# Proportionality on Spatial Data with Context

GEORGIOS J. FAKAS and GEORGIOS KALAMATIANOS, Uppsala University

More often than not, spatial objects are associated with some context, in the form of text, descriptive tags (e.g., points of interest, flickr photos), or linked entities in semantic graphs (e.g., Yago2, DBpedia). Hence, location-based retrieval should be extended to consider not only the locations but also the context of the objects, especially when the retrieved objects are too many and the query result is overwhelming. In this article, we study the problem of selecting a subset of the query result, which is the most representative. We argue that objects with similar context and nearby locations should proportionally be represented in the selection. Proportionality dictates the pairwise comparison of all retrieved objects and hence bears a high cost. We propose novel algorithms which greatly reduce the cost of proportional object selection in practice. In addition, we propose pre-processing, pruning, and approximate computation techniques that their combination reduces the computational cost of the algorithms even further. We theoretically analyze the approximation quality of our approaches. Extensive empirical studies on real datasets show that our algorithms are effective and efficient. A user evaluation verifies that proportional selection is more preferable than random selection and selection based on object diversification.

CCS Concepts: • **Information systems → Data management systems**; **Spatial-temporal systems**; **Retrieval models and ranking**; • **Theory of computation → Design and analysis of algorithms;**

Additional Key Words and Phrases: Proportionality, diversity, fairness, keyword search, Ptolemy's spatial diversity, spatial data, ranking

**ACM Reference format:**
Georgios J. Fakas and Georgios Kalamatianos. 2023. Proportionality on Spatial Data with Context. *ACM Trans. Datab. Syst.* 48, 2, Article 4 (May 2023), 37 pages.
https://doi.org/10.1145/3588434

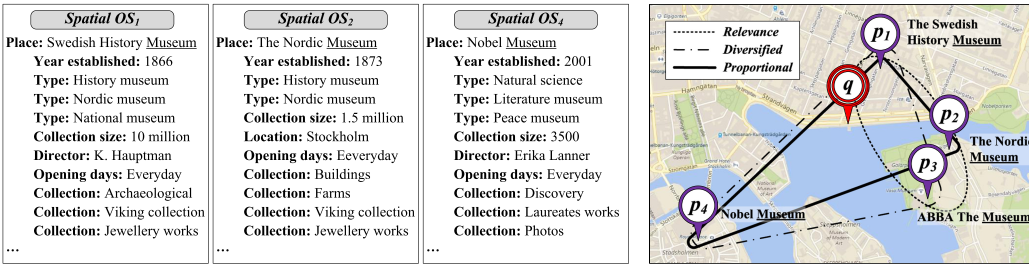
## 1 INTRODUCTION

There is an abundance of public and private datasets which include geo-spatial information exist. For instance, on the web, there are publicly accessible datasets with GIS objects or POIs (e.g., spatialhadoop datasets[1]), datasets with geo-tagged photographs (e.g., flickr), data from online geo-social networks (e.g., Foursquare, Gowalla), semantic knowledge graphs (e.g., YAGO [32], DBpedia), and so on. Acknowledging the significance of discovering datasets and making them

[1] http://spatialhadoop.cs.umn.edu.

This work has received funding from the eSSENCE strategic initiatives on eScience.
Authors' address: G. J. Fakas and G. Kalamatianos, Department of Information Technology, Uppsala University, Lägerhyddsvägen 1, SE-752 37, Uppsala, SWEDEN; emails: {georgios.fakas, georgios.kalamatianos}@it.uu.se.

0362-5915/2023/05-ART4 $15.00
https://doi.org/10.1145/3588434




(a) Spatial Object Summaries (b) Map of Places

Fig. 1. Example of proportionality (querying for *museums* in *Stockholm*).

universally accessible and useful, Google recently introduced *Google Dataset Search*,[2] which facilitates the discovering of web-accessible datasets. Acknowledging also the need for retrieval, various search paradigms have been proposed by the research community. For instance, keyword search paradigms liberate users from technical details such as understanding the nature and structure of the data or a programming language [1, 13, 15, 23, 24, 26, 33, 34, 39, 45, 46, 59].

In this article, we focus on the *location-based* retrieval of spatial entities in datasets. We assume that the spatial objects, besides having a location, are also enriched with some *context*. The context could be either *explicit*, i.e., in the form of descriptive text or tags, or *implicit*, i.e., it could be derived by *linked* neighboring objects in semantic **resource description framework** (**RDF**) graphs. Retrieval models that consider the context of spatial objects, typically combine proximity to a query location and contextual relevance to a set of query keywords [10]. If the context is explicit, popular information retrieval models, such as cosine similarity or tf-idf, can be used to model relevance. Examples of datasets on which such models apply are collections of POIs or geo-located flickr photographs annotated with description tags. If the context is implicit, contextual relevance can be defined by considering the linked entities in subgraphs, which include the query keywords. For instance, the search paradigms of [4, 44] consider minimal subgraphs of nodes that collectively contain the keywords, whereas the object summaries (*OS*s) paradigm [15, 17, 18] considers trees rooted at nodes containing the keywords. Examples of datasets on which such models apply are RDF knowledge graphs (e.g., YAGO, DBpedia) and social networks (e.g., Facebook, Gowalla). It is important to note that, regardless of the type of spatial objects and datasets, contextual similarity between objects can be measured using Jaccard similarity between the corresponding sets of items in their context. Namely, the items can be keywords, tags, dataset nodes, RDF graph nodes.

*Example.* The *OS* paradigm [14–17, 19, 20] summarises information about entities and constitutes an example of implicit context in graphs. A spatial *OS* (*sOS*) is a tree rooted at a spatial entity in a database (i.e., a tuple with a location attribute) or an RDF graph and its context is derived by the set of neighboring important entities (linked either directly or indirectly to the spatial root via foreign key links or RDF predicates). For example, consider a user that wishes to get information about museums in Stockholm from DBpedia (Figure 1). A spatial *OS* will comprise a node representing the "Swedish History Museum" as a root and child nodes including contextual information, e.g., "Nordic museum", "History museum", "Viking collections", and so on (spatial $OS_1$).

Overall, the retrieval goal is finding spatial objects, which are near the query user location and relevant to the query context (e.g., keywords, entities). A *retrieval score* for each query result can be defined by combining spatial distance with contextual relevance (e.g., to query keywords). Still, the query results could be too many and may overwhelm the user. A typical approach is to *rank* the

[2]toolbox.google.com/datasetsearch.

results based on their score and return the top-$k$ objects [10, 44]. However, the most relevant spatial objects could be in the same direction w.r.t. the query location and/or could be too similar to each other in terms of context [12, 37, 48]. For instance, consider a user at location $q$ in Figure 1, who is searching for nearby museums; the top-3 places $p_1$, $p_2$, and $p_3$ are all located in the same direction with respect to the query and have almost similar context (2 out of 3 are history museums).

Several studies reveal that users strongly prefer spatially and contextually diversified query results over un-diversified ones and propose algorithms which select a small number of results, which are not only relevant, but also *spatially and contextually diverse* [50, 57]. Recently, Cai et al. [4] introduced diversification on spatial keyword search by combining relevance and diversity. Namely, the output places, in addition to being relevant to the query, should be diverse w.r.t. their context and location. For instance, a diversified query result for Figure 1 could include $p_1$ (a history museum), $p_3$ (ABBA museum) and $p_4$ (Nobel museum). These places are close to the query and at the same time they are diverse because they are located in different directions w.r.t. the query location and they have quite different context.

Still, simple diversity measures disregard the spatial and contextual distribution of the objects; hence, they may fail to retrieve a *representative* subset of the query results, compromising the quality of the answers given to the user. For instance, consider the example above, we see that 2 out of 4 places are history museums in the same direction w.r.t query location. More precisely, these two places share many common nodes (e.g., common Type and Collection nodes) and are located in the same direction w.r.t. query. This reveals that the general area is dominated by (history) museums located on the right side of the query. Therefore, by representing *proportionally* these properties (at the same time facilitating diversity), we assist users to comprehend the area; diversification fails to reveal such insightful information. Thus, in this article, we study selecting a subset of the query results by combining (1) relevance, (2) spatial proportionality w.r.t. the query location and (3) contextual proportionality w.r.t. the descriptive entities of the objects. In our running example, a proportional result will include $p_1$, $p_2$ and $p_4$; where similar and proportional $p_1$ and $p_2$ places are diverse to $p_4$. Our problem definition and solutions are general and can be applied to any search paradigm where the output is a (ranked) set of spatial entities with (either explicit or implicit) context.

The proportionality problem introduces efficiency challenges as we need to perform pairwise comparison to all retrieved objects, in order to determine the frequent common properties. In view of this, we propose novel efficient algorithms addressing contextual and spatial proportionality. Our contributions can be summarized as follows:

— We introduce the problem of proportionality in location-based retrieval for objects with context and show that it is NP-hard. We also propose novel proportionality measures w.r.t. location and context.
— We propose a generic algorithmic framework which (1) calculates proportionality scores, (2) applies a prepossessing and pruning algorithm (i.e., *P&P*) and (3) adapts existing greedy diversification algorithms (i.e., *IAdU* and *ABP*) [4].
— We propose efficient algorithms for contextual proportionality (i.e., $msJh$ and $apC\mathcal{S}$ algorithms).
— We propose novel efficient algorithms for the calculation of spatial proportionality (i.e., grid based algorithms).
— We analyze the approximation bounds of *IAdU*, *ABP*, $apC\mathcal{S}$ and grid based algorithms.
— We present a thorough evaluation on real datasets demonstrating the efficiency of our algorithms. We conduct a user evaluation verifying that proportional results are more preferable than non-proportional or diversified results.

A preliminary version of this paper that introduces the semantics and algorithms on the proportionality problem on spatial objects appears in [36]. In this article, we introduce new algorithms (*P&P* and *apCS*) that reduce the total time by up to one order of magnitude in expensive cases (Sections 5.2 and 6.2). At the same time, the combination of these two algorithms improves the proportionality quality of the results (by increasing $HPF(\mathcal{R})$ score up to 9%). In addition, we enrich the theoretical analysis of our algorithms by including proofs of approximation bounds for *apCS* and grid based algorithms (Sections 8.2 and 8.3). Finally, we provide a more comprehensive evaluation of our algorithms.

The rest of the article is organized as follows. Section 2 presents related work. Section 3 contains the background work. Sections 4 and 5 formalize our problem and introduce the general framework. Sections 6 and 7 propose efficient contextual and spatial proportionality algorithms. Section 8 provides a theoretical analysis of the approximation bounds of algorithms. Section 9 contains our experimental evaluation. Finally, Section 10 concludes the article.

## 2 RELATED WORK

Our proposed proportional selection framework considers (1) the relevance of the objects to the query (i.e., spatial distance and keywords similarity) (2) contextual proportionality and (3) spatial proportionality w.r.t. the query location. To the best of our knowledge, there is no previous work that considers all these together in proportional selection, as Table 1 shows. Hereby, we discuss and compare related work in diversification and proportionality.

**Diversification.** Diversification of query results has attracted a lot of attention as a method for improving the quality of results by balancing relevance to the query and dissimilarity among results [9, 21, 22, 30, 53]. The motivation is that, in non-diversified search methods, users are overwhelmed with many similar answers with minor differences [37]. PerK [48] and DivQ [12] address the diversification problem in keyword search over relational databases; they use Jaccard distance as a measure of similarity between the keywords in the node-sets that constitute the query results.

**Spatial Diversification.** Several works consider spatial diversification, which selects objects that are well spread in the region of interest [7, 43, 47]. In [29, 35], diversity is defined as a function of the distances between pairs of objects. However, considering only the distance between a pair and disregarding their orientation could be inappropriate. In view of this, van Kreveld et al. [52] incorporate the notion of angular diversity, wherein a maximum objective function controls the size of the angle made by a selected object, the query location, and an unselected object. Recently, Cai et al. [4] combine both spatial and contextual diversity and propose a new measure for spatial diversity (to be described in detail in Section 3).

**Contextual Proportionality.** [8, 11, 49, 55] facilitate proportional diversity by considering *topics* (categories) on items' characteristics and then by proportionally representing these topics. In contrast, our work considers proportionality directly on entities (words, nodes, etc.), which is more dynamic and avoids complications of classifying results in topics (Table 1). In [11] (an early work on this area), an election-based method is proposed to address proportionality. However, this method disregards the relevance of items to the query and thus they may result in picking irrelevant items. In [21, 55], this limitation is addressed by considering relevance in the objective function. Proportionality has also been studied in recommendation systems. For instance, [56] facilitates proportionality by considering topics on both users and items' characteristics. Previous work does not solve the proportionality problem, considering spatial relevance and diversity in space and context.

**Spatial Proportionality** has also been studied on Geographical data. For instance, [28] facilitates proportionality by clustering POIs in sub-regions and then by proportionally recommending

Table 1. Related Work vs. Our Work ([this])

| **Contextual Proportionality** | | **Spatial Proportionality** | | **Relevance** |
|---|---|---|---|---|
| Entities | Topics | Query Location | Regions | |
| **[this]**, [21] | [8, 11, 49, 55] | **[this]** | [28] | **[this]**, [21, 55] |

POIs from these sub-regions. This approach is restrictive since proportionality is based on *static regions* rather than dynamic areas around a query location (which is what we propose); in addition, this approach uses the locations of POIs, but disregards their *context* (Table 1).

**Jaccard Similarity Computation.** Our approach involves Jaccard similarity computations for numerous pairs of (small) sets. Existing work on efficient Jaccard similarity calculation between sets focuses on the scalability w.r.t. both (1) the size of sets and (2) the number of sets. For instance, minhash is an approximation algorithm that detects near duplicate web pages. Many of these algorithms are top-$k$ (or threshold based) and thus are designed to terminate fast by pre-processing sets (e.g., sorting or **locality-sensitive hashing** (**LSH**) [3, 40]). Such a processing can be an effective investment for top-$k$ searches; on the other hand, in our case where we need to compare all pairs, it is an unnecessary overhead. Some algorithms (e.g., minhash) construct signatures in order to speed-up comparisons. Similarly, signatures require preprocessing, which is a useful investment on very large sets; however, for moderate to small sets (as in our case), signatures are not effective and this preprocessing does not pay off. In summary, existing eminent techniques that address scalability in operations that involve Jaccard similarity computations are not appropriate for our problem.

## 3 BACKGROUND

In this section, we describe the type of data that we focus on and how existing methodologies can be used for their retrieval. We also discuss in more detail the spatial diversity we use.

**Spatial objects with context.** We consider a large collection of objects which have spatial locations and some form of context. The spatial locations are described by a set of coordinates and common distance metrics apply on them (e.g., Euclidean distance). The context can be in different forms [27, 38]. Specifically, the context can simply be a set of descriptive keywords or tags. Another type of context could be the set of nodes (or RDF entities), which are linked to the object in a graph. Regardless the form of the context and without loss of generality, we use Jaccard similarity to model the similarity between the contexts of two objects.

**Retrieval of relevant spatial objects and their relevance score.** For a given query, we assume that the result of relevant spatial objects (denoted as $\mathcal{S}$) and the respective relevance score per object (denoted as $rF(p_i)$) are given to us. This renders our methodology more general and thus can be combined with any type of retrieval methods (i.e., $rF(p_i)$ definitions) or type of data (i.e., implicit or explicit). Hereby, we discuss how and with what speed we can achieve $\mathcal{S}$ and $rF(p_i)$ scores.

There is a plethora of existing work, that can facilitate the fast retrieval of spatial objects using spatial-keyword properties. Such methodologies can be used for both pruning the whole population of (infinite) objects (i.e., generation of $\mathcal{S}$) and also for the estimation of respective $rF(p_i)$ scores. For instance, [25] outputs the $K$ nearest objects to a query point where each object covers all query keywords. [10] outputs a list of $K$ objects ranked based on their spatial proximity to the query point and textual similarity to the keywords. For such purposes, spatial-keyword indices are introduced. These indices are usually based on R-Tree and its variants, where each minimum bounding rectangle keeps the textual information of the objects located within its bounds by using inverted files [10] or bitmaps [25]. Such methodologies are very fast. According to studies [6], the retrieval cost is quite low (e.g., it ranges from 5–40 ms in the experiments of Reference [6]). Hence,

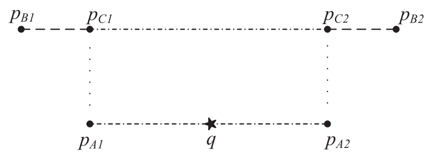



Fig. 2. Ptolemy's spatial diversity ($dS(p_{A1}, p_{A2}) > dS(p_{B1}, p_{B2}) > dS(p_{C1}, p_{C2})$).

our proportional selection problem may be a large factor in the cost of the overall process (i.e., for the $\mathcal{S}$ retrieval and proportional selection times).

**Spatial diversity.** Cai et al. [4] propose Ptolemy's diversity, a new spatial diversity metric, which considers the query location and relative direction of objects from it. Ptolemy's diversity between two places $p_i$ and $p_j$ with respect to a query location $q$ is defined as follows:

$$dS(p_i, p_j) = \frac{||p_i, p_j||}{||p_i, q|| + ||p_j, q||}, \tag{1}$$

where $||p_i, p_j||$ is the Euclidean distance between $p_i$ and $p_j$. $dS(p_i, p_j)$ is naturally normalized to take values in $[0, 1]$, since $||q, p_i|| + ||q, p_j|| \geq ||p_i, p_j||$ (triangle inequality). Two places $p_i$ and $p_j$ receive a maximum diversity score $dS(p_i, p_j) = 1$, if they are diametrically opposite to each other w.r.t. to $q$; e.g., points $p_{A1}$ and $p_{A2}$ in Figure 2. In the same figure, pair of places $(p_{C1}, p_{C2})$ have the same distance as pair $(p_{A1}, p_{A2})$, but $dS(p_{C1}, p_{C2}) < dS(p_{A1}, p_{A2})$, because $p_{C1}$ and $p_{C2}$ are in the same direction w.r.t $q$ (i.e., north of $q$). Pair $(p_{B1}, p_{B2})$ are further from each other compared to the places in pair $(p_{C1}, p_{C2})$ and consequently have a higher diversity score (this can be shown using Pythagorean theorem).

## 4 PROPORTIONAL SELECTION PROBLEM

Consider a query $q$ and its result $\mathcal{S}$, a set of retrieved places. Each place $p_i \in \mathcal{S}$ carries (1) a relevance score $rF(p_i)$ combining the distance to $q$ and potentially other criteria (such as relevance to a set of query keywords [44]), (2) a location, and (3) a context (i.e., a set of contextual items such as keywords, nodes, etc.). Our objective is to find a subset $\mathcal{R}$ of $\mathcal{S}$ that combines a *relevance function* to the query and a *proportionality function* that considers the location and the context of each place. If $K$ and $k$ denote the sizes of $\mathcal{S}$ and $\mathcal{R}$, respectively, then it should be $k < K$. Note that our problem definition is general and is independent from any paradigm used to derive the set $\mathcal{S}$ of retrieved objects. For instance, the places can be geo-textual search results [10], spatial object summaries [15], spatial keyword search results over RDF graphs [44], and so on.

