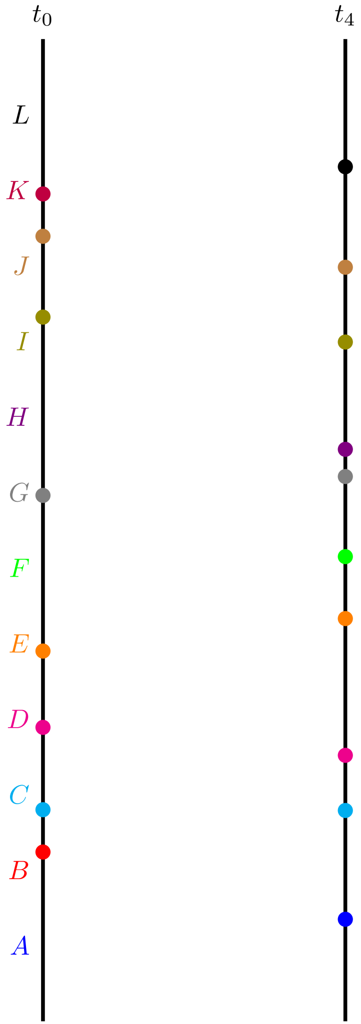
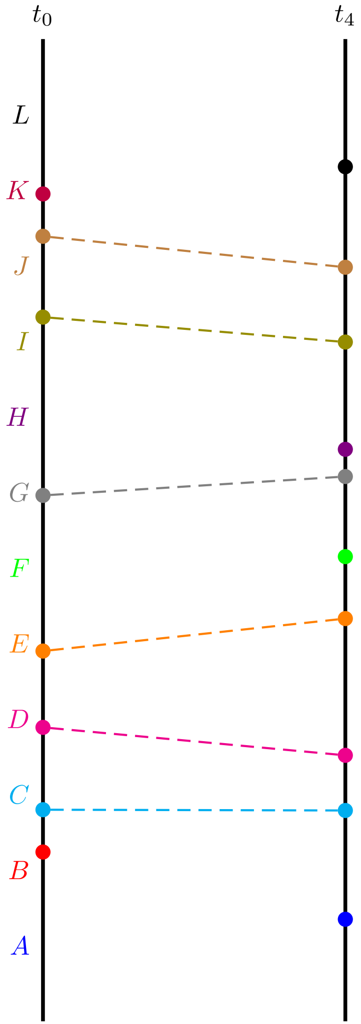
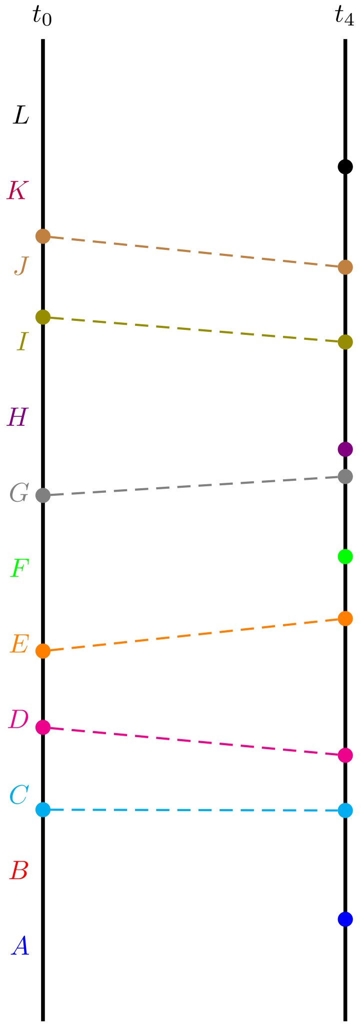
Maximal disks t_0 and t_4



