Knowledge-Based Systems 263 (2023) 110299

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Efficient m-closest entity matching over heterogeneous information networks

Wancheng Long [a,1], Xiaowen Li [b,1], Liping Wang [a,*], Fan Zhang [c], Zhe Lin [a], Xuemin Lin [d]

[a] College of Software Engineering, East China Normal University, Shanghai 200062, China
[b] Huadong Hospital affiliated to Fudan University, 221 West Yan'an Road, Jing'an District, Shanghai 200040, China
[c] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510700, China
[d] Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 2 November 2022
Received in revised form 3 January 2023
Accepted 9 January 2023
Available online 14 January 2023

Keywords:
Entity matching
m-closest search
Heterogeneous information networks

ABSTRACT

This work investigates a novel m-closest entity (mCE) matching problem over geographic heterogeneous information networks (Geo-HINs). That is, given a Geo-HIN G and m query graphs $\{q_1, q_2, \dots, q_m\}$, mCE matching aims to find a group of geographic entities (geo-entities) whose patterns match the query graphs $\{q_1, q_2, \dots, q_m\}$ correspondingly, for which the maximum distance between any geo-entity pair (i.e., the diameter) in the group is minimized. As a fundamental problem, the mCE matching can be applied for many scenarios, e.g., travel itinerary recommendation and city planning. The existing works have not simultaneously considered the characteristics of patterns matching and spatial search so that they cannot solve our problem, which is computationally expensive. To solve this problem efficiently, we propose a unified framework named $Fuzzy - Exact$ framework to process entity matching and spatial search comprehensively, in which pruning abilities at non-spatial and spatial layers are cooperatively explored. Two mutually adaptive auxiliary data structures named $Arc - Tree$ and $Arc - Forest$ are devised to maintain the intermediate search results which are exploited to enhance the search process between non-spatial and spatial layers. Experimental results demonstrate that our algorithm can outperform the baseline methods by 2 orders of magnitude on runtime.

1. Introduction

Heterogeneous information networks (HINs) [1] are formed by multiple typed entities and multiple typed links to model various data, such as social networks, biology, and knowledge graphs. With the prevalence of location based service, plenty of entities in HINs are generated with location related links to identify their geographic info. For example, restaurants or hotels (entities) in Yelp network usually associate with location attributes to store their geographic coordinates. In social networks, such as Twitter, user entities usually contain check-in info and the micro-blogs are labeled with geo-tags. A Geo-HIN is a HIN, where at least one type of entity have geographic location attribute (named geo-entity). Fig. 1 shows a Geo-HIN example of Yelp network, which consists of eight geo-entities (i.e., $e_1, \dots, e_8$). Each geo-entity e_i links to node l_i denoting the location of e_i. Here a geo-entity e_i can be restaurant, hotel, park, or any object with location info. Besides, there are several types of non-geographic entities (non-geo-entities) in Fig. 1, such as user, feature, grade and so on. In this paper, we study the m-closest entity (mCE) matching problem over Geo-HINs. Below is a motivating example.

Example 1. Fig. 1 illustrates a Geo-HIN of Yelp network. u_1 is a customer of Yelp who requests a travel itinerary planning for an unfamiliar city. Fig. 2 shows three query graphs $\{q_1, q_2, q_3\}$ to express the requirements of u_1. In specific, the query graphs contain following contents "find a kids-friendly (t_1) restaurant e_x which is rated with five stars (g_1) and marked by seafood (f_1)", "find a kids-friendly (t_1) hotel e_y which has been booked by u_1's friend u_2 and tagged with swimming pool (f_2)", and "find a kids-friendly (t_1) and indoor (f_3) scenic spot e_z that has been visited by u_1's friend u_2 and is rated with five stars". Through graph matching, we have $e_x = \{e_2, e_5\}$, $e_y = \{e_3\}$, and $e_z = \{e_4\}$. Among the combination results of e_x, e_y, and e_z (i.e., $\{e_2, e_3, e_4\}$, $\{e_5, e_3, e_4\}$), mCE may recommend $\{e_5, e_3, e_4\}$ to u_1 as the geo-diameter is minimized.

The m-closest entity (mCE) matching problem over Geo-HINs can be further formulate as: given a Geo-HIN G and m query graphs in $\{q_1, q_2, \dots, q_m\}$ where each query graph describes the pattern of a geo-entity, the mCE matching aims to find a

* Corresponding author.
E-mail address: lipingwang@sei.ecnu.edu.cn (L. Wang).
[1] Wancheng Long and Xiaowen Li are the joint first authors.

https://doi.org/10.1016/j.knosys.2023.110299
0950-7051/© 2023 Elsevier B.V. All rights reserved.

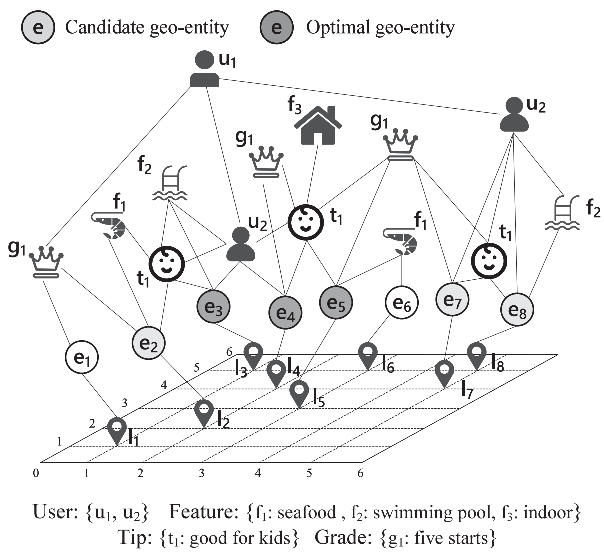
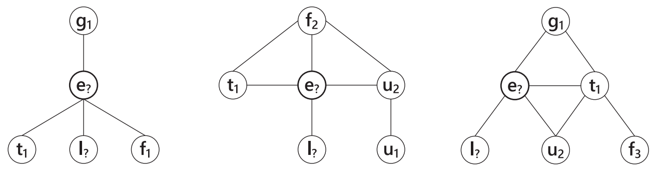

Candidate geo-entity
Optimal geo-entity
User: {u1, u2} Feature: {f1: seafood , f2: swimming pool, f3: indoor}
Tip: {t1: good for kids} Grade: {g1: five starts}

Fig. 1. A Geo-HIN example of Yelp Network.



(a) Query graph q_1	(b) Query graph q_2	(c) Query graph q_3

Fig. 2. Different query graphs.

group of geo-entities matching with the patterns of query graphs $\{q_1, q_2, \ldots, q_m\}$ correspondingly such that the diameter of the geo-entity group is minimized. Here the diameter of a group (aka geo-diameter) is defined as the largest geo-distance between any pair of geo-entities in the group.

Applications. The *m*CE matching can be used in many applications. As discussed in Example 1, it provides considerate travel itinerary planning to meet sophisticated requirements from users. This function is popular in tourist oriented APPs such as Wanderlog and Ctrip. Besides, for city planning, the locations of public facilities should be carefully considered. Given the specified public facilities, *m*CE matching can be used to evaluate the setting according to the POI group within a smallest range. Similar to travel planning, *m*CE matching can also be used to evaluate Internet of Things (IoT) system and location-based service system for recommending proper objects or users. Moreover, it is observed that when each query graph is as simplified as possible, e.g., just including geo-entity, *m*CE matching problem is equal to *m*CK query problem [2]. Thus, *m*CE matching may extend the applications of the *m*CK query.

Existing Studies. To the best of our knowledge, this is the first work to study the *m*-closest entities (*m*CE) matching over Geo-HINs. The most related work to *m*CE matching is *m*-closest keywords (*m*CK) query [2], an important type of the spatial keyword query studied in several existing works (e.g.,[2–4]). However, *m*CK query is located at the spatial database, which aims to find a group of objects that cover all keywords and guarantee the geo-diameter of the group is minimized. Our *m*CE matching is a more complex problem than the existing *m*CK query. The reason is that the *m*CK query solves the *m*-closest problem involving keywords and locations, and *m*CE matching need to address the complex relations between subgraphs matching and locations. For instance, in Example 1, *m*CE will return a POI that contains a kid-friendly swimming pool while *m*CK can only ensure the POI contains a swimming pool and is a kid-friendly place. Note that the relationship between the swimming pool and kid-friendly keywords cannot be represented by *m*CK, which will lead to the resulting swimming pool may be for adults only and it is not expected by the users. Moreover, all keywords are independent and the relationship between them cannot be explicitly specified in the *m*CK query, which leads to it cannot precisely represent the user demand and the query answer may deviate far from user expectation. In regard to this limitation, we will empirically discuss in Section 5.2. In contrast to the *m*CK, *m*CE matching can provide more semantic information and make query requirements more general. Thus, it is necessary to pursuit ingenious method to integrate the process of subgraph matching and spatial search for *m*CE matching.

Challenges and Our Solution. The challenges lie in two folds. The first challenge is that the pattern matching of entities and *m*-closest searching among candidate entities are two relatively independent tasks. To improve query efficiency, a novel cooperated framework should be devised to exploit the intermediate search results of these two search processes. However, the related existing studies mainly focus on the queries that have both textual and spatial constraints [5–10]. To the best of our knowledge, this is the first work to study the query that considers both entity pattern matching and spatial constraints. Compared with keywords, entity pattern matching can describe richer semantic information, while it also brings higher query complexity. Thus, it is important but hard to efficiently answer the queries that consider both entity pattern matching and spatial constraints.

The second challenge is that on account of subgraph matching and *m*-closest search which are both NP-hard [11] [2]. The search space will increase exponentially with the increasing number of vertices in query graphs, and no algorithm that can find the exact answer in polynomial time. In existing studies, the general solution is to return an approximate answer to avoid the large time cost [2,12–15]. However, for approximate solutions, we may sacrifice some properties on the result such as a relaxed spatial constraint or partially matched entities. Moreover, the result of early termination of the exact algorithm in the hard case can be considered as an approximate solution. As the *m*CE problem is studied for the first time, the investigation of exact algorithm is significant.

To address this new and practical problem, we propose an efficient *Fuzzy − Exact* framework to process entity matching and spatial search comprehensively. The cooperation of pruning strategies at both the non-spatial and spatial layers are explored to improve the efficiency of *m*CE matching. Specifically, we design a fuzzy mechanism to avoid exhaustive checking whether each geo-entity is actually eligible. Moreover, two auxiliary data structures named *Arc − Tree* and *Arc − Forest* are introduced to maintain the intermediate search results and then facilitate the search process.

Contributions. The principal contributions in this paper are summarized as follows.

- To our best knowledge, this is the first attempt to investigate the *m*-closest entity matching problem over geographic heterogeneous information networks. Inspired by real-world applications, *m*CE matching considers both the pattern matching and spatial search, which makes the problem more general than *m*CK.
- A novel *Fuzzy − Exact* framework is proposed, where the pruning abilities at non-spatial and spatial layers are cooperatively explored by integrating a fuzzy mechanism. Two mutually adaptive data structures named *Arc − Tree* and *Arc − Forest* are introduced to enhance the search process and make the framework more scalable.

Table 1
Summary of notations.

