Parallel Trajectory-to-Location Join

Shang, Shuo; Chen, Lisi; Zheng, Kai; Jensen, Christian S.; Wei, Zhewei; Kalnis, Panos

Published in:
IEEE Transactions on Knowledge and Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2018.2854705

Publication date:
2018

Document Version
Accepted author manuscript, peer reviewed version

Parallel Trajectory-to-Location Join

Shuo Shang, Lisi Chen, Kai Zheng, Christian S. Jensen, *Fellow, IEEE,* Zhewei Wei, and Panos Kalnis

Abstract—The matching between trajectories and locations, called Trajectory-to-Location join (TL-Join), is fundamental functionality in spatiotemporal data management. Given a set of trajectories, a set of locations, and a threshold θ, the TL-Join finds all (trajectory, location) pairs from the two sets with spatiotemporal correlation above θ. This join targets diverse applications, including location recommendation, event tracking, and trajectory activity analyses. We address three challenges in relation to the TL-Join: how to define the spatiotemporal correlation between trajectories and locations, how to prune the search space effectively when computing the join, and how to perform the computation in parallel. Specifically, we define new metrics to measure the spatiotemporal correlation between trajectories and locations. We develop a novel parallel collaborative (PCol) search method based on a divide-and-conquer strategy. For each location o, we retrieve the trajectories with high spatiotemporal correlation to o, and then we merge the results. An upper bound on the spatiotemporal correlation and a heuristic scheduling strategy are developed to prune the search space. The trajectory searches from different locations are independent and are performed in parallel, and the result merging cost is independent of the degree of parallelism. Studies of the performance of the developed algorithms using large spatiotemporal data sets are reported.

Keywords—Trajectory-to-location join, Parallel Processing, Spatial networks, Spatial databases

1 Introduction

With the continuous proliferation of GPS-enabled mobile devices (e.g., vehicle navigation systems and smart phones) and the rapid development of online map-based services (e.g., Google Maps[1], and MapQuest[2]), it is easy to collect and share trajectories, e.g., at specialized sites such as Bikely[3], GPS-way-points[4], Share-my-routes[5], and Microsoft Geolife[6]. Also, more and more social networking sites, including Twitter[7], Facebook[8], and Foursquare[9], are starting to support trajectory collection and sharing [16], [19]. The availability of massive trajectory data motivates new studies in spatiotemporal data management. The matching between trajectories and locations, called Trajectory-to-Location Join (TL-Join), is fundamental functionality. Given a set T of trajectories, a set O of locations, and a threshold θ, the TL-Join finds all (trajectory, location) pairs from T and O with a spatiotemporal correlation above θ.

The TL-Join may benefit diverse applications, including location recommendation [21], event tracking [28], and trajectory activity analyses [12], [20], [25]. For example, people may want to place new facilities (e.g., shopping malls, banks, and petrol stations) in a city according to available trajectories of the potential customers. They may use the TL-Join to find the locations that join with the most trajectories. Such locations have high visibility to trajectories and may be most attractive to customers. These locations may then maximize the commercial value of new facilities. As another example, when events occur (e.g., accidents or terrorist attacks), the police may want to find eyewitnesses of the events. The TL-Join can find such people by matching their trajectories to the events' locations. In addition, we can use the TL-Join to analyze the activities of trajectories. Depending on the points of interest (e.g., restaurants, shopping malls, and sightseeing places) that a trajectory joins with, we can infer activities related to the trajectory (e.g., dinner, shopping, and tourism).

To the best of our knowledge, this is the first trajectory-to-location matching study that takes into account both the spatial and temporal ranges when computing spatial and temporal correlations. We use a linear method [16], [18], [19] to combine the spatial and temporal correlations into a spatiotemporal correlation metric. In contrast, existing studies typically perform (i) the matching solely in the spatial domain [18], [20], [21], [25], [28] or (ii) using point-to-point matching in the spatial domain or the temporal domain [18], [19], [21], [28]. As a result, they may fail to support time-aware applications. For example, they may match a morning trajectory to an evening activity (e.g., drinking at a bar), or they may match a midnight trajectory to a facility open only during the day (e.g., a bank or a shopping mall). Further, the matched pairs cannot guarantee a long-term and continuous correlation between locations and trajectories in the spatial and temporal domains. For example, a trajectory may have a single sample point and a short duration matching to a

- *Shuo Shang is with King Abdullah University of Science and Technology, Saudi Arabia.*
  E-mail: jedi.shang@gmail.com
- *Lisi Chen is with University of Wollongong, Australia.*
  E-mail: LCHEN012@e.ntu.edu.sg
- *Kai Zheng is with University of Electronic Science and Technology of China. Kai Zheng and Zhewei Wei are corresponding authors.*
  E-mail: zhengkai@uestc.edu.cn
- *Christian S. Jensen is with Aalborg University, Denmark.*
  E-mail: csj@cs.aau.dk
- *Zhewei Wei is with Renmin University of China.*
  E-mail: zhewei@ruc.edu.cn
- *Panos Kalnis is with King Abdullah University of Science and Technology, Saudi Arabia.*
  E-mail: panos.kalnis@kaust.edu.sa

1. https://maps.google.com/
2. https://www.mapquest.com/
3. https://www.bikely.com/
4. https://www.gps-waypoints.net
5. https://www.sharemyroutes.com/
6. https://research.microsoft.com/en-us/projects/geolife/
7. https://www.twitter.com/
8. https://www.Facebook.com/
9. https://www.Foursquare.com/

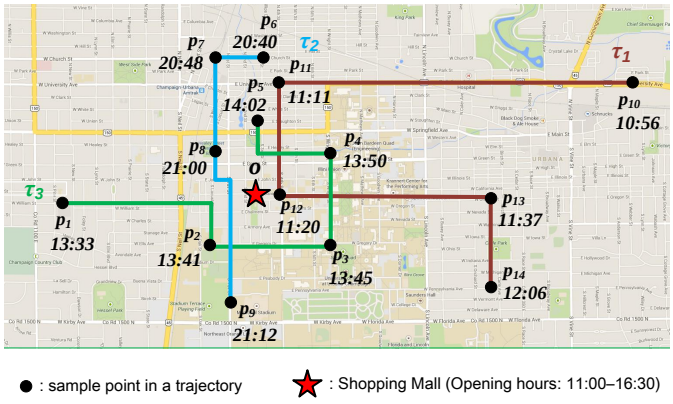


Fig. 1. TL-Join Example

location (its other sample points are too far away from the location); but the result caused by this matching may be of little use in activity analyses because its matching duration is too short to denote a significant relationship. Furthermore, the so-called Semantic Enrichment [12] approach utilizes the stay-time at a location to infer a traveler's activity. This type of matching is not attractive in our intended applications because it is not flexible, i.e., it relies on a fixed visit position (e.g., the intended location) when defining a matching. In contrast, the matching in the TL-Join allows travelers to visit multiple positions close to the intended location within a matching duration. Such flexible matching is appropriate in applications such as location recommendation, event tracking, and activity analyses. The Semantic Enrichment can be viewed as a special case of the matching in the TL-Join.

An example of the TL-Join is shown in Figure 1, where τ_1, τ_2, and τ_3 are trajectories and o is a location (a shopping mall) with opening hours from 11:00 to 16:30. For the matchings that solely consider the spatial domain [18], [20], [21], [25], [28], trajectory-location pairs (τ_1, o), (τ_2, o), and (τ_3, o) are returned because all trajectories are spatially close to o. However, the timestamps of τ_2 do not match the time range (opening hours) of o, so pair (τ_2, o) has little meaning in this scenario. For point-to-point matching [19] in the spatial and temporal domains, pairs (τ_1, o), and (τ_3, o) are returned because (p_{12}, o) and (p_4, o) are matched point pairs, where $p_{12} \in \tau_1$ and $p_4 \in \tau_3$. But there is not a long-term and continuous correlation between p_{12} and o, so this matching result is of little use in applications such as location recommendation, event tracking, and activity analyses. The TL-Join returns the pair (τ_3, o) because it has a long-term and continuous correlation in both the spatial and temporal domains (e.g., matched point pairs (p_2, o), (p_3, o), (p_4, o), and (p_5, o) and a duration of around 20 minutes) and because its spatiotemporal correlation exceeds threshold θ. Notice that the Semantic Enrichment approach [12] considers the stay duration at a location, which means that multiple trajectory sample points are at the corresponding location for the stay duration (the number of sample points depends on the trajectory sampling rate and the stay duration). The TL-Join can support this special case.

The TL-Join is applied in a spatial network because objects move in a spatial network rather than in Euclidean space in many practical scenarios. In a spatial network, network distance is the relevant distance between two objects, and using Euclidean distance [12], [20], [25], [28] may lead to errors.

Table I: Trajectory-to-Location Matchings

Studies	Query Space	Spatiotemporal Matching	Data
RPNN [21]	Network	Spatial only (point-to-point)	1.6 K
ATSQ [28]	Euclidean	Spatial only (point-to-point)	49 K
UOTS [18]	Network	Spatial only (point-to-point)	30 K
PTM [19]	Network	Spatial (point-to-point) and Temporal (point-to-point)	30 K
Semantic Enrichment [12]	Euclidean	Spatial (point-to-point) and Temporal (range)	13 K
VID Join [20], [25]	Euclidean	Spatial only (range)	12 K
TL-Join (our proposal)	Network	Spatial (range) and Temporal (range)	10 M

An overview of a comparison to existing trajectory-location matching studies is shown in Table I. Existing methods [12], [18]–[21], [25], [28] cannot process the TL-Join due to four reasons. (i) Different query types: most trajectory-to-location matching studies [12], [18], [19], [21], [28] are not related to the join operation. For example, RPNN [21] concerns reverse path nearest neighbor querying; ATSQ [28], UOTS [18], and PTM [19] concern trajectory search by locations; and Semantic Enrichment [12] concerns the use of trajectories to infer travelers' activities. Their solutions cannot be used in the TL-Join because the solutions are for different query types. (ii) Different matching functions: existing studies are based on point-to-point matching [18], [19], [21], [28] or spatial-only matching [18], [20], [21], [25], [28], and their solutions are inapplicable to spatiotemporal range matching. (iii) Different query spaces: the VID join [20], [25] is conducted in Euclidean space, and its spatial index and accompanying pruning techniques are not competitive in spatial networks. (iv) Parallel processing requirement: existing centralized trajectory-to-location joins (VID Join) cannot process large trajectory data sets. Based on the experiments reported in the literature [20], [25], the VID join can process at most 12 K trajectories. In contrast, our implementation of the TL-Join can process 10 M trajectories with a reasonable runtime (the PCol solution can process 10 M $\times$ 0.5 M (trajectory, location) pairs with 120 threads in 651 seconds). Table I offers further details on the scale of the data considered, indicating that we consider several orders of magnitude more data than do previous studies.