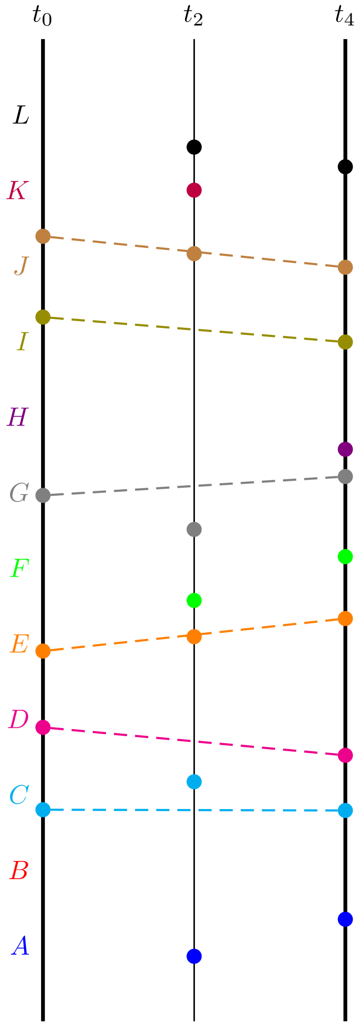
Join t_0 and t_4



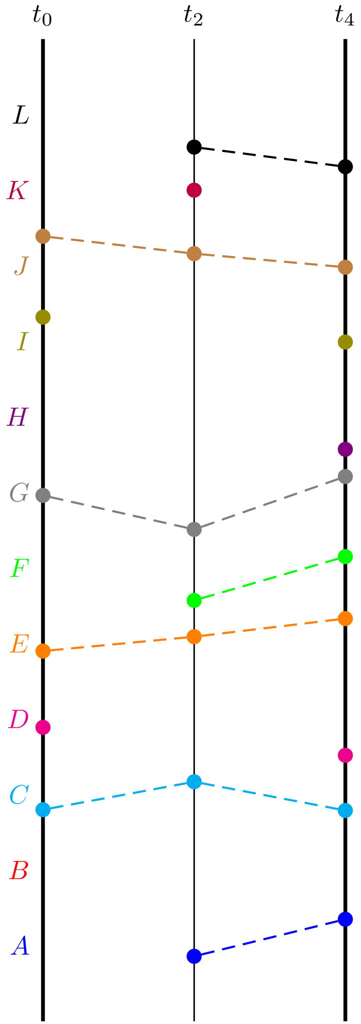
Filter t_0 and t_4



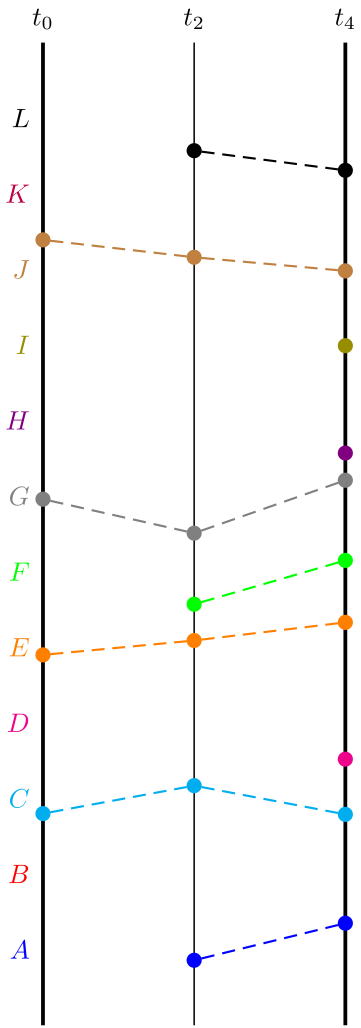
Maximal disks t_2



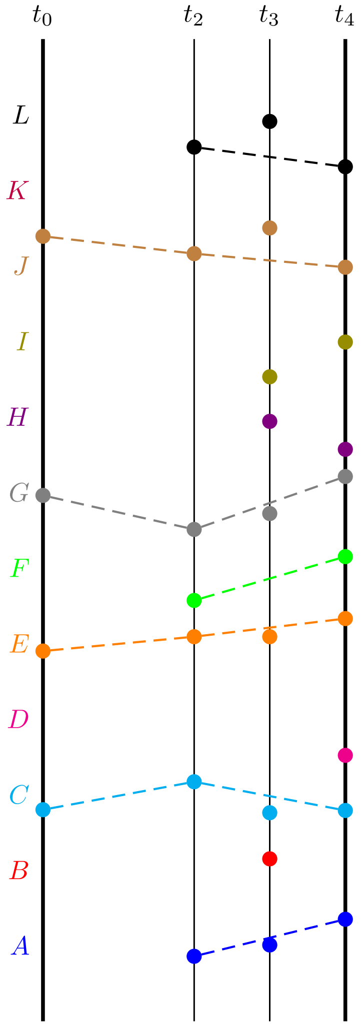
Join t_2



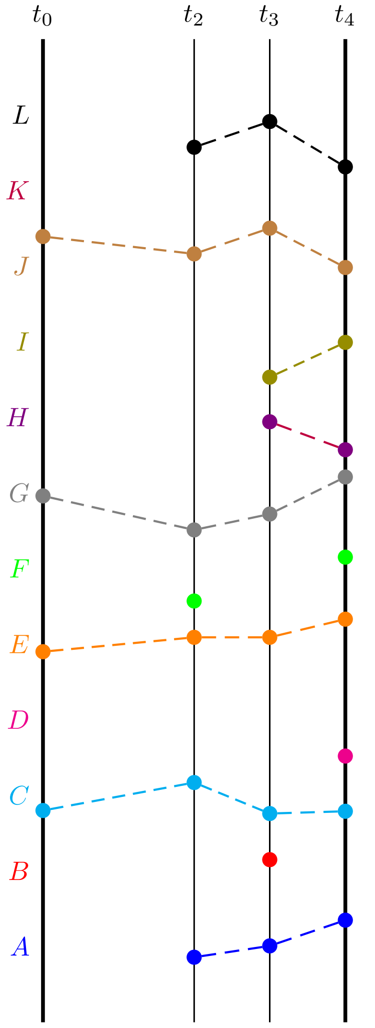