Notation	Definition
G, Q and q	a GeoHIN, a set of query graphs and a query graph
$\|Q\|, m$	the number of query graphs in a mCE matching problem
V_e, V_a	the set of geo-entites, and non-geo-entites
V_s	the set of spatial descriptors
$e(u, v)$	edge between u and v
$q.e$	geo-entity in the query graph q
M	an injective mapping
s_q^*	an exact matched geo-entity of q
$\mathbb{S}_q^*$	the set of all exact matched geo-entities of q
$H[V]$	vertex induced subgraph of graph H by giving V
$\|V_H\|$	the number of vertexes in graph H
S	a group of geo-entities
$\delta(S)$	diameter of S
$dist(e_i, e_j)$	Euclidean distance between geo-entities e_i and e_j
$N(v)$	neighbors of vertex v
$s_q^\sim$	a fuzzy matched geo-entity of q
$\mathbb{S}_q^\sim$	the set of all fuzzy matched geo-entities of q

- Extensive experimental results demonstrate the efficiency and scalability of the proposed framework on five real-world datasets with up to ten million vertices. Our techniques can achieve 2 orders of magnitude improvements on runtime compared with baselines.

2. Preliminaries

In this section, we first formally present some concepts in Section 2.1. Section 2.2 formally defines the problem of mCE matching, and Section 2.3 reviews related work.

Table 1 summarizes the notations frequently used in this paper.

2.1. Preliminary definitions

Definition 1 (*Geo-HIN*). A Geo-HIN is denoted as a graph $G = (V, E, \Delta, L)$, where (1) $V = V_e \cup V_a$ is a collection of vertices (entities). V_e represents the set of geo-entities and V_a is the set of non-geo-entities. (2) $E \subseteq V \times V$ is an edge set which represents relationships between entities; (3) Δ is a vertex label set; (4) $L : V \to \Delta$ is a mapping function, which associates a vertex v with a label $L(v) \in \Delta$.

Note that for ease of description, we assume that Geo-HIN is a vertex-labeled undirected simple graph and our techniques can be easily extended to edge-labeled directed graphs.

Definition 2 (*Spatial Descriptor*)**.** Spatial descriptors are denoted as a vertex set V_s ($V_s \subseteq V_a$), and each $v \in V_s$ describes the spatial information of a vertex $u \in V_e$.

If an entity has its own spatial information, i.e., it connects with a spatial descriptors, then we say such entity is a geo-entity. A formal definition is as follows.

Definition 3 (*Geo-Entity*)**.** Given a vertex $u \in V_e$, if there exists a neighbor $v \in V_s$, then such a vertex u is dubbed as a geo-entity.

In this study, we consider each $v \in V_s$ is a location of an entity in the 2-dimensional Euclidean space. We assume that each geo-entity has a unique spatial descriptor.

Example 2. Fig. 1 is a sample Geo-HIN based on the Yelp Open Dataset, and the geo-entities are represented by bolder circles, i.e., $V_e = \{e_1, e_2, \dots, e_8\}$ and the other vertices are non-geo-entities, i.e., $V_a = \{f_1, f_2, t_1, g_1, u_1, u_2, l_1, l_2, \dots, l_8\}$, $V_s = \{l_1, l_2, \dots, l_8\}$.

Definition 4 (*Geo-Entity Query Graph*)**.** A geo-entity query graph $q = (V_q, E_q, \Delta_q, L_q)$ is a Geo-HIN such that there exactly exists one geo-entity $v \in V_q$. We denote the geo-entity by $q.e$. For simplicity, we use query graph to denote geo-entity query graph when the context is clear.

Definition 5 (*Exact Match*)**.** Given a Geo-HIN G and a geo-entity query graph $q = (V_q, E_q, \Delta_q, L_q)$, a subgraph $H = (V_H, E_H, \Delta_H, L_H)$ of G is a exact match when there exists an injective mapping M from V_q to V_H that satisfy: (1) $\forall u \in V_q$, $L_q(u) = L_H(M(u))$; (2) $\forall e(u, v) \in E_q$, $\exists e(M(u), M(v)) \in E_H$. We say the geo-entity in H, i.e., $M(q_e)$, is an exact matched geo-entity of q in G and denote it by $s_q^*(G)$.

Note that there may be more than one exact matched geo-entity of q in G, thus we use $\mathbb{S}_q^*(G)$ to denote the set of all geo-entity results which belong to same exact matched query graph. When the context is clear, we abbreviate $s_q^*(G)$ and $\mathbb{S}_q^*(G)$ to s_q^* and $\mathbb{S}_q^*$, respectively.

Example 3. Given a Geo-HIN in Fig. 1 and query graphs in Fig. 2, there are two exact matches for q_1, i.e., $\mathbb{S}_{q_1}^* = \{e_2, e_5\}$. Similarly, for exact matched geo-entities of q_2 and q_3, the related geo-entities are $\{e_3\}$ and $\{e_4\}$, respectively.

Definition 6 (*Vertex Induced Subgraph*)**.** Given a Geo-HIN $H = (V, E, \Delta, L)$ and a set of vertices $V' \subseteq V$, a vertex induced subgraph of H constructed on V' is denoted as $H[V'] = (V', E', \Delta', L')$ where $E' = \{e(u, v) | e(u, v) \in E \wedge u \in V' \wedge v \in V'\}$.

Definition 7 (*Diameter of a Group* [2])**.** Given a group S of geo-entities, its diameter is defined as the maximum Euclidean distance between any pair of geo-entities in S, denoted by $\delta(S)$. Here $\delta(S) = max_{e_i, e_j \in S} dist(e_i, e_j)$, where $dist(e_i, e_j)$ is the Euclidean distance between geo-entitiy e_i and geo-entitiy e_j.

Definition 8 (*Feasible Group*)**.** Given a Geo-HIN G, a set of query graphs $Q = \{q_1, q_2, \dots, q_m\}$. If there is a group S of geo-entities such that for each query graph $q_i \in Q$, there exists exactly one exact matched geo-entity $s_{q_i}^* \in S$ correspondingly, then we call such a group S is a feasible group.

It is obvious that each geo-entity in a feasible group can satisfy the pattern of a query graph. As discussed in Example 3, the feasible group for $Q = \{q_1, q_2, q_3\}$ can be $\{e_2, e_3, e_4\}$ or $\{e_5, e_3, e_4\}$.

2.2. Problem definition

In this paper, we tackle the problem of m-closest geo-entities matching over Geo-HIN. Specifically, given a Geo-HIN $G = (V, E, \Delta, L)$ and a set of query graphs $Q = \{q_1, q_2, \dots, q_m\}$ where the cardinal number $m = |Q|$, an *m-closest geo-entities* query is to find a feasible group $S = \{s_{q_1}^*, s_{q_2}^*, \dots, s_{q_m}^*\} \subseteq V_e$ such that the diameter of the group $\delta(S)$ is minimized.

For simplicity, we denote *m-closest geo-entity* matching as *m-closest entity* (mCE for short) matching. The resulting feasible group is called the answer group when the context is clear. Determining exact matched geo-entities for each $q_i \in Q$ (without diameter optimization) is essentially subgraph isomorphism (matching) problem, which is proved to be an NP-Complete problem [11]. Besides, it is an NP-hard problem to find a group S of m geo-entities such that $\delta(S)$ is minimized [2].

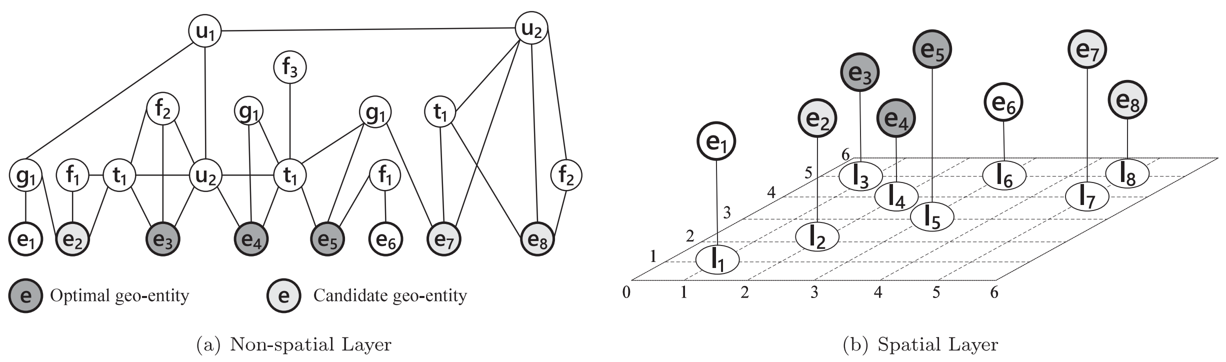


Fig. 3. Example of the hierarchical layers.

2.3. Related work

As the problem definition in Section 2.2, mCE matching is a problem that belongs to the general topics of subgraph matching and spatial query, which both have received a great deal of attention from the database community. We have mentioned that the most closely related problem to mCE matching is mCK query problem in Section 1. The mCK problem was firstly proposed by Zhang et al. [3,4] and has been well studied in [2]. Wan Choi et al. [16] proposed a variant of the mCK problem, but it is still a query under spatial and textual constraints. Nevertheless, the mCK problem only focuses on spatial and textual (keyword) attributes, and the semantic information described by keywords is much weaker than the info depicted by the entity patterns over HINs such as knowledge graphs. As shown in Section 1, mCK queries cannot fully resolve what users need in some scenarios.

For the query of entity patterns over HINs, subgraph matching plays a significant role in it [17–20][21]. Subgraph matching is a problem that wants to find all embeddings in a data graph that are isomorphic to a query graph (usually much smaller than the data graph), where both graphs are labeled graph and that has been extensively studied [22–25][26,27]. However, existing entity matching techniques [28–31][32] have not considered spatial information of entities so that they cannot tackle the mCE problem effectively. In the previous works about spatial constraints queries over HINs [33–36] [37–39], all of them suppose that every vertex in the network has its own spatial information (i.e., they are all geo-entities) so that their algorithms can prune search space both from the subgraph structure side and the spatial constraints side. According to the schema of most real knowledge graphs, we suppose there are both geo-entity and non-geo-entity in Geo-HIN. Here, the non-geo-entity cannot be pruned by spatial constraints and the existing algorithms do not work anymore in our case.

To utilize the existing works, the straightforward idea is to adopt subgraph matching and mCK algorithms step by step. That is filtering out geo-entities which do not satisfy the query graph pattern by subgraph matching first, then taking different eligible geo-entities as the different keywords, and eventually adopting mCK algorithm to get the answer. Unfortunately, this idea is computationally expensive. Our experiments (in Section 5.5) show that its performance almost identifies with the naive enumeration algorithm. It indicates that the adaption of state-of-the-art algorithms (mCK query and subgraph matching) is infeasible for answering the mCE matching problem.

3. Basic solution

To make the logic of the algorithms clearer, we can consider the Geo-HIN G from two layers: (1) non-spatial layer, a vertex induced subgraph (Definition 6) from all entities except spatial descriptors, i.e., $G[V_e \cup V_a/V_s]$. (2) spatial layer, a vertex induced subgraph from geo-entities and spatial descriptors, i.e., $G[V_e \cup V_s]$. Fig. 3 shows the two layers of the Geo-HIN in Fig. 1. After dividing Geo-HIN into two layers, for a high-level idea, we can match entities at the non-spatial layer for beginning, then conduct m-closest search among eligible entities at the spatial layer, and finally retrieve the answer.

Algorithm 1 shows the procedure of hierarchical search. First, we find all exact matched geo-entities $\mathbb{S}^*_{q_i}$ of each $q_i \in Q$ as the candidates of $s^*_{q_i}$. It is essentially a procedure of finding subgraph isomorphism. Sun et al. [27] have shown that there is no silver bullet for this problem. And they summarized which technique can cause better performance under different kinds of workloads. These are still applicable to mCE matching problem. Furthermore, Ren et al. [40] have studied some optimizing techniques for subgraph isomorphism problem when multiple query graphs arrive at the same time. These are naturally applicable for the mCE matching problem as well. When the searching process at the non-spatial layer is completed, we can then carry out m-closest search among the candidate geo-entities to obtain the answer. For ease of understanding, we start with a straightforward feasible group enumeration approach to carry out the m-closest search.

