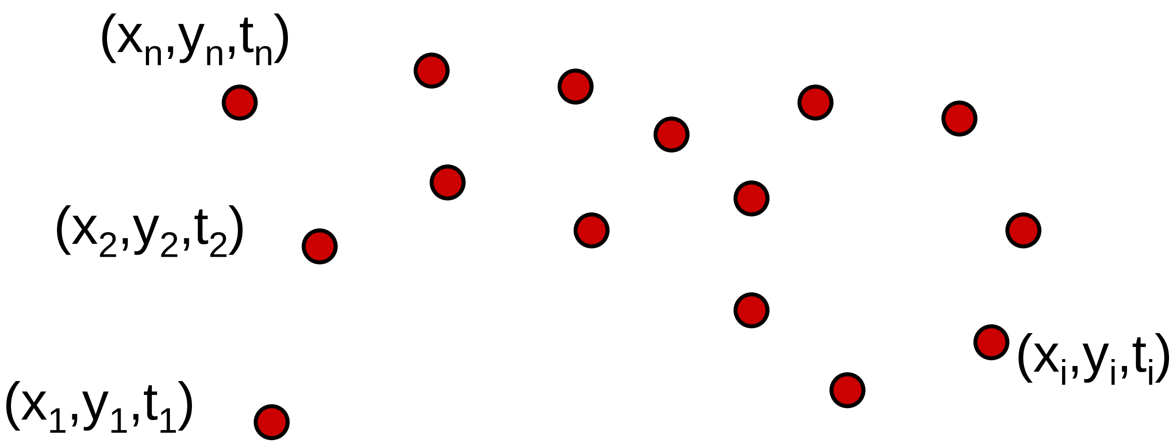
2IL76
Algorithms for Geographic Data

Spring 2015

Lecture 5: Movement Patterns

Trajectory data

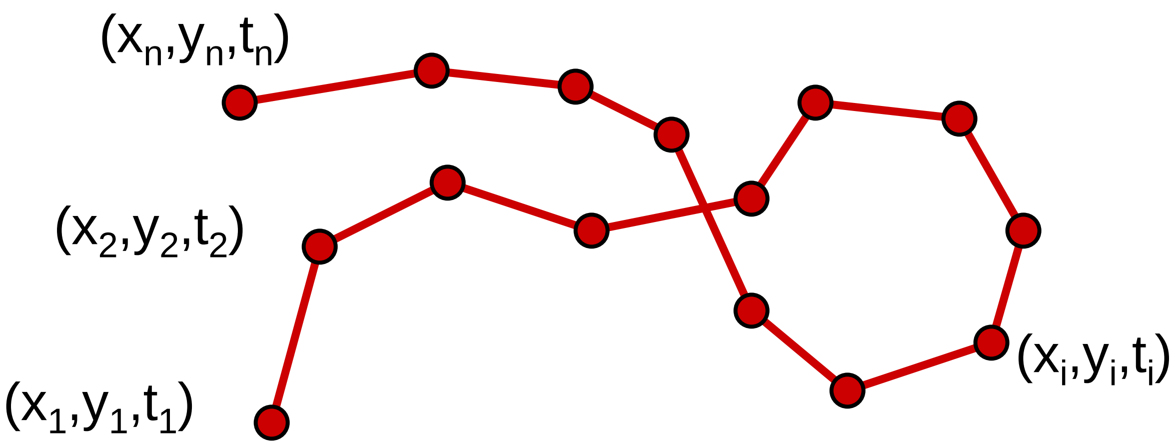
- The data as it is acquired by GPS:
  sequence of triples (spatial plus time-stamp);
  quadruples for trajectories in 3D



Trajectory data

- Typical assumption for sufficiently densely sampled data: constant velocity between consecutive samples

  ➡ velocity/speed is a piecewise constant function



Abstract / general purpose questions

Single trajectory

- simplification, cleaning
- segmentation into semantically meaningful parts
- finding recurring patterns (repeated subtrajectories)

Two trajectories

- similarity computation
- subtrajectory similarity

Multiple trajectories

- clustering, outliers
- flocking/grouping pattern detection
- finding a typical trajectory or computing a mean/median trajectory
- visualization

Movement patterns

Many, many possible movement patterns:

- flocks (group of entities moving close together)
- swarm
- convoys
- herds

- following (is an entity following another entity)
- leadership
- single file
- popular places (place visited by many)
- …

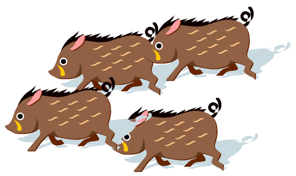
Challenge: useful definitions which are algorithmically tractable

Groups

Groups

- Kalnis, Mamoulis & Bakiras, 2005
- Gudmundsson & van Kreveld, 2006
- Benkert, Gudmundsson, Hubner & Wolle, 2006
- Jensen, Lin & Ooi, 2007
- Al-Naymat, Chawla & Gudmundsson, 2007
- Vieria, Bakalov & Tsotras, 2009
- Jeung, Yiu, Zhou, Jensen & Shen, 2008
- Jeung, Shen & Zhou, 2008
- ...

t_2



t_3

t_4

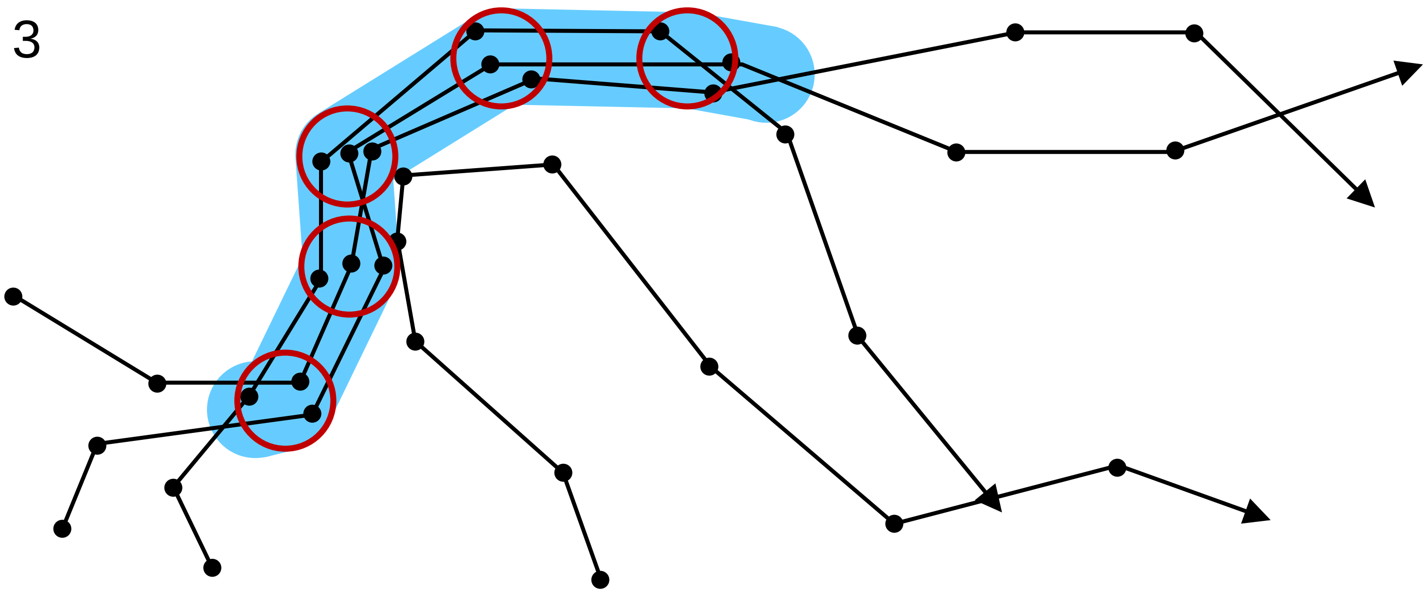
t_1

Groups

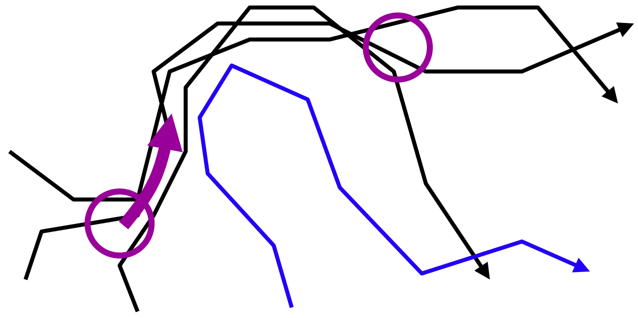
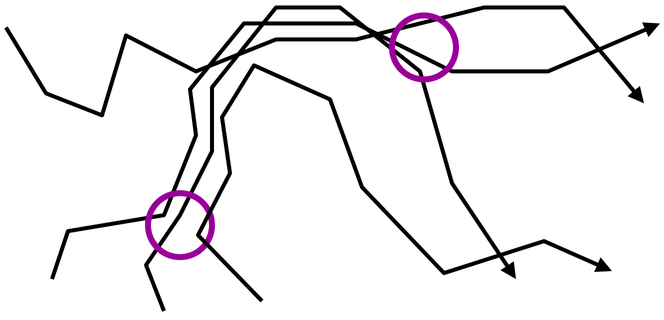
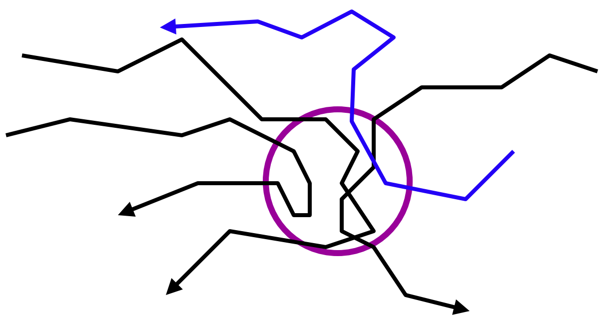
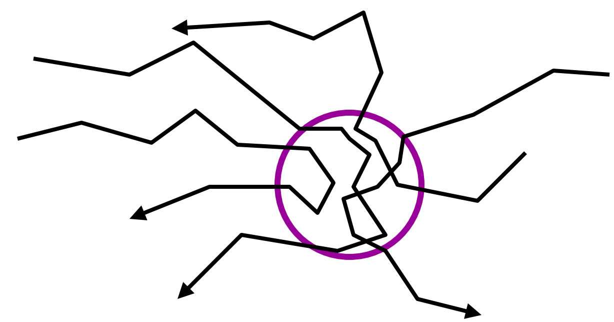
m group size

Time = 0 1 2 3 4 5 6 7 8

m = 3



Groups

	fixed subset	variable subset
flock		
meet		

examples for m = 3

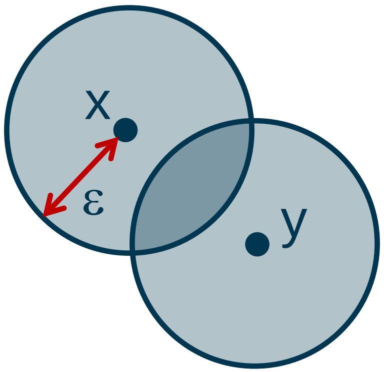
Convoy

Convoy

X set of entities

m convoy size

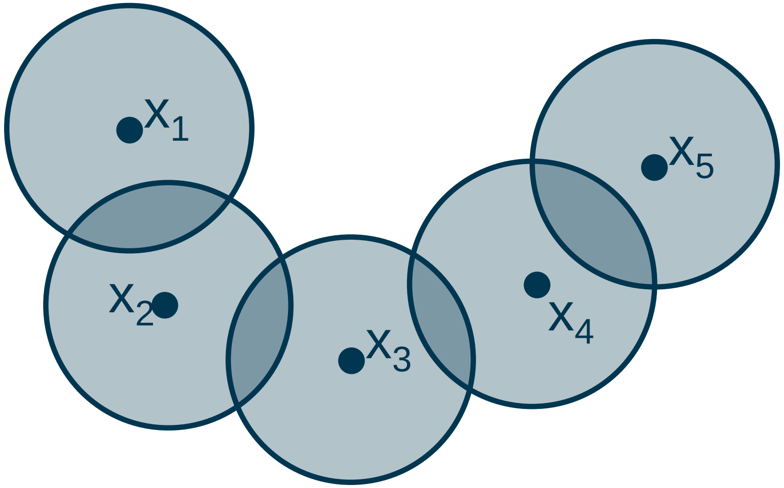
ε distance threshold



Definitions

x, $y \in X$ are directly connected if the ε-disks of x and y intersect

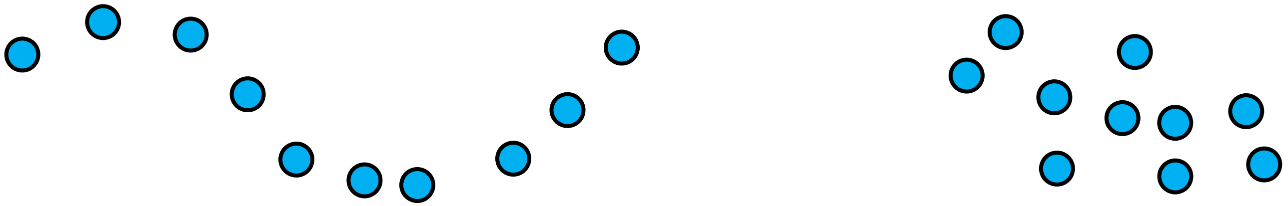
x_1 and x_k are ε-connected if there is a sequence $x_1, x_2, \ldots, x_k$ of entities such that for all i, x_i and x_{i+1} are directly connected



Convoy

Definition

A group of entities forms a convoy if every pair of entities is ε-connected.



[Jeung, Yiu, Zhou, Jensen & Shen, 2008]
[Jeung, Shen & Zhou, 2008]

Leadership & Followers

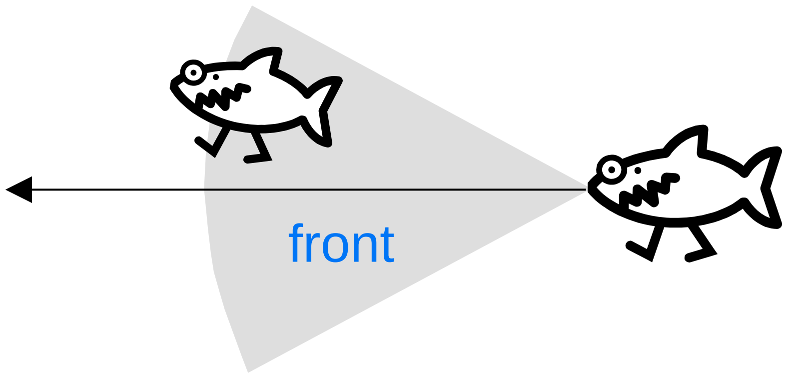
Leadership & Followers

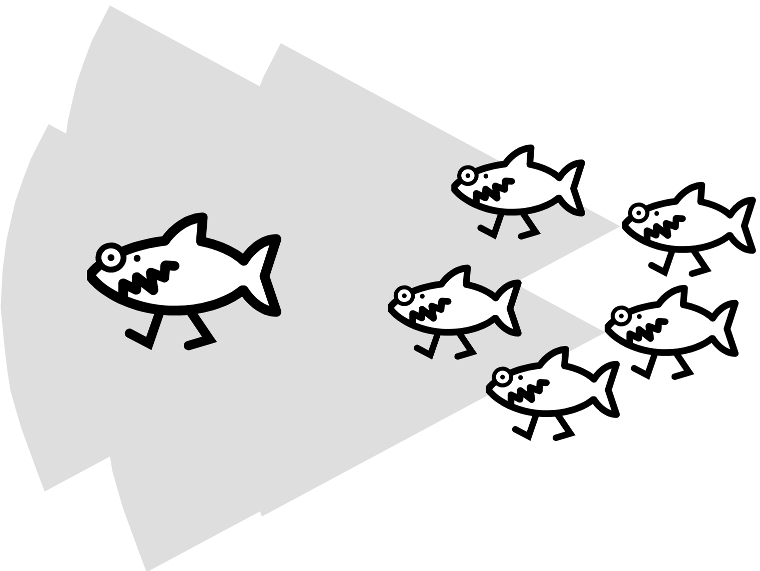
A leader?

Should not follow anyone else!

Is followed by at least m other entities …

For a certain duration …



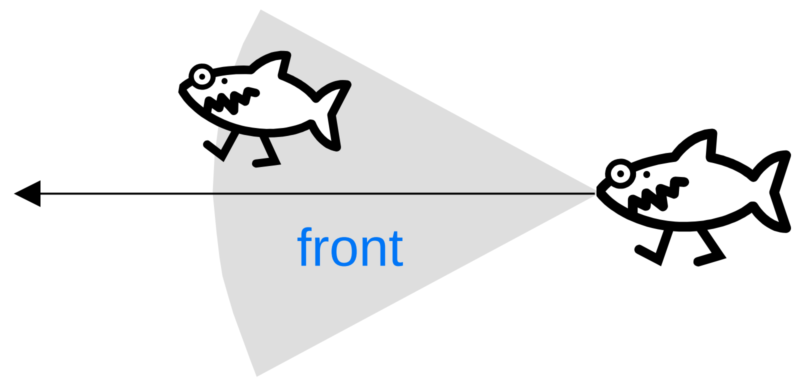


Leadership & Followers

A leader?

Should not follow anyone else!

