \setminus P$ such that $\sigma(o; \mathcal{O}_\ell)$ equals the bound given by (11). To do that, let $\mathcal{X} = \{x(o) \in \mathbb{R}^d | o \in \mathcal{O}_\ell\}$ denote the set of points corresponding to $\mathcal{O}_\ell$. Let $x^*$ be the point in $\mathcal{Z} = \mathcal{U} \setminus \mathcal{D}$ indicated in Theorem 4.4 that maximizes the minimum distance from all the points in $\mathcal{X}$, computed by Equation (10). It suffices then to extend $\mathcal{O}_\ell$, with an object $o$ with score $S_q^{last}$ and feature...
vector $x'$. With this, i) the value of $\sigma(x; O_i)$ equals by construction the bound (10), and ii) the requirements posed by distance-based access are met, since $x'$ belongs to $Z = U \setminus D$. □

4.3. Pulling strategy

Theorem 4.3 shows that the bounded Voronoi diagram $\text{Vor}(X, U)$ can be exploited to determine the candidate probing locations to use for distance-based access. Then, the pulling strategy must determine how to fetch the next object, and how to choose between distance-based and score-based access (when available), as decided by the function $\text{byScoreIsPreferred}$ of Algorithm 3. When distance-based access is selected, the pulling strategy also chooses a probing location among the candidates $v_1, \ldots, v_V$ determined according to the enumeration induced by Theorem 4.4 (this is done by the function $\text{chooseProbLocation}$).

In principle, the pulling strategy can be as simple as a round-robin (RR) scheduling, whereby distance- and score-based access are alternated, and all the candidate probing locations are uniformly explored. Tightness of the bounding scheme and a RR strategy are sufficient to guarantee a form of instance optimality, as shown by Theorem 4.10.

Let $A^{\text{Vor}}$ be the class of $\text{MMR}$-correct, deterministic bounded diversification algorithms complying with the $\text{PBMMR}$ template that select at each step exactly the same object as $\text{MMR}$ and implement distance-based access using the vertices of $\text{Vor}(X, U)$ as probing locations.

**Theorem 4.10.** Algorithm 2 with tight bounding scheme (10) and a RR pulling strategy is instance-optimal wrt. $A^{\text{Vor}}$.

**Proof.** The claim follows from tightness of (10) as in Theorem 5.1 of [Schnaitter and Polyzotis 2008].

First note that the total number of probing locations used by the algorithm is bounded by a constant that depends on $K$ but not on $N$. For example, when $d = 2$, a (non-bounded) Voronoi diagram $\text{Vor}(X)$ where $|X| = K$ has at most $2K - 5$ vertices and $3K - 6$ edges. The bounding region may cause more vertices due to intersections with the edges; therefore up to $2(3K - 6)$ vertices may be added. This indicates that, during the entire execution of the algorithm, there are at most $c = 2K - 5 + 2(3K - 6) = 8K - 12$ probing locations, plus possibly one object source with score-based access.

From now on the proof proceeds similarly to the proof of Theorem 5.1 of [Schnaitter and Polyzotis 2008], which can be seamlessly adapted to our case by establishing a correspondence between a probing location and a relation. We report the proof for convenience.

Let $I$ be an arbitrary bounded diversification problem. Let us also indicate with $F$ Algorithm 2 with tight bounding scheme (10) and an RR pulling strategy, and with $A$ an arbitrary $\text{MMR}$-correct, deterministic bounded diversification algorithm using the vertices of $\text{Vor}(X, U)$ as probing locations. For each probing location $v_i$ (with $0 \leq i \leq V \leq c$), we define $p^F_i = \text{depth}(F, I, i)$ and $p^A_i = \text{depth}(A, I, i)$. We also define $p^{F, \text{max}} = \max_i p^F_i$ and $p^{A, \text{max}} = \max_i p^A_i$. We want to show that $p^{F, \text{max}} \leq p^{A, \text{max}}$. Assume for contradiction that $p^{F, \text{max}} > p^{A, \text{max}}$, or in other words that $F$ does not halt after reading $p^{A, \text{max}}$ complete rounds. At this point, $F$ has seen at least the objects seen by $A$, so it must have buffered the $K$ results returned by $A$. Let $\sigma^{\text{term}}$ denote the diversity-weighted score of the last object selected by $A$. Since $F$ does not return, we know the bound $\tau$ is greater than $\sigma^{\text{term}}$. Since the bounding scheme of $F$ is tight, there is a continuation $I'$ of $I$ in which an unseen object $o'$ has diversity-weighted score $\sigma' > \sigma^{\text{term}}$. When $A$ is executed on $I'$, it will not select $o'$ because it must behave the same as it did for $I$. This is a contradiction because any $\text{MMR}$-correct, deterministic bounded diversification algorithm should have
selected \( \sigma \) before an object with diversity-weighted score \( \sigma^\text{term} \). The contradiction tells us that \( p^{F}_{\text{max}} \leq p^{A}_{\text{max}} \), and it follows that

\[
\text{sumDepths}(F, I) \leq c \cdot p^{F}_{\text{max}} \leq c \cdot p^{A}_{\text{max}} \leq c \cdot \text{sumDepths}(A, I),
\]

indicating the instance optimality of \( F \), since \( c \) does not depend on \( N \).

The algorithms characterized by Theorem 4.10 sample the candidate probing locations uniformly. In order to further decrease \( \text{sumDepths} \), a heuristic selection of the probing location to use next can be employed. This approach yields a potential adaptive (PA) pulling strategy, which lowers the upper bound \( \tau \) more quickly, thus selecting the next object to be added to the current selection earlier.

PA selects as the next probing location the one that is most likely to reduce the upper bound \( \tau \). Therefore, when a distance-based access is to be made, the chosen probing location is the one that maximizes the local threshold \( \tau_v \) computed as in (11), i.e., the point \( v_v^* \), such that \( u^* = \arg \max_{u=1,...,V} \tau_u \). Geometrically, \( v_v^* \) is such that an intersection of its hypersphere \( \Sigma_u \) with an edge of \( \text{Vor}(X, U) \) or with another hypersphere is the point of \( Z \) at maximum minimum distance from all the points in \( X \). With reference to Figure 3, \( v_5 \) would, e.g., be chosen if \( y_{v_5} \) was the point at maximum minimum distance from \( x_1, \ldots, x_5 \). Conventionally, ties are broken in favor of the probing location with the least depth, then the one with the least index in 1, \ldots, \( V \). This strategy is implemented in PSPP.chooseProbLocation.

**Theorem 4.11.** Let \( A^{RR} \) and \( A^{PA} \) be algorithms in \( A^{Var} \) using tight bounding scheme (10) with the RR and the PA pulling strategies, respectively. Then \( \text{sumDepths}(A^{PA}, I) \leq \text{sumDepths}(A^{RR}, I) \) for all bounded diversification problems \( I \).

**Proof.** The proof of this theorem is very similar to the proof of Theorem 4.2 in [Schnaitter and Polyzotis 2008]. There, the authors show that their potential adaptive version of the PBRJ template with a corner bound (PBRJ*) always terminates with a depth less than or equal to the depth of the round robin execution (PBRJc c), for every input relation. Their argument seamlessly adapts to our case by replacing i) PBRJc c with \( A^{PA} \), ii) PBRJc c with \( A^{RR} \), and iii) their upper bound \( S(R_i(p_i)) \) with the local threshold \( \tau_v \) of probing location \( v_i \). The proof proceeds by contradiction, assuming an index \( k \) for which \( \text{depth}(A^{PA}, I, k) > \text{depth}(A^{RR}, I, k) \). We report the adapted proof for convenience. Let \( p_i^{RR} = \text{depth}(A^{RR}, I, i) \) and let \( p_i \) be the depth of \( v_i \) when \( A^{PA} \) decides to pull the \( (p_i^{RR} + 1) \)-th object from probing location \( v_i \). For each \( i \), either \( A^{PA} \) has seen all objects from \( v_i \), seen by \( A^{RR} \) or no unseen object past the \( p_i \)-th object from \( v_i \) can participate in the solution. Thus, \( A^{PA} \) has already seen all objects in the result set \( O^{A^{RR}}_i \) of \( A^{RR} \). Since \( A^{PA} \) pulls an object from \( v_{k-1} \), we know that \( \tau_k \geq \tau_i \), for \( i = 1, \ldots, V \). Two cases are possible.

Case 1: \( \tau_k = \tau_i \). In this case, there was a tie, and \( A^{PA} \) breaks it by pulling from \( v_k \). There are two ways the tie may be broken. The first possibility is that \( p_k < p_i \), which means

\[
p_i \geq p_k + 1 = p_k^{RR} + 1 \geq p_i^{RR}.
\]

The other possibility is that \( p_k = p_i \) and \( k \leq i \), which means

\[
p_i = p_k = p_k^{RR} \geq p_i^{RR}
\]

where the last inequality is based on the conventional order in which \( A^{RR} \) uses the probing locations. Now we have observed that \( p_i \geq p_i^{RR} \), so \( A^{PA} \) has seen all the objects extracted from \( v_i \) that \( A^{RR} \) has seen.
Case 2: $\tau_k > \tau_i$. Since $A^{RR}$ terminates without reading more than $p^{RR}_k = p_k$ objects from $v_k$, we know $\sigma^{RR}_{\text{last}} \geq \tau_k > \tau_i$, where $\sigma^{RR}_{\text{last}}$ is the diversity-weighted score of the last object selected by $A^{RR}$ before termination. This means that no unseen object from $v_i$ can contribute to the result. These cases show that $A^{PA}$ has seen every object that participates in the solution $O_K^{RR}$, so it must have buffered $K$ results. Furthermore, for each probing location $v_i$, we have

$$\tau_i \leq \tau_k \leq \sigma^{RR}_{\text{last}},$$

i.e., the diversity-weighted score of the last object selected by $A^{RR}$ is at least $\tau_i$, for every $i$. Since $\max\{\tau_1, \ldots, \tau_V\}$ coincides with our tight bound, the termination condition of $A^{PA}$ is met, so it will halt before reading the $(p^{RR} + 1)$-th object from $v_k$. Contradiction. Now, since $\text{depth}(A^{PA}, I, i) \leq \text{depth}(A^{RR}, I, i)$ for $i = 1, \ldots, V$, the claim follows.

