CCCG 2018

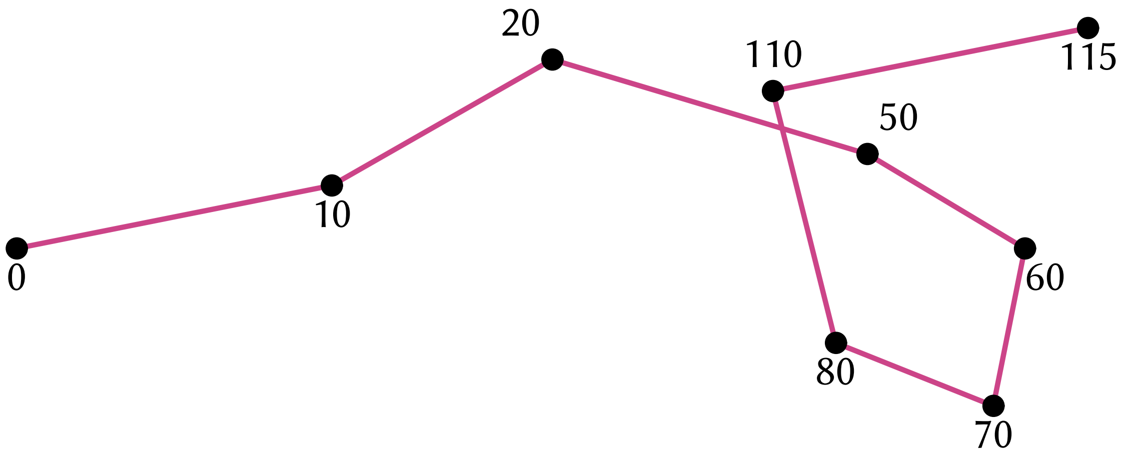
Looking for Bird Nests

Identifying Stay Points with Bounded Gaps

Ali Gholami Rudi

Babol Noshirvani University of Technology

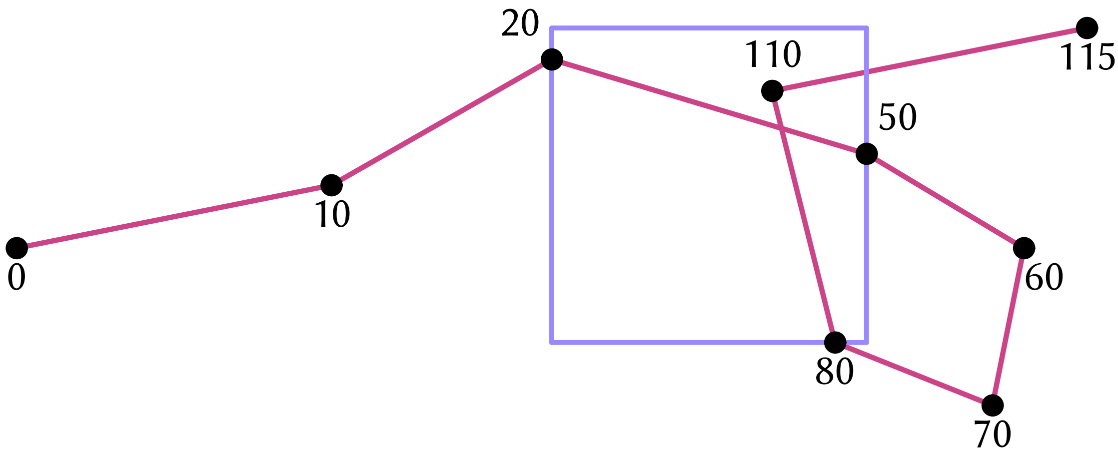
Modeling Trajectories



Vertices: locations with timestamps

Edges: constant speed, straight line

Hotspots, Popular Places, Stay Points



Stay points: where an entity spends a significant amount of time.