Next, the algorithm used for computing the TS-Join [16] cannot process the TL-Join because the query arguments are different (two trajectory sets *vs.* a trajectory and a location sets) and because the matching functions are different (point-to-point matching *vs.* range matching). The TL-Join needs its own specific solutions.

We propose two baseline solutions to the TL-Join, called parallel temporal-first search (PTF) and parallel spatial-first

search (PSF). For PTF, we improve the equal-partition grid index in the TS-Join [16], and we propose a new balanced grid index in the temporal domain (i.e., each leaf node has similar numbers of trajectories and locations so that the parallel computation load is balanced). We define spatiotemporal correlation upper and lower bounds to prune the search space, and we perform the refinement of (trajectory, location) pairs from the leaf nodes towards the root. The computations at each index level occur in parallel. The main drawback of PTF is threefold: (i) weak spatial pruning power (temporal driven pruning), (ii) high merging cost (having more leaf nodes enables more parallel processing, but also higher merging cost), and (iii) additional computation cost to acquire network distances when computing spatial correlations.

Next, PSF is based on a divide-and-conquer strategy and performs better than PTF. For each location o, PSF explores the spatial domain to find trajectories with high spatial correlation to o. In the temporal domain, it checks whether the corresponding timestamps are within the time range of o (temporal correlation). We define upper bounds on the spatial correlation to prune the search space. Each trajectory search is independent and is performed in parallel, and the merging cost is independent of the degree of parallelism. The network distances needed for spatial correlation computations can be derived directly during trajectory searches from locations. The limitation of PSF lies in its weak pruning power in the temporal domain.

To process the TL-Join more efficiently, we propose a novel parallel collaborative search (PCol) approach. PCol uses the parallel mechanism of PSF. For each location o, PCol explores the spatial and temporal domains concurrently to find trajectories with high spatiotemporal correlation to o. We define upper bounds on the spatiotemporal correlation and a heuristic scheduling strategy that result in strong pruning power in the two domains.

To sum up, we make the following contributions:

- We propose a new trajectory-to-location join, called TL-Join, targeting applications such as location recommendation, event tracking, and trajectory activity analyses.
- The TL-Join takes both spatial and temporal range matching into account to compute spatiotemporal correlation. No other proposal provides this functionality.
- We develop two baseline algorithms for computing the TL-Join called parallel temporal-first (PTF) search and parallel spatial-first (PSF) search.
- We develop a parallel collaborative algorithm (PCol) with effective pruning techniques and a heuristic scheduling strategy in the spatial and temporal domains.
- We conduct extensive experiments on large trajectory data sets to study the performance of the developed algorithms. We can handle about 3 orders of magnitude more trajectories than the state-of-the-art VID join.

The rest of the paper is organized as follows. Section 2 defines the setting, including spatial networks, trajectories, locations, and the spatiotemporal correlation metrics considered in the paper; it ends by defining the problem. Parallel temporal-first (PTF) search and parallel spatial-first (PSF) search are covered in Sections 3 and 4, while parallel collaborative (PCol) search is covered in Section 5. Experimental results are presented in Section 6. Related work is covered in Section 7, and conclusions and future directions are presented in Section 8.

2 Preliminaries and Problem Definition

2.1 Spatial Networks

A spatial network is modeled as a connected, undirected, and weighted graph $G = (V, E, F, W)$, where V is a vertex set and $E \subseteq \{\{v_i, v_j\} | v_i, v_j \in V \wedge v_i \neq v_j\}$ is an edge set. A vertex $v_i \in V$ represents a road intersection or an end of a road, and an edge $e_k = \{v_i, v_j\} \in E$ represents a road segment that enables travel between vertices v_i and v_j. Function $F : V \cup E \rightarrow Geometries$ maps a vertex to the point location of the corresponding road intersection and maps an edge to a polyline representing the corresponding road segment. Function $W : E \rightarrow R$ assigns a real-valued weight $W(e)$ to an edge e that represents the corresponding road segment's length.

The shortest path between two vertices v_i and v_j is a sequence of edges linking v_i and v_j such that the sum of the edge weights is minimal. Such a path is denoted by $SP(v_i, v_j)$, and its length is denoted by $sd(v_i, v_j)$. Euclidean-space based spatial indices (e.g., the R-tree [13]) and accompanying techniques are ineffective in network environments due to loose lower bounds.

For simplicity, we assume that the data points considered (e.g., trajectory sample points) are located at vertices. It is straightforward to also support data points on edges. Assume a data point p is on an edge e with given network distances to the two end vertices e_a and e_b. Then, a new vertex is created for p with the appropriate geometry, and edge e is replaced by edges (e_a, p) and (p, e_b) with the appropriate weights and geometries.

2.2 Trajectories and Locations

Raw trajectory samples obtained from GPS devices are typically of the form (longitude, latitude, time), and trajectory sample points are captured periodically at some sampling rate. We assume that all sample points have already been map matched onto the spatial network using a map-matching algorithm (e.g., [4], [24]) and that an object always follows the shortest path when moving between two adjacent sample points p_a and p_b. A trajectory is defined as follows.

Definition: Trajectory
A trajectory τ of a moving object is a finite, time-ordered sequence $\langle v_1, v_2, ..., v_n \rangle$, where $v_i = (p_i, t_i)$, $i \in [1, n]$, with p_i being a sample point (equal to some vertex in $G.V$) and t_i being a timestamp.

Assuming that $\tau.sr$ is the sampling rate of trajectory τ, we have that $t_{i+1} - t_i = \tau.sr$, $i \in [1, n-1]$.

The above modeling of spatial networks and trajectories aligns with previous studies [16], [19].

Definition: Location
A location o contains a spatial attribute $o.p$ and a temporal

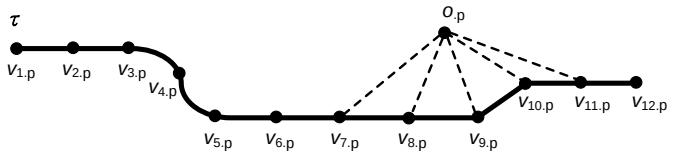


Fig. 2. Spatiotemporal Correlation

attribute $o.R$, where $o.p$ is a vertex in $G.V$ and $o.R$ is a time range. Time range $o.R$ describes the valid duration of o (e.g., opening hours of facilities, or time ranges of events).

The values of timestamps and time ranges are set to be within the range of 24 hours, and the date is not taken into account because in many practical scenarios, like urban transportation, movements are often studied within the range of a day [16], [19].

2.3 Spatiotemporal Correlation

Given a location o and a trajectory τ, K_τ is the set of trajectory sample points in τ that are spatially closest to $o.p$: $\forall v \in K_\tau(\forall v' \in \tau \setminus K_\tau(sd(o.p, v.p) \leq sd(o.p, v'.p)))$. The cardinality of K_τ is set as follows.

$$k_\tau = |K_\tau| = \left\lfloor \frac{d_c}{\tau.sr} \right\rfloor + 1 \tag{1}$$

Here, d_c controls the coupling duration (to describe the term of correlations) between o and τ. Its value is user-defined. We assume that trajectories are sampled uniformly. As different locations may have different coupling duration d_c and different trajectories may have different sampling rates $\tau.sr$, the value of k_τ may be different for different trajectories. The following algorithms support this.

The spatial correlation $\mathrm{C_S}(o, \tau)$ and the temporal correlation $\mathrm{C_T}(o, \tau)$ between o and τ are defined as follows.

$$\mathrm{C_S}(o, \tau) = \frac{\sum_{v_i \in K_\tau} e^{-sd(o.p, v_i.p)}}{k_\tau} \tag{2}$$

$$\mathrm{C_T}(o, \tau) = \frac{|\{v_j.t | v_j \in K_\tau \wedge v_j.t \in o.R\}|}{k_\tau} \tag{3}$$

In the spatial domain, we count the sum of the spatial distances between location o and trajectory sample points in K_τ, while in the temporal domain, we check the validity of the sample points in K_τ by matching their timestamps to the time range $o.R$.

An example that illustrates these definitions is shown in Figure 2, where o is an object and $\tau = \langle v_1, v_2, ..., v_{12} \rangle$ is a trajectory. The coupling duration d_c is 8 minutes, and the sampling rate of τ is 2 minutes, so $k_\tau = \lfloor \frac{8\ minutes}{2\ minutes} \rfloor + 1 = 4 + 1 = 5$. Points $v_7.p$, $v_8.p$,...,$v_{11}.p$ are the top-5 trajectory sample points spatially closest to $o.p$, so $K_\tau = \{v_7.p, v_8.p, ..., v_{11}.p\}$. The value of $\mathrm{C_S}(o, \tau)$ is computed by substituting K_τ into Equation 2. Next, assuming that $v_7.t$=12:50, $v_8.t$=12:52, $v_9.t$=12:54, $v_{10}.t$=12:56, $v_{11}.t$=12:58, and $o.R$ = [12:55, 13:00], we have that $v_7.t \notin o.R$, $v_8.t \notin o.R$, $v_9.t \notin o.R$, $v_{10}.t \in o.R$, and $v_{11}.t \in o.R$. According to Equation 3, $\mathrm{C_T}(o, \tau) = |\{v_{10}.t, v_{11}.t\}|/5 = 2/5 = 0.4$.

A List of Notions

Notion	Description
$G.V$	the set of vertices in graph G
$G.E$	the set of edges in graph G
$sd(p_i, p_j)$	shortest path distance between vertices p_i and p_j
$\tau.sr$	sampling rate of trajectory τ
$o.R$	time range of location o
d_c	coupling duration
K_τ	the set of top-k_τ sample points in τ that are spatially closest to location o
$\mathrm{C_S}$, $\mathrm{C_T}$, $\mathrm{C_{ST}}$	spatial, temporal, and spatiotemporal correlation
λ	the relative importance of the spatial and temporal correlations
UB, LB	global upper and lower bounds

The spatial and temporal correlations of τ are both in the range [0, 1]. We use a linear method [16], [19] to combine the spatial and temporal correlations (Equations 2 and 3), and the spatiotemporal correlation is defined as follows.

$$\mathrm{C_{ST}}(o, \tau) = \lambda \cdot \mathrm{C_S}(o, \tau) + (1 - \lambda) \cdot \mathrm{C_T}(o, \tau) \tag{4}$$

Here, parameter $\lambda \in [0, 1]$ controls the relative importance of the spatial and temporal correlations. The value of λ can be adjusted at query time.

2.4 Problem Definition

Given a set T of trajectories, a set O of locations, and a threshold θ, the trajectory-to-location join (TL-Join) returns the set A of all (trajectory, location) pairs from the two sets whose spatiotemporal correlations are at least θ, i.e., $\forall(\tau_i, o_j) \in A\ (\mathrm{C_{ST}}(\tau_i, o_j) \geq \theta) \wedge \forall(\tau_i', o_j') \in ((T \times O) \setminus A)(\mathrm{C_{ST}}(\tau_i', o_j') < \theta)$.