Theorem 4.10 enables the improvement of performance by a heuristic selection among the candidate probing locations, when score-based and distance-based are alternated in a round robin fashion. We now focus on the performance gain that can be attained by the heuristic selection of the access method to use next. Both distance-based and score-based access can be used to reduce the value of the upper bound $\tau$. Hence, we choose the access method that reduces $\tau$ at a faster rate. To this end, we compute the partial derivatives of $\tau$ wrt. the number of fetched objects. Let $n^S$ denote the number of objects retrieved by means of score-based access, and $n_u$, $u = 1, \ldots, V$, the number of objects retrieved by distance-based access from probing location $v_u$. Function $PS_{agg}.byScoreIsPreferred$ chooses a score-based access if $|\frac{\partial \tau}{\partial n^S}| > |\frac{\partial \tau}{\partial n_u}|$, i.e., if score-based access reduces $\tau$ at a faster rate than distance-based access from the selected probing location $v_u$. As discussed next, these partial derivatives can be either computed exactly if the distribution of scores and, respectively, objects is known, or approximated by using, e.g., linear predictors.

For score-based access, we obtain

$$\frac{\partial \tau}{\partial n^S} = \frac{\partial}{\partial n^S} \left\{ \max_{u=1,\ldots,V} \tau_u \right\} = \frac{\partial}{\partial n^S} \left\{ (1 - \lambda)S_q(n^S) + \lambda \delta_{u^*} \right\}$$

$$= (1 - \lambda) \frac{\partial S_q}{\partial n^S}$$

where $S_q(n^S)$ is the score of the $n^S$-th object in descending order of score. Thus, $\frac{\partial S_q}{\partial n^S}$ expresses the decay rate of the relevance score, which can be computed if the score distribution is given. Alternatively, one might adopt the approximation $\frac{\partial S_q}{\partial n^S} \approx S_q(n^S) - S_q(n^S - 1)$.

Conversely, for the case of distance-based access, we get

$$\frac{\partial \tau}{\partial n_u} = \frac{\partial}{\partial n_u} \left\{ \max_{v=1,\ldots,V} \tau_v \right\} = \frac{\partial}{\partial n_u} \left\{ (1 - \lambda)S_q(n^S) + \lambda \delta_{u^*} \right\}$$

$$= \begin{cases} 0 & u \neq u^*; \\ \frac{\partial \delta_{u^*}}{\partial n_u} & u = u^*. \end{cases}$$

(16)

Now, $\frac{\partial \delta_{u^*}}{\partial n_u}$ can be factored out as $\frac{\partial \delta_{u^*}}{\partial n_u} = \frac{\partial \delta_{u^*}}{\partial r_{u^*}} \frac{\partial r_{u^*}}{\partial n_u}$, where $\frac{\partial \delta_{u^*}}{\partial r_{u^*}}$ expresses how fast the maximum distance from the closest point in $X$ decreases as we increase the radius $r_u$. 

A:18
That is
\[
\frac{\partial \delta_u}{\partial r_u} = \frac{\partial}{\partial r_u} \max_{i=1,\ldots,p_u} \delta_{u_i} = \frac{\partial}{\partial r_u} \min_{x \in X} \|y_{u,i}(r_u) - x\|
\]
\[
= \frac{\partial}{\partial r_u} \sqrt{\|\hat{x}_{i^*} - \hat{x}_{i^*} \|^2 + (\|\hat{x}_{i^*} - v_u\| - r_u)^2}
\]
\[
= \sqrt{\|\hat{x}_{i^*} - \hat{x}_{i^*} \|^2 + (\|\hat{x}_{i^*} - v_u\| - r_u)^2}
\]
\[
\quad \quad \left(17\right)
\]
where \( \hat{x}_{i^*} = \arg\min_{x \in X} \|y_{u,i}(r_u) - x\| \) and \( \hat{x}_{i^*} \) is the orthogonal projection of \( \hat{x}_{i^*} \) on the edge connecting \( v_u \) to \( v_{u,i^*} \).

The term \( \frac{\partial r_u}{\partial n_u} \) expresses, instead, the rate of increase of the search radius \( r_u \) as the number of fetched objects increases. Assuming uniformly distributed points, i.e., \( n_u = \rho \pi r_u^2 \), where \( \rho \) denotes the density per unit area, we obtain
\[
\frac{\partial r_u}{\partial n_u} = \frac{\partial}{\partial n_u} \sqrt{\frac{n_u}{\rho \pi}} = \frac{1}{2\sqrt{\rho \pi n_u}}
\]
\[
\left(18\right)
\]
Alternatively, if the point distribution is unknown, one can approximate \( \frac{\partial r_u}{\partial n_u} \) as \( \frac{\partial r_u}{\partial n_u} \approx r_u(n_u) - r_u(n_u - 1) \).

To conclude, we define SPP as the instance of Algorithm 2 that uses the tight bounding scheme (10), the PA pulling strategy, and the criterion for selecting the access method indicated in Algorithm 3.

4.4. Walkthrough example
In order to exemplify all the steps discussed so far, we walk through an execution of SPP on a toy example. Let us consider the dataset \( \mathcal{O} = \{o_1, o_2, o_3, o_4\} \). The corresponding feature vectors \( x_1, x_2, x_3, x_4 \) are shown in Figure 5(a). These objects have scores \([1, 0.9, 0.8, 0.7]\), respectively. The parameters used in this example are: \( K = 2, \lambda = 0.75 \).
Initially, SPP performs an access by score for initializing the result set $O_K$ (line (1) of Algorithm 2), by retrieving the object $o_1$, i.e., the object with the highest score, so that $O_K = \{o_1\}$. The object is also added to the set of retrieved objects $P$ (line (2)); the discarded region $D$ is initialized to $\varnothing$, and the highest possible score $S^\text{last}_q$ to 1.

Since $K = 2$, there will be exactly one iteration of the loop at line 3. Initially, the threshold $\tau$ is set to $\infty$ and the best diversity weighted score $\sigma^*$ to $-\infty$ (line 4). Since $P$ and $O_K$ coincide (line 5), no current best object $o^*$ is set.

Now the algorithm enters the loop at line 6 to update $\tau$ and $\sigma^*$. To this end, it will access objects either by score or by distance according to the Potential Adaptive pulling strategy (line (7)), which is further detailed in Algorithm 3 for SPP. Such a strategy requires to build the bounded Voronoi diagram having $o_1$ as centroid; as shown in Figure 5(b), the diagram coincides with the bounding region. Since no distance-based access has been made yet, and therefore score-based and distance-based access cannot be compared as discussed in Section 4.3 for function $\text{PSPP}$.byScoreIsPreferred, distance-based access is selected by default. A threshold $\tau_u$ is computed for each vertex $v_u, u = 1, \ldots, 4$ of the Voronoi diagram, as shown in Equation (14). This results in: $\tau_1 = 0.35, \tau_2 = 0.92, \tau_3 = 0.92, \tau_4 = 1.20$. The vertex associated with the highest threshold value $\tau_u$ is selected for an access (in our example, $v_4$). Consequently, the object $o_3$ is retrieved (line (8)), and the set of retrieved objects $P$ is thus updated: $P = \{o_1, o_3\}$ (line (9)). The diversity weighted score is then computed for $o_3$ (line (10)): $\sigma(o_3; O_K) = 0.98$, and, since $\sigma(o_3; O_K) > \sigma^*$, we have $o^* = o_3$ and $\sigma^* = 0.98$. The discarded region $D$ and the score of the last retrieved object $S^\text{last}_q$ are updated (line (11)).

The vertex associated with the highest threshold value $\tau_u$ is again $v_4$. Now that distance-based accesses have been made, we may choose between distance and score. Having made only one score-based access so far, the decay rate of the relevance score is computed by performing a second score-based access; this results in $\frac{\partial \sigma}{\partial u_{\text{score}}} = -0.1$ (see Equation (15)). The rate of increase of the search radius $r_u$ is instead $\frac{\partial r}{\partial u_{\text{score}}} = -0.16$ (see Equation (16)). Consequently, we have $\left| \frac{\partial \sigma}{\partial u_{\text{score}}} \right| < \left| \frac{\partial r}{\partial u_{\text{score}}} \right|$ and thus a distance-based access is again the most convenient access method. Thus, the object $o_2$ is retrieved. The new object is added to the set of retrieved objects: $P = \{o_1, o_3, o_2\}$. Then, its diversity weighted score is computed: $\sigma(o_2; O_K) = 0.84$. The current best choice $o^*$ is unchanged, since $\sigma(o_2; O_K) < \sigma^*$. The discarded region $D$ is updated; the result is shown in Figure 5(d). The bound is updated, resulting in: $\tau_1 = 0.35, \tau_2 = 0.92, \tau_3 = 0.92, \tau_4 = 0.94$. Consequently, $\max_{u=1, \ldots, 4} \tau_u = 0.94$ is less than $\sigma^*$, and thus the procedure for accessing new objects is terminated. Eventually, $o^*$ is added to the result set: $O_K = \{o_1, o_3\}$. The result is shown in Figure 5(e).

5. Batched Access

The pulling strategy presented in Section 4.3 assumes that distance-based access is invoked via an interface that returns the nearest neighbors to a probing location $v_u$ one by one. This allowed us to show the non-obvious fact that diversification with sorted access methods can achieve, with theoretical guarantees, the same quality as the baseline algorithm ($\text{MMR}$), which accesses and reorders all the relevant objects. When distance-based access is used from a probing location $v_u$, the search radius $r_u$ increases as more objects are accessed, but it is not directly controlled. Yet, in some

practical system (e.g., the Gowalla APIs), the available interface also allows specifying the radius $r_u$ of the search, returning all objects at a distance less than $r_u$ from $v_u$. Implementing a one-by-one access by issuing multiple requests at increasing values of the radius might be costly. A more effective solution is to perform, for each of the $K$ iterations, at most one request per probing location with an optimal choice of the radius.

