Search by Pattern in GPS Trajectories

Maros Cavojsky(✉) and Martin Drozda

Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology, Bratislava, Slovakia
{maros.cavojsky,martin.drozda}@stuba.sk

Abstract. In search by pattern in GPS trajectories, user draws a trajectory, the pattern query, and then receives a set of trajectories ranked by their similarity to the pattern query. We argue that when user draws a pattern query, an initial part of this query (prefix of chosen length) should have more weight than the rest of query. We assume that after receiving a set of similar trajectories, user can refine the pattern query in order to receive more relevant results. We give explanation of our approach by means of web search, where a user searches, for example, for "bratislava castle" and then adds a refinement to this query "opening hours", where removing the initial part of query does not make sense, as search for "opening hour" alone would return irrelevant results. This idea has led us to considering pattern search that is weighted toward query prefix. We experimentally evaluate this approach, in our experimentation we apply the Geolife data set (Microsoft Research Asia).

Keywords: GPS trajectory · Geolife data set · pattern search · Needleman-Wunsch algorithm · Smith-Waterman algorithm · Geohash · Hausdorff distance

1 Introduction

In search by pattern, user draws a trajectory, the pattern query, and receives a set of trajectories ranked by their similarity to this query. Preferably, the result is obtained in real time, or with negligible delay (less than 500 ms) [7]. Trajectories represent recorded positions of people or mobile platforms (cars, trucks, drones, pets etc.) equipped with GPS (Global Positioning System), or a similar system. Positions measured by GPS are estimates of real positions subject to various errors. Precision of GPS devices may be influenced by various signal propagation phenomena such as signal reflection, diffraction, scattering or multi-path propagation. In urban environments, GPS satellite occlusion may happen [13].

When comparing a query to one or several trajectories, we can view the problem as an edit distance problem, where the query and all trajectories are transformed to a letter string representation. Edit distance is the number of single letter edits that is necessary until the two representations become identical. Such a problem transformation leads to algorithms that can compute global

J. Taheri et al. (Eds.): MobiCASE 2022, LNICST 495, pp. 117–132, 2023.
https://doi.org/10.1007/978-3-031-31891-7_9

or local alignment of sequences of letters, with the Needleman-Wunsch algorithm [10] and the Smith-Waterman algorithm [16], respectively, being the prime representatives. Both these algorithms are often applied to align DNA or protein sequences. Their advantage is that they can directly align two sequences, whereby their edit distance [15], and thus their similarity can be both computed [4]. Their disadvantage is their time complexity $O(mn)$, where m is the length of the first sequence and n is the length of the other sequence. Dense urban environments give rise to an enormous number of similar trajectories, therefore applying these algorithms may be prohibitively inefficient.

When searching by pattern we can distinguish between two basic cases:

- The search returns top-k similar trajectories, where these trajectories may not have any identical sub-trajectories, or
- the search returns top-k similar trajectories, where it is required that these trajectories share a sub-trajectory, where the least length of this sub-trajectory is parameterized as γ.

In both cases it is necessary to define a similarity relation, in the latter case it is also necessary to choose the least sub-trajectory length. We will define both formally later on.

The former case leads to complex tree pruning approaches based, for example, on the Hausdorff distance [14]. An example of such an approach is the Repose approach introduced by Zheng et al. [20]. This approach relies on a trie to be computed. In order to facilitate search efficiency, this approach applies a signature based representation of trajectories based on the Z-order [5], where a $2d$-trajectory, a sequence of (x, y) positions, is transformed to a $1d$ (binary string) representation.

The latter approach, the focus of our research, assumes that there is a limited motivation for finding similar trajectories that do not share a common sub-trajectory. Similar to the previous approach, we take advantage of a $1d$ trajectory representation. Unlike in the previous case, it is not necessary to compute a trie, instead a Geohash [11] is applied to reduce the number of candidate trajectories, which are later ranked with respect to their Hausdorff distance to the query pattern.

Our results can be succinctly summarized as follow. We compare the performance of our approach to approaches applying the Needleman-Wunsch algorithm and the Smith-Waterman algorithm. For experimental evaluation we use the Microsoft Asia Geolife data set [22] that contains recorded trajectories in Beijing, China. We show that in such a dense urban environment as Beijing, the Needleman-Wunsch algorithm and the Smith-Waterman algorithm do not offer the possibility of a real time query pattern search. Therefore we propose an approach based on Geohashing that aims at real time GPS trajectory querying.