For each place $p_i$ in the retrieved set of places $\mathcal{S}$, we assume that the relevance score $rF(p_i)$ of $p_i$ to the query is known. The exact definition of the relevance function $rF(p_i)$ depends on the retrieval model used; e.g., it could be a linear combination of the Euclidean distance between $p_i$ and the query location $q$ and the relevance of $p_i$'s context to the query keywords [10, 44].

In this section, we first define proportionality with respect to context and location; then, we define a holistic score that tradesoff relevance and proportionality; finally, we define the problem formally. For a place $p_i$, we overload the notation $p_i$ to denote its location and contextual set; we also use $C(p_i)$ to denote the contextual set wherever necessary. Table 2 shows the most frequently used notation in the article.

### 4.1 Proportionality Function

**Contextual proportionality**. We observe in the example of Figure 1 that the places in the retrieved set $\mathcal{S}$ may have common elements in their contexts. For instance, "History museum",

Table 2. Notations

| Notations | Definition |
|---|---|
| $p_i$ | A place ($p_i$ also denotes the location and context of the place) |
| $C(p_i)$ | Set of contextual items of $p_i$ (e.g., keywords or vertices) |
| $\|C(p_i)\|$ or $\|p_i\|$ | Number of elements in contextual set of place $p_i$ |
| $\mathcal{S}$ | A set of $K$ most relevant spatial objects for a given query |
| $\mathcal{R}$ | A subset of $k$ of $\mathcal{S}$ combining relevance and proportionality |
| **General scores** | **Definition** |
| $rF(p_i)$ | Relevance score of $p_i$ w.r.t. $q$ |
| $sC(p_i, p_j)$ | Contextual (Jaccard) similarity |
| $sS(p_i, p_j)$ | Ptolemy's spatial similarity; i.e., $1 - dS(p_i, p_j)$ (Equation (1)) |
| $sF(p_i, p_j)$ | Weighted similarity of $p_i$ and $p_j$ (Equation (13)) |
| **Proportionality scores** | **Definition** |
| $pC\mathcal{S}(p_i)$ | Contextual proportionality of $p_i$ w.r.t. $\mathcal{S}$ (Equation (3)) |
| $apC\mathcal{S}(p_i)$ | Approximated contextual proportionality of $p_i$ w.r.t. $\mathcal{S}$ (Equation (22)) |
| $pC\mathcal{R}(p_i)$ | Contextual proportionality of $p_i$ w.r.t. $\mathcal{R}$ (Equation (4)) |
| $pS\mathcal{S}(p_i)$ | Spatial proportionality of $p_i$ w.r.t. $\mathcal{S}$ (Equation (6)) |
| $pS\mathcal{R}(p_i)$ | Spatial proportionality of $p_i$ w.r.t. $\mathcal{R}$ (Equation (7)) |
| $pF\mathcal{S}(p_i)$ | Weighted summation of $pC\mathcal{S}(p_i)$ and $pS\mathcal{S}(p_i)$ (Equation (11)) |
| $pF\mathcal{R}(p_i)$ | Weighted summation of $pC\mathcal{R}(p_i)$ and $pS\mathcal{R}(p_i)$ (Equation (12)) |
| $pC(p_i)$ | Contextual proportionality score of $p_i$ (Equation (2)) |
| $pS(p_i)$ | Spatial proportionality score of $p_i$ (Equation (5)) |
| $pF(p_i)$ | Combined (contextual and spatial) proportionality of $p_i$ (Equation (8)) |
| $HPF(p_i, p_j)$ | Holistic proportionality between $p_i$ and $p_j$ (Equation (15)) |
| $HPF(\mathcal{R})$ | Holistic proportionality score of $\mathcal{R}$ (Equation (10)) |
| $cHPF(p_i)$ | Proportional contribution of $p_i$ if added to $\mathcal{R}$ (used by IAdU) |

"Nordic museum", "Viking collections", and "Jewelry works" appear in both spatial $OS_1$ and $OS_2$ of $\mathcal{S}$. These contextual elements are representative for the spatial region which includes $OS_1$ and $OS_2$. Therefore, we argue that in the selection of the subset $\mathcal{R}$, we should favor proportionally places that include such frequent contextual elements. At the same time, we argue that results forming $\mathcal{R}$ should be dissimilar as to facilitate diversity. In view of these properties we define the proportional score of a place $p_i$ w.r.t. its context as follows:

$$pC(p_i) = pC\mathcal{S}(p_i) - pC\mathcal{R}(p_i), \tag{2}$$

where

$$pC\mathcal{S}(p_i) = \sum_{p_j \in \mathcal{S}, p_i \neq p_j} sC(p_i, p_j), \tag{3}$$

$$pC\mathcal{R}(p_i) = \sum_{p_j \in \mathcal{R}, p_i \neq p_j} sC(p_i, p_j). \tag{4}$$

Here, $sC(p_i, p_j)$ measures the contextual similarity of two places as the Jaccard similarity between the corresponding sets of elements $C(p_i), C(p_j)$ (e.g., keywords, graph vertices, etc.) in their contexts; i.e., $sC(p_i, p_j) = \frac{|C(p_i) \cap C(p_j)|}{|C(p_i) \cup C(p_j)|}$. $pC\mathcal{S}(p_i)$ aggregates the similarity between $p_i$ and all other places in $\mathcal{S}$. We also define $pC\mathcal{R}(p_i)$ as the similarity of $p_i$ to the rest of places in $\mathcal{R}$. The rationale is that, in our selection, we should penalize $p_i$ if it has large similarity $pC\mathcal{R}(p_i)$ with the rest places in $\mathcal{R}$. Hence, to assess the value of $p_i$ in $R$, we subtract $pC\mathcal{R}(p_i)$ from $pC\mathcal{S}(p_i)$. This is inspired by earlier work in proportionality [11, 21] that follows the same strategy. The proportional score

$pC(p_i)$ of a place ranges in $[0, K-k]$, where $K$ and $k$ denote the amount of elements in $\mathcal{S}$ and $\mathcal{R}$, respectively, since each $sC(p_i, p_j)$ ranges in $[0,1]$.

**Spatial proportionality.** Similarly, we define the proportionality score of a place w.r.t the query location. For instance, in our running example, we observe that the area containing places $p_1$, $p_2$, $p_3$ is a representative neighborhood for the given query (i.e., for both keywords and location), as opposed to the area containing the spatial outlier $p_4$. Therefore, we argue that we should favor proportionally places located in such representative neighborhoods w.r.t. the query location. At the same time, we argue that places should be located in diverse directions w.r.t the query location. In view of these properties, we define the proportionality score of a place as follows:

$$pS(p_i) = pS\mathcal{S}(p_i) - pS\mathcal{R}(p_i), \tag{5}$$

where

$$pS\mathcal{S}(p_i) = \sum_{p_j \in \mathcal{S}, p_i \neq p_j} sS(p_i, p_j), \tag{6}$$

$$pS\mathcal{R}(p_i) = \sum_{p_j \in \mathcal{R}, p_i \neq p_j} sS(p_i, p_j). \tag{7}$$

Here, $sS(p_i, p_j)$ measures the pairwise spatial similarity of two points w.r.t. $q$ by using the complementary of their Ptolemy's spatial diversity (i.e., $sS(p_i, p_j) = 1 - dS(p_i, p_j)$, Equation (1). The rationale of the $pS\mathcal{S}(p_i)$ definition is to favor a place with many neighbors in $\mathcal{S}$ w.r.t $q$. Similarly, $pS\mathcal{R}(p_i)$ favors places spatially diverse to the rest of the places in $\mathcal{R}$. Thus, both $pS\mathcal{S}(p_i)$ and $pS\mathcal{R}(p_i)$ consider the query location $q$. $pS(p_i)$ score also ranges in $[0, K-k]$ Like $pC\mathcal{S}(p_i)$, $pS\mathcal{S}(p_i)$ also requires computing $sS(p_i, p_j)$ for all pairs of places in $\mathcal{S}$. In Section 6, we propose data structures that accelerate these computations.

**Combined scores.** We can combine contextual and spatial proportionality to a *proportionality* score as follows:

$$pF(p_i) = (1-\gamma) \cdot pC(p_i) + \gamma \cdot pS(p_i), \tag{8}$$

where $\gamma \in [0,1]$ controls the relative importance of the two factors. Then, we can combine proportionality and relevance to a holistic score as

$$HPF(p_i) = (1-\lambda) \cdot (K-k) \cdot rF(p_i) + \lambda \cdot pF(p_i), \tag{9}$$

where $\lambda \in [0,1]$ controls the relative importance of relevance and proportionality. We multiply the relevance score $rF(p_i)$ by $K-k$ in order to normalize against $pF(p_i)$ that ranges in $[0, K-k]$. Finally, we can combine these scores for all places in $\mathcal{R}$:

$$HPF(\mathcal{R}) = \sum_{p_i \in \mathcal{R}} HPF(p_i). \tag{10}$$

**Additional useful definitions.** Before we proceed with the problem definition, we also introduce additional definitions that are used throughout the article. First, we introduce weighted ($\gamma$) scores:

$$pF\mathcal{S}(p_i) = (1-\gamma) \cdot pC\mathcal{S}(p_i) + \gamma \cdot pS\mathcal{S}(p_i), \tag{11}$$

$$pF\mathcal{R}(p_i) = (1-\gamma) \cdot pC\mathcal{R}(p_i) + \gamma \cdot pS\mathcal{R}(p_i), \tag{12}$$

$$sF(p_i, p_j) = (1-\gamma) \cdot sC(p_i, p_j) + \gamma \cdot sS(p_i, p_j). \tag{13}$$

$pF\mathcal{S}(p_i)$ (resp. $pF\mathcal{R}(p_i)$) is the combined similarity (contextual and spatial) of $p_i$ and all other places in $\mathcal{S}$ (resp. $\mathcal{R}$), whereas $sF(p_i, p_j)$ is the combined similarity between $p_i$ and $p_j$. Based on the above equations, we can rewrite the definition of the proportionality score $pF(p_i)$ as

$$pF(p_i) = pF\mathcal{S}(p_i) - pF\mathcal{R}(p_i). \tag{14}$$

We also introduce the following pairwise holistic score that can facilitate the heuristics of our greedy algorithms (Section 5):

$$HPF(p_i, p_j) = (1-\lambda)\cdot(K-k)\cdot\frac{rF(p_i)+rF(p_j)}{k-1} + \lambda\cdot\left(\frac{pF\mathcal{S}(p_i)+pF\mathcal{S}(p_j)}{k-1} - 2\cdot sF(p_i,p_j)\right). \tag{15}$$

This score is defined in such a way that the summation of $HPF(p_i, p_j)$ scores of all pairs of places in $\mathcal{R}$ will give us the same score as the summation of $HPF(p_i)$ scores of all places in $\mathcal{R}$ (i.e., $HPF(\mathcal{R}) = \sum_{p_i\in\mathcal{R}} HPF(p_i) = \sum_{p_i,p_j\in\mathcal{R},p_i\neq p_j} HPF(p_i,p_j)$) (Note that Equation (9) can also be defined as $HPF(p_i) = (1-\lambda)\cdot(K-k)\cdot rF(p_i) + \lambda\cdot(pF\mathcal{S}(p_i) - pF\mathcal{R}(p_i))$). Then, the summations of $rF(p_i)$, $pF\mathcal{S}(p_i)$ and $pF\mathcal{R}(p_i)$ for all places in $\mathcal{R}$ are equal to the summations of $\frac{rF(p_i)+rF(p_j)}{k-1}$, $\frac{pF\mathcal{S}(p_i)+pF\mathcal{S}(p_j)}{k-1}$ and $2\cdot sF(p_i,p_j)$ for all pairs in $\mathcal{R}$, respectively, i.e., $\sum_{p_i\in\mathcal{R}} rF(p_i)$, $\sum_{p_i\in\mathcal{R}} pF\mathcal{S}(p_i)$ and $\sum_{p_i\in\mathcal{R}} pF\mathcal{R}(p_i)$, respectively. Thus, we have:

$$HPF(\mathcal{R}) = (1-\lambda)\cdot(K-k)\cdot\sum_{p_i\in\mathcal{R}} rF(p_i) + \lambda\cdot\left(\sum_{p_i\in\mathcal{R}} pF\mathcal{S}(p_i) - \sum_{p_i\in\mathcal{R}} pF\mathcal{R}(p_i)\right). \tag{16}$$

### 4.2 Problem Definition

We define the proportional selection problem as follows.

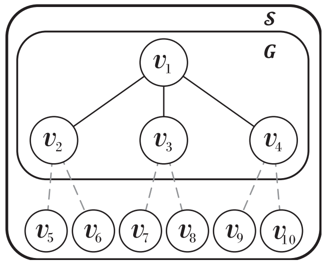
Problem Definition 1. *Given a set of $K$ places $\mathcal{S}$ (where each place carries a relevance score, location and set of contextual items), a query location $q$, and an integer $k < K$, find a set $\mathcal{R}$ of $k$ places that have the highest $HPF(\mathcal{R})$ among all $k$-subsets of $\mathcal{S}$.*

As proven below, this problem is NP-hard; thus, we resort to greedy algorithms for solving it.

Theorem 4.1. *Problem 1 is NP-hard.*

Proof. In order to prove the hardness of our proportionality problem, we construct a reduction from the independent set problem. Given an undirected graph $G(V,E)$ and a positive integer $k$, ($k \leq |V|$), the independent set problem is to decide if $G$ contains an independent set $\mathcal{R}$ of size $k$ (i.e., there is not any edge connecting any pair of nodes in $\mathcal{R}$).

We generate an instance of our problem as follows. Each vertex $v_i$ in $V$ corresponds to a place $p_i$ with a contextual set $C(p_i)$. For every edge $(v_i, v_j)$ in $E$, we add an element $v_{i,j}$ to the contextual sets of both $p_i$ and $p_j$. We now construct the complete set of places $\mathcal{S}$ as follows. First, we add to $\mathcal{S}$ all places that correspond to vertices of $V$. Let $d$ be the maximum degree of any vertex in $V$. For each vertex $v_i \in V$, for which the degree $deg(v_i)$ is less than $d$, we add $d - deg(v_i)$ new places in $\mathcal{S}$ and "connect" them to $v_i$. Namely, for each such new place $p_j$, we add an element $v_{i,j}$ to the contextual sets of both $p_i$ and $p_j$. Finally, we add to the contextual set $C(p_j)$ of each new place $p_j$ $d-1$ elements which are unique to $p_j$ (i.e., no other place has any of these elements in its contextual set). As a result, each $p_i$ corresponding to a vertex in $V$ with a degree less than $d$ will have *exactly one common element* with each of the new places linked to it. In general, all places $p_i$, which correspond to vertices in $V$ will have identical $pC\mathcal{S}(p_i)$ scores because (1) they all have exactly one common element with exactly $d$ places in $\mathcal{S}$ and (2) all places in $\mathcal{S}$ have exactly $d$ elements in their contextual sets. In addition, all places $p_j$ which do not correspond to vertices in $V$ (i.e., all places added later), will have exactly one common element with exactly one place in $\mathcal{S}$. This means that the $pC\mathcal{S}(p_i)$ scores of all $p_i$s corresponding to vertices in $V$ are equal and strictly larger than the $pC\mathcal{S}(p_j)$ scores of all other places $p_j$.




(a) Independent Set

$C(p_1) = \{v_{1,2}, v_{1,3}, v_{1,4}\}$
$C(p_2) = \{v_{1,2}, v_{2,5}, v_{2,6}\}$
$C(p_3) = \{v_{1,3}, v_{3,7}, v_{3,8}\}$
$C(p_4) = \{v_{1,4}, v_{4,9}, v_{4,10}\}$
$C(p_5) = \{v_{2,5}, v_{5a}, v_{5b}\}$
$C(p_6) = \{v_{2,6}, v_{6a}, v_{6b}\}$
$C(p_7) = \{v_{3,7}, v_{7a}, v_{7b}\}$
$C(p_8) = \{v_{3,8}, v_{8a}, v_{8b}\}$
$C(p_9) = \{v_{4,9}, v_{9a}, v_{9b}\}$
$C(p_{10}) = \{v_{4,10}, v_{10a}, v_{10b}\}$

(b) Corresponding Contextual Sets

Fig. 3. Example of reduction.

We can now prove that the $k$-subset $\mathcal{R}$ of $\mathcal{S}$, which maximizes $HPF(\mathcal{R})$ is a $k$ independent set in the original graph $G$. We consider a special case of our problem, where $\lambda = 1$ (i.e., we disregard relevance) and $\gamma = 0$ (i.e., we disregard Ptolemy's diversity). First, all $k$-subsets of $\mathcal{S}$, which include only vertices in $V$ have a common $\sum_{p_i \in \mathcal{R}} pF\mathcal{S}(p_i)$ score (equal to $\sum_{p_i \in \mathcal{R}} pC\mathcal{S}(p_i)$, since $\gamma = 0$), which is higher than the corresponding score of all $k$-subsets, which include some vertex outside $V$. This is because all vertices in such a subset have the maximum possible $pC\mathcal{S}(p_i)$ score (as discussed above). Second, all $k$ independent sets from $V$ correspond to $k$-subsets for which the quantity $\sum_{p_i \in \mathcal{R}} pF\mathcal{R}(p_i)$ is zero. This is because all pairs of places in such a set have no common elements. The reduction takes polynomial time, since the maximum degree of any vertex in $|V|$ is $|V| - 1$, which means that we should add at most $|V| \cdot (|V| - 1)$ edges and vertices. This completes the proof. □

Figure 3 shows an example of the reduction. Consider the graph shown in Figure 3(a), which includes four vertices, such that $v_1$ is connected to all vertices and there are no other edges. A 3-independent set in this graph is $\{v_2, v_3, v_4\}$. For the reduction, we initially define $C(p_1) = \{v_{1,2}, v_{1,3}, v_{1,4}\}$, $C(p_2) = \{v_{1,2}\}$, $C(p_3) = \{v_{1,3}\}$, and $C(p_4) = \{v_{1,4}\}$. Then, for each one of the vertices $\{v_2, v_3, v_4\}$, we connect it to two new vertices, add the corresponding new places to $\mathcal{S}$, and update the corresponding contexts. This results in all four original vertices in $V$ to have the same (maximum) degree 3; hence, all corresponding places have 3 elements in their contexts and any subset with $k = 3$ such vertices have the same (maximum) sum of $pF\mathcal{S}(p_i)$ scores. At the same time, each vertex in the independent set $\mathcal{R} = \{v_2, v_3, v_4\}$ has a zero $pF\mathcal{R}(p_i)$ score. Overall, any $k$ independent set problem can be converted to a special case of our problem for $\lambda = 1$ and $\gamma = 0$.

