Yin HB, Yang DH, Zhang KQ *et al.* Indoor uncertain semantic trajectory similarity join. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(6): 1441-1465 Nov. 2024. DOI: 10.1007/s11390-023-2418-4

Indoor Uncertain Semantic Trajectory Similarity Join

Hong-Bo Yin[1] (尹洪波), Dong-Hua Yang[1, 2] (杨东华), Kai-Qi Zhang[1] (张开旗), *Member, CCF*
Hong Gao[2, *] (高　宏), *Distinguished Member, CCF*, and Jian-Zhong Li[3] (李建中), *Fellow, CCF*

[1] *Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China*

[2] *Center of Analysis, Measurement and Computing, Harbin Institute of Technology, Harbin 150001, China*

[3] *Faculty of Computer Science and Control Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China*

E-mail: hongboyin@hit.edu.cn; yang.dh@hit.edu.cn; zhangkaiqi@hit.edu.cn; honggao@hit.edu.cn; lijzh@hit.edu.cn

Received April 14, 2022; accepted July 6, 2023.

Abstract With the widespread deployment of indoor positioning systems, an unprecedented scale of indoor trajectories is being produced. By considering the inherent uncertainties and the text information contained in such an indoor trajectory, a new definition named Indoor Uncertain Semantic Trajectory is defined in this paper. In this paper, we focus on a new primitive, yet quite essential query named Indoor Uncertain Semantic Trajectory Similarity Join (IUST-Join for short), which is to match all similar pairs of indoor uncertain semantic trajectories from two sets. IUST-Join targets a number of essential indoor applications. With these applications in mind, we provide a purposeful definition of an indoor uncertain semantic trajectory similarity metric named IUS. To process IUST-Join more efficiently, both an inverted index on indoor uncertain semantic trajectories named 3IST and the first acceleration strategy are proposed to form a filtering-and-verification framework, where most invalid pairs of indoor uncertain semantic trajectories are pruned at quite low computation cost. And based on this filtering-and-verification framework, we present a highly-efficient algorithm named Indoor Uncertain Semantic Trajectory Similarity Join Processing (USP for short). In addition, lots of novel and effective acceleration strategies are proposed and embedded in the USP algorithm. Thanks to these techniques, both the time complexity and the time overhead of the USP algorithm are further reduced. The results of extensive experiments demonstrate the superior performance of the proposed work.

Keywords filtering-and-verification framework, indoor uncertain semantic trajectory, inverted index, trajectory similarity join

1 Introduction

For people in different countries and continents, approximately 87% of our lives are spent in indoor space, such as shopping malls, airports, and office buildings, which is disclosed by multiple studies[1]. For example, there are over 100 million passengers boarding at Beijing Capital International Airport in 2018①. Thus, quite a few movements of individuals are contained in such indoor space. What is more, the past few years have witnessed the great breakthroughs in indoor position technologies[2] and the widespread deployment of indoor positioning systems. Driven by the above key factors, an unprecedented scale of indoor trajectories is being produced. Such indoor trajectories can serve as a foundation for a wide variety of indoor applications, which are expected to boom in the coming years②[3].

In indoor space, an indoor positioning system is commonly used to track the positions of indoor mov-

Regular Paper

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. U22A2025, U19A2059, 61732003, 61832003, U1811461, and 62102119, and the Key Research and Development Projects of the Ministry of Science and Technology of China under Grant No. 2019YFB2101902.

*Corresponding Author

①https://en.wikipedia.org/wiki/Beijing_Capital_International_Airport, Apr. 2024.

②https://www.abiresearch.com/press/over-800-million-smartphones-using-indoor-location/, Nov. 2024.

ing users to generate indoor trajectories[4]. The positions of an indoor moving user are captured by the indoor positioning system when it is in the detection ranges of static positioning devices (e.g. RFID sensors or bluetooth base stations) distributed in indoor space[5] as shown in Fig.1. It is a relatively simple example of a shopping mall with one floor, and this example is used for illustration throughout this paper. For an indoor moving user, a sequence of static positioning devices ordered by the timestamps is called the indoor trajectory generated by the indoor positioning system. For example, the generated indoor trajectory of user U_1 is $\chi_1 = [(p_{26}, t_1, t_{101}), (p_{21}, t_{257}, t_{260}), (p_{24}, t_{270}, t_{274}), (p_{23}, t_{338}, t_{341}), (p_{22}, t_{350}, t_{354}), (p_{22}, t_{482}, t_{485})]$. And each tuple (p, ts, te) in such an indoor trajectory records that U_1 was under the detection range of the static positioning device p in the closed time interval $[ts, te]$.

There are inherent uncertainties in indoor trajectories generated by using such an indoor positioning system. In general, because of the limited number of static positioning devices, the whole indoor space is unable to be entirely covered by detection ranges of all static positioning devices. Due to the discreteness of detection ranges, the positions of an indoor moving user between two successive tracking events are not avaliable[5]. And combined with the quite complex indoor topology, more than one potential indoor path may be inferred. Let us take the generated indoor trajectory χ_1 for example. In the open time interval (t_{274}, t_{338}), user U_1 might go through hallway H_{20} or room R_{23} from the detection range of static positioning device p_{24} to that of p_{23}. And in the open time interval (t_{354}, t_{482}), U_1 might be in hallway H_{20} or room R_{22}. Thus, we can infer $2 \times 2 = 4$ potential indoor paths of user U_1 from indoor trajectory χ_1.

As shown in multiple studies[6], knowledge extraction from mobility data can be significantly promoted by considering the semantic information contained in mobility data. From a semantic trajectory (i.e., a text-embedded trajectory), we can know not only where and when the indoor moving user has been, but also what this user has done. For example, one potential indoor path PT of indoor trajectory χ_1 passes rooms R_{21}, R_{22} and R_{23}, which are labeled with "Huawei", "Apple", and "Nike", respectively. From this, we can say that user U_1 might like consumer electronics and sportswear.

By combining the inherent uncertainties in indoor trajectories and semantic information in potential indoor paths, we present a new definition named Indoor Uncertain Semantic Trajectory. And given an indoor trajectory and the indoor topology, the corresponding indoor uncertain sematic trajectory encodes all potential indoor semantic paths of the indoor moving user, where each potential indoor semantic path is a sequence of rooms, doors, hallways, and staircases ordered by the corresponding timestamps. And each potential indoor semantic path is associated with the corresponding sematic information and a likelihood.

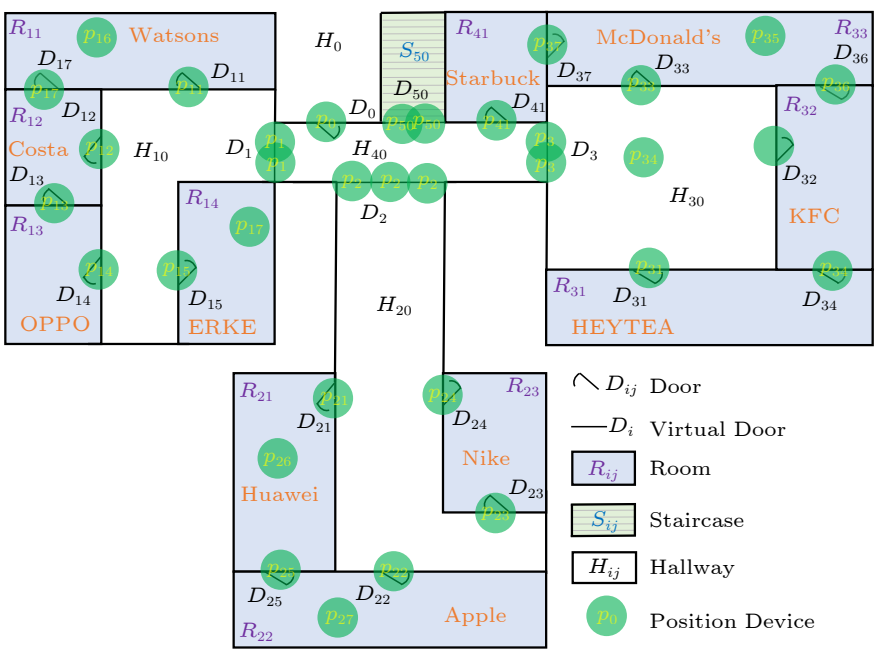

Fig.1. Example of indoor space and static positioning device deployment.

With the increasing use of GPS-enabled devices, massive outdoor trajectories are continuously growing at a high speed[7]. And nowadays, the analysis and the application of outdoor trajectories are attracting more and more people's attention, such as [8–10]. Trajectory similarity join on outdoor trajectories, which serves as a primitive, yet quite essential query, has been widely studied[11, 12]. In this paper, we study a new essential query named Indoor Uncertain Semantic Trajectory Similarity Join (IUST-Join for short). Given two sets of indoor uncertain semantic trajectories $\mathbb{P}$ and $\mathbb{Q}$, IUST-Join is to find all pairs of indoor uncertain semantic trajectories from $\mathbb{P}$ and $\mathbb{Q}$ with spatio-textual similarity exceeding the given similarity threshold θ. IUST-Join can serve as a quite essential query in a number of indoor application scenarios, such as friend recommendation based on similar shopping interests, keyword-aware indoor route recommendation, trajectory near-duplicate detection, geo-text data cleaning, and so on.

Though various trajectory similarity metrics[11, 13] have been proposed, there is no "best" trajectory similarity metric superior to all the others in terms of accuracy, which has been proved through plenty of experiments[14]. Each trajectory similarity metric has its advantages and disadvantages, and how to choose the most suitable one depends on the application scenarios. With the application scenarios of IUST-Join in mind, we provide a purposeful definition of an indoor uncertain semantic trajectory similarity metric named IUS, where both spatial proximity and semantic similarity are considered to determine how much these two indoor uncertain semantic trajectories are similar to each other. The indoor distance between two positions is defined as the shortest walking Euclidean distance in the indoor space by considering the indoor topology[15]. And for IUS, it is the first time that indoor distances are used to measure the spatial proximity between each pair of indoor uncertain semantic trajectories.

In the computation, IUST-Join is far more complicated than trajectory similarity join on outdoor trajectories. Since for each pair of indoor uncertain semantic trajectories, to get their spatial proximity, the spatial proximities between all pairs of potential paths all need to be calculated. What is more, semantic information of indoor uncertain semantic trajectories also needs to be considered in calculating the similarity between each pair of indoor uncertain semantic trajectories. In addition, because of the pretty high time complexity and time overhead, using a brute force approach to process IUST-Join is totally infeasible. To address this, we propose the USP algorithm with lots of novel and effective acceleration strategies embedded, which can reduce the time complexity and accelerate the processing of IUST-Join greatly.

To sum up, the main contributions of our work are listed as follows.

- By considering the inherent uncertainties and the text information contained in indoor trajectories, a new definition named Indoor Uncertain Semantic Trajectory is proposed.
- We are the first to define and formalize a primitive, yet quite essential similarity join query on indoor uncertain semantic trajectories, which is named IUST-Join. IUST-Join targets a number of essential indoor applications. And with these applications in mind, we provide a purposeful definition of the indoor uncertain semantic trajectory similarity metric named IUS.
- To accelerate the computation of IUST -Join, both an inverted index on indoor uncertain semantic trajectories named 3IST and the first acceleration strategy are proposed to form a filtering-and-verification framework. By using this framework, most invalid pairs of indoor uncertain semantic trajectories can be pruned at quite low computation cost.
- Based on this filtering -and-verification framework, we present the USP algorithm, a highly-efficient algorithm in processing IUST-Join. In addition, lots of novel and effective acceleration strategies are proposed and embedded in the USP algorithm. Thanks to these, both the time complexity and the time overhead of processing IUST-Join are reduced greatly.
- Extensive comparison experiments have been done, and the results demonstrate superior performance of our proposed methods. Besides, it is confirmed that by using our proposed USP algorithm, instead of the well-designed baseline TII algorithm, to process IUST-Join, at least 98.5% of the execution time can be easily saved.

The rest of this paper is organized as follows. Section 2 provides the formal definitions of the new essential query IUST-Join and some basic conceptions. To process IUST-Join, Section 3 introduces a baseline method named the TII algorithm. And the USP algorithm with lots of novel and effective acceleration strategies embedded is introduced detailedly in Section 4. Section 5 reports the experimental results and

analysis. Section 6 reviews related work, and Section 7 concludes this work.

2 Preliminaries and Problem Statement

In this section, how to map the indoor space to the extended accessibility base graph, a mainstream method showing the complex indoor topology, is introduced first. Then, the concept of the indoor uncertain sematic trajectory is defined. After that, we propose an indoor uncertain semantic trajectory similarity metric named IUS. Last, based on IUS, a new essential query named IUST-Join is formally defined.

Table 1 lists some key notations used throughout this paper.

Table 1. Key Notations

Symbol	Meaning
$\mathbb{IP}$	Set of all indoor partitions
$\mathbb{R} \subseteq \mathbb{IP}$	Set of all rooms
$\mathbb{D}$	Set of all doors
$\mathbb{P}$	Set of all static positioning devices
$\mathbb{S}$	Set of all staircases
$\mathbb{IL}$	Set of all identity labels
$\mathbb{TL}$	Set of all thematic labels
$IP \in \mathbb{IP}$	An indoor partition
$D \in \mathbb{D}$	A door
$p \in \mathbb{P}$	A static positioning device
$R \in \mathbb{R}$	A room
$S \in \mathbb{S}$	A staircase
$H \in \mathbb{IP}$	A hallway
$[ts, te]$	Closed time interval from ts to te
(ts, te)	Open time interval from ts to te
ΔT	Sampling time interval
U	An indoor moving user
F	Total number of floors

2.1 Extended Accessibility Base Graph

Recently, mapping the complex indoor space to an extended accessibility base graph G_{acc} and using such a graph to process various queries in indoor space attract much more attention, and are adopted by most work[16, 17].

Some notations and functions to be used in the follows are introduced first. An indoor partition $IP \in \mathbb{IP}$ is used to indicate a hallway, a room or a staircase. And each indoor partition is numbered differently in Fig.1. Given a door $D_k \in \mathbb{D}$, the function $D_k.IP()$ returns a set $\{IP_{k_1} \in \mathbb{IP}, IP_{k_2} \in \mathbb{IP}\}$, where an indoor moving user can enter the indoor partition IP_{k_1} or IP_{k_2} through D_k. And given $IP_i \in \mathbb{IP}$, function $IP_i.D()$ gives a set of doors, through each of which an indoor moving user can directly enter IP_i. In other words, $IP_i.D() = \{D_k \in \mathbb{D} | IP_i \in D_k.IP()\}$. In Fig.1, $D_{24}.IP() = \{H_{20}, R_{23}\}$ and $H_{20}.D() = \{D_2, D_{21}, D_{22}, D_{23}, D_{24}\}$.

The extended accessibility base graph is a connected and undirected graph $G_{\text{acc}} = (V, E, E2D, F_{D2D})$. A vertex $v_i \in V$ represents the corresponding indoor partition $IP_i \in \mathbb{IP}$. And there is an edge $e_{i,j} \in E$ between two vertexs v_i and $v_j \in V$ iff an indoor moving user can directly move between the indoor partitions IP_i and $IP_j \in \mathbb{IP}$ $(i \neq j)$ through each door in the set $E2D(e_{i,j}) = IP_i.D() \cap IP_i.D()$. Besides, G_{acc} can be easily changed to be a directed graph when some doors only permit the movements in one direction, such as the security check points in an airport. Fig.2 shows the corresponding extended accessibility base graph G_{acc} of the indoor space shown in Fig.1. The vertexs v_{40}, v_{41} and v_{50} represent the hallway H_{40}, the room R_{41}, and the staircase S_{50}, respectively. The edge $e_{20,23}$ between v_{20} and v_{23} denotes that an indoor moving user can directly move between the hallway H_{20} and the room R_{23} through the door D_{23} or D_{24}.

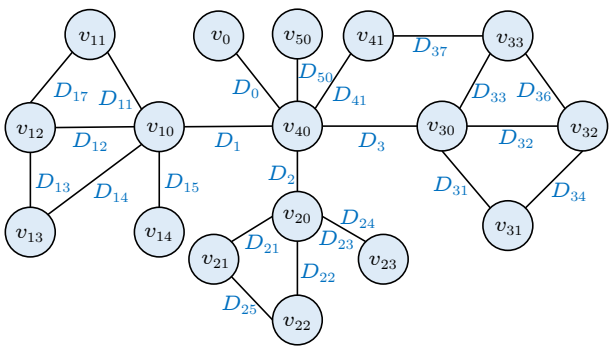

Fig.2. Extended accessibility base graph G_{acc}.

The above introduces that the indoor topology information is integrated in G_{acc} by using V, E and $E2D$. Besides, to support distance-aware indoor queries (e.g. nearest neighbor queries or shortest distance queries), some necessary indoor distances F_{D2D} also need to be integrated into G_{acc}. $\forall D_i, D_j \in \mathbb{D}(i \neq j)$, and the intra-partition door-to-door distance between D_i and D_j is calculated according to the following two cases:

$$F_{D2D}(D_i, D_j, IP_k) = \begin{cases} |D_i, D_j|_{IP_k}, \\ \quad \text{if } IP_k \in (D_i.IP() \cap D_j.IP()), \\ \infty, \text{otherwise}, \end{cases}$$

where $IP_k \in (D_i.IP() \cap D_j.IP())$ means that D_i and D_j are the different doors of the same indoor parti-

tion IP_k. And $|D_i, D_j|_{IP_k}$ is the indoor distance between doors D_i and D_j within the indoor partition IP_k, where the indoor distance is the shortest walking Euclidean distance in the indoor space. Thanks to the existence of F_{D2D}, G_{acc} is able to support distance-aware queries. For example, when we want to get the mininum walking distance from the source door D_s to the destination door D_t, we can keep all unvisited doors in a priority queue, and expand the search from D_s on the graph G_{acc} in a way similar to the classical Dijkstra Algorithm[18].