Is followed by at least m other entities …



For a certain duration …

Many different settings …

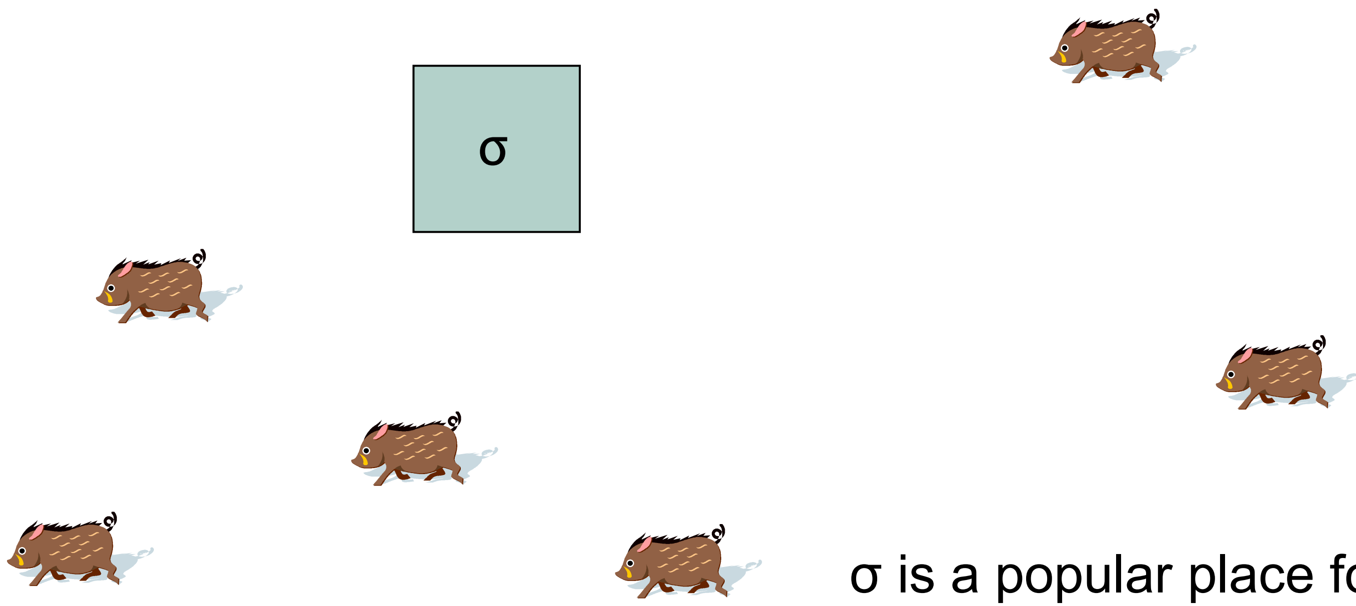
Running time ~$O(n^2 \, t \log n \, t)$ for n entities and t time steps

[Andersson et al. 2007]

Popular places

Popular places

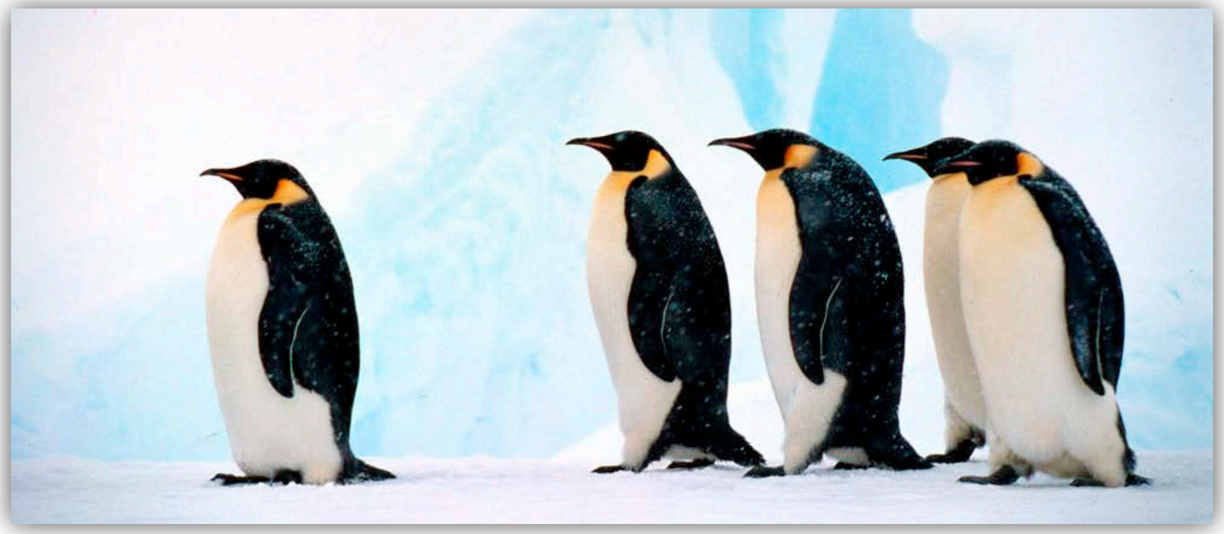
- A region is a popular place if at least m entities visit it



σ is a popular place for
m ≤ 5

[Benkert, Djordjevic, Gudmundsson & Wolle 2007]

Single File



Single file

Single file

intuitively easy to define … but hard to define formally!

Single file

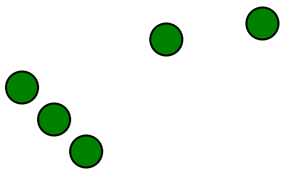
Single file

intuitively easy to define … but hard to define formally!

Single file

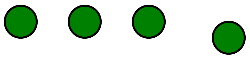
Single file

intuitively easy to define … but hard to define formally!



Towards a formal definition?

- Entities x_1, … , x_m are moving in single file for a given time interval if during this time each entity x_{j+1} is following behind entity x_j for j = 1, … ,m-1



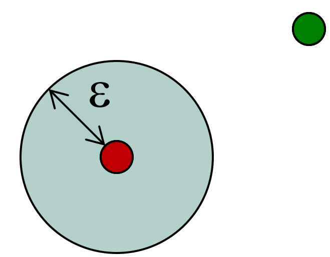
Towards a formal definition?

- Entities $x_1, \ldots, x_m$ are moving in single file for a given time interval if during this time each entity x_{j+1} is following behind entity x_j for $j = 1, \ldots, m-1$

  Following behind?

  Time t

  Time $t' \in [t+\tau_{min}, t+\tau_{max}]$



TU/e Technische Universiteit Eindhoven University of Technology

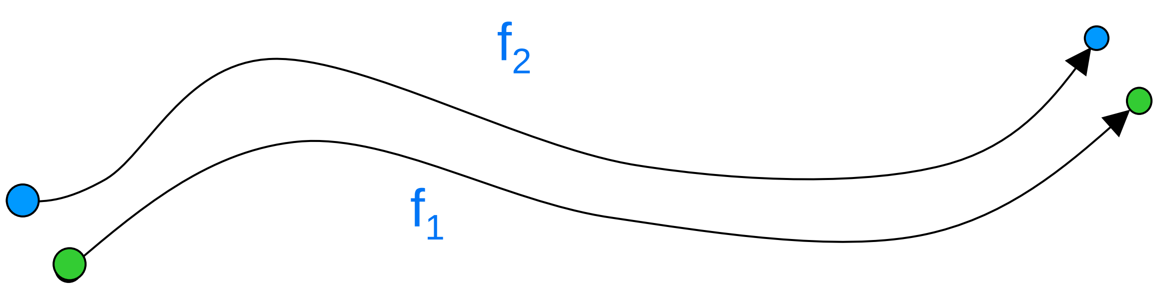
Following behind

x_1 entity with parameterized trajectory f_1 over time interval $[s_1, t_1]$
x_2 entity with parameterized trajectory f_2 over time interval $[s_2, t_2]$

with $s_2 \in [s_1+t_{min}, s_1+t_{max}]$ and $t_2 \in [t_1+t_{min}, t_1+t_{max}]$

x_2 is following behind x_1 in $[s_1,t_1]$ if there exists a continuous, bijective function $\sigma : [s_1, t_1] \rightarrow [s_2, t_2]$ such that $\sigma(s_1)=s_2$ and

$$\forall\, t \in [s_1, t_1]: \sigma(t) \in [t-t_{max}, t-t_{min}] \;\&\; d(f_1(\sigma(t)), f_2(t)) \le \varepsilon$$



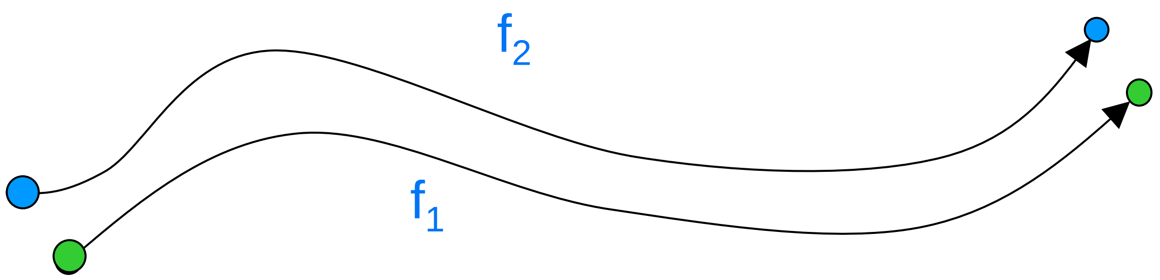
Following behind

x_1 entity with parameterized trajectory f_1 over time interval $[s_1, t_1]$
x_2 entity with parameterized trajectory f_2 over time interval $[s_2, t_2]$

with $s_2 \in [s_1+t_{min}, s_1+t_{max}]$ and $t_2 \in [t_1+t_{min}, t_1+t_{max}]$

x_2 is following behind x_1 in $[s_1,t_1]$ if there exists a continuous, bijective function $\sigma : [s_1, t_1] \rightarrow [s_2, t_2]$ such that $\sigma(s_1)=s_2$ and

$$\forall\, t \in [s_1, t_1]\!: \sigma(t) \in [t-t_{max}, t-t_{min}]\ \&\ d(f_1(\sigma(t)), f_2(t)) \le \varepsilon$$



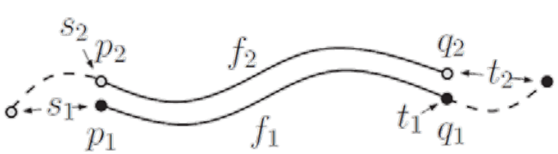
Following behind

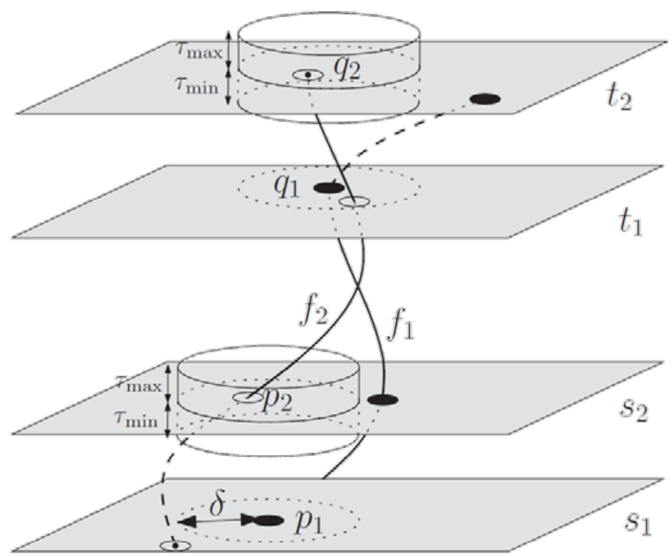
x_1 entity with parameterized trajectory f_1 over time interval $[s_1, t_1]$
x_2 entity with parameterized trajectory f_2 over time interval $[s_2, t_2]$

with $s_2 \in [s_1+t_{min}, s_1+t_{max}]$ and $t_2 \in [t_1+t_{min}, t_1+t_{max}]$

x_2 is following behind x_1 in $[s_1,t_1]$ if there exists a continuous, bijective function $\sigma : [s_1, t_1] \rightarrow [s_2, t_2]$ such that $\sigma(s_1)=s_2$ and

$$\forall\, t \in [s_1, t_1]: \sigma(t) \in [t-t_{max}, t-t_{min}] \;\&\; d(f_1(\sigma(t)), f_2(t)) \leq \varepsilon$$

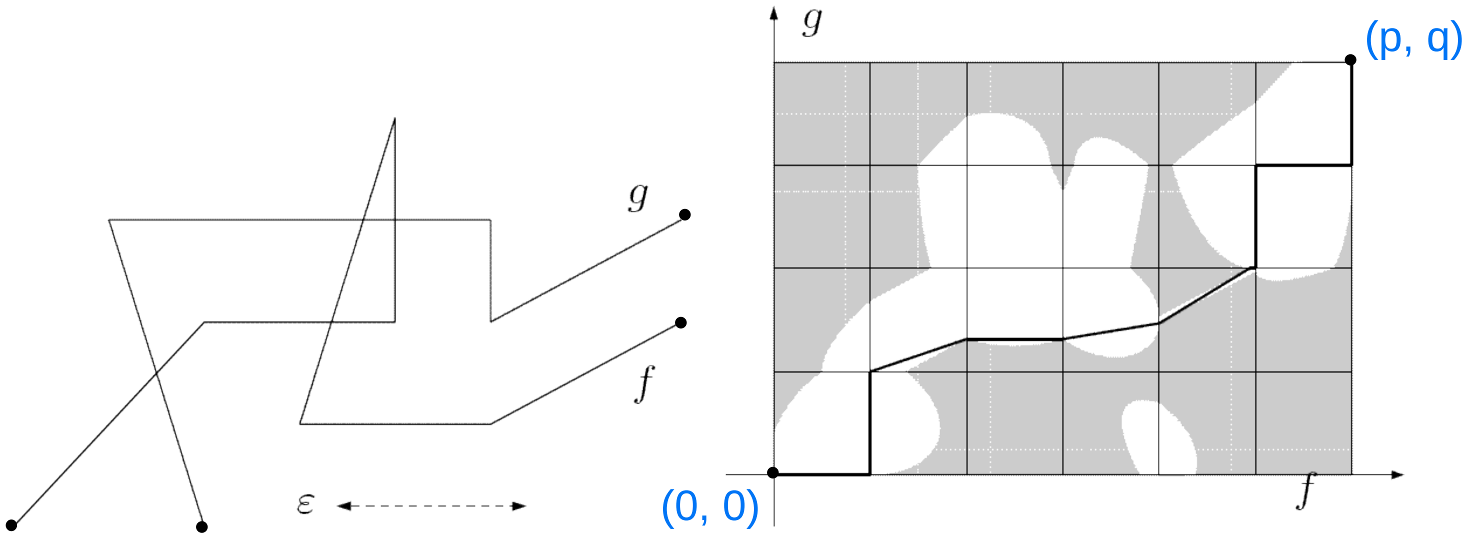




Free space diagram

Definition

If there exists a monotone path in the free-space diagram from (0, 0) to (p, q) which is monotone in both coordinates, then curves P and Q have Fréchet distance less than or equal to ε



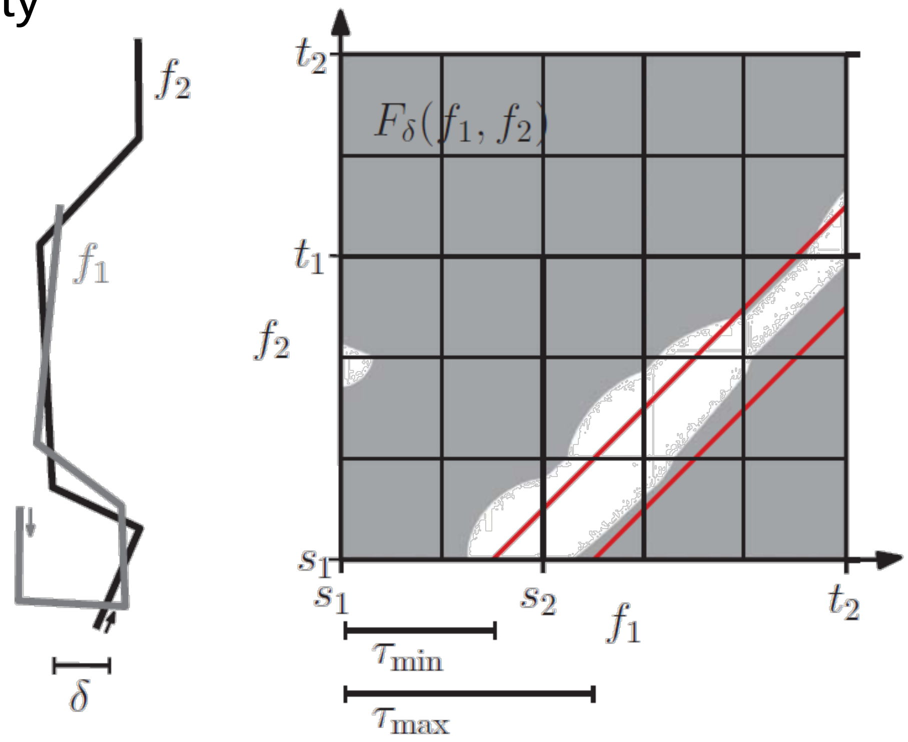
TU/e Technische Universiteit Eindhoven University of Technology

Free space diagram and following behind

- time delay ➡ the path will be restricted to a "diagonal" strip

Running time

O(t+k) where k is the complexity of the diagonal strip



Single file

- One can determine in O($t\ k^2$) time during which time intervals one trajectory is following behind the other.

- If the order between the entities is specified then compute the free space diagram between every pair of consecutive entities

  ➡ O($n\ t\ k^2$)

- If no order then compute the free space diagram between every pair of entities

  ➡ O($n^2\ t\ k^2$)

[Buchin et al. 2008]

Grouping Structure

Grouping Structure

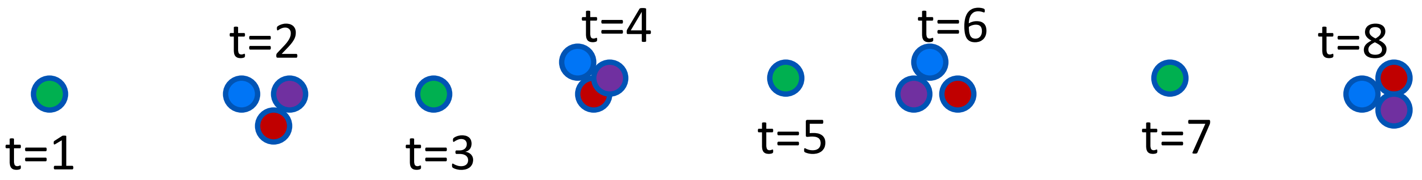
Question

How does one define and compute the ensemble of moving entities forming groups, merging with other groups, splitting into subgroups?