## 5 GENERIC PROPORTIONALITY ALGORITHMIC FRAMEWORK

Our problem (Definition 1) requires $rF(p_i)$, $pC\mathcal{S}(p_i)$, $pS\mathcal{S}(p_i)$ and $sF(p_i, p_j)$ scores. In contrast to the $rF(p_i)$ score which is given to us, the calculation of $pC\mathcal{S}(p_i)$ and $pS\mathcal{S}(p_i)$ is very challenging as it dictates the comparison of all pairs $(p_i, p_j)$ of places in $\mathcal{S}$ (i.e., a quadratic number of pairs), in order to calculate their $sF(p_i, p_j)$. We propose a three-step algorithmic framework (Figure 4). In step 1, we compute the $pC\mathcal{S}(p_i)$ and $pS\mathcal{S}(p_i)$ scores. In step 2, we use a prepossessing and pruning algorithm. Finally, in step 3, we apply greedy algorithms that find $\mathcal{R}$. As we explain below, our main contribution are the first two steps, since we use previously known greedy algorithms for the third step.

### 5.1 Step 1: Compute Proportionality Scores of $\mathcal{S}$

In this step, we calculate the proportionality scores $pC\mathcal{S}(p_i)$ and $pS\mathcal{S}(p_i)$. As we discuss in the following sections, baseline approaches for calculating sub functions $sC(p_i, p_j)$ and $sS(p_i, p_j)$

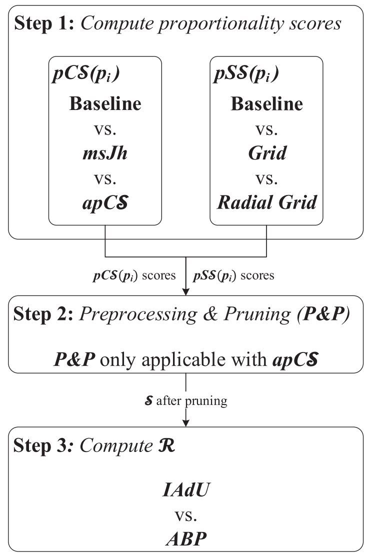



Fig. 4. Generic algorithmic framework.

require up to $|C(p_i)|$ (size of the contextual set) and 20 operations, respectively. Hence, we need a total of $O(K^2 \cdot (|C(p_i)| + 20))$ operations for all pairs of places in $\mathcal{S}$. We introduce tailored algorithms that greatly reduce this complexity in practice (Sections 6 and 7). We also compare them with such baseline approaches [4]. During this step, our algorithms (except from the $apC\mathcal{S}$ algorithm) need also to calculate pairwise scores $sF(p_i, p_j)$ before computing proportionality scores. We cache these scores and reuse them during the execution of our greedy algorithms (as to save time).

Note that the calculation of $sC(p_i, p_j)$ and $pC\mathcal{S}(p_i)$ can be significantly more expensive than the calculation of $sS(p_i, p_j)$ and $pS\mathcal{S}(p_i)$. In view of this, we also propose a very fast $pC\mathcal{S}(p_i)$ calculation algorithm (i.e., $apC\mathcal{S}$), which avoids pairwise scores calculation. In this case, we employ step 2, which prunes fruitless places and during step 3 we only need to calculate pairwise scores for a significantly smaller subset of $\mathcal{S}$.

### 5.2 Step 2: Preprocessing and Pruning of $\mathcal{S}(P\&P)$

During this step, we prune places that cannot make it in $\mathcal{R}$. Such a pruning is very effective for our approximate $apC\mathcal{S}$ algorithm (Section 6.2), which bypasses $sC(p_i, p_j)$ computations and delays them until step 3. More precisely, the $apC\mathcal{S}$ algorithm can calculate very fastly $pC\mathcal{S}(p_i)$ by avoiding $sC(p_i, p_j)$ pairwise comparisons and caching. Recall that the pairwise $sC(p_i, p_j)$ comparisons are among our most expensive computations (requiring also quadratic time). Thus, an effective pruning can reduce expensive fruitless pairwise comparisons.

Given the scores for $rF(p_i)$, $pC\mathcal{S}(p_i)$, and $pS\mathcal{S}(p_i)$, we can estimate useful bounds of $HPF(p_i)$ that can facilitate pruning of places. We can rewrite Equation (9) as follows:

$$HPF(p_i) = (1-\lambda)\cdot(K-k)\cdot rF(p_i)+\lambda\cdot[(1-\gamma)\cdot pC\mathcal{S}(p_i)+\gamma\cdot pS\mathcal{S}(p_i)]-\lambda[(1-\gamma)\cdot pC\mathcal{R}(p_i)+\gamma\cdot pS\mathcal{R}(p_i)]. \tag{17}$$

All components of Equation (17) are known except from the component $HPF_x(p_i) = \lambda[(1-\gamma)\cdot pC\mathcal{R}(p_i) + \gamma \cdot pS\mathcal{R}(p_i)]$. This component ranges in $[\lambda \cdot k, 0]$ facilitating the definition of an upper

**ALGORITHM 1:** Preprocessing and Pruning of $\mathcal{S}$ (*P&P*)

**Input:** Set $\mathcal{S}$, where each $p_i$ in $\mathcal{S}$ carries $rF(p_i)$, $pC\mathcal{S}(p_i)$ and $pS\mathcal{S}(p_i)$ scores
**Output:** Pruned and ranked set $\mathcal{S}$

```
for each p_i in S do
    calculate HPF^ub(p_i) and HPF_lb(p_i) using Equations (18) and (19)
sort(S) on HPF_lb(p_i)
p_k ← S.get(k)
for each p_i in S do
    if HPF^ub(p_i) < HPF_lb(p_k) then
        Remove p_i from S
```

bound $HPF^{ub}(p_i)$ and lower bound $HPF_{lb}(p_i)$ as follows:

$$HPF^{ub}(p_i) = (1-\lambda)\cdot(K-k)\cdot rF(p_i) + \lambda\cdot[(1-\gamma)\cdot pC\mathcal{S}(p_i) + \gamma\cdot pS\mathcal{S}(p_i)], \tag{18}$$

$$HPF_{lb}(p_i) = HPF^{ub}(p_i) - \lambda\cdot k. \tag{19}$$

We can calculate the $HPF^{ub}(p_i)$ and $HPF_{lb}(p_i)$ for all $p_i \in \mathcal{S}$. Then, we sort $\mathcal{S}$ on $HPF_{lb}(p_i)$ and we can easily retrieve $p_k$, the place with the $k$th highest $HPF_{lb}(p_i)$ score. We then compare the scores $HPF^{ub}(p_i)$ for $i \in [k+1, K]$ and prune places for which $HPF^{ub}(p_i) < HPF_{lb}(p_k)$. We formally describe this process in Algorithm 1.

We can see that the known components $HPF^{ub}(p_i)$ and $HPF_{lb}(p_i)$ are bounded by $K-(1-\lambda)\cdot k$ and $K-k$, respectively, whereas the unknown component $HPF_x(p_i)$ is bounded by only $\lambda\cdot k$. Hence, as $K$ increases against $k$, these bounds differences against $HPF_x(p_i)$ also increase and so is the effectiveness of this pruning. More precisely, our experiments have shown that we can achieve pruning of $\mathcal{S}$ up to 90%. This pruning was empirically proved to be very effective only for the case of *apC*$\mathcal{S}$ algorithm (which avoids $sC(p_i, p_j)$ pairwise comparisons). Thus after this pruning, we will need pairwise comparisons only among the non pruned places (which are now approximately only 10% of the total $\mathcal{S}$). We delay $sC(p_i, p_j)$ computations and perform them during the third step. In summary, by combining *apC*$\mathcal{S}$ with pruning and greedy algorithms, we can achieve savings of up to one order of magnitude on the total time. This combination very interestingly has also achieved up to 9% improvement on the holistic $HPF(\mathcal{R})$ score (recall *apC*$\mathcal{S}$ is an approximation). This is because we rank and process places considering their $HPF_{lb}(p_i)$ score. More precisely, during step 3, we feed our greedy algorithms with places in this order. Empirically, our greedy algorithms perform better processing places in $HPF_{lb}(p_i)$ order instead of $rF(p_i)$ order (since $rF(p_i) < HPF_{lb}(p_i)$).

This algorithm has the following time costs. First, we need to calculate the bounds of $HPF(p_i)$ for all places in $\mathcal{S}$, which requires $O(K)$. Then, we need to sort all places which requires $O(K\cdot logK)$. Finally, we need to scan and prune $\mathcal{S}$, which requires $O(K)$ time. This results in the total cost of $O(K + K\cdot logK)$.

### 5.3 Step 3: Compute $\mathcal{R}$

The problem is NP-hard, as we have already shown. We use two alternative greedy algorithms from previous work [4], i.e., *IAdU* and *ABP*. These heuristics have approximation guarantees and have been successfully used by previous works addressing similar problems such as diversification and dispersion [4], [42], and [31]. Hereby, we will focus our description on the heuristics, the respective adaptations and their complexity (efficiency aspects can be found in Reference [4]). In Section 8, we study their approximation bounds.

Both algorithms assume the existence of a ranked set $\mathcal{S}$ of places. In general, we process places ranked on $rF(p_i)$ score. The algorithms also require $sF(p_i, p_j)$ scores which have already been computed and cached during step 1. On the other hand, in case we use the combination of *P&P* and *apC$\mathcal{S}$* algorithms, we process places ranked on $HPF_{lb}(p_i)$. Since the *apC$\mathcal{S}$* algorithm does not provide $sC(p_i, p_j)$ scores, we calculate them here (our experimental times also include this cost).

Although the respective algorithms in Reference [4] employed a dynamic threshold in each iteration as to facilitate an early termination, for our problem definition the threshold was not effective. Our experiments revealed that almost in all cases, the use of a threshold made little difference to the efficiency of the algorithms and thus we avoided using it for simplicity. We only use the pruning of step 2 for the case of *apC$\mathcal{S}$* algorithm.

***IAdU.*** This algorithm iteratively constructs the result set $\mathcal{R}$ by selecting each time the place from $\mathcal{S}$ that maximizes the contribution it can make toward the overall score $HPF(\mathcal{R})$. The contribution $cHPF(p_i)$ of $p_i$ to be added to the current result set $\mathcal{R}$ is defined as follows:

$$cHPF(p_i) = \begin{cases} rF(p_i), & \text{if } \mathcal{R} = \emptyset, \\ \sum_{p_j \in \mathcal{R}} HPF(p_i, p_j), & \text{otherwise.} \end{cases} \tag{20}$$

$cHPF(p_i)$ considers the relevance score and the proportionality of $p_i$ against existing elements in $\mathcal{R}$. In the first iteration, $\mathcal{R}$ is empty, thus the available contribution of a place can only be the corresponding $rF(p_i)$ score. The contributions of all other places are then updated to consider the new entry in $\mathcal{R}$. Then, the algorithm iteratively selects the place $p_i$ that maximizes $cHPF(p_i)$ w.r.t. the current $\mathcal{R}$, adds $p_i$ to $\mathcal{R}$, and updates the contribution of the places not in $\mathcal{R}$. The complexity of the algorithm is $O(K \cdot k \cdot logK + K^2)$. This time includes $K \cdot k$ heap updates (for each place in $\mathcal{S}$) and $K^2$ updates of $HPF(p_i, p_j)$ (for all pairs of places in $\mathcal{S}$).

***ABP.*** This algorithm greedily constructs the result set $\mathcal{R}$ by iteratively selecting the pair of places $(p_i, p_j)$ with the largest $HPF(p_i, p_j)$ score (Equation (15)). *ABP* selects the next pair $(p_i, p_j)$ based on only its $HPF(p_i, p_j)$ value, independently of the relationships of $p_i$ or $p_j$ to places already in $\mathcal{R}$ (in contrast to *IAdU*). Once a pair is selected, both its constituent elements and any pairs they make are removed from further consideration by the algorithm (in a *lazy fashion*). Since a single pair is selected in each iteration, $\lfloor k/2 \rfloor$ iterations apply when the value of $k$ is even. When $k$ is odd, an arbitrary place is chosen to be inserted in the result set $\mathcal{R}$ as its last entity. The worst case complexity of the algorithm is $O(K^2 \cdot \log(K^2))$, which is higher than that of *IAdU*.

## 6 CONTEXTUAL PROPORTIONALITY CALCULATION

$pC\mathcal{S}(p_i)$ scores require the calculation of Jaccard similarity of all pairs of contextual sets of places in $\mathcal{S}$, which can be an expensive process. We propose two novel algorithms, *micro set Jaccard hashing (msJh)* and *apC$\mathcal{S}$*, which are tailored to the characteristics of our sets (i.e., numerous sets of moderate size). Jaccard similarity is a generic measure, appropriate for any type of contextual items (e.g., for sets of keywords, tags, RDF entities, nodes, etc.).

**Baseline approach**. We first discuss a baseline approach for computing the contextual similarities of all pairs of places in $\mathcal{S}$. This approach, for each pair, first creates a hash table with the elements of the first set and then uses it to check for each element in the second set if it appears in the first set. For comparing all pairs in $\mathcal{S}$, we still need to hash all $K$ sets in $\mathcal{S}$. Assume, for simplicity, that all sets have the same size $|p_i|$. The hashing phase costs $O(K \cdot |p_i|)$, as we have to scan all elements from all sets. The comparison phase costs $O(K^2 \cdot |p_i|)$, because for each of the $O(K^2)$ pairs, we need $|p_i|$ checks in the worst case. The baseline approach is expensive if $\mathcal{S}$ contains many places; for instance, for $K = 100$ and $|p_i| = 5$, we need approximately 25,000 operations.

**Minhash** is an eminent technique for the fast calculation of Jaccard similarity on vast amounts of sets of large size. This approach works in two steps. During the first step, we apply $t$ hash

**ALGORITHM 2:** Micro Set Jaccard Hashing (*msJh*)

**Input:** set $\mathcal{S}$<br>
**Output:** (1) $sC(p_i, p_j)$ for all pairs of places in $\mathcal{S}$, (2) $pC\mathcal{S}(p_i)$ for each place in $\mathcal{S}$

1: **for** each $p_i$ in $\mathcal{S}$ **do**<br>
2:     **for** each element $v$ in $p_i$ **do**<br>
3:         **if** $msht[v]$ does not exist **then**<br>
4:             Generate new $msht[v]$ list {Step 1}<br>
5:         Add $p_i$ in the front of $msht[v]$ list<br>
6: **for** each $p_i$ in $\mathcal{S}$ **do**<br>
7:     **for** each element $v$ in $p_i$ **do**<br>
8:         **for** each $p_j$ in $msht[v]$ with $j > i$ **do**<br>
9:             Update Jaccard Score $(p_i, p_j)$ {Step 2}<br>
10:     Update $pC\mathcal{S}(p_i)$

functions (i.e., $K \cdot t$ operations) on each set (where we get $t$ minimum values). During the second step, each pair is compared against the respective $t$ minimum values (i.e., in total $K^2 \cdot t/2$ operations). Thus, in order to compare all pairs, we need in total of $K^2 \cdot t/2 + K \cdot t$ operations. Apparently, this approach can be very efficient when the number of elements ($|p_i|$) in the contextual set of each place $p_i$ is large, as $|p_i|$ does not affect the cost. We implemented this algorithm, in order to compare it with our proposed *msJh* algorithm, but it failed to perform well on our data, where the sets are relatively small.

### 6.1 Micro Set Jaccard Hashing (*msJh*) Algorithm

In view of the limitations of the previous algorithms, we propose the micro set Jaccard hashing (*msJh*) algorithm. The algorithm generates an inverted list for each element with the sets wherein the element appears (i.e., *msht*). The rationale of the *msht* hash table is that we can facilitate a targeted Jaccard comparison. Namely, we facilitate the comparisons of sets only if we know they have common elements (by using *msht*). Our technique is very efficient for small sets and, at the same time computes it exactly (in contrast to minhash and $apC\mathcal{S}$). The algorithm consists of two steps (i.e., Algorithm 2). Figure 5 illustrates an example.

**Step 1: Generate *msht*.** We parse all sets and add on a hash table all elements and the sets wherein they appear (i.e., micro set hash table, denoted as *msht*; lines 1–5). More precisely, for each element, we maintain a reverse list of the sets wherein the element appears (the reverse order of the places in the inverted list facilitates avoidance of redundant checks and we explain this in the following step). Figure 5(b) illustrates the *msht* for the example of Figure 5(a).

**Step 2: Compare sets.** We compare pairs in an economical fashion by utilising *msht*. More precisely, we calculate the intersection of any pair $p_i$ and $p_j$, for pairs with $i < j$ and for each element $v$ in $p_i$ (lines 6–10). For instance, in our example of Figure 5(a), we will process first $p_1$. For each element in $p_1$ (i.e., $\{a, b, c, d\}$), we consult the *msht* as to see in which sets these elements appear. Then, we update the Jaccard (partial) scores accordingly. e.g., $a$ of $p_1$ appears in $p_3$, $p_2$ and $p_1$. Then, we process $b$ of $p_1$, which appears in $p_4$, $p_2$ and $p_1$. Recall that we add elements on *msht* in a reverse order. Thus, we can stop processing an element against sets that have been previously processed or against the set itself. For instance, while processing $p_1$, we will not compare $a$ against $p_1$; also, while comparing $p_2$, we will not compare $a$ against $p_2$ and $p_1$. An illustrative example of the savings of this algorithm (against the baseline algorithm) can be shown in the comparison of $p_3$ and $p_5$. Where, according to *msht*, the two sets have no common elements and this will result in zero operations. On the other hand, the baseline approach will still have to compare these two

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $pCS(p_i)$ |
|---|---|---|---|---|---|---|
| $p_1$:$\{a,b,c,d\}$ | | $\{a,b,c\}$ : 3/5 | $\{a,d\}$ : 2/6 | $\{b,d\}$ : 2/6 | $\{c\}$ : 1/7 | 1.41 |
| $p_2$:$\{a,b,c,e\}$ | | | $\{a,e\}$ : 2/6 | $\{b\}$ : 1/7 | $\{c\}$ : 1/7 | 1.22 |
| $p_3$:$\{a,d,e,f\}$ | | | | $\{d\}$ : 1/7 | $\{\}$ : 0 | 0.81 |
| $p_4$:$\{b,d,g,h\}$ | | | | | $\{\}$ : 0 | 0.62 |
| $p_5$:$\{c,i,j,k\}$ | | | | | | 0.29 |

(a) $pCS$ Calculation

| $v$ | $msHT[v]$ |
|---|---|
| **a** | $p_3, p_2, p_1$ |
| **b** | $p_4, p_2, p_1$ |
| **c** | $p_5, p_2, p_1$ |
| **d** | $p_4, p_3, p_1$ |
| **e** | $p_3, p_2$ |
| **f** | $p_3$ |
| **g** | $p_4$ |
| **h** | $p_4$ |
| **i** | $p_5$ |
| **j** | $p_5$ |
| **k** | $p_5$ |

(b) Micro set hash table ($msht$)

Fig. 5. Example of the $msJh$ algorithm.

sets. Finally, given the intersection of $|p_i|$ and $|p_j|$, we can infer the union by subtracting the size of the intersection from $|p_i| + |p_j|$.