2.2 Indoor Uncertain Sematic Trajectory

As illustrated in Section 1, the positions of an indoor moving user are captured by the indoor positioning system when it enters the detection ranges of static positioning devices. $p_k.DR()$ denotes the detection range of a static positioning device $p_k \in \mathbb{P}$. In Fig.1, the detection range of each static positioning device is in green. Based on the positions in the indoor space, static positioning devices can be distinguished into two categories, i.e., partitioning devices and presence devices. For all partitioning devices, their detection ranges altogether cover all entrances and exits of indoor partitions, which means that an indoor moving user cannot be undetected by any partitioning device when it moves into another indoor partition. For partitioning devices whose detection ranges altogether cover a door or a virtual door, they are all set to a same notation. For instance, three partitioning devices are all named p_2 in Fig.1. For a partitioning device p_i, $p_i.D()$ denotes the corresponding door or virtual door which $p_i.DR()$ covers. Different from partitioning devices, presence devices simply serve to record the presence of some indoor moving user in an indoor partition. Given a presence device p_j, $p_j.IP()$ denotes the indoor partition where p_j is. In Fig.1, p_2, p_{21}, and p_{50} are all partitioning devices. $p_2.D() = D_2$ and $p_{24}.D() = D_{24}$. And p_{16}, p_{26}, and p_{35} are called presence devices. $p_{16}.IP() = R_{11}$ and $p_{26}.IP() = H_{20}$. It is assumed that all static positioning devices detect and report the positions of moving users simultaneously and periodically at a relatively high sampling rate (i.e., the sampling time interval is quite short).

Typically, a raw detection reading of static positioning devices is a triple (U, p, t), which means that indoor moving user U was under the detection range of static positioning device p at timestamp t. In the process of generating indoor trajectories, all such raw detection readings are converted into multiple tracking events in the form of (U, p, ts, te). Such a tracking event means that indoor moving user U is under the detection range of static positioning device p in closed time interval $[ts, te]$. All such tracking events are stored in the user tracking table (UTT). For all the tracking events of indoor moving user U_i in UTT, in the corresponding closed time intervals, U_i is defined to be in the detected state; otherwise, U_i is in the undetected state. The sequence of all tracking events in UTT with the same user U_i ordered by the timestamps is called the indoor trajectory $\chi_i = [(p_{i_1}, ts_1, te_1), (p_{i_2}, ts_2, te_2), \ldots, (p_{i_n}, ts_n, te_n)]$. Table 2 shows an example of UTT. User U_1' indoor trajectory is $\chi_1 = [(p_{26}, t_1, t_{101}), (p_{21}, t_{257}, t_{260}), (p_{24}, t_{270}, t_{274}), (p_{23}, t_{338}, t_{341}), (p_{22}, t_{350}, t_{354}), (p_{22}, t_{482}, t_{485})]$. In closed time intervals $[t_1, t_{101}]$, $[t_{257}, t_{260}]$, $[t_{270}, t_{274}]$, $[t_{338}, t_{341}]$, $[t_{350}, t_{354}]$, and $[t_{482}, t_{485}]$, U_1 is in the detected state; otherwise, U_1 is in the undetected state.

Table 2. User Tracking Table

U	p	ts	te
U_1	p_{26}	t_1	t_{101}
U_1	p_{21}	t_{257}	t_{260}
U_1	p_{24}	t_{270}	t_{274}
U_1	p_{23}	t_{338}	t_{341}
U_1	p_{22}	t_{350}	t_{354}
U_1	p_{22}	t_{482}	t_{485}
⋮	⋮	⋮	⋮

As the analysis shown in Section 1, there are inherent uncertainties in indoor trajectories generated by using such an indoor positioning system. Besides, from a semantic trajectory (i.e., text-embedded trajectory), we can know not only where and when the indoor moving user has been, but also what he/she has done. By combining the inherent uncertainties in indoor trajectories and semantic information in potential indoor paths, we present a new definition named Indoor Uncertain Semantic Trajectory, which is formally defined as follows.

Definition 1. (Indoor Uncertain Sematic Trajectory τ_i). *Given an indoor trajectory χ_i of the indoor moving user U_i and the extended accessibility base graph G_{acc}, the corresponding indoor uncertain sematic trajectory of χ_i is $\tau_i = \{P_{i_k} | k = 1, 2, 3, \ldots\}$, which encodes all potential indoor semantic paths of U_i. As a potential indoor semantic path, $P_{i_k} = (PT, SM, L)$ consists of a potential indoor path PT and the corresponding semantic information SM with the corre-*

sponding likelihood L.

In Definition 1, each potential indoor path $P_{i_k}.PT = [(\kappa_1,\ [ts_1,\ te_1]),\ (\kappa_{1.5},\ (te_1,\ ts_2)),\ (\kappa_2,\ [ts_2,\ te_2]),\ \ldots,\ (\kappa_n, [ts_n, te_n])]$ is a sequence of rooms, doors, hallways, and staircases ordered by the corresponding timestamps, where $\kappa_j \in (\mathbb{IP} \cup \mathbb{D})$ and $\kappa_{j+0.5} \in \mathbb{IP}$. $(\kappa_j,\ ts_j,\ te_j)$ means that U_i is in the detected state, and U_i is in the indoor partition or near the door κ_j in the closed time interval $[ts_j,\ te_j]$. And $(\kappa_{j+0.5},\ te_j,\ ts_{j+1})$ means that U_i is in the undetected state, and U_i is in the indoor partition $\kappa_{j+0.5}$ in the open time interval $(te_j,\ ts_{j+1})$. Table 3 shows the corresponding indoor uncertain sematic trajectory $\tau_1 = \{P_{1_k} | k = 1,\ 2,\ 3,\ 4\}$ of the indoor trajectory χ_1. The indoor moving user U_1 is in the undetected state in $(t_{274},\ t_{338})$. And for the potential indoor path $P_{1_1}.PT$, U_1 is in the room R_{23} in $(t_{274},\ t_{338})$. In $[t_{270},\ t_{274}]$, U_1 is in the detected state, and U_1 is sure to be near the door D_{24}.

To introduce the corresponding semantic information $P_k.SM$ of a potential indoor path $P_{i_k}.PT$, the sematic information contained in indoor partitions is defined first. There are two types of sematic labels associated with each room, i.e., identity label (IL) and thematic label (TL). Each room relates to only one IL, and each IL is associated with a set of TLs. IL refers to the specific name of a room, and TL is a tag relevant to a given IL. $R_m.IL()$ denotes the corresponding IL of the room R_m, and $IL.TLs()$ returns the corresponding TL set of the given IL. For example, in Fig.1 and Fig.3, $R_{22}.IL() =$ Apple, a famous consumer electronics company. And Apple.$TLs() =$ {laptop, smartphone, smartwatch, ...}. Since the sematic information about passesd rooms can reflect the interests of U_i, all ILs and TLs with the corresponding lengthes of the time intervals about rooms, where the user U_i has been, are stored in $P_{i_k}.SM$. $P_{i_k}.SM$ consists of AIL and ATL, which are formally defined as follows.

$$P_{i_k}.SM.AIL = \{(IL_x \in \mathbb{IL}, ILT_x) | ILT_x = \sum_{(\kappa_j,\ ts_j,\ te_j) \in P_{i_k}.PT} (te_j - ts_j) \times IL_x.Match(\kappa_j) + \sum_{(\kappa_{j+0.5},\ te_j,\ ts_{j+1}) \in P_{i_k}.PT} (ts_{j+1} - te_j) \times IL_x.Match(\kappa_{j+0.5})\},$$

$$P_{i_k}.SM.ATL = \{(TL_y \in \mathbb{TL}, TLT_y) | TLT_y = \sum_{(IL_x,\ ILT_x) \in P_{i_k}.SM.AIL} \frac{ILT_x \times TL_y.Map(IL_x)}{IL_x.TLs().size()}\},$$

where if $Input \in \mathbb{R}$ and $Input.IL() = IL_x$, $IL_x.Match(Input) = 1$; otherwise, $IL_x.Match(Input) = 0$. $IL_x.TLs().size()$is to get the number of distinct TLs in the TL set $IL_x.TLs()$. And if $TL_y \in IL_x.TLs()$, $TL_y.Map(IL_x) = 1$; otherwise, $TL_y.Map(IL_x) = 0$. For the potential indoor semantic path P_{1_1} shown in Table 3, $P_{1_1}.SM.AIL = \{(R_{21}.IL(), t_{257} - t_1), (R_{23}.IL(), t_{338} - t_{274}), (R_{22}.IL(), t_{482} - t_{354})\} = \{$(Huawei, $256\Delta T$), (Nike, $64\Delta T$), (Apple, $128\Delta T$)$\}$, where ΔT is the sampling time interval of the indoor position system. Owing to the limitation of the scope, $SM.ATL$ is omitted in Table 3.

The corresponding likelihood of $P_{i_k}.PT$ is $P_{i_k}.L = l(\kappa_{1.5},\ te_1,\ ts_2) \times l(\kappa_{2.5},\ te_2,\ ts_3) \times \ldots \times l(\kappa_{n-0.5},\ te_{n-1},\ ts_n)$. The function $l(\kappa_{j+0.5},\ te_j,\ ts_{j+1})$ returns the likelihood that the user is in the indoor partition $\kappa_{j+0.5}$ in the open time interval $(te_j,\ ts_{j+1})$.

Table 3. Indoor Uncertain Sematic Trajectory τ_1

Time Interval	$P_{1_1}.PT$	$P_{1_2}.PT$	$P_{1_3}.PT$	$P_{1_4}.PT$
$[t_1,\ t_{257})$	R_{21}	R_{21}	R_{21}	R_{21}
$[t_{257},\ t_{260}]$	D_{21}	D_{21}	D_{21}	D_{21}
$(t_{260},\ t_{270})$	H_{20}	H_{20}	H_{20}	H_{20}
$[t_{270},\ t_{274}]$	D_{24}	D_{24}	D_{24}	D_{24}
$(t_{274},\ t_{338})$	R_{23}	R_{23}	H_{20}	H_{20}
$[t_{338},\ t_{341}]$	D_{23}	D_{23}	D_{23}	D_{23}
$(t_{341},\ t_{350})$	H_{20}	H_{20}	H_{20}	H_{20}
$[t_{350},\ t_{354}]$	D_{22}	D_{22}	D_{22}	D_{22}
$(t_{354},\ t_{482})$	R_{22}	H_{20}	R_{22}	H_{20}
$[t_{482},\ t_{485}]$	D_{22}	D_{22}	D_{22}	D_{22}
L	$l(R_{23}, t_{274}, t_{338}) \times l(R_{22}, t_{354}, t_{482})$	$l(R_{23}, t_{274}, t_{338}) \times l(H_{20}, t_{354}, t_{482})$	$l(H_{20}, t_{274}, t_{338}) \times l(R_{22}, t_{354}, t_{482})$	$l(H_{20}, t_{274}, t_{338}) \times l(H_{20}, t_{354}, t_{482})$
$SM.AIL$	{(Huawei, $256\Delta T$), (Nike, $64\Delta T$), (Apple, $128\Delta T$)}	{(Huawei, $256\Delta T$), (Nike, $64\Delta T$)}	{(Huawei, $256\Delta T$), (Apple, $128\Delta T$)}	{(Huawei, $256\Delta T$)}

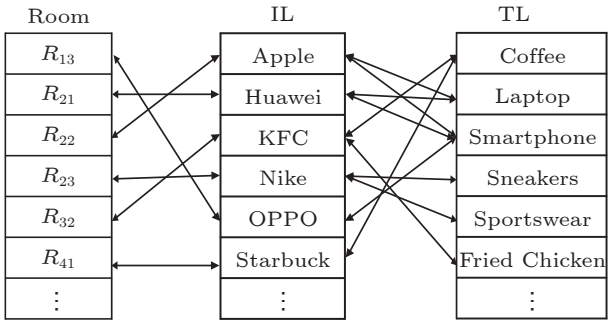

Fig.3. Labels mappings of rooms.

2.3 Indoor Uncertain Semantic Trajectory Similarity Metric IUS

As introduced in Section 1, IUST-Join can serve as a quite essential query in a number of indoor application scenarios, such as friend recommendation based on similar shopping interests, keyword-aware indoor route recommendation, trajectory near-duplicate detection, geo-text data cleaning, and so on. And with these applications in mind, we provide a purposeful definition of an indoor uncertain semantic trajectory similarity metric named IUS for IUST-Join. Both spatial proximity and semantic similarity need to be considered in IUS to determine how much two given indoor uncertain semantic trajectories are similar to each other. Before we define IUS, some needed definitions are formally defined first as follows.

Given a potential indoor path $P_{i_k}.PT = [(\kappa_1, [ts_1, te_1]), (\kappa_{1.5}, (te_1, ts_2)), (\kappa_2, [ts_2, te_2]), \ldots, (\kappa_n, [ts_n, te_n])]$ with $(2n-1)$ triples. $P_{i_k}.PT[f].T()$ returns the length of $P_{i_k}.PT[f]$'s corresponding time interval, i.e.,

$$\forall x \in [1, n], \quad P_{i_k}.PT[2x].T() = ts_{x+1} - te_x,$$
$$P_{i_k}.PT[2x-1].T() = te_x - ts_x.$$

$ST()$ is to get the sum length of all time intervals, and is formally defined as:

$$P_{i_k}.PT.ST() = \sum_{f=1}^{2n-1} P_{i_k}.PT[f].T().$$

$P_{i_k}.rest().PT$ denotes $P_{i_k}.PT$ without the last triple (κ_n, ts_n, te_n), i.e.,

$$P_{i_k}.rest().PT = [(\kappa_1, [ts_1, te_1]), (\kappa_{1.5}, (te_1, ts_2)), \ldots, (\kappa_{n-0.5}, (te_{n-1}, ts_n))].$$

For the potential indoor path $P_{1_1}.PT = [(R_{21}, [t_1, t_{257}]), (D_{21}, (t_{257}, t_{260})), \ldots, (R_{22}, (t_{354}, t_{482})), (D_{22}, [t_{482}, t_{485}])]$ shown in Table 3, $P_{1_1}.PT[1].T() = t_{257} - t_1 = 256\Delta T$ and $P_{1_1}.PT[2].T() = t_{260} - t_{257} = 3\Delta T$. $P_{1_1}.PT.ST() = (t_{257} - t_1) + (t_{260} - t_{257}) + \ldots + (t_{485} - t_{482}) = 484\Delta T$. And $P_{1_1}.rest().PT = [(R_{21}, [t_1, t_{257}]), (D_{21}, (t_{257}, t_{260})), \ldots, (R_{22}, (t_{354}, t_{482}))]$.

The door-to-door indoor distance $d(D_x, D_y)$ is the shortest walking Euclidean distance between two doors D_x and D_y in the indoor space by considering the indoor topology. And based on $d(D_x, D_y)$, $iD(P_{i_k}.PT[a], P_{j_l}.PT[b])$, the indoor distance between $P_{i_k}.PT[a]$ and $P_{j_l}.PT[b]$, is formally defined as:

$$iD(P_{i_k}.PT[a], P_{j_l}.PT[b]) = \begin{cases} d(P_{i_k}.PT[a], P_{j_l}.PT[b]), \\ \quad \text{if } P_{i_k}.PT[a] \in \mathbb{D} \text{ and } P_{j_l}.PT[b] \in \mathbb{D} \\ \min\left\{ \begin{array}{l} d(P_{i_k}.PT[a], D_y)| \\ D_y \in P_{j_l}.PT[b].D() \end{array} \right\}, \\ \quad \text{if } P_{i_k}.PT[a] \in \mathbb{D} \text{ and } P_{j_l}.PT[b] \in \mathbb{IP} \\ \min\left\{ \begin{array}{l} d(D_x, P_{j_l}.PT[b])| \\ D_x \in P_{i_k}.PT[a].D() \end{array} \right\}, \\ \quad \text{if } P_{i_k}.PT[a] \in \mathbb{IP} \text{ and } P_{j_l}.PT[b] \in \mathbb{D} \\ \min\left\{ \begin{array}{l} d(D_x, D_y)|D_x \in P_{j_l}.PT[b].D() \wedge \\ D_y \in P_{j_l}.PT[b].D() \end{array} \right\}, \\ \quad \text{if } P_{i_k}.PT[a] \in \mathbb{IP} \text{ and } P_{j_l}.PT[b] \in \mathbb{IP} \end{cases}.$$

In Fig.1, $iD(R_{22}, R_{23}) = iD(D_{22}, R_{23}) = \min\{ d(D_{22}, D_{23}), d(D_{22}, D_{24})\} = d(D_{22}, D_{23})$. $MaxDist$, a constant, is the maximum door-to-door indoor distance in the given indoor space, i.e.,

$$MaxDist = \max\{d(D_x, D_y)|D_x, D_y \in \mathbb{D}\}.$$

Then based on the above definitions, we propose $\Theta(P_{i_k}, P_{j_l})$, which measures the spatial dissimilarity between two potential indoor semantic paths P_{i_k} and P_{j_l} in indoor distances. It is formally defined as follows.