Example 4. Given a Geo-HIN in Fig. 1 and a set of query graphs in Fig. 2, to answer the mCE matching problem, we firstly find all exact matched geo-entities of each $q_i \in Q$, i.e., $\mathbb{S}^*_{q_1} = \{e_2, e_5\}$, $\mathbb{S}^*_{q_2} = \{e_3\}$, $\mathbb{S}^*_{q_3} = \{e_4\}$, and work out the candidate set $\mathbb{S}^* = \{e_2, e_3, e_4, e_5\}$. Then, we enumerate all feasible groups among the candidates, i.e., $S_1 = \{e_2, e_3, e_4\}$ and $S_2 = \{e_3, e_4, e_5\}$. Since $\delta(S_1) = dist(e_2, e_3) = \sqrt{10} > \delta(S_2) = dist(e_3, e_5) = 2\sqrt{2}$, S_2 is the answer of the mCE matching problem.

Although the basic solution can answer the mCE matching problem correctly, the vast time complexity makes it unpractical in the actual query scenario. At the non-spatial layer, finding all exact matched geo-entities of a query graph (line 3 in the Algorithm 1) is essentially a matter of subgraph isomorphism (match) problem, which has been proved to be an NP-Complete problem [11] and the worst case time complexity is $O(|V_G|^{|V_{q_i}|})$, where $|V_G|$ and $|V(q_i)|$ are the number of vertices in the Geo-HIN G and the query graph q_i respectively. Since we need to find all exact matched geo-entities for each $q_i \in Q$, the total time complexity at the non-spatial layer is $O(\Sigma_{q_i \in Q}|V_G|^{|V_{q_i}|})$. At the spatial layer, m-closest searching among eligible entities is also an NP-hard problem [2]. The time complexity of enumerating all feasible groups is $O(\Pi_{q_i \in Q}|\mathbb{S}^*_{q_i}|)$, where $|\mathbb{S}^*_{q_i}|$ is the number of all exact matched geo-entities of q_i. To calculate the diameter of a group, we need to enumerate every pair of geo-entities in the group, that is $\frac{m\cdot(m-1)}{2}$ calculations, thus the time complexity of

Algorithm 1: Hierarchical Search

```
Input  : A Geo-HIN $G$ and a set of query graphs
         $Q = \{q_1, q_2, ..., q_m\}$
Output : A group $S = \{s^*_{q_1}, s^*_{q_2}, ..., s^*_{q_m}\}$ of exact matched
         geo-entities for each $q_i \in Q$ such that $\delta(S)$ is
         minimized.
/*          non-spatial layer                          */
1 $\mathbb{S}^* \leftarrow \emptyset$;
2 foreach $q_i$ in $Q$ do
3     $\mathbb{S}^*_{q_i} \leftarrow$ find all exact matched geo-entities of $q_i$ from $G$;
4     $\mathbb{S}^* \leftarrow \mathbb{S}^* \cup \mathbb{S}^*_{q_i}$;
5 end foreach
/*          spatial layer                              */
6 $S \leftarrow m$-closest search among $\mathbb{S}^*$;
7 return $S$;
```

calculating the diameter of a group is $O(m^2)$. Therefore, the total time complexity at the spatial layer is $O(m^2 \cdot \Pi_{q_i \in Q} |\mathbb{S}^*_{q_i}|)$.

We can find that the time complexity at the non-spatial layer is much higher than that at the spatial layer and the search space at the spatial layer is the search results at the non-spatial layer. With the number of vertices in query graphs growing, it will lead to two problems: (1) search space at the non-spatial layer will increase exponentially at a much higher rate than that at the spatial layer. As a result, the non-spatial layer takes up most of the overall time overhead; (2) the number of eligible geo-entities will decrease after being filtered by the non-spatial layer. That is, search space of $\cup_{q_i \in Q} \mathbb{S}^*_{q_i}$ at the spatial layer will decrease. Thus, the pruning ability of the algorithm at the spatial layer will be degraded.

Correspondingly, our experiment results (in Section 5.4–5.5) also show these problems, that is, the non-spatial layer takes up almost entire query time consumption and the running time is still intolerable even though using the state-of-the-art algorithms at both the non-spatial layer and the spatial layer. The essence of these problems is that the hierarchical search processes are simply executed sequentially instead of sufficiently taking advantage of the intermediate search results to prune more search space.

4. Fuzzy-exact framework

To address those problems posed by the basic solution, we propose a *Fuzzy − Exact* framework. It aims to avoid the exhaustive exact matching for each geo-entity and effectively exploit the pruning ability at the spatial layer based on the intermediate results generated by the non-spatial layer. For the overview of *Fuzzy − Exact* framework, our main idea is that we initially obtain the geo-entities that are probably exact matched geo-entities at a fraction of the cost, then pruning them at the spatial layer to narrow down the overall search space. Here, we introduce our unified *Fuzzy − Exact* framework from the following two phases, that is, *exact to fuzzy phase* and *fuzzy to exact phase*.

(1) *exact to fuzzy phase.* In this phase, a fuzzy mechanism is designed, which allows generating approximate intermediate results (fuzzy matched geo-entities) instead of finding all exact matched geo-entities $\mathbb{S}^*_{q_i}$ for each $q_i \in Q$ at the beginning. Then those intermediate results will be inputted into the m-closest search module of the spatial layer as the candidate set.

(2) *fuzzy to exact phase.* In this phase, we adequately employ the pruning ability of the spatial layer to improve efficiency and conduct m-closest search to find the m-closest geo-entities, where we give a Theorem 1 to guarantee the answer group satisfying the requirement. Note that m-closest search from the second loop is based on the hints of previous results, which can speed up the searching process.

Last, those two phases are integrated into a unified framework to solve this novel problem by the cooperation of pruning abilities at the non-spatial layer and the spatial layer, which can significantly improve the searching performance and will be empirically discussed later in Section 5.4. Concretely, the pruning strategy at the non-spatial layer allows generating approximate intermediate search results, which not only avoids exhaustive exact matching but also exploits the powerful pruning strategy at the spatial layer. After completing the m-closest search at the spatial layer, the results will be inputted to the non-spatial layer, where we will check whether geo-entities are eligible; if not, the new loop of cooperation between two layers will continue based on the hints from the last loop until we find the required answer group. In the following subsections, we will introduce the details of how these two phases work and how to use the *Fuzzy − Exact* framework to answer the mCE matching problem efficiently.

4.1. Exact to Fuzzy phase

To obtain the geo-entities which are approximately exact matched geo-entities at a fraction of the cost and meanwhile do not filter any geo-entities that are achieved by exact matching. Here, we introduce a fuzzy mechanism, which will relax the restrict of the exact matching.

Definition 9 (*Fuzzy Query Graph*)**.** Given a query graph q, the fuzzy query graph $F(q)$ of q, is a connected subgraph of q such that contains exactly one geo-entity.

In general, we define fuzzy query graph is the ego network of $q.e$, i.e., $F(q) = q[\{q.e\} \cup N(q.e)]$. Because the ego network can represent the major characteristics of the geo-entity. Besides, the impact of different fuzzy query graph definitions on time overhead will be empirically discussed in Section 5.5.

Definition 10 (*Fuzzy Match*)**.** Given a Geo-HIN G and a query graph q, a fuzzy match of q is an exact match of the fuzzy query graph $F(q)$.

We use $\mathbb{S}^{\sim}_q(G)$ to denote the set that contains all fuzzy matched geo-entities of q in G. For simplicity, we use the $s^{\sim}_q$ and $\mathbb{S}^{\sim}_q$ to take the place of $s^{\sim}_q(G)$ and $\mathbb{S}^{\sim}_q(G)$ respectively when the context is clear.

Example 5. Given a Geo-HIN in Fig. 1. and a set of query graphs in Fig. 2., if we define $F(q_i) = q_i[\{q_i.e\} \cup N(q_i.e)]$ for each query graph, then the fuzzy matched geo-entities of q_1, q_2, q_3 are $\{e_2, e_5\}$, $\{e_3, e_8\}$, $\{e_4, e_7\}$ respectively.

Theorem 1. *Given a query graph q, we have* $\mathbb{S}^*_q \subseteq \mathbb{S}^{\sim}_q$

Proof. Let $M(q.e)$ is an exact matched geo-entity of the query graph $q = (V_q, E_q, \Delta_q, L_q)$, i.e., $M(q.e) \in \mathbb{S}^*_q$, and $g = (V_g, E_g, \Delta_g, L_g)$ is a subgraph of G, then we can get (i) $\forall u \in V_q$, $L_q(u) = L_g(M(u))$ and (ii) $\forall e(u, v) \in E_q$, $\exists e(M(u), M(v)) \in E_g$ hold according to Definition 5. Let $F(q) = (V_{F(q)}, E_{F(q)}, \Delta_{F(q)}, L_{F(q)})$ is a fuzzy query graph of q. According to Definition 9, we can know that $\forall u' \in V_{F(q)}$, $u' \in V_q$ (iii) and $\forall e(u', v') \in E_{F(q)}$, $e(u', v') \in E_q$ (iiii). Combining (i) with (iii), we can get $\forall u' \in V_{F(q)}$, $L_{F(q)}(u) = L_g(M(u))$. Combining (ii) with (iiii), we can get $\forall e(u', v') \in E_{F(q)}$, $e(M(u'), M(v')) \in E_g$. Hence $M(q.e) \in \mathbb{S}^{\sim}_q$ according to Definition 10.

Theorem 1 guarantees that we can spend less cost retrieving fuzzy matched geo-entities of q without missing any geo-entity that is an exact matched geo-entity precisely. Besides, the search depth for finding exact matched geo-entities of query graph q is $|V_q|$ and finding fuzzy matched geo-entities is $|V_{F(q)}|$ (that is

Algorithm 2: Fuzzy to Exact

```
Input  : A set $\mathbb{S}^{\sim}$ of candidates obtained from the exact to fuzzy phase
Output : A group $S = \{s^*_{q_1}, s^*_{q_2}, \ldots, s^*_{q_m}\}$ of exact matched
         geo-entities for each $q_i \in Q$ such that $\delta(S)$ is minimized
1  while $|\mathbb{S}^{\sim}| \geq m$ do
2      $S \leftarrow$ m-closest search among $\mathbb{S}^{\sim}$;
3      validate $\leftarrow$ true;
4      foreach $s^{\sim}_{q_i} \in S$ do
5          if $s^{\sim}_{q_i}$ is not a precise exact matched geo-entity then
6              $\mathbb{S}^{\sim} \leftarrow \mathbb{S}^{\sim} / \{s^{\sim}_{q_i}\}$;
7              validate $\leftarrow$ false;
8              break;
9          end if
10     end foreach
11     if validate = true then
12         return $S$;
13     end if
14 end while
15 return $\emptyset$;
```

$|N(q.e)|$+1 in Example 5) in the worst case. Therefore, we can reduce the time overhead at the non-spatial layer by controlling the structures of fuzzy query graphs. Based on this observation, we can obtain the $\mathbb{S}^{\sim}_q$ with less cost initially and then conduct the m-closest search among $\mathbb{S}^{\sim}_q$. Thus, we can utilize spatial prune ability to avoid needlessly searching at the non-spatial layer, which not only reduces the time overhead at the non-spatial layer but also makes the algorithm at the spatial layer more effective.