3 Parallel Temporal-First Search

3.1 Basic Idea

Parallel temporal-first (PTF) search is a baseline approach to TL-Join processing. We improve the equal-partition grid index used in the TS-Join [16], and we propose a new balanced hierarchical grid index in the temporal domain (Section 3.2). We also define upper and lower bounds to prune the search space in the spatial and temporal domains. PTF refines the (trajectory, location) pairs in the same leaf node and merges the results from the leaf nodes towards the root. The join result is then obtained from the root. The computations at the nodes at the same level occur in parallel (Section 3.3). The pseudocode of PTF and its time complexity are given in Section 3.4.

3.2 Balanced Grid Index

In the TS-Join [16], the temporal domain is partitioned into m equal-sized time slots, each of which is assigned to a leaf node. The drawback of this approach is that the distributions of trajectories and locations are imbalanced, and different leaf nodes may have quite different numbers of trajectories (e.g., peak hours may have more, off-peak hours may have fewer, and midnight may have none). Such imbalance yields poor performance in parallel processing. To address this issue, we propose a new balanced grid index in the temporal domain

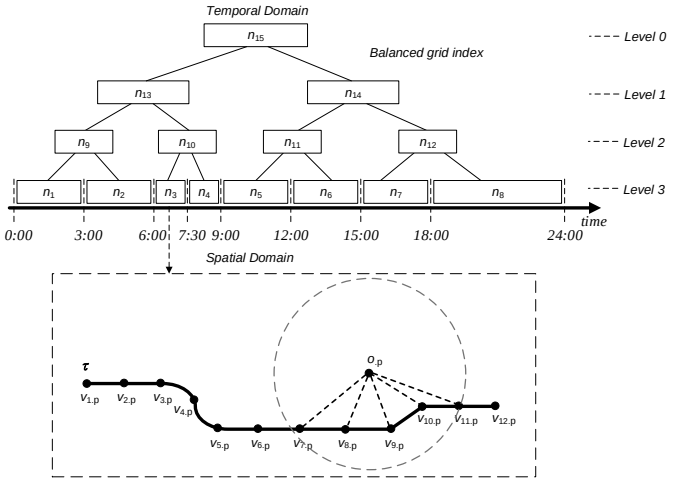


Fig. 3. Example of PTF

for PTF. Here, each leaf node n has a matching-times upper bound $M \geq |n_\tau| \times |n_o|$, where n_τ and n_o are the sets of trajectories and locations contained in n. The optimal value of M that achieves the highest performance is determined through extensive experiments. Notice that the balanced grid index is a temporal index, which indexes the time ranges of trajectories and locations. Other trajectory indexes (e.g., [5], [9], [14]) are spatial index, and they are not suitable for this scenario.

The balanced grid index is constructed as follows. Given a value of M and a slot s = [0, 24:00], we recursively partition s into two equal-sized nodes if $|s_\tau| \times |s_o| > M$, where s_τ and s_o are sets of trajectories and locations in slot s. For example, given a trajectory $\tau = \langle v_1, v_2, ..., v_i \rangle$, its temporal range $\mathrm{range}(\tau) = [v_1.t, v_i.t]$. If $\mathrm{range}(\tau) \subseteq \mathrm{range}(s)$, τ is contained in s. Similarly, given a location o, if $o.R \subseteq \mathrm{range}(s)$, o is contained in s. For example, given $\mathrm{range}(s)$ = [9:00, 12:00], $\mathrm{range}(\tau)$ = [10:00, 11:00], and $o.R$ = [9:30, 11:30], τ and o are contained in s.

Once the partitioning terminates, each slot corresponds to a leaf node. We build a tree structure bottom-up. Assume that there are m nodes at the leaf level. Then we build $\lceil \frac{m}{2} \rceil$ parent nodes. We do this recursively until there is one parent, which is the root. The height of the tree is $\lceil log(m) \rceil + 1$. An example is shown in Figure 3, where n_1, n_2,...,n_8 are leaf nodes and n_{15} is the root. Each trajectory τ and each location o are stored in the lowest node n that fully covers its temporal range, i.e., $\mathrm{range}(\tau) \subseteq \mathrm{range}(n)$ and $o.R \subseteq \mathrm{range}(n)$ and $\mathrm{range}(\tau)$ and $o.R$ are not contained in the range of any child node of n. For example, given $o'.R$ = [9:30, 17:30], o' is stored in n_{14} ($\mathrm{range}(n_{14})$ = [9:00, 24:00]) because $o'.R \subseteq \mathrm{range}(n_{14})$ and $o'.R \nsubseteq \mathrm{range}(n_{11})$ and $o'.R \nsubseteq \mathrm{range}(n_{12})$ (n_{11} and n_{12} are child nodes of n_{14}).

3.3 Filtering, Refinement, and Merging

In the example in Figure 3, a trajectory τ and a location o are stored in node n_3. As they are temporally close to each other, we estimate the upper bound on their temporal correlation $\mathrm{C_T}(o, \tau)$ (cf. Equation 3) as follows.

$$|\{v_j | v_j \in K_\tau \wedge v_j.t \in o.R\}| \leq k_\tau$$
$$\Rightarrow \mathrm{C_T}(o, \tau).ub = 1 \geq \mathrm{C_T}(o, \tau) \quad (5)$$

By substituting Equation 5 into Equation 4, we have that

$$\mathrm{C_{ST}}(o, \tau) = \lambda \cdot \mathrm{C_S}(o, \tau) + (1 - \lambda) \cdot \mathrm{C_T}(o, \tau) \geq \theta$$
$$\Rightarrow \mathrm{C_S}(o, \tau) \geq \frac{\theta - (1 - \lambda) \cdot \mathrm{C_T}(\tau_1, \tau_2).ub}{\lambda} = \frac{\theta - 1 + \lambda}{\lambda}$$

For each "qualified" (trajectory, location) pair (o, τ) (i.e., $\mathrm{C_{ST}}(o, \tau) \geq \theta$), its spatial correlation exceeds the value of $\frac{\theta - 1 + \lambda}{\lambda}$. We define a global lower bound LB_S of the spatial correlation between (trajectory, location) pairs in the same leaf node as follows.

$$LB_S = \frac{\theta - 1 + \lambda}{\lambda} \quad (6)$$

We use network expansion to compute the spatial correlation $\mathrm{C_S}(o, \tau)$ (Equation 2). The network expansion is performed from location o using Dijkstra's algorithm [10]. Dijkstra's algorithm always selects the vertex with the minimum distance label for expansion. Hence, the first k_τ sample points in τ scanned by the expansion are just the top-k_τ sample points closest to o. For example, in Figure 3, assuming $k_\tau = 5$ and $v_7.p, v_8.p, ..., v_{11}.p$ are top-5 first scanned sample points in τ. According to Equation 2, $\mathrm{C_S}(o, \tau) = \frac{1}{5}(e^{-d(v_7.p, o.p)} + e^{-d(v_8.p, o.p)} + ... + e^{-d(v_{11}.p, o.p)}$. If $\mathrm{C_S}(\tau, o) < LB_S$, then $\mathrm{C_{ST}}(o, \tau) < \theta$, and the (trajectory, location) pair (o, τ) can be pruned safely. Otherwise, we compute the exact spatiotemporal correlation $\mathrm{C_{ST}}(o, \tau)$ (Equation 4) and compare to θ to check the pair's validity. The computations in different leaf nodes are independent and occur in parallel.

Having computed the spatiotemporal correlations of the (trajectory, location) pairs in the leaf nodes, we merge the results from the leaf level to the root level (bottom-up). At each level, when two nodes n and n' have the same parent n'', we merge their results and assign this to the parent (e.g., merge n_3, n_4, and n_{10} to obtain the result for n_{10} in Figure 3). In addition to these qualified results ($\mathrm{C_{ST}}(o, \tau) \geq \theta$), we also need to consider the (trajectory, location) pairs (o, τ) in the following three cases: (i) one item is stored in n or n' and another item is stored in n'' (e.g., $\mathrm{range}(\tau) \subseteq \mathrm{range}(n)$ and $o.R \subseteq \mathrm{range}(n'')$); (ii) two items are stored in n'' (e.g., $\mathrm{range}(\tau) \subseteq \mathrm{range}(n'')$ and $o.R \subseteq \mathrm{range}(n'')$); (iii) one item is stored in n and another item is stored in n' (e.g., $\mathrm{range}(\tau) \subseteq \mathrm{range}(n)$ and $o.R \subseteq \mathrm{range}(n')$).

For the first and the second cases, we use the same lower and upper bounds (Equations 5 and 6) and pruning techniques as we use for the (trajectory, location) pairs in the same node. The qualified pairs are stored in n''. For the third case, as trajectory τ and location o are stored in different nodes, we have that $C_T(\tau, o) = 0$. By substituting this into Equation 4, we have that

$$C_{ST}(\tau, o) \geq \theta \Leftrightarrow C_S(\tau, o) \geq \frac{\theta}{\lambda} \quad (7)$$

As the value of $C_S(\tau, o)$ is in the range [0, 1], if $\theta > \lambda$, (trajectory, location) pair (τ, o) is pruned directly. Otherwise,

we compute the spatiotemporal correlation $C_{ST}(\tau, o)$ and compare to θ to check the pair's validity.

The merging processes of adjacent node pairs (e.g., merge n_1 and n_2 to n_9, n_3 and n_4 to n_{10}) at the same level of the tree are independent. Thus they again occur in parallel. Having merged the computation results from the leaf nodes all the way to the root node, the join result in [0:00, 24:00] is found.

3.4 Algorithm and Time Complexity

Algorithm 1: PTF Search

Data: a balanced grid index tree T_r, a trajectory set T, a location set O, and a threshold θ
Result: $\{(\tau, o)|\mathrm{C_{ST}}(\tau, o) \geq \theta, \forall \tau \in T, \forall o \in O\ \}$

```
1  h ← T_r.hight − 1;
2  compute LB_S;
3  for each leaf node n in T_g do
4      for each (trajectory, location) pair (τ, o) in n do
5          compute C_S(τ, o);
6          if C_S(τ, o) < LB_S then
7              prune (τ, o);
8          compute C_ST(τ, o);
9          if C_ST(τ, o) ≥ θ then
10             P_n.add(τ, o);
11 while true do
12     if n_a, n_b ∈ level h, n_a.parent = n_b.parent = n_c
       then
13         merge n_a, n_b, and n_c;
14         compute and store qualified (trajectory, location)
           pairs in P_{n_c};
15     if h = 1 then
16         return P_{n_c};
17     h ← h − 1;
```

The pseudocode of PTF is shown in Algorithm 1. The computation is bottom-up, and h is the current level of computation. Initially, we compute the global spatial lower bound LB_S (Equation 6) for leaf nodes (lines 1–2). For each (trajectory, location) pair (τ, o) in n (i.e., $\mathrm{range}(\tau) \subseteq \mathrm{range}(n)$ and $o.R \subseteq \mathrm{range}(n)$), we compute its spatial correlation $\mathrm{C_S}(\tau, o)$ (Equation 2), and if $\mathrm{C_S}(\tau, o)$ is less than LB_S, pair (τ, o) is pruned (lines 3–7). Otherwise, we compute the exact spatiotemporal correlation $\mathrm{C_{ST}}(\tau, o)$ (Equation 4), and if it is no less than θ, we store (τ, o) in P_n (lines 8–10). Having refined all leaf nodes, we merge the results from the leaf level towards the root. If two nodes n_a and n_b are at the same level and they have the same parent node n_c, we merge the results for n_a, n_b, and n_c (e.g., n_1, n_2, and n_9 in Figure 3) and store the qualified (trajectory, location) pairs in P_{n_c} (lines 11–14). If $h = 1$, the root node n_c is reached, and all (trajectory, location) pairs stored in P_{n_c} are returned. Otherwise, we repeat the procedure for the next level of the tree (line 15–17).

