Example-based Spatial Search at Scale

Hanyuan Zhang*¶‖, Siqiang Luo† , Jieming Shi‡, Jing Nathan Yan§, Weiwei Sun*¶‖
*School of Computer Science, Fudan University, ¶Shanghai Key Laboratory of Data Science
‖ Shanghai Institute of Intelligent Electronics & Systems
†Nanyang Technological University, ‡The Hong Kong Polytechnic University, §Cornell University
{*zhanghy20,wwsun*}*@fudan.edu.cn, siqiang.luo@ntu.edu.sg, jieming.shi@polyu.edu.hk, jy858@cornell.edu*

***Abstract*—Searching spatial objects is a fundamental task in spatial services such as online maps. Traditional search methods are based on *filtering conditions*, burdening users to specify their requirements. This paper focuses on spatial search via examples. Particularly, the user can specify an *example*, which is a set of objects of interest, and the purpose is to find a list of results, each containing a set of objects with similar properties to the given example. We conducted a user study, showing that a search interface based on examples can effectively complement existing approaches. However, the existing example-based search is not scalable, hindering its applications to larger datasets. To address this challenge, we propose two new algorithms, namely HSP and LORA, to efficiently answer example-based spatial queries. HSP is an algorithm based on a hierarchical partitioning of the search space, and it achieves up to 20 times faster than the state-of-the-art algorithm. LORA further improves the efficiency, running up to 5000 times faster than the state-of-the-art algorithm. We present a systematic evaluation to demonstrate the efficacy of our algorithms.**

***Index Terms*—Spatial Query; Data Mining**

I. Introduction

Large-scale spatial services are widely adopted in real life. Examples include trip planning [1]–[3], POI (Point of Interest) recommendations [4], [5] and geo-social analytics [6], [7]. One prevalent operation in spatial services is to extract several objects with desired attributes and relative spatial locations. We illustrate this with several daily life examples.

Example 1. Peter is a manager working in the financial district of a city. He is searching for an apartment and a daycare center for his three-year-old daughter. The daycare centers are typically located in another functional district of the city, as shown in Figure 1. Peter needs to send his daughter to the daycare center early in the morning and go to work right after. He usually gets some food on the way. In addition, he needs to pick up his daughter in the afternoon and then return to work.

Example 2. Ben is a student in Hong Kong. In his daily routine, he goes to the gym by subway from his apartment to take some exercise, and then leaves the gym and goes to school. He usually buys some food on the way. However, with the pressure of increasing rent, Ben plans to move to another apartment that fits his daily routine with the limited budget.

The above examples show the sophisticated specifications of the search intention. In the first example, Peter is searching for an apartment and a daycare center, such that (1) the daycare center is in-between the apartment and his workplace to fit his routines; (2) there is a takeaway on the way from the daycare center to his workplace, and (3) the daycare center is close to his workplace for his convenience to pick up his daughter in the afternoon. Similarly, in the second example, Ben is searching for an apartment and a gym such that (1) the gym is in-between his apartment and school; (2) there is a takeaway from the gym to his school, and (3) the apartment's rental should be within his budget.

†Siqiang Luo is the corresponding author.

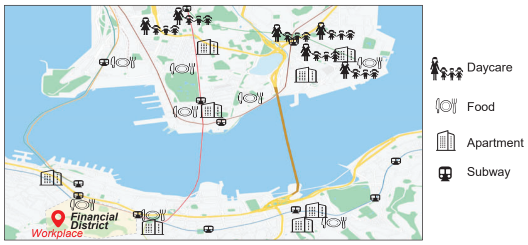


Fig. 1: Illustrating Example 1.

To assist a user in expressing her search intention, existing approaches require her to specify detailed requirements as filtering conditions [8], or specially optimize an objective function for a solution [9], [10]. While these methods are feasible to address the search problem, the sophistication of specifying requirements burdens users in the interaction with the interfaces. Such concerns were raised and discussed in a seminal work [11], where a user study is conducted, showing the superiority of using example-based queries in terms of user-friendliness. Specifically, they show that a significant portion of users alternatively prefers using *examples* to deliver their search interest, compared with filtering conditions.

Figure 2 shows a map service incorporated with the example-based query. The user can input a set of objects as a desired example to the system, e.g., by clicking objects in the map service in the "example selection" panel. The system captures the implicit user intention and searches for desired objects similar to the example. In Figure 2, the user wants to find a set of objects (Apartment, Daycare, Food) with specific relative co-locations (shown as black dotted lines in the "example selection") and corresponding attributes (e.g., reviews) as shown in the "attribute panel". After the user specifies the example, the desired attribute values associated with the example are extracted. The example-based system then returns candidate objects that constitute similar co-location and attributes, as shown in the right panel.

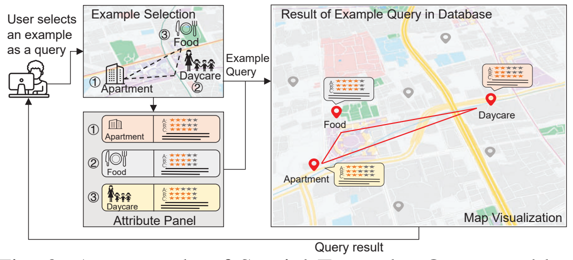


Fig. 2: An example of Spatial Exemplar Query problem.

There are three prominent advantages of the example-based search. The first advantage is that an example set of objects concisely carries information about the relative locations of the objects and desired attributes. This gives the potential power for the system to "guess" the search intention via the example, shifting the burden from the user to the system. Another advantage is that the example is usually available in hand from the user's experience. In Example 2, it is desired to create an example that includes Ben's current apartment, school, as well as his frequently visited gym and food shop, because Ben's future remains the same as the current one. Furthermore, compared with the hard constraints required by filtering conditions that may rule out all candidates, the example-based search guarantees to return the results that are the most similar to the example.

It is important to note that we do *not* argue to completely replace traditional approaches with example-based methods. Instead, our main purpose is to outline and emphasize the potential novelty of the example-based search in spatial queries, which complements existing work and, arguably, opens up a new research direction for spatial search. For this purpose, we conducted a volunteer-based survey on college students using Examples 1 and 2, and asked for their preferred interfaces. The result shows that the example-based interface is more preferred in those scenarios like Examples 1 and 2, but one method cannot completely replace the other, as both interfaces have been preferred by different users. We also observe that some users may prefer the traditional filtering-based approaches. However, even among these users, we observe that 83.6% of them consider using example-based search as a complementary approach. We quote the following representative reasons collected from the survey for choosing the example-based interface.

"*The location of the gym, university, potential apartment, and the subway can be seen clearly on the map for me to make the decision based on the example-based search. One just needs to do some click on the screen.*"

"*It is more convenient to compare the different candidates among the map with everything I care about visible*"

The user study concludes that providing a spatial service with an example-based search interface, in addition to the traditional interface, can enhance its user-friendliness. We refer interested readers to Section IV-C for more details.

Challenges. The pilot study in [11] formulates the example-based search as SEQ (Spatial Exemplar Query). Given an integer k, SEQ takes an example set of user-interested objects, and searches for the k most relevant results to the given example, where a result is an object set with the same size as the example. The results are ranked based on their similarities to the given example, considering both the relative co-locations (called spatial similarity) and properties of the objects (called attribute similarity).

Unfortunately, the computation of SEQ is prohibitive for large datasets, because in the worst case it requires an exhaustive search among all possible combinations of objects. In particular, the method presented in [11] spends more than 24 hours answering a query for a dataset containing $500,000$ objects. However, a real-world location-based service often needs to handle up to 10^7 objects. The high latency hinders the application of SEQ to real-world location-based services. Furthermore, the spatial similarity defined in SEQ relies heavily on the similarities based on the shape formed by the objects, and thus SEQ may return a set of objects which form a similar shape with the example, but their absolute distances can be drastically different. As a result, SEQ cannot guarantee to rule out those unsatisfactory candidate object pairs with dramatically larger object distances than that of the example.

Our Contributions. To address these issues, we make the following contributions.

• (Problem) We reformulate the example-based search by adding a constraint that any output object set is allowed at most a constant factor β of the distance amplification compared with the given example, where the amplification is defined based on the coordinates of the objects (see Section II). We call this problem CSEQ (Constraint SEQ). CSEQ is often more geared to practical needs, and it allows more room for improving the efficiency of the example-based search. We further discuss variants of CSEQ, taking into account the practical requirements.

• (Algorithm) We present an approximate algorithm, LORA, to answer CSEQ. LORA improves the computation efficiency by up to 5000 times compared with the state of the art, while retaining high accuracy. This allows the example-based search to be applied in practice. For example, in a dataset with $500,000$ objects, the state of the art takes more than 24 hours to compute, whereas LORA only spends around 16 seconds, with an MAE error as low as 0.002 (that indicates negligible difference to the exact results).

• (Theory) Given any constant $\epsilon > 0$, LORA can be tuned to have an approximation ratio $1 + \epsilon$ and an additive error $\alpha \cdot \epsilon$, where α is a constant in $(0, 1)$. Furthermore, from our empirical analysis, we found that the result quality of LORA (i.e., the closeness of the returned results to the exact results) can be even higher than that indicated by the theoretical analysis. We also analyze the complexity of LORA, which demonstrates the superb efficiency of LORA.

• (Evaluation) We conducted comprehensive experiments on real datasets and an additional user study. Our results show that LORA can achieve significant performance improvement upon baselines spanning a wide spectrum of parameter settings. The user study signals that example-based search is in general helpful from the perspective of user experience.

II. Problem Definitions

A. Data Model

Following previous studies [12]–[16], we model an object as a point located in the Euclidean space, and we will use the terms *object* and *point* interchangeably. Each object has an associated *category* such as "restaurants", "apartment" and "supermarket", which describes the object's nature. In addition, each object p has a list of attribute-value pairs $\{(a_1, u_1), \dots, (a_h, u_h)\}$. For example, an object representing a hotel has attributes such as "rating", "daily cost" and "capacity". For an object p with attributes $A_p = \{a_1, \dots, a_h\}$, its Attribute Vector is denoted by $U_p = \{u_1, \dots, u_h\}$, where u_i is the value of attribute a_i. A *tuple* is a set of objects. We use m to denote the tuple size. The distance vector of a tuple $t = \{p_1, p_2, ..., p_m\}$ is defined as $V_t = \{d(p_1, p_2),\ d(p_1, p_3),\ \dots,\ d(p_1, p_m),\ d(p_2, p_3), \dots,\ d(p_2, p_m),\ \dots,\ d(p_{m-1}, p_m)\}$, where $d(x, y)$ denotes a distance metric between two objects x and y. By default, we use the Euclidean distance, yet applying other metrics such as traveling distances is possible.