4.2. Fuzzy to exact phase

In the *fuzzy to exact phase*, we have obtained the candidates from the *exact to fuzzy phase*. Then, we need to conduct the m-closest search among them to find the answer group. However, there may be some *dummy vertices* (i.e., $e \in \mathbb{S}^{\sim}$ and $e \notin \mathbb{S}^*$) in the m-closest search result according to Theorem 1. Thus, after retrieving the m-closest geo-entities at the spatial layer, we should turn over each geo-entity to the non-spatial layer for checking whether it is a precise exact matched geo-entity. We call such a procedure cooperation loop between the non-spatial and spatial layers. We keep running the cooperation loop until find a group such that each geo-entity in it is a precise exact matched geo-entity, then achieve the answer group. Algorithm 2 summarizes the above procedures.

From Algorithm 2, we can find that the key procedure in the *fuzzy to exact phase* is the m-closest search. Although the state-of-the-art m-closest search algorithm *EXACT*[2] has been well studied, its prerequisite is there is no *dummy vertex*. Thus, *EXACT* cannot be applied to the *fuzzy to exact phase* directly. Next, we will introduce the m-closest search procedure in our solution.

4.2.1. Basic m-closest search

For a basic m-closest search, it has been proved that the answer group can be covered by a circle that the diameter of the circle is $\frac{2}{\sqrt{3}} \cdot d$[2], where d is the upper bound of the diameter of the answer group. We call such a theorem *Enclosing Theorem*, such a circle *Enclosing Circle* and use D to denote its diameter. Based on *Enclosing Theorem*, we have the following theorem further.

Theorem 2. *Let S be the answer group. Given $\mathbb{S}^{\sim} = \bigcup_{q_i \in Q} \mathbb{S}^{\sim}_{q_i}$ which is the set of candidates from the exact to fuzzy phase, if $S \neq \emptyset$, then there must exist an exact matched geo-entity $e_{pivot} \in \mathbb{S}^{\sim}$, S within the sweeping area of e_{pivot}, where the sweeping area of e_{pivot} is a circle with e_{pivot} as the center and D as the radius.*

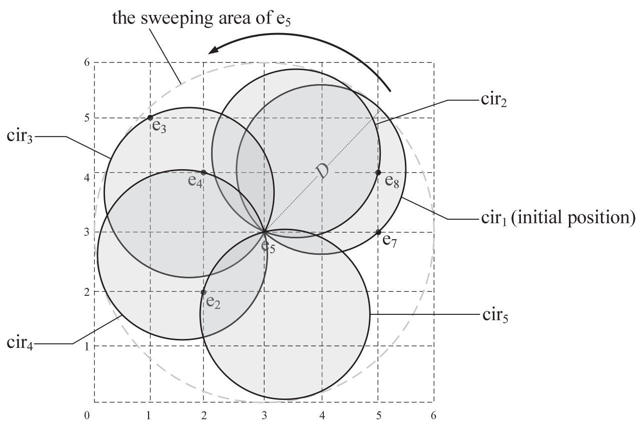


Fig. 4. A Example of the circle-scan procedure with e_5 as the pivot.

Proof. Since S is the answer group and $S \neq \emptyset$, there must exist an exact matched geo-entity $s^*_{q_i} \in \mathbb{S}^{\sim}_{q_i}$ in S according to Theorem 1. Suppose $s^*_{q_j}$ is another arbitrary exact matched geo-entity in S and $s^*_{q_j}$ is not within the sweeping area of $s^*_{q_i}$, i.e., $\exists s^*_{q_{j(j\neq i)}}$ in S, $dist(s^*_{q_j}, s^*_{q_i}) > D$, then $\delta(S) > D$ according to Definition 7. It is contradictory to $\delta(S)$ is minimized. Hence, there must exist an exact matched geo-entity $s^*_{q_i} \in \mathbb{S}^{\sim}_{q_i}$, S within the sweeping area of $s^*_{q_i}$ if $S \neq \emptyset$.

According to Theorem 2, we can obtain the answer group by finding *Enclosing Circles* within the sweeping area of e_{pivot}. To achieve this efficiently, the subprocedure *circle-scan* of *EXACT* will be applied. Example 6 shows a running example of it.

Example 6. As shown in Fig. 4, the candidates we obtained from the exact to fuzzy phase is the fuzzy matched geo-entities in Example 5, i.e., $\{e_2, e_3, e_4, e_5, e_7, e_8\}$ and we take e_5 as the pivot. In circle-scan procedure, we initial fix e_7 on the boundary of Enclosing Circle to determine the initial position of the circle, i.e., cir_1. Currently, the covered vertices are e_5, e_7, e_8 and can form a feasible group, thus we record the group $\{e_5, e_7, e_8\}$. We keep rotating the circle counterclockwise until it returns to its initial position. After that, we can get Enclosing Circles $cir_1, cir_2, cir_3, cir_4, cir_5$ and feasible groups $S_1 = \{e_5, e_7, e_8\}$, $S_2 = \{e_3, e_4, e_5\}$.

Since each vertex in the candidate set could be the pivot, we should conduct *circle-scan* procedure on each of them to obtain their feasible groups, and the answer group must be one of them according to Theorem 2.

Therefore, the last problem we need to resolve is finding the upper bound of the diameter of the answer group. To achieve this, we first pick an arbitrary exact matched geo-entity $s^*_{q_i}$ among candidates. Then retrieving the fuzzy matched geo-entities nearest to it for each $q_j (j \neq i) \in Q$ (this procedure can use the index that proposed by [4]). After that, we check these fuzzy matched geo-entities whether each of them is a precise exact matched geo-entity. If so, we keep it. Otherwise, we remove it from candidates and retrieve the nearest one once again until we find the exact matched geo-entities that can form a feasible group for fuzzy query graphs. The diameter of the group formed by the vertices is the upper bound of diameter of the answer group. Note that picking a specific initial exact matched geo-entity is beneficial to shrink the upper bound, but our preliminary experiments find that it cannot pay off the corresponding time overhead because of the existence of *dummy vertices*. The proposed upper bound well balances the computation cost and the bound accuracy.

Compared to the enumeration approach in the m-closest search procedure of the Algorithm 1, time complexity of that in the *fuzzy* *to exact phase* will become $O(m^2 \cdot \Sigma_{q_i \in Q} \Sigma_{s^{\sim}_{q_i} \in \mathbb{S}^{\sim}_{q_i}} \omega(s^{\sim}_{q_i})^{|V_{F(q_i)}|})$, where $\omega(s^{\sim}_{q_i})$ is the number of vertices within the sweeping area of $s^{\sim}_{q_i}$ in the worst case and $\omega(s^{\sim}_{q_i}) \ll |\mathbb{S}^{\sim}_{q_i}|$ generally.

Nevertheless, it is worth noting that we need to conduct the m-closest search at the beginning of each cooperation loop. If each round of the m-closest search is all from scratch, it may suffer from the inefficiency of repetitive scanning. Based on this observation, we want to collect some intermediate results as the hints during the m-closest search to speed up subsequent searches. The main challenge in doing this is how to preserve the spatial information to make the minimum diameter feasible group retrieving efficient during the cooperation loop. At the same time, the less additional overhead introduced, the better. We have an insight that the polar coordinate system can easily represent the spatial transformation of *circle-scan*. And parameter variation in the transformation is the natural intermediate search result with negligible collection overhead. Based on this, we design auxiliary data structures named *Arc − Tree* and *Arc − Forest* to eliminate the repetitive scanning. In the next section, we will give the details.

4.2.2. Enhanced m-closest search

During the basic m-closest search, we can observe that each vertex within *Enclosing Circle* falls on the boundary of the circle exactly twice (entering the circle and leaving the circle) during the rotation. Therefore, we can build a polar coordinate system for the vertices within the sweeping area. Each vertex is represented by an angle from a particular direction and a distance from the *pivot*. Based on the polar coordinate system, we can collect the intermediate search results of the basic m-closest search.

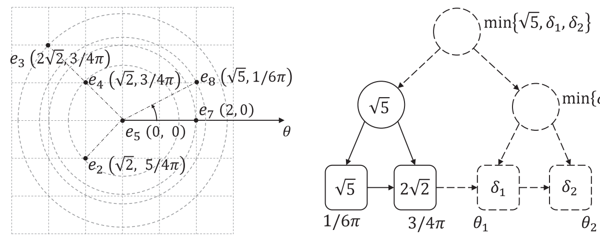
An example of the polar coordinate system is shown in Fig. 5(a). In our polar coordinate system, we use counterclockwise angles where the 3 o'clock position is 0. We determine the initial position of *Enclosing Circle* by fixing the vertex that has the smallest angle in the polar coordinate system on the boundary of the circle. We rotate it counterclockwise whenever a vertex is about to leave the boundary of the circle and such the vertex named *anchor vertex*. Obviously, there is at least one feasible group at the same time. We record the following information as the intermediate search results during the *circle-scan* procedure.

- *angle*: angle of the *anchor vertex* in the polar coordinate system.
- S_{opt}: a feasible group that $\delta(S_{opt})$ is minimized among all feasible groups within the circle. We call such a group is the optimal group of the *anchor vertex*.
- $\delta(S_{opt})$: diameter of the S_{opt}.
- *Tab*: a table of cardinal numbers of each $\mathbb{S}^{\sim}_{q_i}$ within the circle, i.e., $Tab[q_i] = |\mathbb{S}^{\sim}_{q_i}|$, where $|\mathbb{S}^{\sim}_{q_i}|$ is the number of fuzzy matched geo-entities of q_i within the circle.
- *cover*: vertices are covered by the circle.

The above information as the intermediate search results is useful for enhancing the subsequent searches owing to avoiding searching from scratch. The simplest way to maintain them is using a sorted list so that we can retrieve the feasible group with the minimum diameter efficiently. However, when we find the *dummy vertices* and update the corresponding group, it need $O(N)$ time complexity to reorder them, where N is the number of intermediate search results. To make the m-closest search more scalable, we design auxiliary data structures named *Arc − Tree* and *Arc − Forest* with $O(logN)$ time complexity both in query and update of the intermediate search results. The performance between the sorted list and the *Arc − Tree* & *Arc − Forest* will be empirically discussed in Section 5.5. Subsequently, we will elaborate the definitions of *Arc − Tree* & *Arc − Forest* and how to utilize them in the m-closest search.

Algorithm 3: Construct *Arc-Tree*

```
Input  : A pivot $e_i$ and a set $SA_{e_i}$ of candidates that within the
         sweeping area of $e_i$
Output : Arc-Tree $AT_{e_i}$
/* initiation                                                      */
$P \leftarrow \emptyset$;
foreach $e_j \in SA_{e_i}$ do
    $e_j$-in $\leftarrow$ getInAngle();
    $P$.addTuple($e_j$-in,in,$e_j$);
    $e_j$-out $\leftarrow$ getOutAngle();
    $P$.addTuple($e_j$-out,out,$e_j$);
end foreach
/* circle-scan                                                     */
sort $P$ by angle in ascending order;
initialize $Tab$;
$Cover \leftarrow \emptyset$;
foreach tuple(angle, type, $e_j$) $\in P$ do
    if type = in then
        $Cover \leftarrow Cover \cup \{e_j\}$;
        $Tab[q_j] \leftarrow Tab[q_j] + 1$;
    end if
    else
        if $\forall q_i \in Q$, $Tab[q_i] > 0$ then
            $LN \leftarrow$ current intermediate search results;
            if $AT_{e_i} \neq \emptyset$ then
                $AT_{e_i}$.lastNode.next $\leftarrow LN$;
            end if
            $AT_{e_i}$.addLeafNode($LN$);
        end if
        $Cover \leftarrow Cover / \{e_j\}$;
        $Tab[q_j] \leftarrow Tab[q_j] - 1$;
    end if
end foreach
return $AT_{e_i}$;
```


e_3 $(2\sqrt{2}, 3/4\pi)$
e_4 $(\sqrt{2}, 3/4\pi)$
e_8 $(\sqrt{5}, 1/6\pi)$
e_7 $(2, 0)$
e_5 $(0, 0)$
θ
e_2 $(\sqrt{2}, 5/4\pi)$
$\min\{\sqrt{5}, \delta_1, \delta_2\}$
$\sqrt{5}$
$\min\{\delta_1, \delta_2\}$
$\sqrt{5}$
$2\sqrt{2}$
δ_1
δ_2
$1/6\pi$
$3/4\pi$
θ_1
θ_2

(a) A Polar Coordinate System (b) An *Arc-Tree*

Fig. 5. Examples of the polar coordinate system and the *Arc − Tree*.