Let $|T|$ and $|O|$ denote the cardinalities of trajectory set T and location set O. We use $|V|$ and $|E|$ to denote the numbers of vertices and edges in G. Then $O(|V| \log |V| + |E|)$ is the time complexity of computing the spatial correlation between a trajectory and a location by using Dijkstra's algorithm. PTF follows the filter-and-refine paradigm, and the time complexity of the filtering phase is $O((|V| \log |V| + |E|)|T||O|)$.

The time complexity to verify candidates by computing their exact spatiotemporal correlations is $O(k_\tau|C|)$ (the spatial correlations are computed in the filtering phase, so in the refinement phase we only need to compute the temporal correlations), where $|C|$ is the cardinality of the candidate set and $C \subseteq P \times O$. The total time complexity is $O((|V| \log |V| + |E|)|T||O|) + k_\tau|C|) = O((|V| \log |V| + |E|)|T||O|)$, which does not depend on the candidate set size.

The computations for nodes at the same level of the tree occur in parallel. If we have multiple cores and threads (each leaf node corresponds to a thread), it is possible to accelerate the computation at the leaf level by generating many leaf nodes and processing them in parallel. However, more leaf nodes also leads to more tree levels (m is the number of leaf number, and the height of the tree is $\lceil log(m) \rceil + 1$), which increases the merging cost.

4 Parallel Spatial-First Search

4.1 Basic Idea

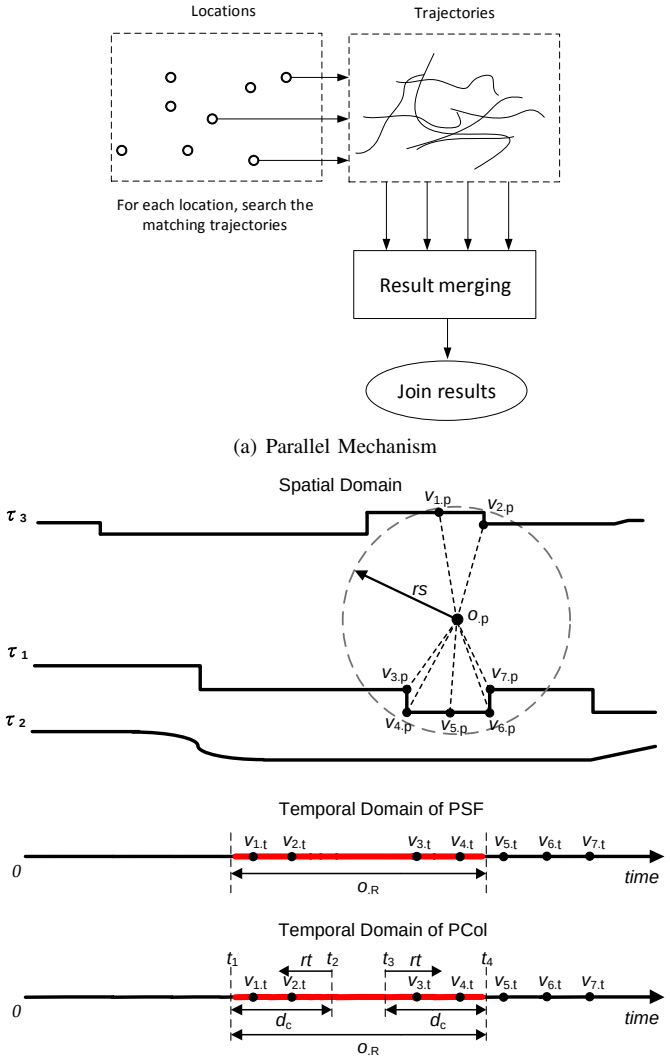
PTF has three weaknesses. (i) Weak spatial pruning power: the pruning is driven by the temporal domain so it has low effectiveness in the spatial domain. (ii) High merging cost: more leaf nodes (each leaf node corresponds to a thread) lead to a higher merging costs, which decreases performance. (iii) Additional network distance computations are needed to compute the spatial correlations (Equation 2), which again yields poor performance.

Parallel spatial first (PSF) search is another baseline for TL-Join computation. Its parallel mechanism is shown in Figure 4(a). For each location $o \in O$, we search the trajectories with high spatiotemporal correlations to o. The trajectory-search processes at different locations are performed in parallel. In the spatial domain, we use network expansion [10] to explore the spatial network and to find trajectories spatially close to o (spatial correlation). In the temporal domain, we check whether the corresponding timestamps are within the time range of o (temporal correlation). Upper and lower bounds on the spatiotemporal correlations are defined to prune the search space. By merging the search results from each location, the solution of the TL-join is found. Compared to PTF, PSF has two advantages. First, its result merging cost is independent of the degree of parallelism. We can simply combine the trajectory-search results of all locations to get the solution. Second, the network distances for the spatial correlation computation can be acquired during the trajectory-search processes. PSF has better time complexity than PTF.

4.2 Filtering, Refinement, and Merging

An example of PSF is given in Figure 4(b), where o is a location and $o.R$ is its time range; τ_1, τ_2, and τ_3 are trajectories; $v_1.p$ and $v_2.p$ are the top-2 vertices in τ_3 spatially

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2854705, IEEE Transactions on Knowledge and Data Engineering



(a) Parallel Mechanism

(b) Trajectory Search

Fig. 4. Examples of PSF and PCol

closest to o; $v_3.p$, $v_4.p$,..., $v_7.p$ are the top-5 vertices in τ_1 spatially closest to o, and $v_1.t$,...,$v_7.t$ are the corresponding timestamps. Assuming $k_{\tau_1} = k_{\tau_2} = k_{\tau_3} = 5$.

In the spatial domain, network expansion is performed from o according to Dijkstra's algorithm [10]. The explored space is a circular region (o, rs) with center o and radius rs. As Dijsktra's algorithm always selects the vertex with the minimum distance label for expansion, the top-k first scanned vertices in τ are the top-k vertices spatially closest to o. For example, in Figure 4(b), $v_1.p$ and $v_2.p$ are the top-2 first scanned vertices in τ_3, and $v_3.p$, $v_4.p$,..., $v_7.p$ are the top-5 first scanned vertices in τ_1.

Assuming a trajectory τ has $\tau.k$ vertices that have been scanned by the expansion from o. If $\tau.k \geq k_\tau$, trajectory τ is called "fully scanned" (e.g., τ_1 in Figure 4(b)). If $k_\tau > \tau.k > 0$, τ is called "partly scanned" (e.g., τ_3 in Figure 4(b)). If $\tau.k = 0$, τ is called "unscanned" (e.g., τ_2 in Figure 4(b)).

For a partly scanned trajectory τ', we estimate an upper bound on its spatial correlation as follows. Assuming that $v_j.p \in \tau'$ is an unscanned vertex in the spatial domain, we have:

$$rs < sd(o.p, v_j.p) \Rightarrow e^{-sd(o.p,v_j.p)} < e^{-rs}$$

By substitution into Equation 2, the spatial correlation upper bound $C_S(\tau, o).ub$ is defined as follows.

$$C_S(\tau, o).ub = \frac{\sum_{v_i.p \in V_s} e^{-sd(o.p,v_i.p)} + \sum_{v_j.p \in V_u} e^{-rs}}{k_\tau} \quad (8)$$

Here, V_s is a set of scanned vertices in τ ($V_s \subset K_\tau$), and V_u is the set of unscanned top-k_τ vertices in τ ($V_s \cup V_u = K_\tau$ and $|V_s \cup V_u| = k_\tau$). Among all partly scanned trajectories in the spatial domain, we define a global upper bound on spatial correlation as

$$UB_S = \max_{\tau \in T_p} \{C_S(\tau, o).ub\}, \quad (9)$$

where T_p is a set of partly scanned trajectories in the spatial domain, and the value of UB_S changes dynamically during the query processing.

Filter-and-Refine: if $UB_S < \theta$, we prune all partly scanned and unscanned trajectories. For fully scanned trajectories, we compute the exact spatial (Equation 2) and temporal (Equation 3) correlations. The spatiotemporal correlation $C_{ST}(\tau, o)$ is derived by combining them (Equation 4). For example, in Figure 4(b), τ_1 is fully scanned, so we compute $C_S(\tau_1, o.p) = \frac{1}{5}(e^{-d(v_3.p,o.p)} + e^{-d(v_4.p,o.p)} + e^{-d(v_5.p,o.p)} + e^{-d(v_6.p,o.p)} + e^{-d(v_7.p,o.p)})$, $C_T(\tau_1, o) = \frac{|\{v_3.t, v_4.t\}|}{5} = \frac{2}{5} = 0.4$, and $C_{ST}(\tau_1, o.p) = \lambda \cdot C_S(\tau_1, o.p) + (1 - \lambda) \cdot C_T(\tau_1, o)$. If the spatiotemporal correlation $C_{ST}(\tau, o)$ does not exceed θ, we prune trajectory τ. Otherwise, (trajectory, location) pair (τ, o) is stored in M_o (the set of matched (trajectory, location) pairs of o). By combining M_o of all objects in O, the result $\bigcup_{o \in O} M_o$ of the TL-Join is found.

Notice that we do not maintain upper bounds on the unscanned trajectories to reduce the computation and storage cost. Given a partly scanned trajectory τ and an unscanned trajectory τ' (e.g., τ_2 in Figure 4(b)), according to Equations 8 and 9, we have:

$$C_S(\tau', o).ub = \frac{\sum_{v_j.p \in V_u} e^{-rs}}{k'_\tau} = e^{-rs} < C_S(\tau, o).ub \leq UB_S \quad (10)$$

Here, $V_u = K_\tau$ and $|V_u| = k'_\tau$. If $UB_S < \theta$, we have that $C_S(\tau', o).ub < \theta$. So the unscanned trajectories can be pruned safely, and it is not necessary to maintain spatial upper bounds.