Definition 2. (Spatial Dissimilarity Metric $\Theta(P_{i_k}, P_{j_l})$). *Given two potential indoor semantic paths P_{i_k} and P_{j_l}, and the numbers of triples in $P_{i_k}.PT$ and $P_{j_l}.PT$ are M and N, respectively.*

$$\Theta(P_{i_k}, P_{j_l}) = \begin{cases} P_{j_l}.PT.ST(), \text{ if } P_{i_k}.PT.ST() = 0 \\ P_{i_k}.PT.ST(), \text{ if } P_{j_l}.PT.ST() = 0 \\ \min\left\{ \begin{array}{l} sub(P_{i_k}.PT[M], P_{j_l}.PT[N]) + \\ \Theta(P_{i_k}.rest(), P_{j_l}.rest()), \\ P_{i_k}.PT[M].T() + \\ \Theta(P_{i_k}.rest(), P_{j_l}), \\ P_{j_l}.PT[N].T() + \\ \Theta(P_{i_k}, P_{j_l}.rest()) \end{array} \right\}, \text{ otherwise,} \end{cases}.$$

$$sub(P_{i_k}.PT[M], P_{j_l}.PT[N]) = |P_{i_k}.PT[M].T() - P_{j_l}.PT[N].T()| + \frac{iD(P_{i_k}.PT[M], P_{j_l}.PT[N])}{MaxDist} \times \min\{P_{i_k}.PT[M].T(), P_{j_l}.PT[N].T()\}.$$

To measure the semantic similarity between two potential indoor semantic paths, not only ILs, but also TLs are considered. Since there may be also great similarity between rooms with different ILs, such as "Huawei" and "Apple". $\Phi()$, measuring the semantic similarity between two potential indoor semantic paths, is formally defined as follows.

Definition 3. (Semantic Similarity Metric $\Phi(P_{i_k}, P_{j_l})$). *Given two potential indoor semantic paths P_{i_k} and P_{j_l},*

$$\Phi(P_{i_k}, P_{j_l}) = \beta \times \frac{\sum\limits_{IL_{x_1} \in \mathbb{IL}} \min\{P_{i_k}.SM.ILT(IL_{x_1}), P_{j_l}.SM.ILT(IL_{x_1})\}}{\sum\limits_{IL_{x_2} \in \mathbb{IL}} \max\{P_{i_k}.SM.ILT(IL_{x_2}), P_{j_l}.SM.ILT(IL_{x_2})\}} + (1-\beta) \times \frac{\sum\limits_{TL_{y_1} \in \mathbb{TL}} \min\{P_{i_k}.SM.TLT(TL_{y_1}), P_{j_l}.SM.TLT(TL_{y_1})\}}{\sum\limits_{TL_{y_2} \in \mathbb{TL}} \max\{P_{i_k}.SM.TLT(TL_{y_2}), P_{j_l}.SM.TLT(TL_{y_2})\}}.$$

The parameter $\beta \in [0,1]$ is to balance the weight between the similarities of AILs and ATLs. $P_{i_k}.SM.ILT(IL_x)$ (resp. $P_{i_k}.SM.TLT(TL_y)$) returns the corresponding time interval length of IL_x (resp. TL_y) from $P_{i_k}.SM.AIL$ (resp. $P_{i_k}.SM.ATL$).

There are some key properties of $\Theta(P_{i_k}, P_{j_l})$ and $\Phi(P_{i_k}, P_{j_l})$ listed as follows:

- $\Theta(P_{i_k}, P_{j_l}) = \Theta(P_{j_l}, P_{i_k})$ and $\Phi(P_{i_k}, P_{j_l}) = \Phi(P_{j_l}, P_{i_k})$;
- $0 \leqslant \Theta(P_{i_k}, P_{j_l}) \leqslant (P_{i_k}.PT.ST() + P_{j_l}.PT.ST())$. The more similar $P_{i_k}.PT$ and $P_{j_l}.PT$ in indoor distances, the smaller $\Theta(P_{i_k}, P_{j_l})$. And $\Theta(P_{i_k}, P_{j_l}) = 0$ iff $P_{i_k}.PT$ and $P_{j_l}.PT$ are identical;
- $0 \leqslant \Phi(P_{i_k}, P_{j_l}) \leqslant 1$. And as P_{i_k} and P_{j_l} are more similar in semantics, $\Phi(P_{i_k}, P_{j_l})$ becomes larger.

For two given indoor uncertain semantic trajectories, our proposed indoor uncertain semantic trajectory similarity metric IUS considers both spatial proximity and semantic similarity of all pairs of potential indoor semantic paths to determine how much these two indoor uncertain semantic trajectories are similar to each other. Based on the definition of Θ and Φ, the metric IUS is formally defined as follows.

Definition 4. (Indoor Uncertain Semantic Trajectory Similarity Metric $IUS(\tau_i, \tau_j)$). *Given a pair of indoor uncertain sematic trajectories τ_i and τ_j,*

$$IUS(\tau_i, \tau_j) = \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L \times (\alpha \times \Theta'(P_{i_k}.PT, P_{j_l}.PT) + (1-\alpha) \times \Phi(P_{i_k}, P_{j_l}))).$$

The parameter $\alpha \in [0,1]$ is to balance the weight between spatial proximity and semantic similarity of two potential indoor semantic paths. $\Theta'(P_{i_k}.PT, P_{j_l}.PT) = (1 - (\Theta(P_{i_k}, P_{j_l})/(P_{i_k}.PT.ST() + P_{j_l}.PT.ST())))$ is to normalize $\Theta(P_{i_k}, P_{j_l})$ into $[0, 1]$.

Since $\Theta(P_{i_k}, P_{j_l}) = \Theta(P_{j_l}, P_{i_k})$ and $\Phi(P_{i_k}, P_{j_l}) = \Phi(P_{j_l}, P_{i_k})$, we can see that $IUS(\tau_i, \tau_j) = IUS(\tau_j, \tau_i) \in [0,1]$. And the more similar τ_i and τ_j, the greater $IUS(\tau_i, \tau_j)$.

2.4 Problem Definition of IUST-Join

For an indoor uncertain semantic trajectory τ_i, all key TLs are stored in the key TL set $\tau_i.KTL()$, which is formally defined as follows.

Definition 5. (Key TL Set $\tau_i.KTL()$). *Given an indoor uncertain semantic trajectory τ_i and a threshold μ,*

$$\tau_i.KTL() = \{TL_y \in \mathbb{IL} | \sum_{P_{i_k} \in \tau_i} (P_{i_k}.L \times P_{i_k}.SM.ATL.TLT(TL_y)) \geqslant \mu\}.$$

If two indoor uncertain semantic trajectories are considered to be similar, a quite reasonable condition is that they share at least one common key TL. This condition is to guarantee that there is some minimum sematic similarity between each pair of indoor uncertain semantic trajectories in the similarity join result. And such similar conditions have been widely applied in many studies, such as [19, 20]. Given two sets of indoor uncertain semantic trajectories $\mathbb{P}$ and $\mathbb{Q}$, IUST-Join is to find all such pairs $(\tau_i \in \mathbb{P}, \tau_j \in \mathbb{Q})$ that satisfy the following two conditions: 1) τ_i and τ_j share at least one common key TL; 2) the indoor uncertain semantic trajectory similarity between τ_i and τ_j is no less than a given threshold θ. IUST-Join is formally defined as follows.

Definition 6. (Indoor Uncertain Semantic Trajectory Similarity Join $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$). *Given two thresholds $\theta \in [0,1]$ and μ, two sets of indoor uncertain semantic trajectories $\mathbb{P}$ and $\mathbb{Q}$, $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ returns*

$$\{(\tau_i, \tau_j) \in \mathbb{P} \times \mathbb{Q} | \tau_i.KTL() \cap \tau_j.KTL() \neq \varnothing, IUS(\tau_i, \tau_j) \geqslant \theta\}.$$

3 Baseline the TII Algorithm

In this section, we first propose an inverted index

named 3IST, which is to prune many invalid pairs of indoor uncertain semantic trajectories at quite low computation cost in the process of IUST-Join. Then, based on 3IST, we propose a baseline algorithm named Traversing the Inverted Index (TII for short) to process IUST-Join, which follows a filtering-and-verification framework. Last, in order to reduce the time cost and the time complexity of the TII algorithm, two additional effective acceleration strategies are also presented and embedded in the TII algorithm.

3.1 Inverted Index 3IST

In Definition 6, only if there is at least one common key TL shared by a pair of indoor uncertain semantic trajectories $(\tau_i, \tau_j) \in \mathbb{P} \times \mathbb{Q}$, might (τ_i, τ_j) be returned by $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$. And thus, in order to filter out many invalid pairs of indoor uncertain semantic trajectories from the candidate result set $\mathbb{P} \times \mathbb{Q}$, we can build an inverted index of all indoor uncertain semantic trajectories (3IST for short) on each key TL $TL_y \in \mathbb{IL}$ and search on such an inverted index.

The inverted index 3IST consists of inverted sets on all key TLs. To build 3IST, we first go through each indoor uncertain semantic trajectory in $\mathbb{P}$ and $\mathbb{Q}$ to get the corresponding key TL set. Then for each key TL $TL_y \in \mathbb{IL}$, we build two inverted sets $TL_y.3IST(\mathbb{P})$ and $TL_y.3IST(\mathbb{Q})$ on $\mathbb{P}$ and $\mathbb{Q}$, respectively, which are formally defined as:

$$TL_y.3IST(\mathbb{P}) = \{\tau_i \in \mathbb{P} | TL_y \in \tau_i.KTL()\},$$
$$TL_y.3IST(\mathbb{Q}) = \{\tau_j \in \mathbb{Q} | TL_y \in \tau_j.KTL()\}.$$

3.2 TII Algorithm

In this subsection, we present a straightforward baseline approach to answer IUST-Join, named TII algorithm. And the pseudo code of the TII algorithm is shown in Algorithm 1. *ResultSet* is first initialized to an empty set (line 1). Then we go through each key TL in the inverted index 3IST (lines 2–10). And for each pair of indoor uncertain semantic trajectories $(\tau_i, \tau_j) \in \mathbb{P} \times \mathbb{Q}$ sharing at least one common key TL, we need to check whether the degree of the similarity $IUS(\tau_i, \tau_j)$ is no less than the given threshold θ (lines 3–9). Finally, by following this processing procedure, we can get the result set of $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$.

Algorithm 1. TII Algorithm

Input: two thresholds θ and μ, two sets of indoor uncertain semantic trajectories $\mathbb{P}$ and $\mathbb{Q}$

Output: the result set of $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$

```
1: ResultSet ← ∅
2: for TL_y ∈ 𝕀𝕃 do
3:     for τ_i ∈ TL_y.3IST(ℙ) do
4:         for τ_j ∈ TL_y.3IST(ℚ) do
5:             if IUS(τ_i, τ_j) ⩾ θ then
6:                 ResultSet.append({τ_i, τ_j})
7:             end if
8:         end for
9:     end for
10: end for
    return ResultSet
```

3.3 Two Efficient Acceleration Strategies Embedded in Algorithm TII

Some notations to be used are firstly introduced here. Suppose that $|\mathbb{D}|$, $|\mathbb{IL}|$, $|\mathbb{TL}|$, $|\mathbb{P}|$, and $|\mathbb{Q}|$ are the sizes of sets $\mathbb{D}$, $\mathbb{IL}$, $\mathbb{IL}$, $\mathbb{P}$, and $\mathbb{Q}$, respectively. $|\tau|_{\max}$ and $|PT|_{\max}$ denote the maximal number of potential indoor semantic paths in indoor uncertain semantic trajectories and the maximal number of tuples in potential indoor paths of indoor uncertain semantic trajectories, respectively. In indoor space, the layout of each floor is usually similar. F denotes the total number of floors. And the average number of doors in each floor is denoted by $D_{\text{avg}} = |\mathbb{D}|/F$.

For the TII algorithm, though the time overhead is reduced greatly by using the inverted index 3IST, the time overhead of computing even a single $\Theta(P_{i_k}, P_{j_l})$ is still completely unacceptable. According to Definition 2, for each pair of potential indoor semantic paths P_{i_k} and P_{j_l}, up to $O(|PT|_{\max}^{|PT|_{\max}})$ times door-to-door indoor distances need to be calculated in the calculation of $\Theta(P_{i_k}, P_{j_l})$. And for each door-to-door indoor distance from the source door D_s to the destination door D_t, all unvisited doors are kept in a priority queue, and the search from D_s on the graph G_{acc} is expanded in a way similar to the classical Dijkstra Algorithm. Thus, the time complexity of computing each door-to-door indoor distance is $O(|\mathbb{D}|^2)$. And thus, the time complexity of computing $\Theta(P_{i_k}, P_{j_l})$ is high up to $O(|PT|_{\max}^{|PT|_{\max}} \times |\mathbb{D}|^2)$, which is completely unacceptable during use.

First, we find that there are lots of repeated calculations of door-to-door indoor distances in a single calculation of $\Theta(P_{i_k}, P_{j_l})$. To avoid this, we adopt a dynamic programming solution, where a $(|P_{i_k}.PT|+1) \times (|P_{j_l}.PT|+1)$ matrix is used in each calculation of $\Theta(P_{i_k}, P_{j_l})$ to store the intermediate results. Due to

this, the needed calculation times of door-to-door indoor distances are reduced greatly from $O(|PT|_{\max}{}^{|PT|_{\max}})$ to $O(|PT|_{\max}{}^2)$.

Second, the results of extensive experiments done by us show that getting door-to-door indoor distances by using the classical Dijkstra Algorithm takes up over 99% of the total execution time in the calculation of $\Theta(P_{i_k}, P_{j_l})$. As a quite natural idea to address this, we present the first pre-calculation method, which is to make all door-to-door indoor distances pre-calculated. In the pre-processing stage, a method similar to the Floryd algorithm is used to search on the extended accessibility base graph G_{acc}, and all door-to-door indoor distances can be gotten in $O(|\mathbb{D}|^3) = O(D_{\text{avg}}{}^3 \times F^3)$. A $|\mathbb{D}| \times |\mathbb{D}| = D_{\text{avg}}{}^2 \times F^2$ matrix named $\boldsymbol{M}_1$ is used to record the calculated door-to-door indoor distances. $\boldsymbol{M}_1$ is kept in the main memory. And by searching $\boldsymbol{M}_1$, we can get any door-to-door indoor distance in constant time.

Unless otherwise noted, these two additional effective acceleration strategies are embedded in the TII algorithm.

3.4 Complexity Analysis of Algorithm TII

Thanks to these two additional effective acceleration strategies embedded in the TII algorithm presented in Subsection 3.3, the time complexity of the TII algorithm computing $\Theta(P_{i_k}, P_{j_l})$ is reduced from $O(|PT|_{\max}{}^{|PT|_{\max}} \times |\mathbb{D}|^2)$ to $O(|PT|_{\max}{}^2)$. For $\Phi(P_{i_k}, P_{j_l})$ defined in Definition 3, for each element IL_x in $\mathbb{IL}$, we need to search at most $|\mathbb{IL}|$ times to get $P_{i_k}.SM.ILT(IL)$ and $P_{j_l}.SM.ILT(IL)$. And for each element TL_x in $\mathbb{IL}$, we need to search at most $|\mathbb{TL}|$ times to get $P_{i_k}.SM.TLT(TL)$ and $P_{j_l}.SM.TLT(TL)$. And thus, the time complexity of computing $\Phi(P_{i_k}, P_{j_l})$ is $O(|\mathbb{IL}|^2 + |\mathbb{TL}|^2)$. Thus for the TII algorithm, by embedding these two effective acceleration strategies, the time complexity of computing $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ is reduced greatly from $O(|\mathbb{P}| \times |\mathbb{Q}| \times |\tau|_{\max}{}^2 \times (|PT|_{\max}{}^{|PT|_{\max}} \times |\mathbb{D}|^2 + |\mathbb{IL}|^2 + |\mathbb{TL}|^2))$ to $O(|\mathbb{P}| \times |\mathbb{Q}| \times |\tau|_{\max}{}^2 \times (|PT|_{\max}{}^2 + |\mathbb{IL}|^2 + |\mathbb{TL}|^2))$.

4 Algorithm USP

For the TII algorithm, though both the time complexity and the time overhead are reduced by using the inverted index 3IST and two additional acceleration strategies, there are still some limitations to be addressed.

- For the first pre-calculation method, though any door-to-door indoor distance can be obtained in constant time, the time overhead $O(D_{\text{avg}}{}^3 \times F^3)$ of getting all door-to-door indoor distances in the pre-processing stage is a little high and the space overhead $O(D_{\text{avg}}{}^2 \times F^2)$ to store the matrix $\boldsymbol{M}_1$ in the main memory is also a little high for large and complex indoor space. And both the time overhead and the space overhead could become much higher when F or D_{avg} gets larger (e.g., the shopping mall is with more floors or with more doors in each floor).
- In the TII algorithm, for any two potential indoor semantic paths P_{i_k} and P_{j_l}, the time complexity of computing the semantic similarity $\Phi(P_{i_k}, P_{j_l})$ is high up to $O(|\mathbb{IL}|^2 + |\mathbb{TL}|^2)$, which is pretty high. And the time overhead becomes much higher as the numbers of distinct ILs and TLs contained in potential indoor semantic paths increase.
- The analysis in Subsection 3.4 shows that by using the TII algorithm, the time complexity of computing $\Theta()$ is reduced from $O(|PT|_{\max}{}^{|PT|_{\max}} \times |\mathbb{D}|^2)$ to $O(|PT|_{\max}{}^2)$. However, the experimental results in Subsection 5.2.1 show that the calculations of $\Theta()$ functions take up over 99.9% of the total execution time of the TII algorithm. This means that the time overhead of computing $\Theta()$ functions is still quite high, and need to be further improved.

In order to overcome these limitations, we propose an algorithm named USP, which consists of many efficient and effective acceleration strategies. The USP algorithm, which is shown in Algorithm 2, follows a filtering-and-verification framework and prunes most invalid pairs of indoor uncertain semantic trajectories at quite low computation cost. The inverted index 3IST is still applied first in USP to filter out invalid indoor uncertain semantic trajectory pairs that do not share any key TLs (lines 4–5). The indoor uncertain semantic trajectory similarity metric $IUS(\tau_i, \tau_j)$ has been formally defined in Definition 4. And based on this, we can rewrite $IUS(\tau_i, \tau_j)$ into (lines 13, 15, and 16):

$$IUS(\tau_i, \tau_j) = \alpha \times Part_1(\tau_i, \tau_j) + (1 - \alpha) \times Part_2(\tau_i, \tau_j),$$

where $Part_1(\tau_i, \tau_j) = \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L \times \Theta'(P_{i_k}.PT, P_{j_l}.PT))$ and $Part_2(\tau_i, \tau_j) = \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L \times \Phi(P_{i_k}, P_{j_l}))$.