Definition 11 (*Arc-Tree*)**.** For a pivot e, the Arc-Tree of e, denoted by AT_e, is a binary search tree that takes $\delta(S_{opt})$ recorded in the leaf node as the comparison key. Each time the information record we retrieved from the AT_e is the leaf node that records the minimum diameter feasible group among all groups within the sweeping area of e.

We construct an *Arc − Tree* by a bottom-up approach. The information record of each *anchor vertex* will be taken as the leaf node of the *Arc − Tree*. And the inner nodes will be recursively generated to record the smaller diameter between its children. Along this way, the root node will finally record the minimum diameter in all feasible groups within the sweeping area of the pivot. It is worth noting that the leaf nodes are ordered by their *angle* naturally because the circle is rotating counterclockwise. Accordingly, we can make a pointer for each leaf node to point to its next leaf node except the last leaf node so that we can retrieve the affected leaf nodes efficiently when we find the *dummy vertex*.

Algorithm 4: Construct *Arc − Forest*

```
Input  : A set $\mathbb{S}^{\sim}$ of candidates obtained from the exact to fuzzy phase
Output : $Arc - Forest$
1 $Arc - Forest \leftarrow \emptyset$;
2 foreach $e_i \in \mathbb{S}^{\sim}$ do
3     $SA_{e_i} \leftarrow$ vertices within the sweeping area of $e_i$;
4     $AT_{e_i} \leftarrow contructArcTree(e_i, SA_{e_i})$;
5     $Arc - Forest.add(AT_{e_i}.root)$;
6 end foreach
7 return $Arc - Forest$;
```

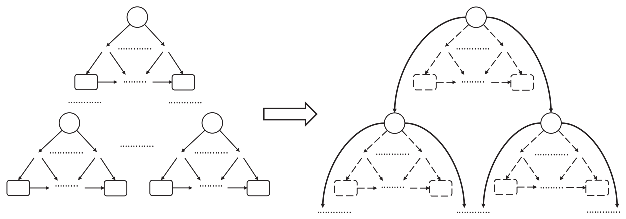


Fig. 6. Construction of the *Arc − Forest*.

Algorithm 3 shows the procedure of constructing *Arc − Tree*. After initialization, we calculate the *in − angle* and *out − angle* which are the coordinates of the *Enclosing Circle*'s center at the moments that the *anchor vertex* enters the circle and leaves the circle. During the rotation, to narrow down the subsequent search space, we create an *Arc − Tree* leaf node only if $\forall q_i \in Q$, $Tab[q_i] > 0$ holds. The reason is that vertices within the circle cannot form a feasible group. An example of the *Arc − Tree* is shown in Fig. 5(b).

Example 7. As shown in Fig. 5(a), we build a polar coordinate system and e_5 is selected as the pivot. We initially fix e_7 on the boundary of the *Enclosing Circle* as it has the smallest angle, i.e., $1/6\pi$. At this moment, each element in *Tab* is greater than 0, thus we record following information: (1) $angle = 1/6\pi$; (2) $S_{opt} = \{e_5, e_7, e_8\}$; (3) $\delta(S_{opt}) = \sqrt{5}$; (4) $Tab = \{q_1{:}1, q_2{:}1, q_3{:}1\}$; (5) $cover = \{e_5, e_7, e_8\}$. We keep rotating the circle, and *Tab* is updated to $\{q_1{:}1, q_2{:}1, q_3{:}0\}$. Thus, we know that there is no feasible group within the cir_2. After the circle returns to its initial position, we get the feasible groups $S_1 = \{e_5, e_7, e_8\}$ and $S_2 = \{e_3, e_4, e_5\}$. Therefore, there are two leaf nodes in the *Arc − Tree* which are represented by the part of solid line in Fig. 5(b). And the parent node of the leaf nodes records the smaller diameter between them, i.e., $\sqrt{5}$. Note that, as shown in Fig. 5(b), the part of dash line means that if there are more feasible groups, we can construct the tree as the process described above recursively.

Generally, there are many *Arc − Trees* since each vertex among candidates could be the pivot. Spontaneously, we design a mutual data structure *Arc − Forest* to avoid linearly scanning all *Arc − Trees*, in which a min-heap is used to maintain the root nodes of *Arc − Trees*. The definition of *Arc − Forest* will be given as follows.

Definition 12 (*Arc-Forest*)**.** An *Arc − Forest* is a min-heap that maintains the root nodes of *Arc − Trees* by using *Arc − Tree*.root.diameter as the comparison key. Each time the node retrieved from the *Arc − Forest* is the root node of the *Arc − Tree* that contains the minimum diameter feasible group among all groups stored in *Arc − Trees*.

Algorithm 4 shows the procedure of constructing an *Arc − Forest*. We can obtain the upper bound *d* of the answer group's diameter from the *greedy m-closest search*. And for each candidate that obtained from the *exact to fuzzy phase*, we find vertices within the sweeping area of it and construct a corresponding *Arc − Tree*. At last, we maintain the root nodes of these *Arc − Trees* by a min-heap, just like Fig. 6 shows.

Moreover, except for the procedure of checking a candidate whether it is a precise exact matched geo-entity, an *Arc − Forest* contains all information that *m*-closest search need, so we only need to conduct the subsequent procedures of *m*-closest search on the *Arc − Forest*, instead of the whole Geo-HIN.

After the construction of *Arc − Tree* and *Arc − Forest*, their beneficial properties below will be applied to enhance the basic *m*-closest search.

Property 1. *The diameter of the top root node in an Arc − Forest is always the minimum among all candidate feasible groups.*

Proof. The proof is straightforward as the diameter of the root node of an *Arc − Tree* is the minimum among all feasible groups within the sweeping area of the *pivot*. And the top root node in an *Arc − Forest* is the root node that has the minimum diameter among all *Arc − Trees*.

Property 2. *$\forall AT_e \in Arc - Forest$, $AT_e.root.diameter$ is monotonically increasing after updating.*

Proof. Suppose $S_{initial}$ is the group that we will retrieve to update initially from the AT_e and its diameter is *d* before updating. If no update is needed, the algorithm directly returns because we have already found the answer. Otherwise, assuming that there is a group $S_{smaller}$ such that $\delta(S_{smaller}) < d$ after updating the $S_{initial}$. Because of $\delta(S_{smaller}) < d$, the group we retrieve to update from AT_e initially should be $S_{smaller}$ rather than $S_{initial}$ according to Definition 11. It is a contradiction. Hence, $\forall AT_e \in Arc - Forest$, $AT_e.root.diameter$ is monotonically increasing after updating.

Property 3. *If $e \notin \mathbb{S}^*$, then we can prune the whole Arc − Tree AT_e safely.*

Proof. The proof is straightforward as there must be an exact matched geo-entity $e_{pivot} \in \mathbb{S}^{\sim}$ and the answer group within the sweeping area of e_{pivot} according to Theorem 2, and *e* is not the e_{pivot}.

According to Property 1, we can easily retrieve the feasible group that has the minimum diameter among all candidate feasible groups. If there are all precise exact matched geo-entities in the group, we can return it directly because it is exactly the answer group according to Property 2. After retrieving the feasible group from the AT_e, we first check the *pivot e* if it is an exact matched geo-entity, and we can prune the whole *Arc − Tree* of it safely according to Property 3 if not. After ensuring the *pivot e* is an exact matched geo-entity, we check the other vertices in the group. When we find there is a vertex $s^{\sim}_{q_i}$ that is not an exact matched geo-entity, we first check if it is the last candidate for $s^*_{q_i}$, i.e., $Tab[q_i] = 1$. If so, we can delete the leaf node because we know that there is no feasible group within the *Enclosing Circle*. And it is worth noting that we set the diameter of the leaf node to infinity instead of deleting it actually so that the *Arc − Tree* can always keep balance without losing the correctness of the result. Correspondingly, if the diameter of the top of *Arc − Forest* is infinity, it represents that we have checked all candidates, and there is no valid answer. When the leaf node is updated, *Arc − Tree* and *Arc − Forest* can be rapidly updated accordingly. The efficient adaptive update feature between *Arc − Tree* and

Algorithm 5: Enhanced m-closest Search

```
Input  : $Arc - Forest$
Output : A group $S = \{s^*_{q_1}, s^*_{q_2}, ..., s^*_{q_m}\}$ of exact matched
         geo-entities of each $q_i \in Q$ such that $\delta(S)$ is minimized
while $Arc - Forest$ is not empty do
    $AT_e \leftarrow Arc - Forest.top()$;
    if $AT_e.\delta(S_{opt}) = \infty$ then
        return $\emptyset$;
    end if
    if $e \notin \mathbb{S}^*$ then
        $Arc - Forest.pop()$;
        continue;
    end if
    $LN \leftarrow AT_e.retrieve()$;
    $validate \leftarrow true$;
    foreach $s^{\sim}_{q_i} \in LN.S_{opt}$ do
        if $s^{\sim}_{q_i}$ is not a precise exact matched geo-entity then
            if $LN.Tab[q_i] = 1$ then
                $LN.\delta(S_{opt}) \leftarrow \infty$;
            end if
            else
                $AT_e.update(LN)$;
            end if
            $Arc - Forest.update()$;
            $validate \leftarrow false$;
            break;
        end if
    end foreach
    if $validate = true$ then
        return $LN.S_{opt}$;
    end if
end while
return $\emptyset$;
```

Arc−Forest enhances the search process and make the framework more scalable. The enhanced m-closest search procedure based on *Arc − Tree* and *Arc − Forest* is summarized in Algorithm 5.

Time and Space Complexity. Let $\Phi(e_{pivot})$ be the set of intermediate search results of the m-closest search with e_{pivot} as the pivot. Time complexity of obtaining the leaf node that contains the minimum diameter group from an *Arc − Tree* is $O(log|\Phi(e_{pivot})|)$ as it is a binary search tree inherently. The total number of the nodes of an *Arc − Tree* is $|\Phi(e_{pivot})| + \lceil\frac{|\Phi(e_{pivot})|}{2}\rceil + \lceil\frac{|\Phi(e_{pivot})|}{4}\rceil + \cdots + 1 \leq 2|\Phi(e_{pivot})| + \lceil log_2|\Phi(e_{pivot})|\rceil$, thus the space complexity of an *Arc − Tree* is $O(|\Phi(e_{pivot})|)$. Since the *Arc − Forest* is a min-heap essentially, time complexity of obtaining the root node of an *Arc − Tree* from it is $O(log\,\Sigma_{s^{\sim}_q \in \mathbb{S}^{\sim}_q}\omega(s^{\sim}_{q_i}))$ and the space complexity is $O(\Sigma_{s^{\sim}_q \in \mathbb{S}^{\sim}_q}\omega(s^{\sim}_{q_i}))$. Thus, the overall time complexity of the whole *Fuzzy − Exact* framework is $O(\Sigma_{q_i \in Q}|V_G|^{|V_{F(q_i)}|} + m^2 \cdot \Sigma_{s^{\sim}_{q_i} \in \mathbb{S}^{\sim}_{q_i}}\omega(s^{\sim}_{q_i})^{|V_{F(q_i)}|} + \tau(AT_{s^{\sim}_{q_i}}) \cdot |LN_i.cover|^2 \cdot log\ \omega(s^{\sim}_{q_i})|\Phi(s^{\sim}_{q_i})|)$, where $\tau(AT_{s^{\sim}_{q_i}})$ is the number of times of querying the *Arc−Tree* $AT_{s^{\sim}_{q_i}}$ and LN_i is the corresponding leaf node retrieved by the query.