4.3 Algorithm and Time Complexity

PSF adopts a divide-and-conquer strategy. For each location o in set O, we retrieve the trajectories with high spatiotemporal correlation to o. The search processes for different locations are independent so they are performed in parallel. Unlike for PTF, PSF has a constant merging cost (its merging cost is independent of the degree of parallelism, and we simply combine the search result of each location to achieve join result). The pseudocode of PSF is shown in Algorithm 2.

Initially, for each location $o \in O$, the set of its matched (trajectory, location) pairs M_o is set to $\emptyset$. The global spatial

upper bound UB_S is set to 0. For each trajectory $\tau \in T$, the number of its scanned vertices $\tau.k$ is set to 0. We perform network expansion from each location o to explore the spatial network (lines 1–4). For each newly scanned vertex p, all trajectories passing P have one more scanned vertex (lines 5–7). If the number of scanned vertices of τ is equal to k_τ (τ is fully scanned), we compute its spatiotemporal correlation $C_{ST}(\tau, o)$ (Equation 4). If the value of $C_{ST}(\tau, o)$ exceeds that of θ, we store (trajectory, location) pair (τ, o) in M_o. Then, we remove τ from the partly scanned trajectory set T_p and update the value of UB_S (lines 8–13). If τ is partly scanned ($0 < \tau.k < k_\tau$), we compute its spatial correlation upper bound $\mathrm{C_S}(\tau, o).ub$, and we update the value of the global spatial upper bound UB_S (lines 14–17). If the value of UB_S does not exceed that of θ, the expansion from o terminates (lines 18–20). Having searched all locations, we combine their results and get the result $\bigcup_{o \in O} M_o$ of the TL-Join (line 21).

Algorithm 2: PSF Search

```
   Data: a set O of locations, a set T of trajectories, and a
         threshold θ
   Result: ⋃_{o∈O} M_o
1  ∀o ∈ O(M_o ← ∅);
2  for each location o in O do
3      UB_S ← 0;
4      ∀τ ∈ T(τ.k ← 0);
5      p ←expand(o);
6      for each trajectory τ passing p do
7          τ.k ← τ.k + 1;
8          if τ.k = k_τ then
9              compute C_ST(τ, o);
10             if C_ST(τ, o) ≥ θ then
11                 M_o.add(τ, o);
12             T_p.remove(τ);
13             update UB_S;
14         if 0 < τ.k < k_τ then
15             update C_S(τ, o).ub;
16             if C_S(τ, o).ub > UB_S then
17                 UB_S ← C_S(τ, o).ub;
18         if UB_S < θ then
19             store M_o;
20             break;
21 return ⋃_{o∈O} M_o;
```

Let $|O|$ denote the cardinality of location set O and let T_θ denote the scanned trajectory set for the search process from each location, which includes the partly and fully scanned trajectories ($T_\theta = T_p \cup T_f$). According to Equations 8 and 9, the maximum spatial expansion radiuses rs is inversely proportional to θ. Assuming the trajectories are uniformly distributed in the spatial domain, it follows that $|T_\theta|$ is inversely proportional to θ. Thus, $|T_\theta|$ is sensitive to the value of threshold θ and the pruning effectiveness. We use $|V|$ and $|E|$ to denote the numbers of vertices and edges in G. Then $O(|V| \log |V| + |E|)$ is the time complexity of network expansion using Dijkstra's algorithm. The time complexity of PSF is $O(|T_\theta||O|(|V|log|V| + |E|))$. If the value of θ is sufficiently large, the time complexity is close to $O(|O|(|V|log|V| + |E|))$.

5 Parallel Collaborative Search

5.1 Basic Idea

The main weakness of PSF lies in its weak temporal pruning power since its pruning is driven by the spatial domain. To overcome that weakness and to process the TL-Join more efficiently, we propose a parallel collaborative (PCol) search algorithm that improves PSF. In contrast to PSF, PCol performs trajectory search in the spatial and temporal domains concurrently. An upper bound on the spatiotemporal correlation and a heuristic search strategy are proposed to prune the search space. PCol follows the same parallel mechanism as PSF (cf. Figure 4(a)). Compared to PSF, PCol has stronger pruning power, which should translate into higher performance.

5.2 Upper Bound

In the spatial domain, PCol, like PSF, adopts network expansion [10] to explore the spatial network and to find trajectories with high spatial correlation to the query location o. In the temporal domain, we partition time range $o.R$ into three parts (if $|o.R| > 2d_c$). An example is shown in Figure 4(b), where $|\mathrm{range}(t_1, t_2)| = |\mathrm{range}(t_3, t_4)| = d_c$ and d_c is the coupling duration between τ and o. Initially we search the trajectory timestamps in $\mathrm{range}(t_2, t_3)$, and then we expand the search from t_2 and t_3 concurrently towards the boundaries of $o.R$, and rt is the radius of the search space. If $|o.R| \leq 2d_c$, we only partition $o.R$ into two parts from the middle point (i.e., merging t_2 and t_3 in Figure 4(b) to the middle point), and then we expand the search from the middle point towards the boundaries.

We estimate the upper bound on the temporal correlation of an unscanned trajectory τ as follows.

$$|\mathrm{range}(t_1, t_2 - rt)| = |\mathrm{range}(t_3 + rt, t_4)| = d_c - rt$$
$$\Rightarrow C_T(\tau, o).ub = \frac{\lfloor \frac{|\mathrm{range}(t_1, t_2 - rt)|}{\tau.sr} \rfloor + 1}{k_\tau} = \frac{\lfloor \frac{d_c - rt}{\tau.sr} \rfloor + 1}{k_\tau} \quad (11)$$

Here, $\tau.sr$ is the sampling rate of trajectory τ, and $\mathrm{range}(t_1, t_2 - rt)$ and $\mathrm{range}(t_3 + rt, t_4)$ are the unscanned spaces in $o.R$. Because trajectories are sampled continuously and uniformly and because $\mathrm{range}(t_2 - rt, t_3 + rt)$ has been scanned in the current step, it is impossible for an unscanned trajectory to appear in both $\mathrm{range}(t_1, t_2 - rt)$ and $\mathrm{range}(t_3 + rt, t_4)$. Notice that for trajectories with non-uniform sampling rate, we simply need to count the number n of sample points in the corresponding time range, or to use the minimum sampling rate of τ to compute the bounds.

By combining the upper bounds on the spatial (Equation 10) and temporal (Equation 11) correlation according to Equation 4, we obtain an upper bound $C_{ST}(\tau.o).ub$ of the spatiotemporal correlation. The value of $C_{ST}(\tau.o).ub$ is used as a global upper bound UB for all unscanned trajectories

in both domains, and it changes dynamically during query processing.

$$C_{ST}(\tau.o).ub = \lambda \cdot C_S(\tau.o).ub + (1-\lambda) \cdot C_T(\tau.o).ub$$

$$\Rightarrow UB = C_{ST}(\tau.o).ub = \lambda \cdot e^{-rs} + (1-\lambda) \cdot \frac{\lfloor \frac{d_c - rt}{\tau.sr} \rfloor + 1}{k_\tau} \quad (12)$$

5.3 Filtering and Refinement

If the value of spatiotemporal upper bound $C_{ST}(\tau.o).ub$ is less than θ, the search in the spatial and temporal domains terminate and all unscanned trajectories are pruned. Then we refine the fully and partly scanned trajectories in the two domains. If a trajectory τ is fully scanned in the spatial domain, we compute its exact spatial, temporal, and spatiotemporal correlations according to Equations 2, 3, and 4. If $C_{ST}(\tau, o) \geq \theta$, we store (trajectory, location) pair (τ, o) in the set of the matched pairs of o. Otherwise, trajectory τ is pruned.

If a trajectory $\tau' = \langle v_1, v_2, ..., v_n \rangle$ is partly scanned in the spatial domain and is unscanned in the temporal domain, we estimate the temporal correlation upper bound $C_T(\tau', o).ub$ as follows.

$$C_T(\tau', o).ub = \frac{1}{k'_\tau} \Big(\left\lfloor \frac{|\text{range}(v_1.t, t_n.t) \cap \text{range}(t_1.t, t_2.t - rt)|}{\tau'.sr} \right\rfloor + \left\lfloor \frac{|\text{range}(v_1.t, t_n.t) \cap \text{range}(t_3.t + rt, t_4.t)|}{\tau'.sr} \right\rfloor + 1 \Big) \quad (13)$$

If a trajectory τ' is scanned in the temporal domain, its temporal correlation upper bound is defined as follows.

$$C_T(\tau', o).ub = \frac{1}{k'_\tau} \Big(\left\lfloor \frac{|\text{range}(v_1.t, t_n.t) \cap o.R|}{\tau'.sr} \right\rfloor + 1 \Big) \quad (14)$$

By combining the spatial correlation upper bound (Equation 8) and temporal correlation upper bound (Equations 13 and 14) according to Equation 4, we obtain a spatiotemporal correlation upper bound $C_{ST}(\tau', o).ub$ as follows.

$$C_{ST}(\tau', o).ub = \begin{cases} \lambda \cdot C_S(\tau', o).ub \\ \quad +(1-\lambda) \cdot C_T(\tau', o).ub & \text{if Case 1} \\ (1-\lambda) \cdot C_T(\tau', o).ub & \text{if Case 2} \end{cases} \quad (15)$$

Case 1: τ' is partly scanned in the spatial domain.
Case 2: τ' is unscanned in the spatial domain.

If the value of $C_{ST}(\tau', o).ub$ is less than that of θ, we prune trajectory τ'. Otherwise, we refine the trajectory in the spatial domain until it is fully scanned. Then we compute its exact spatiotemporal correlation and compare to θ.

5.4 Heuristic Scheduling

We propose a heuristic method to schedule the two query sources in the spatial and temporal domains (i.e., expansion center $o.p$ in the spatial domain, and expansion centers t_2 and t_3 in the temporal domain). Our target is to let more trajectories be scanned in the both domains, which is helpful to (i) reduce the number of scanned trajectories to be refined and to (ii) improve the pruning power of Equation 16.

For example, T_S is the set of scanned trajectories in the spatial domain, and T_T is the set of scanned trajectories in the temporal domain. We refine $|T_S \cup T_T|$ trajectories in total. If we are able to increase the intersection between T_S and T_T (the trajectories that have been scanned in both domains), we can reduce the total number of trajectories to be refined and can improve the query efficiency correspondingly. Moreover, we can use the spatial and temporal correlation upper bounds (Equation 16) of the trajectories in $T_S \cap T_T$ to prune the search space, which yields better pruning than using only the spatial or the temporal upper bound (if trajectories only have been scanned in one domain).