The algorithm has the following time costs. During the first step, we need to create the micro hash table, which requires $O(K \cdot |p_i|)$ time, where $|p_i|$ is the average number of elements in a set in $\mathcal{S}$. During the second step, we build the intersections of all pairs of $p_i$s. Thus, assuming again for simplicity that all sets have common size $|p_i|$, we need $O(K^2 \cdot |p_i|)$ time (i.e., the worst case is when all sets are equal), i.e., the same cost as the baseline approach in the worst case. However, in practice, the pairs of sets will not have high overlap; hence, the algorithm is much faster than the baseline approach, as we verify experimentally.

### 6.2 *apCS*: Approximate Calculation of *pCS* Algorithm

Our previous algorithms, baseline and $msJh$, dictate the calculation of $sC(p_i, p_j)$ for all pairs in order to calculate the $pCS(p_i)$ score, which still requires quadratic time. Our experiments revealed that their costs remain significantly more expensive than the spatial and greedy algorithms costs and dominate the total time. Thus, we propose $apCS$, a linear algorithm that bypasses pairwise comparisons and can very efficiently calculate a high quality approximation of $pCS(p_i)$ (denoted as $apCS(p_i)$). The algorithm's rationale is based on the relaxation of the Jaccard similarity by replacing the normalizing denominator of a specific pair (i.e., $|p_i \cup p_j|$) to a generic normalization denominator, which is common for all pairs (i.e., $|p_i|$ by assuming a common size $|p_i|$ of sets). In this article, we focus on the case of sets with a common size (although our approach can be generalized for variable sizes), which is often a requirement by many applications. For instance, keyword extraction frameworks typically produce a top-$k$ set of keywords [5, 51, 54, 58]. On graphs, object extractions frameworks also typically employ a top-$k$ nodes paradigm (e.g., Size-$l$ object summaries) [15, 21].

More precisely, we propose to use $asC(p_i, p_j) = \frac{|p_i \cap p_j|}{|p_i|}$ as the approximation of $sC(p_i, p_j) = \frac{|p_i \cap p_j|}{|p_i \cup p_j|}$. Thus, we have the approximated $pCS(p_i)$ score as follows:

$$apCS(p_i) = \sum_{p_j \in \mathcal{S}, p_i \neq p_j} asC(p_i, p_j) = \sum_{p_j \in \mathcal{S}, p_i \neq p_j} \frac{|p_i \cap p_j|}{|p_i|}, \tag{21}$$

which can also be defined as

$$apCS(p_i) = \sum_{t_j \in p_i} \frac{c(t_j) - 1}{|p_i|}, \tag{22}$$

**ALGORITHM 3:** *apCS*: Approximated Calculation of *pCS* Algorithm

**Input:** set $\mathcal{S}$
**Output:** $apCS(p_i)$ for each place in $\mathcal{S}$

1: **for** each $p_i$ in $\mathcal{S}$ **do**
2:     **for** each element $v$ in $p_i$ **do**
3:         **if** $cH[v]$ does not exist **then**
4:             Generate new entry $cH[v]$ with $c(v) = 1$ {Step 1}
5:         **else**
6:             $c(v) = c(v) + 1$
7: **for** each $p_i$ in $\mathcal{S}$ **do**
8:     **for** each element $v$ in $p_i$ **do**
9:         $apCS(p_i) = apCS(p_i) + \frac{c(v)-1}{|p_i|}$ {Step 2}

| | $apCS(p_i)$ |
|---|---|
| $p_1$:$\{a, b, c, d\}$ | $(2+2+2+2)/4 = 2.00$ |
| $p_2$:$\{a, b, c, e\}$ | $(2+2+2+1)/4 = 1.75$ |
| $p_3$:$\{a, d, e, f\}$ | $(2+2+1+0)/4 = 1.25$ |
| $p_4$:$\{b, d, g, h\}$ | $(2+2+0+0)/4 = 1.00$ |
| $p_5$:$\{c, i, j, k\}$ | $(2+0+0+0)/4 = 0.50$ |

(a) *apCS* Calculation

| $v$ | $c(v)$ |
|---|---|
| **a:**$\{p_1, p_2, p_3\}$ | 3 |
| **b:**$\{p_1, p_2, p_3\}$ | 3 |
| **c:**$\{p_1, p_2, p_3\}$ | 3 |
| **d:**$\{p_1, p_2, p_4\}$ | 3 |
| **e:**$\{p_2, p_3\}$ | 2 |
| **f:**$\{p_3\}$ | 1 |
| **g:**$\{p_4\}$ | 1 |
| **h:**$\{p_4\}$ | 1 |
| **i:**$\{p_5\}$ | 1 |
| **j:**$\{p_5\}$ | 1 |
| **k:**$\{p_5\}$ | 1 |

(b) Cardinality Hash Table ($cH$)

Fig. 6. Example of the *apCS* algorithm.

where $c(t_j)$ is the number of occurrences of $t_j$ in all places in $\mathcal{S}$. Namely, $c(t_j) - 1$ indicates in how many other contextual sets $t_j$ appears. Therefore, for any pair, for each common $t_j$, we have an increment by 1 to $|p_i \cap p_j|$. Thus, by accumulating them and normalizing with $|p_i|$ we get $apCS(p_i)$ as defined above. Note that this simplification would have not be possible if we did not have a common denominator in our Equation (21).

The algorithm is described in Algorithm 3 and consists of two steps. Figure 6 exemplifies the algorithm using the example of Figure 5.

**Step 1: Generate hash table** $cH$**.** We build a hash table ($cH$) with the number of times an element occurs within places in $\mathcal{S}$. Namely, we parse all sets and create an entry for each unique element $v$ on $cH$. For each subsequent occurrence of an element, we increment its cardinality $c(v)$ by 1 (lines 1–6, Figure 6(b)). The $cH$ table can facilitate the efficient computation of $apCS(p_i)$ as defined by Equation (22).

**Step 2: Compute** $apCS(p_i)$**.** We iteratively parse all contextual sets and sum up the cardinalities of the participating elements by using $cH$. Using Equation (22), we then compute the score of each place in $\mathcal{S}$ (lines 7–9, Figure 6(a)).

Note that this algorithm bypasses pairwise comparisons and caching, which are also needed by the greedy algorithms. Hence, we calculate these scores on demand during the greedy algorithms using Equation (21). The combination of this algorithm with *P&P* algorithm (Section 5.2) that prunes a significant number of fruitless places renders this algorithm very beneficial toward the

**ALGORITHM 4:** Grid Based $pSS$ Calculation

**Input:** (1) set $\mathcal{S}$, (2) $G(G_c, G_z, |G|)$ (grid parameters)
**Output:** (1) $sS(p_i, p_j)$ for all pairs of places in $\mathcal{S}$, (2) $pSS(p_i)$ for each place in $\mathcal{S}$

```
Generate empty grid G(q, 2 · f̄p, |G|)                                  {Step 1}
for each p in S do
    Assign p to the cell c_i which contains p                          {Step 2}
    |c_i| = |c_i| + 1
for each cell c_i with |c_i| > 0 do
    for each cell c_j with |c_j| > 0 and j ≥ i do
        pSS(c_i) = pSS(c_i) + |c_j| · sS(cc_i, cc_j)                    {Step 3}
    pSS(c_i) = pSS(c_i) − 1
```

required total time (as without pruning, this algorithm may not be that beneficial since contextual comparisons are expensive).

This algorithm can be orders of magnitude faster than *msJh* and baseline algorithms. The effect of this approximation (1) on the $HPF(\mathcal{R})$ score is surprisingly positive (and this is because of its combination with the *P&P* algorithm) and (2) on the ranking $\mathcal{S}$ is rather minor. We provide theoretical analysis (e.g., $apCS$ against $pCS$ guarantees an approximation ratio of 2) and extensive experimentation that verify these advantages.

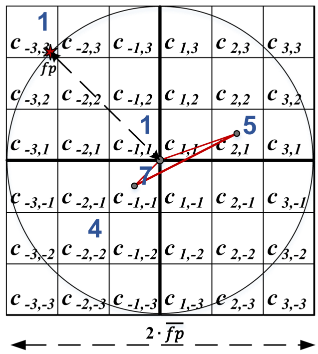
The algorithm has the following time costs. We first need to create the cardinality hash table, which requires $O(K \cdot |p_i|)$ time, where $|p_i|$ is the common number of elements in a set in $\mathcal{S}$. We then process all places in $\mathcal{S}$ and update their $apCS(p_i)$ score by consulting the hash table inducing an additional $O(K \cdot |p_i|)$ cost. Thus, the total cost remains $O(K \cdot |p_i|)$, which is linear to $K$.

## 7 SPATIAL PROPORTIONALITY CALCULATION

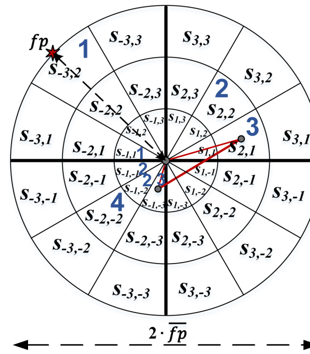
The computation of $pSS(.)$ is demanding as we need to compare all $O(K^2)$ pairs in $\mathcal{S}$. Furthermore, computing Ptolemy's $sS(p_i, p_j)$ is expensive. Specifically, for each distance $||p_i, p_j||$ between two places we need six operations, i.e., $\sqrt{(p_i.x - p_j.x)^2 + (p_i.y - p_j.y)^2}$. We need three distance computations per pair (i.e., for $||p_i, p_j||$, $||p_i, q||$ and $||p_j, q||$). Finally, we also need two more operations, i.e.: (1) the addition of $||p_i, q||$ and $||p_j, q||$ at the denominator and finally (2) the division of the nominator and denominator. Thus, in total, we need 20 operations for each $dS(p_i, p_j)$. We refer to this brute-force computation approach as the ***baseline* algorithm**. Considering its high cost, we propose Grid based $pSS(.)$ approaches, which reduce the cost by one order of magnitude (at some approximation loss).

### 7.1 Grid Based $pSS$ Calculation

We propose an efficient grid based algorithm for $pSS(.)$, which accelerates the computation of Ptolemy's similarity $sS(p_i, p_j)$. We investigate its application on two grid structures, i.e., a squared and a radial grid structure. More precisely, we create a regular grid centered on $q$, which covers the locations of all places in $\mathcal{S}$ and assign each place $p_i$ in $\mathcal{S}$ to the corresponding cell. We approximate $sS(p_i, p_j)$ of any pair of places by replacing their real coordinates with the coordinates of the centers of the respective cells. This approach can decrease drastically the computational cost of $pSS(p_i)$ at a small compromise on approximation. The rationale of proposing a radial grid is that it has smaller cell sizes near the query location and could give a better approximation when many places are located very close to query location. The grid-based approach also has an important and useful property (which we prove). Namely, the $sS(,)$ of the centers of any two cells is independent from the size of the cells. Thus, we can pre-compute the $sS(,)$ scores for the centers of any pair of cells



(a) $pSS$ Squared Grid



(b) $pSS$ Radial Grid

Fig. 7. $pSS$ Grid examples (annotated with $|c_i|$).

and use these scores for any query. Recall that $sS(,)$ calculation requires up to 20 operations. Hence, if we use the pre-computed scores, we reduce this cost to 1 operation only. Algorithm 4 illustrates the algorithm with a pseudo code, and Figure 7 illustrates a running example.

*7.1.1 Squared Grid and Algorithm.* Hereby, we describe the steps of the algorithm when using a squared grid.

**Step 1: Generate the $pSS(.)$ grid.** We define the grid $G$ by a triplet $G(G_c, G_z, |G|)$. The grid is divided into square cells and hence itself is a square. $G_c$ is the center of the grid and it is aligned to the query location $q$. $G_z$ is the length of each of the grid's sides, which is set to $2\overline{fp}$, where $\overline{fp}$ is the distance between $q$ and the furthest point from $q$ in $\mathcal{S}$ (see the example of Figure 7(a)). $|G|$ is the number of cells in the grid. A larger $|G|$ decreases the approximation error but also increases the cost of $pSS(p_i)$ computation.

Each grid row or column has $|g|$ cells, where $|g| = \sqrt{|G|}$. Value $|g|$ should be an even number, because the number of cells on the left (bottom) of the grid's center $G_C$ is equal to the number of cells on the right (top) of $G_C$, as determined by $\overline{fp}$. Each cell $c_i$ contains a number of places, denoted by $|c_i|$. For each query, a good choice of $|G|$ should be such that $|G| \approx K$, according to our experiments.

**Step 2. Allocate places to cells.** We allocate each place $p$ to the cell that contains $p$ and maintain a counter $|c_i|$ for the number of places in each cell. For each cell $c_i$, its center, denoted as $cc_i$, represents (i.e., approximates) the locations of all places in $c_i$.

**Step 3. Calculate $pSS(,)$.** Let us assume that $sS(cc_i, cc_j)$ between the centers $(cc_i, cc_j)$ of every pair of cells $(c_i, c_j)$ has been pre-computed and is accessible from a matrix $sSM$. We calculate the $pSS(c_i)$ of a cell, by considering the cardinality $|c_i|$ and the cardinality $|c_j|$ of all other cells together with the precomputed $sS(cc_i, cc_j)$ scores, by adapting Equation (6) as follows:

$$pSS(c_i) = \sum_{c_j \in G} |c_j| \cdot (sS(cc_i, cc_j)) - 1. \tag{23}$$

$pSS(c_i)$ represents the score for any place $p$ residing in $c_i$ and will be the same for all places in $c_i$, i.e., $pSS(p) = pSS(c_i)$ for each $p$ in $c_i$. In the computation of $pSS(c_i)$, we also consider all places in $c_i$; $c_i$ includes $|c_i|$ collocated places with $sS(p, p_j) = 1$ for all $p, p_j$ in $c_i$. We subtract 1 in order to disregard the comparison of a place against itself. We consider all cells with $|c_i| > 0$.

**Precomputation.** The algorithm requires that the $sS(cc_i, cc_j)$ scores between all cell centers are pre-computed for any resolution and position of $G$. This is possible because of the nature of

Ptolemy's similarity, which makes it independent from the scale of distances between points; only their relative orientation to $q$ matters. We prove this property in Theorem 7.1, at the end of this section. Specifically, the $sS(cc_i, cc_j)$ score depends on the relative position of cells $c_i$ and $c_j$ w.r.t. the center of the grid, where this position is measured in terms of number of cells. For example, in Figure 7, $sS(cc_{-1,1}, cc_{-1,-1})$ equals to $1 - 1/\sqrt{2}$ and depends only on the relative positions of the cells w.r.t. the grid center, but not on their sizes. Hence, by pre-computing all scores for a large grid $G_{MAX}$ which can be superimposed on top of any query, we can use the pre-computed values. If the query requires a smaller grid (recall that $|G| \approx K$), where $|G| \leq |G_{MAX}|$, then we use only the pre-computed scores of the respective subset of $G_{MAX}$.

**Complexity.** For step 1, in order to generate the grid, we need $O(|G|)$ time. During Step 2, we need $O(K)$ operations to assign $K$ places to cells. For step 3, in order to calculate the $p\mathcal{SS}()$ for a pair of cells, we need two operations (i.e., multiplying $|c_j|$ by $sS(cc_i, cc_j)$). In the worst case, the $K$ places will be in different cells. Thus, for calculating $p\mathcal{SS}(c_i)$, we will need $2 \cdot K$ operations. Hence, for the whole grid with $K$ cells, we will need $O(K^2)$ operations in the worst case. The space complexity is $O(K)$, since $|G| \approx K$, while the storage requirements for pre-computation are $O(|G_{MAX}|)$.

*7.1.2 Radial Grid.* An alternative to the square grid approximation is a radial grid $R$, which is defined by sectors formed by (1) circles and (2) lines as follows. We use a set of $r_c$ homocentric circles, all centered at the grid center $R_c$ (i.e., the query location $q$). These circles have as radii multiples of a constant $c_z$, where the outmost circle has diameter $2 \cdot \overline{fp}$. We also use a set of $R_d$ lines that divides the space into equal slices (any two consecutive lines have a common angle). These lines' lengths are set to the diameter of the outmost circle (Figure 7(b)). The algorithm (i.e., Algorithm 4) remains the same; but here, we have a radial grid and sectors (instead of cells). The rationale of using a radial grid is that it has smaller cell sizes near the query location and could give a better approximation when many places are located very close to $q$. We set $R_d = 2 \cdot r_c$, which results in $|R| = 2 \cdot R_d \cdot r_c$ sectors. Hence, the radial grid can be denoted by $R(R_c, R_z, |R|)$, where (1) $R_c$ is the center of the grid ($q$), (2) $R_z$ is the length of the diameter and is set to $2 \cdot \overline{fp}$, and (3) $|R|$ is the number of sectors (cells) in the grid (i.e., $R_d^2$). Note that $R_z = 2 \cdot r_c \cdot c_z$. Each $s_i$ may contain a number of places, denoted as $|s_i|$. We use the center $sc_i$ of a sector $s_i$ as the representative point, defined by the intersection between a circle having as radius the average radii of the two circles that define it and the diameter having as angle the average angle of the two diameters that define the sector. We can see that Theorem 7.1 (i.e., we can pre-compute and reuse the $sS(,)$ of sectors) applies here as well. Finally, we can easily see that the same time and space analysis as of the square grid applies here as well. For instance, during step 3, which is the most demanding step, in the worst case, the $K$ places will be placed in $K$ different sectors; thus, we will still need $O(K^2)$ operations.

*7.1.3 Scale-Free Property of Ptolemy's Similarity.* Given a pair of points $(p_i, p_j)$ and a query location $q$, we now prove that their $sS(p_i, p_j)$ score remains the same if we multiply their difference to $q$ in all dimensions by the same factor $f$. Formally:

Theorem 7.1. *Let $p_i$ and $p_j$ be two points with coordinates $(x_i, y_i)$ and $(x_j, y_j)$, respectively. Let $q$ be a query location with coordinates $(x_q, y_q)$. Let $p'_i$ and $p'_j$ be two points with coordinates $(x'_i, y'_i)$ and $(x'_j, y'_j)$, respectively, such that:*

$$(x'_i - x_q) = f \cdot (x_i - x_q),\ (y'_i - y_q) = f \cdot (y_i - y_q),\ (x'_j - x_q) = f \cdot (x_j - x_q),\ \textit{and}\ (y'_j - y_q) = f \cdot (y_j - y_q).$$

*It holds that $sS(p_i, p_j) = sS(p'_i, p'_j)$.*

Proof. We have $sS(p'_i, p'_j) = 1 - \frac{||p'_i, p'_j||}{||p'_i, q|| + ||p'_j, q||} = 1 - \frac{\sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2}}{\sqrt{(x'_i - x_q)^2 + (y'_i - y_q)^2} + \sqrt{(x'_j - x_q)^2 + (y'_j - y_q)^2}}$.