More details about Algorithm 2 will be introduced in the followings.

Algorithm 2. USP Algorithm

Input: two sets of indoor uncertain semantic trajectories $\mathbb{P}$ and $\mathbb{Q}$, and two thresholds $\theta \in [0,1]$ and μ defined in Definition 6

Output: the result set of $IUST\text{-}Join(\mathbb{P},\mathbb{Q})$

```
ResultSet ← ∅
VisitedState ← NewMatrix(ℙ, ℚ)
for TL_y ∈ 𝕃𝕃 do
    for τ_i ∈ TL_y.3IST(ℙ) do
        for τ_j ∈ TL_y.3IST(ℚ) do
            if VisitedState[i][j] ≠ 0 then
                continue
            else
                VisitedState[i][j] ← 1
            end if
            upart_1 ← UPart_1(τ_i, τ_j)
            if α × upart_1 + (1 − α) ⩾ θ then
                part_2 ← Part_2(τ_i, τ_j)
                if α × upart_1 + (1 − α) × part_2 ⩾ θ then
                    part_1 ← Part_1(τ_i, τ_j)
                    if α × part_1 + (1 − α) × part_2 ⩾ θ then
                        ResultSet.append((τ_i, τ_j))
                    end if
                end if
            end if
        end for
    end for
end for
return ResultSet
```

4.1 The Second Pre-Calculation Method

As shown in the first limitation in Section 4, under the condition that any door-to-door indoor distance can be still gotten in constant time, to further reduce the time overhead of obtaining the matrix $\boldsymbol{M}_1$ and the occupied main memory to store the matrix $\boldsymbol{M}_1$, we present the second pre-calculation method.

For the second pre-calculation method, in the pre-processing stage, an array set M_2, instead of $\boldsymbol{M}_1$, is created and stored in the main memory. $\forall z \in \{1, 2, \ldots, F\}$, and $\boldsymbol{M}_2[z]$ is a matrix, whose size is $O({D_{\text{avg}}}^2)$. $\boldsymbol{M}_2[z]$ stores all door-to-door indoor distances, where doors are all on the z-th floor. A method similar to the Floryd algorithm can be used to search on part of the extended accessibility base graph G_{acc}, and $\boldsymbol{M}_2[z]$ can be gotten in $O({D_{\text{avg}}}^3)$. And thus, the time overhead of getting the array M_2 in the pre-processing stage is only $O({D_{\text{avg}}}^3 \times F)$, not $O({D_{\text{avg}}}^3 \times F^3)$ for $\boldsymbol{M}_1$, and the total space overhead of M_2 in the main memory is only $O({D_{\text{avg}}}^2 \times F)$, instead of $O({D_{\text{avg}}}^2 \times F^2)$ for $\boldsymbol{M}_1$.

In the indoor space, the layouts of staircases on different floors are usually the same. Then $\forall D_a, D_b \in \mathbb{D}$, and the door-to-door indoor distance from the door D_a to the door D_b can be gotten by using the array M_2 as shown below:

$$M_1[a,b] = M_2.d(D_a, D_b) = \begin{cases} M_2[D_a.f][a,b], & \text{if } D_a.f = D_b.f, \\ \min \left\{ \begin{matrix} M_2[D_a.f].D2S(D_a, S_c) + \\ S_c.d(D_a.f, D_b.f) + \\ M_2[D_b.f].S2D(S_c, D_b) | S_c \in \mathbb{S} \end{matrix} \right\}, & \text{otherwise.} \end{cases}$$

$D_a.f$ denotes which floor the door D_a is on. $\mathbb{S}$ is the set of all staircases in the indoor space. $M_2[D_a.f].D2S(D_a, S_c)$ denotes the door-to-door indoor distance from the door D_a to the door of the staircases S_c on the $(D_a.f)$-th floor. Similarly, $M_2[D_b.f].S2D(S_c, D_b)$ is the door-to-door indoor distance from the door of the staircases S_c on the $(D_b.f)$-th floor to the door D_b. Obviously, both $M_2[D_a.f].D2S(D_a, S_c)$ and $M_2[D_b.f].S2D(S_c, D_b)$ can be directly gotten from the matrices $M_2[D_a.f]$ and $M_2[D_b.f]$ in short and constant time, respectively. $S_c.d(D_a.f, D_b.f)$ returns the indoor walking distance from the door of staircases S_c on the $(D_a.f)$-th floor to the door of staircases S_c on the $(D_b.f)$-th floor in staircase S_c. By pre-calculation, $S_c.d(D_a.f, D_b.f)$ can be also gotten in short and constant time. And thus, for the second pre-calculation method, with the help of the array M_2, any door-to-door indoor distance $M_2.d(D_a, D_b)$ can be gotten in $O(|\mathbb{S}|)$, where $|\mathbb{S}|$ is the number of all staircases in the indoor space. Since $|\mathbb{S}|$ is usually quite small and can be considered to be a constant, the time complexity of computing $M_2.d(D_a, D_b)$ can be also considered as $O(1)$. At the same time, the time cost and the occupied main memory about door-to-door indoor distances in the pre-processing stage are reduced greatly from $O({D_{\text{avg}}}^3 \times F^3)$ to $O({D_{\text{avg}}}^3 \times F)$ and from $O({D_{\text{avg}}}^2 \times F^2)$ to $O({D_{\text{avg}}}^2 \times F)$, respectively. Unless otherwise noted, the second pre-calculation method, instead of the first one, is used in the USP algorithm.

4.2 Semantic Information Encoding

As shown in the second limitation in Section 4, to further reduce the time overhead and the time complexity of computing the semantic similarity $\Phi(P_{i_k}, P_{j_l})$, we propose a specfic encoding style named Sematic Label Encoding (SL-encoding for short), which is a little similar to one-hot encoding. For the corresponding semantic information $P_{i_k}.SM$ of a potential indoor semantic path P_{i_k}, we use SL-encoding to encode each tuple in $SM.AIL$ and $SM.ATL$. An example of SL-encoding is shown in Fig.4. Each tuple

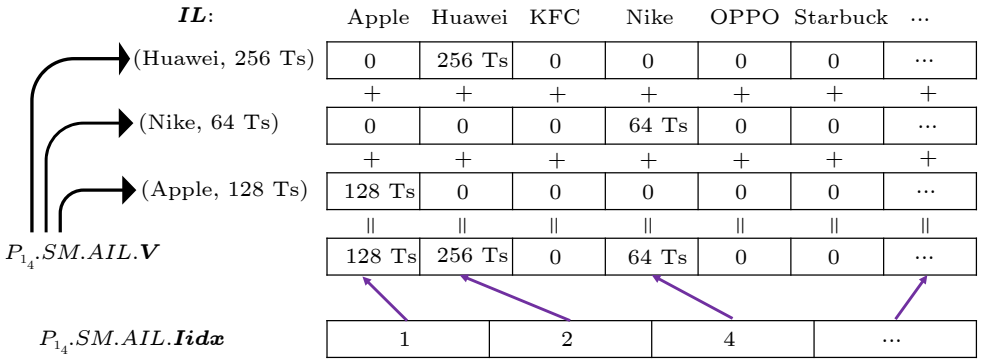

Fig.4. SL-encoding for $P_{1_1}.SM.AIL$ in Table 3. ΔT is the sampling time interval of the indoor position system.

$(\boldsymbol{IL}_x, \boldsymbol{ILT}_x) \in AIL$ is encoded as a vector $\boldsymbol{IL}_x.V$, whose size is $|\mathbb{IL}|$. And a vector $\boldsymbol{TL}_y.V$, whose size is $|\mathbb{TL}|$, is used to encode each tuple $(\boldsymbol{TL}_y, TLT_y) \in ATL$. The corresponding position of $\boldsymbol{IL}_x$ in $\boldsymbol{IL}_x.V$ is set to $\boldsymbol{ILT}_x$. The corresponding position of $\boldsymbol{TL}_y$ in $\boldsymbol{TL}_y.V$ is set to TLT_y. And other positions in $\boldsymbol{IL}_x.V$ and $\boldsymbol{TL}_y.V$ are all set to 0. Then AIL is encoded as a vector $\boldsymbol{AIL}.V$, whose size is $|\mathbb{IL}|$. And another vector $\boldsymbol{ATL}.V$, whose size is $|\mathbb{TL}|$, is used to encode ATL.

$$\forall u \in [1, |\mathbb{IL}|], AIL.V[u] = \sum_{(IL_x,\ ILT_x) \in AIL} IL_x.V[u],$$
$$\forall v \in [1, |\mathbb{TL}|], ATL.V[v] = \sum_{(TL_y,\ TLT_y) \in ATL} TL_y.V[v].$$

After AIL and ATL are encoded by SL-encoding, Definition 3 can be redefined as Definition 7.

Definition 7. (Semantic Similarity Metric $\Phi(P_{i_k}, P_{j_l})$). *Given a parameter $\beta \in [0,1]$ and two potential indoor semantic paths of different indoor uncertain sematic trajectories P_{i_k} and P_{j_l},*

$$\begin{aligned}
\Phi(P_{i_k}, P_{j_l}) &= \beta \times \\
&\frac{\sum_{IL_{x_1} \in \mathbb{IL}} \min\{P_{i_k}.SM.ILT(IL_{x_1}), P_{j_l}.SM.ILT(IL_{x_1})\}}{\sum_{IL_{x_2} \in \mathbb{IL}} \max\{P_{i_k}.SM.ILT(IL_{x_2}), P_{j_l}.SM.ILT(IL_{x_2})\}} + \\
&(1-\beta) \times \\
&\frac{\sum_{TL_{y_1} \in \mathbb{TL}} \min\{P_{i_k}.SM.TLT(TL_{y_1}), P_{j_l}.SM.TLT(TL_{y_1})\}}{\sum_{TL_{y_2} \in \mathbb{TL}} \max\{P_{i_k}.SM.TLT(TL_{y_2}), P_{j_l}.SM.TLT(TL_{y_2})\}} \\
&= \beta \times \\
&\frac{\sum_{u_1 \in [1, |\mathbb{IL}|]} \min\{P_{i_k}.SM.AIL.V[u_1], P_{j_l}.SM.AIL.V[u_1]\}}{\sum_{u_2 \in [1, |\mathbb{IL}|]} \max\{P_{i_k}.SM.AIL.V[u_2], P_{j_l}.SM.AIL.V[u_2]\}} + \\
&(1-\beta) \times \\
&\frac{\sum_{v_1 \in [1, |\mathbb{TL}|]} \min\{P_{i_k}.SM.ATL.V[v_1], P_{j_l}.SM.ATL.V[v_1]\}}{\sum_{v_2 \in [1, |\mathbb{TL}|]} \max\{P_{i_k}.SM.ATL.V[v_2], P_{j_l}.SM.ATL.V[v_2]\}}.
\end{aligned} \tag{1}$$

Parameter β is to balance the weight between the similarities of AILs and ATLs.

After AIL and ATL are encoded into $AIL.V$ and $ATL.V$ by SL-encoding, respectively, it is quite obvious that $P_{i_k}.SM.AIL$, $P_{i_k}.SM.ATL$, $P_{j_l}.SM.AIL$, and $P_{j_l}.SM.ATL$ all need to be scanned only once in the calculation of the semantic similarity $\Phi(P_{i_k}, P_{j_l})$. And thus, the time complexity of computing the semantic similarity $\Phi(P_{i_k}, P_{j_l})$ is reduced greatly from $O(|\mathbb{IL}|^2 + |\mathbb{TL}|^2)$ to $O(|\mathbb{IL}| + |\mathbb{TL}|)$. And because of these, the time complexity of the USP algorithm computing $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ can be reduced from $O((|PT|_{\max}^2 + |\mathbb{IL}|^2 + |\mathbb{TL}|^2) \times |\tau|_{\max}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$ to $O((|PT|_{\max}^2 + |\mathbb{IL}| + |\mathbb{TL}|) \times |\tau|_{\max}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$.

With consideration of that most positions in $AIL.V$ and $ATL.V$ are set 0, we propose two indexs named $\boldsymbol{Iidx}$ and $\boldsymbol{Tidx}$ to index $AIL.V$ and $ATL.V$, respectively. With the help of $\boldsymbol{Iidx}$ and $\boldsymbol{Tidx}$, the time complexity of computing the semantic similarity $\Phi(P_{i_k}, P_{j_l})$ can be further reduced. As an example shown in Fig.4, $\boldsymbol{Iidx}$ and $\boldsymbol{Tidx}$ are two vectors, each position of which stores a pointer. The i-th pointer in $\boldsymbol{Iidx}$ (resp. $\boldsymbol{Tidx}$) points to the i-th position in $AIL.V$ (resp. $ATL.V$). And the information stored by the i-th position in $AIL.V$ (resp. $ATL.V$) is not 0.

As shown in Algorithm 3, the PUI algorithm shows the process of calculating $\Phi(P_{i_k}, P_{j_l})$ in detail by using both $\boldsymbol{Iidx}$ and $\boldsymbol{Tidx}$. Some variables are firstly initialized (line 1), and we want to get $\Phi(P_{i_k}, P_{j_l}) = \beta \times (uSumI/dSumI) + (1-\beta) \times (uSumT/dSumT)$ finally. $uSumI$ and $dSumI$ can be gotten by following lines 2–24. And in a similar manner, $uSumT$ and $dSumT$ can be gotten by following lines 25–48. Each element of $P_{i_k}.SM.Iidx$ and $P_{j_l}.SM.Iidx$ is scanned one by one (lines 2–24). $Iidx.size()$ and $Tidx.size()$ in line 2, line 4, line 26 and line 28 are to get the sizes of the corresponding vectors $\boldsymbol{Iidx}$ and $\boldsymbol{Tidx}$, respectively. The functions $SumRestI()$ in line 21 and $SumRestT()$ in line 45 are respectively defined as:

Algorithm 3. PUI Algorithm

Input: two potential indoor semantic paths P_{i_k} and P_{j_l}, and the parameter $\beta \in [0,1]$ defined in Definition 3 and Definition 7

Output: $\Phi(P_{i_k}, P_{j_l})$, the semantic similarity between P_{i_k} and P_{j_l}

1: $Ii \leftarrow 1$, $Ij \leftarrow 1$, $uSumI \leftarrow 0$, $dSumI \leftarrow 0$
2: **while** $Ii \leqslant P_{i_k}.SM.Iidx.size()$ **do**
3: $IdxIi \leftarrow P_{i_k}.SM.Iidx[Ii]$
4: **while** $Ij \leqslant P_{j_l}.SM.Iidx.size()$ **do**
5: $IdxIj \leftarrow P_{j_l}.SM.Iidx[Ij]$
6: **if** $IdxIi > IdxIj$ **then**
7: $dSumI \leftarrow dSumI + P_{j_l}.SM.AIL.V[IdxIj]$
8: $Ij \leftarrow Ij + 1$
9: **else if** $IdxIi = IdxIj$ **then**
10: $uSumI \leftarrow uSumI + \min\{P_{i_k}.SM.AIL.V[IdxIi], P_{j_l}.SM.AIL.V[IdxIj]\}$
11: $dSumI \leftarrow dSumI + \max\{P_{i_k}.SM.AIL.V[IdxIi], P_{j_l}.SM.AIL.V[IdxIj]\}$
12: $Ii \leftarrow Ii + 1$, $Ij \leftarrow Ij + 1$
13: break
14: **else**
15: $dSumI \leftarrow dSumI + P_{i_k}.SM.AIL.V[IdxIi]$
16: $Ii \leftarrow Ii + 1$
17: break
18: **end if**
19: **end while**
20: **if** $Ij = P_{j_l}.SM.Iidx.size() + 1$ **then**
21: $dSumI \leftarrow dSumI + SumRestI(P_{i_k}.SM, Ii)$
22: **end if**
23: **end while**
24: $dSumI \leftarrow dSumI + SumRestI(P_{j_l}.SM, Ij)$
25: $Ti \leftarrow 1$, $Tj \leftarrow 1$, $uSumT \leftarrow 0$, $dSumT \leftarrow 0$
26: **while** $Ti \leqslant P_{i_k}.SM.Tidx.size()$ **do**
27: $IdxTi \leftarrow P_{i_k}.SM.Tidx[Ti]$
28: **while** $Tj \leqslant P_{j_l}.SM.Tidx.size()$ **do**
29: $IdxTj \leftarrow P_{j_l}.SM.Tidx[Tj]$
30: **if** $IdxTi > IdxTj$ **then**
31: $dSumT \leftarrow dSumT + P_{j_l}.SM.ATL.V[IdxTj]$
32: $Tj \leftarrow Tj + 1$
33: **else if** $IdxTi = IdxTj$ **then**
34: $uSumT \leftarrow uSumT + \min\{P_{i_k}.SM.ATL.V[IdxTi], P_{j_l}.SM.ATL.V[IdxTj]\}$
35: $dSumT \leftarrow dSumT + \max\{P_{i_k}.SM.ATL.V[IdxTi], P_{j_l}.SM.ATL.V[IdxTj]\}$
36: $Ti \leftarrow Ti + 1$, $Tj \leftarrow Tj + 1$
37: break
38: **else**
39: $dSumT \leftarrow dSumT + P_{i_k}.SM.ATL.V[IdxTi]$
40: $Ti \leftarrow Ti + 1$
41: break
42: **end if**
43: **end while**
44: **if** $Tj = P_{j_l}.SM.Tidx.size() + 1$ **then**
45: $dSumT \leftarrow dSumT + SumRestT(P_{i_k}.SM, Ti)$
46: **end if**
47: **end while**
48: $dSumT \leftarrow dSumT + SumRestT(P_{j_l}.SM, Tj)$
49: **return** $\beta \times (uSumI/dSumI) + (1-\beta) \times (uSumT/dSumT)$

$$SumRestI(SM, Ii) = \sum_{z=Ii}^{SM.Iidx.size()} SM.AIL.V[SM.Iidx[z]],$$

$$SumRestT(SM, Tj) = \sum_{z=Tj}^{SM.Tidx.size()} SM.ATL.V[SM.Tidx[z]].$$

$|Iidx|_{\max}$ and $|Tidx|_{\max}$ are used to denote the maximal sizes of vectors ***Iidx*** and ***Tidx***, respectively. Usually, $|Iidx|_{\max} << |\mathbb{IL}|$, and $|Tidx|_{\max} << |\mathbb{TL}|$. With the help of indexs ***Iidx*** and ***Tidx***, it is quite clear that $P_{i_k}.SM.Iidx$, $P_{j_l}.SM.Iidx$, $P_{i_k}.SM.Tidx$, and $P_{j_l}.SM.Tidx$ all need to be scanned only once by the PUI Algorithm in the calculation of semantic similarity $\Phi(P_{i_k}, P_{j_l})$. And thus, by using the PUI Algorithm, the time complexity of computing $\Phi(P_{i_k}, P_{j_l})$ is further reduced from $O(|\mathbb{IL}|^2 + |\mathbb{TL}|^2)$ to $O(|Iidx|_{\max} + |Tidx|_{\max})$. Thus, the time complexity of computing $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ can be also further reduced from $O((|PT|_{\max}{}^2 + |\mathbb{IL}|^2 + |\mathbb{TL}|^2) \times |\tau|_{\max}{}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$ to $O((|PT|_{\max}{}^2 + |Iidx|_{\max} + |Tidx|_{\max}) \times |\tau|_{\max}{}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$. Unless otherwise noted, the PUI Algorithm is embedded in line 13 of Algorithm 2 to accelerate the calculation of $\Phi(P_{i_k}, P_{j_l})$.