… define and compute … ➡ formalization + algorithms

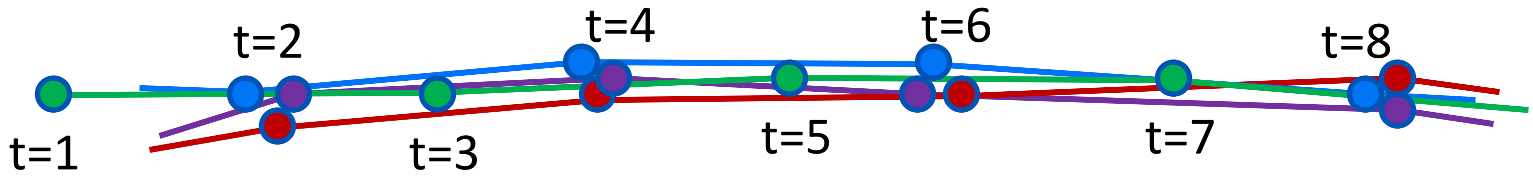
Previous work

- Flocks [Gudmundsson, Laube, Wolle, Speckmann, …. (2005-)]
- Herds [Huang, Chen, Dong (2008)]
- Convoys [Jeung, Yiu, Zhou, Jensen, Shen (2008); Aung, Tan (2010)]
- Swarms [Li, Ding, Han, Kays (2010)]
- Moving groups/clusters [Kalnis, Mamoulis, Bakiras (2005); Wang, Lim, Hwang (2008); Li, Ding, Han, Kays (2010)]



This approach

- Use whole trajectory (interpolated) instead of discrete time stamps only (as opposed to herds, swarms, convoys, …)
- Study the whole grouping structure with merging, splitting, … (as opposed to finding flocks)
- Use a mathematically clean model
- Complexity and efficiency analysis
- Implementation and testing for plausibility



Grouping

Grouping

Grouping

- Three criteria for a group:
  - big enough (size m)
  - close enough (inter-distance d)
  - long enough (duration δ)

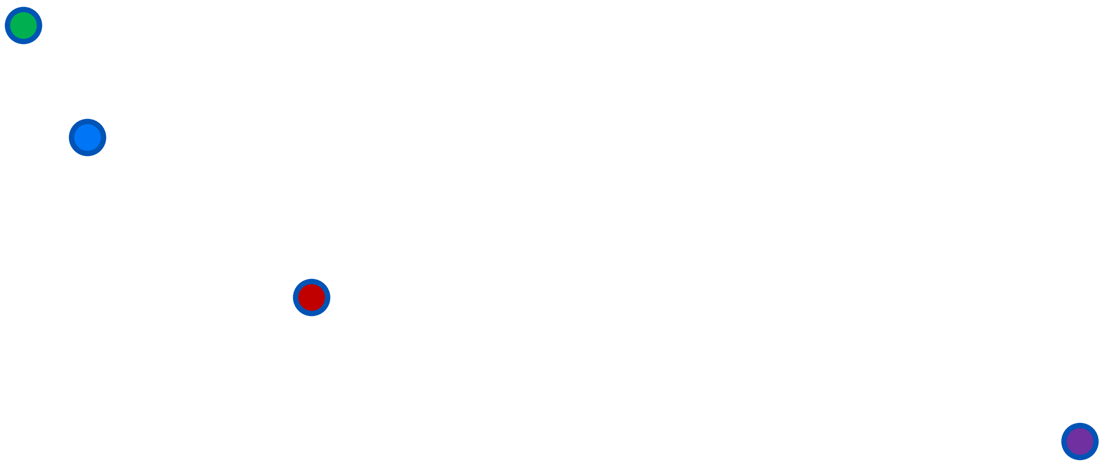
- Only maximal groups are relevant

  Otherwise, assuming m=4, if 8 entities form a group during δ (or longer), then also all 162 subgroups of size at least 4 during that same time interval

  (*maximal in group size, starting time or ending time*)

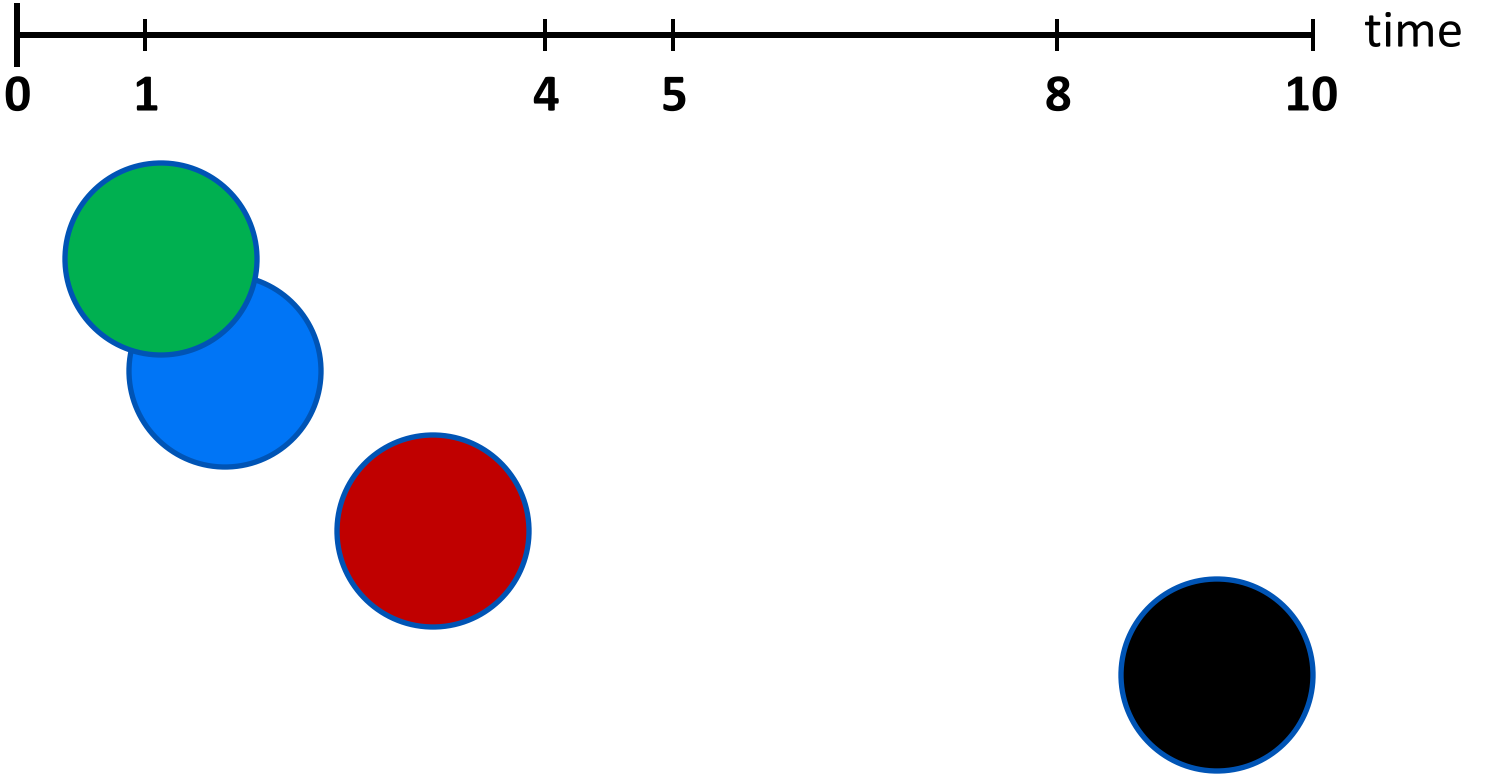
Grouping

- Trace the connected components of moving disks whose radius is half the specified inter-distance, d/2



Grouping

- Trace the connected components of moving disks whose radius is half the specified inter-distance, d/2



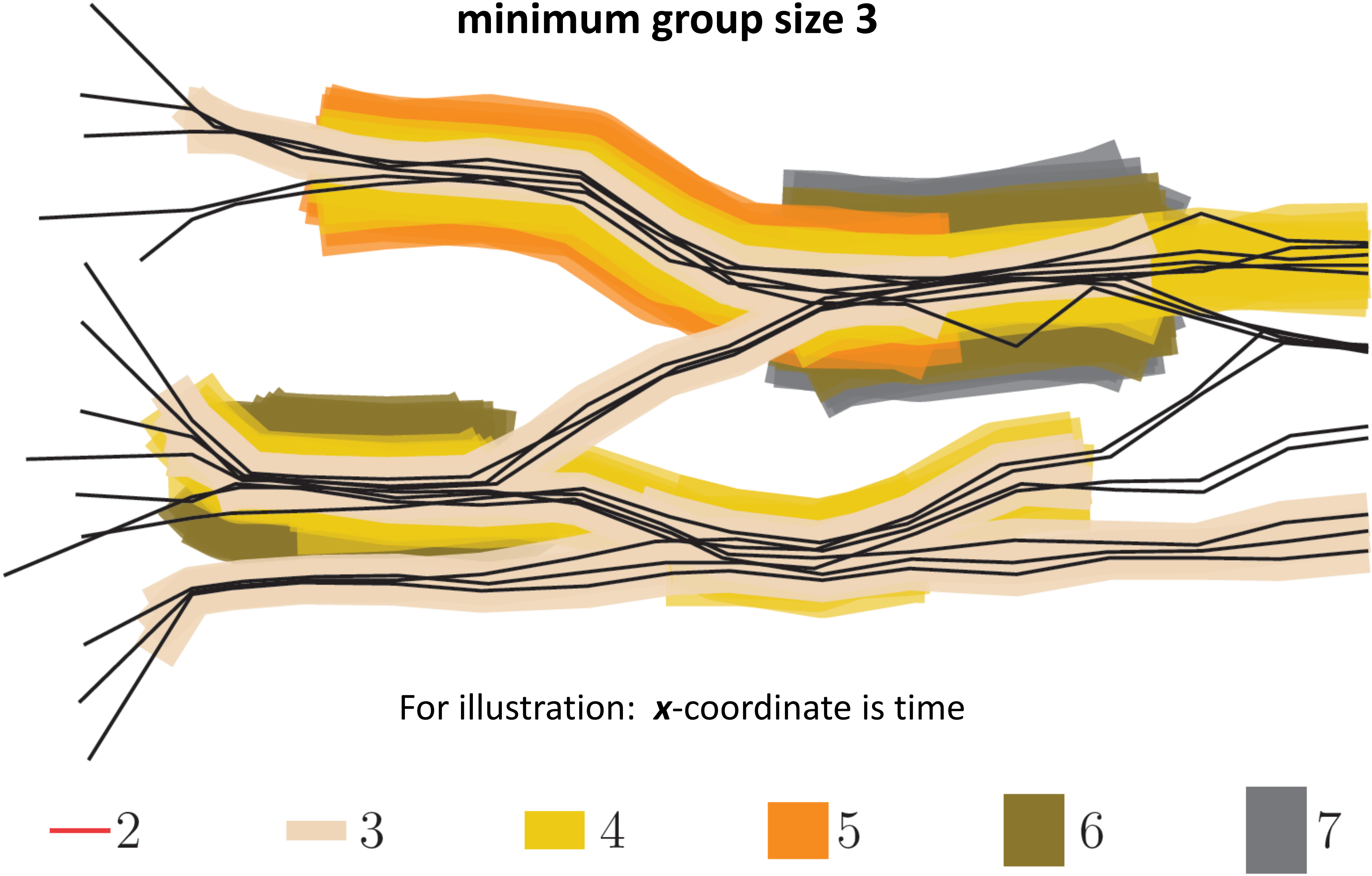
Grouping

- Maximal groups (m=2, δ =3):
  - { green, blue }: [0-4]
  - { green, blue, red }: [1-4]
  - { blue, red}: [1-5]
  - { green, purple }: [8-10]

- Maximal groups (m=3, δ =3):
  - { green, blue, red }: [1-4]

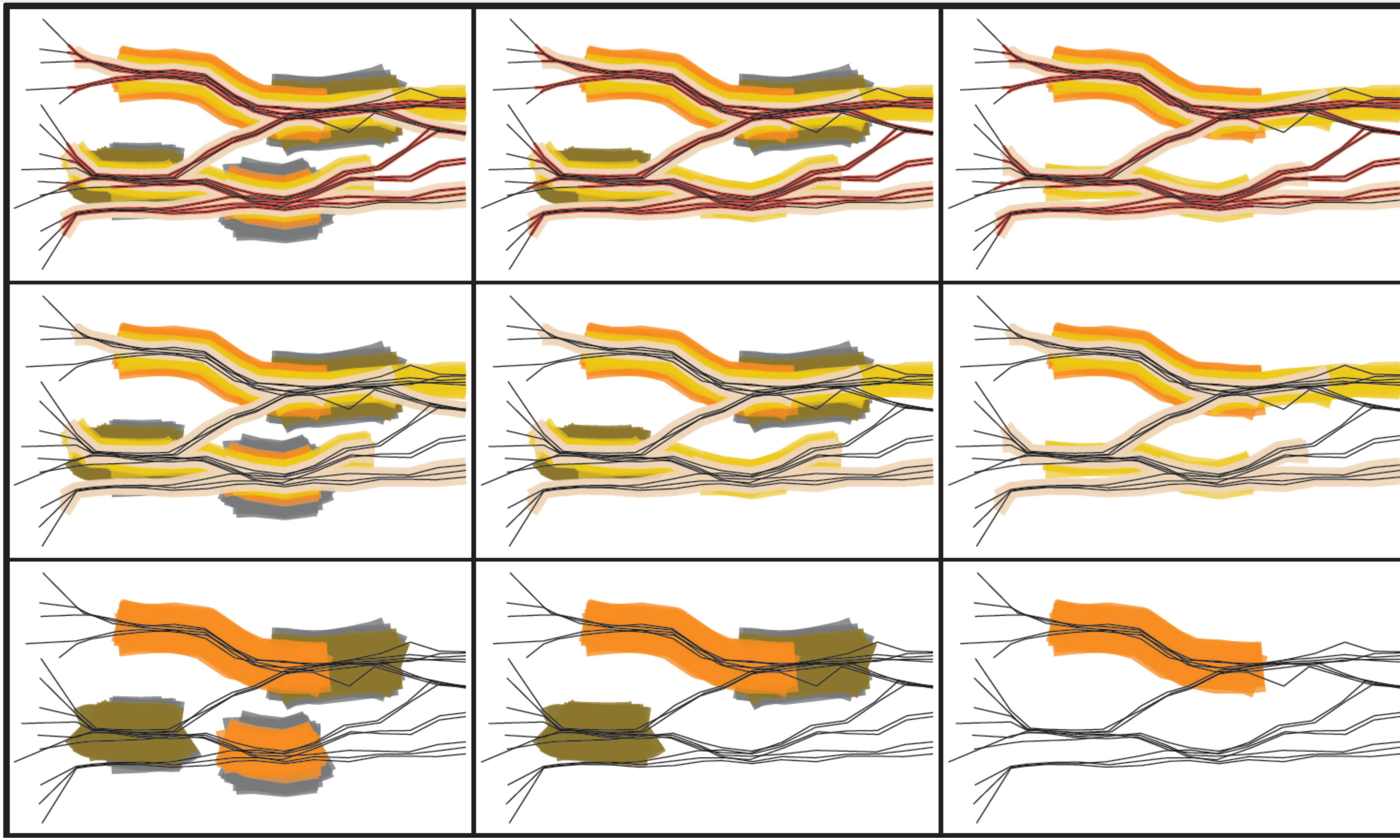
- Maximal groups (m=2, δ =4):
  - { green, blue }: [0-4]
  - { blue, red}: [1-5]

minimum group size 3



For illustration: ***x***-coordinate is time

Technische Universiteit Eindhoven University of Technology

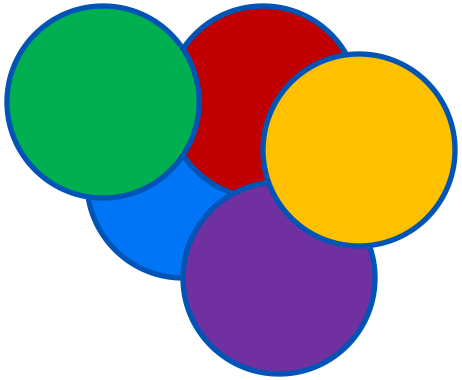


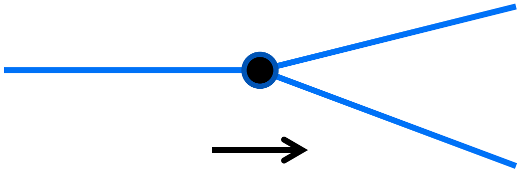
Grouping Structure

Reeb graph (from computational topology)

structure that captures the changes in connectivity of a process, using a graph

- Edges are connected components
- Vertices changes in connected components (events)



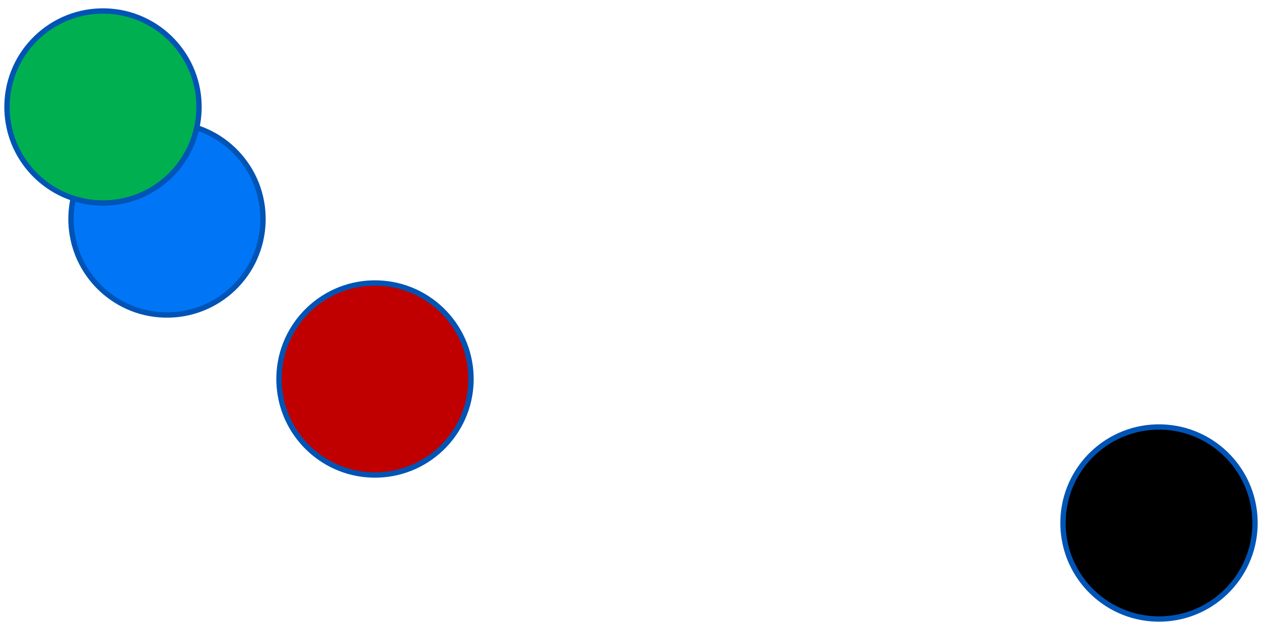


from 1 to 2 connected components

Grouping Structure

Reeb graph

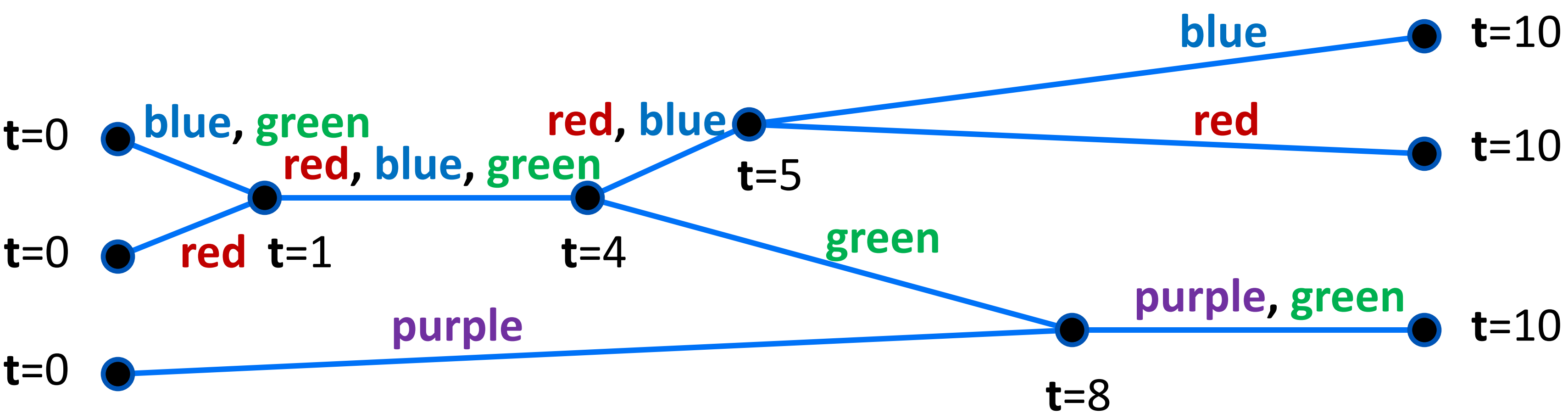
disregard group size (m = 1) and duration (δ = 0)