We also have $x'_i - x'_j = f \cdot (x_i - x_q) - f \cdot (x_j - x_q) = f \cdot (x_i - x_j)$ and similarly $y'_i - y'_j = f \cdot (y_i - y_j)$, $x'_i - x_q = f \cdot (x_i - x_q)$, $y'_i - y_q = f \cdot (y_i - y_q)$, $x'_j - x_q = f \cdot (x_j - x_q)$, $y'_j - y_q = f \cdot (y_j - y_q)$. Hence, $sS(p'_i, p'_j) = 1 - \frac{\sqrt{f \cdot (x_i - x_j)^2 + f \cdot (y_i - y_j)^2}}{\sqrt{f \cdot (x_i - x_q)^2 + f \cdot (y_i - y_q)^2} + \sqrt{f \cdot (x_j - x_q)^2 + f \cdot (y_j - y_q)^2}} =$
$1 - \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sqrt{(x_i - x_q)^2 + (y_i - y_q)^2} + \sqrt{(x_j - x_q)^2 + (y_j - y_q)^2}} = 1 - \frac{||p_i, p_j||}{||p_i, q|| + ||p_j, q||} = sS(p_i, p_j)$. □

Now, consider a grid $G$ that is centered at $q$. For every pair of cells $c_i, c_j$ in the grid $G$, let $(cc_i, cc_j)$ be the corresponding pair of cell centers. Based on Theorem 7.1, score $sS(cc_i, cc_j)$ is independent from the cell size $c_z$ and only depends on the relative positions of $c_i, c_j$ w.r.t. the grid's center, measured in terms of number of cells. For example, in Figure 7, the grid cells are given identifiers, based on their relative position (in number of cells) to the grid center. Based on Theorem 7.1, the $sS(cc_{a,b}, cc_{c,d})$ score between any two cell centers $cc_{a,b}$ and $cc_{c,d}$ depends only on the grid-based coordinates $(a, b)$ and $(c, d)$ of cells $c_{a,b}$ and $c_{c,d}$ and not on the sizes of the cells. This is because in two grids $G$ and $G'$, the ratio of the differences between cell centers $cc_{a,b} \in G$ and $cc'_{a,b} \in G'$ and the corresponding grid centers in each dimension is the same for any $(a, b)$. In addition, $sS(cc_{a,b}, cc_{c,d})$ is the same for any position of the grid center. Summing up, the same pre-computed $sS(cc_{a,b}, cc_{c,d})$ values are used for any query location $q$ and any grid size $G_z$ and number of cells $|G|$.

## 8 THEORETICAL ANALYSIS

In this section, we analyze the approximation bounds of our (squared and radial) grid based algorithms, the *apCS* algorithm, and greedy algorithms (*IAdU* and *ABP*).

### 8.1 Bounds of Greedy Algorithms

Our proofs are based on the assumption that $HPF(u, v)$ satisfies the triangle inequality. For this purpose, we first investigate when does $HPF(u, v)$ satisfy the triangle inequality. Then, by using this key observation, we can trivially prove the approximation loss.

LEMMA 8.1. *Given a set of distance functions $dF_1(u, v), \ldots, dF_n(u, v)$ that satisfy triangle inequality, then their weighted summation (denoted as $dF(u, v) = \sum w_i \cdot dF_i(u, v)$) also satisfies triangle inequality, as given by*

$$dF(u, v) + dF(v, w) \geq dF(u, w).$$

PROOF. By definition of $dF(u, v)$, the inequality can be rewritten as: $\sum w_i \cdot dF_i(u, v) + \sum w_i \cdot dF_i(u, w) \geq \sum w_i \cdot dF_i(v, w)$. Thus,

$w_1 \cdot dF_1(u, v) + w_1 \cdot dF_1(v, w) \geq w_1 \cdot dF_1(u, w)$,
$\vdots$
$w_n \cdot dF_n(u, v) + w_n \cdot dF_n(v, w) \geq w_n \cdot dF_n(u, w)$.

The addition of these equations completes the proof. □

In general, any diversity function $dF(u, v)$ maintains its triangle inequality properties as long as the constituent components follow triangle inequality. Since from Reference [4], we know that $dS(u, v)$ (i.e., $1 - sS(u, v)$) satisfies the inequality and from Reference [41] we see that $dC(v, w)$ (i.e., $1 - sC(u, v)$), which is a Jaccard distance is a metric and hence satisfies the triangle inequality; then, we can infer that $dF(u, v)$ (i.e., $1 - sF(u, v)$) also satisfies triangle inequality.

THEOREM 8.2. *$HPF(u, v)$ (Equation (15)) satisfies the Triangle Inequality when $rF(v) \geq \frac{\lambda \cdot (k-1)}{(1-\lambda) \cdot (K-k)}$.*

Proof. By expanding $HPF(u, v)$ we get:

$(1-\lambda) \cdot \frac{K-k}{k-1} \cdot (rF(u) + rF(v)) + \lambda \cdot (\frac{1}{k-1} \cdot (pF\mathcal{S}(u) + pF\mathcal{S}(v)) - 2 \cdot sF(u,v)) + (1-\lambda) \cdot \frac{K-k}{k-1} \cdot (rF(v) + rF(w)) + \lambda \cdot (\frac{1}{k-1} \cdot (pF\mathcal{S}(v) + pF\mathcal{S}(w)) - 2 \cdot sF(v,w))$
$\geq (1-\lambda) \cdot \frac{K-k}{k-1} \cdot (rF(u) + rF(w)) + \lambda \cdot (\frac{1}{k-1} \cdot (pF\mathcal{S}(u) + pF\mathcal{S}(w)) - 2 \cdot sF(u,w))$
$\implies (1-\lambda) \cdot \frac{K-k}{k-1} \cdot rF(v) + \lambda \cdot \frac{1}{k-1} \cdot pF\mathcal{S}(v) - \lambda \cdot sF(u,v) - \lambda \cdot sF(v,w) \geq -\lambda \cdot sF(u,w)$
$\implies (1-\lambda) \cdot \frac{K-k}{k-1} \cdot rF(v) + \lambda \cdot \frac{1}{k-1} \cdot pF\mathcal{S}(v) - \lambda \cdot (sF(u,v) + sF(v,w) - sF(u,w)) \geq 0$
$\implies (1-\lambda) \cdot \frac{K-k}{k-1} \cdot rF(v) + \lambda \frac{1}{k-1} \cdot pF\mathcal{S}(v) - \lambda \cdot (1 - dF(u,v) - dF(v,w) + dF(u,w)) \geq 0.$

Considering that $dF(u, v)$ ranges in $[1, 0]$ and satisfies triangle inequality (according to Lemma 8.1), then the minimum value for $dF(u, v) + dF(v, w) - dF(u, w)$ is 0. Then we have:
$(1-\lambda) \cdot \frac{K-k}{k-1} \cdot rF(v) + \lambda \frac{1}{k-1} \cdot pF\mathcal{S}(v) - \lambda \cdot 1 \geq 0$
$\implies (1-\lambda) \cdot (K-k) \cdot rF(v) + \lambda \cdot pF\mathcal{S}(v) \geq \lambda \cdot (k-1)$
$\implies rF(v) \geq \frac{\lambda \cdot (k-1)}{(1-\lambda) \cdot (K-k)}$. □

For further simplification, we drop $pF\mathcal{S}(v)$ (which is the summation of $K - k$ places (including $sF(u, v)$ and $sF(v, w)$) and thus should be a significant value.

If we see more carefully this inequality, it holds in most pragmatic cases and our default settings. For $\lambda = 0.5$ and $K = 10 \cdot k = 10k$, then we get: $rF(v) \geq \frac{k-1}{10k-k} \implies rF(v) \geq \frac{k}{9k} \implies rF(v) \geq 1/9$. In summary, we have triangle inequality when $rF(v) \geq 0.1$. This is a pragmatic case as results with smaller $rF(v)$ are not really relevant and they never make it in the $\mathcal{S}$.

**Approximation Bounds**. Given $HPF(u, v)$ satisfies triangle inequality, *IAdU* and *ABP* algorithms can achieve approximation ratios of 4 and 2, respectively. For such conditions, these bounds are proved by Reference [4] and are based on earlier work in References [31] and [42].

### 8.2 Bounds of the *apCS* Algorithm

We study the worst-case of the $apC\mathcal{S}(p_i)$ produced by the $apC\mathcal{S}$ algorithm against the exact $pC\mathcal{S}(p_i)$. Recall that in this algorithm, we assume that all places have a common set size. The algorithm considers a relaxation of Jaccard similarity ($asC(p_i, p_j)$) by replacing the denominator $|p_i \cap p_j|$ with $|p_i|$ (where $|p_i| = |p_j|$).

Theorem 8.3. *Given a set $\mathcal{S}$ of places with a common contextual set size, in the worst case $apC\mathcal{S}$ will give us* $\frac{apC\mathcal{S}(p_i)}{pC\mathcal{S}(p_i)} = 2$.

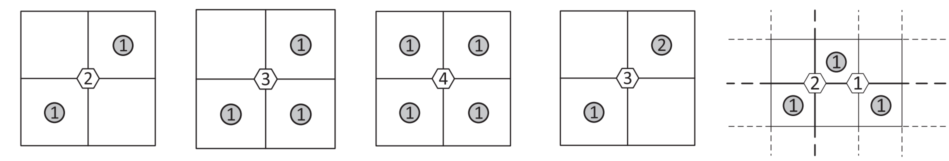
Proof. $pC\mathcal{S}(p_i)$ and $apC\mathcal{S}(p_i)$ are sums of the pairwise components $sC(p_i, p_j)$ (i.e., the Jaccard similarity $\frac{|p_i \cap p_j|}{|p_i \cup p_j|}$) and $asC(p_i, p_j)$ ($\frac{|p_i \cap p_j|}{|p_i|}$), respectively. Thus, we have the following ratio:

$$\frac{asC(p_i, p_j)}{sC(p_i, p_j)} = \frac{\frac{|p_i \cap p_j|}{|p_i|}}{\frac{|p_i \cap p_j|}{|p_i \cup p_j|}} = \frac{|p_i \cup p_j|}{|p_i|} < 2. \tag{24}$$

Since $|p_i|$ is fixed for $\mathcal{S}$, we can easily see that the ratio is maximized when $|p_i \cup p_j|$ is maximized. More precisely, this is the case when there is only one common element between the two sets. In this case, we get $|p_i \cup p_j| = 2 \cdot |p_i| - 1$, thus this makes the worst case ratio less than 2. Note that although $|p_i \cap p_j|$ is maximized when $|p_i \cap p_j| = 0$, this case will result to $asC(p_i, p_j) = sC(p_i, p_j) = 0$. We can now see that the following also holds:

$$\frac{apC\mathcal{S}(p_i)}{pC\mathcal{S}(p_i)} = \frac{\sum_{p_j \in \mathcal{S}} asC(p_i, p_j)}{\sum_{p_j \in \mathcal{S}} sC(p_i, p_j)} < 2. \tag{25}$$

This completes the proof. □




(a) $D_G.1$ ($ap = \infty$) (b) $D_G.2$ ($ap = 5$) (c) $D_G.3$ ($ap = 5$) (d) $D_G.4$ ($ap = 0.33$) (e) $D_G.5$ ($ap = 1.8$)

Fig. 8. Cases D: Error decrement (hexagons and circles indicate the original and new location of places, respectively; numbers indicate the amount of co-located places).

Note that we can achieve this worst case ratio even when the two sets do not have the same size. Namely, when we normalize $asC(p_i, p_j)$ using the size of the largest set between $|p_i|$ and $|p_j|$. We can get the same worst case when we consider $p_i$ is marginally larger (or equal to $p_j$). For instance, consider the case of two sets, $p_1$ and $p_2$ with sizes $|p_1| = 99$ and $|p_2| = 100$ and also consider they have one element in common. Then, their Jaccard similarity will be $1/198$. By considering the denominator being $|p_2| = 100$, then we have $1/100$.

Furthermore, we have another useful property among $sC(p_i, p_j)$, $asC(p_i, p_j)$ and $|p_i \cap p_j|$. They have a monotonic relationship. Namely, the relative order of any pair of sets remains the same for $sC(p_i, p_j)$, $asC(p_i, p_j)$ and $|p_i \cap p_j|$ scores. This property has a positive impact on the ranking of $\mathcal{S}$ and $HPF(\mathcal{R})$ scores, which is also verified experimentally.

THEOREM 8.4. $sC(p_i, p_j) > sC(p_k, p_l) \Leftrightarrow asC(p_i, p_j) > asC(p_k, p_l) \Leftrightarrow |p_i \cap p_j| > |p_k \cap p_l|$.

PROOF. $sC(p_i, p_j) > sC(p_k, p_l) \Leftrightarrow \frac{|p_i \cap p_j|}{|p_i \cup p_j|} > \frac{|p_k \cap p_l|}{|p_k \cup p_l|} \Leftrightarrow \frac{|p_i \cap p_j|}{2|p_i| - |p_i \cap p_j|} > \frac{|p_k \cap p_l|}{2|p_i| - |p_k \cap p_l|} \Leftrightarrow$ $2|p_i||p_i \cap p_j| - |p_i \cap p_j||p_k \cap p_l| > 2|p_i||p_k \cap p_l| - |p_i \cap p_j||p_k \cap p_l| \Leftrightarrow |p_i \cap p_j| > |p_k \cap p_l|$. Since all sets have a common size, we can see that the above inequality also holds for $asC(p_i, p_j) > asC(p_k, p_l)$ (i.e., by dividing by $|p_i|$). □

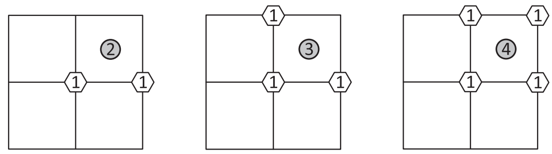
## 8.3 Bounds of Grid Based Algorithms

We study the worst-case of the approximation quality of $pSS(\mathcal{S}) = \sum_{pi \in \mathcal{S}} pSS(p_i)$ produced by our grid based algorithms. More precisely, we study how the ratio, $ap$, of the optimal $pSS(\mathcal{S})$ (denoted as $pSS_o(\mathcal{S})$) to the approximated $pSS(\mathcal{S})$ (denoted as $pSS_a(\mathcal{S})$) ranges (i.e., its lower and upper bounds). As we will discuss shortly, our approximation algorithm can either increase or decrease the $sS(p_i, p_j)$ score of a pair of places, which consequences to have both an upper and lower bound of $ap$.

We prove our bounds by induction. We first study base cases for small values of $K$ and prove the bounds of their worst case. Namely, for $K = 4$, we found that $ap$ ranges between a lower bound of $AP_{LB} = 1/4$ and an upper of $AP_{UB} = 5$. Then, we prove the respective worst cases of the induction cases. We prove that $AP_{LB} \cdot \frac{K-1}{K+1} \leq ap \leq AP_{UB} \cdot \frac{K+1}{K-1}$ (Theorems 8.5 and 8.6). For large values of $K$, we can easily see that both $\frac{K+1}{K-1}$ and $\frac{K-1}{K+1}$ become negligible.

Our approximation algorithm can compute either a higher or a lower value compared to the actual score $sS(p_i, p_j)$ of a pair of places. Thus, we study the bounds of the worst case of the two cases separately:

- **Error due to** $sS(p_i, p_j)$ **decrement (Case D, Figure 8)**; i.e., after relocation, $sS(p_i, p_j)$ scores of pairs are decreased, e.g., the maximum decrease from 1 to 0.
- **Error due to** $sS(p_i, p_j)$ **increment (Case I, Figure 9)**; i.e., after relocation, $sS(p_i, p_j)$ scores of pairs are increased, e.g., the maximum increase from 0 to 1.




(a) $I_G.1$ ($ap = 0$) (b) $I_G.2$ ($ap = 1/10$) (c) $I_G.3$ ($ap = 1/4$)

Fig. 9. Cases I: Error increment (hexagons and circles indicate the original and new location of places, respectively; numbers indicate the amount of co-located places).

### 8.3.1 *Estimation of the ap Upper Bound Due to $sS(p_i, p_j)$ Decrement (Case D) on Squared Grid.*

**Base Case.** We illustrate this case, that can give us an upper bound of *ap* (i.e., $AP_{UB} = 5$ for $K = 4$), with Figure 8. Where, all places are collocated on the center of the grid, which maximizes their $sS(,)$ (i.e., $sS(,) = 1$) and therefore also maximizes the $pSS_o(\mathcal{S})$ score (i.e., $pSS_o(\mathcal{S}) = K \cdot (K-1)$). Then, we assume that our algorithm relocates these places in such a way that minimizes $pSS_a(\mathcal{S})$, thus maximizing *ap*.

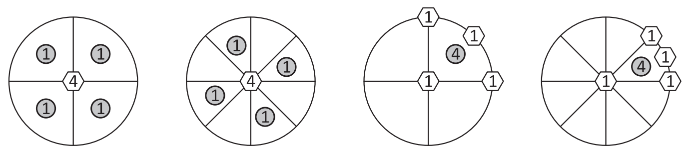
For $K = 2$, we can have the worst case ($D_G.1$), i.e., $pSS_o(\mathcal{S}) = 2, pSS_a(\mathcal{S}) = 0$ and $ap = \infty$; when two places co-located on the center of the grid are then relocated to the centers of diametrically opposite cells. This will result to the maximum loss of $sS(,)$ of the pair, from 1 to 0. Following the same scenario of co-located places, the addition of a third place will result to the maximum *ap*, if the three places are relocated in three different cells (case $D_G.2$). Namely, $pSS_o(\mathcal{S}) = K \cdot (K-1) = 3 \cdot 2 = 6$ and $pSS_a(\mathcal{S}) = 2 \cdot 0 + 4 \cdot a = 1.2$ (where $a = 1 - 1/\sqrt{2} \approx 0.3$); thus $ap = \frac{6}{1.2} = 5$. For a fourth place, we get a maximum *ap* when all places are relocated in four different cells (case $D_G.3$). Namely, $pSS_o(\mathcal{S}) = 4 \cdot 3 = 12$ and $pSS_a(\mathcal{S}) = 4 \cdot 0 + 8 \cdot a = 2.4$, thus $ap = \frac{12}{2.4} = 5$. Note that any other arrangement, e.g., such as $D_G.4$ or $D_G.5$ will not give a higher *ap*. In case $D_G.4$, where our grid based algorithms co-locate two places, we get $ap = 2/6$. The case of $D_G.5$, where one place is located on the border with another cell, if our grid algorithm relocates this place to another cell (which increases *ap*), this will result to $ap = \frac{2}{1.1} = 1.8$; note that the fact that the third place is not co-located with the two places reduces the $pSS_o(\mathcal{S})$. In summary, with this base case, we have for $K = 4$ $AP_{UB} = 5$.

**Induction.** Hereby, we study the induction step of this case.

Theorem 8.5. *Given a set $\mathcal{S}$ with $K$ places where the upper bound of ap is $AP_{UB}$, we would like to prove that the upper bound $AP_{UB} \cdot \frac{k+1}{k-1}$ holds for $\mathcal{S}'$ with $K+1$ places (i.e., by adding a new place).*

Proof. Let's assume that we have all places co-located, so we have $sS(,) = 1$ for all pairs, which will give us the maximum possible score for $K$, i.e., $pSS_o(\mathcal{S}) = K \cdot (K-1)$. After applying the grid based algorithm, let's assume that, the places are distributed in such a way that $AP_{UB} = pSS_o(\mathcal{S})/pSS_a(\mathcal{S})$, thus we can infer that $pSS_a(\mathcal{S}) = pSS_o(\mathcal{S})/AP_{UB}$.