In the following, we therefore employ the notion of batched access, i.e., an access that, given a probing location $v$ and a radius $r$, returns all the objects found at a distance at most $r$ from $v$. In order to examine the impact of batched access on performance, we introduce a variant of SPP, named SPP_{BA} (SPP with batched access), which addresses this case.

In Section 5.1 we formulate the problem of identifying the optimal choice of the radii to be used at each probing location in order to select the next object when only batched access is used, and discuss a concrete solution for the case of objects uniformly distributed over the bounding region. In Section 5.2, we solve the problem when both batched and score-based access are available, and describe the pseudocode of SPP_{BA} in detail. In Section 5.3, we show how the solution of the problem initially presented in Section 5.1 can be found much more efficiently based on geometrical considerations on the problem at hand. Finally, in Section 5.4 we discuss how to handle batched access when the objects are not uniformly distributed over the bounding region.

5.1. Finding the optimal radii at each probing location

In this section we formalize the problem of computing the optimal choice of the radii for the queries issued at each probing location, with the goal of maximizing the chances of identifying a good object to be added to the current selection. This is achieved by minimizing an upper bound on the diversity-weighted score of unseen objects, expressing the optimization problem in the radius variables $r_1, \ldots, r_V$.

With reference to Figure 6, let $y_{u_i}(r_u)$ be the intersection point between the edge $e_{v_u,v_{u_i}}$ connecting $v_u$ and $v_{u_i}$, and the circumference centered in $v_u$ with radius $r_u$. Furthermore, let $x_{u,i}$ be either of the two centroids that identify edge $e_{v_u,v_{u_i}}$ in the Voronoi diagram, let $x_{u,i}^L$ be its orthogonal projection on $e_{v_u,v_{u_i}}$, and let $\delta_{u,i}(r_u)$ represent the distance between $x_{u,i}$ and $y_{u_i}(r_u)$ (indicated by a dashed red line in Figure 6).
The goal is to minimize the value of an upper bound $\tau_B$ representing the maximum distance to all the objects in the current selection $O_I$ of an arbitrary point in the non-discarded region $Z = \mathcal{U} \setminus \mathcal{D}$, which is obtained from Equations (11) and (14) by letting $\lambda = 1$ (since only batched access is used). Here, each probing location contributes a local threshold used to determine the global upper bound, which depends only on the distances $\delta_{u_i}(r_u)$, i.e.,

$$
\tau_B(r_1, \ldots, r_V) = \max_{u=1, \ldots, V} \left( \max_{i=1, \ldots, p_u} \delta_{u_i}(r_u) \right)
$$

The minimization problem is subject to a constraint on the maximum total number $M_B$ of objects retrieved by batched access:

$$
\begin{align*}
\text{minimize} & \quad \tau_B(r_1, \ldots, r_V) \\
\text{subject to} & \quad \sum_{u=1}^{V} n_u(r_u) \leq M_B \land r_u \leq r_u^{\max}
\end{align*}
$$

where $n_u(r_u)$ is the number of objects expected to be found at a distance up to $r_u$ from $v_u$, and $r_u^{\max}$ is the maximum radius of the hypersphere beyond which the threshold $\tau_B$ stops decreasing. An analytic expression for $r_u^{\max}$ is given in Section 5.3. The value of $M_B$ is chosen based on a tradeoff between the cost of batched access in terms of $\text{sumDepths}$ and the quality of the result $O_K$ (given by $F(O_K)$), both growing as $M_B$ increases. The expected cost of batched access equals $M_B K$, since we retrieve, on average, $M_B$ objects for each of the $K$ iterations of the algorithm. The choice of $M_B$ is further explored experimentally in Section 6.

When the distribution of objects in the space is known, this problem can be conveniently reformulated. In the following, we illustrate the case of uniform distribution, which is common and allows a simple estimate of the number of accessed objects. This can be done for other distributions as well, as shown in Section 5.4. With a uniform density of $\rho$ objects per unit area, we have $n_u(r_u) = \rho \pi r_u^2$ and therefore Problem (20) is equivalent to:

$$
\begin{align*}
\text{minimize} & \quad z \\
\text{subject to} & \quad \rho \pi \sum_{u=1}^{V} r_u^2 \leq M_B \land r_u \leq r_u^{\max} \\
& \quad \left\| x_{u,i} - x_{u,i}^* \right\|^2 + \left( \left\| x_{u,i}^* - v_u \right\| - r_u \right)^2 \leq z
\end{align*}
$$

where we expanded $\tau_B$, introduced the dummy variable $z$, and squared the values of the distances, i.e.,

$$
\begin{align*}
\delta_{u_i}^2(r_u) &= \left\| x_{u,i} - x_{u,i}^* \right\|^2 + \left\| x_{u,i}^* - y_u(r_u) \right\|^2 \\
&= \left\| x_{u,i} - x_{u,i}^* \right\|^2 + \left( \left\| x_{u,i}^* - v_u \right\| - r_u \right)^2
\end{align*}
$$

The obtained Problem (21) is equivalent to the initial Problem (20), as $\delta$ is nonnegative, and the square function is monotonically increasing for nonnegative values. In particular, Problem (21) is a Quadratically Constrained Quadratic Problem in $V+1$ optimization variables, solvable using off-the-shelf solvers such as CVX (http://cvxr.com/cvx/).

As soon as a solution for Problem (21) is found, and thus the optimal radii for the various probing locations are determined, the corresponding batched accesses are made. For each of the $K$ iterations, the retrieved objects are stored in a buffer and kept for the subsequent iterations. Among the overall set of retrieved objects in the buffer, we select the one with the largest diversity weighted score.

### 5.2. Handling score-based access along with batched access

Problem (20) is limited to the case in which the set $O$ is accessed using only batched access. If score-based access is available too, the problem statement has to be modified...
in order to include both access methods. The overall threshold consists of two parts. The former is related to score-based access and the latter to batched access:

$$\tau(r_1, \ldots, r_V, n^S) = (1 - \lambda)\tau_S(n^S) + \lambda \tau_B(r_1, \ldots, r_V)$$

where $n^S$ is the total number of accesses by score. In this case, the budget should also include the amount $M_S$ of accesses by score, and thus it can be expressed as $M = M_S + M_B$. Consequently, Problem (20) is reformulated as follows:

$$\begin{align*}
\text{minimize} & \quad \tau(r_1, \ldots, r_V, n^S) \\
\text{subject to} & \quad \sum_{u=1}^{V} n_u(r_u) + (n^S - \bar{n}^S) \leq M \land r_u \leq r_u^{\max}, \quad u = 1, \ldots, V
\end{align*}$$

where $\bar{n}^S$ represents the accesses by score performed before the current iteration. The component $\tau_S(n^S)$ is minimized when the budget is used entirely, i.e., $M_S$ objects are retrieved by score-based access. If the score distribution is known, $\tau_S(n^S)$ can be expressed analytically, e.g., as $1 - \alpha n^S$ for scores varying linearly in $n^S$.

According to these considerations, the expression for the threshold $\tau$ can be reformulated as follows:

$$\tau = (1 - \lambda)(1 - \alpha n^S) + \lambda \max_{u = 1, \ldots, V} \max_{i = 1, \ldots, p_u} \delta_{u,i}(r_u)$$

For each value of $M_B$ in $0, \ldots, M$, first Problem (20) is solved, thus obtaining a value for $\tau_B(r_1, \ldots, r_V)$; then the value of $M_S$ corresponding to $M_B$ is computed as $M_S = M - M_B$. For that, the threshold $\tau_S(n^S)$ for score-based access can be expressed as follows:

$$\tau_S(n^S) = 1 - \alpha n^S = 1 - \alpha(\bar{n}^S + M_S)$$

The overall threshold $\tau$ is computed for each value of $M_B$. The lowest value of the threshold $\tau$ is eventually chosen; the corresponding radii $r_1, \ldots, r_V$ and number of accesses by score $n^S$ represent the best choice for the strategy comprising both batched access and score-based access. The pseudocode for $\text{SPP}_{BA}$ illustrating the steps described so far is shown in Algorithm 4.

### 5.3. Solving the Batched Access problem efficiently

Problem (20) introduces a large number of constraints, as can be seen in the expanded version (21), where the number of constraints equals $\sum_{u=1}^{V} p_u + V + 1$, i.e., it is dominated by the number of edges in the Voronoi diagram. Even if the problem is addressed using an off-the-shelf solver, obtaining the solution is computationally expensive. However, most of the constraints can be shown to be redundant, and therefore can be removed from the formulation of the problem without affecting its solution.

When solving Problem (20), the value of $r_u$ is upper bounded by

$$r_u^{\max} = \max_{i = 1, \ldots, p_u} r_u^{\max}$$

where

$$r_u^{\max} = \|\bar{x}_{u,i} - \bar{v}_u\|$$

Consider again Figure 6. For each value of the radius $r_u$ in $[0, r_u^{\max}]$, a function $\delta_{u,i}(r_u)$ indicating the distance between the centroid $\bar{x}_{u,i}$ and the intersection $y_{u,i}(r_u)$ is computed. The distance function $\delta_{u,i}(r_u)$ decreases as the radius $r_u$ increases, since the intersection point $y_{u,i}(r_u)$ moves toward the point $\bar{x}_{u,i}$. The maximum value of the distance function is the same for all $p_u$ outgoing edges and is attained when $r_u = 0$. Indeed, $\delta_{u,i}^{\max} = \delta_{u,i}(0) = \|\bar{x}_{u,i} - \bar{v}_u\|$ is the same for each edge, due to the fact that all the centroids that determined the outgoing edges of $\bar{v}_u$ are at the same distance from
\textbf{ALGORITHM 4}: Pseudocode of $\text{SPP}_{\text{BA}}$ handling both batched access and score-based access.