Grouping Structure

Reeb graph

disregard group size (m = 1) and duration (δ = 0)



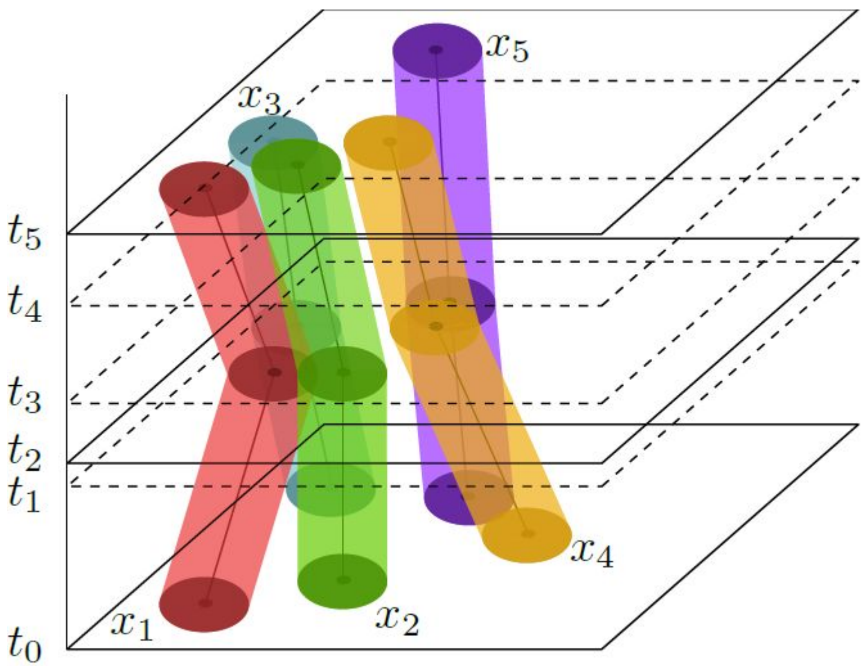
edges ~ connected components

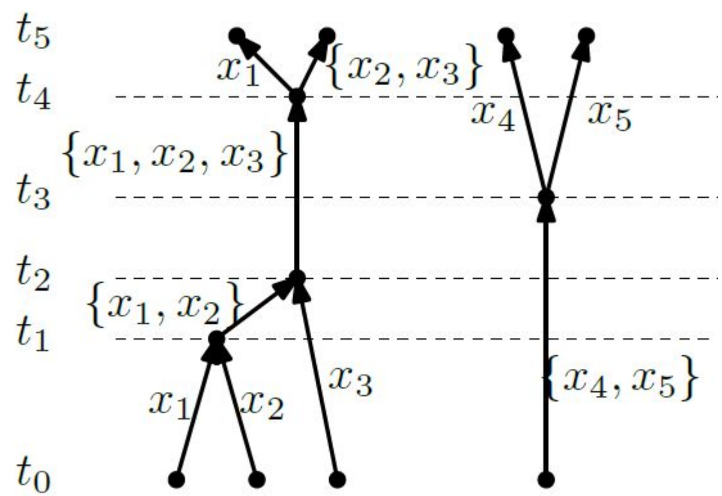
vertices ~ events (changes in connected components)

Grouping Structure

Reeb Graph

disregard group size (m = 1) and duration (δ = 0)



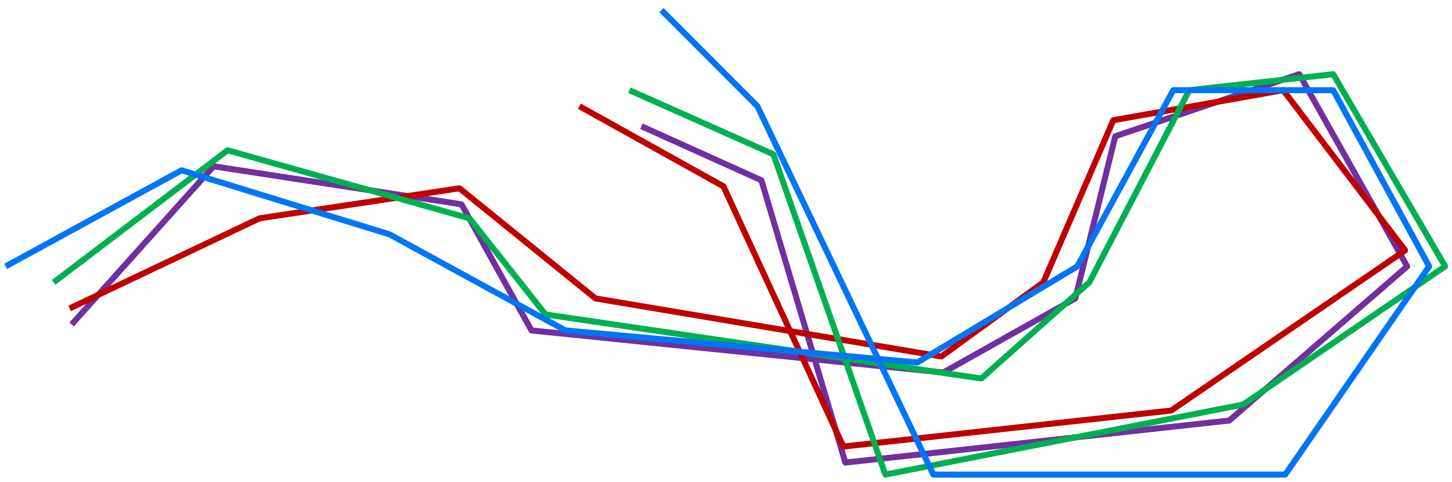


edges ~ connected components

vertices ~ events (changes in connected components)

Computing the Grouping Structure

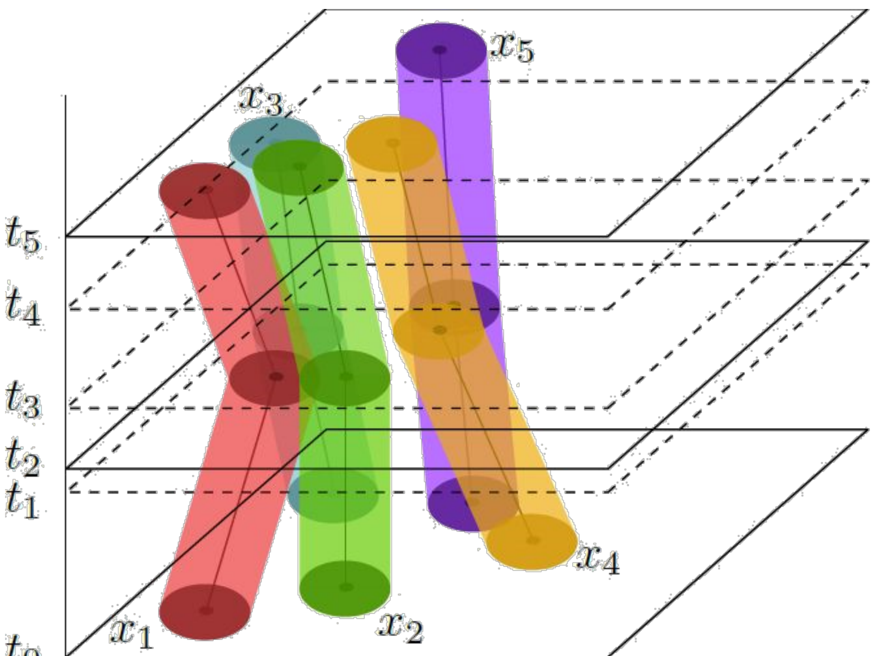
- Assume t time steps and n entities
- Assume piecewise-linear trajectories and constant speed on pieces
- The Reeb graph has $O(t\ n^2)$ vertices and edges;
  this bound is tight in the worst case
- Its computation takes $O(t\ n^2 \log t\ n)$ time

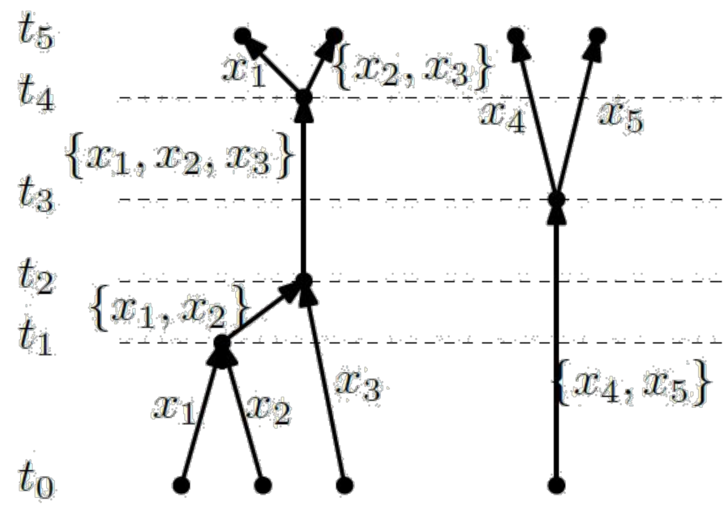


Reeb graph

Four type of vertices

start vertex (time t_0)	in-degree 0, out-degree 1
end vertex (time t_t)	in-degree 1, out-degree 0
merge vertex	in-degree 2, out-degree 1
split vertex	in-degree 1, out-degree 2





Technische Universiteit Eindhoven University of Technology

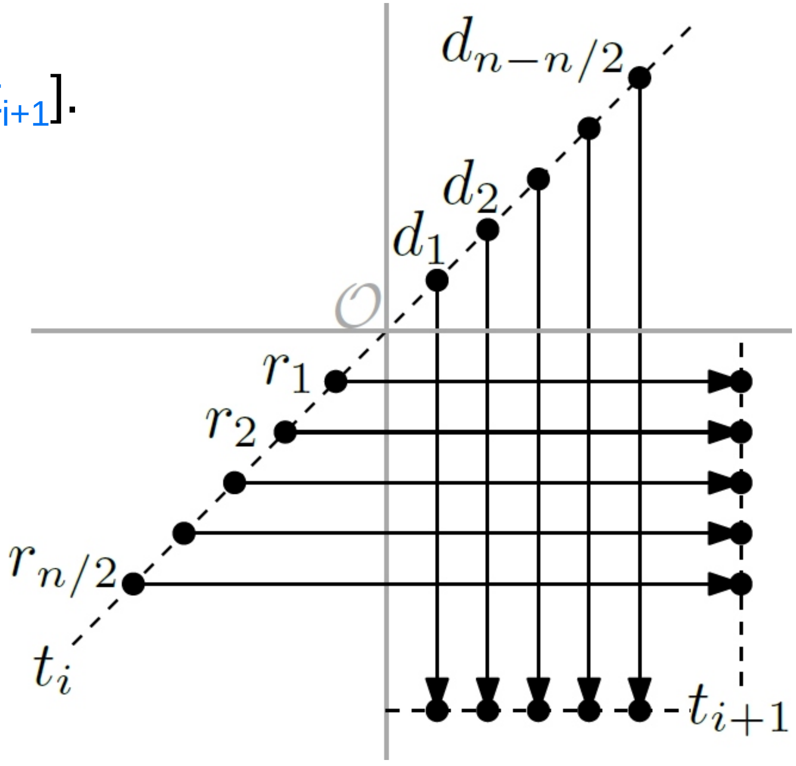
Complexity of the Reeb graph

Observation

Reeb graph of has $\Omega(t\ n^2)$ vertices and edges

Proof

Construction for one time step $[t_i, t_{i+1}]$.



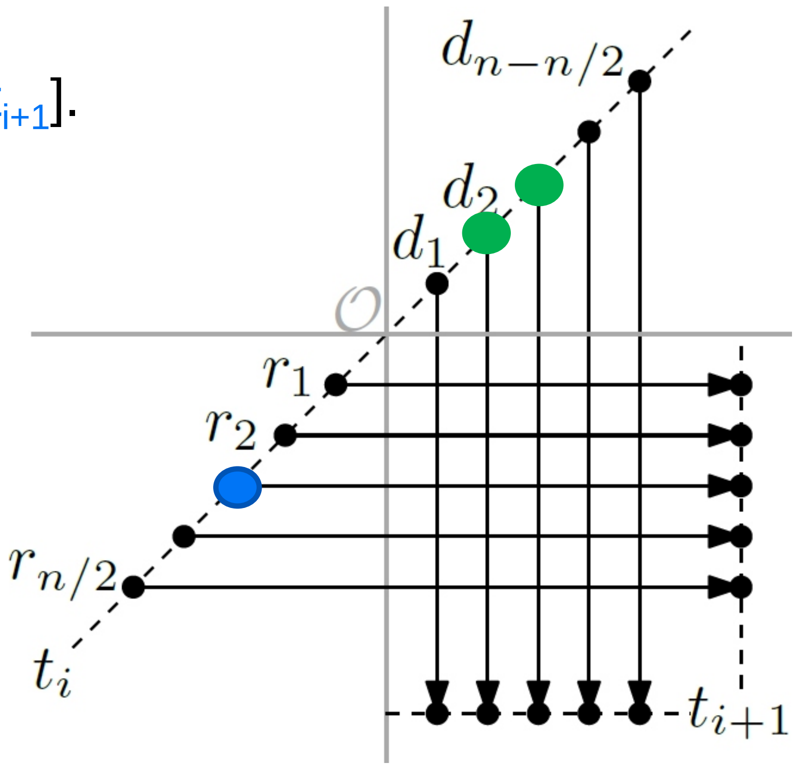
Complexity of the Reeb graph

Observation

Reeb graph of has $\Omega(t\, n^2)$ vertices and edges

Proof

Construction for one time step $[t_i, t_{i+1}]$.



Complexity of the Reeb graph

Observation

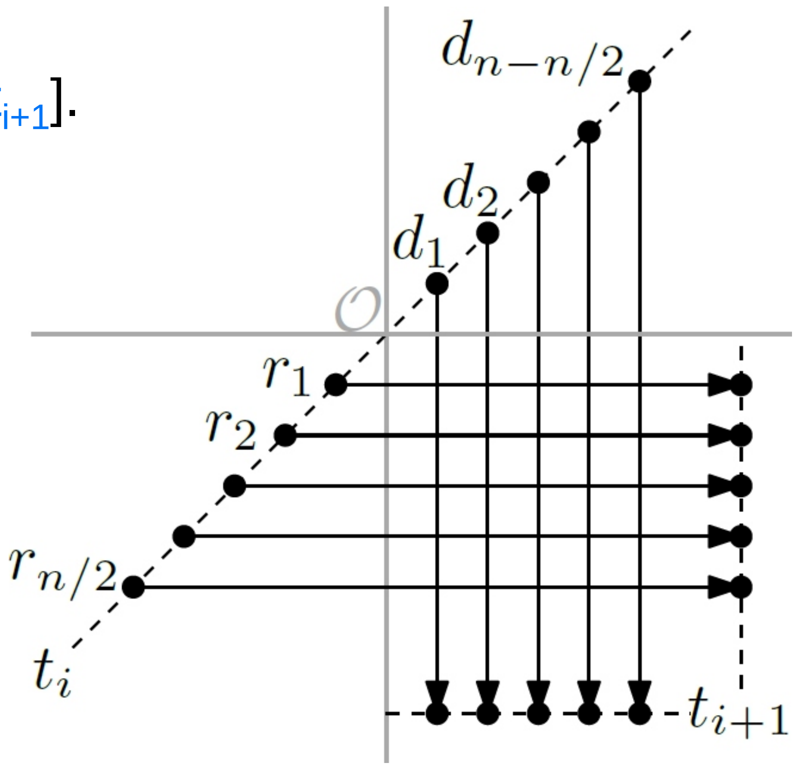
Reeb graph of has $\Omega(t\ n^2)$ vertices and edges

Proof

Construction for one time step $[t_i, t_{i+1}]$.

➡ $n/2 \cdot n/2 = \Omega(n^2)$

components / time step ■



TU/e Technische Universiteit Eindhoven University of Technology

Complexity of the Reeb graph

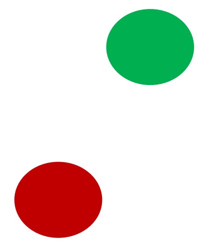
Observation

Reeb graph of has $O(t\ n^2)$ vertices

Proof

Consider two entities during one time interval [t_i, t_{i+1}].

This interval can generate at most two vertices in the Reeb graph.