B. SEQ and CSEQ

SEQ (Spatial Exemplar Query) and CSEQ (Constraint SEQ) are built upon similarities between tuples. We are looking to compare two size-m tuples t_1 and t_2 which are in the same category, which means that the i-th ($1 \le i \le m$) items of t_1 and t_2 are in the same category. The *spatial similarity* of t_1 and t_2 is computed by the Cosine similarity (denoted as $cos(\cdot, \cdot)$) of their corresponding distance vectors:

$$\texttt{SIM}_s(t_1, t_2) = cos(V_{t_1}, V_{t_2}) \tag{1}$$

If two points p_1, p_2 are of the same category, we measure their attribute similarity by

$$\texttt{SIM}_a(p_1, p_2) = cos(U_{p_1}, U_{p_2}) \tag{2}$$

For tuples t_1 and t_2, their attribute similarity is

$$\texttt{SIM}_a(t_1, t_2) = \frac{1}{m} \sum_{i=1}^{m} \texttt{SIM}_a(t_1[i], t_2[i]) \tag{3}$$

where $t_1[i]$ (resp. $t_2[i]$) denotes the i-th object in tuple t_1 (resp. t_2). Given a tuning parameter $\alpha \in [0, 1]$, we combine $\texttt{SIM}_a$ and $\texttt{SIM}_s$ to define the *Tuple Similarity*:

$$\texttt{SIM}(t_1, t_2) = \alpha \cdot \texttt{SIM}_s(t_1, t_2) + (1 - \alpha) \cdot \texttt{SIM}_a(t_1, t_2) \tag{4}$$

Given an integer k and example t^*, the SEQ problem in [11] aims to discover top-k similar tuples to t^*. While this definition is natural, we observe that it ignores the absolute object distances which can be important in applications. For example, users may be favorable to objects that construct similar shapes with the example, conditioned on that the distances between objects are reasonably bounded. To mitigate this problem, we revise the original SEQ problem by adding a norm constraint. Particularly, given a constant $\beta \ge 1$, we say two tuples t_1 and t_2 satisfy the β-norm constraint, if and only if $\frac{1}{\beta} \le \frac{||V_{t_1}||}{||V_{t_2}||} \le \beta$, where V_{t_1} and V_{t_2} are the distance vectors of t_1 and t_2, and $||\cdot||$ denotes the 2-norm measure. We define a norm-constrained SEQ, dubbed as CSEQ, as follows.

TABLE I: Frequently used notations.

Notation	Description
p	A point of interest with attributes
m	The number of points in the example
t	A tuple of points
t^*	A given example tuple
V_t	The distance vector of t
$\|\|V_t\|\|$	The norm of distance vector of t
β	The norm constraint
S	A subspace divided from the entire space
S^c / S^a	A core/auxiliary subspace
S^u	The union of S^c and S^a, called an ac-subspace
g	A cell divided from S
G_j^a	A list of points of category a in cell g_j
ξ	A sampling parameter in LORA

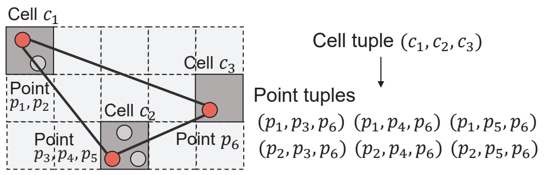
Definition 1 (CSEQ): Given an integer $k > 0$, a constant $\beta \ge 1$, and a size-m example tuple t^*. The β-Constraint Spatial Exemplar Query (CSEQ) returns top-k similar size-m tuples $t_1, \cdots, t_k$ to t^* with respect to the tuple similarity, such that: 1) t_i ($i \in [1, k]$) has the same category as t^*; 2) t_i and t^* satisfy the β-norm constraint.

Remarks on Problem Variations. CSEQ can be adapted to various applications or user needs. First, setting β to infinity converts CSEQ to SEQ. Hence, any algorithm that can answer CSEQ can also answer SEQ. CSEQ can also be easily extended to handle applications where certain points in the result tuples have to be fixed. For example, in Example 1 in Section I, the workplace is a fixed location that must also appear in the result. We name this problem as CSEQ-FP (CSEQ with fixed points). From the computation perspective, CSEQ-FP virtually reduces the enumeration cost, making computation easier. Therefore, for ease of presentation, we will mainly illustrate how to design algorithms for CSEQ, which is the general form of SEQ and CSEQ-FP. In experiments, nevertheless, we evaluate the effectiveness of the algorithms for all of SEQ, CSEQ, and CSEQ-FP. Moreover, it is also possible to extend a CSEQ algorithm to cover scenarios where some distance pairs are not interested to be considered because those pairs can be simply skipped in the computation.

C. State of the Art in Brief

We refer to the state-of-the-art algorithm in [11] as DFS-Prune Given an example tuple $t^* = \{p_1^*, p_2^*, \dots, p_m^*\}$, DFS-Prune examines all candidate tuples $t = \{p_1, p_2, \dots, p_m\}$ where p_i has the same category as p_i^*. For each candidate tuple t, DFS-Prune progressively examines its prefixes in an ascending order: $\{p_1\}$, $\{p_1, p_2\}$, …, $\{p_1, p_2, \dots, p_m\}$. For each prefix of t, DFS-Prune derives an upper bound of $\texttt{SIM}(t, t^*)$, which can be used to determine whether t is unpromising to become the top-k tuples. (Namely, t is unpromising if the upper bound is already smaller than the current k-th maximum similarity value.) Once t is pruned based on a certain prefix, further examination of longer prefixes will be skipped.

To facilitate the prefix-based pruning, DFS-Prune enumerates a tuple t by selecting t's points in turn for each dimension. For each dimension, the points are ordered in a descending order of their attribute similarities to the respective point



Cell c_1
Cell c_3
Cell c_2
Point p_1, p_2
Point p_3, p_4, p_5
Point p_6
Cell tuple (c_1, c_2, c_3)
Point tuples
(p_1,p_3,p_6) (p_1,p_4,p_6) (p_1,p_5,p_6)
(p_2,p_3,p_6) (p_2,p_4,p_6) (p_2,p_5,p_6)

Fig. 3: Point tuples in a cell tuple.

in the example t^*. Suppose that certain prefix $\{p_1, ..., p_i\}$ has been determined during enumeration, and their attribute similarities to the respective point in example t^* are $\{r_1, ..., r_i\}$ respectively. Then for any candidate tuple that has not been enumerated, its attribute similarity with respect to t^* has been shown to be upper bounded by $m - i + \sum_{j=1}^{i} r_i$.

DFS-Prune also employs an upper bound of spatial similarity as follows. Let the distance vector of any tuple t with the same prefix $\{p_1, p_2, ..., p_i\}$ be $V_t = \{y_1, y_2, ..., y_u, ?, ..., ?\}$ where $u = i(i-1)/2$ and ? are unknown values. Given the distance vector $V_{t^*} = (x_1, ..., x_{m'})$ of t^*, where $m' = m(m-1)/2$, DFS-Prune gives the following spatial similarity upper bound of any candidate tuple t with prefix $\{p_1, p_2, ..., p_i\}$.

$$\texttt{SIM}_s(t, t^*) \le \sqrt{\frac{A^2}{C} + \sum_{j=u+1}^{m'} {x'_j}^2} \tag{5}$$

where $x'_j = \frac{x_j}{\sqrt{\sum_{i=1}^{m'} x_i^2}}$, $A = \sum_{j=1}^{u} x'_j y_j$, $C = \sum_{j=1}^{u} y_j^2$.

III. Our Solution

The key to improving the efficiency of evaluating CSEQ is to reduce the number of candidate tuples to be examined. We first outline the following three steps to achieve this goal.

Space Partitioning. The first step is to reduce the search space by partitioning the whole search space into smaller subspaces so that many unpromising candidate tuples are immediately ruled out. To explain, observe that for the given example t^*, a valid candidate tuple should not contain two points whose distance exceeds $\beta||V_{t^*}||$, as this would break the CSEQ's β-norm restriction. In other words, if there is a tuple containing two points that have a distance greater than $\beta||V_{t^*}||$, then the tuple's norm must exceed $\beta||V_{t^*}||$ because it must be greater than the distance between any two points in the tuple. This violates the β-norm constraint, and as a result, the tuple cannot be an output of CSEQ. Based on this property, one straightforward idea is to divide the entire search space into multiple subspaces with diagonal lengths at most $\beta||V_{t^*}||$, and search each subspace respectively. This method, however, will overlook candidates whose points span multiple subspaces, resulting in repeated examinations of the same candidate tuple. We address this issue by encircling each subspace a band-shaped auxiliary subspace of width $\beta||V_{t^*}||$. As such, we ensure that each candidate resides within the union of one subspace and its auxiliary subspace. For efficiency, we further make sure that each candidate is searched *only once*. We will elaborate this technique in Sections III-A and III-B.

Two-Phase Pruning. The second step involves a two-phase pruning. On each subspace, we impose a grid with cell sizes of $d \times d$, with d as a parameter. The idea is to conduct pruning at the cell level. To avoid confusion, let us refer to a tuple as a *point tuple*; each point tuple t can be uniquely mapped to a *cell tuple*, which consists of the cells that respectively contain the points in t. As shown in Figure 3, a cell tuple that consists of cells c_1, c_2 and c_3 contains six point tuples. Then, the similarity between the example tuple and a cell tuple can be estimated (more details in Section III-C), and unpromising cell tuples are directly pruned without touching the point tuples contained in it. In other words, the search is realized in two phases: we first conduct pruning based on the cell tuples, followed by checking the validity of the point tuples inside the remaining cell tuples.

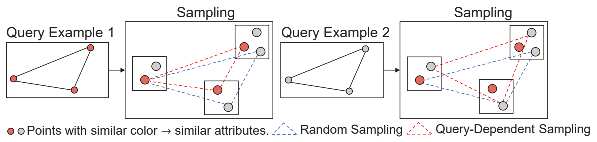


Fig. 4: Benefits of Query-Dependent Sampling.