4.3 Three Acceleration Strategies

As shown in the third limitation in Section 4, the time overhead of computing the spatial proximity between a pair of given indoor uncertain sematic trajectories is still quite high. To address this, we present three acceleration strategies embedded in the USP algorithm, which is shown in Algorithm 2, to further reduce the time overhead.

The first acceleration strategy is combined with the inverted index 3IST to form a more effective filtering-and-verification framework. And by the USP algorithm following such a framework, more invalid indoor uncertain sematic trajectory pairs can be pruned at quite low computation cost.

VisitedState is a matrix whose size is $|\mathbb{P}| \times |\mathbb{Q}|$, and each position in ***VisitedState*** is initialized to 0 (line 2 of Algorithm 2). It is quite likely that $\exists(\tau_i, \tau_j) \in \mathbb{P} \times \mathbb{Q}$ and $|\tau_i.KTL() \cap \tau_j.KTL()| \geqslant 2$. And the existence of ***VisitedState*** is to prevent the repeated calculations of the indoor uncertain semantic trajectory similarity metric $IUS(\tau_i, \tau_j)$ in such a case (lines 6–10).

If there is an upper bound of of $IUS(\tau_i, \tau_j)$ less

than θ, a threshold in Definition 6, then the indoor uncertain sematic trajectory pair (τ_i, τ_j) can be directly pruned without the complicated calculation of $Part_1(\tau_i, \tau_j)$ (line 15). Since the time complexity of $Part_1(\tau_i, \tau_j)$ is still high up to $O(|PT|_{\max}^2 \times |\tau|_{\max}^2)$. In the follows, we will introduce how to derive the upper bounds of $IUS(\tau_i, \tau_j)$.

First, we can get an upper bound $UPart_1()$ of $Part_1()$ at quite low computation cost.

Theorem 1. *Given P_{i_k} and P_{j_l}, two potential indoor semantic paths of different indoor uncertain sematic trajectories,*

$$\Theta(P_{i_k}, P_{j_l}) \geqslant |P_{i_k}.PT.ST() - P_{j_l}.PT.ST()|.$$

Proof. From Definition 2, the definition of the spatial dissimilarity metric $\Theta(P_{i_k}, P_{j_l})$, we can see that each element pair of $P_{i_k}.PT$ and $P_{j_l}.PT$ is one-to-one matched by the function $sub()$. For the unmatched elements of $P_{i_k}.PT$ and $P_{j_l}.PT$, the sum of the corresponding result $T()$ is contained in the final result of $\Theta(P_{i_k}, P_{j_l})$. And for the matched pair $P_{i_k}.PT[M]$ and $P_{j_l}.PT[N]$, in the function $sub(P_{i_k}.PT[M], P_{j_l}.PT[N])$, the difference $|P_{i_k}.PT[M].T() - P_{j_l}.PT[N].T()|$ is also contained in the final result of $\Theta(P_{i_k}, P_{j_l})$. And thus, Theorem 1 can be easily gotten from these. □

Then, for the the spatial proximity metric $\Theta'()$ defined in Definition 4, we can get that

$$\begin{aligned}&\Theta'(P_{i_k}.PT, P_{j_l}.PT)\\ &= (1 - \frac{\Theta(P_{i_k}, P_{j_l})}{P_{i_k}.PT.ST() + P_{j_l}.PT.ST()})\\ &\leqslant \frac{2 \times \min\{P_{i_k}.PT.ST(), P_{j_l}.PT.ST()\}}{P_{i_k}.PT.ST() + P_{j_l}.PT.ST()}.\end{aligned}$$

Since $\forall P_a, P_b \in usT, P_a.PT.ST() = P_b.PT.ST()$ and $\sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L) = 1$, then we can get that (lines 15 and 13 of Algorithm 2)

$$\begin{aligned}part_1 &= Part_1(\tau_i, \tau_j)\\ &= \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L \times\\ &\quad \Theta'(P_{i_k}.PT, P_{j_l}.PT))\\ &\leqslant (\sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L)) \times\\ &\quad \frac{2 \times \min\{P_{i_1}.PT.ST(), P_{j_1}.PT.ST()\}}{P_{i_1}.PT.ST() + P_{j_1}.PT.ST()}\\ &= \frac{2 \times \min\{P_{i_1}.PT.ST(), P_{j_1}.PT.ST()\}}{P_{i_1}.PT.ST() + P_{j_1}.PT.ST()}\\ &= UPart_1(\tau_i, \tau_j) = upart_1.\end{aligned}$$

Second, an upper bound of $Part_2()$ is also needed. Obviously, $\forall P_{i_k}, P_{j_l}$, and $\Phi(P_{i_k}, P_{j_l}) \in [0, 1]$. Thus, we can also get that

$$\begin{aligned}Part_2(\tau_i, \tau_j) &= \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L \times \Phi(P_{i_k}, P_{j_l}))\\ &\leqslant 1 \times \sum_{P_{i_k} \in \tau_i} \sum_{P_{j_l} \in \tau_j} (P_{i_k}.L \times P_{j_l}.L) = 1.\end{aligned}$$

Last, for two indoor uncertain sematic trajectories τ_i and τ_j, the upper bounds of $Part_1()$ and $Part_2()$ are used to get two upper bounds of $IUS(\tau_i, \tau_j) = \alpha \times Part_1(\tau_i, \tau_j) + (1 - \alpha) \times Part_2(\tau_i, \tau_j)$. And in the USP algorithm shown in Algorithm 2, if a derived upper bound is less than θ, a threshold defined in Definition 6, the pair (τ_i, τ_j) can be directly pruned without the complicated calculation of $Part_1(\tau_i, \tau_j)$. The first upper bound of $IUS(\tau_i, \tau_j)$ is $\alpha \times upart_1 + (1 - \alpha)$ (line 12). By using the PUI algorithm shown in Algorithm 3 to computing $\Phi()$ functions, the time overhead of $Part_2(\tau_i, \tau_j)$, i.e., $O((|Iidx|_{\max} + |Tidx|_{\max}) \times |\tau|_{\max}^2)$, is much lower than that of $Part_1(\tau_i, \tau_j)$, i.e., $O(|PT|_{\max}^2 \times |\tau|_{\max}^2)$. And thus, if the condition in line 12 is not satisfied, we choose to calculate $Part_2(\tau_i, \tau_j)$ (line 13), instead of $Part_1(\tau_i, \tau_j)$ (line 15), to get the second upper bound of $IUS(\tau_i, \tau_j)$ is $\alpha \times upart_1 + (1 - \alpha) \times part_2$ (line 14). By using these two conditions in line 12 and line 14, many invalid indoor uncertain sematic trajectory pairs can be directly pruned at quite low computation cost without calculating the exact result of $Part_1(\tau_i, \tau_j)$.

If the complicated calculation of $Part_1(\tau_i, \tau_j)$ cannot be avoided (line 14), both the second and the third acceleration strategies are embedded in line 15 of the USP algorithm and used to accelerate the calculation.

The second acceleration strategy is to reuse part of the calculated matrix. To get the result of $Part_1(\tau_i, \tau_j)$, we need to get $\Theta(P_{i_k}, P_{j_l})$ ($\forall P_{i_k} \in \tau_i$ and $\forall P_{j_l} \in \tau_j$) first.

Fig.5(a) and Fig.5(b) show the matrices used in the dynamic programming to compute $\Theta(P_{1_1}, P_{j_l})$ and $\Theta(P_{1_2}, P_{j_l})$, respectively. And P_{1_1} and P_{1_2} are shown in Table 3. Take the matrix shown in Fig.5(a) for example. It is assumed that $|P_{1_1}.PT| = 0$ in the first column and $|P_{j_l}.PT| = 0$ in the first row. $\forall x \in \{1, 2, \ldots, |P_{1_1}.PT|\}$ and $\forall y \in \{1, 2, \ldots, |P_{j_l}.PT|\}$, and the label on the $(x+1)$-th column and the label on the $(y+1)$-th row are the x-th triple of $P_{1_1}.PT$ and the y-th triple of $P_{j_l}.PT$, respectively. And the time interval in each label is omitted for simplicity. The content of each position in the matrix is calculated by

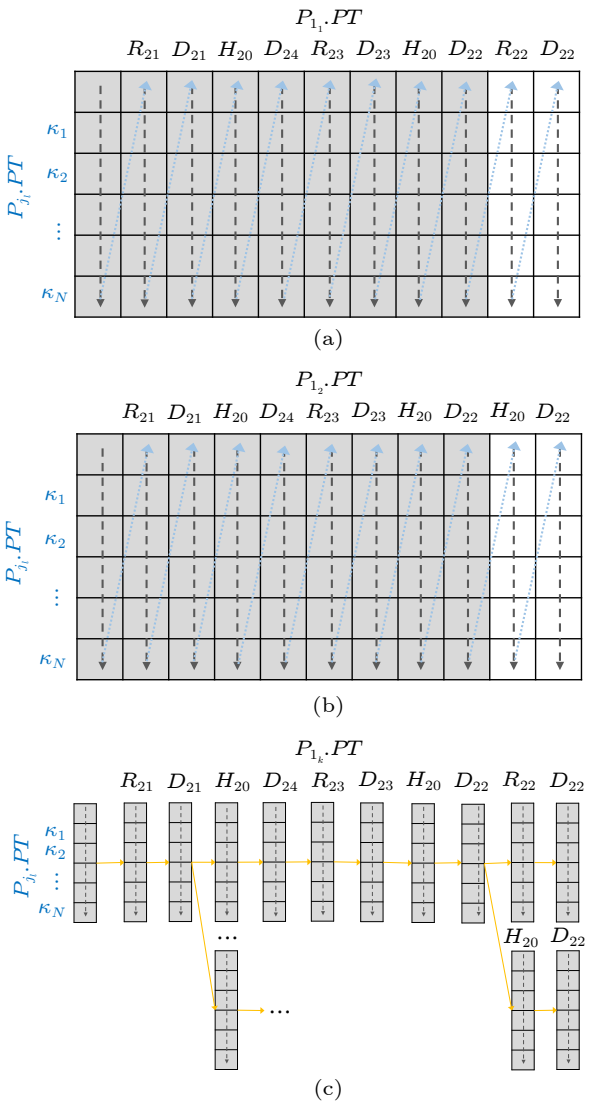

Fig.5. Reuse of calculated matrix. (a)Matrix used in the dynamic programming to compute $\Theta(P_{1_1}, P_{j_l})$. (b) Matrix used in the dynamic programming to compute $\Theta(P_{1_2}, P_{j_l})$. (c) Index storage structure for matrices IM for $P_{j_l}.PT$.

following the sequence of arrows.

For an indoor uncertain sematic trajectory τ_i, $\forall P_{i_k}, P_{i_{k'}} \in \tau_i$, potential indoor paths $P_{i_k}.PT$ and $P_{i_{k'}}.PT$ usually share many common triples. For example, $P_{1_1}.PT$ and $P_{1_2}.PT$ listed in Table 3 share the first eight triples. Then for a same potential indoor path $P_{j_l}.PT$, in the matrices used in the dynamic programming to compute $\Theta(P_{1_1}, P_{j_l})$ and $\Theta(P_{1_2}, P_{j_l})$, it is quite clear that the contents of the positions in shadow as shown in Fig.5(a) and Fig.5(b) are completely the same. In other words, after computing $\Theta(P_{1_1}, P_{j_l})$, in the calculation of $\Theta(P_{1_2}, P_{j_l})$, the contents of the positions in shadow can be directly copied from the matrix, which is used in the dynamic programming to compute $\Theta(P_{1_1}, P_{j_l})$ without any extra calculations.

In order to make full use of such a useful property in the calculation of $Part_1(\tau_i, \tau_j)$, we propose a novel index storage structure for matrices named IM built on each corresponding PT of $P_{i_k} \in \tau_i$ and $P_{j_l} \in \tau_j$. As an example shown in Fig.5(c), in the calculation of $Part_1(\tau_1, \tau_j)$, IM for $P_{j_l}.PT$ is constantly updated. After the calculation of $\Theta(P_{1_1}, P_{j_l})$, IM for $P_{j_l}.PT$ is initialized to 11 columns with pointers, which are shown in the top half of Fig.5(c). In the calculation of $\Theta(P_{1_2}, P_{j_l})$, since the first eight triples are shared by $P_{1_1}.PT$ and $P_{1_2}.PT$, the first $1+8=9$ columns of IM for $P_{j_l}.PT$ are directly inserted into the matrix of $\Theta(P_{1_2}, P_{j_l})$. After the calculation of $\Theta(P_{1_2}, P_{j_l})$, another two columns labeled with H_{20} and D_{22} of the matrix are inserted into IM for $P_{j_l}.PT$ for the next use, which is shown in the lower right hand corner of Fig.5(c).

The third acceleration strategy is to early terminate the calculation of $Part_1(\tau_i, \tau_j)$. The calculation of $Part_1(\tau_i, \tau_j)$ in line 15 of Algorithm 2 is to verify whether $\alpha \times part_1 + (1-\alpha) \times part_2 \geqslant \theta$ in line 16. In other words, we want to know whether $Part_1(\tau_i, \tau_j) \geqslant (\theta - (1-\alpha) \times part_2)/\alpha$. We define a lower bound LB and an upper bound UB of $Part_1(\tau_i, \tau_j)$, which are calculated as follows:

$$\forall \tau_i' \subseteq \tau_i \text{ and } \forall \tau_j' \subseteq \tau_j, LB = \sum_{P_{i_k} \in \tau_i'} \sum_{P_{j_l} \in \tau_j'} (P_{i_k}.L \times P_{j_l}.L \times \Theta'(P_{i_k}.PT, P_{j_l}.PT)),$$

$$UB = LB + \sum_{(P_{i_k}, P_{j_l}) \in ((\tau_i \times \tau_j)/(\tau_i' \times \tau_j'))} (P_{i_k}.L \times P_{j_l}.L) \times \frac{2 \times \min\{P_{i_1}.PT.ST(), P_{j_1}.PT.ST()\}}{P_{i_1}.PT.ST() + P_{j_1}.PT.ST()}.$$

In the process of calculating $Part_1(\tau_i, \tau_j)$, the progressively larger lower bound LB and the progressively smaller upper bound UB of $part_1$ are constantly generated. In this process, we check whether there exists a lower bound LB or an upper bound UB of $part_1$ meeting the condition that $LB \geqslant (\theta - (1-\alpha) \times part_2)/\alpha$ or $UB < (\theta - (1-\alpha) \times part_2)/\alpha$. If $LB \geqslant (\theta - (1-\alpha) \times part_2)/\alpha$, then the condition in line 16 must be satisfied. And if $UB < (\theta - (1-\alpha) \times part_2)/\alpha$, then the condition in line 16 must not be satisfied. And thus, the calculation of $Part_1(\tau_i, \tau_j)$ can be early terminated, and the condition in line 16 can be determined directly. If the potential indoor paths with greater likelihoods are selected in τ_i' and τ_j', the tighter upper bound UB and the tighter lower bound LB can be generated, and the third acceleration strategy will be more effective in reducing the time overhead.

In summary, by combining the acceleration strategies presented in Subsection 4.1 and Subsection 4.2, the time complexity of the USP algorithm is only $O((|PT|_{\max}^2 + |Iidx|_{\max} + |Tidx|_{\max}) \times |\tau|_{\max}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$, instead of $O((|PT|_{\max}^2 \times |\mathbb{D}|^2 + |\mathbb{IL}|^2 + |\mathbb{TL}|^2) \times |\tau|_{\max}^2 \times |\mathbb{P}| \times |\mathbb{Q}|)$, the time complexity of the straightforward baseline TII algorithm. Besides, by using the three acceleration strategies proposed in Subsection 4.3, lots of unnecessary calculations can be easily avoided in the USP algorithm at quite low computation cost.