TU/e Technische Universiteit Eindhoven University of Technology

Complexity of the Reeb graph

Observation

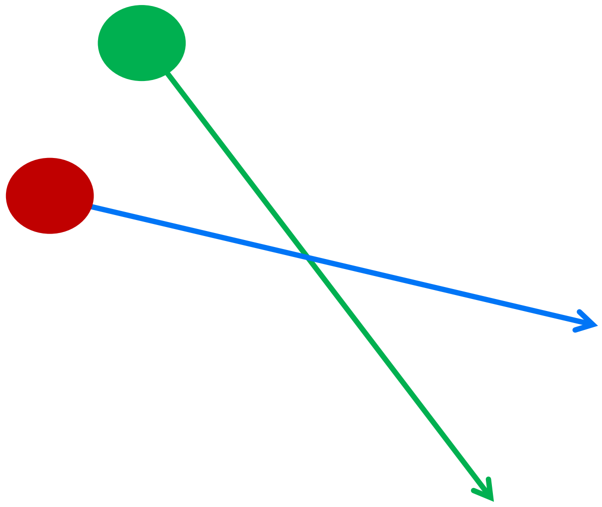
Reeb graph of has $O(t\ n^2)$ vertices

Proof

Consider two entities during one time interval $[t_i, t_{i+1}]$.

This interval can generate at most two vertices in the Reeb graph.

➡ $O(n^2)$ components/time step ■



Computing the Reeb graph

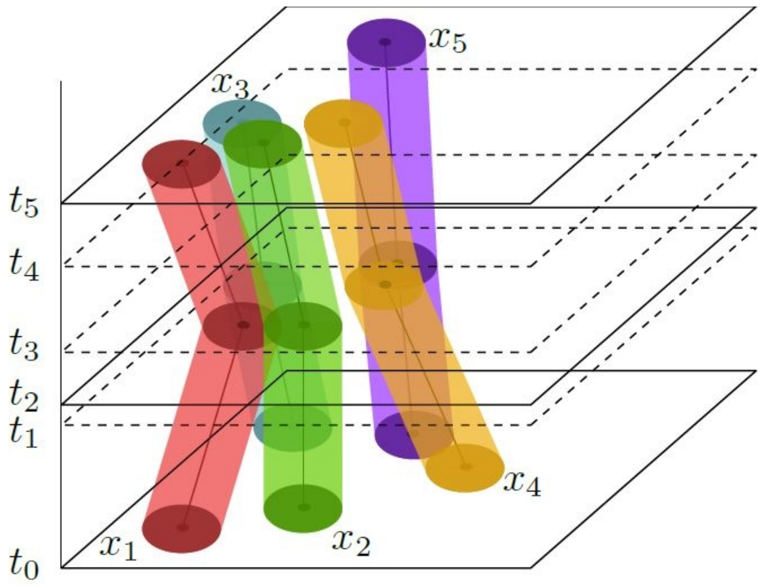
Compute all events

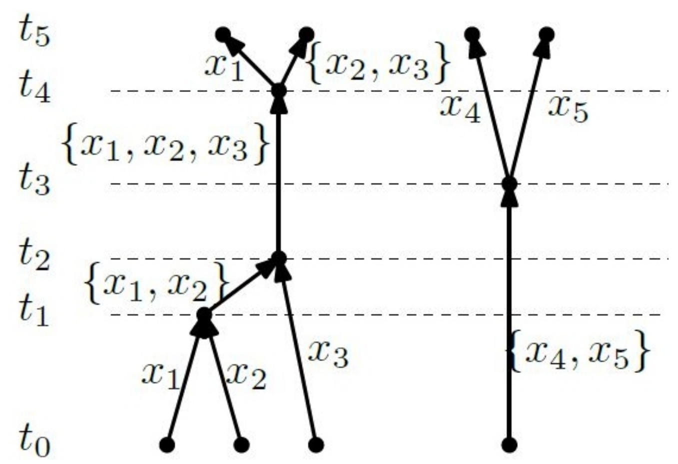
For every pair of skewed cylinders compute the time of intersection.

➡ O(t n²)

Sort the events with respect to time

➡ O(t n² log t n)





Computing the Reeb graph

Compute & sort all events

➡ O(t n² log t n)

Initialize graph

➡ O(n²)

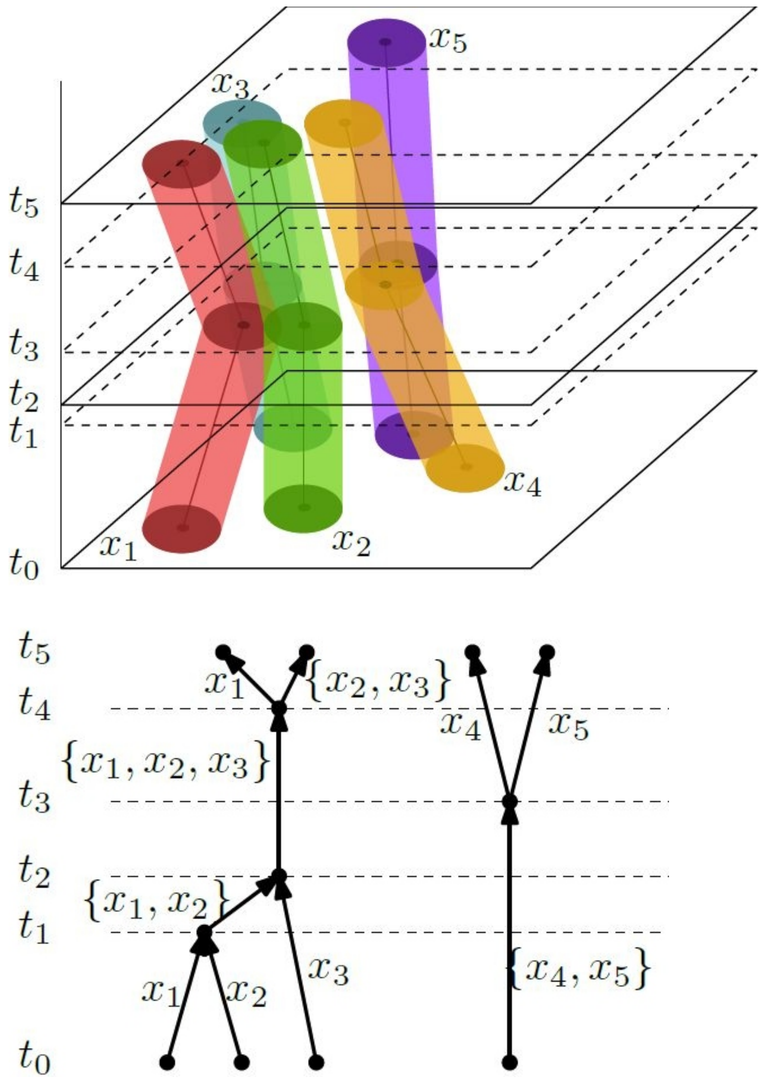
Handle events by increasing time

➡ O(t n² log n)

using a dynamic ST-tree

[Sleator & Tarjan'83]

[Parsa'12]

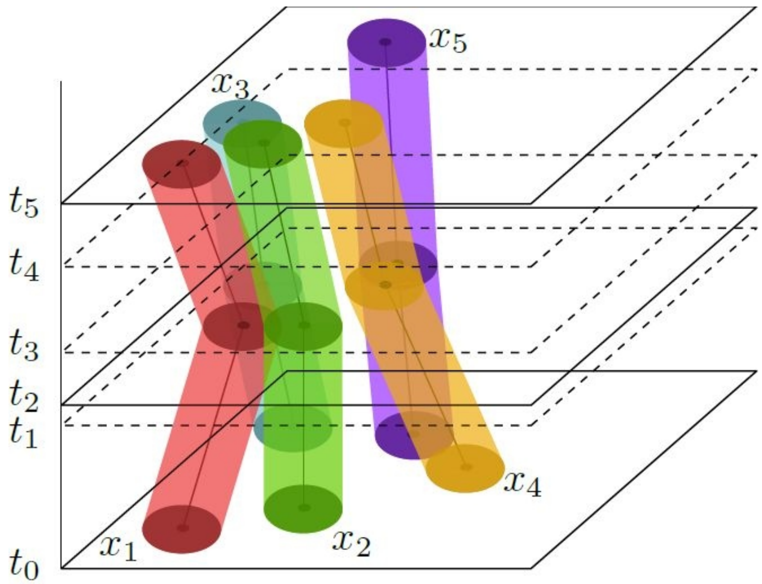


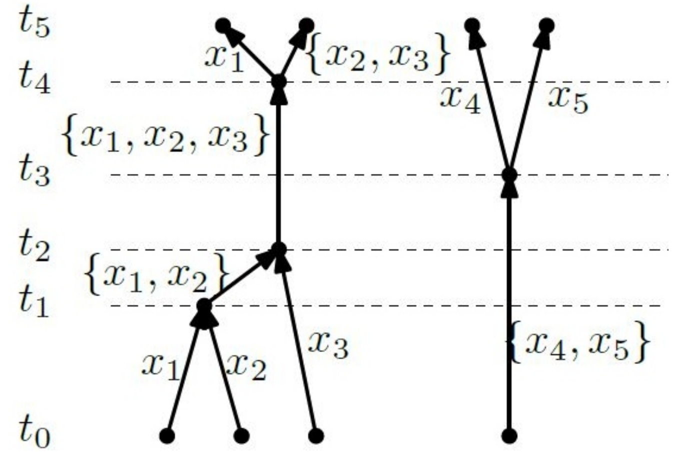
TU/e Technische Universiteit Eindhoven University of Technology

Computing the Reeb graph

Theorem

The Reeb graph can be computed in $O(t\, n^2 \log t\, n)$ time

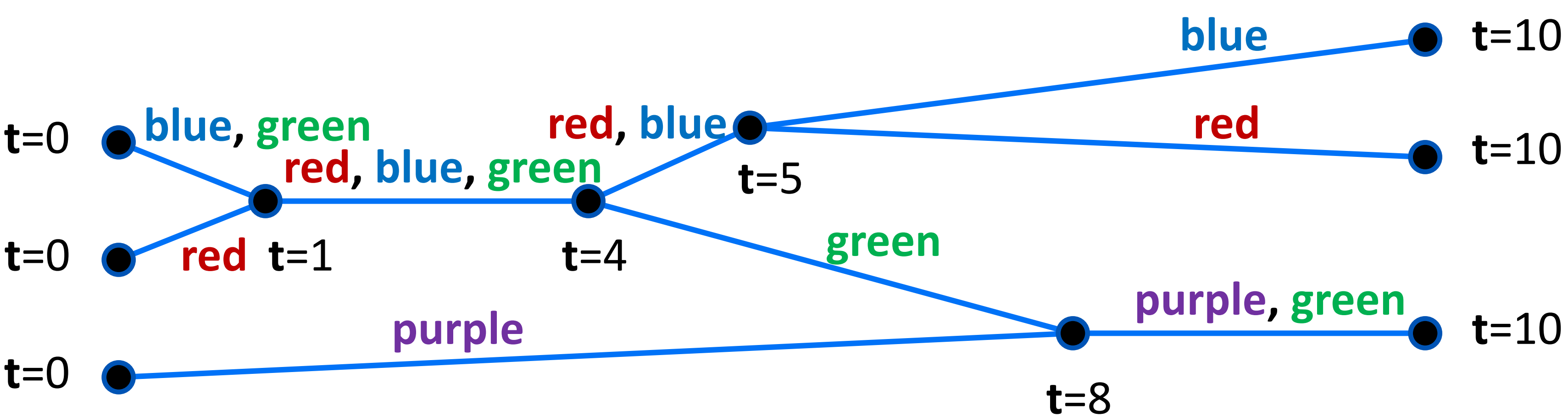




Number of maximal groups

Each entity follows a directed path in the Reeb graph

- starting at a start vertex and
- ending at an end vertex



Number of maximal groups

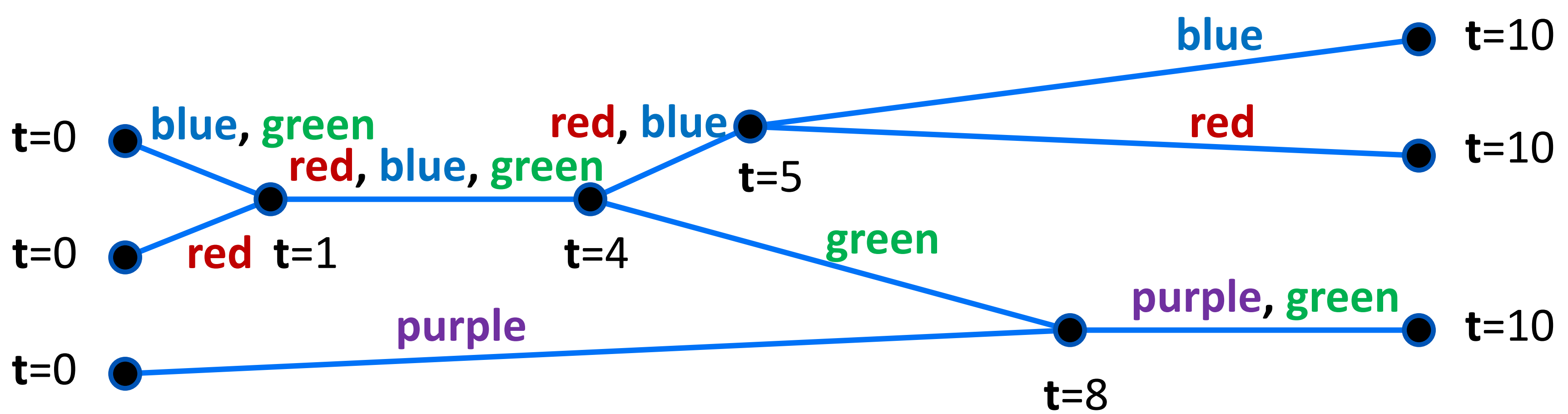
Theorem

There are O(t n^3) maximal groups.

Proof

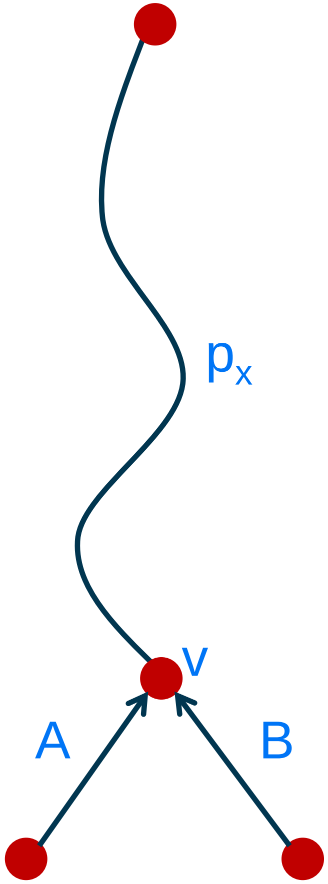
There are O(t n^2) vertices in the Reeb graph.

Prove that O(n) maximal groups can start at a start or merge vertex.



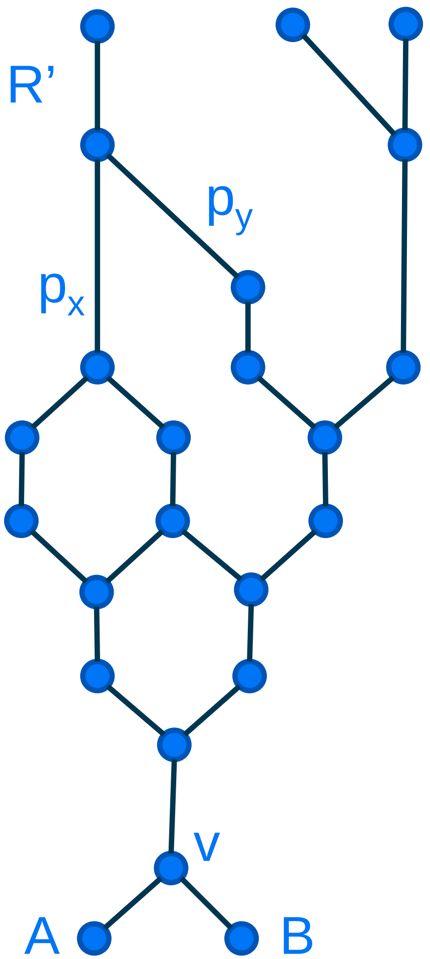
Number of maximal groups

- merge vertex v, sets A and B merge at v
- p_x the path of entity $x \in A \cup B$ starting at v



Number of maximal groups

- merge vertex v, sets A and B merge at v
- p_x the path of entity $x \in A \cup B$ starting at v
- R' union of all such paths from entities in $A \cup B$
- R' directed acyclic subgraph of Reeb graph



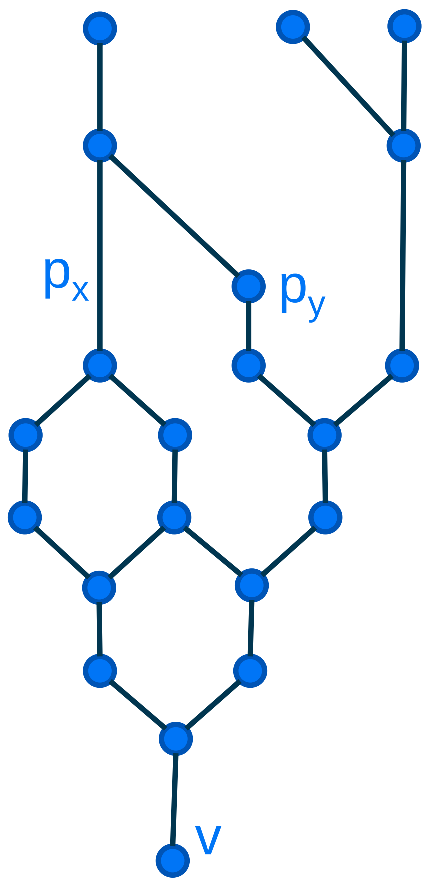
Unravel R'

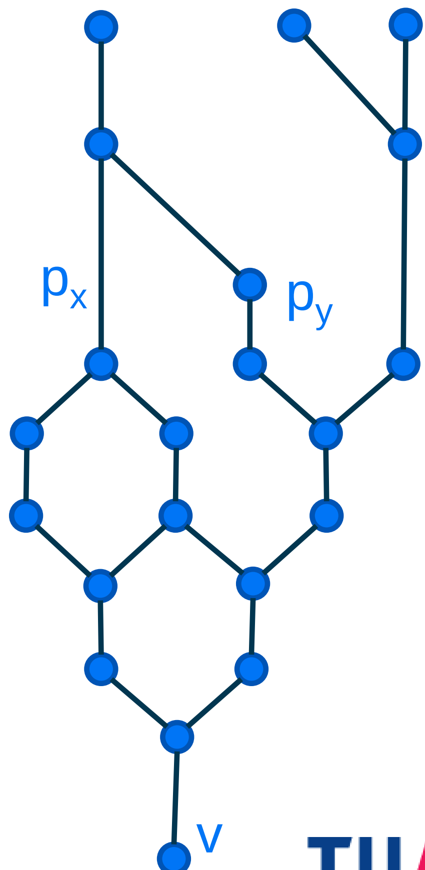
if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u