Query-Dependent Sampling. In the third step, we apply a sampling technique where only ξ points are preserved in each cell during the search, with ξ as a parameter. One straightforward idea is to *randomly sample* ξ points from each cell. However, this simple approach may significantly affect the accuracy. An example for $\xi = 1$ is shown in Figure 4, where each cell only retains one node to answer a query. Consider the two examples with significantly different attribute-values, one indicated by red nodes and the other indicated by gray nodes. Our query-dependent sampling can successfully select the nodes (one from each cell, linked by dotted red lines) with the most similar attribute-values to the example in both queries. In contrast, the random sampling technique, which randomly selects one node from each cell, easily fails to retain desired tuples (see the selected nodes linked by dotted blue lines). We will discuss the detailed techniques in Section III-C.

Next, we will first present HSP, which is a basic and exact algorithm that employs the space partitioning technique. Then, we consolidate the above three techniques to design the LORA algorithm.

A. Hierarchical Space Partitioning Scheme

In this section, we develop a technique to partition the whole space into subspaces, such that we only need to enumerate points in the same subspace to generate valid tuples instead of doing so in the entire space.

The partitioning technique consists of two steps. First, it hierarchically divides the space into subspaces from the middle of horizontal or vertical dimensions alternatively, as shown in Figure 5; the partitioning stops when the subspace contains no

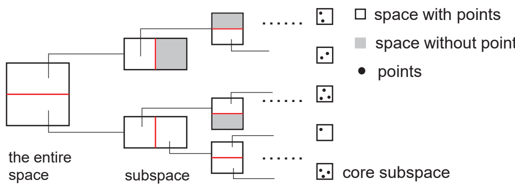


Fig. 5: Space partitioning.

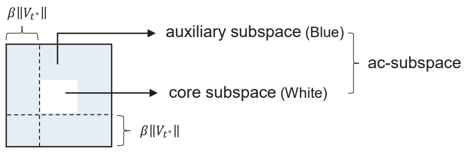


Fig. 6: A core subspace and its auxiliary subspace.

points, or each subspace has a diagonal length smaller than $\beta||V_{t^*}||$. Denote the subspaces with diagonal length smaller than $\beta||V_{t^*}||$ as *core subspaces*. It is obvious that all core subspaces are disjoint, and the union of them is the whole space. Also, each point belongs to exactly one core subspace. Note that the distance between any points within the same core subspace is smaller than $\beta||V_{t^*}||$. This allows us to filter out point tuples that are far apart from each other and clearly do not satisfy the distance constraints, thus reduces the number of candidates that need to be searched.

Second, we surround each core subspace S^c with a band-shaped *auxiliary subspace* S^a, such that for every point in a core subspace, all CSEQ candidate tuples containing it are within the union of the core subspace and its corresponding auxiliary subspace. Specifically, given a core subspace, its auxiliary subspace is a band-shaped space surrounding the core subspace with width $\beta||V_{t^*}||$, as shown in Figure 6, where a core subspace is in white and its auxiliary subspace is shaded. Note that the auxiliary subspaces of different core subspaces can overlap, and a point may be in multiple auxiliary subspaces, but only belongs to exactly one core subspace.

Given a core subspace S^c, denote the union of the core subspace and its auxiliary subspace S^a as its *ac-subspace* S^u. For each point in a core subspace, all CSEQ candidate tuples containing the point must be within the corresponding ac-subspace, which can be proved by contradiction. Specifically, every point in the core subspace has a distance larger than $\beta||V_{t^*}||$ to *any* point outside the corresponding ac-subspace, which contradicts the definition of CSEQ. Further, considering that all core subspaces are disjoint (i.e., a point is in exactly one core subspace) and their union is the entire space, it is sufficient to search within the ac-subspaces of all core subspaces to solve CSEQ without missing any valid tuples.

B. Exact Solution: HSP

A naive solution to CSEQ is to enumerate all possible candidate tuples in each ac-subspace S^u. Although this naive solution will not miss any results, the downside is that a candidate tuple may be enumerated multiple times, leading to inefficiency. For instance, given an example tuple t^* whose first point is in category a, it is possible that a point p_1 in category a falls in a core subspace and the auxiliary subspace of another core subspace (i.e., point p_1 is in multiple ac-subspaces). If we blindly search all ac-subspaces, a candidate tuple containing p_1 as the first element will be enumerated multiple times since p_1 is in many ac-subspaces.

In this section, we address this issue by leveraging the properties of core subspaces and enumerating each candidate tuple only once. Specifically, all core subspaces are disjoint, and a point is exactly in one core subspace. Therefore, for a candidate tuple $t = (p_1, p_2, ..., p_m)$, we only enumerate it when p_1 is in a core subspace, and ignore the cases when p_1 is in auxiliary subspaces.

Algorithm 1: Exact-DFS(i, S, t, G, R)

```
Input: Level id $i$, Space $S$, point tuple $t$, list of point lists
       $G$, top-$k$ result $R$;
if $i > m$ then
    if $\frac{1}{\beta} \leq \frac{||V_t||}{||V_{t^*}||} \leq \beta$ then
        Update $R$ with $t$;
else
    Let $a$ be the category of $i$-th point in $t^*$;
    for each point $p_j \in G^i$ with category $a$ do
        if $i > 1$ or $p_j$ is in the core-subspace of $S$ then
            Set the $i$-th element $t_i$ of tuple $t$ as $p_j$ ;
            Calculate the upper bound $\overline{\mathtt{SIM}}_a$ and $\overline{\mathtt{SIM}}_s$;
            Let $R_{min}$ be the minimum similarity in $R$;
            if $|R| < k$ or $\alpha\overline{\mathtt{SIM}}_s + (1-\alpha)\overline{\mathtt{SIM}}_a > R_{min}$
              then
                Exact-DFS($i+1, S, t, G, R$) ;
```

Lemma 1: Given all ac-subspaces, if a CSEQ candidate tuple $t = (p_1, p_2, ..., p_m)$ is enumerated only when processing the ac-subspace containing p_1 in its core subspace, then after processing all ac-subspaces, every candidate tuple will be enumerated exactly once.

Proof 1: Due to space limitations, we put the proof in our technical report [17].

We first present an exact and basic solution HSP for the CSEQ problem. HSP iterates all ac-subspaces; in each ac-subspace, HSP enumerates candidate tuples in a similar fashion to DFS-Prune, and prunes unpromising candidate tuples based on refined upper bounds for attribute similarity and spatial similarity. Compared with DFS-Prune [11], HSP conducts its search in much smaller ac-subspaces, leading to much higher efficiency. Further, it provides three refined techniques as follows.

1) Modified first-point selection. Within each ac-subspace, HSP enumerates candidate tuples only when their first points are located in the core subspace. Such enumeration is beneficial because it guarantees that each candidate tuple is enumerated at most once (see Lemma 1).

2) Refined upper bound for SIM$_a$. For a candidate tuple with prefix $\{p_1, ..., p_i\}$, we let the attribute similarities between each point p_j $(1 \leq j \leq i)$ and the respective point in example t^* be $\{r_1, ..., r_i\}$ respectively. Then, for any tuple that has not been enumerated, there is an upper bound of its attribute similarity with respect to t^*:

$$\overline{\mathtt{SIM}}_a = \sum_{j=1}^{i} r_j + \sum_{j=i+1}^{m} \overline{r}_j \tag{6}$$

where $\overline{r}_j$ denotes the maximum attribute similarity between a point in the ac-subspace and the j-th point in t^*. This upper bound holds because the attribute similarities corresponding to the first i dimensions are bounded by r_i and those for the remaining $m - i$ dimensions are bounded by $\overline{r}_j$. This bound refines the bound from DFS-Prune because $\overline{r}_j \leq 1$.

Algorithm 2: HSP

```
Input: Space $S$, point set $P$, the example $t^*$, $\beta$;
Output: Top-$k$ result $R$;
1 Get ac-subspaces $\mathbb{S} = \{S_1^u, S_2^u, ..., S_h^u\}$ (Sec. III-B);
2 Result set $R \leftarrow \emptyset$;
3 Lists $G \leftarrow \emptyset$;
4 for $S_i^u \in \mathbb{S}$ do
5     for each point $t^*[j]$ in the example $t^*$ do
6         Sort each point $p$ in $S_i^u$ with the same category as
          $t^*[j]$ in descending order of $\text{SIM}_a(p, t^*[j])$, and
          store them in $G^j$;
7         Add $G^j$ into $G$ ;
8     Set tuple $t$ as empty ;
9     Exact-DFS$(1, S_i^u, t, G, R)$;
```

3) Refined upper bound for SIM$_s$. HSP also employs a refined spatial similarity bound. Specifically, in deriving the inequality in Eq. 5, DFS-Prune defines $w = \frac{\sum_{j=u+1}^{m'} y_j^2}{\sum_{j=1}^{u} y_j^2}$, and gives an upper bound

$$\texttt{SIM}_s(t, t^*) \leq \frac{A}{\sqrt{C}} \cdot \frac{1}{\sqrt{1+w}} + \sqrt{\sum\nolimits_{j=u+1}^{m'} x'^2_j}\sqrt{\frac{w}{w+1}} \quad (7)$$

Applying the norm constraint in w, we have

$$\frac{||V_{t^*}||^2}{\beta^2 C} - 1 \leq w = \frac{\sum_{j=1}^{m'} y_j^2 - \sum_{j=1}^{u} y_j^2}{\sum_{j=1}^{u} y_j^2} \leq \frac{\beta^2 ||V_{t^*}||^2}{C} - 1 \quad (8)$$

Then, we derive that $\frac{1}{\sqrt{w+1}} \leq \frac{\beta\sqrt{C}}{||V_{t^*}||}$ and $\sqrt{\frac{w}{w+1}} = \sqrt{1 - \frac{1}{w+1}} \leq \sqrt{1 - \frac{C}{\beta^2 ||V_{t^*}||^2}}$. Applying these inequalities to Eq. 7, we obtain a new bound

$$\texttt{SIM}_s(s, t^*) \leq \frac{\beta A}{||V_{t^*}||} + \sqrt{\frac{\sum_{j=u+1}^{m'} x_j^2}{||V_{t^*}||^2}}\sqrt{1 - \frac{C}{\beta^2 ||V_{t^*}||^2}} \quad (9)$$

where $A = \sum_{j=1}^{u} x'_j y_j$ and $C = \sum_{j=1}^{u} y_j^2$. During pruning, we select the upper bound $\overline{\texttt{SIM}}_s$ as the tighter one between Eq. 5 and the new bound.