\textbf{Input}: result size $K$; bounding region $\mathcal{U}$; budget $M$

\textbf{Output}: Selection $\mathcal{O}_K$ from the objects enclosed in $\mathcal{U}$

\textbf{Parameters}: Initialization Strategy IS

\begin{enumerate}
\item $\mathcal{O}_K := \{\text{IS.initialObject}\}$
\item $P := \mathcal{O}_K$; $\bar{n}^S := 0$
\item \textbf{while} $(|\mathcal{O}_K| < K)$
\item \hspace{1em} $v_1, \ldots, v_V :=$ vertices of Vor($X, \mathcal{U}$), where $X = \{x(o) \in \mathbb{R}^d | o \in \mathcal{O}_K\}$
\item \hspace{1em} $\tau^* := \infty$
\item \hspace{1em} $\sigma := 0$
\item \hspace{1em} $M_B := 0 \ldots M$
\item \hspace{1em} $\bar{\sigma}_S := \tau_S := 1 - \alpha(\bar{n}^S + M_B)$
\item \hspace{1em} $\tau := (1 - \lambda)\tau_S + \lambda\sigma$
\item \hspace{1em} \textbf{if} ($\tau < \tau^*$) \then $r_1, \ldots, r_v := (r_1, \ldots, r_V)$; $n^S := M_S$
\item $\sigma^* := \arg\max_{o \in P \cup P_B \cup P_S} \sigma(o; \mathcal{O}_K)$
\item $\mathcal{O}_K := \mathcal{O}_K \cup \{\sigma^*\}$
\item \textbf{return} $\mathcal{O}_K$
\end{enumerate}

Each of these constraints is a convex quadratic expression in the radius variable $r_u$, i.e.,

$$
\delta_{u,i}^2 = \|x_{u,i} - \bar{x}_{u,i}\|^2 + (\|x_{u,i} - v_u\| - r_u)^2 = \|x_{u,i} - \bar{x}_{u,i}\|^2 + \|x_{u,i} - v_u\|^2 - 2\|x_{u,i} - v_u\| r_u + r_u^2 = \|v_u - \bar{x}_{u,i}\|^2 + (\|x_{u,i} - v_u\| - r_u)^2
$$

It follows that, for $r_u \geq 0$, $\delta_{u,i}^2 > \delta_{u,i}^* \iff \|x_{u,i} - v_u\| < \|x_{u,i} - v_u\| < r_u^\text{max}$. As a consequence, we can sort the distance functions corresponding to the $p_u$ outgoing edges in increasing order of $r_u^\text{max}$, retaining only the first one. Let $\delta_u$ denote such a function, which is guaranteed to dominate the others for all values of $r_u \geq 0$. In Figure 7, for each vertex, we represent as a solid (respectively, dashed) line the dominating (respectively, dominated) distance functions. Removing the constraints corresponding to dominated distance functions leads to a formulation of Problem (21) with only $2V + 1$ constraints, which is linear in the number of vertices.
Consequently, the Problem (21) can be restated as follows:

\[
\begin{align*}
\text{minimize} & \quad z \\
\text{subject to} & \quad \rho \pi \sum_{u=1}^{V} r_u^2 \leq M_B \land \\
& \qquad r_u \leq r_{u}^{\max} \land \delta_u^2(r_u) \leq z, \quad u = 1, \ldots, V
\end{align*}
\]

The solution of Problem (30) can be efficiently obtained by observing the fact that the value of the objective function \( z \) is non-increasing as the radii \( r_u \) increase, each in its range \([0, r_{u}^{\max}]\). For any given value of \( z \), it is possible to compute the corresponding set of radii \( (r_1, \ldots, r_V) \) by finding the intersection between each of the dominating distance functions and a horizontal line at height \( z \), as illustrated in Figure 8. If there is no intersection, we set \( r_u = 0 \). Once the radii are determined, the expected number of retrieved objects by batched access equals \( M_B = \sum_{u=1}^{V} n_u(r_u) \).

In order to find the optimal solution \( (r_1, \ldots, r_V) \), and thus the corresponding value of the objective function \( z^* \), we perform a dichotomic search on the variable \( z \). Note that \( z^* = \tau_B(r_1, \ldots, r_V)^2 \).

For a given radius configuration \( r_1, \ldots, r_V \), the variable \( z \) is lower-bounded by the highest distance function value \( z = \max_{u=1,\ldots,V} \delta_u^2(r_u) \). Therefore, the optimal radius configuration is the one that minimizes the lower-bound for \( z \), i.e., \( z_{\min} = \max_{a=1,\ldots,V} \min_{r_u=0,\ldots,r_{u}^{\max}} \delta_u^2(r_u) = \max_{u=1,\ldots,V} \delta_u^2(r_{u}^{\max}) \).

The possibility to let radius \( r_{u}^{\hat{u}} \) for vertex \( \hat{u} = \arg\max_{u=1,\ldots,V} \delta_u(r_{u}^{\max}) \) be as close as possible to \( r_{u}^{\max} \) depends on the budget availability, as stated in Problem (30).

The search space for \( z \) can be then delimited in the interval \([z_{\min}, z_{\max}]\), where

\[
\begin{align*}
z_{\max} & = \max_{u=1,\ldots,V} \delta_u^2(0) \\
\text{(31)} \\
\text{and} \quad z_{\min} & = \max_{u=1,\ldots,V} \delta_u^2(r_{u}^{\max}) \\
\text{(32)}
\end{align*}
\]

Equation (31) and (32) state that \( z \) is upper-bounded by the highest \( \delta_u^{\max} \) and lower-bounded by the highest \( \delta_u^{\min} \), respectively.
**ALGORITHM 5:** Pseudocode of the solution of Problem (30)

Input: *Budget* $M_B$

Output: Radii $r$

1. $z_{\text{min}} = \max_{u=1,\ldots,V} \delta_u^2(r_{\text{max}}^u)$;
2. $z_{\text{max}} = \max_{u=1,\ldots,V} \delta_u^2(0)$;
3. $(r_1^*, \ldots, r_V^*) = \text{DichotomicSearch}(z_{\text{min}}, z_{\text{max}})$;

**ALGORITHM 6:** Dichotomic search for the optimal radii $(r_1^*, \ldots, r_V^*)$

Input: *Start* $z_l$, *End* $z_u$

Output: Radii $r$

1. $z = \frac{z_l + z_u}{2}$;
2. Compute $(r_1, \ldots, r_V)$ associated to the value $z$;
3. if $(0 \leq M_B - \bar{M}_B < \epsilon)$ then
   4. return $(r_1, \ldots, r_V)$;
5. if $(M_B > \bar{M}_B)$
   6. $z_l = z$;
7. if $(M_B < \bar{M}_B)$
   8. $z_u = z$;
9. return DichotomicSearch($z_l, z_u$);

Let $z_l$ and $z_u$ denote, respectively, the lower and upper bound of the current interval, which are initialized as $z_l = z_{\text{min}}$ and $z_u = z_{\text{max}}$. If the expected number of retrieved objects $\bar{M}_B$ exceeds the target budget $M_B$, we set $z_l = z$, otherwise $z_u = z$. When $0 \leq M_B - \bar{M}_B < \epsilon$, we terminate the search. Algorithm 5 shows the above-mentioned procedure.

---

**Fig. 8.** The dominating distance functions (one for each vertex) for three vertices. The objective function $z$ is chosen between the values $z_{\text{min}}$ and $z_{\text{max}}$. 

5.4. Handling non-uniform data densities

The problem statement in (21) assumes that the objects in \( \mathcal{O} \) are uniformly distributed in the bounding region \( \mathcal{U} \) and thus their density \( \rho \) is constant and obtained by dividing the total number of available objects by the area of the bounding region.

However, in some practical scenarios this assumption does not apply. The object distribution in the bounding region might be intrinsically non-uniform, as in the case of apartments for sale in a given region, because in some neighborhoods the apartments can be more concentrated than in others. In many cases the object distribution is known, because it is based on the knowledge of the application domain, or it is estimated based on the objects retrieved by previous queries. Hence, Problem (21) can be simply restated, by replacing the constant value of \( \rho \), with the values of the density at each probing location, i.e., \( \rho_v, \; v = 1, \ldots, V \).

Conversely, in the adversarial case in which the object distribution is unknown, their density needs to be somehow estimated. To this end, batched accesses are performed from a set of dummy probing locations \( \mathbf{d}_u, \; u = 1, \ldots, P \), so as to collect an initial set of density values \( \rho(\mathbf{d}_u) \). The dummy probing locations are uniformly distributed on a grid that covers the whole bounding region. In this phase, a radius \( r_d^u \) is chosen for each dummy probing location \( \mathbf{d}_u \). Then, in order to compute the sample density values, we assume that the data density is constant around each probing location, i.e., points are uniformly distributed in the hypersphere of radius \( r_d^u \) centered in the probing location.

Hence, our estimated density can be expressed as

\[
\hat{\rho}(\mathbf{d}_u) = \frac{n_u(r_d^u)}{\pi \cdot (r_d^u)^2} \tag{33}
\]

The choice of the number \( P \) of dummy probing locations and the set of radii \( r_d^u, \; u = 1, \ldots, P \), depend on the tradeoff between the number of accessed objects and the quality of the result of the interpolation. The latter influences the quality of the current selection, as expressed by the objective function in (2). The greater the number \( P \) of dummy probing locations, the higher the quality of the result, but at the cost of an increased number of accessed objects. Choosing larger radii allows a more accurate estimation of the data density, because visiting a wider area results in a better understanding of the underlying data distribution, but also increases the number of accessed objects. A reasonable heuristic approach consists in fixing a constant value \( r_d^u \) for all the radii, because the data distribution is not known at the beginning and all probing locations are equally promising.