Priority labels of the query sources in the two domains are then defined as follows. At each time, we only search the top-ranked query source (the query source has a larger value of its label) until a new query source takes its place.

$$q.l = \begin{cases} \lambda \cdot |(T_S \cup T_T) \setminus T_S| & \text{if Case 3} \\ (1-\lambda) \cdot |(T_S \cup T_T) \setminus T_T| & \text{if Case 4} \end{cases} \quad (16)$$

Case 3: q is in the spatial domain ($q = o.p$).
Case 4: q is in the temporal domain (q is for t_2 and t_3).
Here λ and $(1-\lambda)$ control the relative importance of the spatial and the temporal domains (Equation 4).

5.5 Algorithm and Time Complexity

The PCol search procedure is detailed in Algorithm 3. The query arguments include a location o, a trajectory τ, and a threshold θ. The query result is returned in $\bigcup_{o \in O} M_o$. Initially, UB and the priority labels are set to 0 and M_o is set to $\emptyset$. If the value of $2d_c$ is less than that of $|o.R|$, we scan the timestamps in $\text{range}(t_2, t_3)$ (cf. Figure 4(b))(lines 1–5). We select the top-ranked query source q from heap H as the current-search query source, and we expand the search from q. We update the value of UB (Equation 12). If the value of UB is less than θ, we prune all unscanned trajectories in the two domains (lines 6–11). Then we refine all scanned trajectories in the two domains. If a trajectory τ is fully scanned in the spatial domain, we compute its exact spatiotemporal correlation $C_{ST}(\tau.o)$ (Equation 4) and compare it to θ. If $C_{ST}(\tau.o) \geq \theta$, (trajectory, location) pair (τ, o) is added in M_o. Otherwise, τ is pruned (lines 12–17). If a trajectory τ' is partly scanned, we compute its spatiotemporal upper bound $C_{ST}(\tau'.o).ub$ (Equation 16). If $C_{ST}(\tau'.o).ub < \theta$, τ' is pruned. Otherwise, we further refine trajectory τ' and compute $C_{ST}(\tau'.o)$. If $C_{ST}(\tau'.o) \geq \theta$, (τ', o) is added to M_o. Otherwise, τ' is pruned (lines 18–27). Set M_o is stored. If q is not at the top of heap H, we update q to be the top-ranked query source (lines 28–31). By combining the matching sets of all locations, the solution $\bigcup_{o \in O} M_o$ of the TL-Join is found (line 32).

Let T'_θ denote the scanned trajectory set for the search process from each location. In the spatial domain, the time complexity is $O(|T'_\theta||O|(|V|log|V| + |E|))$ (the same as PSF), while in the temporal domain, the time complexity is $O(|T_\theta||O|)$. The time complexity of PCol is $O(|T'_\theta||O|(|V|log|V| + |E|)) + O(|T'_{\theta'}||O|) = O(|T'_\theta||O|(|V|log|V| + |E|))$. If the value of θ is sufficiently large, the time complexity is close to $O(|O|(|V|log|V| + |E|))$.

Algorithm 3: PCol Search

Data: a set O of locations, a set T of trajectories, and a threshold θ
Result: $\bigcup_{o \in O} M_o$

```
∀o ∈ O(M_o ← ∅);
for each location o in O do
    UB ← 0; ∀q ∈ H(q.l ← 0);
    if 2d_c < |o.R| then
        scan timestamps in range(t_2, t_3);
    q ← H.top;
    while true do
        expand(q);
        update UB;
        if UB < θ then
            prune all unscanned trajectories;
            for each spatially fully scanned trajectory τ do
                compute C_ST(τ.o);
                if C_ST(τ.o) ≥ θ then
                    M_o.add(τ, o);
                else
                    prune τ;
            for each partly scanned trajectory τ' do
                compute C_ST(τ'.o).ub;
                if C_ST(τ'.o).ub < θ then
                    prune τ';
                else
                    refine τ' and compute C_ST(τ'.o);
                    if C_ST(τ'.o) ≥ θ then
                        M_o.add(τ', o);
                    else
                        prune τ';
            store M_o;
            break;
        if q ≠ H.top then
            q ← H.top;
    return ⋃_{o∈O} M_o;
```

The time complexity of PCol is the same as that of PSF, and the advantage of PCol lies in that it has a higher pruning power and defines a smaller candidate set T'_θ.

6 Experimental Results

We report on experiments with real and synthetic spatial data that offer insight into the properties of the developed algorithms.

6.1 Settings

We use two spatial networks, namely the Beijing Road Network (BRN) and the New York Road Network (NRN)[10], which contain 28,342 vertices and 27,690 edges, and 95,581 vertices and 260,855 edges, respectively. The graphs are stored using adjacency lists. In BRN, we use a real taxi trajectory data set collected by the T-drive project [27], while in NRN, we use a real taxi trajectory data set from New York[10]. The time range of a trajectory is 1–2 hours. We use real location data in BRN, i.e., for POIs (e.g., restaurants and shopping malls), we use real locations and real time ranges (opening hours, e.g., 3–5 hours for a restaurant, 7–10 hours for a shopping mall), and for accidents, we use real locations and synthetic time ranges (e.g., 0.5–2 hours), and we use synthetic location data in NRN.

In the experiments, the index structure of PTF (cf. Section 3) and the spatial networks of PSF and PCol (when running Dijkstra's expansion [10], cf. Sections 4 and 5) are memory resident, as the memory occupied is 34 MB and 44 MB for BRN and 42 MB and 55 MB for NRN. Trajectories and locations are also memory resident for all algorithms, and they occupy 279 MB for BRN and 2.2 GB for NRN. All algorithms are implemented in Java and run on a cluster with 10 data nodes. Each node is equipped with two Intel® Xeon® Processors E5-2620 v3 (2.4GHz) and 128GB RAM. Unless stated otherwise, experimental results are averaged over 10 independent trials using different query inputs. The main performance metrics are runtime and the number of location-trajectory pair visits. The number of location-trajectory visits is used as a metric because it reflects the number of data accesses. In multi-threaded executions, the total runtime is the maximum runtime among all individual threads.

Trajectories in T are selected randomly from the real data sets. The parameter settings are listed in Table II. For PTF (Section 3), the best performance is achieved when the index contains 56 leaf nodes for BRN and 545 leaf nodes for NRN, and when each leaf node contains at most 8,192 (trajectory, location) pairs ($M = 8,192$) in BRN and at most 16,384 (trajectory, location) pairs ($M = 16,384$) in NRN. Compared to the equal-partition grid index [16], the performance of the balanced grid index is improved by around 20%. Because computing network distances online is time-consuming, we pre-compute the all-pairs shortest paths distances in the graphs (for PTF only, not for PSF and PCol). PTF, PSF (Section 4), and PCol (Section 5) are denoted by "PTF," "PSF," and "PCol" in subsequent figures. The PCol algorithm without the heuristic scheduling strategy is denoted by "PCol-w/o-h."

6.2 Pruning Effectiveness

First, we study the pruning effectiveness of the algorithms using the default settings. The results are shown in Table III, where the reported candidate and pruning ratios are defined as follows.

$$Candidate\ ratio = \frac{|C|}{|T||O|}$$

$$Pruning\ ratio = 1 - Candidate\ ratio,$$

where $|C|$ is the size of the candidate set. The pruning ratio shows how many trajectory-location pairs are pruned, while the candidate ratio shows how many trajectory-location pair remains (to be processed in the next step). The candidate ratio

10. https://publish.illinois.edu/dbwork/open-data/

Table II: Parameter Settings

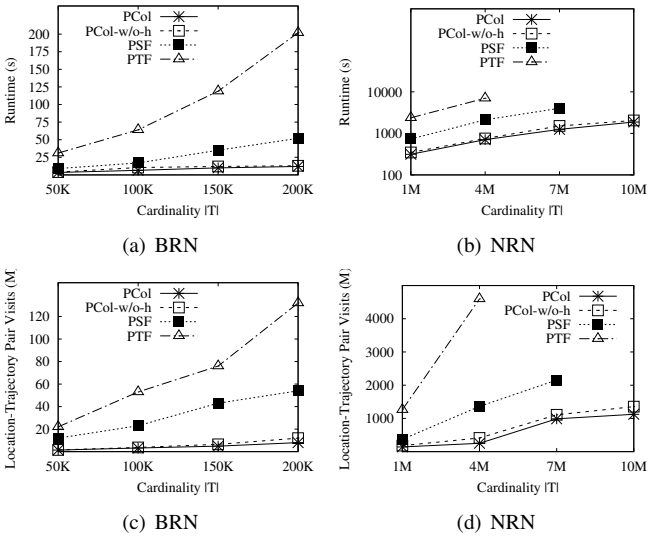
	NRN	BRN
Trajectory cardinality $\|T\|$	1,000,000–10,000,000 /default 1,000,000	50,000–200,000 /default 100,000
Location cardinality $\|O\|$	500,000–2,000,000 /default 500,000	25,000–100,000 /default 50,000
Average location time range $o.R$	1–7 hours /default 1 hour	1–7 hours /default 1 hour
Coupling duration d_c	20–40 minutes /default 25 minutes	20–40 minutes /default 25 minutes
Threshold θ	0.9–0.98/ default 0.96	0.9–0.98/ default 0.96
Preference parameter λ	0.1–0.9/ default 0.5	0.1–0.9/ default 0.5
Thread count m	24–120/ default 24	24–120/ default 24

is directly proportional to the running time. We see that the candidate ratio of PCol is only 6.1–11.5% of that of PTF and 12–37.5% of that of PSF. Further, the heuristic scheduling strategy reduces the candidate ratio by 14–25%.