Filter t_2



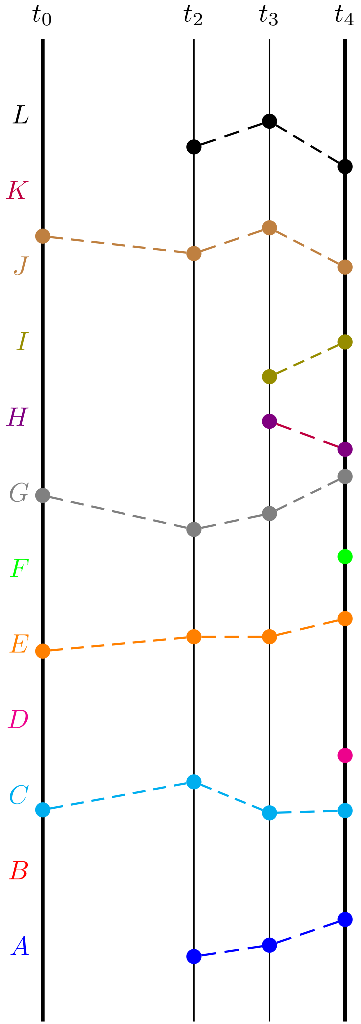
Maximal disks t_3



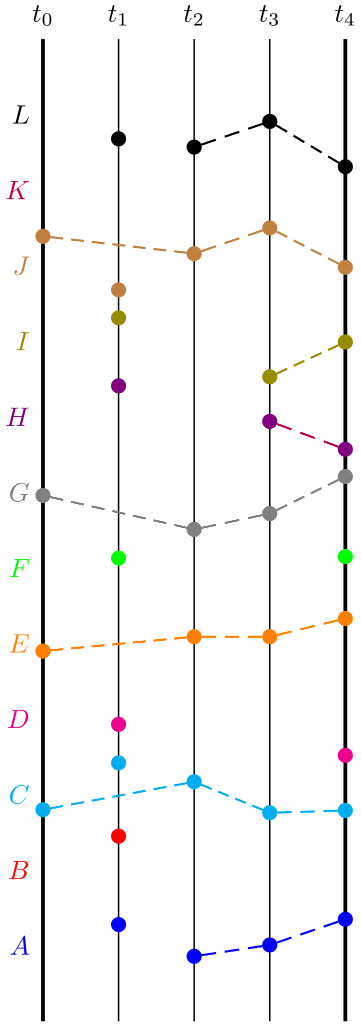
Join t_3



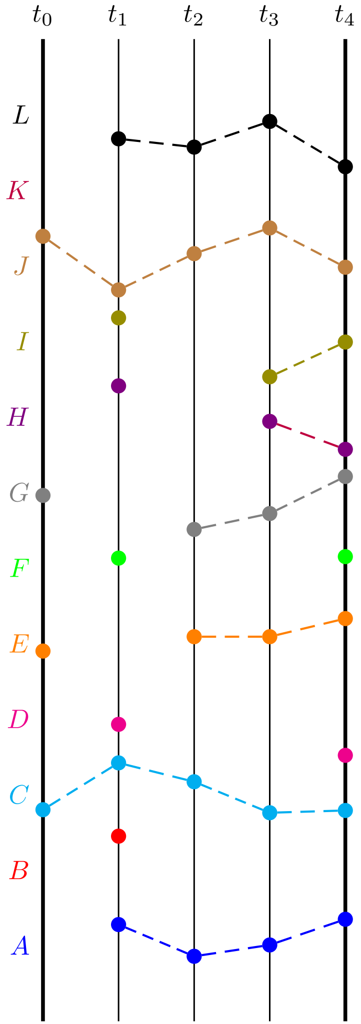
Filter t_3



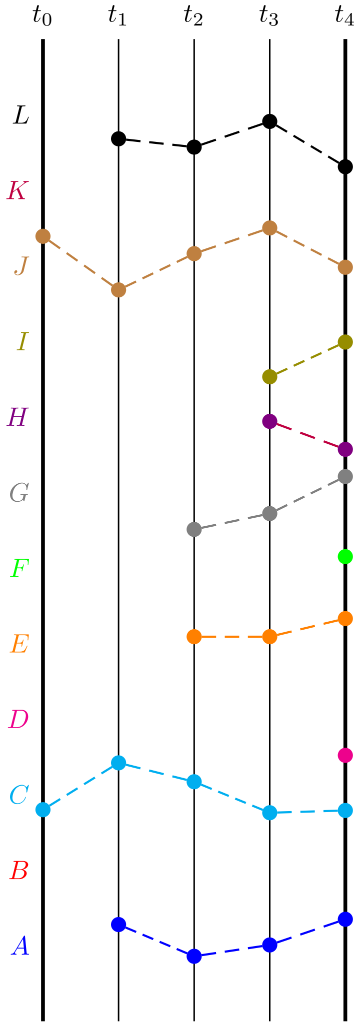
Maximal disks t_1



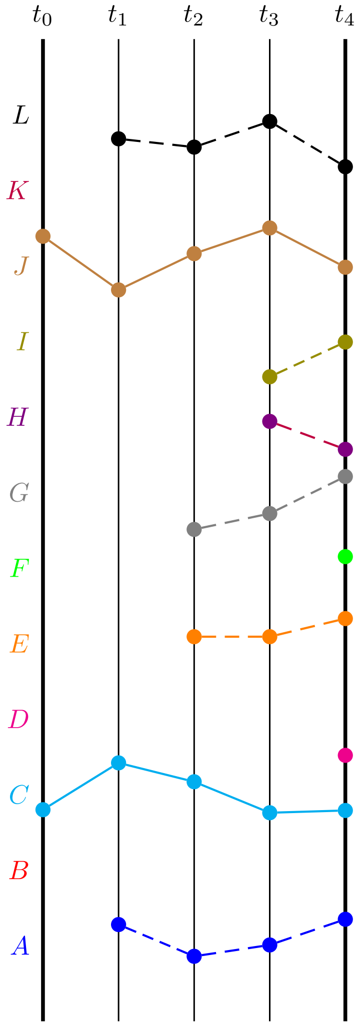