Then, the density value \( \rho(\mathbf{x}) \) for each point \( \mathbf{x} \in \mathcal{U} \) can be estimated by means of interpolation, which takes as input a set of known density values in the bounding region \( \mathcal{U} \) and provides an estimate of the density function \( \hat{\rho}(\mathbf{x}) \) in any arbitrary location \( \mathbf{x} \). Therefore, the density value \( \rho_u \) for a given probing location \( \mathbf{v}_u \) can be obtained as:

\[
\rho_u = \hat{\rho}(\mathbf{v}_u).
\]

6. EXPERIMENTAL STUDY

In this section, we characterize the performance of SPP (and of its \( \text{SPP}_{\text{BA}} \) variant), by investigating: i) the I/O cost reduction with respect to the exhaustive access to all relevant objects, as granted by distance-based and/or score-based access; this reduction is measured in terms of the \text{sumDepths} metrics; ii) the CPU time overhead due to the computation of the bounding scheme and of the pulling strategy; iii) the impact on I/O and CPU time of the size of the data set, the number of results, the score distribution, the pulling strategy, and the parameter determining the balance between relevance and diversity; iv) the trade-off between I/O cost reduction and the quality of the diversified result set for \( \text{SPP}_{\text{BA}} \), both with and without score-based access; v) the effect
Table I. Operating parameters (defaults in bold).

<table>
<thead>
<tr>
<th>Full name</th>
<th>Parameter</th>
<th>Tested values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of data set</td>
<td>N</td>
<td>1000, 10000, 100000, 1000000</td>
</tr>
<tr>
<td>Number of results</td>
<td>K</td>
<td>5, 10, 15, 20, 25</td>
</tr>
<tr>
<td>Availability of score-based access</td>
<td>-</td>
<td>no, yes</td>
</tr>
<tr>
<td>Type of score distribution</td>
<td>-</td>
<td>Uniform, Exponential</td>
</tr>
<tr>
<td>Pulling strategy</td>
<td>-</td>
<td>RR, PA</td>
</tr>
<tr>
<td>Balance between relevance and diversity</td>
<td>λ</td>
<td>(0, ..., 0.75, ..., 1)</td>
</tr>
<tr>
<td># of objects to retrieve per iteration (only SPPBa)</td>
<td>M</td>
<td>1, 2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50</td>
</tr>
<tr>
<td>Bounding region aspect ratio</td>
<td>-</td>
<td>1:1, 1:2, 1:3, 1:4</td>
</tr>
</tbody>
</table>

on CPU time of the optimization of radii calculation and the impact of data density estimation in SPPBa.

The tests are performed using 2-dimensional as well as 3-dimensional data sets. The relevant parameters are shown in Table I, with defaults in bold.

6.1. Methodology

**Data sets.** The experimental phase exploits both real and synthetic data sets in two as well as three dimensions – in total, four different families of data sets.

The first family consists of a collection of data sets of 2-dimensional objects synthetically generated by setting the data set size N and the score distribution according to the values in Table I. For each configuration of the parameters, we generate 10 sets of N objects in order to compute the average performance. Each object o is assigned a random score $S_q(o)$, sampled from the chosen distribution (uniform, between 0 and 1, or exponential, with mean 1 and normalized in the [0, 1] range), and a feature vector $x(o)$. The feature vectors are obtained by sampling a uniform distribution over a square region $U$ with area 1.

The second family consists of data sets of 3-dimensional objects synthetically obtained as in the 2-dimensional case. Again, we generate 10 sets of N objects in order to compute average performance. Each object o comes with a random score $S_q(o)$ (sampled from a uniform distribution) and a 3-dimensional feature vector $x(o)$ (sampled from a uniform distribution over a cubic region $U$ with volume 1).

The third family contains real data sets retrieved by using a public web service. Several web service APIs providing score-based, distance-based, and batched accesses are available on the Web, spanning several application domains. Concrete services can, e.g., be found in local event retrieval systems, such as ArtBeat (http://www.nyartbeat.com/resources/doc/api), Bandsintown (http://www.bandsintown.com/api/overview), and Eventful (http://api.eventful.com). Other examples include real estate properties retrieval systems, such as Nestoria APIs (http://www.nestoria.com), AgentStorm (https://www.stormrets.com/docs/api_getting_started.html), Idealista (http://www.idealista.com/labs/api.htm?action=help), and OnboardInformatics-listings (http://www.onboardinformatics.com/web-services/listings-search-engine/). In this work, the data set is built out of real estate properties in Rome, Paris and London, fetched from the Nestoria APIs. Our solution has been implemented as an extension of a larger project (Search Computing project4) that aims to address multi-domain queries. A common interface is provided for accessing registered Web services, including Nestoria. Specifically, when the Nestoria APIs are used to fetch objects, a large number of calls is, in general, required in order to retrieve the whole data set, since the interface allows us to retrieve objects in batches of pages. Each call is in the form of a REST query. In these calls, the user selects, among all the available parameters,

---

4http://www.search-computing.it
the bounding region of the area of interest and a set of parameters describing the property, i.e., the property type, the listing type, the number of rooms, the minimum and maximum price. In the case of batched access, a probing location and a radius can be specified, too. Each data set contains information needed to answer the type of query introduced in Section 1 regarding a search for flats in a city within given price range (e.g., in London between £200,000 and £300,000). For each object \( o \), \( x(o) \) is a 2-dimensional vector whose components are latitude and longitude, normalized so that they lie in the range \([0, 1]\); the score \( S_q(o) \) is either based on price or recentness of the announcement. Note that the mentioned data sources are external and not under our control. Therefore the cost due to building indices for score- or distance-based access is entirely incurred by the data source and not visible at the invoker’s side. Consequently, such a cost is correctly disregarded when computing the \( \text{sumDepths} \) metrics. Moreover, although, internally, a data source might sometimes need to “visit” more objects in order to compute the result of an access, we only see (and therefore include in \( \text{sumDepths} \)) those objects that are actually returned by the access. As for the time, we observed experimentally that an access substantially requires constant time, with slight fluctuations due to network delays.

The fourth family includes two data sets, each consisting of a collection of images fetched with the Flickr APIs (http://www.flickr.com/). The data sets have different sizes and contain, respectively, \( N = 3727 \) and \( N = 119264 \) images. Each image \( o \) in the data sets is associated with a 3-dimensional feature vector \( x(o) \), whose components are defined as the average intensity of color components according to the RGB color model; the score \( S_q(o) \) is defined by linearizing in the range \([0, 1]\) the position of the image in the result set returned by Flickr in response to the keywords used for retrieving the collection, so that the first image has score 1, and the last 0.

All the 2-dimensional data sets are tested against batched access, distance-based access, and score-based access, whereas the 3-dimensional data sets are tested against batched access and score-based access.

**Methods and evaluation metrics.** The results obtained using the different parameter configurations listed in Table I are compared using \( \text{sumDepths} \), which is especially relevant in scenarios where the cost of retrieving objects dominates that of computing the bounding scheme and the pulling strategy, like the case in which search services are invoked over the Web. The \( \text{sumDepths} \) metrics is normalized by the size of the data set \( N \) in order to highlight the percentage of relevant objects accessed to find \( O_K \). Note that the \( \text{MMR} \) algorithm, used as a baseline, retrieves all the \( N \) objects to compute \( O_K \). In order to characterize the behavior of the pulling strategy, we also record the fraction of objects retrieved by score-based access wrt. the total number of accessed objects. The total CPU time is computed in seconds and comprises the computation of the bounding scheme and of the pulling strategy, but it does not include the time needed for fetching the objects, as this is implicitly captured by the \( \text{sumDepths} \) metrics and depends on external, uncontrolled factors. For an end-to-end comparison of the algorithms when accessing real data sources, we have also reported the overall execution time, which includes the time needed to fetch the objects. All the reported results are obtained as the average over ten runs: for the synthetic data sets, each run applies to a data set obtained by varying the random data generation seeds; for the real data sets, each run varies the random set of probing locations used to obtain the initial estimation of the object density function in the region \( \mathcal{U} \).

All the tests have been performed on a computer with the Mac OS X 10.6 operating system, an Intel® Core2-Duo processor at 2.66GHz and 4Gb RAM.

6.2. Results
6.2.1. SPP. Figure 9 shows the results obtained using the 2-dimensional synthetic data sets. In the stacked bar charts representing the normalized \textit{sumDepths} of experiments with both distance-based and score-based access: i) the darker portion of the bar at the bottom represents the objects retrieved by means of distance-based access; ii) the lighter portion of the bar at the top represents the objects retrieved by means of score-based access. In general, SPP reduces the fetched objects to a number significantly smaller than \( N \), at the cost of computing the bounding scheme and the pulling...
strategy, which in the experiments is always less than 1 second and thus of the same order of magnitude as the time for fetching one object from a Web data source.

**Size of data set - \( N \):** For a given value of \( K \), normalized \( \text{sumDepths} \) decreases as the data set size increases. For example, when \( K = 10 \), scores are uniformly distributed and score-based access is not used (Figure 9(a)), 44% of the objects need to be visited for \( N = 1000 \). This value drops to 4% for \( N = 100000 \). In general, the larger the data set, the smaller the fraction of fetched objects. As expected, the CPU time increases with
because it is related to (non-normalized) \( \text{sumDepths} \). This trend is also confirmed for \( N = 1000000 \) (not shown in the figures for readability). In this case, when \( K = 10 \), normalized \( \text{sumDepths} \) drops to 1%.

**Number of results - \( K \):** The \( \text{sumDepths} \) metrics grows linearly with the number of diversified objects \( K \), at a rate that depends on the data set size: the larger is \( N \) the smaller is the rate. The CPU time also shows a similar trend. This behavior is apparent throughout Figures 9-12.