In a nutshell, HSP (Algorithm 2) enumerates candidate tuples in each ac-subspace and prunes unpromising candidate tuples by the above upper bounds. Specifically, HSP first sorts the points of each category in descending order of their attribute similarities to the given example t^* (Algorithm 2 Line 6). Then, HSP enumerates candidate tuples (Algorithm 2 Line 9), which can be realized by the Exact-DFS function (Algorithm 1). Invoking Exact-DFS(i, S, t, G, R) first selects the i-th point in the tuple (Algorithm 1 Line 8) and then selects the next point for the tuple during recursion (Algorithm 1 Line 12). The points are always selected in the ac-subspace, except for the first point, which is selected in the core subspace due to Lemma 1 (Algorithm 1 Line 7). Before selecting the next point for the tuple, if the upper bound derived based on the current prefix is less than the current k-th maximum similarity to t^*, we can skip all candidates sharing the same prefix (Algorithm 1 Lines 9-11). Once all the m points have been selected (Algorithm 1 Line 1), HSP first checks if the newly formed tuple t meets the norm constraint $\frac{1}{\beta} \leq \frac{||V_t||}{||V_{t^*}||} \leq \beta$ (Algorithm 1 Line 2), then compares it with the current maintained top-k results, and finally updates the results (Algorithm 1 Line 3).

Algorithm 3: LORA

```
Input: All ac-subspaces $\{S_1^u, S_2^u, ..., S_h^u\}$, example $t^*$;
Output: Approximate top-$k$ results $R$;
1 Set the priority queue $R$ as empty;
2 for each subspace $S$ in all ac-subspaces $\{S_1^u, S_2^u, ..., S_h^u\}$ do
3     Split $S$ into cells of size $d \times d$, denoted by
      $\hat{G} = \{g_1, g_2, ..., g_w\}$;
4     for each point $t^*[i]$ in $t^*$ do
5         $a \leftarrow$ category of $t^*[i]$;
6         Set point lists $G_1^a, ..., G_w^a$ as empty;
7         for each point $p_i \in S$ with category $a$ do
8             Let $g_j$ be the cell containing $p_i$;
9             $G_j^a \leftarrow G_j^a \cup \{p_i\}$;
10        for $j \in \{1, ..., w\}$ do
11            Point-Sample($G_j^a$, $t^*[i]$);
12    $G \leftarrow \{G_j^a | j, a\}$;
13    for each point $t^*[i]$ of each point in $t^*$ do
14        for each cell $g_z$ in $\hat{G}$ do
15            $\texttt{SIM}_a(t^*[i], g_z) \leftarrow \max_{p \in g_z}\{\texttt{SIM}_a(t^*[i], p)\}$;
16        Sort $g_z$ in descending order of $\texttt{SIM}_a(t^*[i], g_z)$;
17    Update $R$ in Cell-Tuple-Enum$(1, \emptyset, \hat{G}, G, R)$ ;
```

C. LORA: Local Approximation

The HSP algorithm needs to iterate all ac-subspaces, and enumerate all candidate tuples within each ac-subspace. Although an ac-subspace is much smaller compared with the whole data space, it is still expensive to process each ac-subspace to obtain all candidate tuples within it. In order to reduce the cost per ac-subspace, we propose an approximate algorithm, called LORA (LOcal Representative Approximation), which efficiently processes ac-subspaces and produces approximate results with rigorous theoretical guarantees.

One key observation inspiring LORA is that, in the real-world POI datasets, points in the same category are often clustered in a small region, e.g., many restaurants in a shopping mall. These restaurants are in the same category and have similar attributes. Intuitively, if one can first handle groups of same-category points in small regions to identify rough results, and then refine them to get the final top-k tuples, the time cost incurred per ac-subspace can be significantly reduced.

Specifically, as shown in Figure 7, given an ac-subspace S^u, we first grid it into small cells, with cell size $d \times d$. We consider that the points located in the same cell are in a group. In S^u, we can select one cell for each category in t^* to form a *cell tuple*, which is a tuple that contains m cells.

In the high-level idea, LORA conducts the enumeration of tuples in two phases. It first enumerates cell tuples (rough phase) for each ac-subspace, and then for each cell tuple, it generates the point tuples (refining phase) whose points are respectively located in each cell in the cell tuple.

The detailed procedure can be found in Algorithms 3, 4, 5. Algorithm 3 is the main function that calls Algorithm 4 that describes the cell-tuple enumeration. Algorithm 4 is a recursive function and invokes Algorithm 5 that describes the point-tuple enumeration. Next, we will describe the three

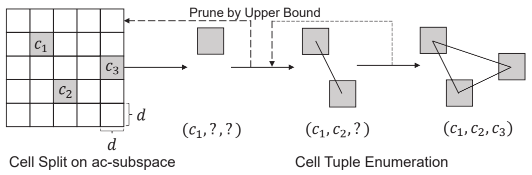


Fig. 7: The cell tuple enumeration process of LORA.

algorithmic procedures. To facilitate the discussion, we will first focus on illustrating the main flow of each procedure and defer the details to separate paragraphs if necessary.

In Algorithm 3, for each ac-subspace S, after imposing a grid on S (Line 3), we inspect the category a of each point $t^*[i]$ in t^*, and assign the points in S to the corresponding category lists G_i^a (Lines 4-9); ultimately, each list G_j^a contains all the points with category a in cell g_j. We then conduct a Point-Sample procedure based on each list G_j^a (Lines 10-11), followed by invoking the Cell-Tuple-Enum algorithm to extract the final top-k result R. As the Point-Sample procedure is a relatively independent module about sampling techniques applied in each cell, we defer the interesting discussion to Section III-C. For now, we focus on the main candidate generation function of Algorithm 3, which is the Cell-Tuple-Enum algorithm. To prepare the input for the Cell-Tuple-Enum algorithm, for each point v in the example, we sort the cells in the ac-subspace in a descending order of their maximum attribute similarity to point u (Lines 13-16). Here, the maximum attribute similarity between point u and a cell is the maximum attribute similarity between point u and each point in the cell. Hence, we have overall m sorted lists of cells stored in $\hat{G}$ as the input to the Cell-Tuple-Enum algorithm.

1) Cell-Tuple Enumeration: The cell-tuple enumeration is realized by a recursive function Cell-Tuple-Enum (Algorithm 4), which lists the cell-tuples in a depth-first-search (DFS) manner. This is similar to the Exact-DFS function employed in HSP, except that we "enumerate cell tuples based on m sorted cell lists" instead of "enumerate point tuples based on m sorted point lists". That is, we select one cell from each of the m cell lists to create cell tuples. The cells in each list are selected in descending order of their maximum attribute similarity to the respective point in t^*. We enforce effective cell tuple pruning during the search, to immediately prune those cell tuples that cannot generate top-k point tuple candidates (Lines 6-7). Specifically, we compute an upper bound $\bar{V}$ of the attribute similarity between a cell tuple and the example t^* (details will be explained shortly in the paragraph "Cell-Tuple-based Upper Bound".) Then, the overall similarity upper bound is computed by $\alpha \cdot 1 + (1-\alpha) \cdot \bar{V}$ because the spatial similarity is bounded by 1 and the attribute similarity is bounded by $\bar{V}$. If this upper bound does not exceed the current top-k similarities, we can safely skip the cell tuple for point tuple generation (Line 7). Otherwise, we invoke Cell-Tuple-Enum (explain shortly in Subsection "Point-Tuple Enumeration") to check the point tuples within the cell tuple (Line 8).

Cell-Tuple-based Upper Bound. The upper bound is based on the prefix and can be computed as follows. Suppose at some

Algorithm 4: Cell-Tuple-Enum(i, l, $\hat{G}$, G, R)

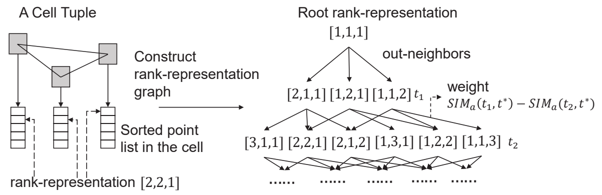
```
Input: Level id $i$, cell tuple $l$, cell list $\hat{G}$, point lists $G$,
       current top-$k$ result $R$;
1 if $i > m$ then
2     Update $R$ in Point-Tuple-Enum($G, R$);
3 else
4     for each cell $g_j$ in cell list $\hat{G}$ do
5         Set the $i$-th cell of $l$: $l[i] \leftarrow g_j$ ;
6         Calculate the upper bound $\overline{V}$ (see Section III-C
            Cell-Tuple based Upper Bound paragraph) ;
7         if $|R| < k$ or $\alpha + (1-\alpha) * \overline{V} > R_{min}$ then
8             Cell-Tuple-Enum($i+1, l, \hat{G}, G, R$) ;
```

point we have constructed a prefix of the cell tuple, denoted as $c_1c_2\ldots c_j$, where c_i is a cell. Then, we can compute the maximum attribute similarity s_i between a point in cell c_i and $t^*[i]$, where $t^*[i]$ is the i-th point in the example t^*. For any remaining dimension j, the respective attribute similarity is upper bounded by the maximum attribute similarity between each node in the ac-subspace and $t^*[j]$.

Example. An example procedure of the pruning-based cell tuple enumeration is shown in Figure 7. Suppose the cells c_1, c_2, c_3 have the maximum attribute similarities to $t^*[1]$, $t^*[2]$, $t^*[3]$, respectively. Then, LORA sequentially selects c_1, c_2 and c_3 to form a cell tuple, according to the DFS search. When prefixes "$(c_1,?,?)$" or "$(c_1,c_2,?)$" are formed, LORA checks whether the candidate cell tuple can be pruned based on the prefix-based upper bound. For "$(c_1,c_2,?)$", we compute the maximum attribute similarity between c_1 (resp. c_2) and $t^*[1]$ (resp. $t^*[2]$), and then obtain the maximum attribute similarity between any candidate point and $t^*[3]$. The sum of these attribute similarities can be regarded as an upper bound of the attribute similarity, which is then combined with a spatial similarity upper bound "1" to form the final similarity upper bound. If this upper bound does not exceed the k-th similarity currently found, then $(c_1,c_2,?)$ is pruned, and the search will go back to the state $(c_1,?,?)$.