Join t_1



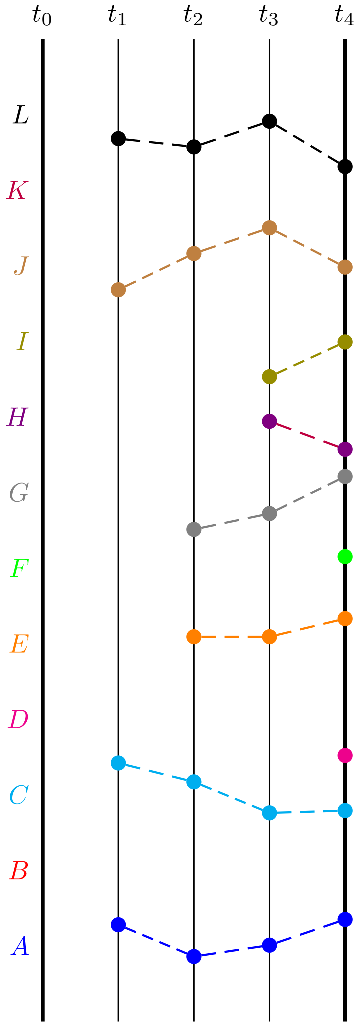
Filter t_1



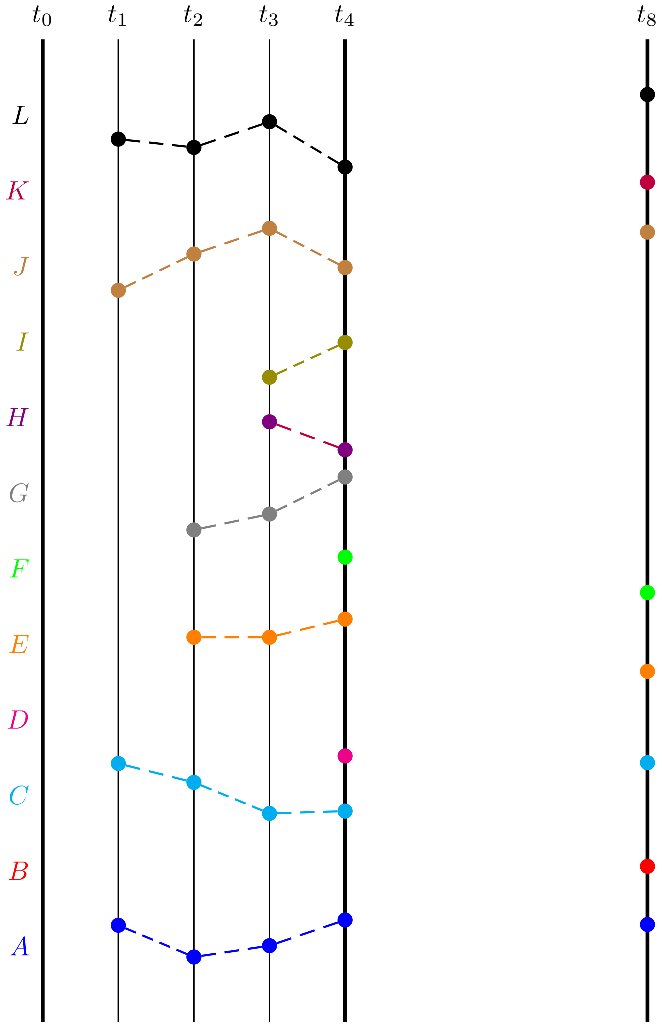
Flocks t_0 - t_4



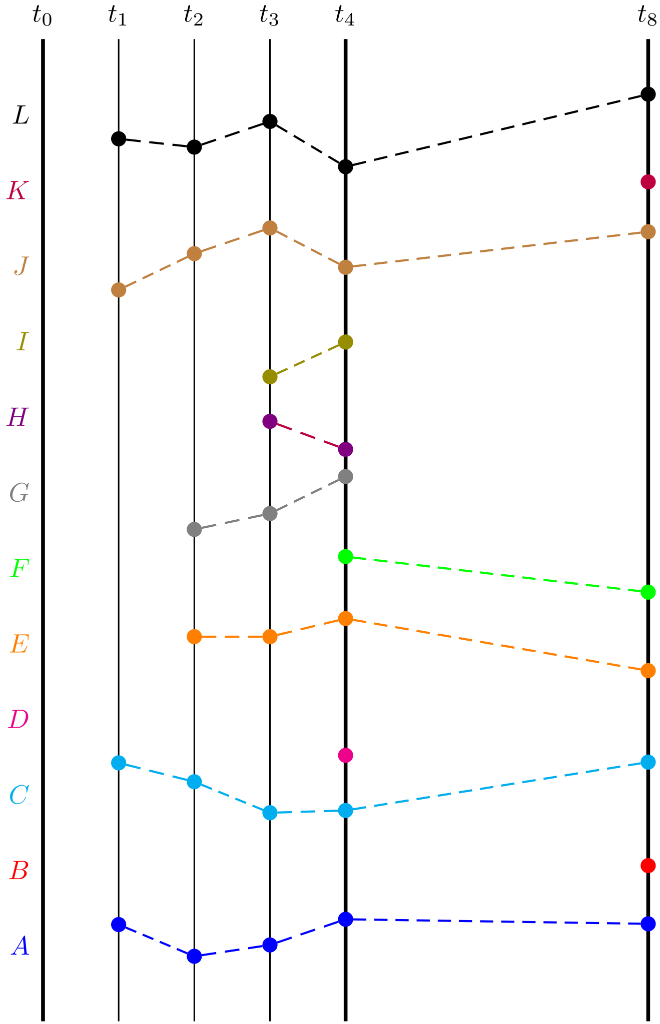
Prune t_0 - t_4



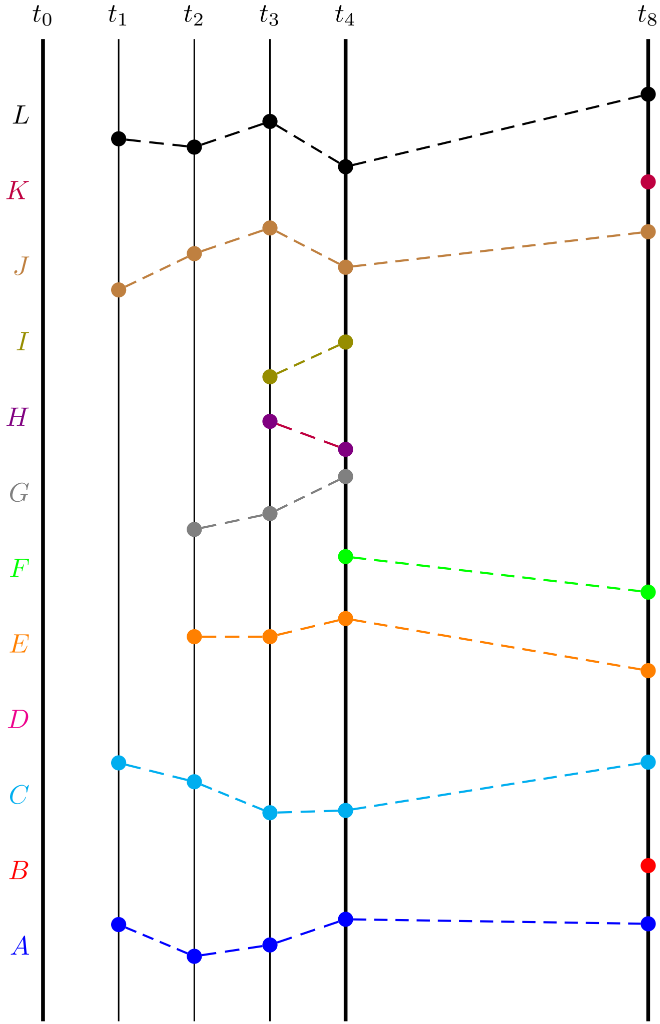
Maximal disks t_8