This document is organized as follow. In Sect. 2, we review the related work. In Sect. 3, we explain why we consider pattern by search that is weighted toward pattern query prefix. In Sect. 4, we introduce sequence alignment algorithms such as the Needleman-Wunsch algorithm and the Smith-Waterman algorithm. In Sect. 5, we explain our methodology for efficient search by pattern. In this section we also

overview our setup for experimental evaluation. In Sect. 6, we present our experimental results, and finally, in the next section we present our conclusions.

2 Related Work

Whether the Needleman-Wunsch algorithm is suitable for pairwise trajectory comparison is in detail investigated in [3,4]. The authors point out that GPS trajectories often form clusters, also referred to as nests, that emerge due to signal reflection and multi-path signal propagation. Therefore they focus on clarifying whether this algorithm is also applicable in such detrimental conditions.

The existence of nests has attracted the attention of Yang et al. [17], where they introduce the notion of noise points. The authors suggest that one possibility for computing noise points is to compute move points and stay points, where noise points are computed as their complement.

Move Ability, proposed by Luo et al. [8], is also often applied to compute nests, or noise points. Move Ability computes the distance of end points of a GPS trajectory and compares it with the sum of distances of each successive GPS position in this GPS trajectory. Cavojsky et al. [4] apply Move Ability to filter out nests, so that GPS trajectory similarity is not affected by this phenomenon.

Zheng et al. [20] propose an approach, that they call Repose, for efficient comparison of GPS trajectories. To achieve a high degree of efficiency, they compute a trie that allows for comparing a pattern query with a set of pivot trajectories. After a necessary amount of pruning is done, a sub-set of candidate GPS trajectories is computed, the Hausdorff distance between pattern query and candidate GPS trajectories is computed, then the candidate GPS trajectory having the least distance is selected. This approach can be extended to top-k search. A deficiency of Repose is its focus on search efficiency without considering various signal propagation phenomena and their impact on measured trajectories.

Yin et al. [18] address yet another great challenge of pattern search in GPS trajectories. They propose an approach for error-bounded GPS trajectory compression, while keeping support for range queries.

The last challenge that we mention in this short literature review is GPS trajectory clustering with enhanced privacy protection of users. Zhao et. al. [19] propose to add Laplacian noise in order to achieve increased user privacy. The authors apply the Edinburgh Informatics Forum Pedestrian Database [6] that captures pedestrian traffic at the main building of the School of Informatics at the University of Edinburgh. This data set likely has no or negligible amount of noise caused by signal propagation phenomena.

3 Psychology of Search by Pattern

Let us consider the example shown in Fig. 1. It shows 5 different GPS trajectories that we want to align, and then to reason about their similarity.

While these GPS trajectories share the same direction and some of them overlap, their degree of similarity is unclear. If b becomes the pattern query, the

similar trajectories should only be a and d. If a becomes the pattern query, the user would expect b as well as d as the results, since trajectories that are similar at their beginning might be *intuitively perceived* as more similar than trajectories that are similar at later trajectory stages.

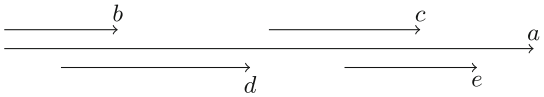


Fig. 1. GPS trajectories a,b,c,d and e to be aligned.

The rational behind giving more weight to similarity at the start rather than at the end of a trajectory is an assumption that user first starts drawing the most important portion of the search query. This then may get followed by a refinement that aims to decrease the number of possible results. This is similar to web searching, where a keyword root query is entered (for example "bratislava castle"), and then in order to obtain more relevant results, one or several keywords are added (for example "opening hours"), influenced by the results obtained in the previous search iteration. This may be considered an instance of *confirmation bias* [12], where user seeks to find a suitable result, however, then he refines the pattern query by adding to it in order to obtain a result that supports his prior believes. Notice that removing the root query ("bratislava castle") does not make sense, as searching only for ("opening hours") would return irrelevant results.

The simplest model for this kind of behavior is to discount future search keywords exponentially [1], i.e. giving less importance to refinement than to root query. This can be thus modeled as:

$$\lambda \exp^{-\lambda x},$$

where $\lambda > 0$ is a parameter of the exponential distribution, called rate. We may also require that the first γ search keywords have the same weight. This extends to a search by pattern, where user has to draw a pattern that spans at least γ different hashcodes, then a refinement can be drawn. Results are later ranked by computing their Hausdorff distance to search query.