**Impact of score-based access:** Figure 9(a) and Figure 9(c) compare the behavior with and without score-based access. With score-based access, a non-negligible fraction of objects is retrieved using this access method. Overall, normalized \( \text{sumDepths} \) decreases when score-based access is used, because

\( i) \) the upper bound \( \tau \) decreases more rapidly;

\( ii) \) the objects with the largest diversity-weighted score are reported sooner.

The benefit of score-based access is particularly evident for small data sets and large values of \( K \). For example, when \( N = 1000 \) and \( K = 10 \), a 25% reduction of \( \text{sumDepths} \) is observed when score-based access is used.

**Impact of score distribution:** When an exponential score distribution is used to simulate a scenario with non-uniform scores, \( \text{SPP} \) needs score-based access in order to perform well. This is because only a few objects have high scores, and thus, if only distance-based accesses is available, the upper bound \( \tau \) decreases very slowly and it may be necessary to access most objects, even for small values of \( K \). Conversely, the joint use of distance- and score-based access significantly reduces normalized \( \text{sumDepths} \), and as few as 2% of the objects are retrieved when \( N = 100000 \) and \( K = 10 \) (Figure 9(e)).

**Impact of the pulling strategy:** Figure 10(a) and Figure 10(b) show that the accesses made with the RR pulling strategy are almost always more than twice as many as the accesses made by PA. Specifically, we compared the two strategies under different spatial distributions: Figure 10(a) shows results when the data are generated according to a uniform distribution; conversely, in Figure 10(b) data are drawn from a mixture of three different bi-variate Gaussian distributions to simulate a scenario characterized by non-uniform distributions. The means of the distributions are sampled at random within the bounding region and the variance is set equal to 0.05 for all of them.

**Relevance vs. diversity - \( \lambda \):** The solid lines in Figure 10(c) show \( \text{sumDepths}/N \) when \( \lambda \) varies in the \([0, 1]\) interval (if \( \lambda = 1 \), only diversity is considered), with default values for all other parameters. When score-based access is not used, the number of distance-based accesses decreases as \( \lambda \to 1 \), because for small values of \( \lambda \), the emphasis on relevance is best addressed with score-based access. Conversely, for large values of \( \lambda \), only a small fraction of objects is visited, e.g., as few as 0.8%. The availability of score-based access induces a more robust performance across different values of \( \lambda \). The fraction of objects retrieved by score-based access is shown in the dashed line. This result is confirmed when the algorithm is tested against a real data set, i.e., the objects in the Nestoria data set relative to Rome, as shown in Figure 10(d).

**Impact of initialization on \( \text{PBMMR} \):** We tested \( \text{PBMMR} \) with two different initialization strategies, used for setting the initial object in the result set. The first one assumes the availability of score-based access and selects the object with highest score in \( \mathcal{O} \). The second one selects a random object in \( \mathcal{O} \), by choosing a random probing location in \( \mathcal{U} \) and making a distance-based access from there. We measured the value of the objective function with both strategies, with the default values of the parameters listed in Table I and with both uniform and exponential score distributions. We observed, over 10 realizations, that choosing a random object rather than the object with highest score determines an average decrease in the value of the objective function of around 2.5% for uniform distribution, and 3.5% for exponential distribution.
Fig. 11. sumDepths and total CPU time for 2-dimensional real data sets (source: Nestoria). Relevance based on price.

Fig. 12. sumDepths and total CPU time for 2-dimensional real data sets (source: Nestoria). Relevance based on recentness.

**Required CPU time:** Figures 9(b), 9(d), and 9(f) show that, in all tested parameter configurations, the total overhead due to the computation of the bounding scheme and of the pulling strategy increases linearly with $N$ and $K$, but is always below 1 second. When $N = 1000000$, this time goes up to 0.28s when $K = 10$ and reaches a maximum of 1.2s when $K = 25$ (not shown in the figure). In average, these times roughly correspond to the time required to fetch one single object from a data source, thus making our approach beneficial even for very slight improvements in the sumDepths.

**Impact of bounding region aspect ratio:** We tested SPP using rectangular bounding regions all having unit area, but different aspect ratios. Figure 10(e) shows that, in all tested configurations, the normalized sumDepths does not vary significantly.

**Analysis on the real bi-dimensional data sets:** Figures 11 and 12 report the results obtained on the 2-dimensional data sets extracted from Nestoria, with relevance based on price of the flat and, respectively, on recentness of the announcement. We observe that i) the gain due to SPP is more significant for larger data sets (e.g., London); ii) score-based access is seldom chosen (and thus cannot be appreciated in Figures 11 and 12) because the score decay rate is low, especially for the most relevant objects (e.g., many objects have the largest score possible). CPU time is slightly worse when recentness is considered due to a large number of ties in the score (many announcements with the same date).

6.2.2. Batched Access. The behavior of SPP$_{BA}$ has been tested on the 2-dimensional and 3-dimensional synthetic data sets, by evaluating the percentage of visited ob-
jects $\text{sumDepths}/N$, the value of the objective function $F(\mathcal{O}^\text{batched})_K$ obtained when only batched access is used, and the value of the objective function $F(\mathcal{O}^\text{batched, S})_K$ obtained when both batched access and score-based access are exploited. When using default parameters, i.e., with a total number of objects $N = 10000$, Figure 13 plots the values of these indicators against the value of $M$ — the “budget” of SPPBA, i.e., the expected number of retrieved objects per iteration. Since the result set produced by SPPBA, be it $\mathcal{O}^\text{batched}$ or $\mathcal{O}^\text{batched, S}$, in general differs from $\mathcal{O}^\text{SPP}$, Figure 13 shows the reference value of the objective function achieved by MMR on the same data sets, as a dotted grey line.

As visible in Figure 13(a), $\text{sumDepths}/N$ (solid blue line), $F(\mathcal{O}^\text{batched})$ (solid red line), and $F(\mathcal{O}^\text{batched, S})$ (dashed red line) all grow with $M$. When score-based access is used the quality grows faster and reaches $F(\mathcal{O}^\text{SPP})$ already with $M = 25$ and $\text{sumDepths}/N \approx 2\%$. Notice that SPP requires $\text{sumDepths}/N \approx 14\%$ (cf. Figure 9(c) with $K = 10$ and $N = 100000$) to achieve exactly the same result as MMR. A similar result is obtained in the 3-dimensional case illustrated in Figure 13(b), although $F(\mathcal{O}^\text{SPP})$ is reached with $M = 30$ and $\text{sumDepths}/N \approx 3\%$.

**Size of data set for batched access** - $N$: The number of visited objects per iteration $M$ has been set equal to the smallest value among those shown in Table I at which the difference between the values of the objective functions of MMR and SPPBA is less than 5%, determined experimentally as in Figure 13. Specifically, $M$ is set equal to 10, 20 and 30, for $N = 10000$, 100000 and 1000000, respectively, both for the 2-dimensional and the 3-dimensional case. In the 2-dimensional case, Figure 14(a) shows that, for a given value of $K$, normalized $\text{sumDepths}$ decreases as the data set size increases. For instance, for $K = 10$, approximately $8\%$ of the objects are visited for $N = 10000$; this value drops to $0.3\%$ for $N = 100000$. This result is further confirmed for $N = 1000000$ objects (not shown in the figure). In such a case, approximately $0.09\%$ of the objects are visited for $K = 10$, while, when $K = 25$, $0.2\%$ of the objects are visited. The same behavior appears in Figure 14(b) for the 3-dimensional case. In particular, for $K = 10$, $8\%$ of the objects are visited for $N = 10000$, while, for $N = 1000000$, the percentage of accessed objects amounts to only $0.2\%$. Again, the trend is confirmed when $N = 1000000$. In this case, when $K = 10$, SPPBA requires to access approximately $0.09\%$ of the objects. These results compare favorably with SPP: for example, in the 2-dimensional data set with $K = 10$ and $N = 100000$, SPP requires to access approximately $4\%$ of the objects, as shown in Figure 9(c).
Number of results for batched access - $K$: Figure 14 shows that the $sumDepths$ metrics grows linearly with $K$, at a rate that depends on the data set size: the larger the data set size $N$, the smaller the growth rate.

Relevance vs. diversity for batched access - $\lambda$: Figure 15 illustrates $sumDepths/N$ as $\lambda$ varies in the $[0, 1]$ interval, with default values for all other parameters, on the 2-dimensional synthetic data set (Figure 15(a)) and 3-dimensional synthetic data set (Figure 15(b)). The solid lines show $sumDepths/N$ with (in blue) and without (in yellow) score-based access, while the dashed blue line represents the fraction of objects retrieved by score-based access. The experimental values of $sumDepths/N$ conform to the expected number of retrieved objects, which is $(K - 1) \cdot M + 1$, because $M$ is the average number of retrieved objects for each iteration except the first one, in which only one object is fetched by the initialization strategy. The number of accesses by score decreases as $\lambda \to 1$ for both the 2-dimensional and 3-dimensional cases, due to the fact that if $\lambda = 1$ only diversity is considered. A similar analysis is made for the 2-dimensional and 3-dimensional real data sets in Figure 16, which reports how $sumDepths/N$ varies using the real estate data set for Rome (Figure 16(a)) and the small 3-dimensional data set (Figure 16(b)).

Required CPU time for batched access: Figures 17(a) and 17(b) show (for the 2-dimensional and the 3-dimensional case, respectively) a stacked bar chart representing the breakdown of the total CPU time into the time required for $i$) the computation.
Fig. 16. Balance of relevance vs. diversity using Batched Access for real data sets

Fig. 17. CPU time spent using the CVX external solver vs. the optimized solution of Algorithm 5

of the Voronoi diagram (lower part of the bar), and ii) the computation of the optimal radii to use for batched access (upper part of the bar). This allows us to compare a previous version of the algorithm, proposed in [Fraternali et al. 2012], based on an external solver, with the current optimized solution described in Algorithm 5. The figure shows the breakdown of the CPU time needed for each iteration of the outer loop at line 3 in Algorithm 4, which computes the running solution $O_{\ell}$, $\ell = 1, \ldots, K$. The figure reports side by side the results obtained by using two different ways of computing the optimal radii: the left bar represents the CPU time spent when the CVX external solver is used, as described in Section 5.1 (as is done in [Fraternali et al. 2012]); the right bar represents the CPU time spent when the dichotomic search of Algorithm 5 is employed. The computation of the Voronoi diagram requires the same CPU time regardless of the way of computing the radii, and, when $\ell = 10$, it is approximately equal to 0.002s for the 2-dimensional case and to 0.023s for the 3-dimensional case. Conversely, the CPU time needed to compute the optimal values of the radii is significantly reduced when Algorithm 5 is adopted. Indeed, when the external solver is used, the CPU time is approximately equal to 2.4s for the 2-dimensional and 5.9s for the 3-dimensional case. These values drop to, respectively, 0.003s and 0.005s, with the proposed method, thus being negligible.