5 Experimental Evalution

In this section, the performances of our USP algorithm on processing $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ is evaluated in detail.

5.1 Experimental Setup

5.1.1 Datasets

As far as we know, there is no real dataset being public or available, which contains indoor trajectories in a shopping mall. And thus, as what has been done in most related work[21, 22], a real-world floorplan[③] of a shopping mall is used. Each floor with 96 rooms, four hallways, and four staircases takes 136.8 m×136.8 m. We decompose the irregular hallways into smaller but regular indoor partitions. By duplicating such a floor, a multi-floor indoor space is generated, where each two adjacent floors are connected by four stairways. Then on each floor, there are 148 doors or virtual doors, and 109 indoor partitions in total. The detection ranges of all partitioning devices cover all entrances and exits of indoor partitions. In order to generate indoor uncertain semantic trajectories with different average number of potential indoor semantic paths, the distribution and the number of presence devices on each floor are varied. About the semantic information of the room in such indoor space, each room is randomly assigned to one IL, and each IL is associated with some TLs. These ILs and TLs are obtained from the online information of multiple shopping malls.

In order to generate indoor uncertain semantic trajectories in such indoor space, lots of trajectories are generated first by following the most commonly used method named Vita[23]. Then, according to the distribution of static positioning devices, these trajectories are transformed into the corresponding indoor trajectories. Finally, 2 000 indoor uncertain semantic trajectories are generated. There are 4.1 potential indoor semantic paths on average in each indoor uncertain semantic trajectory. And the average number of indoor partitions and doors in each potential indoor path is 64.9. In the following experiments, the testing datasets of indoor uncertain semantic trajectories, i.e., $\mathbb{P}$, $\mathbb{Q}$, and $\mathbb{T}$, with different sizes are all randomly selected from these 2 000 generated indoor uncertain semantic trajectories.

5.1.2 Experimental Environment

All methods are implemented in C++ and in a main memory fashion. All evaluations are conducted on a PC with a 3.60 GHz Intel® Core™ i9-9900K CPU and 32 GB RAM. Each experiment is repeated over three times, and the average result is reported.

The parameter settings are listed in Table 4.

Table 4. Parameter Settings

Variable	Range	Default
α	[0.1, 0.9]	0.3
F	[2, 8]	7.0
$I2T$	[1, 9]	5.0
$\|\mathbb{P}\|$	[50, 200]	200.0
$\|\mathbb{Q}\|$	[50, 200]	200.0
$\|\mathbb{T}\|$	[50, 200]	200.0
$\|\mathbb{IL}\| = \|\mathbb{TL}\|$	[2 000, 10 000]	10 000.0

5.2 Performance Evaluation for Algorithm USP

5.2.1 Impacts of the Number of Indoor Uncertain Semantic Trajectories

We first evaluate the total execution time of the USP algorithm and the TII algorithm in processing a query $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ w.r.t. varying $|\mathbb{P}|$ or $|\mathbb{Q}|$ from 50 to 200. $|\mathbb{P}|$ and $|\mathbb{Q}|$ are the sizes of the set $\mathbb{P}$ and $\mathbb{Q}$, respectively. The results are reported in Fig.6. The results show that the total execution time of the TII algorithm increases nearly linearly with the increase of $|\mathbb{P}|$ or $|\mathbb{Q}|$. And the USP algorithm is obviously faster than the TII algorithm. By using the USP algorithm, instead of the TII algorithm, at least 98.5% of the total execution time can be easily saved. And the

[③] https://www.deviantart.com/mjponso/art/Floor-Plan-for-a-Shopping-Mall-86 396 406, Nov. 2024.

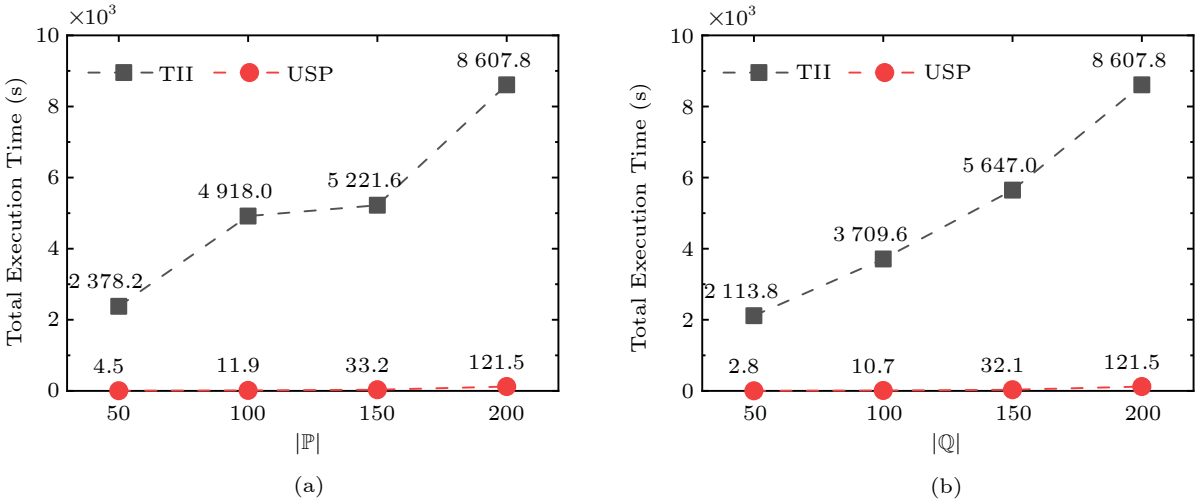

Fig.6. Evaluation of the total execution time: varying $|\mathbb{P}|$ or $|\mathbb{Q}|$. (a) Varying $|\mathbb{P}|$. (b) Varying $|\mathbb{Q}|$.

acceleration gets more obvious with the decrease of $|\mathbb{P}|$ or $|\mathbb{Q}|$. Besides, there is a key point needing further discussion in Fig.6(a). As illustrated in Subsection 5.1.1, the sets $\mathbb{P}$ (resp. $\mathbb{Q}$) with different sizes are all randomly selected from 2 000 generated indoor uncertain semantic trajectories. And the set $\mathbb{P}$ with size of 100 is not necessarily a subset of the set $\mathbb{P}$ with size of 150. Thus as shown in Fig.6(a), the randomness of testing datasets most likely leads to that the total execution time only increases a little after $|\mathbb{P}|$ is changed from 100 to 150.

In the process of an $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ query, the symbol $TotalPairs$ represents the number of all indoor uncertain semantic trajectory pairs in the candidate result set $\mathbb{P} \times \mathbb{Q}$, i.e., $TotalPairs = |\mathbb{P}| \times |\mathbb{Q}|$. And the symbol $FilteredPairs$ represents the number of invalid indoor uncertain semantic trajectory pairs pruned by using the inverted index 3IST. Since the inverted index 3IST is embedded in both the USP algorithm and the TII algorithm, both $TotalPairs$ and $FilteredPairs$ are the same in the USP algorithm and the TII algorithm. Then, $FilteredPairs$ and $TotalPairs$ of the USP algorithm are evaluated in processing a query $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ w.r.t. varying $|\mathbb{P}|$ or $|\mathbb{Q}|$ from 50 to 200, and the results are shown in Fig.7. By just using the inverted index 3IST, it is quite obvious that most invalid pairs of indoor uncertain semantic trajectories are pruned at quite low computation cost. Only at most 1.3% of all potential indoor uncertain semantic trajectory pairs left need to be further checked. Besides for 3IST, with the variation of $|\mathbb{P}|$ or $|\mathbb{Q}|$, the pruning ratio of invalid indoor uncertain semantic trajectory pairs is quite stable.

After that, we want to analyze the total execution time of the USP algorithm and the TII algo-

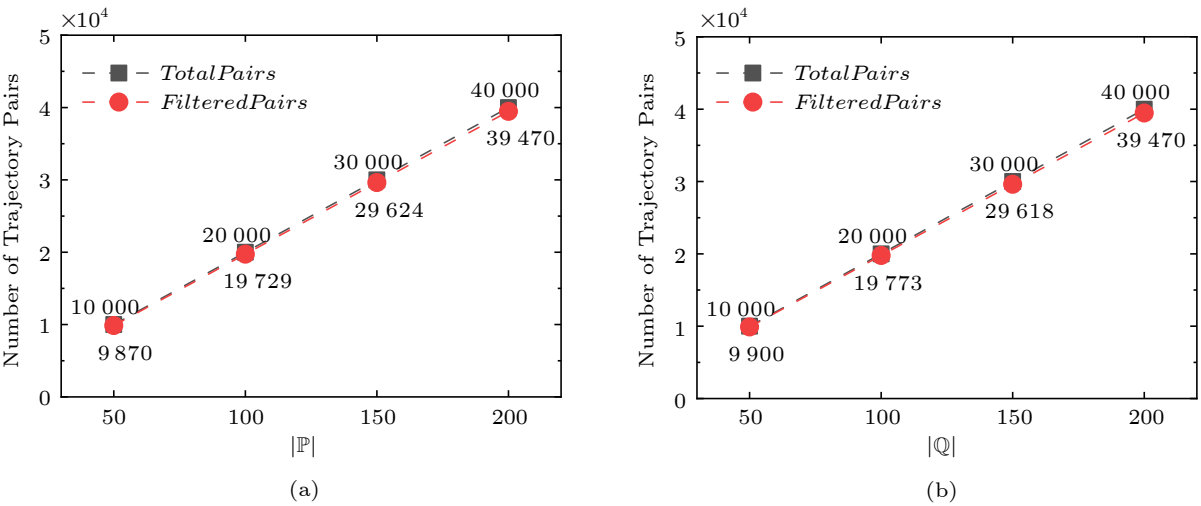

Fig.7. Evaluation of the inverted index 3IST: varying $|\mathbb{P}|$ or $|\mathbb{Q}|$. (a) Varying $|\mathbb{P}|$. (b) Varying $|\mathbb{Q}|$.

rithm in detail. As defined in Definition 4 and Definition 5, to process an $IUST\text{-}Join(\mathbb{P}, \mathbb{Q})$ query, many times of $\Theta()$ and $\Phi()$ functions need to be executed in both the USP algorithm and the TII algorithm. We divide the total execution time of the USP algorithm or the TII algorithm into three main parts, i.e., the execution time of $\Theta()$ functions, $\Phi()$ functions, and all the other parts. For both the USP algorithm and the TII algorithm, we evaluate the corresponding ratios of the the execution time of all $\Theta()$ functions, the execution time of all $\Phi()$ functions and the execution time of all the other parts to the total execution time w.r.t. varying $|\mathbb{P}|$ or $|\mathbb{Q}|$, and the results are reported in Fig.8. For both the USP algorithm and the TII algorithm, the execution of $\Theta()$ functions consumes extremely large part of the total execution time. This indicates that the calculation of $\Theta()$ is quite time-consuming. The execution time of all the other parts is always a little longer than that of $\Phi()$. For the TII algorithm, the corresponding ratios of the three main parts are all relatively stable. And the execution time of $\Phi()$ and all the other parts only take up less than 0.1% of the total execution time. For the USP algorithm, the corresponding ratio of $\Theta()$ functions gets much higher with the increase of $|\mathbb{P}|$ or $|\mathbb{Q}|$. Thus, we have reason to suspect that much more $\Theta()$ functions need to be executed in the USP algorithm with the increase of $|\mathbb{P}|$ or $|\mathbb{Q}|$. Such a guess will be verified in the following experiments. For the execution time of $\Theta()$ functions, the corresponding ratio in the execution of the TII algorithm is always a little higher than that in the execution of the USP algorithm.

To further figure out why the USP algorithm only needs no more than 1.5% of the total execution time that the TII algorithm needs, and for the TII algorithm, why the corresponding ratio of $\Theta()$ is always a little higher than that in the execution of the USP algorithm, the number of the executed $\Theta()$ functions is evaluated w.r.t. varying $|\mathbb{P}|$ or $|\mathbb{Q}|$, and the results are reported in Fig.9. It can be easily seen that

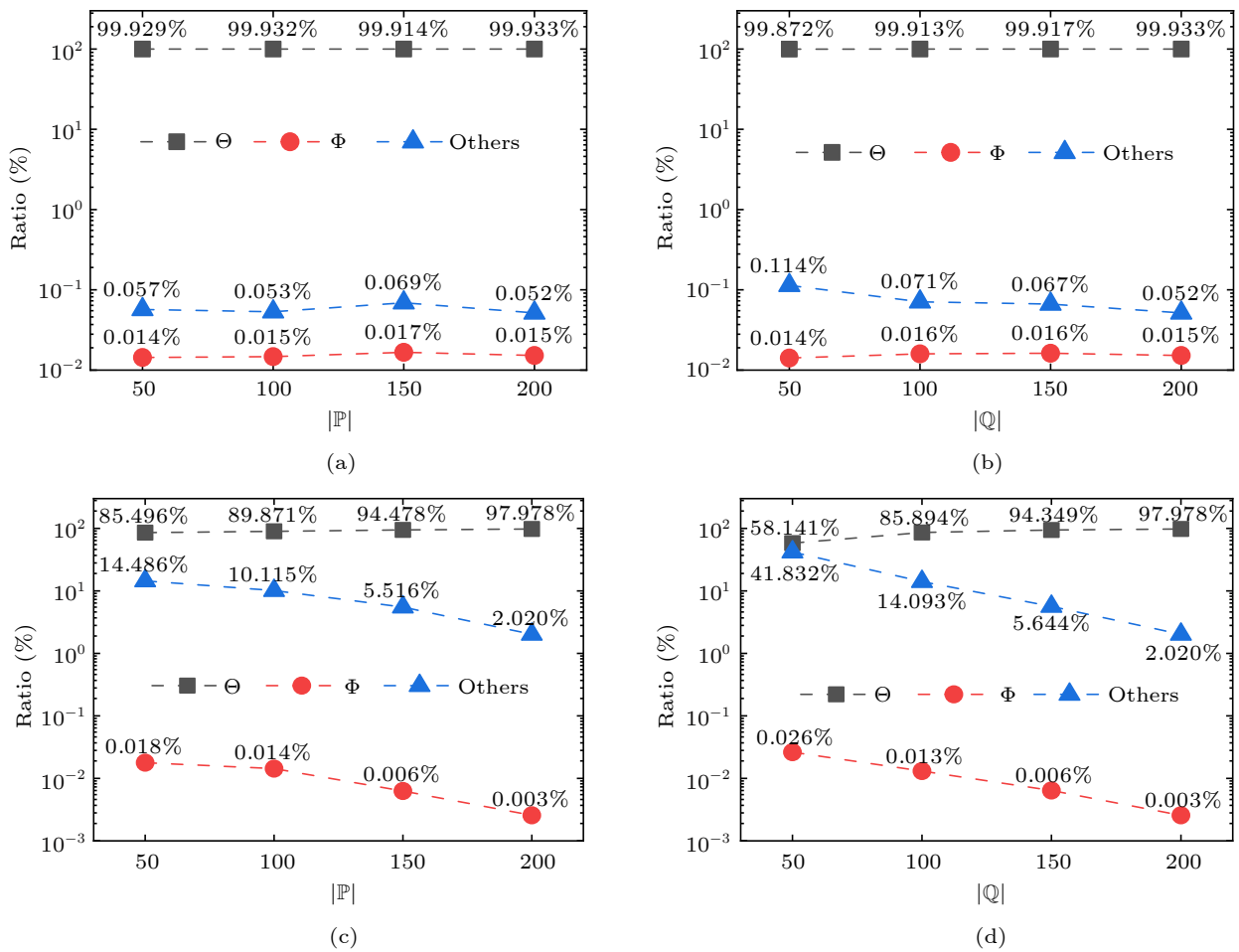

Fig.8. Evaluation of three ratios: varying $|\mathbb{P}|$ or $|\mathbb{Q}|$. (a) The TII algorithm: varying $|\mathbb{P}|$. (b) The TII algorithm: varying $|\mathbb{Q}|$. (c) The USP algorithm: varying $|\mathbb{P}|$. (d) The USP algorithm: varying $|\mathbb{Q}|$.

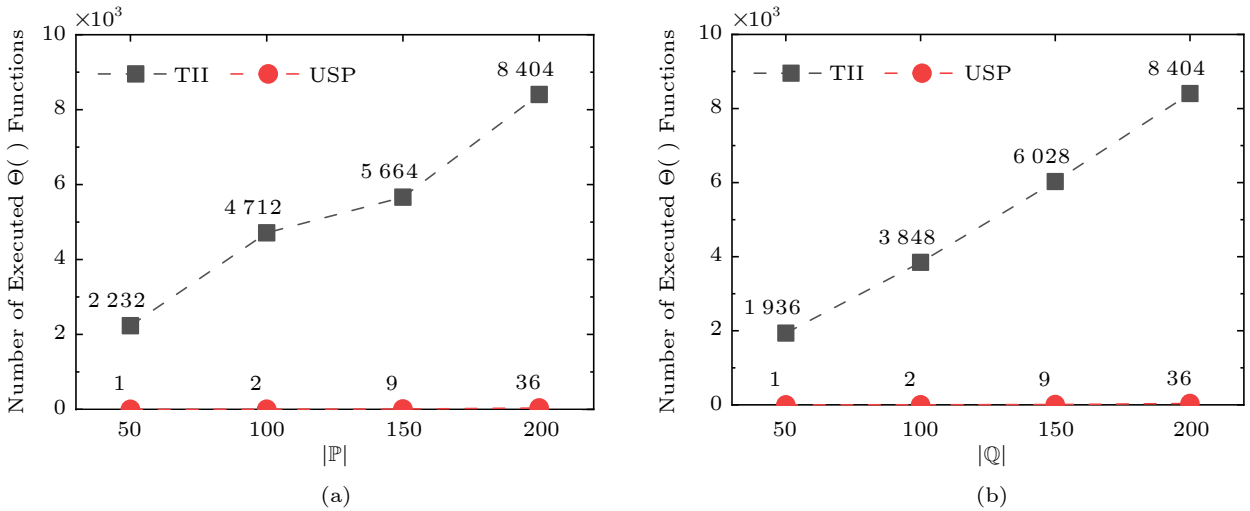

Fig.9. Evaluation of the number of the executed $\Theta()$ functions: varying $|\mathbb{P}|$ or $|\mathbb{Q}|$. (a) Varying $|\mathbb{P}|$. (b) Varying $|\mathbb{Q}|$.