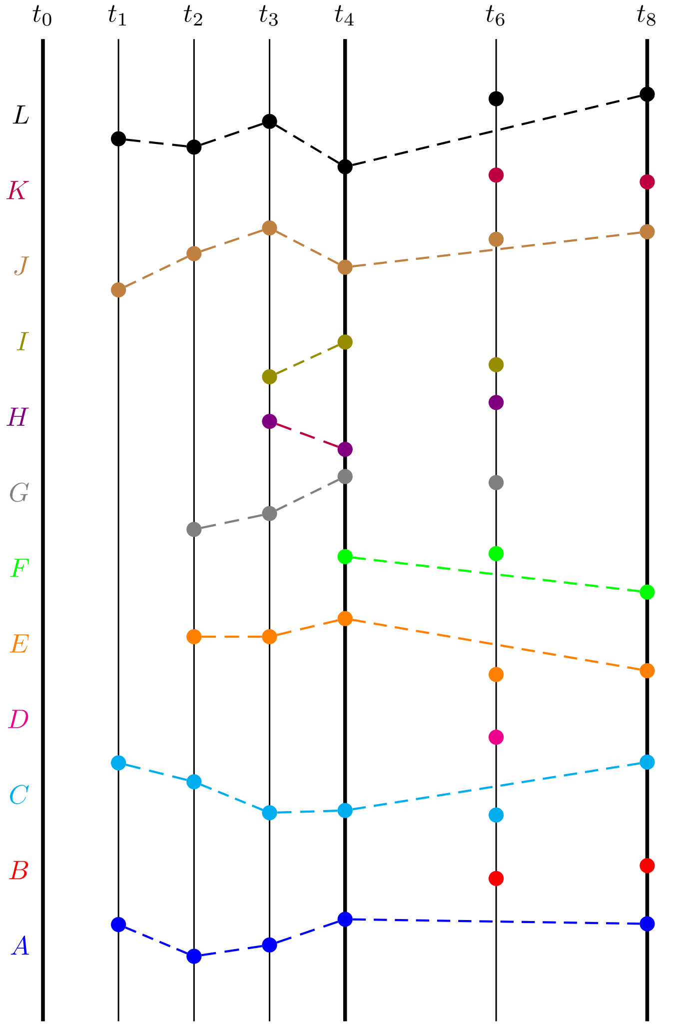
Join t_8



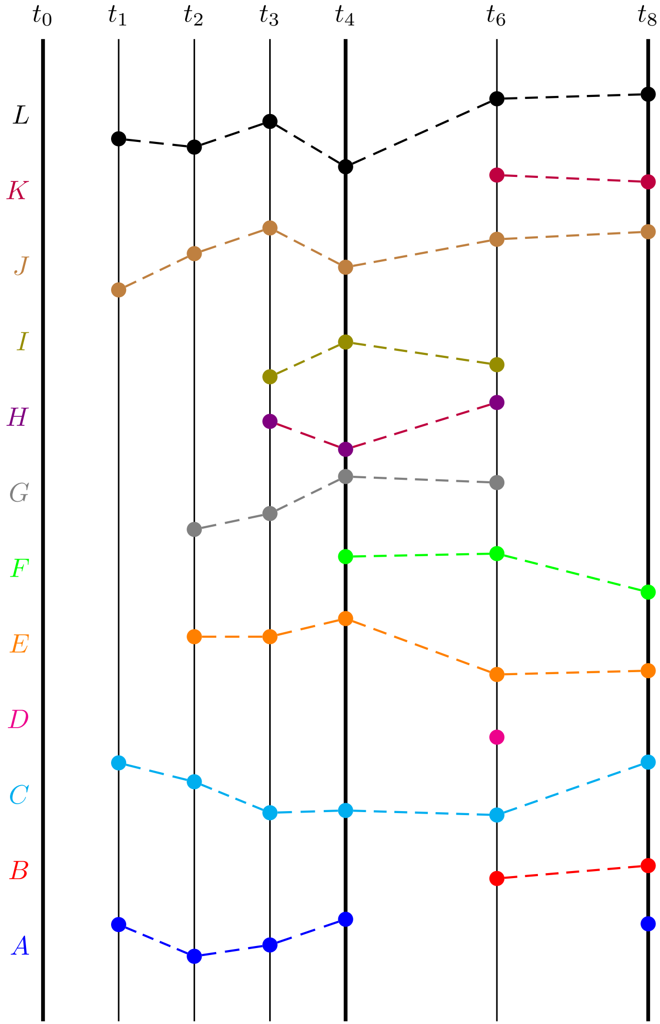
Filter t_8



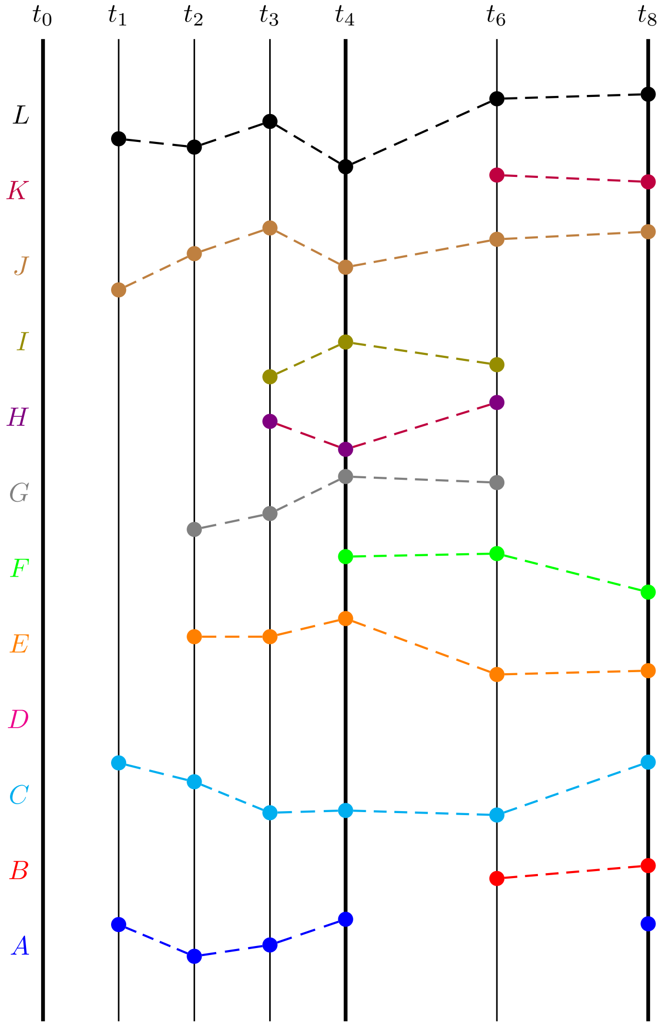
Maximal disks t_6



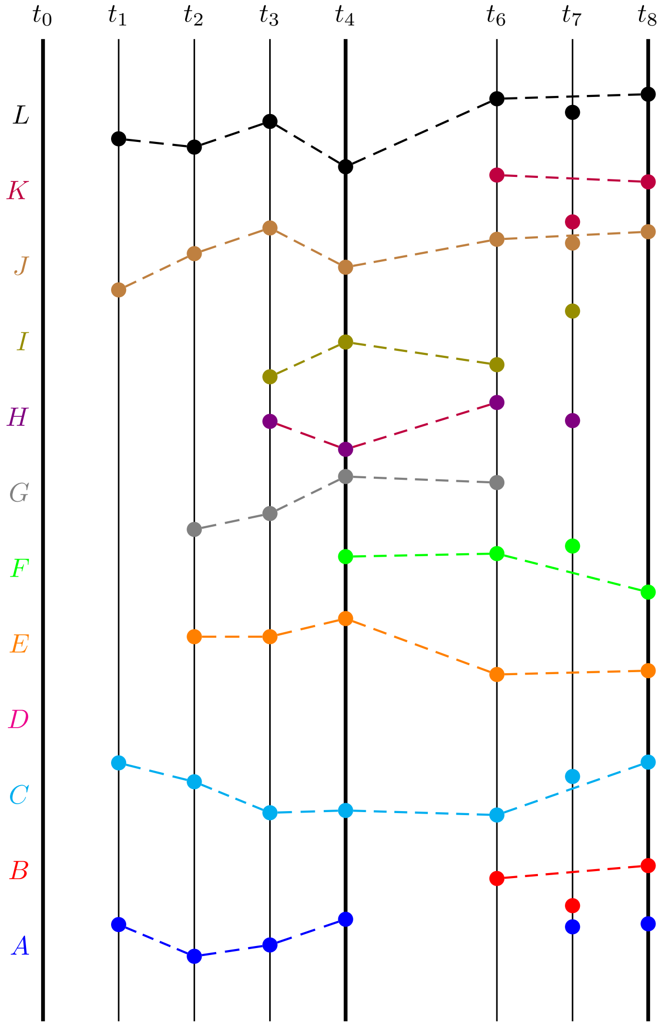
Join t_6



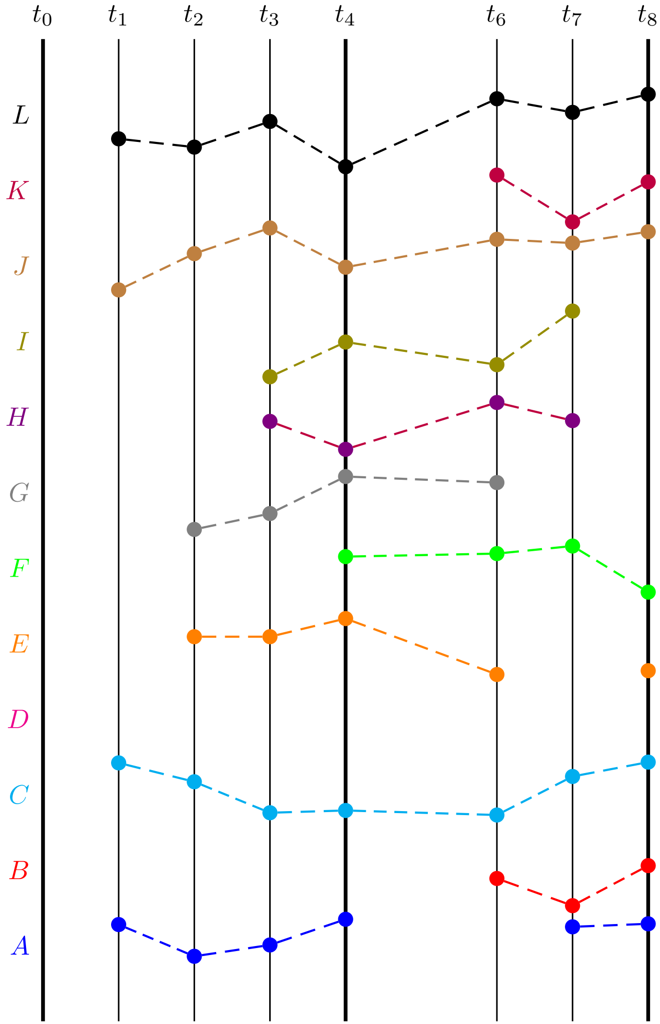
Filter t_6