Number of maximal groups

Unravel R'

if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u

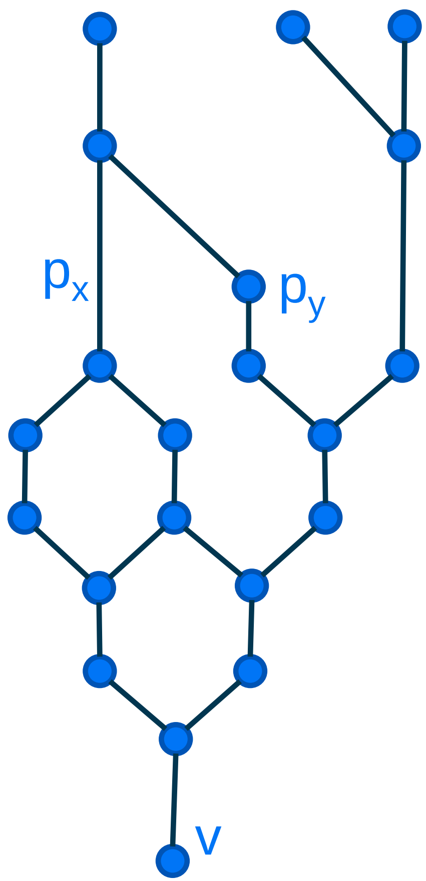


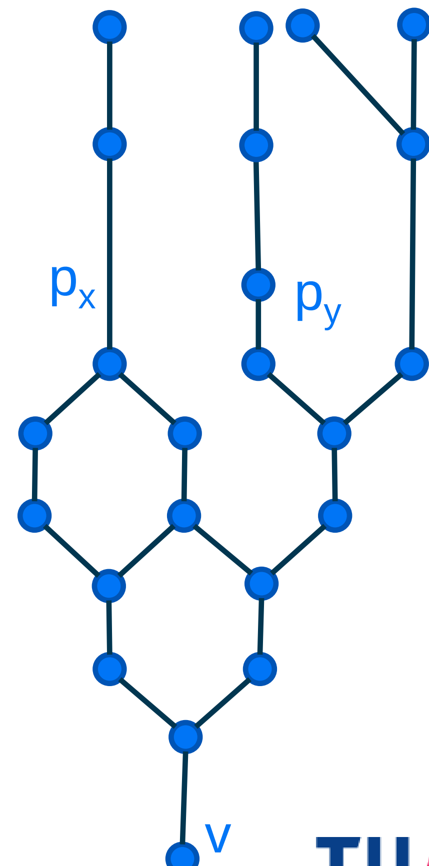


Number of maximal groups

Unravel R’

if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u

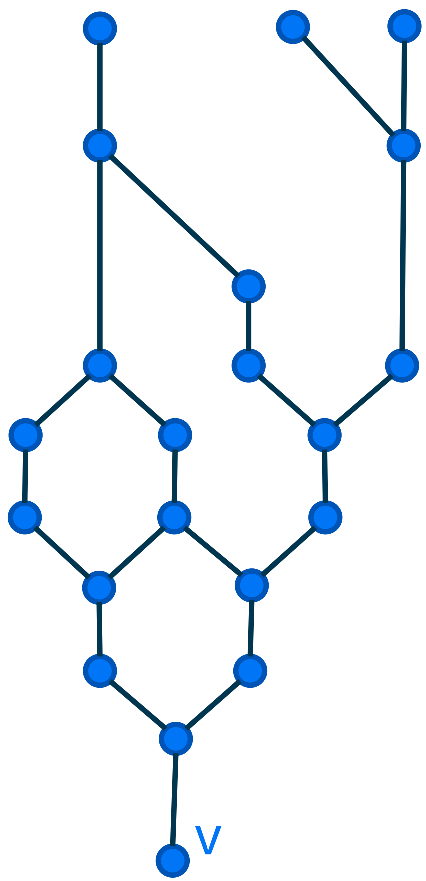


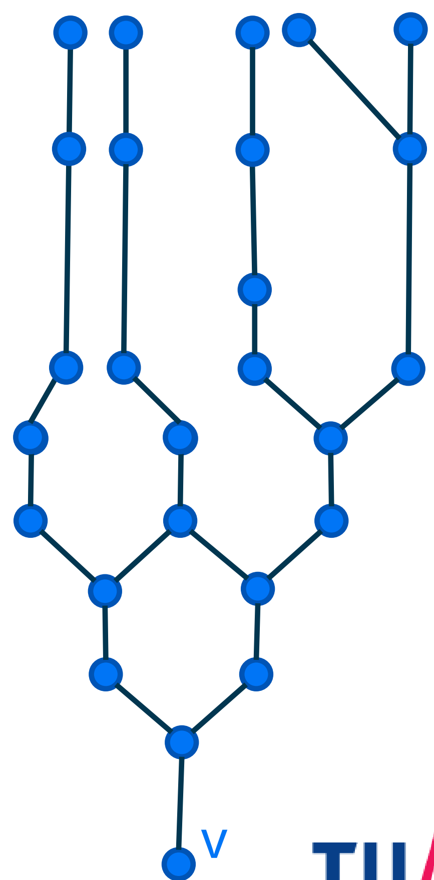


Number of maximal groups

Unravel R'

if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u

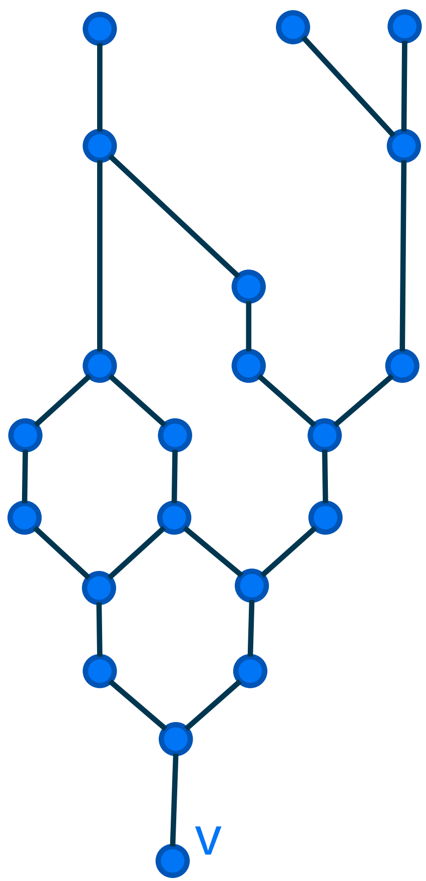


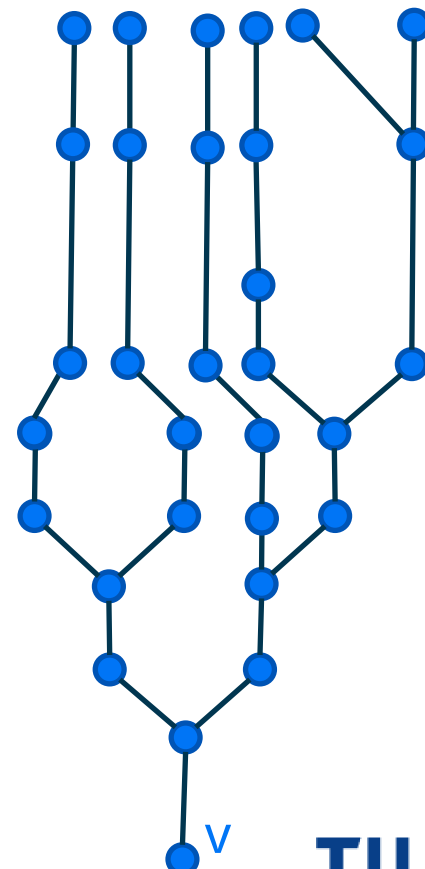


Number of maximal groups

Unravel R’

if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u





Number of maximal groups

Unravel R’

if p_x and p_y split and then merge at a vertex u then duplicate the paths starting at u

- a maximal group can end only at an end or split vertex
- number of leaves = |A| + |B| ≤ n

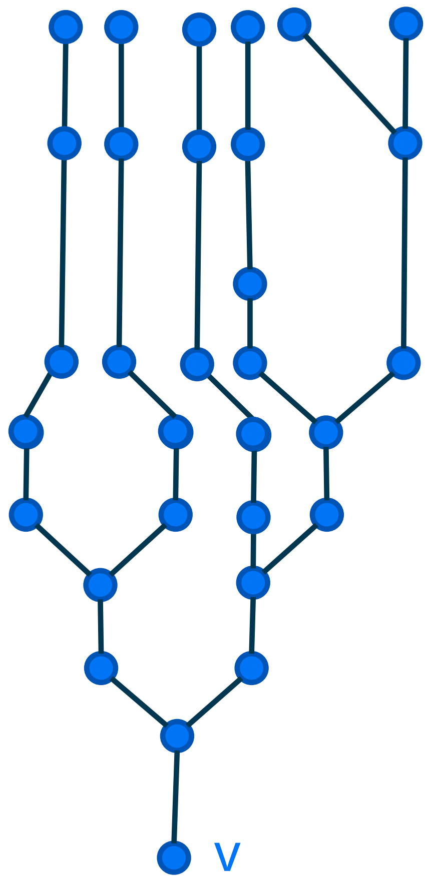
Number of split vertices?

- Every vertex has degree at most 3

  ➡ O(n) split vertices

- There can be at most one maximal group starting at v and ending at a split vertex w

  ➡ O(n) maximal groups starting at v



Technische Universiteit Eindhoven University of Technology

Number of maximal groups

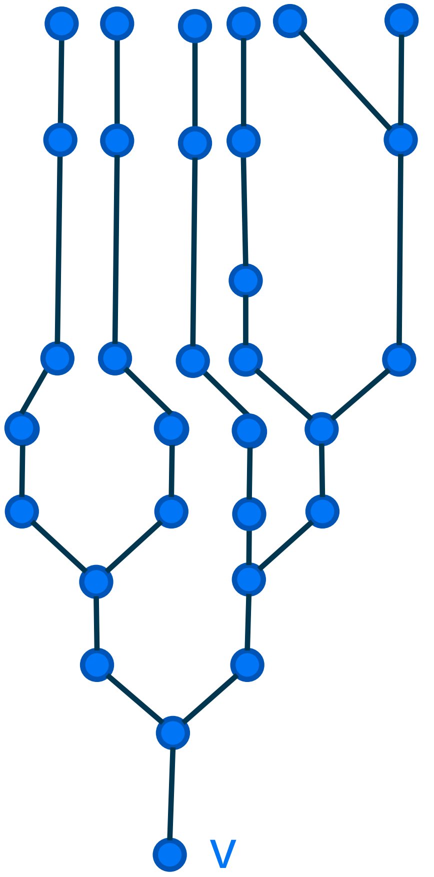
Theorem

There are $O(t\ n^3)$ maximal groups.

Proof

There are $O(t\ n^2)$ vertices in the Reeb graph.

At most $O(n)$ maximal groups can start at a start or merge vertex. ■

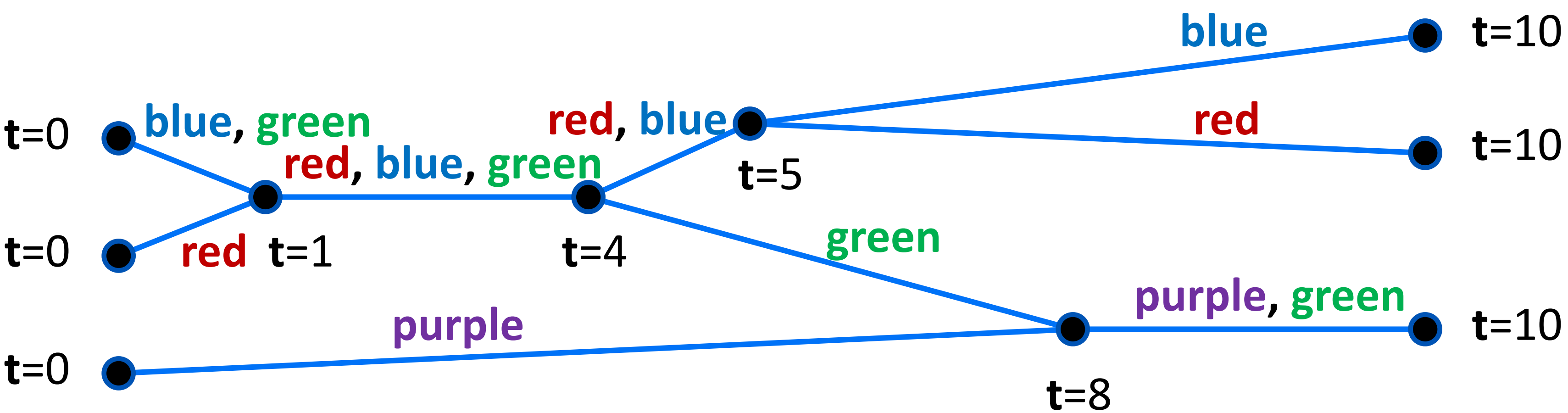


TU/e Technische Universiteit Eindhoven University of Technology

Computing the Maximal Groups

Given a value for group size m and duration δ and distance d:

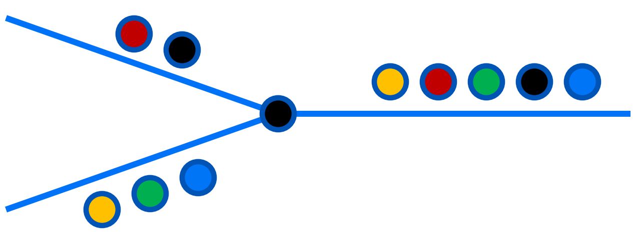
1. Compute the Reeb graph using distance d
2. Annotate its edges and vertices
3. Process the vertices in time-order, maintaining known maximal groups
4. Filter the maximal groups (using m and δ)



Computing the Maximal Groups

Given a value for group size m and duration δ and distance d:

1. Compute the Reeb graph using distance d
2. Annotate its edges and vertices
3. Process the vertices in time-order, maintaining known maximal groups
4. Filter the maximal groups (using m and δ)



merge

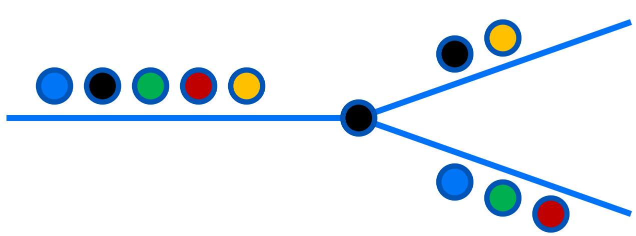
No existing maximal group ends, the maximal groups of the two branches are joined and maintained with the new branch

One new maximal group starts and is maintained

Computing the Maximal Groups

Given a value for group size m and duration δ and distance d:

1. Compute the Reeb graph using distance d
2. Annotate its edges and vertices
3. Process the vertices in time-order, maintaining known maximal groups
4. Filter the maximal groups (using m and δ)



split

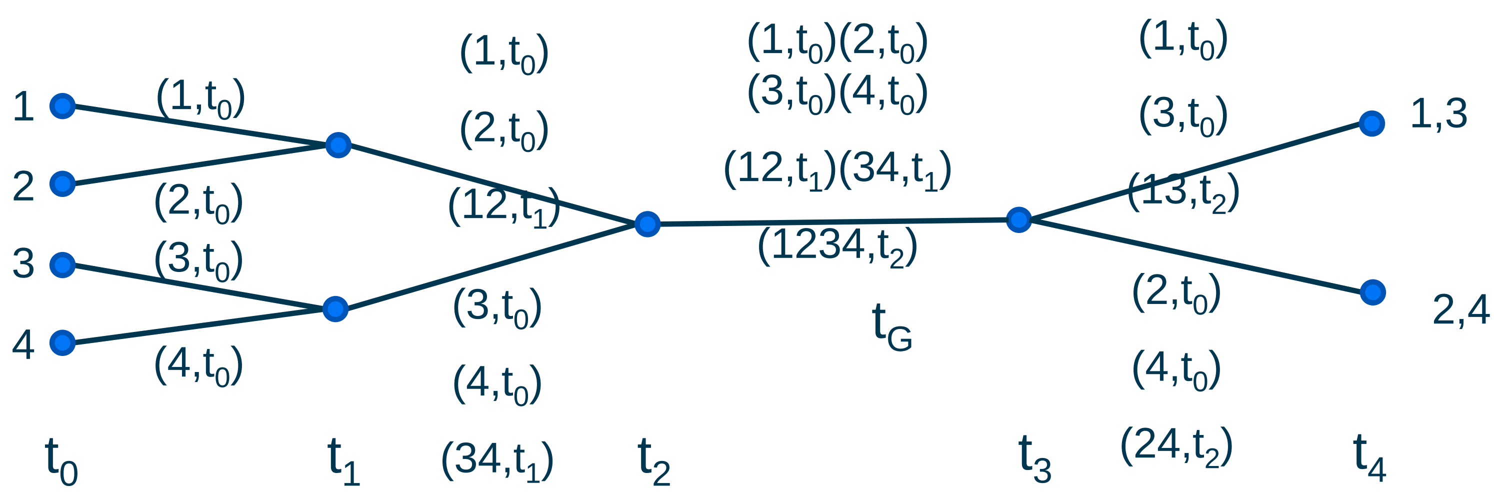
Any existing maximal group with at least one of each new component ends and is reported

New maximal groups can start on both branches; they are maintained

Computing the maximal groups

Input: Given a set of trajectories and three parameters m, δ and ε.