Fig.6(a) and Fig.9(a) are quite similar, and so do Fig.6(b) and Fig.9(b). This indicates that for both the TII algorithm and the USP algorithm, the number of the executed $\Theta()$ functions determines the total execution time to a great extent. And much fewer $\Theta()$ functions needing to be executed make the USP algorithm need much less execution time than the TII algorithm. Besides, for the three acceleration strategies presented in Subsection 4.3, though their abilities of avoiding $\Theta()$ functions to be executed are always remarkable, their such abilities decrease a little with the increase of $|\mathbb{P}|$ or $|\mathbb{Q}|$. Much more $\Theta()$ functions need to be executed in the USP algorithm with the increase of $|\mathbb{P}|$ or $|\mathbb{Q}|$. In addition, how many $\Theta()$ functions to be executed can be avoided by using either of the three acceleration strategies presented in Subsection 4.3 will be studied in detail by using extensive experiments later in Subsection 5.2.4.

5.2.2 *Impacts of Pre-Calculation for Door-to-Door Indoor Distances*

The calculation of function $\Theta()$ is time-consuming, and we find that there are lots of repeated calculations of door-to-door indoor distances. Thus, we propose the first and the second pre-calculation methods in Subsection 3.3 and Subsection 4.1, respectively, to make door-to-door indoor distances pre-calculated to reduce the time cost. For these two methods, we evaluate the pre-calculation time w.r.t. the various total number of floors F in the shopping mall. From the results shown in Fig.10, we find the first method always needs more pre-calculation time than the second one. In addition, the pre-calculation time of the first one grows quite faster and that of the second one grows nearly linearly. These phenomena are matched with the corresponding time complexities $O({D_{\text{avg}}}^3 \times F^3)$ and $O({D_{\text{avg}}}^3 \times F)$, respectively.

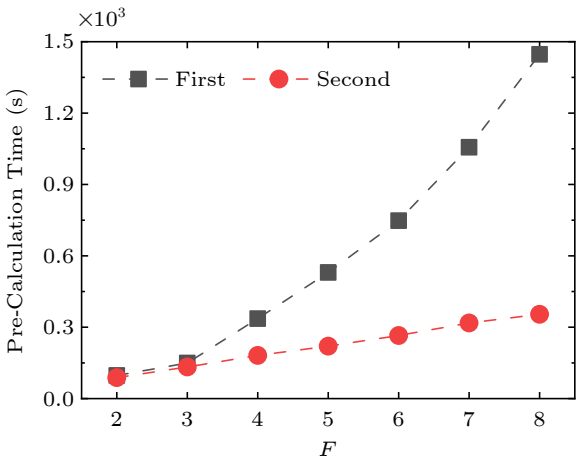

Fig.10. Evaluation of the pre-calculation time: varying F.

In the following experiment, it is assumed that the pre-calculation time is also considered in the process of IUST-Join. The following four exection time metrics are used.

- $\overline{USP}$: the exection time of the USP algorithm, where part of door-to-door indoor distances have been pre-calculated by the second pre-calculation method;
- $\overline{USP\_A}$: the exection time of the USP algorithm, where all door-to-door indoor distances have been pre-calculated by the first pre-calculation method;
- $\overline{USP\&SP}$: the sum of $\overline{USP}$ and the pre-calculation time of the second pre-calculation method;
- $\overline{USP\_A\&FP}$: the sum of $\overline{USP\_A}$ and the pre-calculation time of the first pre-calculation method.

The total number of floors F is fixed as 7. Then we evaluate the execution time on processing an $IUST\text{-}Join(\mathbb{P},\mathbb{Q})$ query w.r.t. varying $|\mathbb{P}|$ or $|\mathbb{Q}|$ from 50 to 200, and the results are reported in Fig.11. It can be found that by using the first pre-calculation method to make all door-to-door indoor distances pre-calculated, $\overline{USP\_A}$ is a little less than $\overline{USP}$, where only part of the door-to-door indoor distances have been pre-calculated. However, when the corresponding pre-calculation time is also considered as part of the total exection time, for one IUST-Join query, using the USP algorithm with part of door-to-door indoor distances pre-calculated by the second pre-calculation method is much more time-saving than using the USP algorithm with all door-to-door indoor distances pre-calculated by the first pre-calculation method. This means that if a limited number of IUST-Join queries need to be processed, the USP algorithm is more preferred.

We also try to use the USP algorithm to process one IUST-Join query, where none of door-to-door indoor distances has been pre-calculated. This algorithm is used to process the same query $IUST\text{-}Join(\mathbb{P},\mathbb{Q})$, where $|\mathbb{P}|=50$ and $|\mathbb{Q}|=200$. However, this algorithm is too time-consuming to get the final result in five days. Thus, we choose to stop the experiment. To conclude, considering the consumed time, the USP algorithm is completely infeasible when the door-to-door indoor distances are not pre-calculated.

5.2.3 Impacts of Semantic Information Encoding

In the following experiments, we evaluate the following three methods.

- *USP*. It is the USP algorithm introduced detailedly in Section 4. And each semantic similarity $\Phi()$ function is calculated by using the PUI algorithm, i.e., Algorithm 3.
- *USP\SL*. It is the USP algorithm where the semantic information of indoor uncertain semantic trajectories is not encoded by SL-encoding, and each semantic similarity $\Phi()$ function is computed by using a naive method.
- *USP\PUT*. It is the USP algorithm where the semantic information of indoor uncertain semantic trajectories is encoded by SL-encoding, and each semantic similarity $\Phi()$ function is calculated by following (1).

As introduced in Subsection 2.2, each room relates to only one IL, and each IL is associated with a set of TLs. In this experiment, we evaluate the average execution time of a single semantic similarity calculation $\Phi()$ w.r.t. varying $I2T$, i.e., the number of TLs associated with each IL ranges from 1 to 9, and the results are shown in Fig.12(a). Obviously, the PUI Algorithm is the fastest algorithm in computing the semantic similarity $\Phi()$. And by using the PUI Algorithm, instead of the naive method in USP\SL, up to 88.8% time can be saved in the semantic similarity calculations. As analyzed in Subsection 4.2, such a result is matched with that the time complexity of the PUI algorithm, i.e., $O(|Iidx|_{\max}+|Tidx|_{\max})$. And the time complexity of the PUI Algorithm is the lowest. With the increase of $I2T$, more TLs need to be processed in the semantic similarity calculations. Thus both USP and USP\SL are slower in semantic similarity calculations. However, with the variation of $I2T$, USP\PUI performs quite stably. This is because two vectors, whose sizes are $|\mathbb{IL}|$ and $|\mathbb{TL}|$, respective-

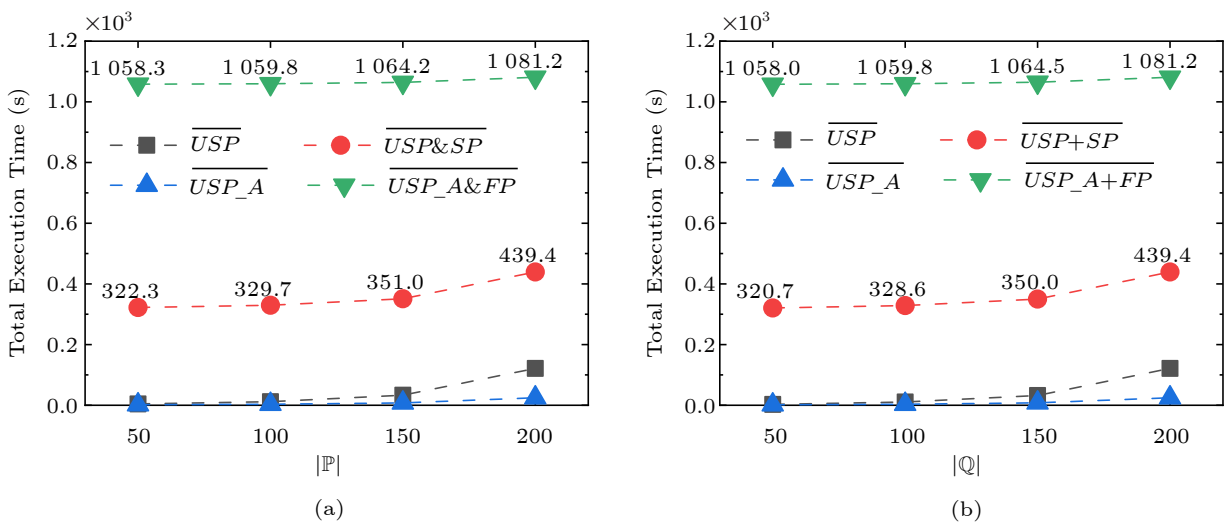

Fig.11. Evaluation of two pre-calculation methods: varying $|\mathbb{P}|$ or $|\mathbb{Q}|$. (a) Varying $|\mathbb{P}|$. (b) Varying $|\mathbb{Q}|$.

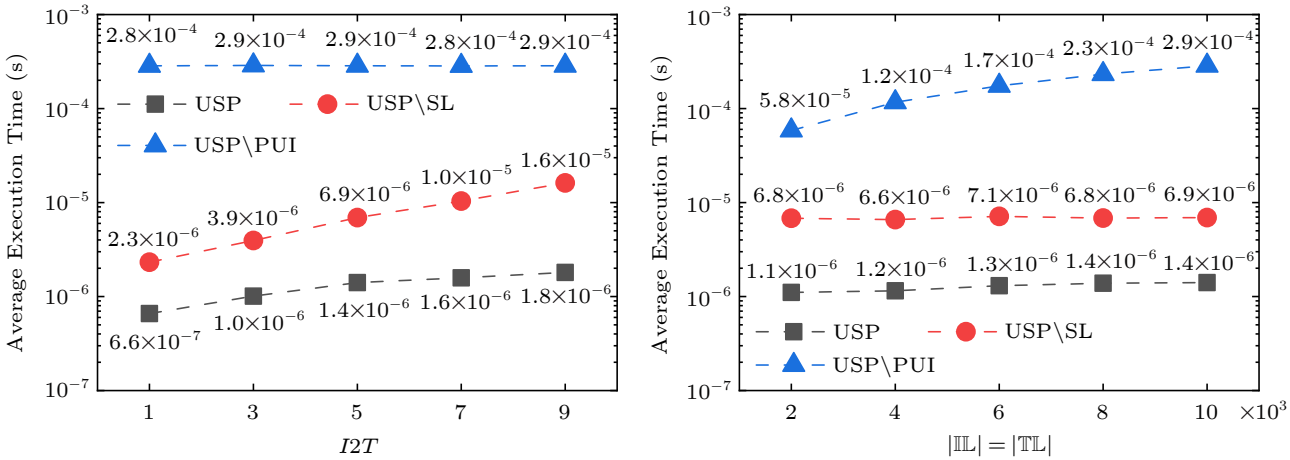

Fig.12. Evaluation of the average execution time of a single $\Phi()$ function. (a) Varying $I2T$. (b) Varying $|\mathbb{IL}|$ and $|\mathbb{TL}|$.

ly, are used to encode AIL and ATL, respectively, and the calculation of (1) is not sensitive to $I2T$. Thus in computing the semantic similarity $\Phi()$, USP\PUI is sure to be sensitive to $|\mathbb{IL}|$ and $|\mathbb{TL}|$, which is studied below.

In this experiment, the average execution time of a single semantic similarity calculation $\Phi()$ is also evaluated w.r.t. varying $|\mathbb{IL}|$ and $|\mathbb{TL}|$ at the same time, and the results are shown in Fig.12(b). The execution time of USP\PUI in computing the semantic similarity $\Phi()$ grows nearly linearly with the increase of $|\mathbb{IL}|$ and $|\mathbb{TL}|$, which is matched with the corresponding time complexity $O(|\mathbb{IL}| + |\mathbb{TL}|)$. And it is obvious that both USP and USP\SL perform relatively stably with the variation of $|\mathbb{IL}|$ and $|\mathbb{TL}|$. In computing the semantic similarity $\Phi()$, though less execution time is needed by USP\PUI with the decrease of $|\mathbb{IL}|$ and $|\mathbb{TL}|$, USP\PUI still consumes more time than both USP and USP\SL. This is because for USP\PUI, much more calculations are needed in all positions of $|\mathbb{IL}|$ and $|\mathbb{TL}|$. And by using the PUI Algorithm, instead of the naive method in USP\SL, up to 83.7% time can be saved in the semantic similarity calculations.

Then we evaluate the total execution time of USP and USP\SL in processing IUST-Join w.r.t. varying $I2T$, $|\mathbb{IL}|$, and $|\mathbb{TL}|$, and the results are reported in Fig.13(a) and Fig.13(b), respectively. We can see that by using the PUI algorithm instead of the naive method, though the execution time of computing the semantic similarity $\Phi()$ is reduced greatly, the reduced execution time in processing IUST-Join is quite limited. To find out the reason, we divide the total execution time into three main parts, i.e., the execution time of $\Theta()$ functions, $\Phi()$ functions and all the other parts. Then the corresponding ratio of the execution time of $\Phi()$ functions to the total execution time is evaluated w.r.t. varying $I2T$, $|\mathbb{IL}|$, and $|\mathbb{TL}|$.

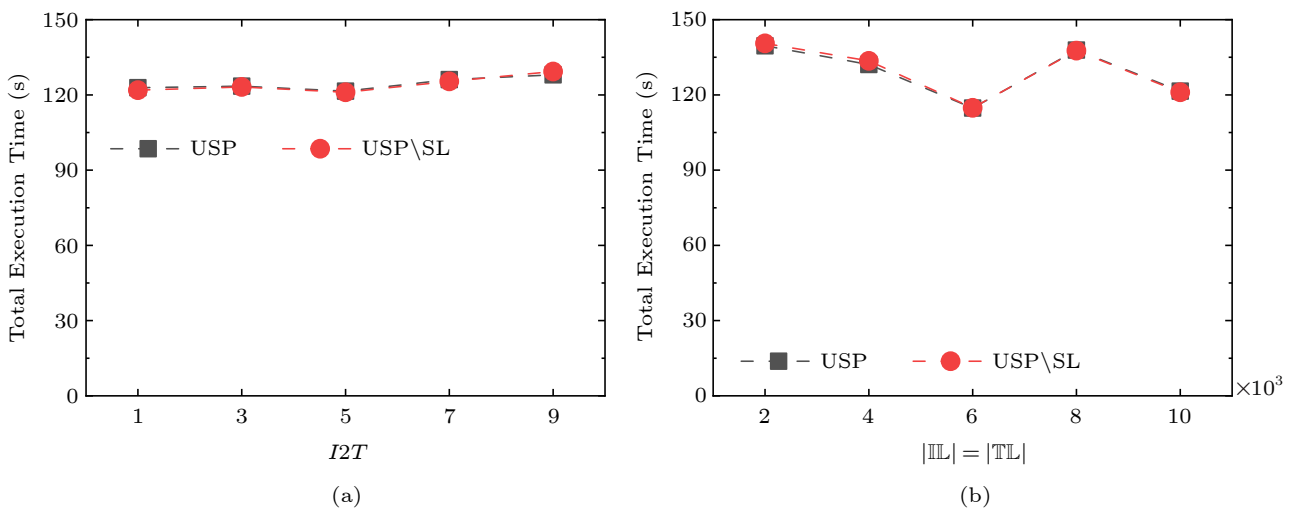

Fig.13. Evaluation of the total execution time. (a) Varying $I2T$. (b) Varying $|\mathbb{IL}|$ and $|\mathbb{TL}|$.

And the results are reported in Fig.14(a) and Fig.14(b), respectively. It is quite clear that for either the PUI algorithm or the naive method, the execution time of $\Phi()$ functions is far less than the total execution time, which is also confirmed by the results shown in Fig.8. In other words, if we want to reduce the execution time of processing IUST-Join greatly, more attention may need to be put in reducing the execution time of $\Theta()$ functions.

5.2.4 *Impacts of Three Acceleration Strategies*

In the following experiments, we evaluate the following four methods.

- USP: the USP algorithm, which is intorduced detailedly in Section 4;
- USP\First: the USP algorithm, but without using the first acceleration strategy presented in Subsection 4.3;
- USP\Second: the USP algorithm, but without using the second acceleration strategy presented in Subsection 4.3;
- USP\Third: the USP algorithm, but without using the third acceleration strategy presented in Subsection 4.3.