2) Point-Tuple Enumeration: The procedure is shown in Algorithm 5, which is designed based on two observations: (1) two point tuples contained in the same cell tuple have similar spatial similarities to the example t^*, and thus Algorithm 5 focuses on getting point tuples with the highest attribute similarities to t^*; (2) for any final top-k result, there must be an ac-subspace containing it as the top-k result within the ac-subspace. Hence, it is sufficient to only enumerate the top-k results for each ac-subspace.

Particularly, for each cell tuple, we first construct m lists of



A Cell Tuple
Construct rank-representation graph
Sorted point list in the cell
rank-representation [2,2,1]
Root rank-representation
[1,1,1]
out-neighbors
[2,1,1] [1,2,1] [1,1,2] t_1
weight
$SIM_a(t_1,t^*) - SIM_a(t_2,t^*)$
[3,1,1] [2,2,1] [2,1,2] [1,3,1] [1,2,2] [1,1,3] t_2
....

Fig. 8: The rank-representation graph of LORA.

Algorithm 5: Point-Tuple-Enum(G, R)

```
Input: $G$ contains point lists for each category, $R$ contains current top-$k$ result;
1  Set maximum priority queue $Q \leftarrow \emptyset$;
2  Insert $\{r_0, 0\}$ into $Q$;
3  visited[$r_0$]$\leftarrow true$;
4  while $Q$ is not empty and $|R| < k$ do
5      Pop the front element of $Q$ as $[t, dis(r_0, t)]$;
6      if $\frac{1}{\beta} \leq \frac{||V_t||}{||V_{t^*}||} \leq \beta$ then
7          if $|R| < k$ or $dis(r_0, t) \leq \min_{v \in R}\{dis(r_0, v)\}$ then
8              Update $R$ with $t$;
9      if $|R| \geq k$ and $dis(r_0, t) > \min_{v \in R}\{dis(r_0, v)\}$ then
10         Return;
11     for each neighbor $v$ of $t$ in $G_{attr}$ do
12         $w(t, v) \leftarrow \mathtt{SIM_a(t, t^*)} - \mathtt{SIM_a(v, t^*)}$;
13         if $v$ was not visited or
             $dis(r_0, v) > dis(r_0, t) + w(t, v)$ then
14             $dis(r_0, v) \leftarrow dis(r_0, t) + w(t, v)$;
15             Insert $[v, dis(r_0, v)]$ into $Q$;
16             visited[$v$]$\leftarrow$ true;
```

points, where the i-th list contains the points in the ac-subspace that have the same category as $t^*[i]$. The points in the i-th list are sorted in descending order of their attribute similarities to $t^*[i]$ (due to observation (1)). Then, a point tuple can be represented by the corresponding ranks in these m point lists, which we refer to as *rank-representation*. For example, when $m = 3$, $[1, 1, 2]$ is a rank-representation of the tuple formed by the 1st-rank, 1st-rank, and 2nd-rank points respectively picked from the three point lists. We say point tuple t_2 is an out-neighbor of point tuple t_1, if t_2's rank-representation only differs from t_1's in one dimension and in that dimension $t_2's$ rank is one position lower than $t_1's$. For example, $[1, 1, 2]$ is an out-neighbor of $[1, 1, 1]$ because its ranking in the third dimension is lower by one position.

Given a cell tuple, we can construct a graph G_{attr} for an efficient enumeration. We let each point tuple generated from the cell tuple be a graph node; a node t_1 has a directed edge to another node t_2 if and only if t_2 is an out-neighbor of t_1; the weight of the edge (t_1, t_2) is $\mathtt{SIM}_a(t_1, t^*) - \mathtt{SIM}_a(t_2, t^*)$, where $\mathtt{SIM}_a$ denotes the attribute similarity. Obviously, the graph is "rooted at" node $[1, 1, 1]$, denoted by r_0. The graph G_{attr} is illustrated in Figure 8 and has the following interesting property.

Lemma 2: Among all the point tuples generated from a cell tuple, a point tuple has the i-th attribute similarity to t^*, if and only if it is the i-th closest node to r_0 in G_{attr}, where the closeness is measured by the shortest path distance $\mathtt{dis}(\cdot, \cdot)$. Particularly, for any node $t \in G_{attr}$, we have

$$\mathtt{SIM_a(t, t^*)} = \mathtt{SIM_a(r_0, t^*)} - \mathtt{dis(r_0, t)}$$

By Lemma 2, searching for the point tuples with top-k attribute similarities to t^* can be conducted in a way similar to finding top-k shortest paths from the root node r_0 in G_{attr} (Algorithm 5). We initialize a priority queue Q with an element $[r_0, 0]$ (Lines 1-2), which means that the root node r_0 with distance 0 (from r_0 itself) is put in Q, and the priority

Algorithm 6: Point-Sample(G^a_j, $t^*[i]$)

```
1  Sort each point $p_i$ in each point list $G^a_j$ in descending order
     of its attribute similarity to $t^*[i]$, the latter being the point
     in $t^*$ of category $a$;
2  Keep the first $\xi$ points in $G^a_j$ as the sampled points;
```

is based on the distances from r_0. Then, we perform multiple iterations (Line 4), each of which extracts the front element $[t, d_t]$ from Q (Line 5). This means that t is the node in Q with the shortest distance from r_0 (or equivalently, the maximum attribute similarity to t^*). If t satisfies the distance constraint defined in CSEQ (Line 6), then we check the possibility of t being top-k attribute-similar results (Lines 7 - 8). Particularly, we can directly compare the shortest distance $dis(r_0, t)$ and the shortest distances between r_0 and nodes in R for this purpose due to Lemma 2 (Line 7). If there are k results and no more promising results can be further obtained, the program terminates (Lines 9-10). Afterwards, we check the out-neighbor v of t and examine whether $d_v < d_t + w(t, v)$ (Line 13). Here, $\mathtt{SIM}_a(t, t^*) - \mathtt{SIM}_a(v, t^*)$ is the weight of the edge (v, t) in G_{attr} (Line 12).

D. Query-Dependent Sampling

Sampling is a powerful technique to improve the computation efficiency. In LORA, we sample the points in each cell to speed up the computation. Unfortunately, applying the typical random sampling here is not satisfactory as it may significantly impact the accuracy (recall the example in Figure 4). We suggest a query-dependent sampling to address this issue, and show that this approach gives high efficiency and accuracy.

Specifically, the query-dependent sampling is achieved by the Point-Sample procedure (Algorithm 6); that is, we only consider a point tuple as a candidate, if it consists of m points in the sample sets. In LORA's sampling, we employ a *query-dependent* fashion of sampling; that is, for the query example tuple t^* and a cell tuple consists of cells c_i $(1 \leq i \leq m)$, each cell c_i samples its points by retaining top-ξ points with the highest attribute similarity to $t^*[i](1 \leq i \leq m)$, the i-th point in t^*. The intuition is that, such sampled points have higher attribute similarities to the corresponding points in t^*, and hence, they are highly possible to constitute point tuples with higher similarities to t^*.

Accuracy Analysis. From the experiments, we observe that even when ξ is relatively small (e.g., $\xi = 10$), LORA still gives high accuracy, demonstrating the effectiveness of our sampling techniques. When ξ is sufficiently large, the following result shows the approximation ratio of LORA.

Theorem 3: Given the top-k optimal result tuples $t_1, t_2, ..., t_k$ for CSEQ with respect to the example t^*, and the approximate top-k result tuples $\hat{t}_1, \hat{t}_2, ..., \hat{t}_k$ output by LORA, $\forall i \in \{1, 2, .., k\}$, we have

$$\mathtt{SIM}(t_i, t^*) \leq (1 + \frac{2\beta d\sqrt{m^2 - m}}{\|V_{t^*}\|}) \cdot \mathtt{SIM}(\hat{t}_i, t^*) + \alpha \cdot \frac{2\beta d\sqrt{m^2 - m}}{\|V_{t^*}\|}$$

By Theorem 3 and let $d \leq \epsilon \frac{\|V_{t^*}\|}{2\beta\sqrt{m^2 - m}}$ for an approximation parameter ϵ, we have

$$\mathtt{SIM}(t_i, t^*) \leq (1 + \epsilon) \cdot \mathtt{SIM}(\hat{t}_i, t^*) + \alpha\epsilon$$

On the other hand, as t_i's are optimal results, we have

$$\mathtt{SIM}(\hat{t}_i, t^*) \leq \mathtt{SIM}(t_i, t^*)$$

This gives us $\mathtt{SIM}(\hat{t}_i, t^*) \leq \mathtt{SIM}(t_i, t^*) \leq (1+\epsilon)\cdot\mathtt{SIM}(\hat{t}_i, t^*) + \alpha\epsilon$. Hence, given any example t^*, the tuning parameter d allows LORA's accuracy to be arbitrarily close to the exact solution.

IV. Experiments

We evaluate the efficiency and accuracy of LORA, HSP, and DFS-Prune, and compare them in various settings.

Datasets. We use the Yelp POI dataset [18], which has 77,444 POIs and 1395 categories, and the Gaode dataset [19], which contains 10,000,000 POIs and 20 categories. Each POI is associated with its geographical location and attributes. The attributes include "rating", "number of reviews", and "sub-categories". We also synthesize various sized datasets for the scalability testing, as shown in Table II.

Queries. For each test, we create 100 queries, each is an example consisting of objects with different categories. As the Yelp dataset covers a smaller spatial area, the objects that make up the query examples are randomly chosen. In contrast, the Gaode dataset covers a larger spatial area. Hence, we bound the distances between the objects in each example to make the example more meaningful.

Parameters. We vary the tuple size in [2, 6] (3 by default), the k value in [1, 9] (5 by default), α in [0.1, 0.9] (0.5 by default), and β in [1, 9] (1.5 by default). For LORA's experiments, we introduce the cell split parameter D, indicating that each ac-subspace is divided into $D \times D$ cells. The parameter D uniquely determines a parameter d. The query cost is averaged among all the queries; we measure the result quality of the approximate algorithm LORA by the mean absolute error (MAE) of LORA's results, which is computed by averaging the absolute errors of the similarities between the exact top-k results and LORA's top-k results.