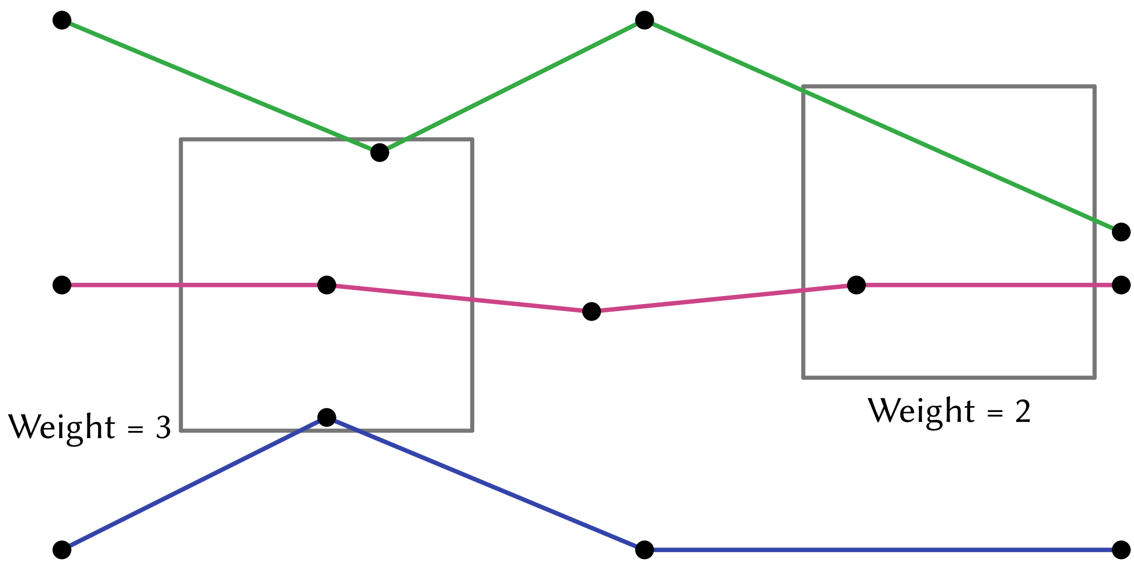
Shape: axis-aligned square

Criteria:

- Number of visits (Benkert, Djordjevic, Gudmundsson, Wolle; 2010)
- Duration of visits (Gudmundsson, van Kreveld, and Staals; 2013)

Based on the Number of Visits

Benkert, Djordjevic, Gudmundsson, Wolle (2010)



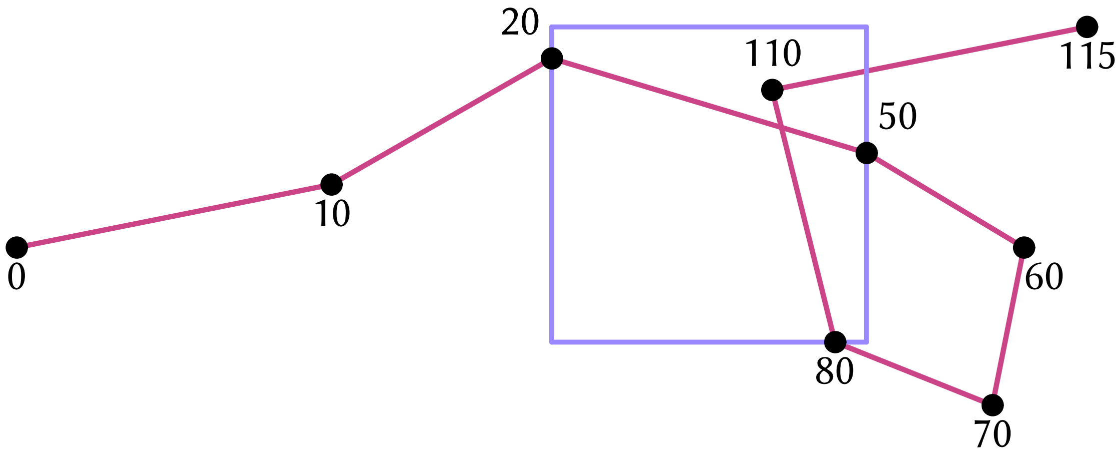
Discrete model: $O(n \log n)$.

Continous model: $O(n^2)$.