Let's study the case where we add the new place ($K+1$). Let $pSS_o(\mathcal{S}')$ and $pSS_a(\mathcal{S}')$ be the optimal and approximated scores, respectively, for this new set ($\mathcal{S}'$). Let's assume again that the new place is co-located with the existing $K$ places, which will give us the maximum possible score for $K+1$ places, i.e., $pSS_o(\mathcal{S}') = (K+1) \cdot K$. After applying the grid based algorithm, let's assume that the places are arranged in such a way that minimizes the $pSS_a(\mathcal{S}')$ score. Let's assume that $pSS_a(\mathcal{S}')$ can be as bad as $pSS_a(\mathcal{S})$, i.e., $pSS_a(\mathcal{S}') = pSS_a(\mathcal{S})$. Note that since, $pSS(\mathcal{S}') = pSS(\mathcal{S}) + \sum_{p_i \in \mathcal{S}} pSS(p_j)$, we can easily see that $pSS(\mathcal{S}')$ score is a monotonic function with regards to $K$, i.e., by adding new places, the score can only increase; thus, we can safely infer that this score can remain the same in the worst case. Now, we can calculate $ap = \frac{pSS_o(\mathcal{S}')}{pSS_a(\mathcal{S}')} = \frac{pSS_o(\mathcal{S}')}{pSS_a(\mathcal{S})} = \frac{pSSo(\mathcal{S}')}{\frac{pSS_o(\mathcal{S})}{AP_{UB}}} = \frac{K \cdot (K+1)}{\frac{K \cdot (K-1)}{AP_{UB}}} = AP_{UB} \cdot \frac{K+1}{K-1}$. □




(a) $D_R.1$ ($ap = 5$) (b) $D_R.2$ ($ap = 5$) (c) $I_R.1$ ($ap = 1/4$) (d) $I_R.2$ ($ap = 0.36$)

Fig. 10. Base cases of the radial grid algorithm.

*8.3.2 Estimation of the ap Lower Bound Due to $sS(p_i, p_j)$ Increment (Case I) on Squared Grid.*

**Base Case.** Analogously, we illustrate this case, that can give us a lower bound of *ap* (i.e., $AP_{LB} = 1/4$ for $K = 4$), with (Figure 9). In this case, we assume that originally places were located in such a way that minimizes $pSS_o(\mathcal{S})$. Then, our algorithm relocates all places on the same location which will give us the maximum $pSS_a(\mathcal{S}) = K \cdot (K-1)$ score.

For $K = 2$, we can have the worst case ($I_G.1$), i.e., $pSS_o(\mathcal{S}) = 0$, $pSS_a(\mathcal{S}) = 2$ and $ap = 0$; when the first place is on the center and the second on the edge of a cell (i.e., $sS(,) = 0$). After relocation, both will be relocated on the center of the cell, thus $pSS_a(\mathcal{S}) = 2$. Following the same scenario, let's assume that a third place exists in such a way that minimizes $sS(,)$ with the existing place not placed on the center. We can easily see that if the two non centered places are on the remote corners of the cell (case $I_G.2$), this will result to the minimum possible $sS(,)$ among these two places, so $pSS_o(\mathcal{S}) = 0.6$ and $pSS_a(\mathcal{S}) = 6$, thus $ap = 1/10$. Let's assume a fourth place exists on the fourth corner (case $I_G.3$); this way, the three places will be as remote as possible w.r.t. center, then we have $pSS_o(\mathcal{S}) = 2.94$, $pSS_a(\mathcal{S}) = 12$ and $ap = 1/4$. In summary, with this case, we have for $K = 4$ $AP_{LB} = 1/4$.

**Induction.** Hereby, we study the induction step of this case.

THEOREM 8.6. *Given a set $\mathcal{S}$ with K places that give us $ap = AP_{LB}$, we would like to prove that the lower bound $AP_{LB} \cdot \frac{k-1}{k+1}$ holds for $\mathcal{S}'$ with $K + 1$ places (i.e., by adding a new place).*

PROOF. We can easily reverse the previous proof. Let's assume that our grid based algorithm relocates all places on the same location, thus we have $pSSa(\mathcal{S}) = K \cdot (K-1)$, which is the maximum score we can get for $K$. Given that $AP_{LB} = \frac{pSS_o(\mathcal{S})}{pSS_a(\mathcal{S})}$, we can infer that $pSS_o(\mathcal{S}) = AP_{LB} \cdot pSS_a(\mathcal{S})$.

Let's study the case where we add the new place ($K+1$). Let's assume that the places are arranged in such a way that minimizes the $pSS_o(\mathcal{S}')$ score, i.e., $pSS_o(\mathcal{S}')$ can be as bad as the $pSS_o(\mathcal{S})$ ($pSS_o(\mathcal{S}') = pSS_o(\mathcal{S})$) (recall that $pSS(\mathcal{S})$ scores are monotonic to newly added places). Let's assume again that the new place is relocated on same location with the existing $K$ places, thus $pSS_a(\mathcal{S}') = (K+1) \cdot K$. Now, we can calculate $ap = \frac{pSS_o(\mathcal{S}')}{pSS_a(\mathcal{S}')} = \frac{pSS_o(\mathcal{S})}{pSSa(\mathcal{S}')} = \frac{AP_{LB} \cdot pSS_a(\mathcal{S})}{pSS_a(\mathcal{S}')} = \frac{AP_{LB} \cdot K \cdot (K-1)}{K \cdot (K+1)} = AP_{LB} \cdot \frac{K-1}{K+1}$. □

*8.3.3 Radial Grid.* The bounds of the worst cases of the radial grid algorithm have the same behavior as the squared grid based algorithm. We can easily see that we can achieve the same worst case bounds for both base and induction cases. More precisely with $Rd = 2$ and $K = 4$, we can construct analogously the same respective cases and get the same upper and lower bounds of *ap*, i.e., $AP_{UB} = 5$ and $AP_{LB} = 1/4$, respectively (cases $D_R.1$ and $I_R.1$, Figure 10). Note that as $Rd$ increases, we achieve better *ap* bounds for the base cases (i.e., closer to 1), (cases $D_R.2$ and $I_R.2$, Figure 10). Finally, we can easily see that we can use the same inductions here as well. Thus, the bounds remain the same as for the squared grid based algorithm.

## 9 EXPERIMENTS

In this section, we evaluate the efficiency and approximation quality of the proposed proportionality framework. Finally, we present a user evaluation and testing of our approach.

### 9.1 Setup

**Datasets.** We used the datasets that have been used in References [4, 44]; namely, DBpedia and Yago2 (version 2.5). The DBpedia RDF graph has 8,099,955 vertices and 72,193,833 edges. Among all vertices, 883,665 are places with coordinates. Yago2 has 8,091,179 vertices and 50,415,307 edges. Among these vertices, 4,774,796 are places. In general, our techniques had similar behavior on both datasets; for brevity, we present all results on DBpedia and skip some results on Yago2 if they are similar. For the experiments, where we test the performance of our grids, we also used synthetically generated data, which will be discussed in detail later.

**Queries.** In the evaluation, we selected locations and keywords, to form a total of 100 queries, such that the number of retrieved places per query is at least 2,000. For each place $p_i$ in the query result, we compute its relevance score $rF(p_i)$ to the query $q$ by combining the Jaccard similarity to the keywords and the normalized distance of $p_i$ to the query location (normalization by dividing to the largest distance of the city) [2, 4].

**Experimental settings.** Our methodology and algorithms are evaluated by varying a different number of problem parameter values. First, we experimented with different sizes $K$ of the retrieved set $\mathcal{S}$. For a given query, for each value $K$, we selected from the query results, the $K$ most relevant places to form $\mathcal{S}$ according to $rF(p_i)$. $K$ varies in $\{20, 50, \mathbf{100}, 200, 400, 1{,}000, 2{,}000\}$, with 100 being the default value (we also present extensive results for $K = 1{,}000$). Second, we experimented with different values of $|p_i|$, i.e., the number of elements in the contextual sets of $p_i$s in $\mathcal{S}$. In all experiments, we use a common set size $|p_i|$ per $p_i$. For a given $\mathcal{S}$, we formed the contextual sets of the places included in it, by using keywords from neighboring vertices to $p_i$ in the corresponding RDF graph, until the desired $|p_i|$ is reached for each $|p_i|$. That is, we enriched (or constrained) the contextual sets of the places on demand by adding (or removing) keywords, in order to satisfy the requirement of the required $|p_i|$ by the experiment. During this phase, we can detect and remove any noise in the data such as repetitions that can impact the quality of our algorithms. The tested $|p_i|$ values range in $\{20, 40, 50, 60, \mathbf{100}, 150, 200, 400\}$, with 100 being the default value. Third, we experimented with different values of the grid size $|G|$; i.e., values in $\{36, 64, \mathbf{100}, 144, 196\}$ with a default of $|G| = 100$. Fourth, we experimented with values of $k$ in $\{5, \mathbf{10}, 15, 20\}$ with a default value of 10. We experimented with different values of the weights $\lambda$ and $\gamma$, with default $\lambda = \gamma = 0.5$.

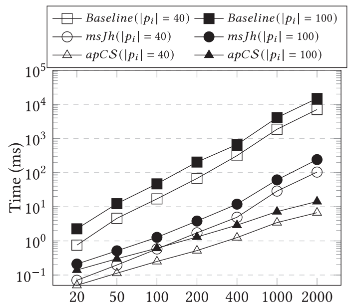
**Platform.** All methods were implemented in Java and were conducted in memory. We used a 2.7 GHz dual-core (boost @ 3.48 GHz) quad-thread machine with 16 GB (DDR4 @ 3,300 MHz) of memory, running Windows 10.

### 9.2 Efficiency

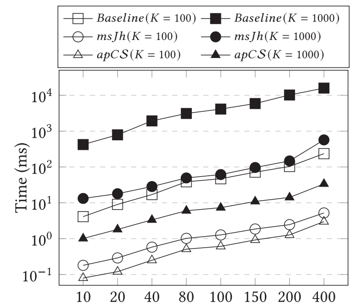
In this section, we measured the average run-time costs of the tested algorithms on our queries for the various parameter values.

*9.2.1 Contextual and Spatial Proportionality Algorithms.* We study the efficiency of our solutions for contextual and spatial proportionality computation, presented in Sections 6 and 7.

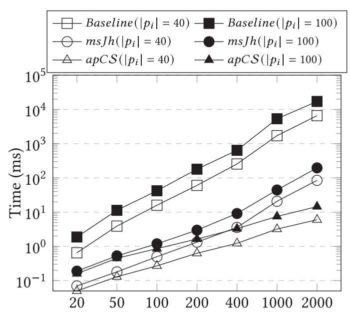
**Contextual Proportionality.** Figure 11 compares the performance of our $ms\mathcal{J}h$ and $ap\mathcal{CS}$ algorithms against the baseline algorithm for calculating $p\mathcal{CS}(p_i)$ for all $p_i \in \mathcal{S}$. Figure 11 reveals that all algorithms have similar behaviour with regards to $K$ and $|p_i|$ increments, namely, required times also increase. As expected, $ap\mathcal{CS}$ is the fastest, then $ms\mathcal{J}h$ and lastly the baseline algorithm. We also observe that as $K$ and $|p_i|$ increase the time difference among these algorithms also in-



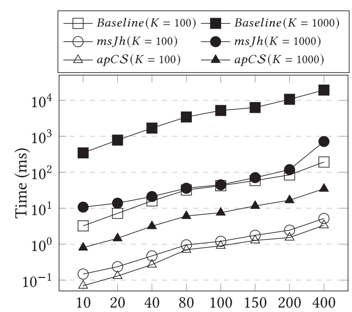

(a) Varying $K$, **DBpedia**




(b) Varying $|p_i|$, **DBpedia**




(c) Varying $K$, **Yago2**



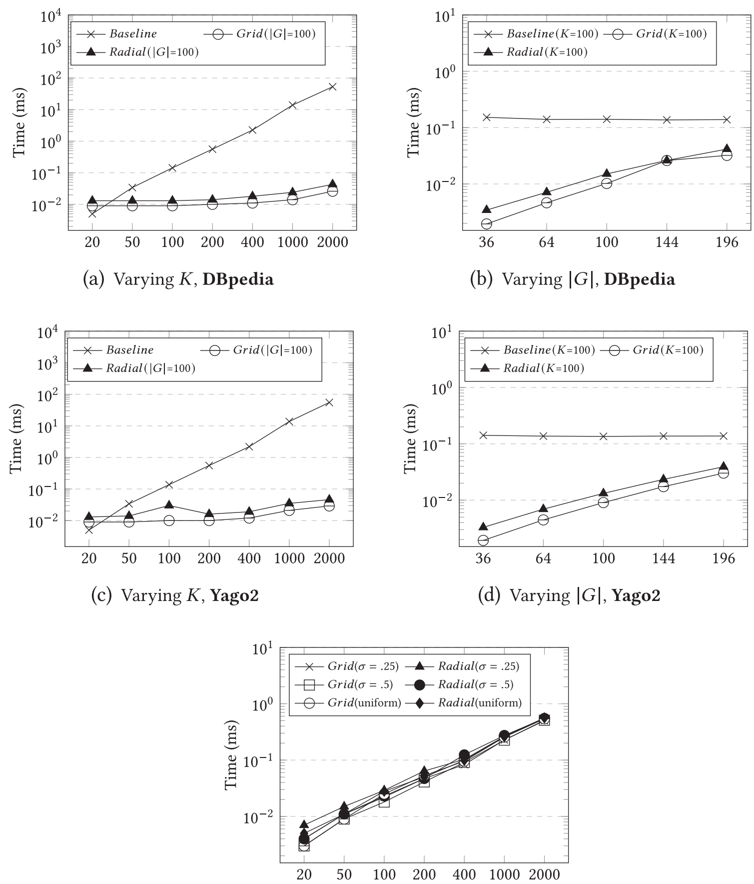

(d) Varying $|p_i|$, **Yago2**

Fig. 11. Efficiency of msJh and $apC\mathcal{S}$ algorithms (Jaccard).

creases. For instance in DBpedia for $K = 100$ and $|p_i| = 100$, the required times are 0.6 ms, 1.3 ms, and 46.9 ms, whereas for $K = 2{,}000$ and $|p_i| = 100$ the required times are 14.2 ms, 240.8 ms, and 14908.9 ms, respectively. We see that $apC\mathcal{S}$ can be more than one and three orders of magnitude faster than $ms\mathcal{J}h$ and baseline algorithms, respectively.

We also implemented minhash and compared it with $apC\mathcal{S}$ and $ms\mathcal{J}h$. However, minhash performed poorly for our settings, minhash never outperforms $apC\mathcal{S}$ and outperforms $ms\mathcal{J}h$ only when $K$ and $|p_i|$ become larger than 1,000 and 200, respectively. For instance for $K = 1{,}000$ and $|p_i| = 400$, $apC\mathcal{S}$ requires only 33.6 ms, whereas $ms\mathcal{J}h$ and minhash requires 569.4 ms and 310.3 ms, respectively. Thus, we do not present further details.

**Spatial Proportionality.** In Figure 12, we present the performance of our squared and radial grids techniques against the baseline algorithm for calculating the $pS\mathcal{S}(p_i)$ for all $p_i \in \mathcal{S}$ (i.e., for all pairs in $\mathcal{S}$). We see that our algorithms outperform the baseline algorithm by at least one order of magnitude for all settings and datasets. We also observe that the squared grid approach is almost always slightly faster than the radial one. Figures 12(a) and 12(c) show that the performance gap between the baseline and the grid-based algorithms increases with $K$. Figures 12(b) and 12(d) show that the size of the grid $|G|$ marginally affects the time of the grid-based algorithms. We get similar results on both datasets. Finally, in Figure 12(e), we tested the efficiency of grid-based proportionality computation on synthetically generated locations of places. For this purpose, we generated $20, \ldots, 2{,}000$ ($K$) random locations around the query location $q$ to model the retrieved set $\mathcal{S}$, using different spatial distributions: uniform and Gaussian. In the Gaussian distributions each place coordinate was generated having as mean the corresponding coordinate of $q$ and a standard deviation of either 0.25 or 0.5. Note that the baseline approach had much larger cost and was omitted from this sub-figure in order for the difference between the other methods to be easier to see.




(a) Varying $K$, **DBpedia**

(b) Varying $|G|$, **DBpedia**

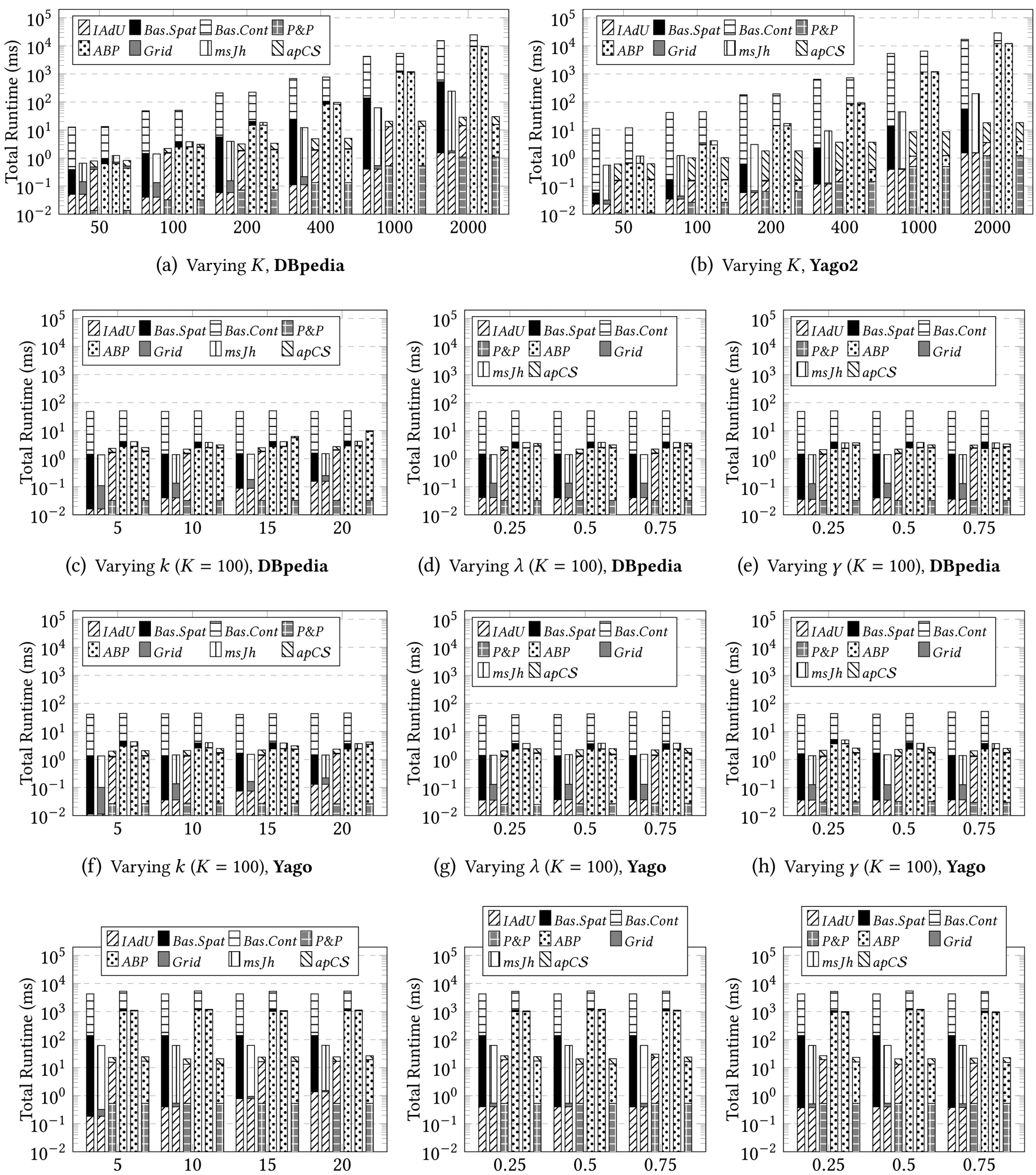
(c) Varying $K$, **Yago2**

(d) Varying $|G|$, **Yago2**

(e) Varying $K$ ($|G| = 100$), **Synthetic**

Fig. 12. Efficiency of squared and radial grid algorithms.