Since the matching problem at the non-spatial layer is NP-Complete [11], and the exhaustive search for the m-closest problem at the spatial layer is inevitable has been proved in [2]. We develop several optimizing strategies exploiting properties at both the non-spatial layer and the spatial layer to improve efficiency.

Optimizing Strategy 1. *Suppose AT_{e_i} contains the feasible group that has the minimum diameter. According to Property 2, the corresponding leaf node may not be retrieved after updating, because there may be a better feasible group in another Arc − Tree AT_{e_j}. Based on this observation, we can do the lazy update, i.e., we conduct updates only when we have retrieved the leaf node, instead of updating all affected leaf nodes at once.*

Optimizing Strategy 2. *In most cases, Enclosing Circles are likely to overlap, which means a vertex may appear in more than one candidate group. It will lead to the vertex may be checked whether it is an exact matched geo-entity more than once. Based on this observation, we can make a cache to assure that the candidate vertices will be actually checked only once, i.e., return the cached check result directly if the vertex has been checked before.*

Optimizing Strategy 3. *Tian et al. [13] proposed a generalized two level index for the approximate subgraph matching. First level of the index is B$^+$Tree, which stores the label, degree, neighbor connection among vertices in the graph. And the Bloom filter [41] is used in the second level of the index to store the label information of the neighbors of the vertex (we call such an index BLOOM index). We can take advantage of the index to obtain the fuzzy matched geo-entities in exact to fuzzy phase efficiently, instead of conducting costly subgraph matching procedure.*

5. Experimental evaluation

In this section, extensive experiments are conducted to verify both the effectiveness and the efficiency of the algorithms. All source code and datasets used in the experiment are available in "https://github.com/Morgan279/mCE".

5.1. Experiment setup

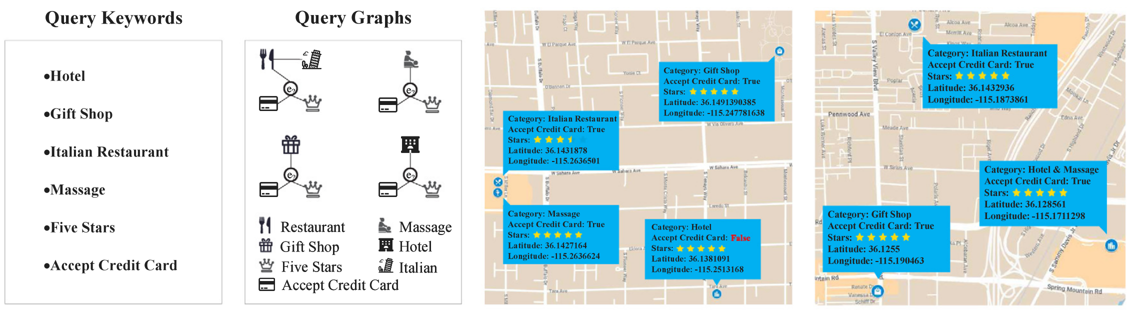
Table 2
Statistics of datasets.

Dataset	$\lvert V\rvert$	$\lvert V_e\rvert$	$\lvert E\rvert$
Yelp (**YP**)	24,153,607	241,642	70,148,956
Foursquare (**FS**)	18,920	2,603	16,348
Wikidata (**WD**)	53,585	2,277	51,334
Gowalla (**GW**)	4,005,215	209,393	3,795,822
Brightkite (**BK**)	2,991,662	156,449	2,835,213

Datasets. In our experiments, we use 5 public real-world datasets, whose statistics are summarized in Table 2. Yelp Open Dataset[2] is a subset of businesses of Yelp, reviews, and user data. The dataset contains entities (businesses) with both highly rich geo-entities and non-geo-entities. Foursquare[3][42] is a location technology platform offering business solutions and consumer products. Wikidata[4] is a free and open knowledge base [43]. The data used in the experiment was collected from the English Wikipedia (December 2018). Since the original dataset Foursquare and Wikidata lack specific coordinate information to calculate distance between geo-entities, we obtain the spatial descriptors from the Yelp Open Dataset for them. Gowalla[5] and Brightkite[6] are location-based social networking website where users share their locations by checking-in. Since they have only spatial descriptors and no geo-entities, we retrieve the geo-entities from the Yelp Open Dataset for them.

Queries. To construct queries in our experiments, we consider three factors: $|V_q|$, $|Q|$ and the dataset size, where $|V_q|$ and $|Q|$ are the number of vertices in a query graph and the number of query graphs in a query respectively. And we control dataset

2 https://www.yelp.com/dataset
3 https://sites.google.com/site/yangdingqi/home/foursquare-dataset
4 http://snap.stanford.edu/data/wikipedia-article-networks.html
5 http://snap.stanford.edu/data/loc-Gowalla.html
6 http://snap.stanford.edu/data/loc-Brightkite.html



(a) Query Input of mCK (b) Query Input of mCE (c) Query Answer of mCK (d) Query Answer of mCE

Fig. 7. Case study on Yelp dataset.

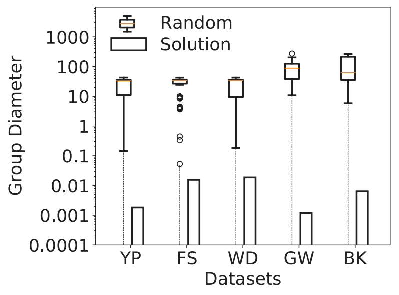


Fig. 8. Group diameter comparison results.

size by limiting the maximum number of lines to be read in. Following previous researches of the subgraph isomorphism [25,26], we generate query graphs by selecting subgraphs from the data graph to guarantee that at least one match exists for each query graph. For each query, we generate 20 different query graph sets to gather 20 running time statistics and finally take the average of them as the running time of the query. Besides, unless indicated otherwise, the fuzzy query graph is defined as the ego network of $q.e$, i.e., $F(q) = q[\{q.e\} \cup N(q.e)]$.

Algorithms. To the best of our knowledge, there are no existing works that investigate the mCE matching problem. In this paper, we implement and evaluate 2 baseline algorithms and 2 advanced algorithms as follows.

- *ExactEnum* - Hierarchical Search Algorithm in Section 3 using enumeration algorithm to conduce the m-closest search procedure, which gets all exact matched geo-entities first and enumerate all possible group among them to find the minimized diameter one.
- *ExactScan* - Hierarchical Search Algorithm in Section 3 using *EXACT* to conduce the m-closest search procedure, which gets all exact matched geo-entities first and conduct m-closest search by the state-of-the-art m-closest search algorithm [2] among them.
- *F2EScan* - A mCE matching algorithm that using the *Fuzzy-Exact* framework with the $Arc-Tree$ and the $Arc-Forest$ to obtain the answer group.
- *IF2EScan* - F2EScan algorithm with using BLOOM index to retrieve fuzzy matched geo-entities in the *exact to fuzzy phase*.

At the non-spatial layer, we use the state-of-the-art subgraph matching algorithm VC [26] to perform matching tasks uniformly. For ease of description, we name *Exact- Enum* and *ExactScan* as the *ExactFirst* algorithms, *F2E- Scan* and *IF2EScan* as the *FuzzyExact* algorithms.

Environment. We perform our experiments on a CentOS Linux server (Release 7.8.2003) with Quad-Core Intel Xeon(R) CPU (E3-1220 3.00 GHz) and 64G memory. All the algorithms are implemented in Java 8.

5.2. Case study

In this section, we give a case study on Yelp dataset to show the superiority of mCE over mCK in the scenario of itinerary travel planning. On a trip, a tourist wants to eat at an Italian restaurant, get a massage, buy some gifts for friends, and finally find a hotel to rest in, expecting all of these places to have a 5-star rating and accept credit cards. Other than that, the distance between these places should be as short as possible. As shown in Fig. 7(a) and 7(b), the query input of mCK is a set of independent keywords while mCE explicitly describes the characteristics of each query entity. Fig. 7(c) shows the query answer of mCK deviates far from the user expectation, as the hotel does not support credit card payment and the Italian restaurant just has a 3.5-star rating. This is because mCK just simply find a set of geo-entities that can cover the keywords without considering the correlation among them. On the contrary, as shown in Fig. 7(d), the query answer of mCE can completely satisfy the user demand because the query graphs can precisely represent the user expectation. It demonstrates that mCE is more feasible than mCK for scenarios where users want to explicitly specify the relationship between entity and attribute or between attribute and attribute.

5.3. Effectiveness evaluation

In the effectiveness evaluation experiment, we generate feasible groups by randomly picking satisfied geo-entities and using the diameters of them to compare with the optimal solution. As shown in Fig. 8, the randomly selected groups' diameters are represented by a box-plot where the orange line is the median value. The optimal solution achieved by our algorithm as the comparison is shown with histogram.

We can see from Fig. 8 that the optimal solution is much smaller than the diameter of random selection on every real dataset. Especially, as the optimal solution does not belong to any random selections, the optimal value is even smaller than the minimum of the box-plot. Here smaller diameter means the reduction of distance and expenditure under the same constraints, which is a benefit for planning or recommendation tasks. The comparison result verifies that our algorithm is effective for different domains.

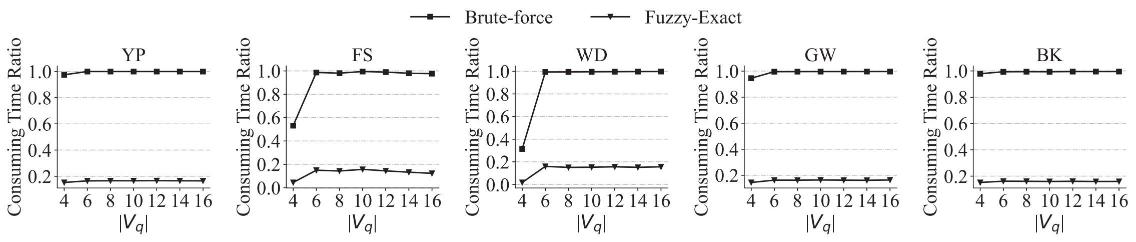


Fig. 9. Time consumption ratio at the non-spatial layer with varying $|V_q|$.

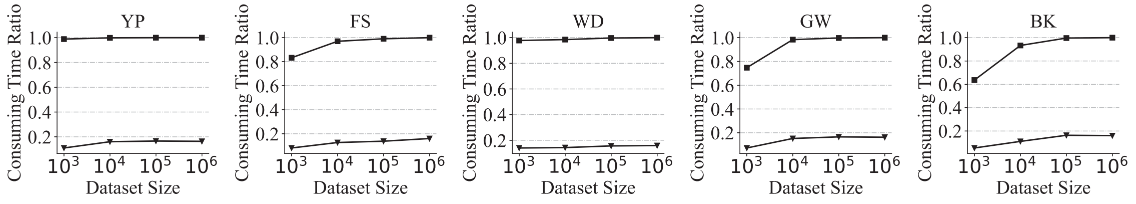


Fig. 10. Time consumption ratio at the non-spatial layer with varying dataset size.