Based on the Durations of Visits

Gudmundsson, van Kreveld, Staals (2013)

Maximum or total visit duration



Stay Points with Bounded Gaps

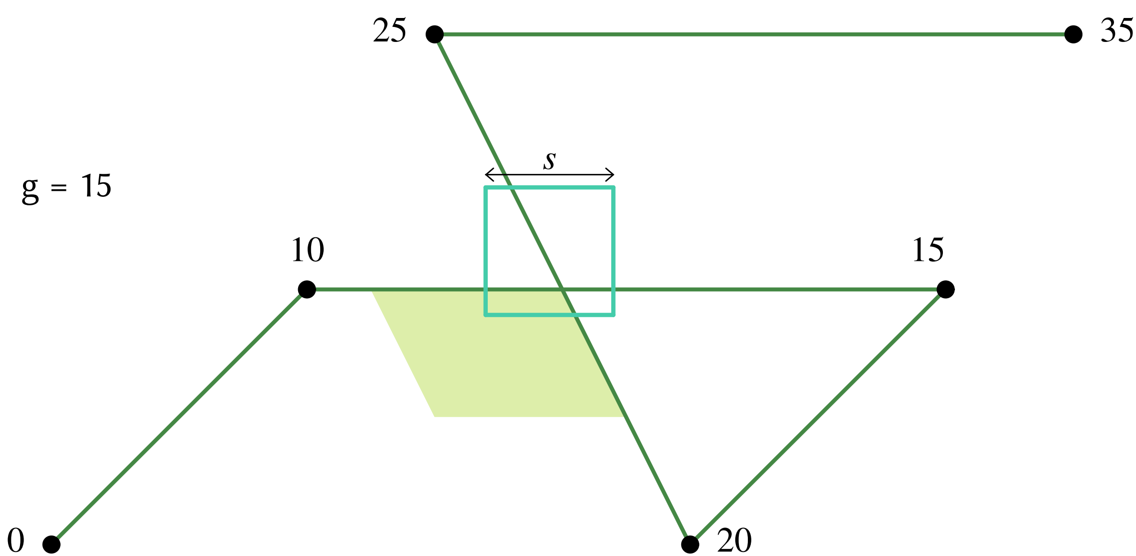
Allowing the entity to leave the region for short intervals

- A bird returning to its nest to feed its chicks.
- Leaving the cinema for the bathroom

Finding such stay points among multiple intersting places

- Arboleda, Bogorny, Patio (2017)
- Potential stay points (intersting places) are given as input

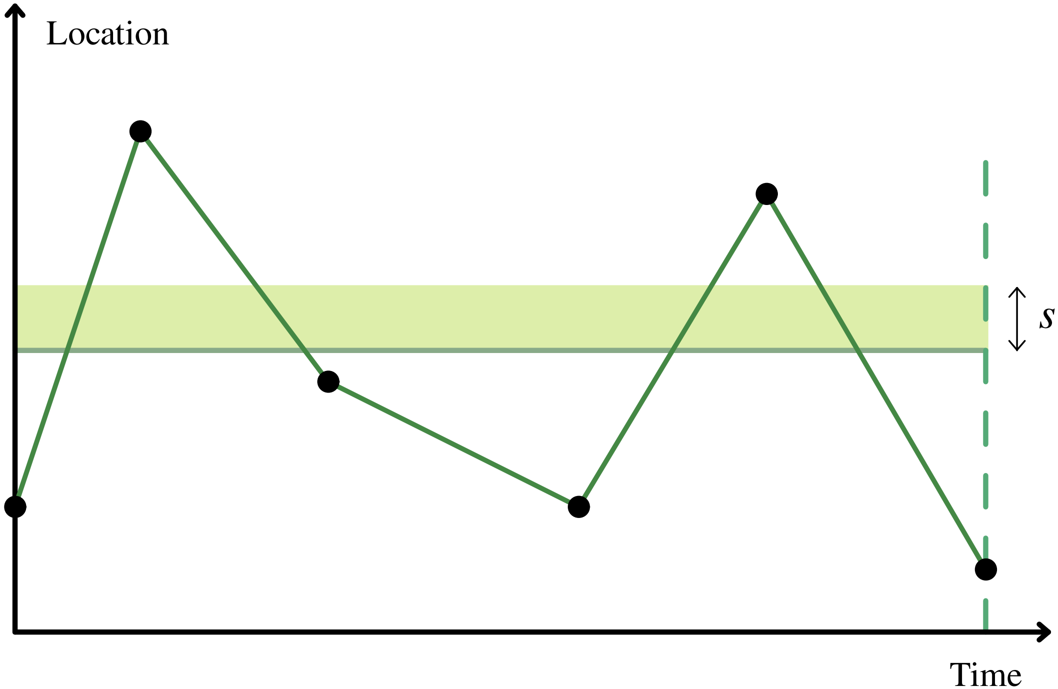
Our Goals



Stay points: axis-aligned squares with fixed side length (s)
- The entity should never be outside the regions for
  - more than the maximum allowed absence time (g)