Given two point sets A and B, the Hausdorff distance [14] between A and B is defined as:

$$H(A, B) = \max\{h(A, B), h(B, A)\}.$$

Assuming that A and B are compact sets, we can formulate h as follows:

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|, \tag{1}$$

where $\|\cdot\|$ is a norm defined on the plane. Herein we apply the Euclidean L_2 norm. The sets A and B are points in a pattern query and trajectories to which we compare.

4 Algorithms Overview

Sequence alignment is the process of aligning the letters of a pair of sequences, such that the number of matched letters is maximized. We can describe the alignment between two sequences with the following notation:

```
TACGGGCCCGCTA-C
||   | x| ||| |
TA---G-GC-CTATC
```

The vertical lines "—" (pipes), represent matching letters. Mismatches are marked by "x". Gaps are indicated by the dash "-", they are inserted between letters to keep space of missing letters in order to optimize the number of matches. In the above example, the letters A (adenine), T (thymine), G (guanine) and C (cytosine) correspond to the four known types of DNA nucleotides.

Two sequences can be aligned by computing the minimum number of single letter edits that yield these two sequences identical. The number of these edits is referred to as "edit distance" or Levenshtein distance [9]. Sellers showed that approaches formulated in terms of minimizing edit distance and maximizing similarity (e.g. Needleman-Wunch algorithm) are equivalent [15]. Wagner-Fischer algorithm [9] is often applied to compute edit distance. Wagner-Fischer algorithm, Needleman-Wunsch Algorithm and Smith-Waterman Algorithm are examples of dynamic programming algorithms.

4.1 Needleman-Wunsch Algorithm (NWA)

NWA is a global alignment algorithm, meaning the result always aligns the entire input sequences. The algorithm is using scoring matrix for its sequence alignment, in simpler case defined as:

$$S_{a,b} = \begin{cases} 1, & \text{if } a = b, \\ -1, & \text{if } a \neq b. \end{cases}$$

In addition to a scoring matrix, algorithm also applies penalties for gaps. The most common gap penalty is the linear gap penalty, defined as follows:

$$W_L(d) = Gd,$$

where the gap penalty W_L is proportional to the length d of the gap by a parameter G. There are cases where more complicated approach is necessary, for example, an "affine gap penalty" penalizes opening a gap by one parameter, and extending the gap by another parameter. Such a gap penalty can by defined as follows:

$$W_L(d) = G + (d-1)E,$$

where we include a gap open penalty G and a gap extension parameter E proportional to the length d of the gap.

The algorithm computes the score matrix F using a recurrence relation, such that the values of a given cell of the matrix F are defined in terms of the neighboring cells. NWA applies for computing an alignment of two sequences x and y the following recurrence relation:

$$F_{i,j} = max \begin{cases} F_{i-1,j} + G, & \text{skip a position in x,} \\ F_{i,j-1} + G, & \text{skip a position in y,} \\ F_{i-1,j-1} + S_{x[i],y[j]}, & \text{match / mismatch.} \end{cases}$$

4.2 Smith-Waterman Algorithm (SWA)

In many cases we are only interested in aligning a portion of the sequence to find a local alignment. Furthermore, we do not necessarily want to force the first and last letter to be aligned. SWA is an alignment algorithm that has these properties. We can define a set of boundary conditions for the scoring matrix $F_{i,j}$, namely that the score is 0 at the boundaries:

$$F_{i,0} = 0, \text{ for } 1 \leq i \leq \text{length}(x),$$

$$F_{0,j} = 0, \text{ for } 1 \leq i \leq \text{length}(y).$$

For the rest of score matrix we apply the following recurrence relation:

$$F_{i,j} = max \begin{cases} F_{i-1,j} + G, & \text{skip a position in x,} \\ F_{i,j-1} + G, & \text{skip a position in y,} \\ F_{i-1,j-1} + S_{x[i],y[j]}, & \text{match / mismatch,} \\ 0, & \text{zero-out negative scores.} \end{cases}$$

The key difference between NWA (global alignment) and SWA (local alignment) is that when computing the global alignment we start backtracking from the lower right cell of the matrix. To compute the local alignment we instead start at multiple cells which all share the maximum score in matrix and we keep backtracking until we reach zero score.

5 Methodology

In our experimental evaluation we investigate the suitability of three approaches for comparing trajectories:

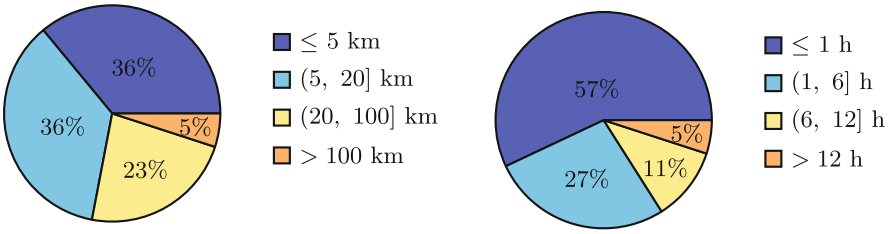
- our approach based on geohashing,
- Needleman-Wunsch Algorithm (NWA), and
- Smith-Waterman Algorithm (SWA).