Table III: Pruning Effectiveness for TL-Join

	PTF	PSF	PCol-w/o-h	PCol
Candidate ratio (BRN)	0.98%	0.51%	0.07%	0.06%
Pruning ratio (BRN)	99.02%	99.49%	99.93%	99.94%
Candidate ratio (NRN)	0.26%	0.08%	0.04%	0.03%
Pruning ratio (NRN)	99.74%	99.92%	99.96%	99.97%

6.3 Effect of Trajectory Cardinality $|T|$



(a) BRN (b) NRN

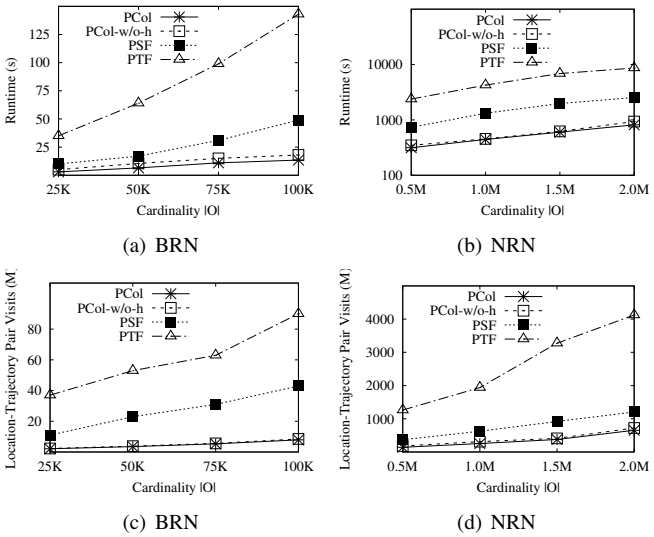
(c) BRN (d) NRN

Fig. 5. Effect of Trajectory Cardinality $|T|$

Figure 5 shows the effect of trajectory cardinality $|T|$ on the performance of the algorithms. Intuitively, a larger $|T|$ causes more (trajectory, location) pairs to be processed (cf. the complexity analysis in Sections 3.4, 4.3, and 5.5), meaning that the runtime and the number of (trajectory, location) pair visits are expected to increase for all algorithms. We see that PCol outperforms PTF by almost an order of magnitude and that it outperforms PSF by 230–300% in terms of both runtime and (trajectory, location) pair visits; and we see that the heuristic scheduling strategy can further improve PCol by 15–33% in terms of both runtime and (trajectory, location) pair visits. PCol is able to process 1 M trajectories ($|T|$ = 1 M and $|O|$ = 0.5 M) in 314 seconds and 10 M trajectories ($|T|$ = 10 M and $|O|$ = 0.5 M) in 1,874 seconds with the default 24 threads (see Figure 5(b)). These results demonstrate the importance of balancing the pruning power in the spatial and temporal domains (Section 5.2) and the benefit of the heuristic scheduling strategy (Section 5.4).

The runtime is not fully aligned with the number of (trajectory, location) pair visits because the algorithms expend computational effort on maintaining the bounds and priority labels (for PCol) used to prune the search space. The resulting cost may offset the benefits of the reduction in the number of (trajectory, location) pair visits. In particular, the filter phase of PTF computes and maintain bounds for almost all trajectory pairs.

6.4 Effect of Location Cardinality $|O|$



(a) BRN (b) NRN

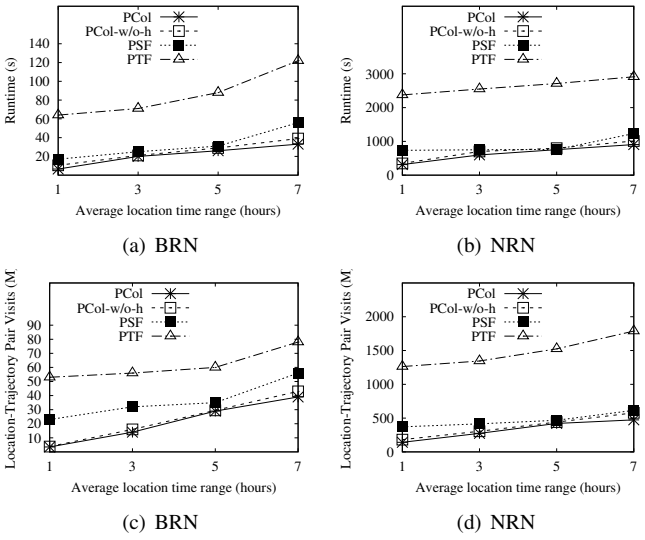
(c) BRN (d) NRN

Fig. 6. Effect of Location Cardinality $|O|$

Next, we study the effect of location cardinality $|O|$ on the performance of the algorithms. Similar to the effect of the trajectory cardinality $|T|$, a larger $|O|$ implies a longer runtime and more (trajectory, location) pairs to be processed for all algorithms. From Figure 6, we see that PCol has a clear advantage over PTF, PSF, and PCol-w/o-h. PCol is able to process 2 M locations ($|T|$=1 M and $|O|$= 2 M) in 815 seconds (see Figure 6(b)).

6.5 Effect of Average Location Time Range $o.R$

Figure 7 shows the effect of varying location time range $o.R$ on efficiency. A larger $o.R$ may lead to a higher temporal correlation (cf. Equation 3). So we may have more qualified (trajectory, location) pairs to refine, meaning that the runtime and the number of (trajectory, location) pair visits are expected to increase for all algorithms. Moreover, for PTF, a larger $o.R$ leads to more locations to be stored in non-leaf nodes, which



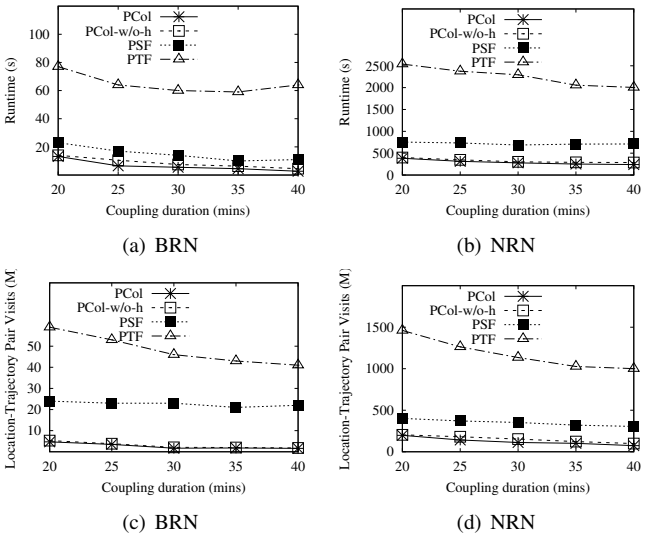
(a) BRN (b) NRN

(c) BRN (d) NRN

Fig. 7. Effect of Average Time Range $o.R$

offsets the benefit of parallel processing. For PCol, a larger $o.R$ may weaken its pruning power (Equation 11). So the runtime and the number of (trajectory, location) pair visits of PTF and PCol increase faster than those of PSF. But PCol still holds a clear advantage over PTF, PSF, and PCol-w/o-h.

6.6 Effect of Coupling Duration d_c



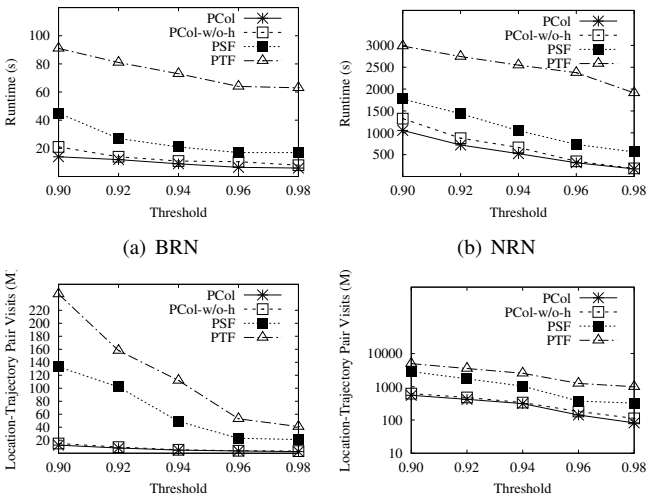
(a) BRN (b) NRN

(c) BRN (d) NRN

Fig. 8. Effect of Coupling Duration d_c

The next study concerns the effect of coupling duration d_c on the efficiency of the algorithms. As can be seen in Figure 8, a larger d_c leads to a larger k_τ and may lead to a smaller temporal correlation (cf. Equation 3). So we may have fewer qualified (trajectory, location) pairs to refine, meaning that the runtime and number of (trajectory, location) pair visits are expected to decrease for all algorithms. In addition, a larger d_c may enhance the pruning power of PCol (cf. Equations 11 and 12).

6.7 Effect of Threshold θ



(a) BRN (b) NRN

(c) BRN (d) NRN

Fig. 9. Effect of Threshold θ

We show results when varying threshold θ in Figure 9. A larger θ leads to higher pruning effectiveness (cf. Sections 3.3, 4.2, and 5.3). Thus, the larger θ becomes, the smaller the search space becomes. Therefore, the runtime and number of (trajectory, location) pair visits are expected to decrease correspondingly for all algorithms. In addition, in PTF, a larger θ is useful in reducing the similarity computation (see Equation 6), which further enhances the efficiency. In Figure 9(b), we see that when $\theta = 0.98$, PCol is able to process 1 M trajectories ($|T|$ = 1 M and $|O|$ = 0.5 M) in 174 seconds.

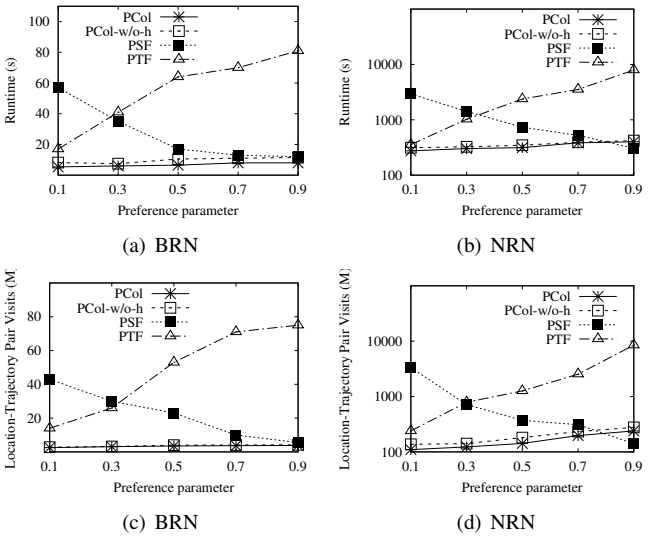
We also test that when $\theta = 0.5$ in BRN and NRN, PCol is able to process 100 K and 1 M trajectories under 240 threads in 33 seconds and 2440 seconds.

6.8 Effect of Preference Parameter λ

Figure 10 shows the effect of varying preference parameter λ. Parameter λ enables adjusting the relative preference of spatial and temporal similarity (see Equation 4). When λ = 1, the TL-Join is in the spatial domain only, and when λ = 0, only temporal similarity is considered. Figure 10 shows that the spatial domain needs more search effort than the temporal domain. When λ increases, the pruning power of PTF is weakened because its pruning is driven by the temporal domain (cf. Section 3.3). On the other hand, the pruning power of PSF is enhanced as it uses spatial upper bound to prune the search space (cf. Section 4.2). When λ is close to 1, the efficiency of PSF is very close to that of PCol.