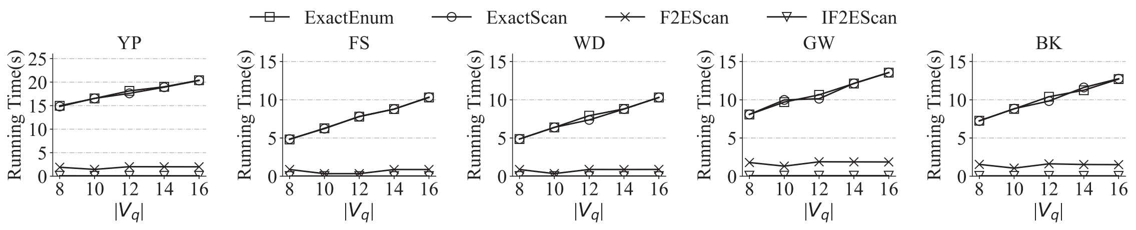


Fig. 11. Running time on real datasets with $|Q| = 3$ and varying $|V_q|$.

5.4. Effect of pruning abilities between two layers

To evaluate the cooperation performance of pruning strategies at two layers, the proposed framework is compared to the Brute-force approach, i.e., *ExactEnum* and *Fuzzy − Exact* framework (*F2EScan*), in terms of the time consumption ratio of non-spatial layer on five datasets. The number of vertex $|V_q|$ in query graph q varies from 4 to 16 and the size of data set varies from 10^3 to 10^6. The time consumption ratio is $\frac{t_1}{t1+t2}$, where t_1, t_2 represent the total time costs at the non-spatial layer and the spatial layer respectively.

When the number of vertex $|V_q|$ in query graph q varies, we can observe from Fig. 9, that (1) our framework consistently performs superior to the Brute-force approach on five datasets. And the time cost of the non-spatial layer in our *Fuzzy − Exact* framework only occupies approximate 20%, which means that the cooperation of pruning strategies at two layers is indeed effective and efficient. Meanwhile, our framework is robust to the $|V_q|$; (2) The time cost of the Brute-force approach at the non-spatial layer takes about 100% in 32 out of 35 cases owing to the exhaustive exact matching. The reason is that search space of the non-spatial layer increases exponentially with the increasing number of vertices in query graphs. Meanwhile, the changes of $|V_q|$ have small influences on results.

As the size of datasets varies from 10^3 to 10^6 on five datasets, we can observe similar result trends from Fig. 10. It also illustrates that the proposed unified framework based on the exploration and exploitation of two pruning strategies is beneficial for completing *m*CE matching and is much better than baselines in all cases. For the Brute-force approach, the effect of the pruning strategy at the spatial layer is negligible since the search space at the spatial layer depends on the number of eligible geo-entities, which is negatively related to the number of vertices in query graphs. Thus when data gradually expands, time costs are produced in the non-spatial layer, which can be effectively alleviated by our fuzzy mechanism.

In summary, our *Fuzzy − Exact* framework can save 80% expensive exact matching compared to the Brute-force approach on five datasets owing to the cooperation of pruning abilities between two layers. It directly leads to the significant performance improvement in time cost. Meanwhile, experimental results are also consistent with the theoretical analysis in Section 4. In the next subsection, we will show more detail about the performance of the framework.

5.5. Performance evaluation

In this set of experiments, we evaluate the impact of the number of nodes in query graph q (i.e., $|V_q|$), the number of query graphs (i.e., $|Q|$) and the dataset size. Fig. 11. shows the running time of 4 algorithms with fixed $|Q| = 3$, fixed dataset size = 10^6 and $|V_q|$ varied. Fig. 12 shows the running time with fixed $|Q| = 6$, fixed dataset size = 10^6 and $|V_q|$ varied. Fig. 13 shows the running time with fixed $|V_q| = 8$, fixed dataset size = 10^6 and $|Q|$ varied. Fig. 14 shows the running time with fixed $|V_q| = 16$, fixed dataset size = 10^6 and $|Q|$ varied. Next, we will analyze the performance of the 4 algorithms in several aspects.

Effect of $|V_q|$ **and** $|Q|$. Figs. 11–12 shows the running time with $|V_q|$ varied. All *ExactFirst* algorithms take a longer time with the increasing of $|V_q|$ because the growth of $|V_q|$ results in the increasing of the search breadth. All *FuzzyExact* algorithms consistently outperform the others, and the performance gap between *ExactFirst* algorithms and *FuzzyExact* algorithms becomes wider and wider with the $|V_q|$ increasing. Figs. 13–14. shows the experiment results with $|Q|$ varied. Compared with $|Q|$, $|V_q|$ does not have a significant impact for *FuzzyExact* algorithms on the running time.

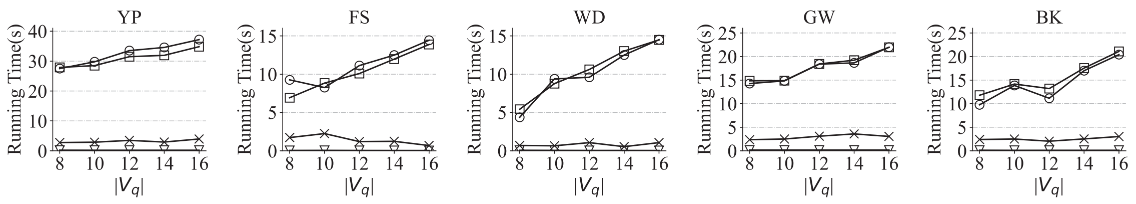


Fig. 12. Running time on real datasets with $|Q| = 6$ and varying $|V_q|$.

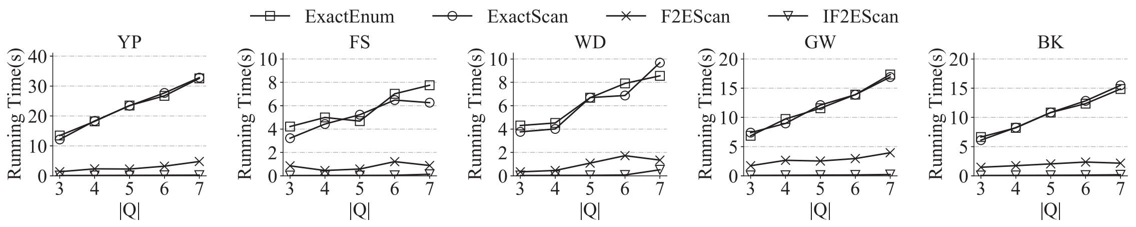


Fig. 13. Running time on the real datasets with $|V_q| = 8$ and varying $|Q|$.

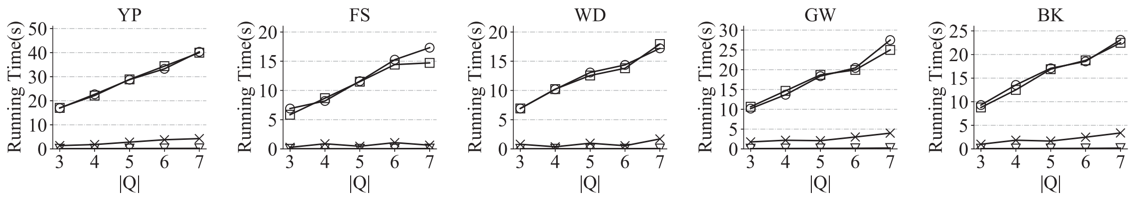


Fig. 14. Running time on the real datasets with $|V_q| = 16$ and varying $|Q|$.

For the running time aspect, *F2EScan* and *IF2EScan* always take less time than *ExactEnum* and *ExactScan*. *F2EScan* outperforms *ExactEnum* and *ExactScan* by up to one order of magnitude, and *IF2EScan* outperforms them by up to two orders of magnitude. The experiment results demonstrates the efficiency of our proposed *Fuzzy-Exact* framework. The improvement is put down to the framework takes fuzzy matched geo-entities as the intermediate search results to avoid checking whether each vertex of G is an exact matched geo-entity. That reduces the costly overhead at the non-spatial layer and makes the pruning ability of the algorithms at the spatial layer more effective.

In the above experiments, we can find that there is no significant performance gap between *ExactEnum* and *ExactScan*, which indicates that the performances between the enumeration algorithm and *EXACT* are almost identical at the spatial layer. This is probably because there are few candidates after filtering geo-entities by the *ExactFirst* algorithms at the non-spatial layer. As the result, the pruning ability of algorithms at the spatial layer becoming ineffective, it leads to the performance of *ExactScan* degrading to the same level as that of *Exact-Enum*.

In contrast, with the increasing of $|V_q|$ or $|Q|$, there is just a slight impact on the running time of *F2EScan*, and there is no notable impact for *IF2EScan*, which demonstrates the scalability and robustness of our proposed *Fuzzy-Exact* framework.

In summary, our techniques can work well on the Geo-HIN such that the geo-entities are distinct and spatially sparse. In few cases, the performance of our techniques may weaken. As we have discussed that the m-closest search is an NP-Hard problem, when geo-entities are very spatially dense, the relevant geo-entities in the sweeping areas will be numerous and the search space at the spatial layer will grow exponentially with its increase. Especially, when there are many similar geo-entities, it may arise more false positives, which increases the cooperation loops. Here, our techniques still perform superior to baselines.

Evaluation of fuzzy levels. Here, the fuzzy level is defined as the difference between fuzzy query graph and origin query graph. We explore the time overheads among different fuzzy levels, i.e., $q.e$ with its one-hop neighbors, ego network of $q.e$, and $q.e$ with its two-hop neighbors. We can observe from Fig. 15 that query time overhead does not necessarily increase or decrease as the fuzzy level increases. The principle of picking fuzzy query graph definition is representing as many geo-entity characteristics as possible with as few vertices as possible, and the experiment results show that ego network may be a good choice because it shows good effects in most cases.

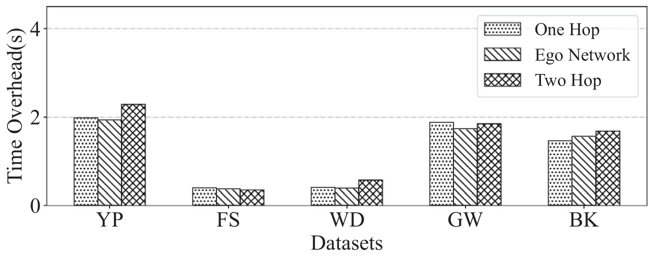


Fig. 15. Time overheads among different fuzzy levels.

Evaluation of the Arc-Tree and Arc-Forest. We evaluate the scalability and the efficiency of the $Arc-Tree$ & $Arc-Forest$ with numbers of intermediate search results and cooperation loops. We conduct a sequence of experiments on different volumes of synthetic intermediate search results in terms of time overhead. The number of intermediate searching results varies from 10^5 to 5×10^5 with fixed 10^3 cooperation loops. The cooperation loops vary from 10^3 to 5×10^3 with fixed 10^5 intermediate searching results. The data structure is used for comparison is a sorted list which can also eliminate the repetitive scanning. Fig. 16 shows the total time overheads of the cooperation loops between the $Arc - Tree$ & $Arc - Forest$ and a sorted list. Time overhead of the sorted list increases linearly as the number of

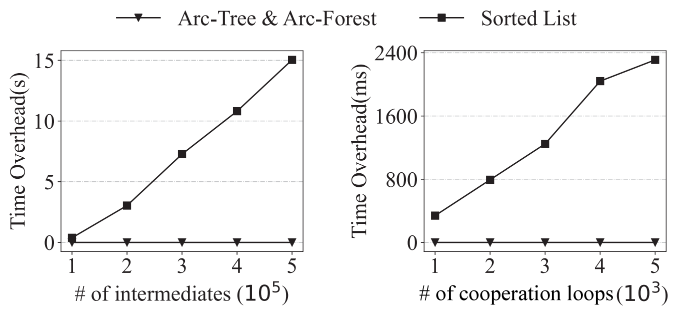


Fig. 16. Time overheads between *Arc* − *Tree* & *Arc* − *Forest* and sorted list with the number of intermediates varied (Left) and the number of cooperation loops (Right).