Stay maps: the lower left corner of all stay points

One-Dimensional Trajectories



Lemma: The stay map of a trajectory in R^1 is continuous

Exact Algorithm for 1D

Event points: points in R^1

i) a trajectory vertex lies on that point

ii) the time gap between two visits to that point is exactly g

Lemma: The set of event points of a trajectory can be computed in $O(n \log n)$ time.

Sweeping the time-location plane vertically.

Lemma: The stay map of a trajectory in R^1 starts and ends at an event point or at distance s from one.

Otherwise, we can move the leftmost (similarly, rightmost) stay point to the left (right) to obtain a new leftmost

Exact Algorithm for 1D

Lemma: we can answer in $O(n)$ time whether a point is in the stay map or not, and if not, whether the stay map is on its left side or on its right side.

Algorithm for trajectories in R^1 (with the time complexity $O(n \log n)$):

- Obtain the event points and points at distance s from them.
- Perform a binary search to find the left end point.
- Perform a binary search to find the right end point.

Notation:

- $T(a, b)$: the sub-trajectory from time a to time b
- $P(a, b)$: the lowest left corners of all squares of side length s that contain any part of $T(a, b)$.
- $M(0, t)$: the stay map of $T(0, t)$.

Exact 2D Stay Maps — Algorithm

Algorithm: incrementally compute $M(0, D)$

- Let $M(0, g) = P(0, g)$;
  $P(0, g)$ is the union of polygons $P(u, v)$ for all edges uv in $T(0, g)$.
- Compute $M(0, b)$ from $M(0, a)$, in which $M(0, a)$ is the last computed stay map and b is the smallest value after a such that $b - g$ or b is the timestamp of a trajectory vertex.
  - Let V be the difference between $M(0, a)$ and $M(0, b)$; we compute V to obtain $M(0, b)$.

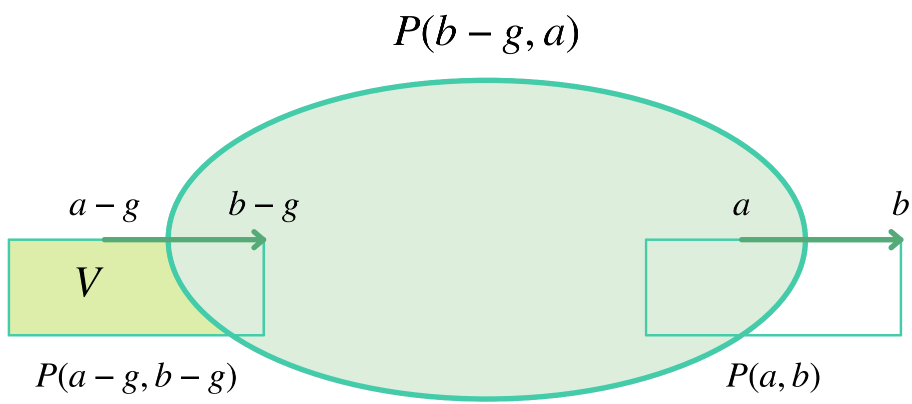
Exact 2D Stay Maps — V

Need to compute V, the difference between $M(0, a)$ and $M(0, b)$.

- Let $V = V' \setminus P(b - g, a)$, where

$$V' = \bigcup_{0 \le \delta \le g} P(a - g, a - g + \delta) \setminus (P(a - g + \delta, b - g) \cup P(a, a + \delta))$$

- The shape of V' depends on $T(a, b)$ and $T(a - g, b - g)$.



- If $P(a - g, b - g)$ and $P(a, b)$ are disjoint, $V' = P(a - g, b - g)$.