All three approaches apply the same set of search patterns and all experiments are based on the Geolife data set.

The Geolife data set (Microsoft Research Asia) was collected by 182 users in a period of over three years (from April 2007 to August 2012). This data set

contains 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of 48,000+ h. These trajectories were recorded by different GPS loggers and GPS capable phones, and have a variety of sampling rates. This data set recorded a broad range of users' outdoor movements, including not only life routines such as "go home" and "go to work" but also entertainment and sports activities, such as shopping, sightseeing, dining, hiking, and cycling, for details see [21–23].

Several relevant statistics for the Geolife data set are shown in Figs. 2(a) and 2(b); see [2] for additional useful statistics.



(a) Trajectories by distance (km). (b) Trajectories by duration (hours).

Fig. 2. Distribution of trajectories in Geolife data set.

5.1 Pre-processing Geolife Data Set

One of the options how to pre-process trajectories is using Geohash [11], a hierarchical spatial data structure, that encodes a GPS location into a string of digits and letters. Encoding is done with subdividing space into buckets of grid shape, called Z-order curve. Latitude and longitude is encoded by base32, that uses digits 0–9 and almost all lower case letters except "a", "i", "l" and "o". If geohashes share the same prefix, the points within these geohashes are spatially close. The opposite may not hold, since points can be close to each other but may not share a prefix. An example of Geohash grid is shown in Fig. 3.

By using Geohash we are able to assign to every trajectory a set of geohashes that contain the points that belong to these trajectories. Before searching by pattern, we calculate the set of geohashes that represent the search pattern. We then select trajectories which have a non-empty intersection with this set of geohashes.

Let us consider the example depicted in Fig. 4, where solid circles are recorded locations and dotted circles are calculated locations by linear interpolation to fill geohashes between locations with gaps.

Pre-processing then consists of these steps:

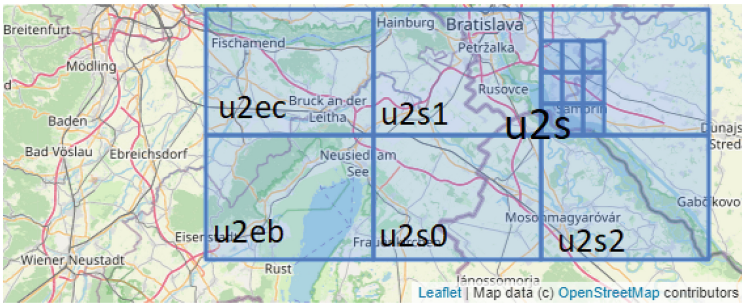


Fig. 3. Geohash.

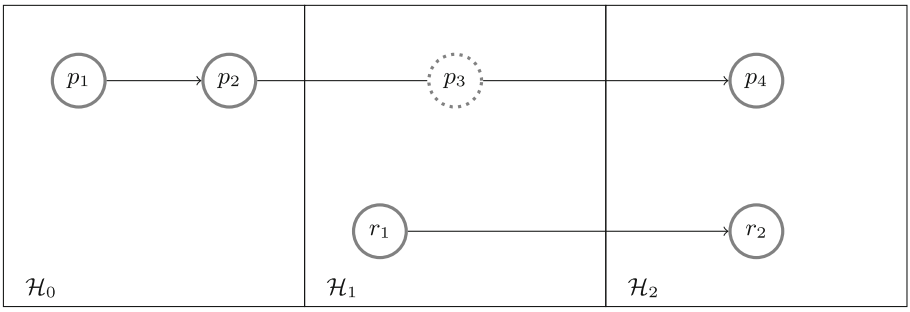


Fig. 4. Trajectories p and r encoded with geohashes.

- All positions are transformed to geohashes. We use geohashes with the length of 7, however, any granularity can be achieved by using shorter or longer geohashes.
- Repeatedly occurring geohashes are deleted, i.e. the next geohash in the series is always different from the previous one.
- If necessary, missing geohashes between positions are calculated by linear interpolation. If we do not include geohashes for gaps, our approach is unable to compare trajectories; see [4] for more information on gaps and on how they emerge.
- We do not consider trajectories that are both short and require interpolation. Interpolated geohash trajectories with length less then 6 geohashes are not considered in our evaluation.