1. Compute the Reeb graph using distance ε. Time: $O(\tau n^2 \log \tau n)$
2. Process the vertices in time-order, maintaining maximal groups



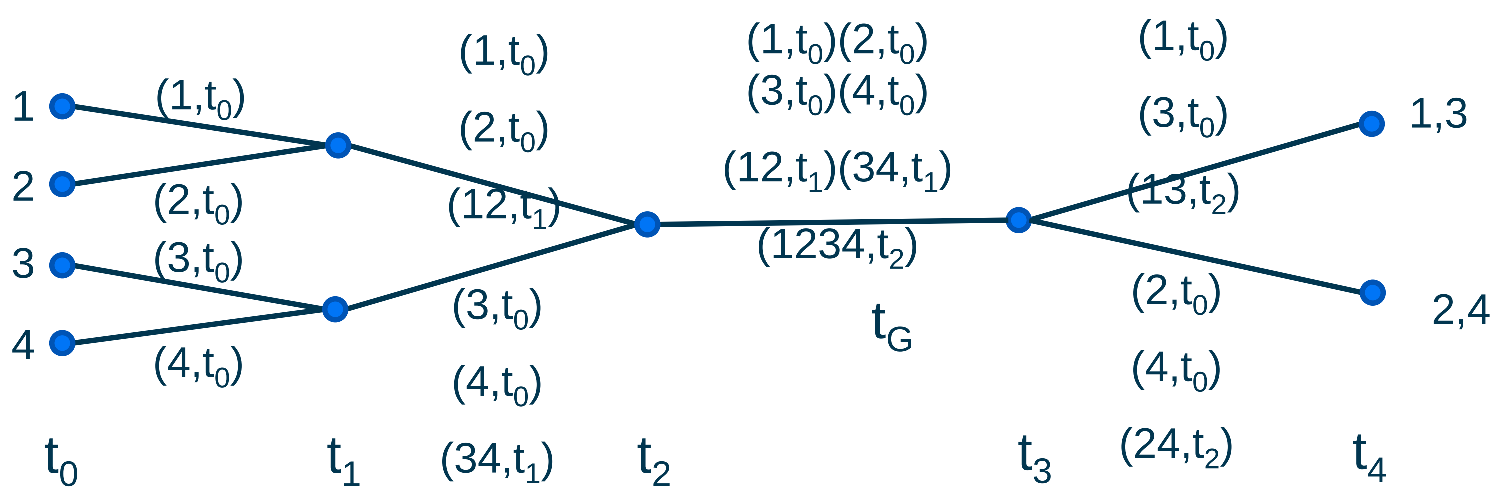
Technische Universiteit Eindhoven University of Technology

Computing the maximal groups

2. Maintain maximal groups

Each edge e=(u,v) is labelled with a set of maximal groups G_e.

Note: A group G becomes maximal at a vertex.

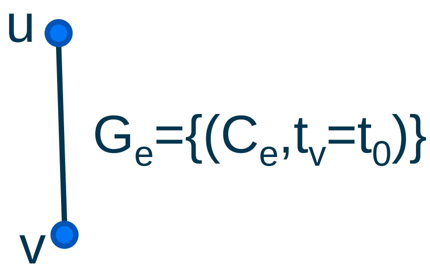


Computing the maximal groups

2. Annotate its edges and vertices

Each edge $e=(u,v)$ is labelled with a set of maximal groups G_e.
Note that a group G becomes maximal at a vertex v.

a. Start vertex: Set $G_e = \{(C_e,t_v=t_0)\}$



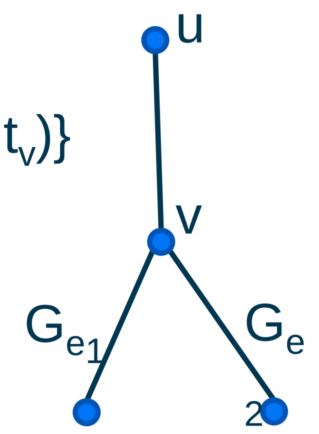
Computing the maximal groups

2. Annotate its edges and vertices

Each edge $e=(u,v)$ is labelled with a set of maximal groups G_e.
Note that a group G becomes maximal at a vertex v.

a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$
b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v).

$$G_e = \{G_{e_1} \cup G_{e_2} \cup (C_e, t_v)\}$$

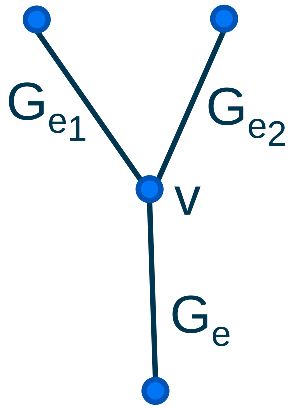


Computing the maximal groups

2. Annotate its edges and vertices

Each edge $e=(u,v)$ is labelled with a set of maximal groups G_e.
Note that a group G becomes maximal at a vertex v.

a. Start vertex: Set $G_e = \{(C_e,t_v=t_0)\}$
b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e,t_v).
c. Split vertex:

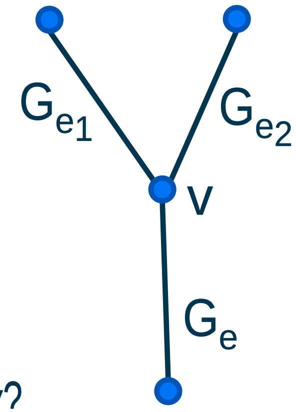


Computing the maximal groups

2. Annotate its edges and vertices

Each edge e=(u,v) is labelled with a set of maximal groups G_e.
Note that a group G becomes maximal at a vertex v.

a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$
b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v).
c. Split vertex:
   - A group may end at v (if group splits)
   - A group may start at v on either e_1 or e_2
   - A group may continue on e_1 or e_2

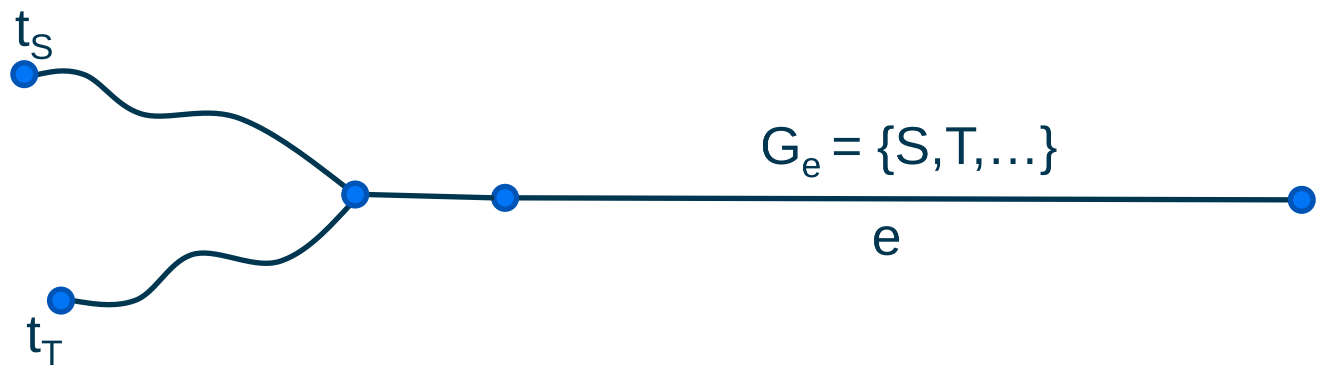


Question: How can we compute these groups efficiently?

Store maximal groups

Observation: Assume S, T in G_e.

S starting at t_S and T starting at t_T.

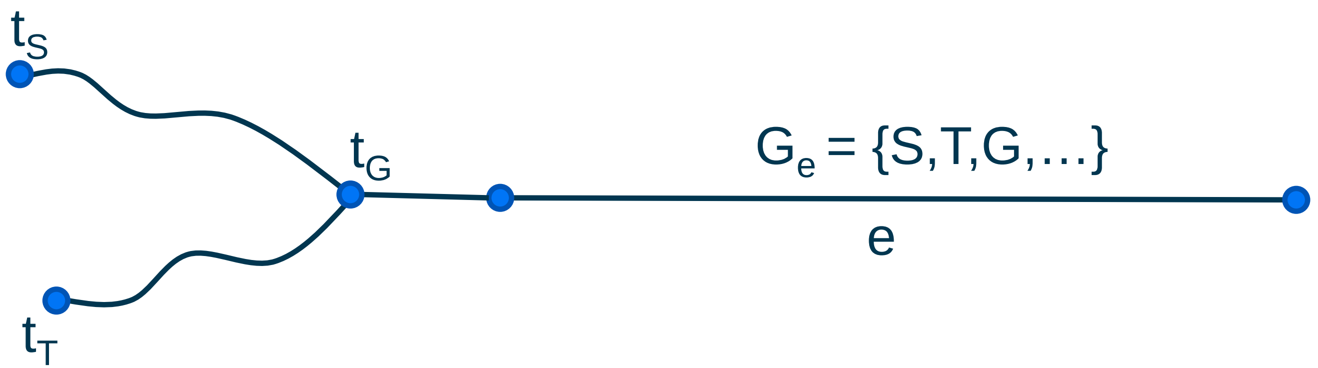


Store maximal groups

Observation: Assume S, T in G_e.

S starting at t_S and T starting at t_T.

- $\exists G \supseteq S \cup T$ in G_e with starting time $t_G \geq \max\{t_S, t_T\}$.



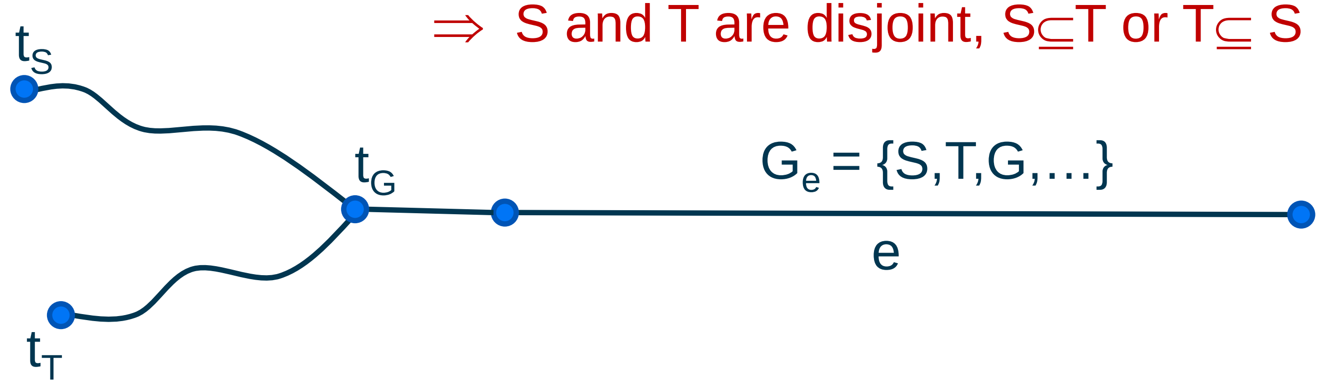
Store maximal groups

Observation: Assume S, T in G_e.

S starting at t_S and T starting at t_T.

- $\exists G \supseteq S \cup T$ in G_e with starting time $t_G \geq \max\{t_S, t_T\}$.
- If $S \cap T \neq \emptyset$ then $S \subseteq T$ or $T \subseteq S$.

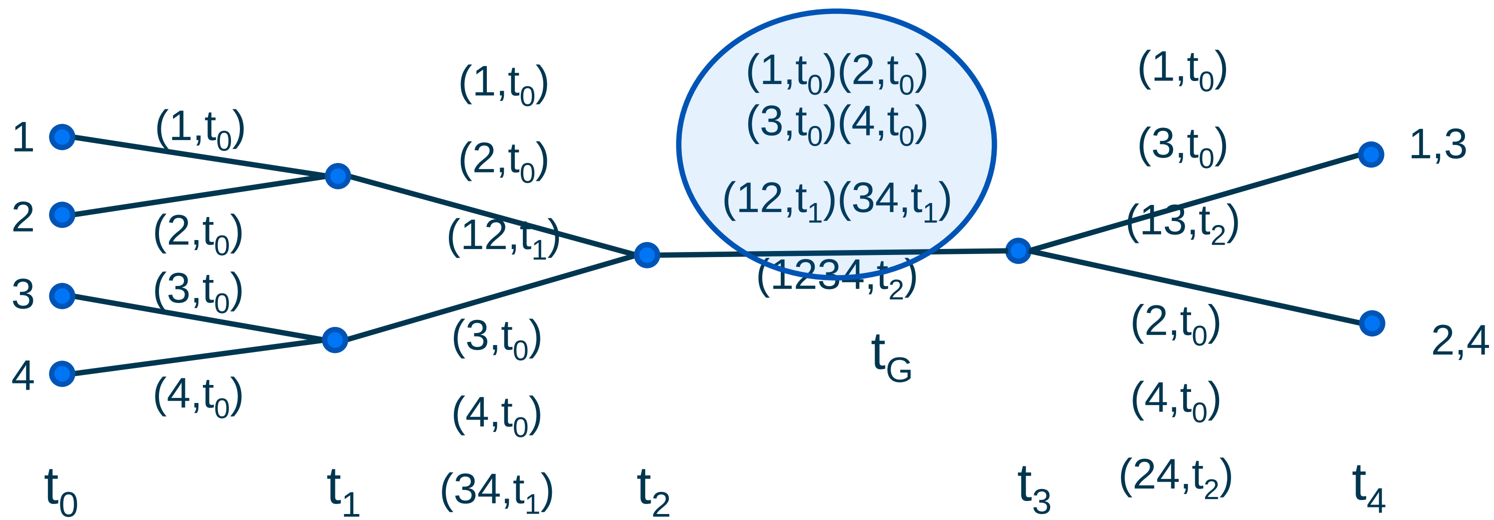
$\Rightarrow$ S and T are disjoint, $S \subseteq T$ or $T \subseteq S$



Store maximal groups

Represent G_e by a tree T_e.

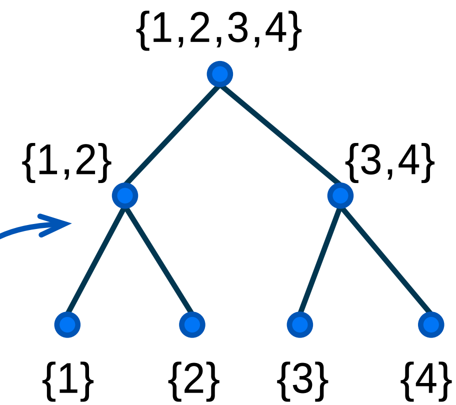
Each node v in T_e represents a group G_v in G_e.
The children of v are the largest subgroups of of G_v.

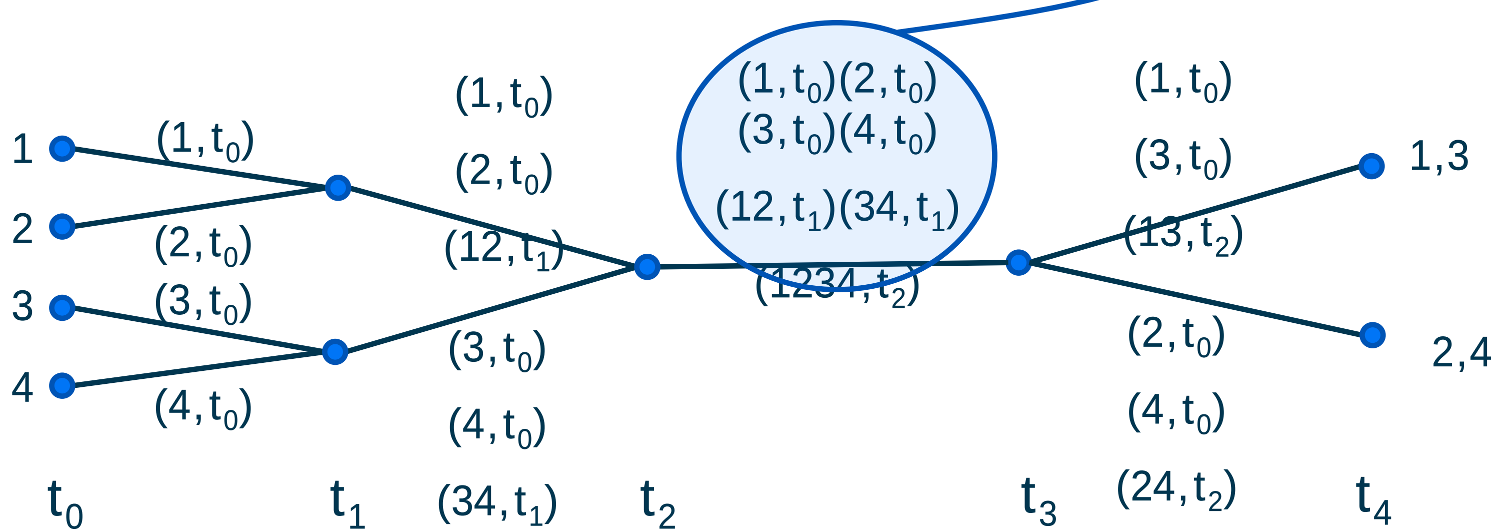


Store maximal groups

Represent G_e by a tree T_e.

Each node v in T_e represents a group G_v in G_e.
The children of v are the largest subgroups of of G_v.