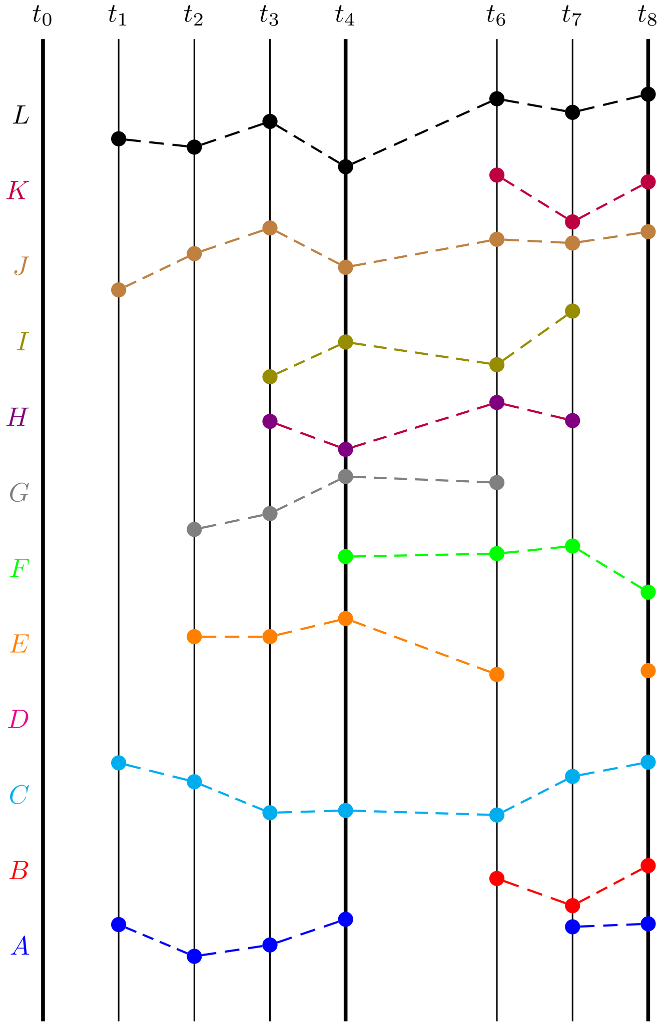
Maximal disks t_7



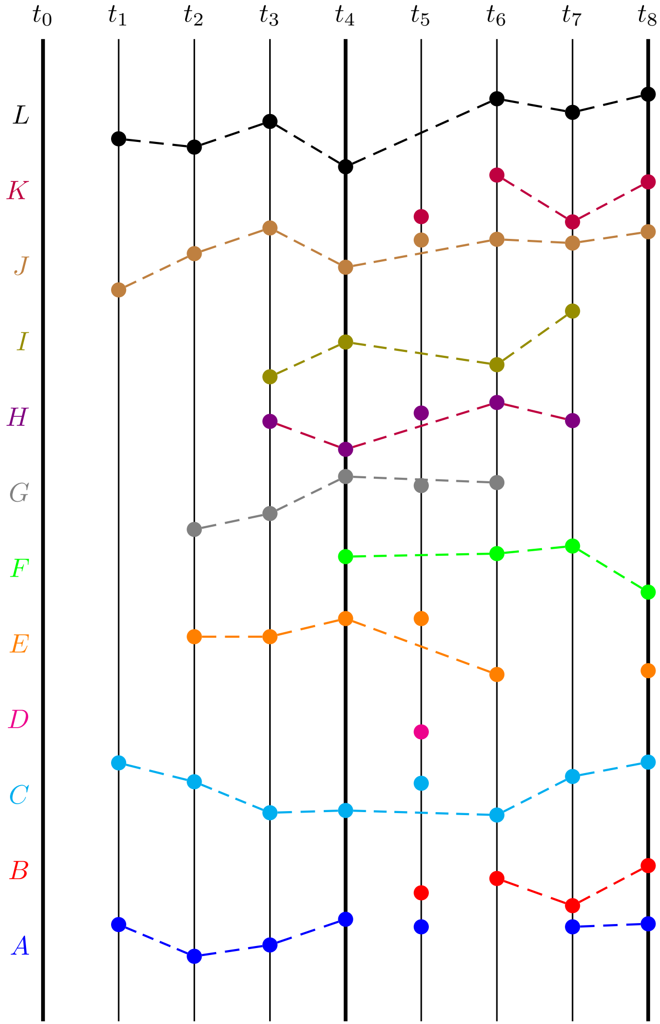
Join t_7



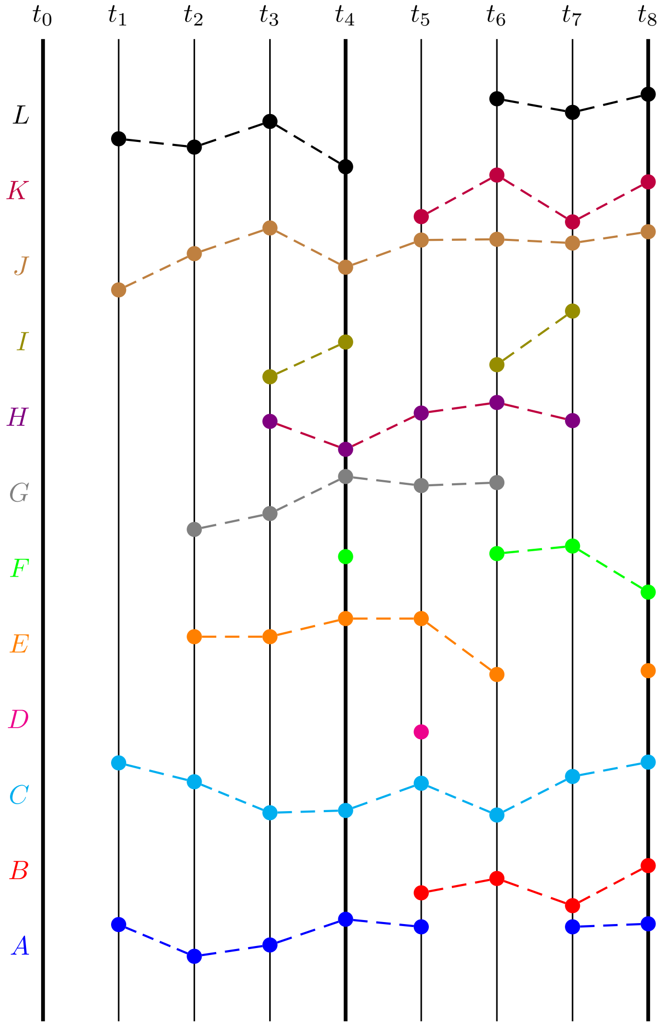
Filter t_7



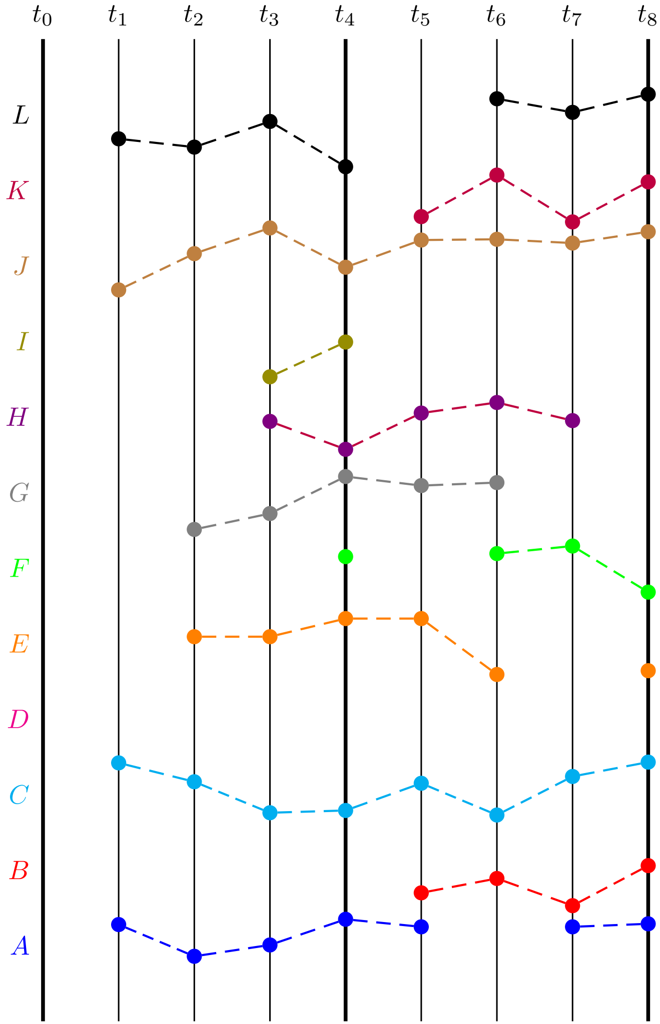
Maximal disks t_5



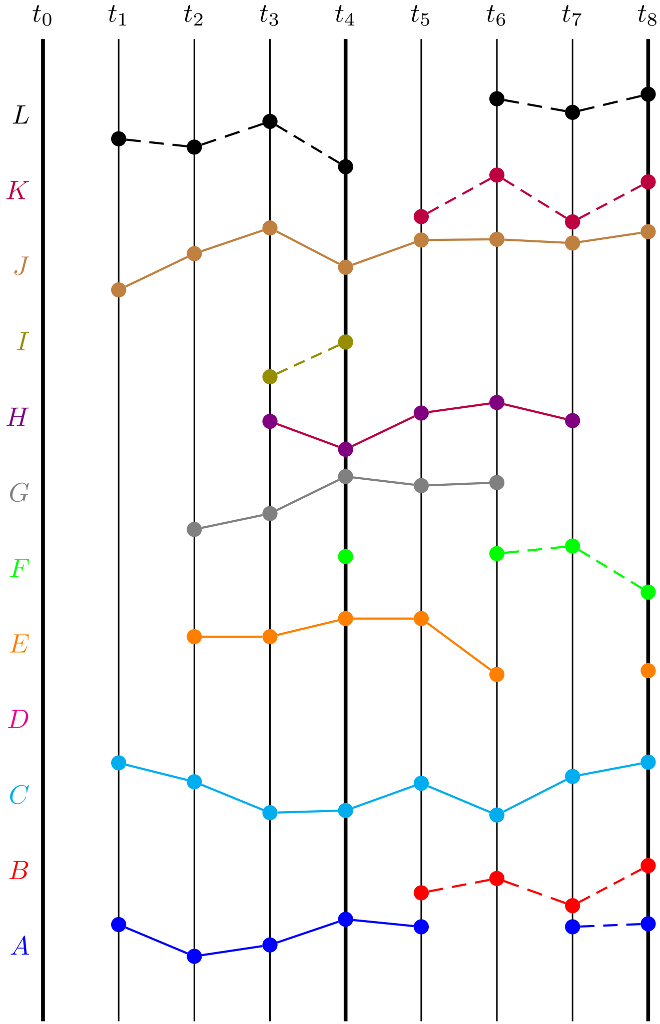