As already described above, when applying NWA, SWA or our approach for similarity, we run these approaches on tracks, not on original GPS trajectories that we use to compute tracks.

Definition 1. *Let $\mathcal{H}_i$ be the i-th geohash, id the track identifier, and ℓ the track length. Track t is then defined as a sequence of pairs $t = [(\mathcal{H}_0, id), (\mathcal{H}_1, id), \cdots, (\mathcal{H}_{\ell-1}, id)]$, where $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ are neighboring geohashes.*

5.2 Our Approach Based on Geohashing

To search similar tracks with geohashing we first assign tracks to a hash map. We create hash keys by concatenating every γ geohash codes in track. Each hash key in hash map points to a set of tracks ids, that share that tuple of geohash sequence. The considered approach is shown in Fig. 5, where the tracks t_1, t_2 and t_3 are assigned to four hash map keys.

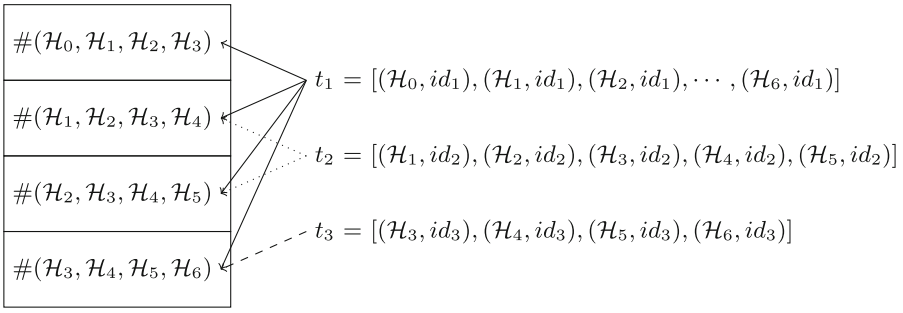


Fig. 5. Hashing tracks based on geohash codes for $\gamma = 4$.

In order to find tracks that are similar to search query $q = [(\mathcal{H}_0, id_0), \cdots, (\mathcal{H}_4, id_0)]$, the following steps get applied:

- Split q to tuples by applying a moving window of the size γ that at each iteration moves to the right by one geohash.

  For example for a track of length 5, and $\gamma = 4$ we compute two tuples (hash map keys):
  $[(\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3), (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4)]$.

- Retrieve tracks from keys of hash map based on tuples from pattern, and compute the union of retrieved tracks.

  For example:
  $r = \{id_1, id_2\} \cup \{id_1, id_2, id_3\} = \{id_1, id_2, id_3\}$
- Rank r by similarity to q, applying the Hausdorff distance between q and each result in r.

5.3 Comparison of Methods

The approaches considered herein can be succinctly characterized as follow:

Needleman-Wunsch Algorithm (NWA)

- time complexity to compute optimal global alignment(s) is $O(mn)$

- backtracking starts only in lower right cell
- backtracking ends in upper left cell
- allows for negative scores

Smith-Waterman Algorithm (SWA)

- time complexity to compute all local alignments is $O(mn)$
- backtracking starts at cells with maximum score
- backtracking ends at cell with 0 score
- negative scores are zeroed out

Our approach based on geohashing

- time complexity to find a set of similar tracks is $O(1)$, however, a hash map has to be first computed, which can be done when iterating through GPS trajectories during pre-processing
- union of track ids can be computed in linear time by merging assuming that ids are ordered, otherwise they need to be ordered in $O(|r| \log |r|)$

As a side note, we would like to mention that NWA is suitable for search if we expect trajectories to be nearly identical, on the other hand, SWA is suitable when we expect trajectories to be similar at least in some segment. How our approach based on geohashing compares to NWA and SWA is one of the goals of our experimental evaluation.

6 Experimental Results

For searching in the Geolife data set, we apply two pattern queries shown in Figs. 6 and 7. They both run through the city center of Beijing. The former one is a short query with the length of 11 km and the latter is a trip from the airport to the city center, its length is 39 km. The long pattern query contains the short pattern query in order to simplify reasoning about obtained results.