6.9 Effect of Thread Count m

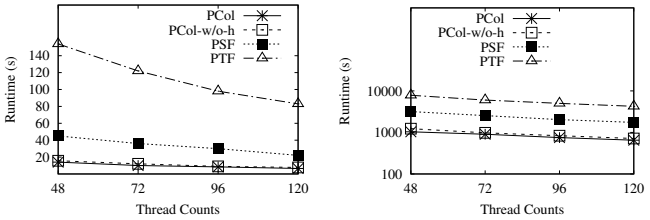
We study the effect of thread count m on the efficiency of the algorithms using large trajectory data sets ($|T|$ = 200 K and $|O|$ = 100 K for BRN and $|P|$ = 10 M and $|O|$ = 0.5 M for NRN). The results are shown in Figure 11. We see that PCol outperforms PTF by almost an order of magnitude in term



(a) BRN (b) NRN

(c) BRN (d) NRN

Fig. 10. Effect of Preference Parameter λ



(a) BRN (b) NRN

Fig. 11. Effect of Thread Count m

of runtime and outperforms PSF by almost 300% in term of runtime. In BRN, PCol is able to process 200 K $\times$ 100 K (trajectory, location) pairs with 120 threads in 8.3 seconds, while in NRN, the PCol is able to process 10 M $\times$ 0.5 M (trajectory, location) pairs with 120 threads in 651 seconds.

We increase the thread count from 48 to 120 (2.5 times). This improves the runtimes of PSF and PCol by a factor of around 2.1, while the runtime of PTF is improved by a factor of around 1.8. The main reason for the smaller improvement of PTF is that more threads (more leaf nodes) leads to a higher merging cost (see Section 3).

7 Related Work

7.1 Trajectory-to-Location Matching

Existing trajectory-to-location matching studies typically consider (i) matching solely in the spatial domain [15], [18], [20]–[22], [25], [28] or (ii) use point-to-point matching [18], [19], [21], [28] in the spatial or temporal domain. For the first case, the matching results do not support time-aware applications, while for the second case, the matched (trajectory, location) pairs are unable to capture the continuous correlations between trajectories and locations in the spatial and temporal domains. The so-called Semantic Enrichment approach [12] utilizes the stay time at a location to infer a traveler's activity. It uses point-to-point matching in the spatial domain and range matching in the temporal domain. This matching scheme is not feasible for location recommendation because it relies on a constraint on the stay time (e.g., 30 minutes) of travelers at a location. For example, if a traveler stay at some points of interest (e.g., restaurants, shopping malls, and sightseeing places) for more than 30 minutes, we can infer the trajectory accompanied activities (e.g., dinner, shopping, and tourist).

Trajectory-to-location matching may bring significant benefits to diverse applications. RPNN (reverse path nearest neighbor query [21]) targets the application of location ranking and recommendation. For example, when setting a new facility, RPNN uses the number of matched trajectories to define the influence factors of location candidates, and then finds the most influential location for the new facility to maximize its commercial value. ATSQ [28], UOTS [18], and PTM [19] are location-based trajectory search queries and they are useful in travel planning and carpooling recommendation (e.g., using historic trajectories for travel planning, or recommending travelers with similar travel trajectories for carpooling). The Semantic Enrichment [12] uses trajectories to analyze traveler's activities.

Most existing centralized trajectory-to-location join algorithms (e.g., VID Joins [20], [25]) operate in Euclidean space and cannot process large trajectory data sets. From the experiments reported in the literature [20], [25], the VID joins can process at most 12 K trajectories. In contrast, the TL-Join is performed in a spatial network and can process 10 M trajectories with a reasonable runtime, some three orders of magnitude more trajectories than for the VID joins.

7.2 Trajectory Similarity Join

Trajectory similarity joins [2], [3], [7], [11], [14], [16], [23] target applications such as trajectory near-duplicate detection, data cleaning, ridesharing recommendation, and traffic congestion prediction. Developing such joins typically involves a definition step and a query processing step. First, a similarity function, e.g., Sim, is defined to evaluate the spatial and temporal similarities between two trajectories, e.g., τ and τ'. Second, an efficient algorithm is developed to retrieve the spatiotemporally similar trajectory pairs. The trajectory similarity function should be symmetrical, i.e., $\mathrm{Sim}(\tau, \tau') = \mathrm{Sim}(\tau', \tau)$. Most existing trajectory similarity joins (e.g., [2], [3], [7], [11], [14], [23]) use a time interval threshold to constrain the temporal proximity of two trajectories. In contrast, the TS-Join [16] defines trajectory similarity in a continuous manner. The best connected trajectory (*BCT*) [8] and its variants [18], [19], [28] cannot be used in the trajectory similarity joins due to being asymmetric. Several similarity functions for time-series data also exist, including Dynamic Time Warping [26], Longest Common Subsequence [1], and Edit Distance on Real sequence [6].

The TS-Join [16], [17] is based on a divide-and-conquer strategy. For each trajectory τ, the algorithm retrieves trajectories that are similar to τ. The trajectory-search processes are independent of each other and are performed in parallel. The TS-Join algorithm cannot process the TL-Join due to their different query arguments (trajectories vs. trajectories and

locations), and their different matching functions (point-to-point matching vs. range-based matching). The TL-Join needs its own specific solutions.

8 Conclusion and Future Work

We studied the efficient processing of a novel Trajectory-to-Location join (TL-Join) operation in spatial networks, which may benefit diverse applications such as location recommendation, and trajectory activity analysis. We developed three parallel algorithms: parallel temporal-first search (PTF), parallel spatial-first search (PSF), and parallel collaborative search (PCol). We also defined upper and lower bounds and a heuristic scheduling strategy to enable effective search space pruning. The performance of the developed algorithms were studied empirically in extensive experiments on large spatial data sets.

Two future research directions exist. First, it is of interest to take the visiting sequence of trajectory sample points into account when matching trajectories and locations. To do this, new upper and lower bounds on the spatiotemporal correlation and a new heuristic scheduling strategy are needed. Second, it is of interest to extend existing techniques to support a top-k TL-Join without a matching threshold θ. This calls for updated pruning techniques, including adding pruning to the same thread and updating the corresponding upper and lower bounds (without a given threshold).

References

[1] R. Agrawal, K. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB*, pages 490–501, 1995.

[2] P. Bakalov, M. Hadjieleftheriou, E. J. Keogh, and V. J. Tsotras. Efficient trajectory joins using symbolic representations. In *MDM*, pages 86–93, 2005.

[3] P. Bakalov and V. J. Tsotras. Continuous spatiotemporal trajectory joins. In *GSN*, pages 109–128, 2006.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, 2005.

[5] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing large trajectory data sets with SETI. In *CIDR*, 2003.

[6] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[7] Y. Chen and J. M. Patel. Design and evaluation of trajectory join algorithms. In *ACM GIS*, pages 266–275, 2009.

[8] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: an efficiency study. In *SIGMOD*, pages 255–266, 2010.

[9] V. T. de Almeida and R. H. Güting. Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33–60, 2005.

[10] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math*, 1:269–271, 1959.

[11] H. Ding, G. Trajcevski, and P. Scheuermann. Efficient similarity join of large sets of moving object trajectories. In *TIME*, pages 79–87, 2008.

[12] B. Furletti, P. Cintia, C. Renso, and L. Spinsanti. Inferring human activities from GPS tracks. In *UrbComp@KDD 2013*, pages 5:1–5:8, 2013.

[13] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[14] S. Ray, A. D. Brown, N. Koudas, R. Blanco, and A. K. Goel. Parallel in-memory trajectory-based spatiotemporal topological join. In *2015 IEEE International Conference on Big Data*, pages 361–370, 2015.

[15] S. Shang, L. Chen, C. S. Jensen, J. Wen, and P. Kalnis. Searching trajectories by regions of interest. *IEEE Trans. Knowl. Data Eng.*, 29(7):1549–1562, 2017.

[16] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis. Trajectory similarity join in spatial networks. *PVLDB*, 10(11):1178–1189, 2017.

[17] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis. Parallel trajectory similarity joins in spatial networks. *VLDB J.*, 27(3):395–420, 2018.

[18] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167, 2012.

[19] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou. Personalized trajectory matching in spatial networks. *VLDB J.*, 23(3):449–468, 2014.

[20] S. Shang, K. Xie, K. Zheng, J. Liu, and J. Wen. VID join: Mapping trajectories to points of interest to support location-based services. *JCST*, 30(4):725–744, 2015.

[21] S. Shang, B. Yuan, K. Deng, K. Xie, and X. Zhou. Finding the most accessible locations: reverse path nearest neighbor query in road networks. In *ACM GIS*, pages 181–190, 2011.

[22] S. Shang, K. Zheng, C. S. Jensen, B. Yang, P. Kalnis, G. Li, and J. Wen. Discovery of path nearby clusters in spatial networks. *IEEE Trans. Knowl. Data Eng.*, 27(6):1505–1518, 2015.

[23] N. Ta, G. Li, and J. Feng. Signature-based trajectory similarity join. *IEEE TKDE*, 29(4):870–883, 2017.

[24] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *SSDBM*, pages 379–388, 2006.

[25] K. Xie, K. Deng, and X. Zhou. From trajectories to activities: a spatio-temporal join approach. In *LBSN*, pages 25–32, 2009.

[26] B. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[27] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE TKDE*, 25(1):220–232, 2013.

[28] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang. Towards efficient search for activity trajectories. In *ICDE*, pages 230–241, 2013.

Shuo Shang is a research scientist at KAUST. He obtained his Ph.D. in computer science from The University of Queensland, Australia. His research interests include spatiotemporal databases, spatial trajectory computing, urban computing, and location based social media. He is the Track Co-chair of the 2018 IEEE International Congress on Big Data and the Demo Co-chair of the of the 2017 APWeb/WAIM Joint Conference. He has served as PC member, session chair, guest editor, and invited reviewer for many prestigious conferences and journals, including SIGMOD, VLDB, ICDE, CIKM, TKDE, The VLDB Journal, ACM TIST, IEEE TITS, ACM TSAS, GeoInformatica, and WWW Journal.

Lisi Chen is Lecturer of computer science at University of Wollongong. He obtained his Ph.D. in computer science from Nanyang Technological University. His research interests include geo-textual data management, spatial keyword query evaluation, and location based social networks.

Kai Zheng is a full Professor with the University of Electronic Science and Technology of China. He received his PhD degree in Computer Science from The University of Queensland in 2012. He has been working in the area of spatial-temporal databases, uncertain databases, social-media analysis, and in-memory databases.

Christian S. Jensen is Obel Professor of Computer Science at Aalborg University, Denmark. He was recently at Aarhus University for three years and at Google Inc. for one year. His research concerns data management and data-intensive systems, and its focus is on temporal and spatiotemporal data management. Christian is an ACM and an IEEE fellow, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences. He is editor-in-chief of ACM Transactions on Database Systems and was an editor-in-chief of The VLDB Journal from 2008 to 2014.

Zhewei Wei is an associate professor at Renmin University of China. He obtained his Ph.D in Computer Science and Engineering from The Hong Kong University of Science and Technology. His research interests include streaming algorithms and data structures.

Panos Kalnis is a professor at KAUST. He received his Diploma in Computer Engineering from the Computer Engineering and Informatics Department, University of Patras, and PhD from HKUST. His research interests include Database outsourcing and Cloud Computing, Mobile Computing, and Spatiotemporal and High-dimensional Databases. He is an associate editor of TKDE, and The VLDB Journal.