In the first experiment, the total execution time and the number of executed $\Theta()$ functions of these four methods are evaluated in processing IUST-Join, w.r.t. varying the parameter α from 0.1 to 0.9, where the parameter α is used in Definition 4. The results are shown in Fig.15. The trends in Figs.15(a) and 15(b) are quite similar, which shows that the number of $\Theta()$ functions needing to be executed determines the total execution time to a great extent. With the

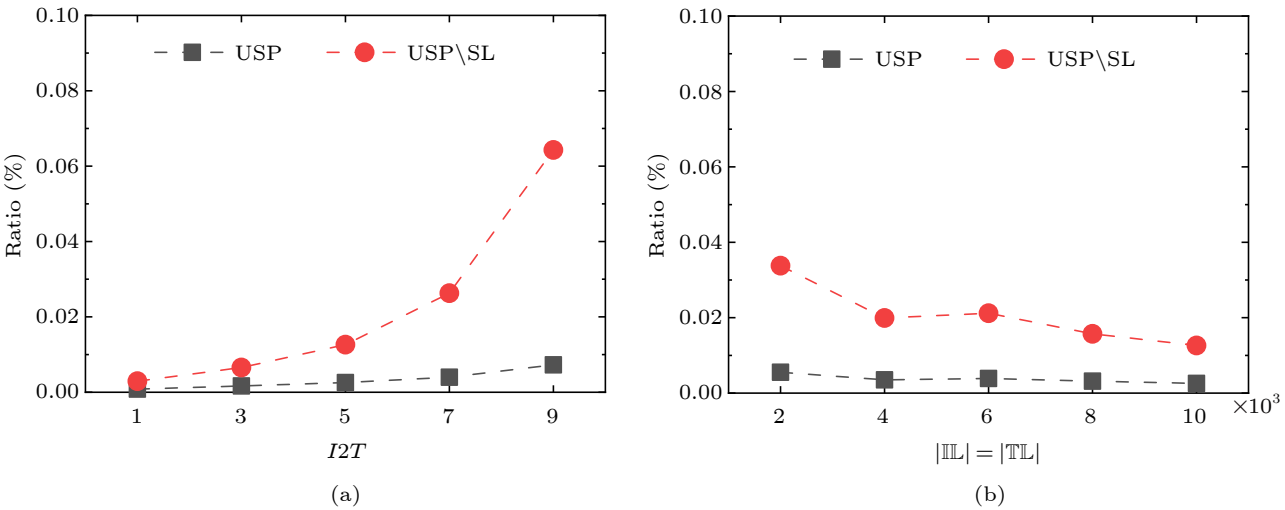

Fig.14. Evaluation of the corresponding ratio of the execution time $\Phi()$ functions to the total execution time. (a) Varying $I2T$. (b) Varying $|\mathbb{IL}|$ and $|\mathbb{TL}|$.

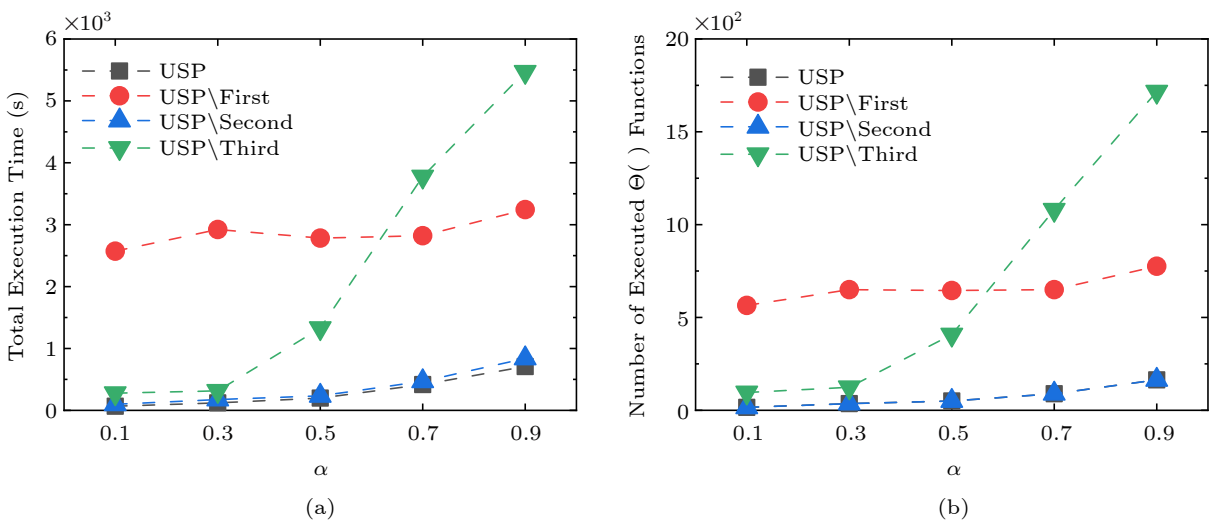

Fig.15. Evaluation of the total execution time and the number of executed $\Theta()$ functions: varying α. (a) Total execution time. (b) Number of executed $\Theta()$ functions.

increase of α, the importance of the spatial proximity between two indoor uncertain semantic trajectories increases in terms of the similarity metric IUS. And thus, for the given IUST-Join, more $\Theta()$ functions need to be executed in these four methods, which is proved by Fig.15(b). Through analyzing the data, we can get that by using the first and the third acceleration strategies, at least 78.9% and 71.0% of the executed $\Theta()$ functions can be avoided, and 89.9% and 79.7% of the total execution time can be easily saved on average, respectively. In order to reduce the execution time of $\Theta()$ functions, the second acceleration strategy is to reuse part of the calculated matrix. And thus, for USP and USP\Second, the numbers of $\Theta()$ functions needing to be executed are completely the same. The results show that by using the second acceleration strategy, 20.9% of the total execution time can be saved on average. And the acceleration effect of the second acceleration strategy on the execution time of $\Theta()$ functions is studied in the follows.

We evaluate the average execution time of a single $\Theta()$ function of USP and USP\Second w.r.t. varying α from 0.1 to 0.9, and the results are reported in Fig.16. By reusing part of the calculated matrix, on average 20.5% of the average execution time of a single $\Theta()$ function can be reduced. And with the decrease of α, the acceleration effect of using the second acceleration strategy is much more obvious.

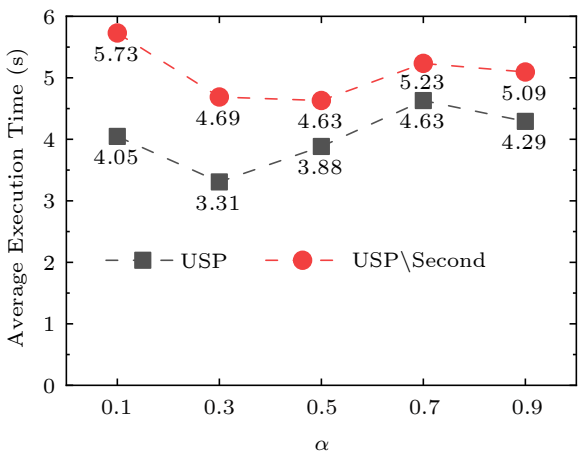

Fig.16. Evaluation of the average execution time of a single $\Theta()$ function: varying α.

6 Related Work

Indoor Space Data Model. Though indoor space has smaller extents, distinct entities, such as doors, walls, different kinds of rooms, hallways, and staircases, altogether form much more complex indoor topology that constrains and enables the movements. And thus, before the execution of queries in indoor space, the conversion from the complex indoor space to an indoor space data model is the most fundamental operation. 3D models for indoor space were proposed by [24], and they focus on topological relationships between indoor partitions. However, these models are not able to support indoor distance calculations. IndoorGML[25] and CityGML④ are two methods where indoor space is described in XML. Recently, much more attention is attracted by mapping the indoor space to a graph, where the door-to-door graph[15] and the extended accessibility base graph[16] are two commonly used methods. Graph algorithms can be adopted to search on such a graph to support various indoor spatial queries, such as the indoor distance calculation between two indoor positions. For the door-to-door graph, each door is mapped to a graph vertex. And there is a weighted edge between two vertexs if and only if the corresponding doors belong to a same indoor partition. The indoor distance between these two doors is the weight on the corresponding edge. For the extended accessibility base graph, each indoor partition is represented as a graph vertex. And there is an edge between two indoor partitions if and only if these two indoor partitions share a same door. In this paper, we choose to map the complex indoor space to the extended accessibility base graph G_{acc}, since most recent work[16, 17] has adopted this method.

Queries in Indoor Space. Among location-based services, route planning is always the most popular and useful one. Recently, route planning in various indoor venues such as airports, railway stations, and shopping malls is increasingly needed and attracts more attention. Some work has been done to support various indoor navigations with different qualifications and qualifications, such as [17, 21]. As two kinds of primitive, yet quite essential indoor queries, nearest neighbor (NN) queries[16] and range queries[16, 26] are to find the needed static indoor POIs (points of interest). Besides, there are also some studies on online indoor moving objects, such as distance-aware spatial joins[27], snapshot kNN search[15, 28], and continuous range monitoring[29]. Among them, the distance-aware spatial join is to find all such pairs of indoor moving objects, i.e., the distance between each pair of indoor moving objects is within a given distance. And thus, the distance-aware spatial join is completely different from IUST-Join firstly defined in

④http://www.citygml.org/, Nov. 2024.

this paper. IUST-Join can be used in a number of essential indoor applications, such as friend recommendation based on similar shopping interests, geo-text data cleaning, keyword-aware indoor route recommendation, trajectory near-duplicate detection, and so on. Therefore, we choose to detailedly study and efficiently process IUST-Join in this paper.

7 Conclusions

In this paper, we proposed the highly-efficient USP algorithm to process a new essential query named IUST-Join. The results of extensive experiments showed the effectiveness and efficiency of the USP algorithm on processing IUST-Join. By just using the inverted index 3IST embedded in the USP algorithm, as invalid pairs, at least 98.7% of all potential indoor uncertain semantic trajectory pairs can be pruned at quite low computation cost. Besides, by using our proposed USP algorithm to process IUST-Join, instead of the well-designed baseline TII algorithm, at least 98.5% of the execution time can be easily saved. In future work, we will do more work on how to efficiently process other useful queries on indoor uncertain semantic trajectories.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Klepeis N E, Nelson W C, Ott W R, Robinson J P, Tsang A M, Switzer P, Behar J V, Hern S C, Engelmann W H. The National Human Activity Pattern Survey (NHAPS): A resource for assessing exposure to environmental pollutants. *Journal of Exposure Science & Environmental Epidemiology*, 2001, 11(3): 231–252. DOI: 10.1038/sj.jea.7500165.

[2] Li F, Zhao C, Ding G, Gong J, Liu C, Zhao F. A reliable and accurate indoor localization method using phone inertial sensors. In *Proc. the 2012 ACM Conference on Ubiquitous Computing*, Sept. 2012, pp.421–430. DOI: 10.1145/2370216.2370280.

[3] Werner M. Indoor Location-Based Services -Prerequisites and Foundations. Springer, 2014. DOI: 10.1007/978-3-319-10699-1.

[4] Baba A I, Jaeger M, Lu H, Pedersen T B, Ku W S, Xie X. Learning-based cleansing for indoor RFID data. In *Proc. the 2016 International Conference on Management of Data*, Jun. 2016, pp.925–936. DOI: 10.1145/2882903.2882907.

[5] Niedermayer J, Züfle A, Emrich T, Renz M, Mamoulis N, Chen L, Kriegel H P. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *Proceedings of the VLDB Endowment*, 2013, 7(3): 205–216. DOI: 10.14778/2732232.2732239.

[6] Zhang C, Zhang K, Yuan Q, Peng H, Zheng Y, Hanratty T, Wang S, Han J. Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. In *Proc. the 26th Int. Conf. World Wide Web*, Apr. 2017, pp.361–370. DOI: 10.1145/3038912.3052601.

[7] Xie X, Mei B, Chen J, Du X, Jensen C S. Elite: An elastic infrastructure for big spatiotemporal trajectories. *The VLDB Journal*, 2016, 25(4): 473–493. DOI: 10.1007/s00778-016-0425-6.

[8] Chen L, Gao Y, Fang Z, Miao X, Jensen C S, Guo C. Real-time distributed co-movement pattern detection on streaming trajectories. *Proceedings of the VLDB Endowment*, 2019, 12(10): 1208–1220. DOI: 10.14778/3339490.3339502.

[9] Fang Z, Chen L, Gao Y, Pan L, Jensen C S. Dragoon: A hybrid and efficient big trajectory management system for offline and online analytics. *The VLDB Journal*, 2021, 30(2): 287–310. DOI: 10.1007/s00778-021-00652-x.

[10] Fang Z, Pan L, Chen L, Du Y, Gao Y. MDTP: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *Proceedings of the VLDB Endowment*, 2021, 14(8): 1289–1297. DOI: 10.14778/3457390.3457394.

[11] Shang S, Chen L, Wei Z, Jensen C S, Zheng K, Kalnis P. Trajectory similarity Join in spatial networks. *Proceedings of the VLDB Endowment*, 2017, 10(11): 1178–1189. DOI: 10.14778/3137628.3137630.

[12] Shang Z, Li G, Bao Z. DITA: Distributed in-memory trajectory analytics. In *Proc. the 2018 International Conference on Management of Data*, Jun. 2018, pp.725-740. DOI: 10.1145/3183713.3183743.

[13] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases. In *Proc. the 1994 ACM SIGMOD Int. Conf. Management of Data*, May 1994, pp.419–429. DOI: 10.1145/191839.191925.

[14] Yuan H, Li G. Distributed in-memory trajectory similarity search and join on road network. In *Proc. the 35th IEEE International Conference on Data Engineering*, Apr. 2019, pp.1262–1273. DOI: 10.1109/ICDE.2019.00115.

[15] Yang B, Lu H, Jensen C S. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *Proc. the 13th International Conference on Extending Database Technology*, Mar. 2010, pp.335–346. DOI: 10.1145/1739041.1739083.

[16] Lu H, Cao X, Jensen C S. A foundation for efficient indoor distance-aware query processing. In *Proc. the 28th IEEE International Conference on Data Engineering*, Apr. 2012, pp.438–449. DOI: 10.1109/ICDE.2012.44.

[17] Shao Z, Cheema M A, Taniar D, Lu H, Yang S. Efficiently processing spatial and keyword queries in indoor venues. *IEEE Trans. Knowledge and Data Engineering*, 2021, 33(9): 3229–3244. DOI: 10.1109/TKDE.2020.2964206.

[18] Dijkstra E W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959, 1(1): 269–271. DOI: 10.1007/BF01386390.

[19] Liu S, Li G, Feng J. A prefix-filter based method for spatio-textual similarity join. *IEEE Trans. Knowledge and Data Engineering*, 2014, 26(10): 2354–2367. DOI: 10.1109/

TKDE.2013.83.

[20] Chen L, Shang S, Jensen C S, Yao B, Kalnis P. Parallel semantic trajectory similarity join. In *Proc. the 36th IEEE International Conference on Data Engineering*, Apr. 2020, pp.997–1008. DOI: 10.1109/ICDE48307.2020.00091.

[21] Liu T, Li H, Lu H, Cheema M A, Shou L. Towards crowd-aware indoor path planning. *Proceedings of the VLDB Endowment*, 2021, 14(8): 1365–1377. DOI: 10.14778/3457390.3457401.

[22] Liu T, Feng Z, Li H, Lu H, Cheema M A, Cheng H, Xu J. Shortest path queries for indoor venues with temporal variations. In *Proc. the 36th IEEE International Conference on Data Engineering*, Apr. 2020, pp.2014–2017. DOI: 10.1109/ICDE48307.2020.00227.

[23] Li H, Lu H, Chen X, Chen G, Chen K, Shou L. Vita: A versatile toolkit for generating indoor mobility data for real-world buildings. *Proceedings of the VLDB Endowment*, 2016, 9(13): 1453–1456. DOI: 10.14778/3007263.3007282.

[24] Lee J. A spatial access-oriented implementation of a 3-D GIS topological data model for urban entities. *GeoInformatica*, 2004, 8(3): 237–264. DOI: 10.1023/B:GEIN.0000034820.93914.d0.

[25] Kim J S, Yoo S J, Li K J. Integrating IndoorGML and CityGML for indoor space. In *Proc. Int. Symp. Web and Wireless Geographical Information Systems*, Apr. 2014, pp.184–196. DOI: 10.1007/978-3-642-55334-9_12.

[26] Yuan W, Schneider M. Supporting continuous range queries in indoor space. In *Proc. the 11th International Conference on Mobile Data Management*, May 2010, pp.209–214. DOI: 10.1109/MDM.2010.21.

[27] Xie X, Lu H, Pedersen T B. Distance-aware join for indoor moving objects. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(2): 428–442. DOI: 10.1109/TKDE.2014.2330834.

[28] Xie X, Lu H, Pedersen T B. Efficient distance-aware query evaluation on indoor moving objects. In *Proc. the 29th IEEE International Conference on Data Engineering*, Apr. 2013, pp.434–445. DOI: 10.1109/ICDE.2013.6544845.

[29] Yang B, Lu H, Jensen C S. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *Proc. the 18th ACM Conference on Information and Knowledge Management*, Nov. 2009, pp.671–680. DOI: 10.1145/1645953.1646039.

Hong-Bo Yin received his B.S. degree in computer science and technology from Harbin Institute of Technology, Harbin, in 2018. He is currently working toward his Ph.D. degree in the Faculty of Computing, Harbin Institute of Technology, Harbin. His research interests include trajectory compression and query processing on trajectories.

Dong-Hua Yang received his B.S. M.S. and Ph.D. degrees from Harbin Institute of Technology, Harbin, in 1999, 2003, and 2008, respectively. He is a vice professor of the Faculty of Computing and the Center of Analysis, Measurement and Computing, Harbin Institute of Technology, Harbin. His research interests include big data management and analytics.

Kai-Qi Zhang received his B.S. and Ph.D. degrees in computer science from Harbin Institute of Technology, Harbin, in 2013 and 2020, respectively. He is currently a lecturer in the Faculty of Computing, Harbin Institute of Technology, Harbin. His main research interests include query processing, massive data management, and data-intensive computing.

Hong Gao received her B.S. and M.S. degrees in computer science from Heilongjiang University, Harbin, in 1988 and 1991, respectively. And she received her Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, in 2004. She is currently a professor with the Center of Analysis, Measurement and Computing, Harbin Institute of Technology, Harbin. Her research interests include query processing, sensor networks, and massive data management.

Jian-Zhong Li received his B.S. degree from Heilongjiang University, Harbin, in 1975. He is currently a professor in Faculty of Computer Science and Control Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen. His research interests include data management systems, sensor networks, and data intensive computing.