6.1 Searching with Needleman-Wunsch Algorithm

For searching using NWA we apply the parameters specified in Table 1, these parameters are also applied in a previous study on trajectory similarity by Cavojsky and Drozda [3]. The similarity between two tracks is therein defined as follows:

Definition 2. *Let C_a and C_b be two tracks. Tracks C_a and C_b are similar if these tracks have at least α matches and no more than β subsequent mismatches.*

We run NWA incrementally by increasing the length of pattern query by 5 Geohash codes (approx. 600 m) up to the full length of search pattern for both the short and long pattern. In other words, we start with a prefix of the pattern query having the length of 5 Geohashes, subsequently we increase its length by 5

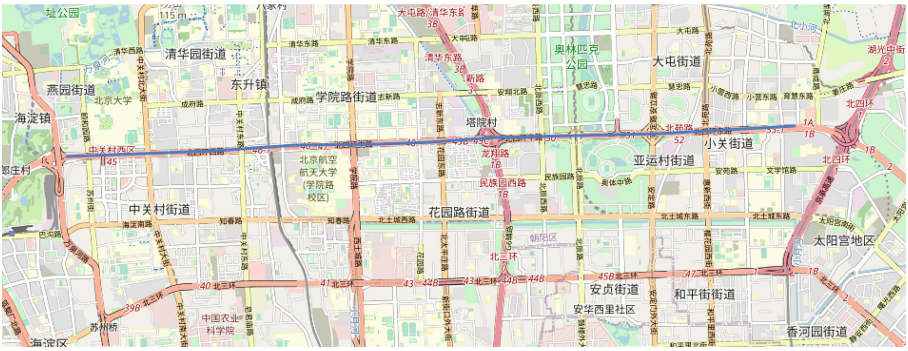


Fig. 6. Short pattern with the length of 11 km (starts on the right).

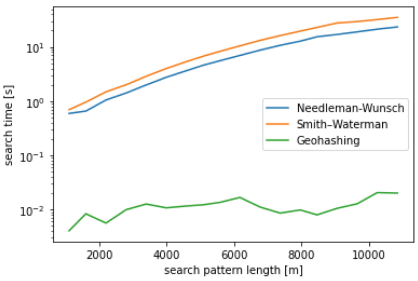
Fig. 7. Long pattern with the length of 39 km (starts on the right).

Table 1. Needleman-Wunsch Algorithm parameters

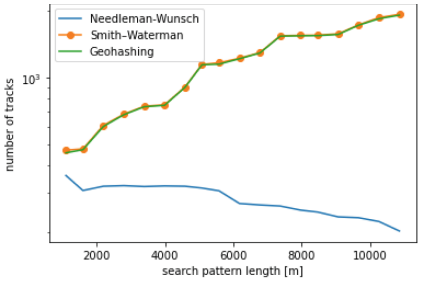
match score	1
mismatch penalty	−1
gap penalty	0
α	$\gamma = 4$
β	3

Geohashes, until full pattern query length is achieved. This is necessary in order to evaluate the scalability of NWA for different query lengths.

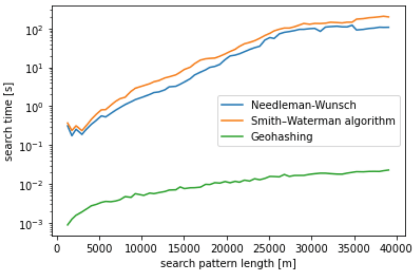
Figures 8(a) and 8(c) show the results for these two search patterns. We can see that as the length of either search pattern increases, so does increase the time to compute similar tracks in the Geolife data set. We consider these results to be unsuitable for real time search, since they are beyond what can be considered as "negligible delay", where Liu and Heer [7] show that a delay of 500 ms already results in decreased user activity (in use cases with high interaction).



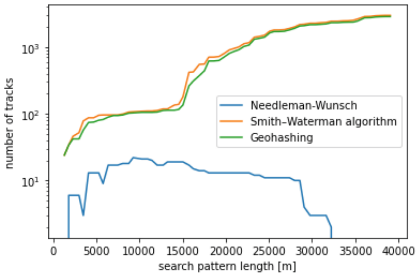
(a) Short pattern by time.



(b) Short pattern by number of tracks.



(c) Long pattern by time.



(d) Long pattern by number of tracks.

Fig. 8. Comparison of different methods based on search time and found tracks.

Figures 8(b) and 8(d) show the number of similar track. As expected, the number of these tracks decreases with the pattern query length.

6.2 Searching with Smith-Waterman Algorithm

For searching using Smith-Waterman Algorithm we used parameters specified in Table 2. We set mismatch and gap penalties to –1, because our goal is to search for maximal alignment. If two tracks have at least γ subsequent matches we call them similar.