Algorithms. We compare HSP and LORA with the state-of-the-art search method DFS-Prune [11]. In our experiments, we also evaluate LORA and HSP on SEQ and CSEQ-FP.

A. Performance for CSEQ

1) Overall Comparison: We compare the performance of LORA, HSP, and DFS-Prune on different data for the CSEQ problem. For this purpose, we sample different numbers of POIs from two real datasets, namely Yelp and Gaode. Table II shows the per-query costs for different methods. For a fair comparison, the running time for LORA has included the costs for space partitioning and sorting within cells.

We observe that the state-of-the-art method, DFS-Prune, takes more than 300 seconds to answer a CSEQ query for a small dataset with 50,000 POIs, implying an unsatisfactory performance in real-world applications. In contrast, in the same dataset, our exact algorithm HSP performs 15 times faster compared with DFS-Prune, reducing the query cost to about 21.61 seconds. Furthermore, LORA has the best efficiency, taking only 1.14 seconds to process the query. For larger datasets, the efficiency advantage can be more significant, and LORA is up to 5000 times faster than DFS-Prune with just a minor accuracy loss (LORA MAE). We also notice that LORA's running time scales sub-linearly with dataset size, demonstrating substantially better scalability than DFS-Prune, whose cost scales super-linearly.

The high efficiency of HSP comes from its hierarchical partitioning scheme, which prunes a large number of tuples that do not satisfy the norm constraint. LORA has stronger scalability because it also effectively conducts pruning based on query-dependent sampling techniques, being less sensitive to the growth of datasets. We conclude that LORA strikes an outstanding balance between efficiency and accuracy, and the accuracy loss experienced by the LORA is negligible. To further investigate the worst-case errors of LORA, we report the Maximum Error (MAX), and the Standard Deviation (STD) of errors for LORA, in Table III. These statistics, being very close to 0, demonstrate that LORA's result quality is very close to that of the exact result even in the worst case. Furthermore, LORA can be deployed for a dataset containing 10^7 POIs, at which data size all exact algorithms cannot finish a query with reasonable running times. To conclude, LORA greatly enhances the scalability of the example-based search.

2) Parameters Analysis: We conduct parameter analysis on all the relevant parameters in CSEQ, to demonstrate that LORA achieves high efficiency and accuracy across a wide spectrum of parameter settings compared with DFS-Prune. We consider four sampled datasets. For both Gaode and Yelp, we sample two datasets containing $10,000$ POIs and $50,000$ POIs. We show the effectiveness of each method using the *average similarity*. For the top-k results obtained by each method, the average similarity is the average of the k similarities between the top-k results and the given example t^*. Due to space limitations, we omit the discussion on parameters k and m (the obvious ones) and put it in the technical report [17].

Grid Resolution D. As shown in Figures 9(a.1) and (a.3), in terms of efficiency, LORA is the best, followed by HSP and DFS-Prune, regardless of the changes of D. Particularly, In Gaode dataset with $50,000$ POIs (see Figure 9(a.1)), DFS-Prune's efficiency is about 300 seconds per query, which is impractical. While HSP significantly improves the running time down to 20 seconds. LORA dramatically improves the efficiency for typical D values. LORA's running time increases with D, because when each cell becomes smaller (larger D), fewer points fall into a single cell. Hence, fewer points can be filtered out by LORA's sampling strategy, resulting in a longer running time. For accuracy, as shown in Figures 9(a.2) and (a.4), with the increase of D, the approximate result obtained by LORA becomes closer to the exact results (i.e., the results obtained by HSP), as indicated by their similarities. Since the number of cells increases with D, more points cannot be filtered out by LORA and will be participated in the point-enumeration process. This gradually approaches the exact similarities obtained by enumerating all points.

Similarity Weight α. As shown in Figures 9(c.1)-(c.4), the relative performance superiority of LORA is only slightly

TABLE II: The efficiency of HSP and LORA on real datasets.

Yelp Dataset						Gaode Dataset					
#POIs	Time Cost (s)			LORA MAE	LORA Speedup	#POIs	Time Cost (s)			LORA MAE	LORA Speedup
	DFS-Prune	HSP	LORA				DFS-Prune	HSP	LORA		
1,000	0.51	0.04	0.06	0.0001	8.5	10,000	8.87	1.11	0.38	0.002	20+
5,000	7.92	0.21	0.51	0.00006	15.53	50,000	348.30	21.61	1.14	0.002	300+
10,000	31.44	1.51	1.46	0.00003	21.53	100,000	3734.34	55.92	1.21	0.001	3,000+
30,000	142.81	7.59	5.18	0.00002	27.57	500,000	>24hours	592.14	16.05	0.002	>5,000
50,000	436.12	20.87	8.02	0.00001	54.38	1,000,000	>24hours	3748.71	16.17	0.003	>5,000
77,445	1173.69	60.02	11.56	0.00001	101.53	10,000,000	>24hours	>24hours	126.89	<0.012	>1,000

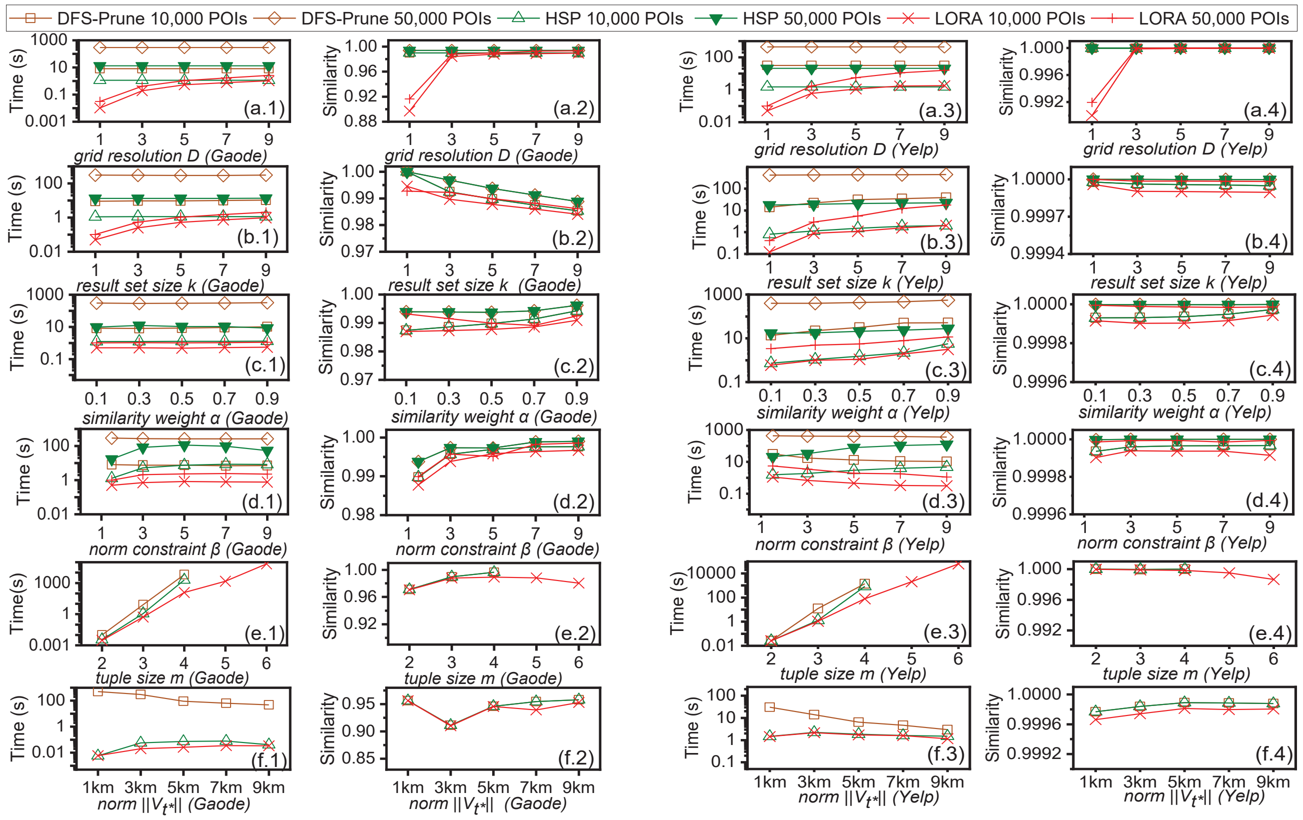


Fig. 9: The performance for different parameters of DFS-Prune, HSP and LORA on Gaode and Yelp datasets.

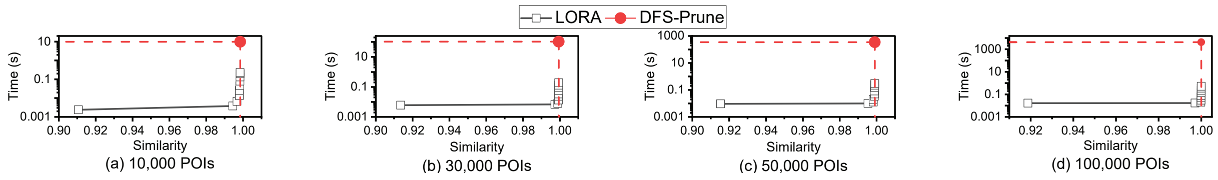


Fig. 10: The running time and similarity of LORA and DFS-Prune in SEQ problem.

TABLE III: More details of LORA's accuracy.

#POIs	Yelp		#POIs	Gaode	
	STD	MAX		STD	MAX
5,000	0.0001	0.0009	10,000	0.007	0.092
10,000	5.1e-05	0.0005	50,000	0.008	0.082
30,000	2.7e-05	0.0002	100,000	0.004	0.032
50,000	3.4e-05	0.0004	500,000	0.008	0.061
77,445	1.8e-05	0.0001	1,000,000	0.010	0.081

affected by α. The reason is that as long as both spatial similarity and attribute similarity need to be simultaneously considered, their relevant computation costs are comparable. Hence, tuning α does not significantly affect the performance. Furthermore, for all the tests, the average similarities to the given example are higher than 98%. Particularly, for Yelp, all the similarities of LORA are close to 1, showing the high effectiveness of LORA. This is because the POI in the Yelp dataset has been associated with abundant text information. Hence, it is prone to produce a higher attribute similarity value between a candidate tuple and the given example.