Join t_5



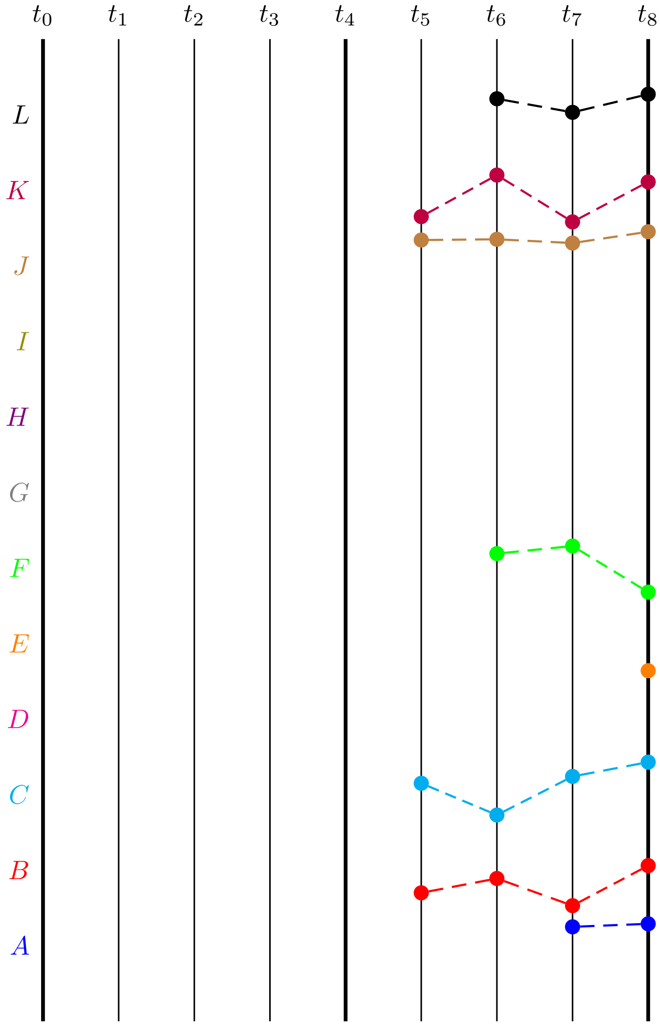
Filter t_5



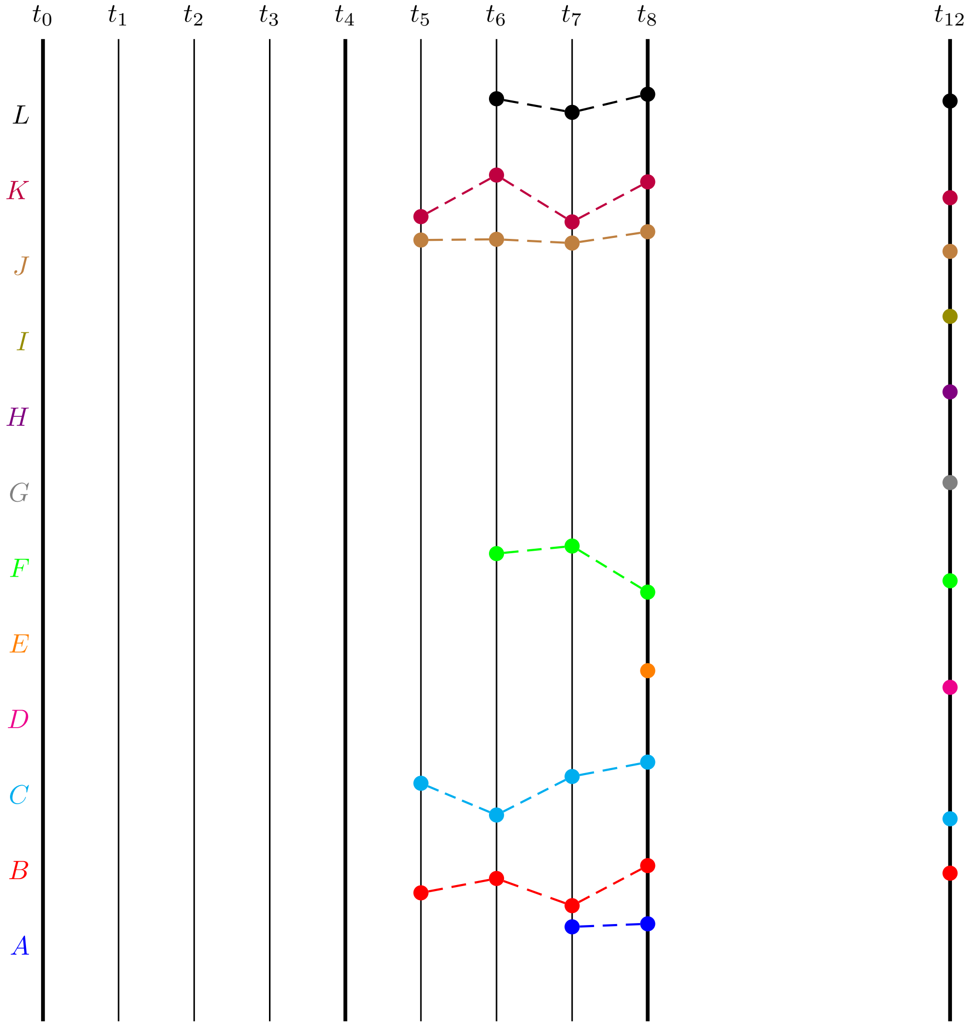
Flocks t_4 - t_8



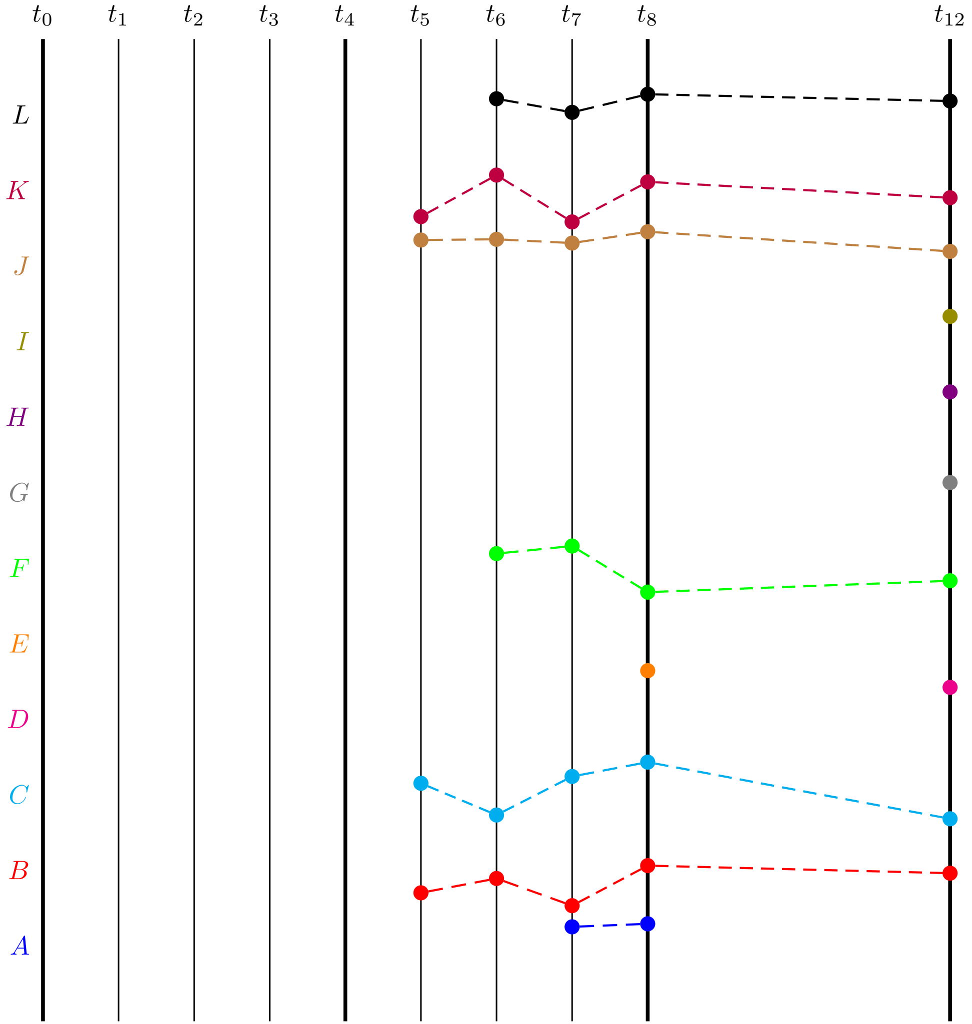
Prune t_4 - t_8



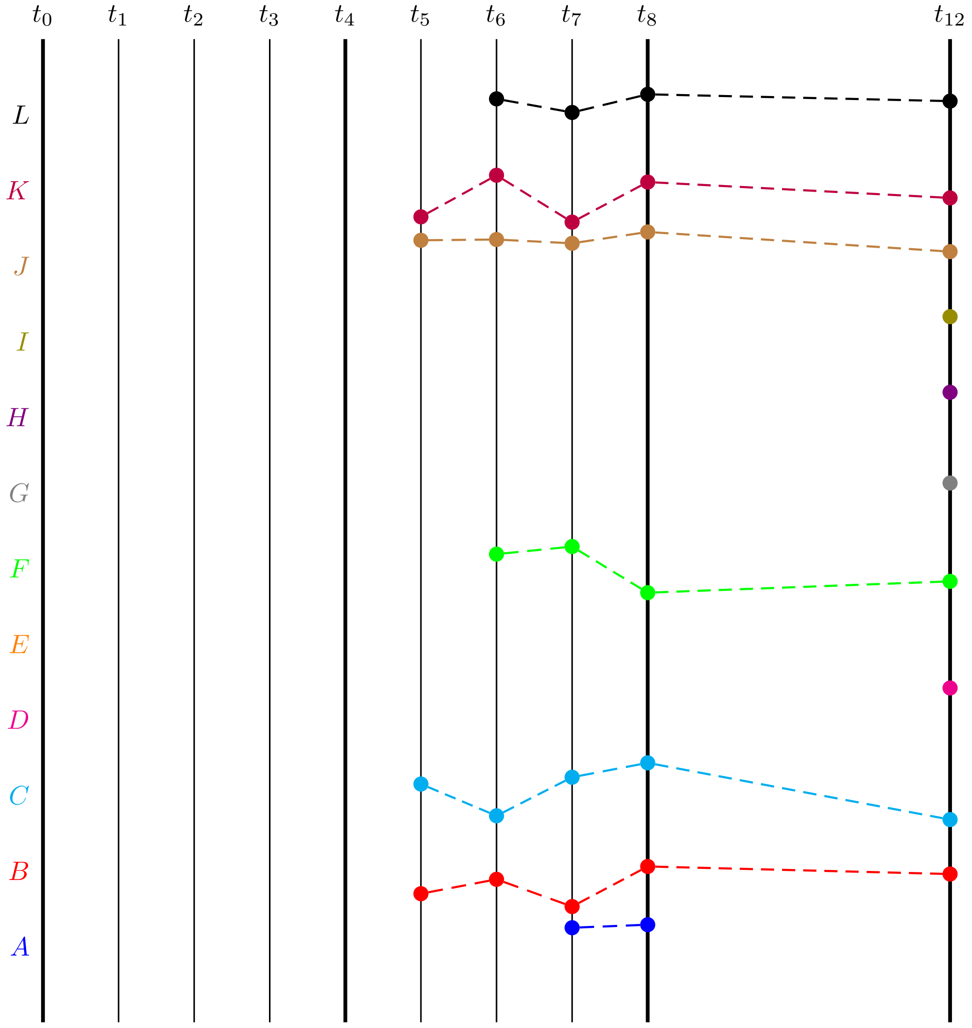
Maximal disks t_{12}