*9.2.2 Generic Proportionality Algorithmic Framework (Greedy Algorithms).* Next, we measure the (average of) the combined costs of the greedy (*IAdU* and *ABP*) with the prepossessing and pruning, contextual and spatial proportionality algorithms. For the proportionality calculation, we compare our optimized algorithms (i.e., *msJh*, *apCS* and grid based algorithms, which are the most efficient options) against the respective baselines. Figure 13 shows the results on DBpedia and Yago2 for different values of $K$ and $k$ (the results on Yago2 are similar and they are partly omitted for brevity). Each bar adds up the total cost of the corresponding combination. (1) The bottom part is the cost of the *P&P* (when applicable) algorithm, (2) the second part is the cost of the greedy algorithm, (3) the third part is the cost of computing spatial proportionality scores and (4) the top part is the cost of computing contextual proportionality scores (This order was chosen as to facilitate better visibility). For each parameter setting, we depict six bars. Namely, we combine each greedy algorithm with the baseline algorithms (first and fourth bars), we combine each greedy algorithm with the fastest approximated algorithms (grid and *apCS*) (third and sixth bars), which also include the *P&P* cost. Finally, we also include for comparisons against the best case of Reference [36], the combination of each greedy with *msJh* and grid based algorithms (second and fifth bars).




(a) Varying $K$, **DBpedia**

(b) Varying $K$, **Yago2**

(c) Varying $k$ ($K$ = 100), **DBpedia**

(d) Varying $\lambda$ ($K$ = 100), **DBpedia**

(e) Varying $\gamma$ ($K$ = 100), **DBpedia**

(f) Varying $k$ ($K$ = 100), **Yago**

(g) Varying $\lambda$ ($K$ = 100), **Yago**

(h) Varying $\gamma$ ($K$ = 100), **Yago**

(i) Varying $k$ ($K$ = 1000), **DBpedia**

(j) Varying $\lambda$ ($K$ = 1000) **DBpedia**

(k) Varying $\gamma$ ($K$ = 1000), **DBpedia**

Fig. 13. Efficiency of the generic algorithmic framework (total time by combining all algorithms).

According to our generic proportionality framework, we compute proportionality scores for all places and pairs just once in step 1 and then reuse these scores multiple times in step 3. However, in the case where we combine *P&P* and *apCS* (where we do not have pairwise $sC(p_i, p_j)$ scores), we calculate them during step 3 (thus, for this case, our greedy times also include this task).

We study the total time required by our framework. More precisely, we focus on large values of $K$ where required total time increases significantly. We see that our fastest combination is (1) *P&P*, *apCS* and grid based algorithms; (2) then, we have the combination of the *msJh* and grid based

Table 3. P&P Pruning Effectiveness (for Both Datasets)

| Pruning | | | | | | | |
|---|---|---|---|---|---|---|---|
| ***K*** | 20 | 50 | 100 | 200 | 400 | 1000 | 2000 |
| % | 0 | 15 | 62 | 73 | 84 | 87 | 89 |

| Pruning | | | | |
|---|---|---|---|---|
| ***k*** | 5 | 10 | 15 | 20 |
| % | 76 | 62 | 42 | 19 |

algorithms; and (3) lastly, we have the baselines. For small values of $K$, the first two combinations are similarly good (i.e., (1) *P&P*, *apCS* with grid based algorithms and (2) *msJh* with grid based algorithms). Furthermore, we also observe that *IAdU* is always faster than *ABP*; where for small values of $K$ the difference is minor but as $K$ increases, the differences increases dramatically.

In general, the respective total times increase as $K$ and $k$ increase and so are also their time differences. For instance, in DBpedia for $K = 100$ and $k = 10$, the required total times are for *IAdU* 2.19 ms, 1.40 ms, 48.35 ms, and for *ABP* 3.10 ms, 3.82 ms, 50.77 ms, whereas for $K = 2{,}000$ the required total times are for *IAdU* 28.73 ms, 242.63 ms, 15440 ms and for *ABP* 30.04 ms, 9640 ms, 24838 ms, respectively. We see that for large $K$, our fastest combination (*P&P*, *apCS*) is up to one order of magnitude faster than its counterparts.

We observe that the time of *P&P* remains a small proportion of the total time and that also increases with $K$. *IAdU* always requires small proportional times. On the other hand, *ABP* becomes very expensive for large $K$ (note that *P&P* reduces $K$ and this reduces *ABP* time). Recall that when using the *apCS* algorithm, we do not get the pairwise $sC(p_i, p_j)$ scores and we perform them during step 3; this justifies why the greedy algorithms need more time in this case.

In Table 3, we also illustrate the pruning effectiveness of *P&P* on $\mathcal{S}$. More precisely, we depict the percentage (average) of pruned objects from $\mathcal{S}$ for the default settings for both datasets. As discussed in Section 5.2, we see that as the difference between $K$ and $k$ increases the pruning also increases. Furthermore, our experimental analysis revealed that as $K$ increases, the relevance of the newly added objects significantly decreases (e.g., an object is added which is very far and minimally relevant to the query). These new objects, with very small relevance scores, are pruned by our algorithms. This further increases the pruning effectiveness of *P&P* for large $K$. In summary, for large values of $K$ this algorithm becomes very efficient.

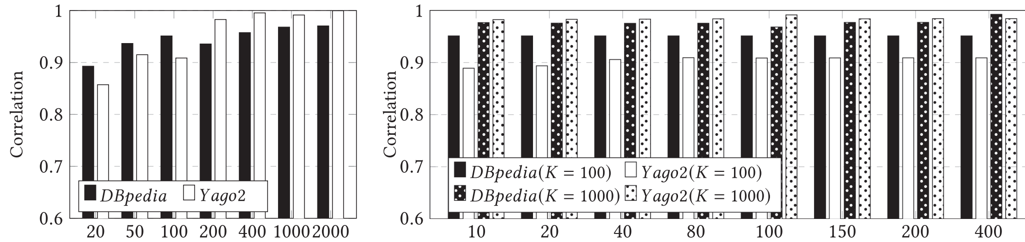
We have also tested the *P&P* algorithm in combination with our other proportionality algorithms. However, we did not get any time improvements on the total times thus, we avoided any further discussion and presentation of results. Our experimentation showed that the achieved pruning could not compensate for the additional overhead of the algorithm. More precisely, the total costs can be up to 30% higher than simply processing all places in $\mathcal{S}$ by using pre-calculated $sF(p_i, p_j)$ scores.

As expected, the weights $\lambda$ and $\gamma$ have impact only on the greedy algorithms and thus their impact remains insignificant against the total time (thus we omit further discussion due to lack of space).

In summary, the *IAdU* algorithm in combination with *P&P*, *apCS* and grid based algorithms constitute the fastest approach. The experimental results justify our focus on processing efficiently the contextual and spatial proportionality scores and use them as many times as necessary in the greedy algorithms.

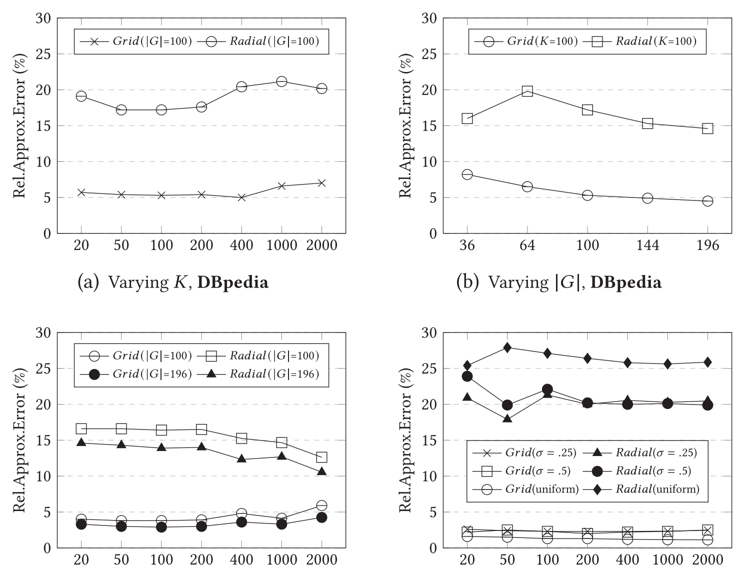
### 9.3 Approximation Quality

*9.3.1 apCS Algorithm.* We study the behavior of the approximated $apCS(p_i)$ on (1) the ranking of places in $\mathcal{S}$ and (2) the holistic score $HPF(\mathcal{R})$ of $\mathcal{R}$. In Figure 14, we compare the rankings of $\mathcal{S}$ based on $pCS(p_i)$ and $apCS(p_i)$ using the Spearman correlation metric. We can see that the two rankings are highly correlated, in most cases their correlation is above 90% (e.g., for $K > 20$ or for




(a) Varying $K$ (b) Varying $|p_i|$ ($K = 100$)

Fig. 14. Ranking correlation of $\mathcal{S}$ sorted on $pC\mathcal{S}(p_i)$ versus $apC\mathcal{S}(p_i)$.




(a) Varying $K$, **DBpedia** (b) Varying $|G|$, **DBpedia**

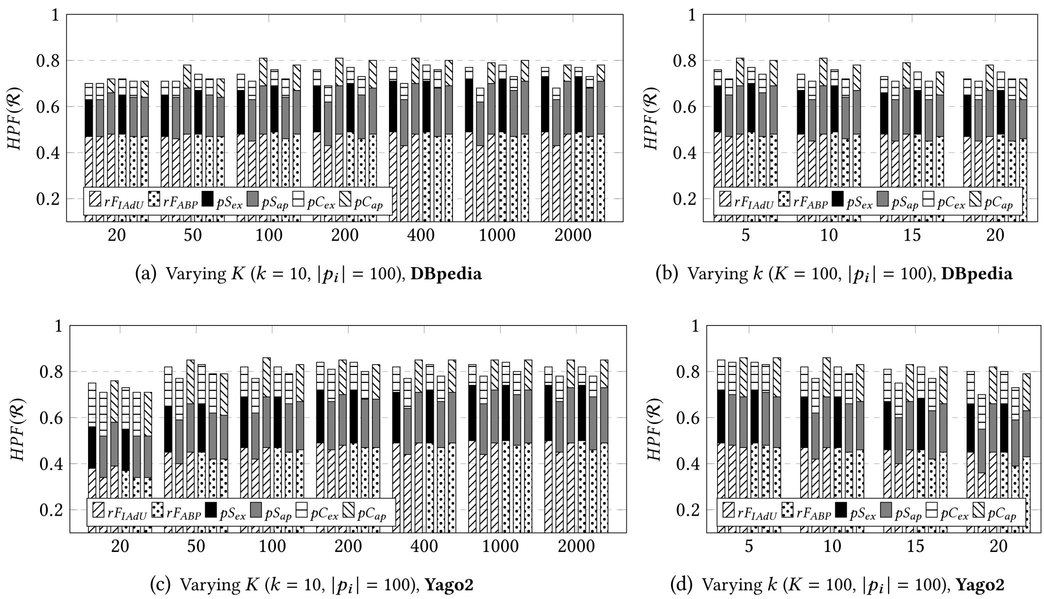
(c) Varying $K$ , **Yago2** (d) Varying $K$, **Synthetic**

Fig. 15. Effectiveness of squared and radial grid algorithms (Relative approximation error).

$|p_i| > 20$). We also studied how this approximation affects $HPF(\mathcal{R})$ (Figure 16), which we discuss shortly in Section 9.3.3.

*9.3.2 Grid Based Algorithms.* We compare the approximate $pS\mathcal{S}(p_i)$ scores for the whole $\mathcal{S}$ (i.e., $\sum_{p_i \in \mathcal{S}} pS\mathcal{S}(p_i)$) produced by the two grid approaches against the optimal one (produced by baseline). Figure 15 presents the relative approximation error of the $\sum_{p_i \in \mathcal{S}} pS\mathcal{S}(p_i)$ of the competitive approaches. We observe that the squared grid is always better than the radial grid and that $K$ does not affect this error. We also observe that increasing $|G|$ (i.e., making the grid finer) leads to a reduction of the relative approximation error and that, in general, a $|G| \approx K$ is a good choice (see Figure 15(b)). We also tried various distributions (Figure 15(d)) that also present similar results. We conclude that the squared grid with $|G| \approx K$ is an appropriate choice with a negligible error of around 5% or lower in practice. We also studied how this approximation affects $HPF(\mathcal{R})$ (Figure 16), which we discuss in Section 9.3.3.

*9.3.3 Generic Proportionality Algorithmic Framework (Greedy Algorithms).* We assess the approximation quality of the combination of the two greedy algorithms with the approximated and




(a) Varying $K$ ($k = 10$, $|p_i| = 100$), **DBpedia** (b) Varying $k$ ($K = 100$, $|p_i| = 100$), **DBpedia**

(c) Varying $K$ ($k = 10$, $|p_i| = 100$), **Yago2** (d) Varying $k$ ($K = 100$, $|p_i| = 100$), **Yago2**

Fig. 16. Approximation quality.

exact contextual and spatial algorithms. Figure 16 shows the $HPF(\mathcal{R})$ scores for these combinations, different values of $K$ and $k$ and default settings. Each bar adds up the (normalized average) total score of the corresponding weighted combination. Namely, the top part represents $\frac{1}{4\cdot(K-k)}\sum_{p_i\in\mathcal{R}} pC(p_i)$ (denoted as $pC_{ex}$ or $pC_{ap}$ obtained by using the exact or approximated $pC\mathcal{S}(p_i)$, respectively), the middle part represents $\frac{1}{4\cdot(K-k)}\sum_{p_i\in\mathcal{R}} pS(p_i)$ (i.e., $pS_{ex}$, $pS_{ap}$) and the bottom part the relevance $\frac{1}{2}\cdot\sum_{p_i\in\mathcal{R}} rF(p_i)$ (i.e., $rF_{IAdU}$, $rF_{ABP}$). Recall that we cannot obtain the optimal $HPF(\mathcal{R})$ scores due to the high computational cost required. *ABP*, in most cases, achieves (marginally) better $HPF(\mathcal{R})$ score than the respective *IAdU* combination, which reflects their (comparative) approximation quality. For instance, for the default settings, *ABP* performs 1.76% better $HPF(\mathcal{R})$ score than *IAdU* (i.e., 75.7% − 74.4% for the baseline case).

Very interestingly, the use of the approximated $apC\mathcal{S}(p_i)$ improves the $HPF(\mathcal{R})$ score (e.g., up to 9% for $K = 50$ and $k = 10$) instead of worsening it. This is because we combine it with the *P&P* algorithm. *P&P* (apart from pruning fruitless results) also ranks them on $HPF_{lb}(p_i)$ and then feed them in this order to the greedy algorithms. The use of this ranking (instead of the $rF(p_i)$ ranking) appeared to improve greedy algorithms' performance with respect to the final $HPF(\mathcal{R})$ score as $HPF_{lb}(p_i)$ can be significantly larger than $rF(p_i)$.

The approximation compromise of the grid based algorithm is minor. For the default settings, the difference on $HPF(\mathcal{R})$ scores using the exact spatial scores against the approximated spatial scores on *IAdU* and *ABP* is 4.8% and 5.9%, respectively. On the other hand, the combination of the grid based algorithm with the $apC\mathcal{S}$ and *P&P* algorithms achieves an improvement over the baseline $HPF(\mathcal{R})$ scores of 7.38% on *IAdU* and 2.66% on *ABP*.

Regarding the parameters $K$ and $k$, we observe a consistent overall effect on the resulting scores. Increasing $K$ generally correlates with an increase in $HPF(\mathcal{R})$ since there are more places available to construct the best possible $\mathcal{R}$. This, however, does not hold for all cases as the greedy algorithms do not guarantee to return the best result. Increasing $k$ causes a decrease in $HPF(\mathcal{R})$ in all cases. This is a reasonable result since each additional place added in $\mathcal{R}$ contributes to $HPF(\mathcal{R})$

less than the already selected places. The $\lambda$ and $\gamma$ weights have marginal impact on the relative approximation quality (thus details are omitted for brevity).

### 9.4 User Evaluation

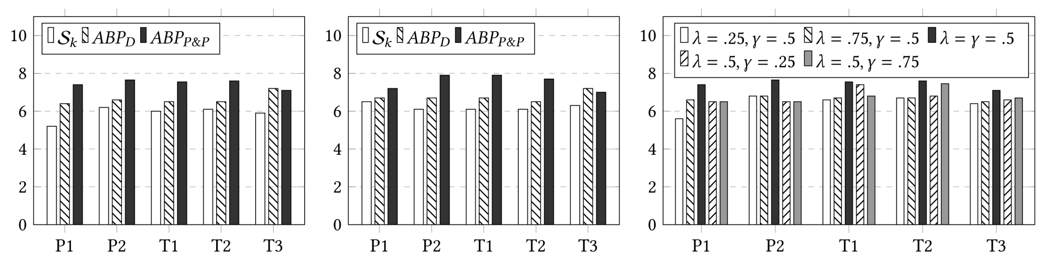
We also conducted a user evaluation (i.e., user preference and usability testing), which confirms the preference of users to proportional results. We asked help from ten evaluators (none of them was involved in this article). First, we familiarized them with the query concepts and relevance metrics. We also explained to them the concepts of proportionality and diversity; to avoid any bias, we avoided to discuss their advantages or disadvantages. Then, we presented to them ten random queries from both datasets and their results according to the three alternative frameworks. Namely, $\mathcal{S}_k$ (i.e., the top-$k$ places in $\mathcal{S}$ with the largest $rF(p_i)$), $ABP_D$ (i.e., diversification results produced by $ABP$ [4], since $ABP$ was shown to have superior approximation quality to $IAdU$) and our proportional $ABP_{P\&P}$ (i.e., proportional results produced by the combination of $ABP$ with $apC\mathcal{S}$ and $P\&P$, since this combination was shown superior among other options). For each task, we asked them to give a score in a scale of one to ten. In order to assist evaluators with their tasks, we also presented a map with the places (Figure 18 illustrates such examples of maps with places), their contextual sets and useful statistics (for each query). We presented the output of each method in a random order (to avoid any bias).