Norm Constraint β. As shown in Figures 9(d.1)-(d.4), LORA's running time is only slightly affected by β, as LORA's complexity is more relevant to the constant ξ. However, we observe that a larger β leads to higher similarity, owing to the

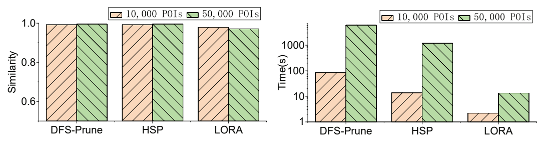


Fig. 11: The performance for CSEQ-FP.

fact that more candidates are admitted with a larger β. Again, we see all methods output results that have similarities higher than 98%, with all different β's and datasets.

Example Scale $||V_{t^*}||$. As shown in Figures 9(f.1)-(f.4), for a smaller $||V_{t^*}||$ value (e.g., 1km), HSP and LORA outperform DFS-Prune by more than one order of magnitude because a smaller $||V_{t^*}||$ makes a more pronounced space pruning in hierarchical partitioning. As a result, HSP's performance becomes closer to LORA's with the increase of $||V_{t^*}||$. Meanwhile, we observe that LORA's similarity curve almost overlaps with that of the exact solution.

B. Performance for SEQ and CSEQ-FP

Figure 10 reports LORA and DFS-Prune's performance on SEQ (by setting β to $+\infty$ in CSEQ). The experiments are conducted on four Gaode sampled POI sets, which consist of 10,000 to 100,000 POIs; the efficiency (per-query time) and similarity (averaged on top-k results similarities to the example t^*) are reported. LORA's result accuracy depends on the grid resolution D. We therefore test the query cost and similarities with D ranges in [1, 10], for DFS-Prune and LORA. Within each sub-figure of Figure 10, there are 10 data points showing the performance of LORA corresponding to $D \in [1, 10]$. When D becomes larger, the grid becomes finer and therefore LORA achieves a similarity closer to that of DFS-Prune (which is the best similarity that can be achieved as DFS-Prune is an exact algorithm). In the meanwhile, the query cost of LORA goes up with the increase of the similarity. One crucial observation is that, even when LORA achieves a similarity that is extremely close to DFS-Prune, LORA still runs around 3 orders of magnitude faster than DFS-Prune. This demonstrates that LORA strikes an excellent balance between efficiency and accuracy for answering SEQ.

We can also extend our algorithms to handle the CSEQ-FP problem that returns tuples with some fixed points. For evaluating the performance of CSEQ-FP, each query corresponds to a size-5 example, with two fixed points that must also appear in the result tuples. Figure 11 reports the results for two sub-datasets of Gaode, showing that LORA also performs drastically better than baselines for CSEQ-FP, as well as retaining very high accuracy. Particularly, it reduces the time cost by two orders of magnitude compared with DFS-prune, while the similarities are higher than 0.98. It is also worth noting that the exact algorithm HSP improves the efficiency by an order of magnitude compared to DFS-prune. In a nutshell, this experiment demonstrates that our proposed algorithms suit the CSEQ-FP problem as well.

C. User Studies

Our user study focuses on two tasks: (1) Do users prefer the example-based search over the filtering-based search under some real-world scenarios? (2) To what extent will users be happy to combine example-based and filtering-based search?

Methodology. We recruited 8 male and 5 female graduate students online. All user studies were conducted online. After consenting to the study, participants received a 5-minute handwritten tutorial introducing basic concepts of example-based search and filtering spatial query search (an adapted one from Google Maps). Then participants are required to answer all questions within 25 minutes.

Survey Questions. There are 4 questions in general where we tried to simulate different real-world applications, for example, the scenarios described in Example 1 and 2. Each question contains five sub-questions where participants evaluated the effectiveness of different search interfaces, explain the reason, and compare whether they would like to have example-based search and filtering-based search hand-in-hand, and briefly give reasons of their choice. The central idea of the survey is to find when users would prefer the example-based search in some general real-world situations, and explore whether having both example-based search and filtering will be an option that participants would consider in the future.

Results. As for the question that we asked our participants to evaluate which interface can help them better in finding POIs, roughly 61.63% of the participants found that the example-based search works better in real-world scenarios compared to the 38.38% of those who chose the filtering-based search, which implies that in the scenarios we created, the example-based search can help participants more effectively find their ideal locations. We also asked our participants to consider whether they would like to have an interface serving both the example-based search and the filtering-based search. Among participants who chose filtering-based search over example-based search, roughly 83.6% of them would like to see such an interface supporting both. This implies that example-based search has the potential to work hand-in-hand with the existing spatial object search engines to satisfy a wider range of users and even more general search requirements.

Qualitative Response. *1) Examples are helpful when there are multiple constraints.* Given the multiple search conditions listed in the questionnaire, 11 participants mentioned that the example-based search interface made their search easier. P2 directly pointed out that "Because I have multiple constraints across many objects.", and P7 also signaled that "It is more convenient to compare the different candidates among the map with everything I care about visible", which implies that the example-based search may offer a better human-device interaction. In addition, adding constraints from scratch might be a burden to users, as indicated by P9 "The filtering takes more time for me.". In general, our participants felt the burden lifted and more time-efficient with the example-based search.
2) Filtering invites iterations. We also found that the example-based search has some weaknesses. Particularly, among participants who preferred filtering, they reported that through filtering they might be able to find more specific information. For example, P3 mentioned that "The first priority is to cut

the budget", and P11 mentioned that "I might also have preferences over breakfast and daycare". These implied that participants expected to know more details through filtering.

This provides a window for the example-based search to work complementarily with filtering: by adding the filtering function on top of the example-based search, users can enjoy the convenience and more effective interface confronting various constraints, and meanwhile, leverage the filtering to find their ultimately personalized objects of interest.

V. Other Related Works

Learning Queries by Example. SEQ is inspired by the query-by-example paradigm, or QBE, which was originally proposed for relational databases [20]–[30] and graphs [31]–[33]. The general purpose of QBE is to assist non-professional users to understand the database better. QBE first allows users to input multiple desired example output tuples. Then, it learns the possible query intent from the user and identifies the queries whose results cover all those user-given examples. In general, QBE for relational databases is very different from SEQ or CSEQ in map services, as the former depends on reasoning the foreign-key relationships or tuple properties of the given results to reconstruct the SQL queries, while the latter focuses on finding similar spatially located objects to the given example object set. Furthermore, in SEQ or CSEQ, it is not feasible to request users to input a large number of examples in the map service; hence, the solutions for QBE cannot be applied in answering SEQ or CSEQ.

Exploration by Example. There is recent interest in the topic of exploring massive data through examples [34]–[41]. The paradigm of explore-by-example focuses on discovering the query intent by multiple rounds of user interaction and exploration. For example, [34] provides a system which allows the user to interactively label tuples as "interesting" or "not interesting". Through the process, it can construct a model to describe the user interest. Recent techniques also model this process as an active-learning [38] or reinforcement-learning [37] process to facilitate query-intent discovery process. Clearly, the focus of these works is different from SEQ or CSEQ, as they focus on a multiple-round user interaction process to capture the user interest queries.

VI. Conclusion

We present an in-depth study on the example-based query in spatial services, and provide two algorithms HSP and LORA. HSP is an exact algorithm that improves the state-of-the-art method by up to 20 times, and LORA is a highly-accurate approximate algorithm that improves upon the state of the art by up to four orders of magnitude. We conducted a user study to demonstrate the effectiveness of example-based search.

VII. Proof of Theorem 3

Our proof contains two parts. First, we prove that for each tuple t_i of the exact top-k results, there exists a corresponding tuple t'_i generated from the same cell tuple as t_i, satisfying $\texttt{SIM}(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}(t'_i, t^*)+\alpha\cdot\gamma$, where $\gamma = \frac{2\beta d\sqrt{m^2-m}}{\|V_{t^*}\|}$. Then, we prove that after replacing t'_i with the i-th result tuple $\hat{t}_i$ found by LORA, the inequality $\texttt{SIM}(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}(\hat{t}_i, t^*) + \alpha\cdot\gamma$ holds.

To prove the first part, it is sufficient to show (i) $\texttt{SIM}_a(t_i, t^*) \leq \texttt{SIM}_a(t'_i, t^*)$ and (ii) $\texttt{SIM}_s(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}_s(t'_i, t^*)+\gamma$. Then applying $\texttt{SIM}(t_i, t^*) = \alpha\cdot\texttt{SIM}_s(t_i, t^*) + (1-\alpha)\cdot\texttt{SIM}_a(t_i, t^*)$ directly gives us $\texttt{SIM}(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}(t'_i, t^*) + \alpha\cdot\gamma$ (denoted by Inequality (iii)).

Proof of Inequality (i). Suppose that tuple $t_i = \{p_1, p_2, ..., p_m\}$ is the i-th result of the optimal top-k results. Each point p_j of t_i falls into a cell g_j, and these cells form a cell tuple $\mathbb{C} = (g_1, g_2, ..., g_m)$. Let $R = \{t_1, t_2, ..., t_k\}$ and S be the set of point tuples generated in the cell tuple $\mathbb{C}$. Then among $R \cap S$, tuple t_i has the j-th largest attribute similarity to the respective point in t^*. By Lines 11-16 in Alg. 5, LORA can get a corresponding point tuple t'_i, which has the j-th attribute similarity (to the respective point in t^*) among S. This further implies $\texttt{SIM}_a(t_i, t^*) \leq \texttt{SIM}_a(t'_i, t^*)$.