Figure 18 shows the total CPU time of SPP$_{BA}$ as the size of the data set and result set vary. The overhead increases as $N$ and $K$ increase. When the 3-dimensional data set
is considered (Figure 18(b)), the required amount of time for the execution increases with respect to the 2-dimensional case (Figure 18(a)). This is due to the fact that the number of vertices in the Voronoi diagram increases and so does the CPU time, since the number of constraints in Problem (30) grows (linearly) with the number of vertices. This trend is also confirmed when $N = 1000000$ (not shown in the figures): the required CPU time for the 2-dimensional case is $0.3s$ when $K = 10$, and it reaches a maximum of $0.87s$ with $K = 25$. This result is confirmed also in the 3-dimensional case, where the CPU time is approximately equal to $0.42s$ with $K = 10$.

**Analysis on real data sets for batched access:** Real data sets are often characterized by a non-uniform distribution of the objects in the vector space, so that the density $\rho$ might vary significantly within the bounding region. Thus, it is interesting to investigate the impact on the quality of the result set when this aspect is neglected, emphasizing the need for handling non-uniform data densities, according to the method described in Section 5.4. Indeed, a simplistic approach consists in assuming a constant value $\bar{\rho}$ of the density in the whole bounding region. In this case, it might happen that $\bar{\rho}$ is not equal to the actual density at the probing locations $\rho(v_u)$. If $\bar{\rho} > \rho(v_u)$, the radii computed by solving (20) are smaller than they should be. Hence, fewer objects are retrieved and, consequently, the construction of the result set $O_K$ is performed on a set of objects that is too small and not well diversified. This results in a sensible loss in the quality of the result set with respect to MMR, as shown in Figure 20(a). Conversely, if $\bar{\rho} < \rho(v_u)$, more objects than the planned budget $M$ are retrieved at each iteration. Therefore, the objective function might reach the same value as MMR, but sumDepths increases.

In order to overcome these issues and achieve a good quality of the result set without exceeding the budget, different values of the density should be used for each probing location, as described in Section 5.4. Figure 20(b) shows that an appropriate estimation of the data densities makes the difference between the values of the objective functions of MMR and SPPBA negligible.

Figure 21 provides further results obtained on the Nestoria data sets, when relevance is based on price. Similar results were obtained when relevance is based on recentness. Figure 21(a) shows the value $\text{sumDepths}/N$ when the density is known, as provided by prior knowledge on the application domain. The choice of the radii used by batched access is obtained by setting a target of $M = 15$ objects at each iteration, since we found that this budget enabled to achieve a value of the objective function close to the one attained by MMR. The results are in line with those obtained on synthetic data sets. For example, when $K = 10$, the fraction of retrieved objects varies.
Table II. CPU times for the MMR and SPPBA algorithms on the real data sets

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rome</th>
<th>Paris</th>
<th>London</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMR</td>
<td>0.0315</td>
<td>0.1627</td>
<td>0.3607</td>
</tr>
<tr>
<td>SPPBA</td>
<td>0.0354</td>
<td>0.1318</td>
<td>0.2948</td>
</tr>
</tbody>
</table>

between 3% for Rome and 0.4% for London, i.e., it decreases with the size of the data set. We also considered the challenging case in which the spatial distribution of the objects is completely unknown and thus needs to be estimated on the fly as discussed in Section 5.4. The number of dummy probing locations used for density estimation was set equal to \( P = 25 \). The radius for batched access was chosen in such a way that approximately 0.25% of the total area within the bounding region is explored in the density estimation phase. The same radius was used for all dummy probing locations.

A simple linear interpolation method was adopted to estimate the value of the density in any point within the bounding region.

As expected, estimating the density requires visiting more objects than in the case in which the density is assumed to be known, as can be seen by comparing Figure 21(c) and Figure 21(a). In part, this is due to the bootstrapping phase, which retrieves additional objects. Indeed, in the experiments, the number of additional objects was, respectively, 32, 56 and 111, for Rome, Paris and London. This is indicated by the lower part of the stacked bars in Figure 21(c). In addition, due to errors in the estimated density, the values of \( \text{sumDepths}/N \) reported in the upper part of the stacked bars in Figure 21(c) are not always the same as those in Figure 21(a). The difference is more pronounced in data sets characterized by markedly non-uniform object distributions, as in the case of Rome and Paris. The difference in CPU time between Figure 21(d) and Figure 21(b) is due to the extra computation cost incurred by the density interpolation algorithm.

Figure 19 shows a comparison in terms of overall execution time incurred by both MMR and SPPBA. In particular, the CPU time is reported in the lower part of the stacked bar, while the upper part of the stacked bar shows the access time, i.e., the time needed to download the objects requested during the accesses. The results indicate that i) the CPU time is approximately the same for both MMR and SPPBA and it is totally negligible with respect to the access time (indeed, the lower part of the bars is hardly visible in the figure), and ii) SPPBA is about 25 times faster than MMR in all cases. This is due to the fact that MMR needs to download the entire data set before proceeding with the evaluation of the result set. The available API does not allow us to retrieve the whole data set with a single call. Indeed, data is returned in batches of pages, each one containing a subset of the whole data set. Even by choosing the largest page size available in the API, a large number of pages needs to be accessed to download the entire data set. This is not reasonable for data sets involving thousands of objects or more, especially in scenarios where online queries are performed with handheld devices. The CPU times required by SPPBA and MMR are reported in Table II. Note that, for SPPBA, the CPU time also includes the time needed to estimate the data density (around 40% of the total CPU time used by SPPBA), which is therefore also negligible.

Figure 22 shows the results obtained for the 3-dimensional case using two Flickr data sets of different sizes, when a target of, respectively, \( M = 15 \) and \( M = 30 \) objects is retrieved at each iteration. In the RGB color space, the feature vectors are mostly distributed along the segment connecting the points \([0, 0, 0]^T\) and \([1, 1, 1]^T\). Therefore, the bounding region, which corresponds to the unit cube with edges parallel to the coordinate axes, is almost empty in areas close to the other vertices. For this reason, the retrieved RGB feature vectors are mapped to a rotated coordinate system by means of a linear transformation, which is learnt by means of Principal Component Analysis.
In this way, the bounding region enclosing the objects is more uniformly populated. The location of the probing locations and the corresponding radii are determined in this new coordinate system. Then, they are mapped to the original domain by the inverse rotation, so that batched accesses can be made as usual. Figure 22 confirms the results obtained on the 2D data sets. Indeed, when $K = 10$, only 3.5% of the objects are retrieved in the small data set, and 0.2% in the large data set. The total CPU time, shown in Figure 22(b), grows proportionally to both $N$ and $K$, although it remains within an acceptable range even for very high values of $N$, as it is the case with the large data set.

In summary, the experiments highlight that bounded diversification with batched access can achieve the same quality of the result set as the baseline MMR algorithm, both with synthetic and real data sets. For the synthetic data sets, with a good choice of the budget $M$ of accessed objects per iteration, we could produce a diversified result of $K = 10$ objects by accessing always less than 3% of the objects when $N = 10000$, and less than 0.3% when $N = 100000$. When applied to real data sets characterized by a non-uniform object distribution, the results sets were produced by accessing always less than 5% and, respectively, less than 0.4% of the objects, for the case of small ($< 6000$) and large ($> 60000$) data sets. We also achieved encouraging results in the challenging case in which the object distribution is completely unknown. In this case, the fraction of retrieved objects varied between 2% and 28%, where the additional accesses are mostly due to the errors in the estimated density.
7. RELATED WORK
The work illustrated in this paper is related to previous studies on query result diversification, both in information retrieval and in databases, on spatial diversification in Geographical Information Systems, and on top-k query processing techniques.
**Result Set Diversification.** Diversification is the problem of considering both relevance and diversity in ranking the results of a query, with the aim of constructing a more attractive result set for the user.

The early work [Carbonell and Goldstein 1998] introduced the MMR (Maximum Marginal Relevance) algorithm for addressing the construction of a diversified result set, which implicitly adopts the objective function shown in Equation (2), and uses parameter ($\lambda$) for specifying the trade-off between relevance and diversity.

The more recent work [Gollapudi and Sharma 2009] provides an axiomatic framework for characterizing diversification systems and discusses two optimization objectives and the corresponding algorithms (MaxSum and MaxMin). The authors also show that the optimization problems with the objective functions MaxSum and MaxMin are NP-hard, and derive heuristic algorithms for both cases with a 2-approximation optimality guarantee.

The results of Gollapudi and Sharma are extended in [Borodin et al. 2012], where the MaxSum diversification problem is generalized to the case of submodular set functions for measuring the quality of the result set (instead of just modular, i.e., linear functions as in [Gollapudi and Sharma 2009]). In addition, constraints on the cardinality of the result set (here, $|O_K| = K$), are also generalized to constraints expressed by a general matroid. This allows capturing a different form of diversification, namely on the sources the data come from. In this broader context, a heuristic algorithm is proposed and proved to be 2-approximate. The greedy algorithm discussed in the paper can be regarded as a generalization of the MMR algorithm, thereby providing, according to the authors, theoretical evidence why MMR is a legitimate approach for diversification. This further corroborates our choice of adopting MMR as a baseline for investigating the problem of bounded diversification.

