Searching for empirical evidence on traffic equilibrium

S1 Appendix

In this study, we apply the map matching algorithm developed by [1] with small modifications. As the algorithm is originally developed for low-sampling-rate GPS trajectories, we remove/replace certain parts in order to achieve a higher performance. Essentially, the algorithm considers candidate points around each GPS observation (see Supplementary Figure 1), builds a candidate graph that considers all possible sequences (see Supplementary Figure 2) and returns the trajectory with the highest spatiotemporal matching score (see Supplementary Figure 3). The main difference we propose here is the computation of spatial and temporal scores. Following is the spatial score from the original formulation:

\[
F_s(c_{i-1}^t \rightarrow c_i^j) = N(c_i^j | 0, \sigma^2) \cdot \frac{\text{dist}(p_i, p_{i-1})}{\text{path}(c_{i-1}^t, c_i^j)}
\]

where \(c_{i-1}^t\) is the \(t\)th candidate for the previous observation, \(c_i^j\) is the \(j\)th candidate for the current observation, \(N\) is the probability density value for the standard normal distribution with variance \(\sigma^2\), \(\text{dist}(p_i, p_{i-1})\) is the euclidean distance between GPS observations \(p_i\) and \(p_{i-1}\) and \(\text{path}(c_{i-1}^t, c_i^j)\) is the shortest path distance between candidate points \(c_{i-1}^t\) and \(c_i^j\). While the first term indicates how likely a GPS observation can be matched with a candidate point, the second term favors the shortest paths that are similar to straight lines. This assumption is intuitively valid for two GPS points that are far apart (e.g. low-sampling-rate observations); but, as the sampling rate is 10-40s in our data set, we realized that the second term exacerbates the matching quality particularly around intersections. Therefore, removing the second term, the spatial score becomes \(F_s(c_i^j) = N(c_i^j | 0, \sigma^2)\).

[1] considers the cosine distance between the average speed from \(c_{i-1}^t\) to \(c_i^j\) and typical speed values along the roadway links to measure the temporal similarity. Again, bearing in mind the relatively high sampling rate in our data set, we decided to use the temporal information to filter unexpectedly high speed values. First, we calculate the average speed \(\bar{v}\) and the utmost possible speed \(\bar{v}_u\) from \(c_{i-1}^t\) to \(c_i^j\).

\[
\bar{v} = \frac{\text{path}(c_{i-1}^t, c_i^j)}{\Delta_{i-1 \rightarrow i}} , \quad \bar{v}_u = \frac{\sum_{k \in p} (v_k \cdot l_k)}{\sum_{k \in p} l_k}
\]

where \(\Delta_{i-1 \rightarrow i}\) is the time difference between GPS observations, \(p\) is the sequence of links from \(c_{i-1}^t\) to \(c_i^j\), \(v_k\) is the speed limit and \(l_k\) is the length of link \(k\). Note \(\sum_{k \in p} l_k = \text{path}(c_{i-1}^t, c_i^j)\). The temporal similarity score reads as follows.

\[
F_t(c_{i-1}^t \rightarrow c_i^j) = \begin{cases} 
1 & \bar{v} \leq \bar{v}_u \\
1 - \left(\frac{\bar{v} - \bar{v}_u}{\alpha \cdot \bar{v}_u}\right)^2 & \bar{v}_u < \bar{v} \leq (\alpha + 1) \cdot \bar{v}_u \\
0 & \bar{v}_u < \bar{v}
\end{cases}
\]

where \(\alpha\) is a coefficient that defines the allowed speed observation and \(F_t\) takes values between 0 and 1. Finally, the spatial similarity score is \(F(c_{i-1}^t \rightarrow c_i^j) = F_s(c_i^j) \cdot F_t(c_{i-1}^t \rightarrow c_i^j)\). For
Algorithm 1: ST Matching

**Input:** Road network $G$, a trajectory $T: p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_n$

**Output:** The matched sequence $P: c_{i1}^1 \rightarrow c_{i2}^2 \rightarrow \ldots \rightarrow c_{in}^n$

1. Initialize $tList \leftarrow \emptyset$
2. foreach 1 to $n$ do
   3. $s = \text{GetCandidates}(p_i, G, r)$ // candidates within radius $r$
   4. $tList.add(s)$ // expand the list
   5. $G_T = \text{ConstructGraph}(tList)$ // build graph using all possible sequences (see Supplementary Figure 2)
5. end
6. return $\text{FindMatchedSequence}(G_T)$

Figure 1: Algorithm for spatio-temporal map matching

Further details, please see Supplementary Figure 1-3. The parameters used in the algorithm are $r=100[m]$, $\sigma=20[m]$ and $\alpha=0.3$.

Figure 2: Candidate graph ($G_T$) [1]
Algorithm 2: FindMatchedSequence

**Input:** Candidate graph $G_T$

**Output:** The sequence with the highest score $P : c^1 \rightarrow c^2 \rightarrow ... \rightarrow c^n$

1. Initialize $f = []$ // the highest score so far
2. Initialize $pre = []$ // the parent of the current candidate
3. foreach $c^j_1$ do
   4. $x^j_1 = \text{dist}(c^j_1, p_1)$ // euclidean distance between $p_1$ and $c^j_1$
   5. $f[c^j_1] = N(x^j_1|0, \sigma^2)$ // observation probability based on standard normal distribution with std. dev. $\sigma$
4. end
5. foreach $i=2$ to $n$ do
   6. foreach $c^j_i$ do
      7. $max = -\infty$
      8. foreach $c^j_{i-1}$ do
         9. $alt = f[c^j_{i-1}] + F(c^j_{i-1} \rightarrow c^j_i)$ // add spatiotemporal score $F$
         10. if $alt > max$ then
             11. $max = alt$
             12. $pre[c^j_i] = c^j_{i-1}$
             13. $f[c^j_i] = max$
      14. end
   15. end
7. Initialize $rList \leftarrow \emptyset$
8. $c = \text{argmax}_{c^n}(f[c^n])$
9. foreach $i=n$ to $2$ do
   10. $rList.add(c)$
   11. $c = pre[c]$
12. end
13. $rList.add(c)$
14. return $\text{FindMatchedSequence}(G_T)$

Figure 3: Algorithm for matching sequences

References