Proof of Inequality (ii). Let the distance vectors of t_i, t'_i and t^* be $V_{t_i} = (y_1, y_2, ..., y_{m'})$, $V_{t'_i} = (y'_1, y'_2, ..., y'_{m'})$ and $V_{t^*} = (x_1, x_2, ..., x_{m'})$, where $m' = \frac{m(m-1)}{2}$. We have

$$\begin{aligned}\texttt{SIM}_s(t_i, t^*) =& \frac{\sum_{i=1}^{m'} y_i x_i}{\|V_{t_i}\|\,\|V_{t^*}\|} \leq \frac{\sum_{i=1}^{m'}(y'_i + 2\sqrt{2}d)x_i}{\|V_{t_i}\|\,\|V_{t^*}\|} \\ &\leq \frac{1}{\|V_{t^*}\|}\frac{\sum_{i=1}^{m'} y'_i x_i + 2\sqrt{2}d\sum_{i=1}^{m'} x_i}{\left\|V_{t'_i}\right\|}\frac{\left\|V_{t'_i}\right\|}{\|V_{t_i}\|} \\ &= \texttt{SIM}_s(t', t^*)\frac{\left\|V_{t'_i}\right\|}{\|V_{t_i}\|} + \frac{2\sqrt{2}d\sum_{i=1}^{m'} x_i}{\|V_{t^*}\|\,\|V_{t_i}\|}\end{aligned}$$

Note that $\frac{2\sqrt{2}d\sum_{i=1}^{m'} x_i}{\left\|V_{t_i}\right\|} \leq 2\sqrt{2}d\frac{\sqrt{m'}\sqrt{\sum_{i=1}^{m'} x_i^2}}{\sqrt{\sum_{i=1}^{m'} y_i^2}} \leq 2\sqrt{2m'}\beta d$, further by $\frac{\left\|V_{t'_i}\right\|}{\left\|V_{t_i}\right\|} \leq (1 + \frac{2\sqrt{2m'}\beta d}{\|V_{t^*}\|})$ (A detailed proof is in our technical report [17]) and $m' = \frac{m(m-1)}{2}$, we have

$$\texttt{SIM}_s(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}_s(t'_i, t^*) + \gamma$$

Next, we prove the second part. For the k pairs (t_i, t'_i) satisfying Inequality (iii), we can assume $\texttt{SIM}(t'_j, t^*) \geq \texttt{SIM}(t'_i, t^*)$ for $j < i$ without loss of generality. To explain, if for some $j < i$, $\texttt{SIM}(t'_j, t^*) < \texttt{SIM}(t'_i, t^*)$ holds, then because $\texttt{SIM}(t_i, t^*) \leq \texttt{SIM}(t_j, t^*) \leq (1+\gamma)\cdot\texttt{SIM}(t'_j, t^*) + \alpha\cdot\gamma < (1+\gamma)\cdot\texttt{SIM}(t'_i, t^*) + \alpha\cdot\gamma$, the Inequality (iii) for t_i and t'_i still holds. Furthermore, since LORA have enumerated t'_1, t'_2, ..., t'_k together with other candidates, and finally obtain $\hat{t}_1$, ..., $\hat{t}_k$ as the top-k result, we have $\texttt{SIM}(t'_i, t^*) \leq \texttt{SIM}(\hat{t}_i, t^*)$. Hence $\texttt{SIM}(t_i, t^*) \leq (1+\gamma)\cdot\texttt{SIM}(\hat{t}_i, t^*) + \alpha\cdot\gamma$. □

VIII. Acknowledgements

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG18/21), NTU startup grant (#020948-00001) and in part by Singapore MOE AcRF Tier 1 Seed Funding (RS05/21). This research is also supported in part by the National Natural Science Foundation of China under grant 61772138 and 62172107.

References

[1] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *International symposium on spatial and temporal databases*. Springer, 2005, pp. 273–290.

[2] T. Hashem, T. Hashem, M. E. Ali, and L. Kulik, "Group trip planning queries in spatial databases," in *International Symposium on Spatial and Temporal Databases*. Springer, 2013, pp. 259–276.

[3] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, "Shortest path and distance queries on road networks: towards bridging theory and practice," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2013, pp. 857–868.

[4] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2537–2551, 2017.

[5] Z. Yao, Y. Fu, B. Liu, Y. Liu, and H. Xiong, "Poi recommendation: A temporal matching between poi popularity and user regularity," in *Proceedings of IEEE 16th ICDM*. IEEE, 2016, pp. 549–558.

[6] W. Luo and A. M. MacEachren, "Geo-social visual analytics," *Journal of spatial information science*, vol. 2014, no. 8, pp. 27–66, 2014.

[7] W. Luo, P. Yin, Q. Di, F. Hardisty, and A. M. MacEachren, "A geovisual analytic approach to understanding geo-social relationships in the international trade network," *PloS one*, vol. 9, no. 2, p. e88666, 2014.

[8] Y. Fang, R. Cheng, G. Cong, N. Mamoulis, and Y. Li, "On spatial pattern matching," in *Proceedings of IEEE 34th ICDE*. IEEE, 2018, pp. 293–304.

[9] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: an experimental evaluation," *Proceedings of the VLDB Endowment*, vol. 6, no. 3, pp. 217–228, 2013.

[10] C. Long, R. C.-W. Wong, K. Wang, and A. W.-C. Fu, "Collective spatial keyword queries: a distance owner-driven approach," in *Proceedings of the ACM SIGMOD*, 2013, pp. 689–700.

[11] S. Luo, J. Hu, R. Cheng, J. Yan, and B. Kao, "Seq: Example-based query for spatial objects," in *Proceedings of the ACM CIKM*, 2017, pp. 2179–2182.

[12] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proceedings of the VLDB Endowment*. Elsevier, 2003, pp. 802–813.

[13] M. A. Cheema, W. Zhang, X. Lin, Y. Zhang, and X. Li, "Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks," *The VLDB Journal*, vol. 21, no. 1, pp. 69–95, 2012.

[14] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin, "SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index," *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 1–12, 2014.

[15] T. Bozkaya and M. Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces," in *Proceedings of the ACM SIGMOD*, 1997, pp. 357–368.

[16] S. Luo, B. Kao, G. Li, J. Hu, R. Cheng, and Y. Zheng, "Toain: a throughput optimizing adaptive index for answering dynamic k nn queries on road networks," *Proceedings of the VLDB Endowment*, vol. 11, no. 5, pp. 594–606, 2018.

[17] Supplementary material for example-based spatial search at scale. [Online]. Available: https://www.dropbox.com/s/3wlphqk89ro5ea0/ICDE_2022_P58_Supplementary_Material.pdf?dl=0

[18] Yelp dataset. [Online]. Available: https://www.yelp.com/dataset

[19] S. I. Center, "Map POI (Point of Interest) data," 2017. [Online]. Available: https://doi.org/10.18170/DVN/WSXCNM

[20] H. Li, C. Chan, and D. Maier, "Query from examples: An iterative, data-driven approach to query construction," *Proceedings of the VLDB Endowment*, vol. 8, no. 13, pp. 2158–2169, 2015.

[21] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri, "S4: top-k spreadsheet-style search for query discovery," in *Proceedings of the ACM SIGMOD*, 2015, pp. 2001–2016.

[22] W. C. Tan, M. Zhang, H. Elmeleegy, and D. Srivastava, "Reverse engineering aggregation queries," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1394–1405, 2017.

[23] D. V. Kalashnikov, L. V. Lakshmanan, and D. Srivastava, "Fastqre: Fast query reverse engineering," in *Proceedings of the ACM SIGMOD*, 2018, pp. 337–350.

[24] A. Bonifati, R. Ciucanu, and S. Staworko, "Learning join queries from user examples," *ACM Transactions on Database Systems (TODS)*, vol. 40, no. 4, pp. 1–38, 2016.

[25] W. C. Tan, M. Zhang, H. Elmeleegy, and D. Srivastava, "Regal+ reverse engineering spja queries," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1982–1985, 2018.

[26] C. Wang, A. Cheung, and R. Bodik, "Interactive query synthesis from input-output examples," in *Proceedings of the ACM SIGMOD*, 2017, pp. 1631–1634.

[27] D. M. L. Martins, "Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities," *Information Systems*, vol. 83, pp. 89–100, 2019.

[28] P. Barceló and M. Romero, "The complexity of reverse engineering problems for conjunctive queries," *arXiv preprint arXiv:1606.01206*, 2016.

[29] A. Fariha and A. Meliou, "Example-driven query intent discovery: Abductive reasoning using semantic similarity," *arXiv preprint arXiv:1906.10322*, 2019.

[30] P. Orvalho, M. Terra-Neves, M. Ventura, R. Martins, and V. Manquinho, "Squares: a sql synthesizer using query reverse engineering," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2853–2856, 2020.

[31] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri, "Querying knowledge graphs by example entity tuples," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 10, pp. 2797–2811, 2015.

[32] J. Han, K. Zheng, A. Sun, S. Shang, and J.-R. Wen, "Discovering neighborhood pattern queries by sample answers in knowledge base," in *Proceedings of IEEE 32nd ICDE*. IEEE, 2016, pp. 1014–1025.

[33] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri, "Towards a query-by-example system for knowledge graphs," in *Proceedings of workshop on graph data management experiences and systems*, 2014, pp. 1–6.

[34] E. Huang, L. Peng, L. Di Palma, A. Abdelkafi, A. Liu, and Y. Diao, "Optimization for active learning-based interactive database exploration," *Proceedings of the VLDB Endowment*, vol. 12, no. 1, pp. 71–84, 2018.

[35] K. Dimitriadou, O. Papaemmanouil, and Y. Diao, "Explore-by-example: An automatic query steering framework for interactive data exploration," in *Proceedings of the ACM SIGMOD*, 2014, pp. 517–528.

[36] K.Dimitriadou, O.Papaemmanouil and Y.Diao, "Aide: an active learning-based approach for interactive data exploration," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 2842–2856, 2016.

[37] O. Bar El, T. Milo, and A. Somech, "Automatically generating data exploration sessions using deep reinforcement learning," in *Proceedings of the ACM SIGMOD*, 2020, pp. 1527–1537.

[38] E. Huang, L. Palma, L. Cetinsoy, Y. Diao, and A. Liu, "Aideme: An active learning based system for interactive exploration of large datasets," in *Proceedings of the 33rd NIPS*, 2019.

[39] X. Ge, X. Zhang, and P. K. Chrysanthis, "Exnav: An interactive big data exploration framework for big unstructured data," in *Proceedings of IEEE International Conference on Big Data*. IEEE, 2020, pp. 503–512.

[40] T. Milo and A. Somech, "Automating exploratory data analysis via machine learning: An overview," in *Proceedings of the ACM SIGMOD*, 2020, pp. 2617–2622.

[41] G. Vargas-Solar, M. Farokhnejad, and J. Espinosa-Oviedo, "Towards human-in-the-loop based query rewriting for exploring datasets," in *Proceedings of the Workshops of the EDBT/ICDT Joint Conference*, 2021.