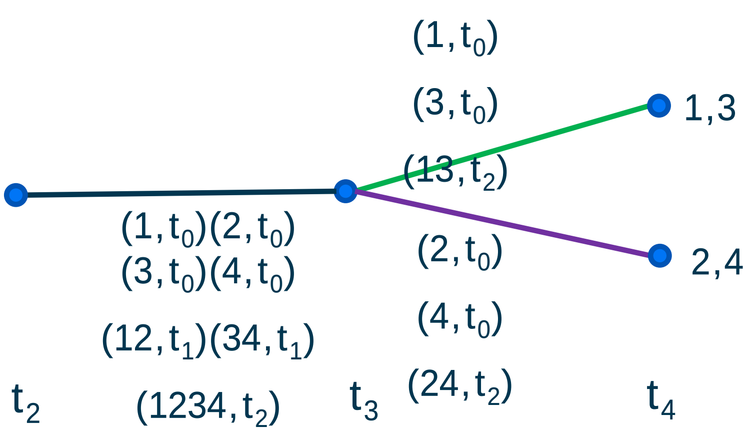


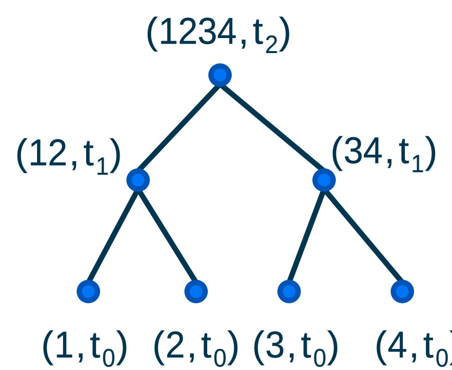
Computing the maximal groups

2. Annotate its edges and vertices

Each edge e=(u,v) is labelled with a set of maximal groups G_e.

a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$ ✓
b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v). ✓
c. Split vertex:
   - A group may end at v (if group splits)
   - A group may start at v on either e_1 or e_2
   - A group may continue on e_1 or e_2





Computing the maximal groups

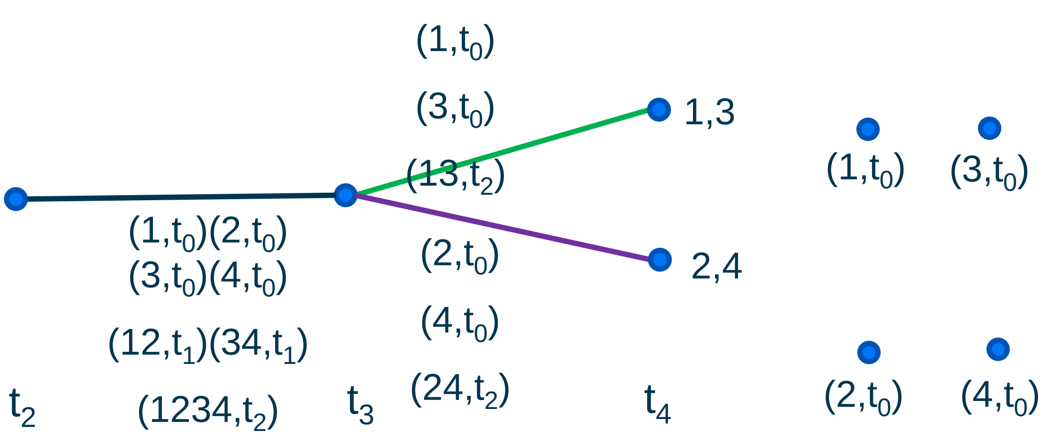
2. Annotate its edges and vertices

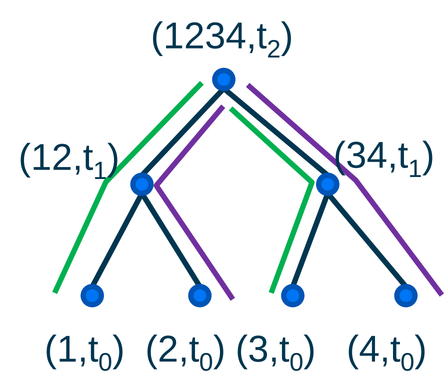
Each edge e=(u,v) is labelled with a set of maximal groups G_e.

a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$

b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v).

c. Split vertex:
  - A group may end at v (if group splits)
  - A group may start at v on either e_1 or e_2
  - A group may continue on e_1 or e_2



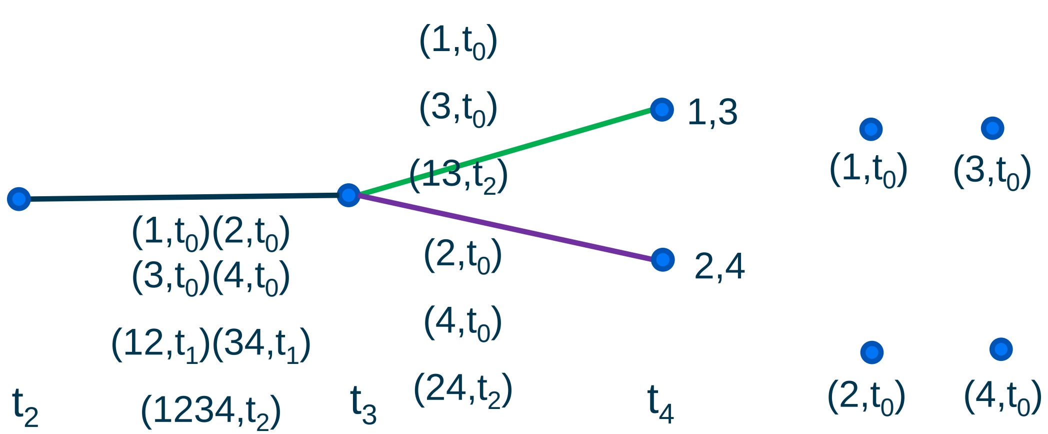


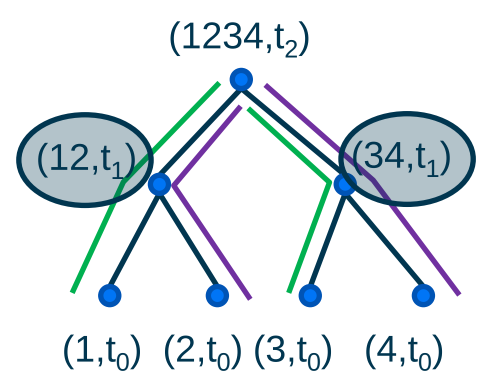
Computing the maximal groups

2. Annotate its edges and vertices

Each edge e=(u,v) is labelled with a set of maximal groups G_e.

a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$ ✓
b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v). ✓
c. Split vertex:
   - A group may end at v (if group splits)
   - A group may start at v on either e_1 or e_2
   - A group may continue on e_1 or e_2





Computing the maximal groups

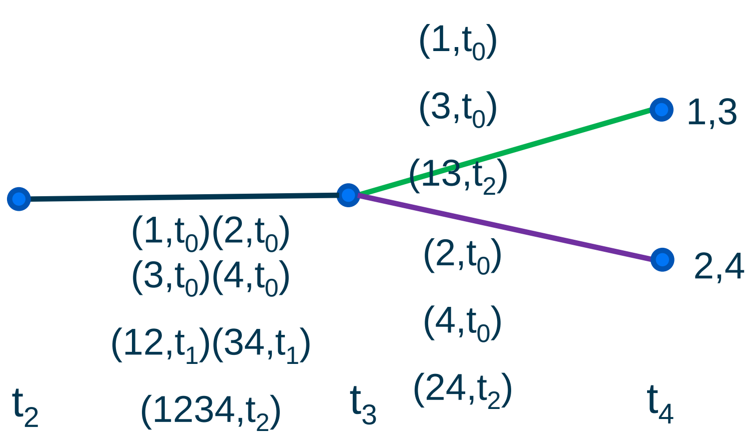
2. Annotate its edges and vertices

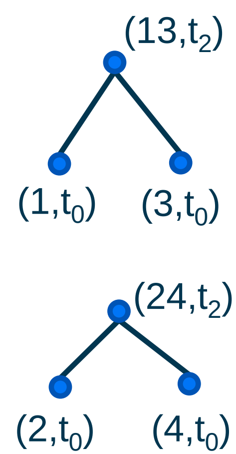
Each edge e=(u,v) is labelled with a set of maximal groups G_e.

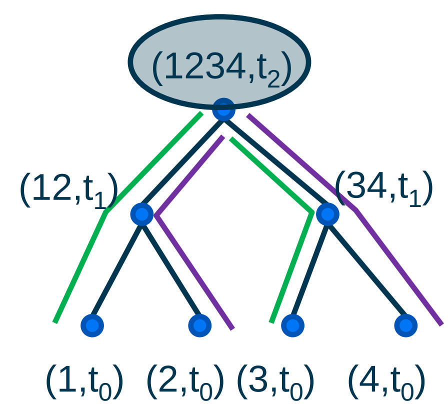
a. Start vertex: Set $G_e = \{(C_e, t_v = t_0)\}$

b. Merge vertex: Propagate the maximal groups from "children" to e and add (C_e, t_v).

c. Split vertex:
   - A group may end at v (if group splits)
   - A group may start at v on either e_1 or e_2
   - A group may continue on e_1 or e_2







Computing the maximal groups

Analysis

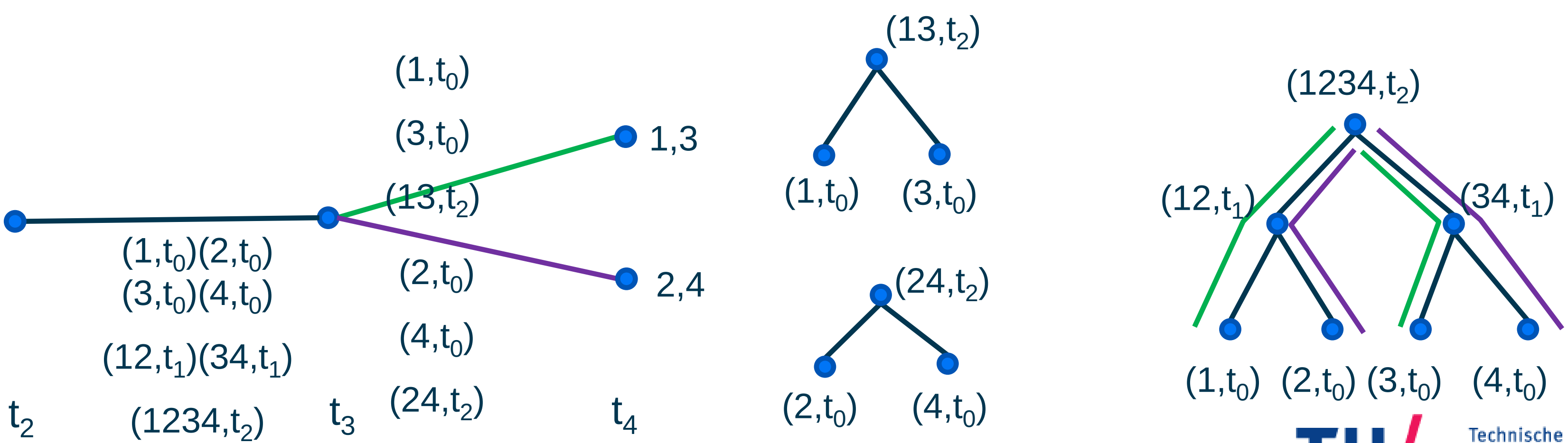
Start vertex: O(1)

Merge vertex: O(1)

Split vertex: $O(|T_e|)$ and $|T_e|=O(n)$

Total time: #vertices in the Reeb graph $\times$ O(n) = $O(\tau n^3)$

+ the total output size



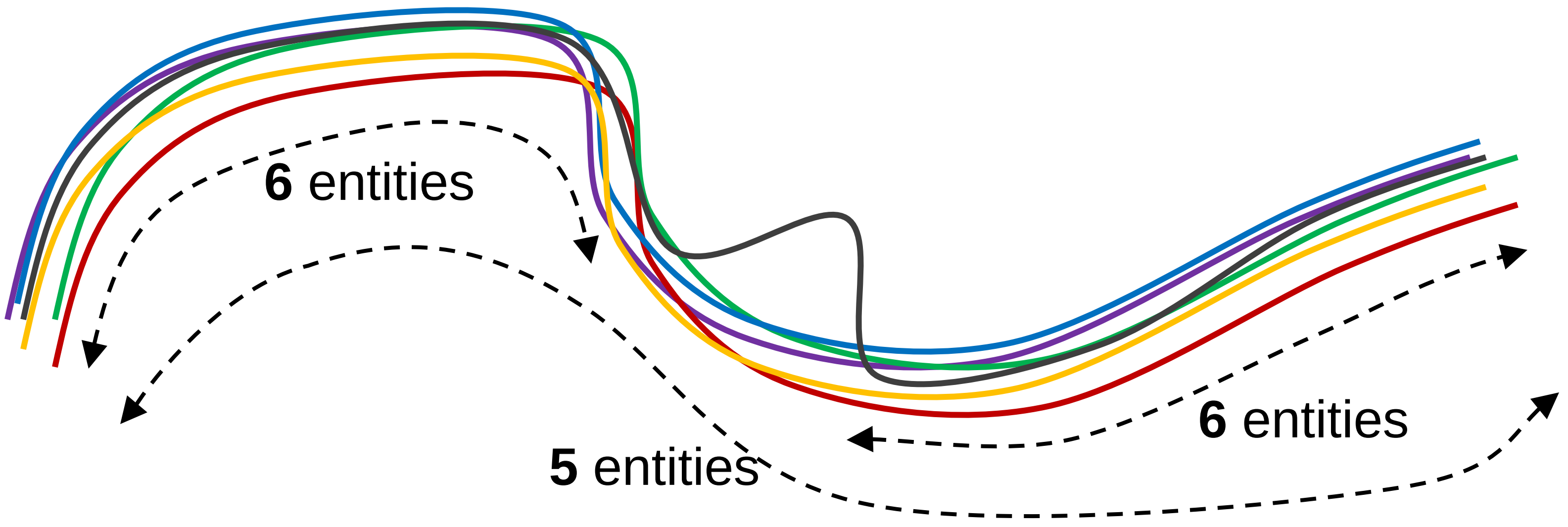
Computing the maximal groups

- Processing a vertex takes linear time
  ➡ computing all maximal groups costs $O(t\ n^3)$ time
  (*plus output size*)

- There are at most $O(t\ n^3)$ maximal groups,
  this bound is tight in the worst case

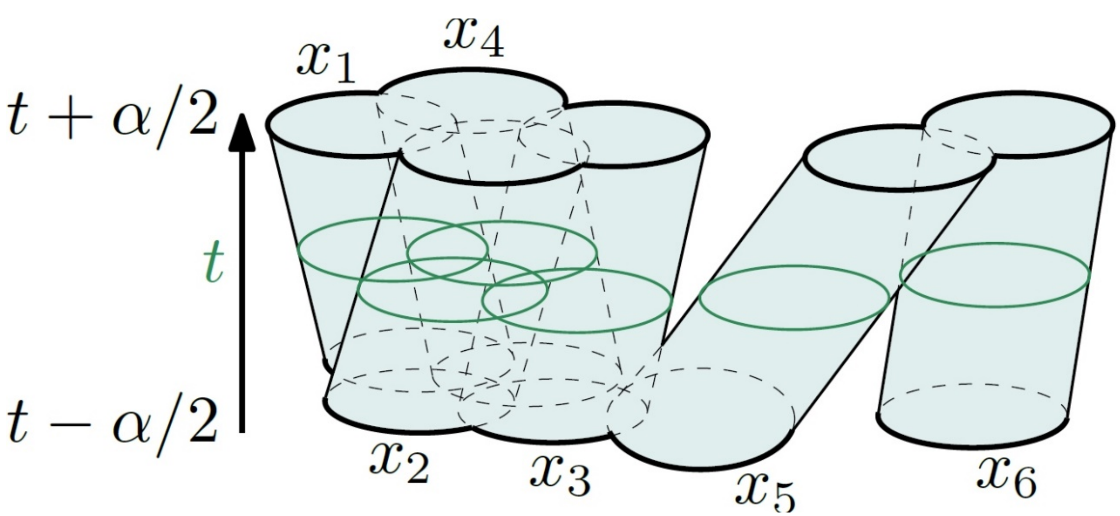
Robustness

- If a group of 6 entities has 1 entity leaving very briefly, should we really see this as
  - two maximal groups of size 6, and
  - one maximal group of size 5 ?



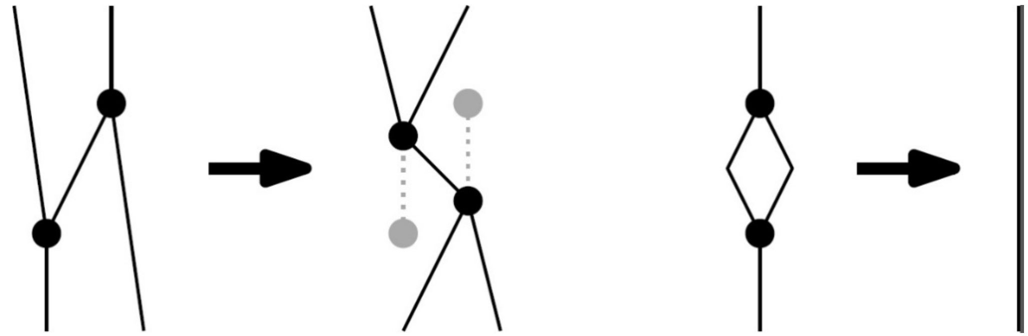
Robust grouping structure

- Entities are **α**-connected at time t if they are directly connected at some time t‘ in [t-**α**/2, t+**α**/2]



Robust grouping structure

- modify Reeb graph



Passing encounter

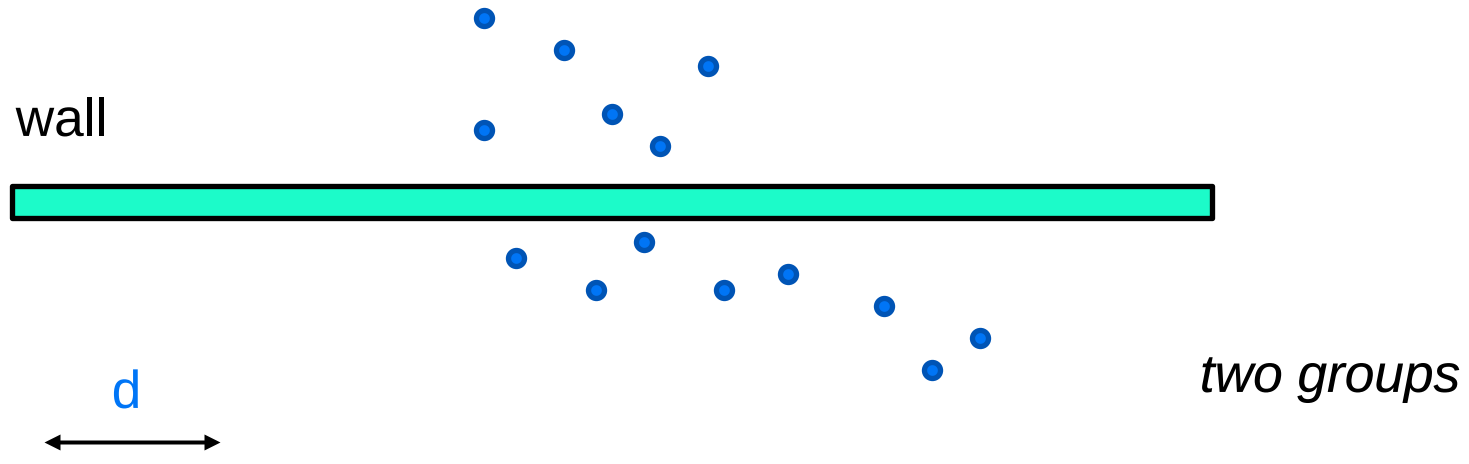
Collapse encounter

- at most O(t n^3) changes in the Reeb graph
- compute robust maximal groups in O(t n^3 log n) time (plus output size)

Grouping in environments

Upcoming

if distance should not be measured in a straight line, but geodesic amidst obstacles, what can we do?



The grouping structure

- A simple, clean model for grouping / moving flocks / …
- Proofs of desirable properties
- Algorithms for the computation of the grouping structure and the maximal groups, with efficiency bounds
- Adaptations to get robust grouping
- Plausible, based on implementation