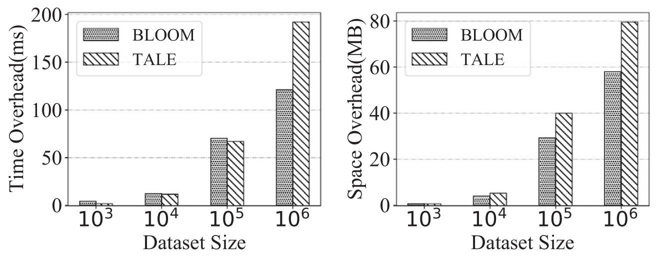


Fig. 17. Time and space overheads on BLOOM index and TALE index.

intermediate search results or the number of cooperation loops grows while the *Arc* − *Tree* & *Arc* − *Forest* keeps few overheads consistently. This is because that the sorted list need $O(N)$ time complexity to complete a round of cooperation loop because it need N operations to reorder the intermediate search results at worst case, where N is the number of intermediate search results. Though it can eliminate the repetitive scanning with respect to the out-of-order list, and the *Arc* − *Tree* & *Arc* − *Forest* just need $O(logN)$. The results illustrate that using *Arc* − *Tree* & *Arc* − *Forest* to maintain the intermediate search results can make the *Fuzzy* − *Exact* framework more scalable when the number of intermediates or cooperation loops is large.

Evaluation of the indexes. We also study the time and space overheads on different indexes, i.e., BLOOM Index (second level of the TALE Index) and TALE Index. The experiment results are shown in Fig. 17. We can find that when dataset size is greater than 10^5, TALE Index takes more time overhead than BLOOM Index, while space overhead of TALE Index is always greater than BLOOM Index. This indicates that even though TALE Index stores more structural information of the Geo-HIN, it does not always consume less time than BLOOM Index. And the reason is TALE Index takes extra time to scan B^+Tree, whereas it does not lead to better pruning ability with aspect to BLOOM Index.

6. Conclusion

In this paper, we proposed a novel *Fuzzy* − *Exact* framework to handle a new *m*-closest entity matching problem over geographic heterogeneous information networks, which is more practical and a challenge. Here geo-entities and non-geo-entities are processed by a unified framework, where exhaustive exact matching for each geo-entity is avoided through the fuzzy mechanism. At the same time, two adaptive data structures named *Arc* − *Tree* and *Arc* − *Forest* are introduced to maintain the intermediate search results which are exploited to enhance the search process between non-spatial and spatial layers. Pruning abilities at both the non-spatial and spatial layers are effectively cooperated to improve the query efficiency. Comprehensive experimental results prove the efficiency and scalability of the proposed framework.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my data and code at the Attach File step.

Acknowledgments

This work is supported by the National Key R&D Program of China under grant 2018AAA0102502.

References

[1] Y. Sun, J. Han, X. Yan, P.S. Yu, T. Wu, PathSim: Meta path-based top-K similarity search in heterogeneous information networks, Proc. VLDB Endow. 4 (11) (2011) 992–1003, http://dx.doi.org/10.14778/3402707.3402736.
[2] T. Guo, X. Cao, G. Cong, Efficient algorithms for answering the m-closest keywords query, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 405–418.
[3] D. Zhang, Y.M. Chee, A. Mondal, A.K. Tung, M. Kitsuregawa, Keyword search in spatial databases: Towards searching by document, in: 2009 IEEE 25th International Conference on Data Engineering, IEEE, 2009, pp. 688–699.
[4] D. Zhang, B.C. Ooi, A.K. Tung, Locating mapped resources in web 2.0, in: 2010 IEEE 26th International Conference on Data Engineering, ICDE 2010, IEEE, 2010, pp. 521–532.
[5] Z. Cai, G. Kalamatianos, G.J. Fakas, N. Mamoulis, D. Papadias, Diversified spatial keyword search on RDF data, VLDB J. 29 (5) (2020) 1171–1189, http://dx.doi.org/10.1007/s00778-020-00610-z.
[6] D. Zhang, Y. Li, X. Cao, J. Shao, H.T. Shen, Augmented keyword search on spatial entity databases, VLDB J. 27 (2) (2018) 225–244, http://dx.doi.org/10.1007/s00778-018-0497-6.
[7] G. Kalamatianos, G.J. Fakas, N. Mamoulis, Proportionality in spatial keyword search, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 885–897.
[8] A. Mahmood, W.G. Aref, Query processing techniques for big spatial-keyword data, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1777–1782.
[9] G. Cong, C.S. Jensen, Querying geo-textual data: Spatial keyword queries and beyond, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 2207–2212.
[10] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 2011, pp. 349–360.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[12] H. He, A.K. Singh, Closure-tree: An index structure for graph queries, in: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA, IEEE Computer Society, 2006, p. 38, http://dx.doi.org/10.1109/ICDE.2006.37.
[13] Y. Tian, J.M. Patel, Tale: A tool for approximate large graph matching, in: 2008 IEEE 24th International Conference on Data Engineering, IEEE, 2008, pp. 963–972.
[14] L. Liu, B. Du, H. Tong, et al., G-finder: Approximate attributed subgraph matching, in: 2019 IEEE International Conference on Big Data, Big Data, IEEE, 2019, pp. 513–522.
[15] S. Zhang, J. Yang, W. Jin, SAPPER: Subgraph indexing and approximate matching in large graphs, Proc. VLDB Endow. 3 (1) (2010) 1185–1194, http://dx.doi.org/10.14778/1920841.1920988, URL http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R105.pdf.
[16] D. Choi, J. Pei, X. Lin, Finding the minimum spatial keyword cover, in: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, IEEE Computer Society, 2016, pp. 685–696, http://dx.doi.org/10.1109/ICDE.2016.7498281.
[17] Y. Fang, K. Wang, X. Lin, W. Zhang, Cohesive subgraph search over big heterogeneous information networks: Applications, challenges, and solutions, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2829–2838.
[18] D. Seyler, P. Chandar, M. Davis, An information retrieval framework for contextual suggestion based on heterogeneous information network embeddings, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 953–956.

[19] S. Fan, C. Shi, X. Wang, Abnormal event detection via heterogeneous information network embedding, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 1483–1486.

[20] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, J. Ye, Heteta: Heterogeneous information network embedding for estimating time of arrival, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2444–2454.

[21] L. Zou, J. Mo, L. Chen, M.T. Özsu, D. Zhao, gStore: Answering SPARQL queries via subgraph matching, Proc. VLDB Endow. 4 (8) (2011) 482–493, http://dx.doi.org/10.14778/2002974.2002976, URL http://www.vldb.org/pvldb/vol4/p482-zou.pdf.

[22] H. He, A.K. Singh, Graphs-at-a-time: Query language and access methods for graph databases, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 405–418.

[23] W.-S. Han, J. Lee, J.-H. Lee, Turbo$_{\text{iso}}$: Towards ultrafast and robust subgraph isomorphism search in large graph databases, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 337–348.

[24] F. Bi, L. Chang, X. Lin, L. Qin, W. Zhang, Efficient subgraph matching by postponing cartesian products, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 1199–1214.

[25] B. Bhattarai, H. Liu, H.H. Huang, Ceci: Compact embedding cluster index for scalable subgraph matching, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1447–1462.

[26] S. Sun, Q. Luo, Subgraph matching with effective matching order and indexing, IEEE Trans. Knowl. Data Eng. (2020) 1, http://dx.doi.org/10.1109/TKDE.2020.2980257.

[27] S. Sun, Q. Luo, In-memory subgraph matching: An in-depth study, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 1083–1098.

[28] Y. Gao, X. Liu, J. Wu, T. Li, P. Wang, L. Chen, ClusterEA: Scalable entity alignment with stochastic training and normalized mini-batch similarities, 2022, arXiv preprint arXiv:2205.10312.

[29] C. Ge, X. Liu, L. Chen, B. Zheng, Y. Gao, LargeEA: Aligning entities for large-scale knowledge graphs, Proc. VLDB Endow. 15 (2) (2021) 237–245, http://dx.doi.org/10.14778/3489496.3489504, URL http://www.vldb.org/pvldb/vol15/p237-gao.pdf.

[30] C. Ge, X. Liu, L. Chen, B. Zheng, Y. Gao, Make it easy: An effective end-to-end entity alignment framework, in: SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 777–786, http://dx.doi.org/10.1145/3404835.3462870.

[31] A. Jain, S. Sarawagi, P. Sen, Deep indexed active learning for matching heterogeneous entity representations, Proc. VLDB Endow. 15 (1) (2021) 31–45, http://dx.doi.org/10.14778/3485450.3485455, URL http://www.vldb.org/pvldb/vol15/p31-jain.pdf.

[32] J. Wang, Y. Li, W. Hirota, E. Kandogan, Machop: An end-to-end generalized entity matching framework, in: R. Bordawekar, O. Shmueli, Y. Amsterdamer, D. Firmani, R. Marcus (Eds.), AiDM '22: Proceedings of the Fifth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, Philadelphia, Pennsylvania, USA, 17 June 2022, ACM, 2022, pp. 2:1–2:10, http://dx.doi.org/10.1145/3533702.3534910.

[33] N. Armenatzoglou, S. Papadopoulos, D. Papadias, A general framework for geo-social query processing, Proc. VLDB Endow. 6 (10) (2013) 913–924, http://dx.doi.org/10.14778/2536206.2536218, URL http://www.vldb.org/pvldb/vol6/p913-papadopoulos.pdf.

[34] Y. Fang, R. Cheng, X. Li, S. Luo, J. Hu, Effective community search over large spatial graphs, Proc. VLDB Endow. 10 (6) (2017) 709–720, http://dx.doi.org/10.14778/3055330.3055337, URL http://www.vldb.org/pvldb/vol10/p709-fang.pdf.

[35] J. Shi, N. Mamoulis, D. Wu, D.W. Cheung, Density-based place clustering in geo-social networks, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 99–110.

[36] Y. Li, R. Chen, J. Xu, Q. Huang, H. Hu, B. Choi, Geo-social k-cover group queries for collaborative spatial computing, in: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, IEEE Computer Society, 2016, pp. 1510–1511, http://dx.doi.org/10.1109/ICDE.2016.7498399.

[37] Y. Gao, X. Qin, B. Zheng, G. Chen, Efficient reverse top-k Boolean spatial keyword queries on road networks, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1205–1218, http://dx.doi.org/10.1109/TKDE.2014.2365820.

[38] Y. Gao, B. Zheng, G. Chen, W. Lee, K.C.K. Lee, Q. Li, Visible reverse k-nearest neighbor query processing in spatial databases, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1314–1327, http://dx.doi.org/10.1109/TKDE.2009.113.

[39] G. Chen, J. Zhao, Y. Gao, L. Chen, R. Chen, Time-aware Boolean spatial keyword queries, IEEE Trans. Knowl. Data Eng. 29 (11) (2017) 2601–2614, http://dx.doi.org/10.1109/TKDE.2017.2742956.

[40] X. Ren, J. Wang, Multi-query optimization for subgraph isomorphism search, Proc. VLDB Endow. 10 (3) (2016) 121–132, http://dx.doi.org/10.14778/3021924.3021929, URL http://www.vldb.org/pvldb/vol10/p121-ren.pdf.

[41] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426, http://dx.doi.org/10.1145/362686.362692.

[42] D. Yang, D. Zhang, Z. Yu, Z. Yu, Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs, in: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013, pp. 479–488.

[43] D. Vrandecic, M. Krötzsch, Wikidata: A free collaborative knowledgebase, Commun. ACM 57 (10) (2014) 78–85, http://dx.doi.org/10.1145/2629489.