Table 2. Smith-Waterman Algorithm parameters.

match score	1
mismatch penalty	–1
gap penalty	–1
subsequent matches	$\gamma = 4$

The experiment is done the same way as in the case with NWA, i.e. the length of pattern query is incrementally increased in order to test with still longer pattern queries.

Figures 8(a) and 8(b) show the results for the two pattern queries. We can again see that the time to compute an alignment cannot be considered negligible. Figures 8(b) and 8(b) show the number of found tracks.

6.3 Searching with Geohashing

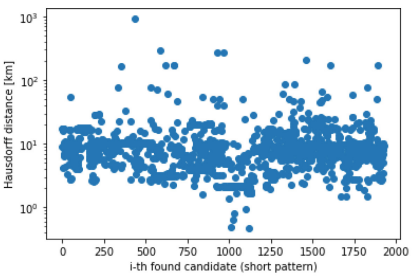
We applied the same search procedure as for NWA and SWA for both the short and long pattern. Figures 8(a) and 8(c) show the time needed to compute r, i.e. the set similar tracks, that still need to get ranked by its similarity to pattern query. We can see that compared to both NWA and SWA, the search time is considerably decreased. We can consider these results as having negligible delay, thus suitable for real time search. Figures 8(a) and 8(c) show the number of similar tracks.

The results for the search by pattern for NWA, SWA and our approach based on geohashing shown in Fig. 8 indicate that the latter approach is a dominant approach in terms of computational time. The results also indicate that SWA and our approach based on geohashing are similar in terms of what they find, i.e. local alignments. Notice that our experimental results are based on a data set with a certain number of measured GPS positions, therefore we are unable to provide statistical significance for our results. We only have two samples for each search pattern length. We are currently focusing on creating a synthetic data set that could resolve this problem [2], however, as a proof of concept the results give a clear direction for future research.

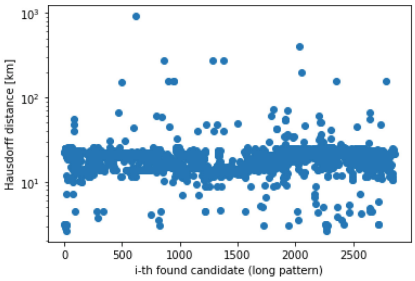
6.4 Search by Pattern Starting with γ Identical Geohashes

We have argued that a general pattern search that compares any search pattern to a set of tracks with equal weight may not be necessary. Therefore we suggested that any found tracks need to start with γ geohashes that are identical to initial γ geohashes of the search pattern.

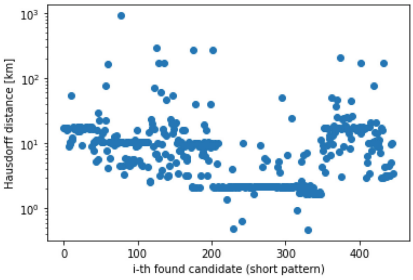
In order to compare the general approach and our approach based on γ identical geohashes, we compute the Hausdorff distance for the results obtained by these two approaches. These results are shown in Figs. 9(a) and 9(b) for the general approach and in Figs. 9(c) and 9(d) for our approach. We can see that requiring that initial γ geohashes are identical has a great potential to further decrease computational burden. Notice that the computed Hausdorff distances may seem large (500m to tens of kilometers), however, the Hausdorff distance is computed as the maximum minimum distance between two sets of points, rather than an average or minimum distance; see Eq. 1 for details. In the case of the long pattern shown in Fig. 9(d), the number of reasonable results has shrunk to just three possibilities.



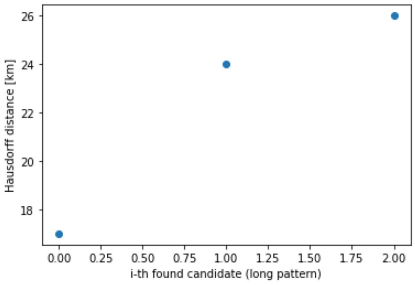
(a) Short pattern, match at any position.



(b) Long pattern, match at any position.



(c) Short pattern, initial γ geohashes are identical.



(d) Long pattern, initial γ geohashes are identical.

Fig. 9. Hausdorff distance for considered tracks using geohashes.

7 Conclusion

We argue that if search by pattern in GPS trajectories is similar to web search, user first draws the most important part of the query, and then influenced by the result, a query refinement may get added. This idea has led us to designing for a specific type of pattern search, where it is not necessary to compare search query with a potentially very large set of recorded trajectories. We pointed out that such a general approach might lead to computing a trie associated with complex pruning rules.