Exact 2D Stay Maps — Complexity

- $P(b - g, a)$ is the union of $O(n)$ simple polygons.
- The union of the differences (V) for all iterations of the algorithm, containing $O(n^2)$ simple polygons.
- An $O(n^2)$ implementation seems unlikely.

Approximate 2D Stay Maps — Defi.

Definition:

- $(1+\varepsilon)$-approximate stay point: the entity is never outside the region for more than $g + \varepsilon g$ time.
- $(1+\varepsilon)$-approximate stay map: all exact stay points and possibly some of its $(1+\varepsilon)$-approximate stay points.

A snapshot: $P(t, t+g)$, in which $0 \leq t \leq D - g$

The lowest left corner of every stay point should appear in each snapshot.

Approximation algorithm:

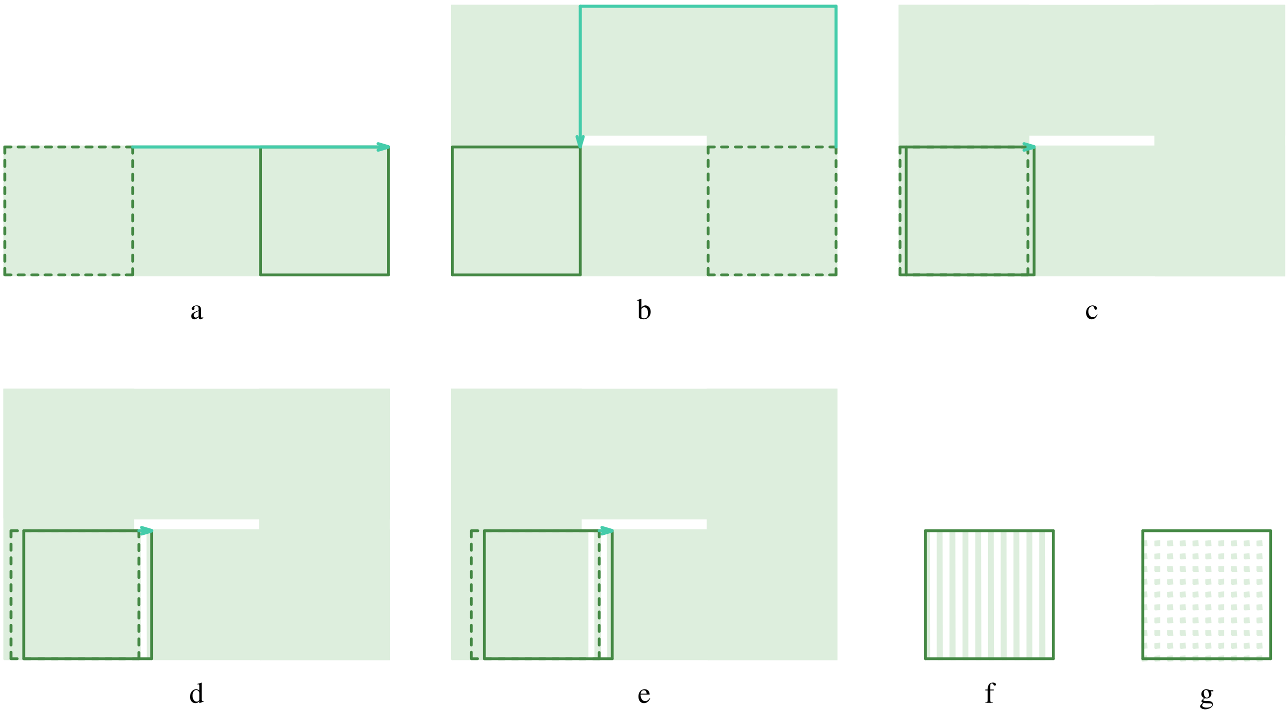
- Let $\lambda = \varepsilon g$
- Compute snapshots $P(t, t+g)$ for $t = i\lambda$, for integral values of i from 0 to D/λ.
- Compute the intersection of these snapshots.

Approximate 2D Stay Maps — Analysis 16

The output contains the lowest left corner of every stay point.

For every square whose lowest left corner is in the output:
the entity cannot be outside for more than $g + \varepsilon g$

Output Complexity



Multi-trajectory Stay Points

Multi-trajectory stay maps

- Each stay point is visited by at least one of the entities
  - in any interval of duration g