*9.4.1 User Preference Study.* In this study, we asked evaluators to evaluate and express their preference w.r.t. (**P1**) the general content of results (by considering how representative and informative they are) and (**P2**) their ranking. The P1 and P2 bars in Figure 17(a) and (b) average the evaluators' preference scores of the three methodologies, for the two criteria (i.e., general content and ranking), for $k = 10$ and $k = 20$ (using the default settings). For the first criterion (general content), we observe that the users prefer proportional, then diversified and lastly non diversified results. For the second criterion (ranking), users prefer proportional and diversified results. For instance for $k = 10$, the average scores of $\mathcal{S}_k$, $ABP_D$, $ABP_{P\&P}$ on the two tasks are 5.7, 6.5, and 7.5, respectively. The study revealed that the top places are typically proportional at the same time facilitating both diversity and representation of $\mathcal{S}$; whereas, only some bottom results had some similarity to previous ones. e.g., the top five places are proportional and repetitions appear in the bottom 5 places (e.g., additional museums). This type of bird's eye view is preferable by users.

*9.4.2 Usability Test.* We conducted a comparative study of the usability of the three paradigms. Usability is the ease of use and learnability of a human-made object; namely, how efficient it is to use (for instance, whether it takes less time to accomplish a particular task), how easy it is to learn and whether it is more satisfying to use.[3] We gave them three tasks to complete (for each query and paradigm) and asked them to give a score and also to justify their answers (where possible). Namely, to score them considering (1) the ease of accomplishing each task, (2) how easy and (3) satisfying are to learn and use.

The three tasks were about the understanding and the extraction of information about the queries' results and the entire $\mathcal{S}$. Task 1 (**T1**) "How easily can you infer the area with many collocated places of interest?" For instance in Stockholm, how easily can you infer that Gamla Stan is an area with many collocated museums; so someone can visit this area and can visit more than one museums. Task 2 (**T2**) "How easily can you infer the most representative type of places in the area?" e.g., an arts or history museum in Stockholm. Task 3 (**T3**) "How easily can you infer at least three different types of places of interest in the area?" e.g., so someone can choose from all types of museums in Stockholm.

[3] www.wikipedia.org/wiki/Usability.




(a) Evaluation Tasks, $k = 10$ (b) Evaluation Tasks, $k = 20$ (c) Evaluation Tasks on $ABP_{P\&P}$, $k = 10$, varying $\lambda, \gamma$

Fig. 17. User evaluation and usability test.

The T1–T3 bars in Figure 17(a) and (b) average the evaluators' usability scores of the three methods per query and per task. The results show that evaluators preferred firstly proportional, then diversified and lastly non-diversified results for both datasets. For instance for $k = 10$, the average scores of $\mathcal{S}_k$, $ABP_D$, $ABP_{P\&P}$ on the three tasks are 6, 6.7, and 7.5, respectively. The evaluators also provided justifications for their scores. They explained that, in general, they prefer the concept of proportionality as it also considers frequent properties; which is a property other types do not consider. They found diversification very useful in covering the most diverse places (addressing T3); however, they pointed out that rare but important elements may appear, which again can be to some extend misleading. They found the non-diversified results more misleading as very important and relevant places are too dominant in them.

Figure 17(c) depicts the preference of users for the various values of $\lambda$ and $\gamma$ for $k = 10$ using the $ABP_{P\&P}$ algorithm. Other settings also gave interestingly good results; however, in most cases results from the default setting were more preferable.

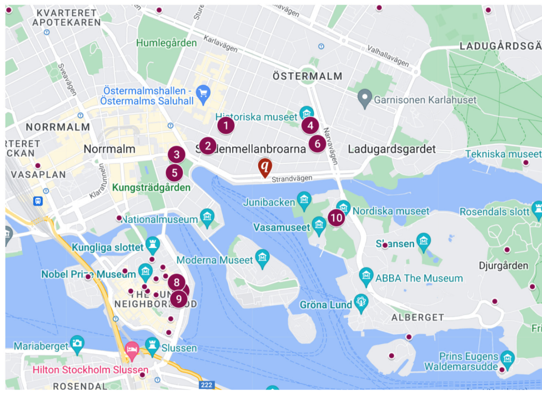
## 10 CONCLUSIONS

In this work, we extend spatial keyword search to support proportional selection of the retrieved places. Our framework combines relevance and proportionality, w.r.t. both context and location. After proving the hardness of the problem, we identify the bottlenecks of proportional selection and propose techniques that greatly reduce its computational cost in practice. We use our methods as modules of two greedy algorithms (*IAdU* and *ABP*). Our experiments on real data verify the approximation quality and efficiency of our algorithms and confirm that our framework is preferred by human evaluators. More precisely, the greedy *IAdU* algorithm in combination with the $apC\mathcal{S}$ and squared grid algorithms appears to be the best choice for our paradigm as it is the fastest of all options and at the same time achieves the best $HPF(\mathcal{R})$.

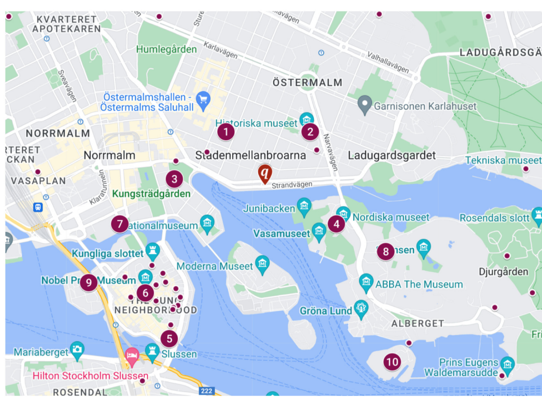
In our future work, we will study alternative scoring functions for the contextual and spatial search components (e.g., road network distance in place of Euclidean distance). Another direction of future work is the study of fairness, as our algorithms can also facilitate fairness. For instance, consider areas associated with demographic or political groups, we can use our contextual or spatial proportionality algorithms as to ensure fair representation of places in such areas.

# APPENDIX

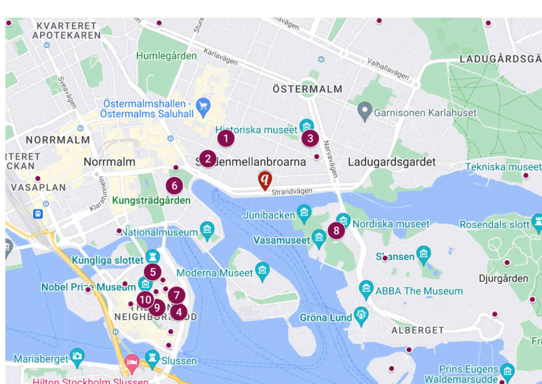
## A USER EVALUATION (EXAMPLES OF PLACES ON MAPS)




(a) $S_k$




(b) $ABP_D$




(c) $ABP_{P\&P}$

Fig. 18. The Top−10 results for the keyword *historical* on DBpedia using the default setting (big circles indicate selected places (with a number indicating their ranking) and small circles indicate unselected places).

## REFERENCES


[1] Pritom Ahmed, Mahbub Hasan, Abhijith Kashyap, Vagelis Hristidis, and Vassilis J. Tsotras. 2017. Efficient computation of top-k frequent terms over spatio-temporal ranges. In *Proceedings of the 2017 ACM International Conference on Management of Data SIGMOD*. 1227–1241.

[2] Ritesh Ahuja, Nikos Armenatzoglou, Dimitris Papadias, and George J. Fakas. 2015. Geo-social keyword search. In *Proceedings of the Advances in Spatial and Temporal Databases: 14th International Symposium SSTD*. 431–450.

[3] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. 2007. Scaling up all pairs similarity search. In *Proceedings of the 16th International Conference on World Wide Web*. 131–140.

[4] Zhi Cai, Georgios Kalamatianos, Georgios J. Fakas, Nikos Mamoulis, and Dimitris Papadias. 2020. Diversified spatial keyword search on RDF data. *VLDB Journal* 29, 5 (2020), 1171–1189.

[5] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. A text feature based automatic keyword extraction method for single documents. In *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018 (Lecture Notes in Computer Science, Vol. 10772)*. Springer, 684–691.

[6] Lisi Chen, Gao Cong, Christian S. Jensen, and Dingming Wu. 2013. Spatial keyword query processing: An experimental evaluation. *PVLDB* 6, 3 (2013), 217–228.

[7] Lisi Chen, Shuo Shang, Chengcheng Yang, and Jing Li. 2020. Spatial keyword search: A survey. *GeoInformatica* 24, 1 (2020), 85–106.

[8] Shiwen Cheng, Anastasios Arvanitis, Marek Chrobak, and Vagelis Hristidis. 2014. Multi-query diversification in microblogging posts. In *Proceedings of the International Conference on Extending Database Technology*. 133–144.

[9] Shiwen Cheng, Marek Chrobak, and Vagelis Hristidis. 2016. Slowing the firehose: Multi-dimensional diversity on social post streams. In *Proceedings of the International Conference on Extending Database Technology*. 17–28.

[10] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment* 2, 1 (2009), 337–348.

[11] Van Dang and W. Bruce Croft. 2012. Diversity by proportionality: An election-based approach to search result diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 65–74.

[12] Elena Demidova, Peter Fankhauser, Xuan Zhou, and Wolfgang Nejdl. 2010. DivQ: Diversification for keyword search over structured databases. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 331–338.

[13] Aggeliki Dimitriou, Dimitri Theodoratos, and Timos K. Sellis. 2015. Top-k-size keyword search on tree structured data. *Information Systems* 47 (2015), 178–193.

[14] Georgios J. Fakas. 2008. Automated generation of object summaries from relational databases: A novel keyword searching paradigm. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering Workshop*. IEEE Computer Society, 564–567.

[15] Georgios J. Fakas. 2011. A novel keyword search paradigm in relational databases: Object summaries. *Data & Knowledge Engineering* 70, 2 (2011), 208–229.

[16] Georgios J. Fakas, Yilun Cai, Zhi Cai, and Nikos Mamoulis. 2018. Thematic ranking of object summaries for keyword search. *Data & Knowledge Engineering* 113 (2018), 1–17.

[17] Georgios J. Fakas and Zhi Cai. 2009. Ranking of object summaries. In *Proceedings of the 2009 IEEE 25th International Conference on Data Engineering*. 1580–1583.

[18] Georgios J. Fakas and Zhi Cai. 2018. Object summaries for keyword search. *Encyclopedia with Semantic Computing and Robotic Intelligence* 2, 2 (2018), 1750002:1–1750002:20.

[19] Georgios J. Fakas, Zhi Cai, and Nikos Mamoulis. 2011. Size-$l$ object summaries for relational keyword search. *International Conference on Very Large Data Bases* 5, 3 (2011), 229–240.

[20] Georgios J. Fakas, Zhi Cai, and Nikos Mamoulis. 2014. Versatile size-$l$ object summaries for relational keyword search. *IEEE Transactions on Knowledge and Data Engineering* 26, 4 (2014), 1026–1038.

[21] Georgios J. Fakas, Zhi Cai, and Nikos Mamoulis. 2015. Diverse and proportional size-$l$ object summaries for keyword search. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 363–375.

[22] Georgios J. Fakas, Zhi Cai, and Nikos Mamoulis. 2016. Diverse and proportional size-l object summaries using pairwise relevance. *VLDB Journal* 25, 6 (2016), 791–816.

[23] Georgios J. Fakas, Ben Cawley, and Zhi Cai. 2011. Automated generation of personal data reports from relational databases. *Journal of Information & Knowledge Management* 10, 2 (2011), 193–208.

[24] Georgios J. Fakas, Antonis C. Kakas, and Christos N. Schizas. 2004. Electronic roads: Intelligent navigation through multi-contextual information. *Knowledge and Information Systems* 6, 1 (2004), 103–124.

[25] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *Proceedings of the International Council for Open and Distance Education*. IEEE Computer Society, 656–665.

[26] Marios Hadjieleftheriou, George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. 2003. On-line discovery of dense areas in spatio-temporal databases. In *Proceedings of the 17th International Symposium on Spatial and Temporal Databases*, Vol. 2750. 306–324.

[27] Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. 2006. Indexing spatiotemporal archives. *VLDB Journal* 15, 2 (2006), 143–164.

[28] Jungkyu Han and Hayato Yamana. 2017. Geographical diversification in POI recommendation: Toward improved coverage on interested areas. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 224–228.

[29] Jayant R. Haritsa. 2009. The KNDN problem: A quest for unity in diversity. *IEEE Data Engineering Bulletin* 32, 4 (2009), 15–22.

[30] Mahbub Hasan, Abhijith Kashyap, Vagelis Hristidis, and Vassilis J. Tsotras. 2014. User effort minimization through adaptive diversification. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 203–212.

[31] Refael Hassin, Shlomi Rubinstein, and Arie Tamir. 1997. Approximation algorithms for maximum dispersion. *Operations Research Letters* 21, 3 (1997), 133–137.

[32] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence* 194 (2013), 28–61.

[33] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. 2003. Efficient IR-style keyword search over relational databases. In *Proceedings of the International Conference on Very Large Data Base*. 850–861.

[34] Vagelis Hristidis and Yannis Papakonstantinou. 2002. Discover: Keyword search in relational databases. In *Proceedings of the International Conference on Very Large Data Base*. 670–681.

[35] Anoop Jain, Parag Sarda, and Jayant R. Haritsa. 2004. Providing diversity in k-nearest neighbor query results. In *Proceedings of the Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference*. 404–413.

[36] Georgios Kalamatianos, Georgios J. Fakas, and Nikos Mamoulis. 2021. Proportionality in spatial keyword search. In *Proceedings of the 2021 International Conference on Management of Data SIGMOD*. ACM, 885–897.

[37] Mehdi Kargar, Aijun An, and Xiaohui Yu. 2014. Efficient duplication free and minimal keyword search in graphs. *Proceedings of the IEEE Transactions on Knowledge and Data Engineering* 26, 7 (2014), 1657–1669.

[38] George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. 2017. Mobile object indexing. In *Proceedings of the Encyclopedia of GIS*. 1256–1266.

[39] Wangchao Le, Feifei Li, Anastasios Kementsietsidis, and Songyun Duan. 2014. Scalable keyword search on large RDF data. *IEEE Transactions on Knowledge and Data Engineering* 26, 11 (2014), 2774–2788.

[40] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets* (2nd ed.). Cambridge University Press.

[41] Michael Levandowsky and David Winter. 1971. Distance between sets. *Nature* 234, 5323 (1971), 34–35.

[42] S. S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. 1991. *Facility Dispersion Problems: Heuristics and Special Cases*. 431–450.

[43] Dimitris Sacharidis, Paras Mehta, Dimitrios Skoutas, Kostas Patroumpas, and Agnès Voisard. 2018. Selecting representative and diverse spatio-textual posts over sliding windows. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. 17:1–17:12.

[44] Jieming Shi, Dingming Wu, and Nikos Mamoulis. 2016. Top-k relevant semantic place retrieval on spatial RDF data. In *Proceedings of the 2016 International Conference on Management of Data*. 1977–1990.

[45] A. B. Siddique, Ahmed Eldawy, and Vagelis Hristidis. 2019. Euler++: Improved selectivity estimation for rectangular spatial records. In *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*. 4129–4133.

[46] Souvik Brata Sinha, Xinge Lu, and Dimitri Theodoratos. 2018. Personalized keyword search on large RDF graphs based on pattern graph similarity. In *Proceedings of the 22nd International Database Engineering & Applications Symposium*. 12–21.

[47] Dimitrios Skoutas, Dimitris Sacharidis, and Kostas Patroumpas. 2018. Efficient progressive and diversified top-k best region search. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 299–308.

[48] Kostas Stefanidis, Marina Drosou, and Evaggelia Pitoura. 2010. PerK: Personalized keyword search in relational databases through preferences. In *Proceedings of the 13th International Conference on Extending Database Technology*. 585–596.

[49] Julia Stoyanovich, Ke Yang, and H. V. Jagadish. 2018. Online set selection with fairness and diversity constraints. In *Proceedings of the EDBT Conference*. 241–252.

[50] Jiayu Tang and Mark Sanderson. 2010. Evaluation and user preference study on spatial diversity. In *Proceedings of the Advances in Information Retrieval: 32nd European Conference on IR Research*, Vol. 5993. 179–190.

[51] M. G. Thushara, Tadi Mownika, and Ritika Mangamuru. 2019. A comparative study on different keyword extraction algorithms. In *Proceedings of the International Conference on Computing Methodologies and Communication (ICCMC)*. 969–973.

[52] Marc Van Kreveld, Iris Reinbacher, Avi Arampatzis, and Roelof Van Zwol. 2005. Multi-dimensional scattered ranking methods for geographic information retrieval. *GeoInformatica* 9, 1 (2005), 61–84.

[53] Marcos R. Vieira, Humberto L. Razente, Maria C. N. Barioni, Marios Hadjieleftheriou, Divesh Srivastava, Caetano Traina, and Vassilis J. Tsotras. 2011. On query result diversification. In *Proceedings of the IEEE 27th International Conference on Data Engineering*. 1163–1174.

[54] Shuai Wang, Zhiyuan Chen, Bing Liu, and Sherry Emery. 2016. Identifying search keywords for finding relevant social media posts. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Mar. 2016), 3052–3058.

[55] Lin Wu, Yang Wang, John Shepherd, and Xiang Zhao. 2013. An optimization method for proportionally diversifying search results. In *Proceedings of the Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD (1)*, Vol. 7818. 390–401.

[56] Xiaolu Xing, Chaofeng Sha, and Junyu Niu. 2017. Improving topic diversity in recommendation lists: Marginally or proportionally? In *Proceedings of the Web and Big Data: First International Joint Conference, APWeb/WAIM (2)*, Vol. 10367. 142–150.

[57] Chengyuan Zhang, Ying Zhang, Wenjie Zhang, Xuemin Lin, Muhammad Aamir Cheema, and Xiaoyang Wang. 2014. Diversified spatial keyword search on road networks. In *Proceedings of theAdvances in Database Technology-EDBT 2014: 17th International Conference on Extending Database Technology*. 367–378.

[58] Xin Zheng, Aixin Sun, Sibo Wang, and Jialong Han. 2017. Semi-supervised event-related tweet identification with dynamic keyword generation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM*. ACM, 1619–1628.

[59] Panfeng Zhou, Donghui Zhang, Betty Salzberg, Gene Cooperman, and George Kollios. 2005. Close pair queries in moving object databases. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems, GIS*. 2–11.

Received 16 June 2022; revised 27 December 2022; accepted 12 March 2023