Instead we investigate the option where initial part of pattern query has more weight than any possible tail. We prepare several experiments based on the Geolife data set that contains GPS trajectories recorded in Beijing, China. We show that giving the initial part of a pattern query more weight has a potential to improve query time and thus how user might interact.

There are several challenges and deficiencies of our approach that we would like to stress. Even though the Geolife data set includes GPS trajectories recorded in a major urban area, it does not by far capture the level traffic that happens in extremely dense urban areas. We are currently working on an approach that could generate large scale synthetic data sets of GPS trajectories [2]. A massive data set could then give a final result on efficiency of various search approaches.

Drawing a pattern query is a major challenge since it requires a well-designed user interface that allows for simple pattern drawing, its refinements and adjustments. As simple as it sounds, it is a quite challenging task connected with a lot of effort aimed at user experience testing.

References

1. Baker, R.D.: Mathematical models of confirmation bias (2022). https://doi.org/10.48550/ARXIV.2202.03072, https://arxiv.org/abs/2202.03072
2. Cavojsky, M., Drozda, M.: Gap analysis of cohave, geolife and T-Drive datasets. In: 14th International Scientific Conference on Distance Learning in Applied Informatics (DIVAI), pp. 356–364. Wolters Kluwer (2022). https://www.divai.sk/assets/divai2022.pdf
3. Čavojský, M., Drozda, M.: Comparison of user trajectories with the needleman-wunsch algorithm. In: Yin, Y., Li, Y., Gao, H., Zhang, J. (eds.) MobiCASE 2019. LNICST, vol. 290, pp. 141–154. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28468-8_11
4. Čavojský, M., Drozda, M., Balogh, Z.: Analysis and experimental evaluation of the Needleman-Wunsch algorithm for trajectory comparison. Expert Syst. Appl. **165**, 114068 (2021). https://doi.org/10.1016/j.eswa.2020.114068
5. Dai, H.K., Su, H.C.: On the locality properties of space-filling curves. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 385–394. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24587-2_40
6. Fisher, B.: Edinburgh informatics forum pedestrian database (2010). https://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/. Accessed 11 July 2022
7. Liu, Z., Heer, J.: The effects of interactive latency on exploratory visual analysis. IEEE Trans. Visual Comput. Graphics **20**(12), 2122–2131 (2014)
8. Luo, T., Zheng, X., Xu, G., Fu, K., Ren, W.: An improved DBSCAN algorithm to detect stops in individual trajectories. ISPRS Int. J. Geo Inf. **6**(3), 63 (2017)
9. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001). https://doi.org/10.1145/375360.375365
10. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**(3), 443–453 (1970)
11. Niemeyer, G.: Geohash. http://geohash.org/. Accessed 13 July 2022
12. Plous, S.: The Psychology of Judgment and Decision Making. Mcgraw-Hill Book Company, New York (1993)
13. Rappaport, T.S., et al.: Wireless Communications: Principles and Practice, vol. 2. Prentice Hall PTR, New Jersey (1996)
14. Rucklidge, W. (ed.): The hausdorff distance, vol. 1173, pp. 27–42. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0015093
15. Sellers, P.H.: On the theory and computation of evolutionary distances. SIAM J. Appl. Math. **26**(4), 787–793 (1974)
16. Smith, T., Waterman, M.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981). https://doi.org/10.1016/0022-2836(81)90087-5
17. Yang, Y., Cai, J., Yang, H., Zhang, J., Zhao, X.: Tad: a trajectory clustering algorithm based on spatial-temporal density analysis. Expert Syst. Appl. **139**, 112846 (2020). https://doi.org/10.1016/j.eswa.2019.112846

18. Yin, H., Gao, H., Wang, B., Li, S., Li, J.: Efficient trajectory compression and range query processing. World Wide Web **25**(3), 1259–1285 (2022)
19. Zhao, X., Pi, D., Chen, J.: Novel trajectory privacy-preserving method based on clustering using differential privacy. Expert Syst. Appl. **149**, 113241 (2020). https://doi.org/10.1016/j.eswa.2020.113241
20. Zheng, B., Weng, L., Zhao, X., Zeng, K., Zhou, X., Jensen, C.S.: Repose: distributed top-k trajectory similarity search with local reference point tries. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 708–719. IEEE (2021)
21. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.Y.: Understanding mobility based on GPS data. In: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 312–321. ACM (2008)
22. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–39 (2010)
23. Zheng, Y., Zhang, L., Xie, X., Ma, W.Y.: Mining interesting locations and travel sequences from gps trajectories. In: Proceedings of the 18th International Conference on World Wide Web, pp. 791–800. ACM (2009)