Several existing approaches (surveyed in [Drosou and Pitoura 2010]) rerank relevant results to introduce diversity; they apply to document topical diversification [Agrawal et al. 2009; Bansal et al. 2010; Rafiei et al. 2010; Capannini et al. 2011] and to structured data [Demidova et al. 2010; Liu et al. 2009; Vee et al. 2008]. Unlike SPP, these approaches scan all the $N$ candidate results [Capannini et al. 2011] reducing topical diversification complexity from $O(NK)$ to $O(N\log K)$, where $K$ is the number of query topics. The work [Vee et al. 2008] diversifies online shopping result sets, by solving exactly a set extraction problem for multi-dimensional attribute-based similarity with a technique quasilinear in $K$ that exploits attribute priorities and a tree index over the data set. Diversification in multiple dimensions is addressed in [Dou et al. 2011], where the problem is reduced to MMR by collapsing diversity dimensions in one composite similarity function.

The recent work [Angel and Koudas 2011] examines diversity-aware search under the angle of performance, proposing a novel algorithm (DivGen) that avoids re-ranking all relevant results by adopting a pulling strategy that intelligently alternates five document sequential and random access methods. Our work also addresses performance, modeled with the $\text{sum Depths}$ metrics, but considers diversification in a different scenario, where objects are embedded in a vector space, and exploits the geometry in order to limit the number of accessed objects.

In [Drosou and Pitoura 2012] the authors address continuous dynamic diversification, defined as the problem of extracting the $k$ most relevant and diverse objects from a data set that varies over time, e.g., a continuous stream. The authors employ a MAXMIN diversification model and apply a solution based on the use of cover trees, with theoretical accuracy bounds. Our work is different in that we do not assume an input stream of objects, but rather focus on minimizing the number of queries and of visited objects, and construct the result set by exploring progressively the region of interest. However, dynamic diversification would be a natural extension, if one consid-
ears the region of interest to be time varying, as, e.g., in the case of a mobile user that wants to receive an updated set of relevant and spatially diversified feeds on activities or points of interest in the explored region, a problem that is part of our future work.

Another recent related work is [Qin et al. 2012], which addresses the efficient computation of diversified top-k queries. The authors model relevance of results as a numeric score and similarity as a function. The similarity function is used to construct a diversity graph, where nodes represent results and edges between two nodes denote that the connected nodes have a value of similarity greater than a given threshold (and thus should not appear together in the diversified result set). The top-k diversification problem is formulated as a special case of maximum weight independent set extraction and solved efficiently using two different graph partition approaches, which improve over the baseline A* heuristic algorithm.

This work builds upon [Fraternali et al. 2012], where the authors focus on algorithms for finding exactly the same result set as MMR. In the current article, we expand on the practically significant case of batched access with the SPP algorithm. In particular, our additional findings for this case include i) a radical improvement of the algorithm, which allows obtaining the result set extremely quickly (see Figure 17 for details), ii) an effective heuristics for handling non-uniform data densities, which are commonly encountered in real data sets (see Section 5.4 and Figures 21(c) and 22(a)), iii) an extensive set of experiments conducted on objects that are described by features represented in either two or three-dimensional vector spaces, and iv) an exhaustive set of tests on synthetic data sets.

In summary, the focus of this paper is on algorithms for extracting on the fly the top k relevant and diversified objects from data sources providing sorted access methods by score or distance. This class of algorithms is very general and applies to all those cases where extracting all the relevant objects and then scanning them for diversification is impractical (e.g., mobile queries). We have chosen to compare with MMR, a popular and well-performing diversification algorithm, and found that an on-the-fly algorithm such as SPP can achieve exactly the same result, while avoiding scanning all the objects, and even the more practical batched-access variants reach a very similar quality level with far fewer accessed objects. Yet, many general-purpose diversification algorithms exist [Vieira et al. 2011]. For some of these (e.g., Motley [Jain et al. 2004]), the same formal apparatus as PBMMR is directly applicable. Other algorithms described in [Vieira et al. 2011] (e.g., GMC) fall outside the PBMMR paradigm, because they require knowing all the objects beforehand as to compute the objective function.

**Spatial Diversification.** Spatial diversification was originally introduced by [van Kreveld et al. 2005], with multi-dimensional scattered ranking, a technique for diversifying results that have a term score and a spatial score. Several algorithms considering the distance from the query and from the closest already ranked points are discussed, with complexity results and optimization techniques related both to average processing time and to rank quality. The scattered ranking approach exploits, as in our work, the geometry of the metric space to reduce the number of operations for creating the ranking; however, the proposed algorithms access all the relevant points to extract a balanced ranking from them. The experiment with Mechanical Turk in [Tang and Sanderson 2010] shows that users prefer spatially diversified rankings over undiversified ones. The complementary problem to spatial diversification is the introduction of diversity in spatial nearest neighbor queries. The papers [Jain et al. 2004; Haritsa 2009] formalize such a problem in the case of value-based diversity of geo-referenced database tuples (KNDN: K-Nearest Diverse Neighbor), show its intractability by reducing it to the independent set problem, and discuss the Motley heuristic algorithms for its resolution. It is worth noticing that the Motley algorithm incrementally builds its result set, as MMR, and selects the object with the highest score whose distance to

the already selected objects exceeds a given threshold. As such, it could directly benefit from the use of sorted access and upper bounds as in PBMMR in order to reduce the number of accessed objects.

**Spatial keyword queries.** Spatial keyword queries integrate spatial and textual relevance to address information retrieval needs over geo-located objects. A standard spatial keyword query takes a location and a set of keywords as inputs and returns objects that contain the keywords and are close to the provided location [Felipe et al. 2008]. The recent work [Cao et al. 2012] provides an introduction to spatial keyword queries, classifies the different flavors of standard and mobile queries, and illustrates advanced cases that incorporate result prestige computed by the co-location of places [Cao et al. 2010], queries that return groups of places [Cao et al. 2011], and queries that return regions with specific characteristics, such as object density [Ni and Ravishankar 2007] or topical interest [Liu et al. 2011]. Our work is different from that of spatial keyword queries because our focus is on the uniform coverage of a spatial region by the objects in the result set; however, an interesting point of contact is the extension of bounded diversification to the case of mobile queries, which could benefit from techniques developed for mobile spatial queries.

**Top-k Query Processing.** A crucial issue in all data intensive systems is the ability to address top-k queries (a.k.a. ranking queries) [Fagin 2002], i.e., to retrieve only the best answers to a query without computing the entire result set. The main design dimensions and tools for top-k queries are surveyed in [Ilyas et al. 2008]. Ranking queries usually require an early-out strategy [Ilyas et al. 2003; Schnaitter and Polyzotis 2008] that, based on a convenient use of thresholds, allows stopping the construction of the results and the inspection of the relations involved in the query. Early termination is necessary when accessing the relations is costly, e.g., when data are over the Web. This is exactly the same context as bounded diversification, where early termination is also an essential factor, used for reducing sumDepths. A ranking problem where objects are immersed in a $d$-dimensional space and their mutual distance plays a critical role is the so-called proximity rank join problem [Martinenghi and Tagliasacchi 2010]. It is an extension of the classical rank join [Ilyas et al. 2003] (i.e., the problem of retrieving the top results of a join operation) to objects equipped with a feature vector, with the aim of finding the best combinations of objects with a high score that are close to a given point (the query) and to each other. The technique used in [Martinenghi and Tagliasacchi 2010] is also similarly based on geometry-driven bounds and pulling strategies, but for a different problem (rank join) and geometry than those addressed in this paper.

The techniques for top-k regret minimization queries, such as those discussed in [Nanongkai et al. 2010], could also be relevant to the top-k bounded diversification queries treated in this paper; [Nanongkai et al. 2010] evaluates various algorithms for extracting a subset of n-dimensional objects from the database in such a way that the (formally defined) average disappointment of users due to the unseen objects is minimized. Regret minimization queries assume (unknown) utility functions that are evaluated on single objects; a similar notion of regret could be developed and evaluated for diversification queries, where the similarity function is evaluated on object pairs.

### 8. CONCLUSIONS AND FUTURE WORK

We have addressed the problem of efficiently diversifying the results of top-k queries over spatial objects contained in a bounded region when only sorted access methods based on distance and/or score are permitted. Using MMR as a baseline for assessing the quality of diversification, we have introduced the PBMMR template that characterizes all MMR-correct algorithms. We have shown a tight bounding scheme that gives PBMMR a form of instance optimality.
Our proposal, SPP, instantiates PBMMR with an adaptive pulling strategy that exploits the geometry of the vector space to quickly identify the next top-k object. Our experiments show that SPP is very effective in reducing objects accesses: in the worst experimental scenario with both kinds of access, we only need to access approximately half of the objects, while in a typical one ($K = 10$, $N = 10000$, uniform distribution) only 14% of the objects are accessed, with the gain increasing with $N$ and with skewed data sets. The speed-up in I/O for accessing objects is obtained to the price of a CPU overhead comparable to the time for accessing only one object, while providing the same result as PBMMR, which accesses all the $N$ objects.

The SPP$_{B}$ heuristic further reduces the CPU time overhead by visiting batches of objects at a time. Objects are selected by defining hyperspheres in the vector space, whose radii are computed by solving an optimization problem. Our experiments show that, by choosing a reasonable size of the batch of objects retrieved at each batched access, the required CPU time is below 1 second when $K = 10$, while ensuring the same result quality as PBMMR and accessing only 2% of the objects for the 2-dimensional data set and 3% of the objects for the 3-dimensional data set. Tests were performed on real data, too, showing that the performance of SPP$_{B}$ remains good also when data density is not uniform.

Future work will concentrate on the case in which the bounded region is not fixed, as for example when a mobile user navigates a region and issues repeated queries that have to be diversified over a moving sub-region.

REFERENCES


Top-k Diversity Queries over Bounded Regions