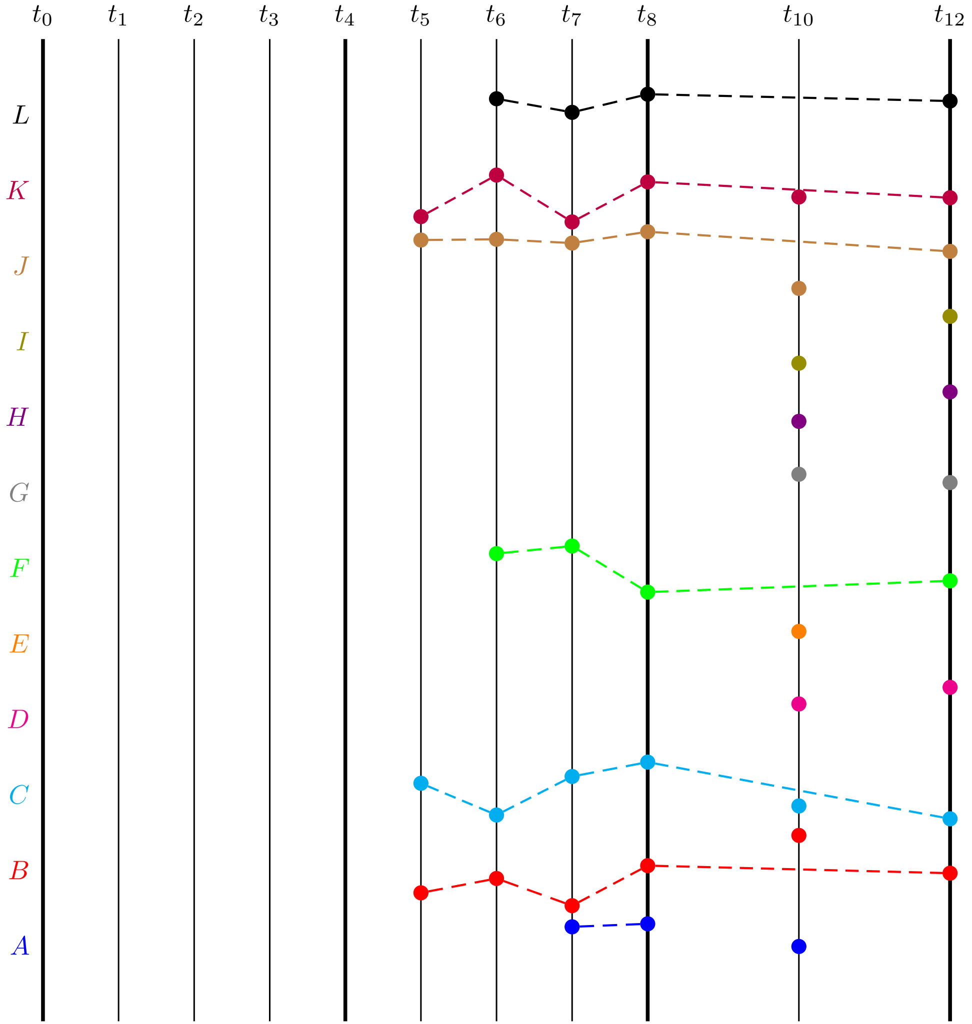
Join t_{12}



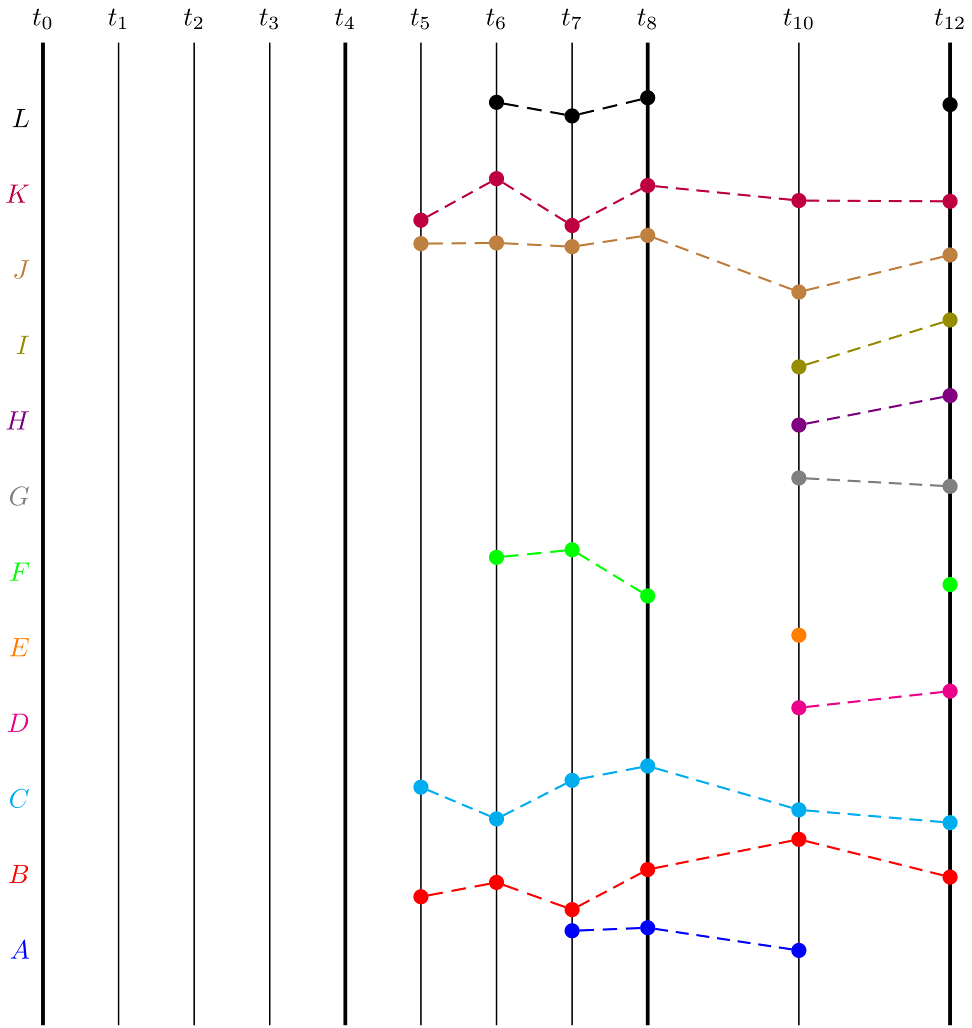
Filter t_{12}



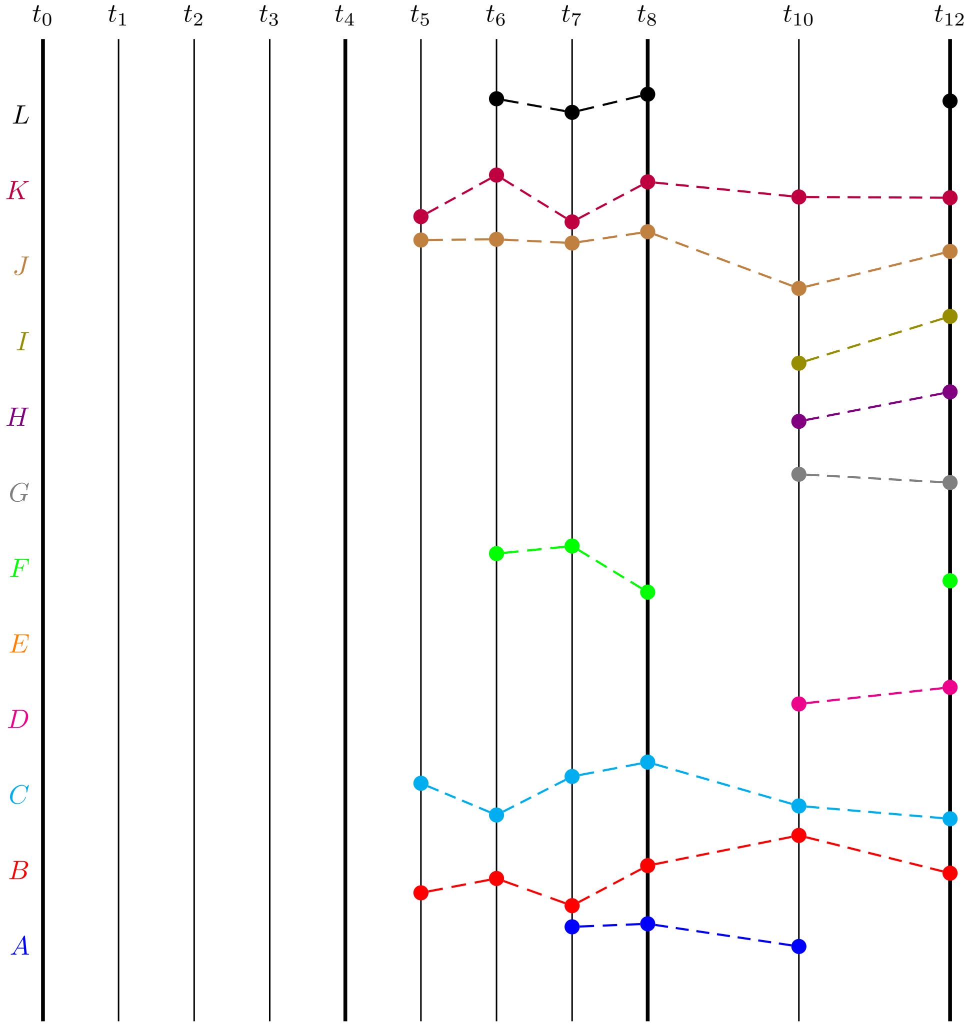
Maximal disks t_{10}



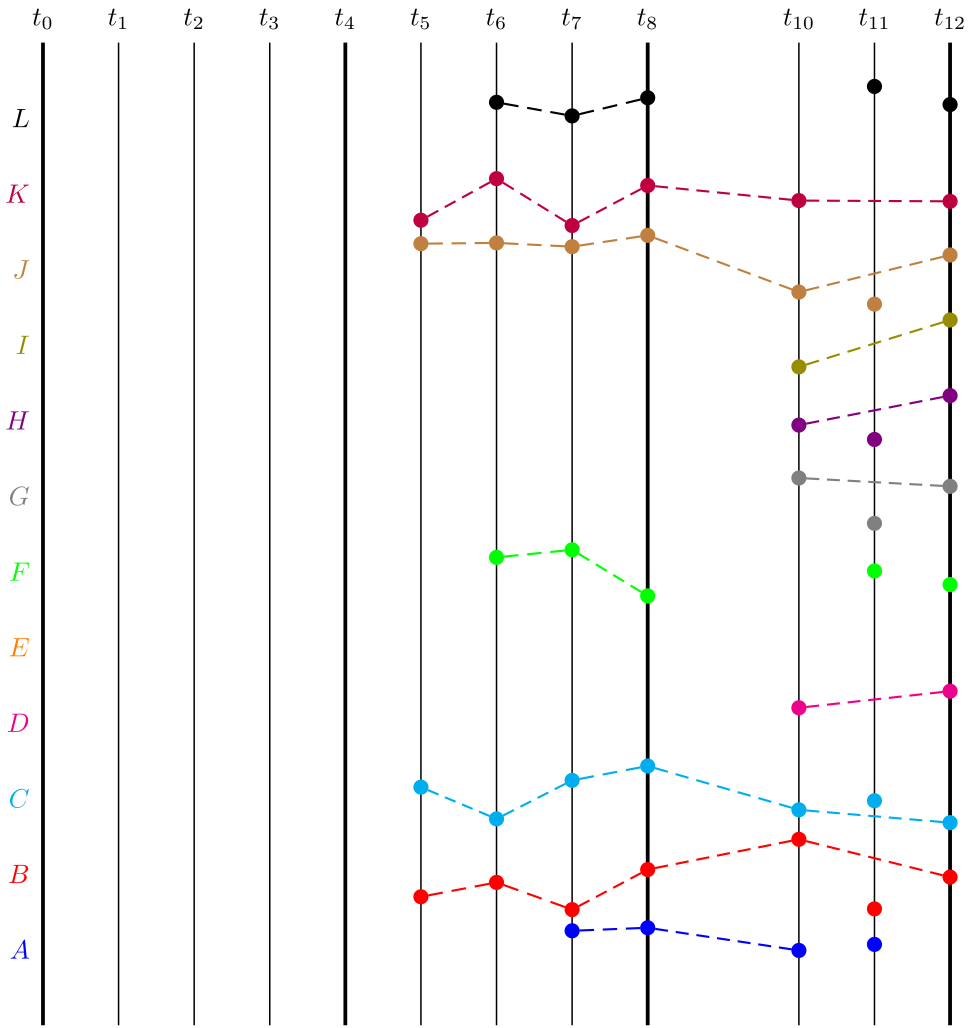
Join t_{10}



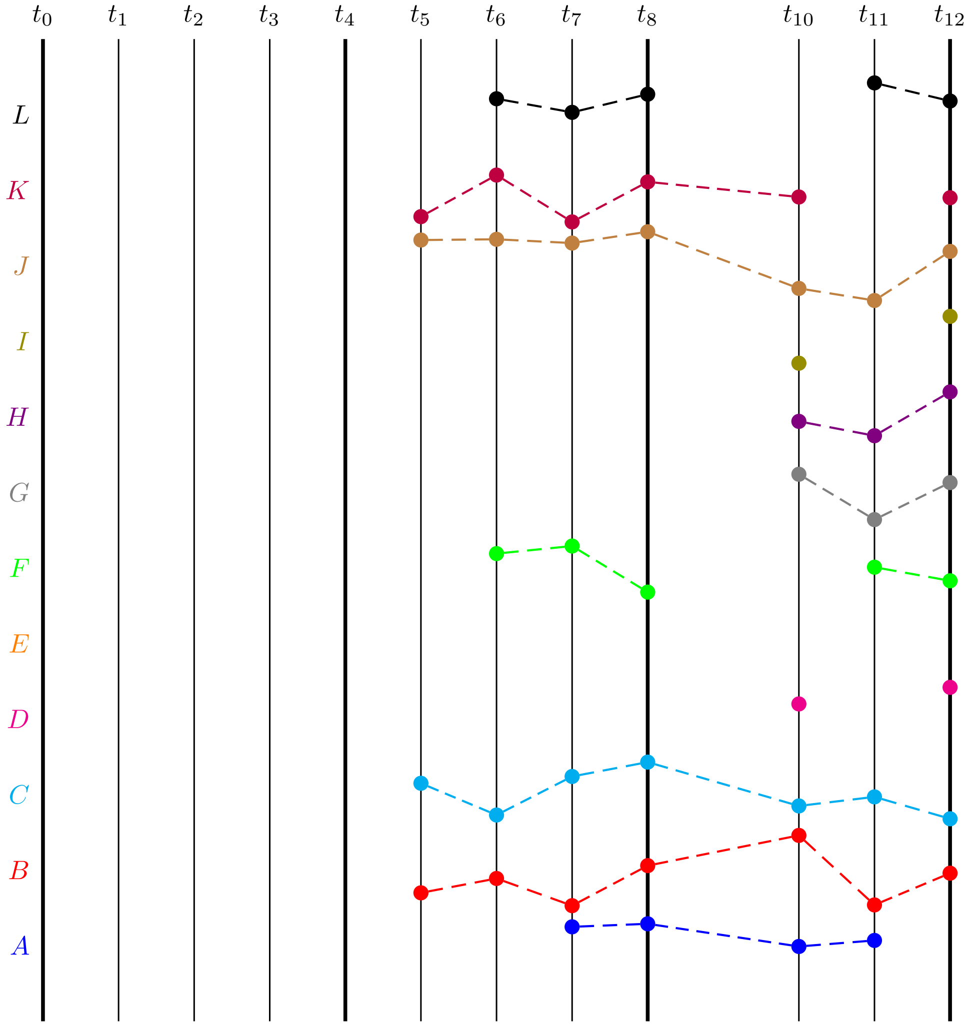
Filter t_{10}



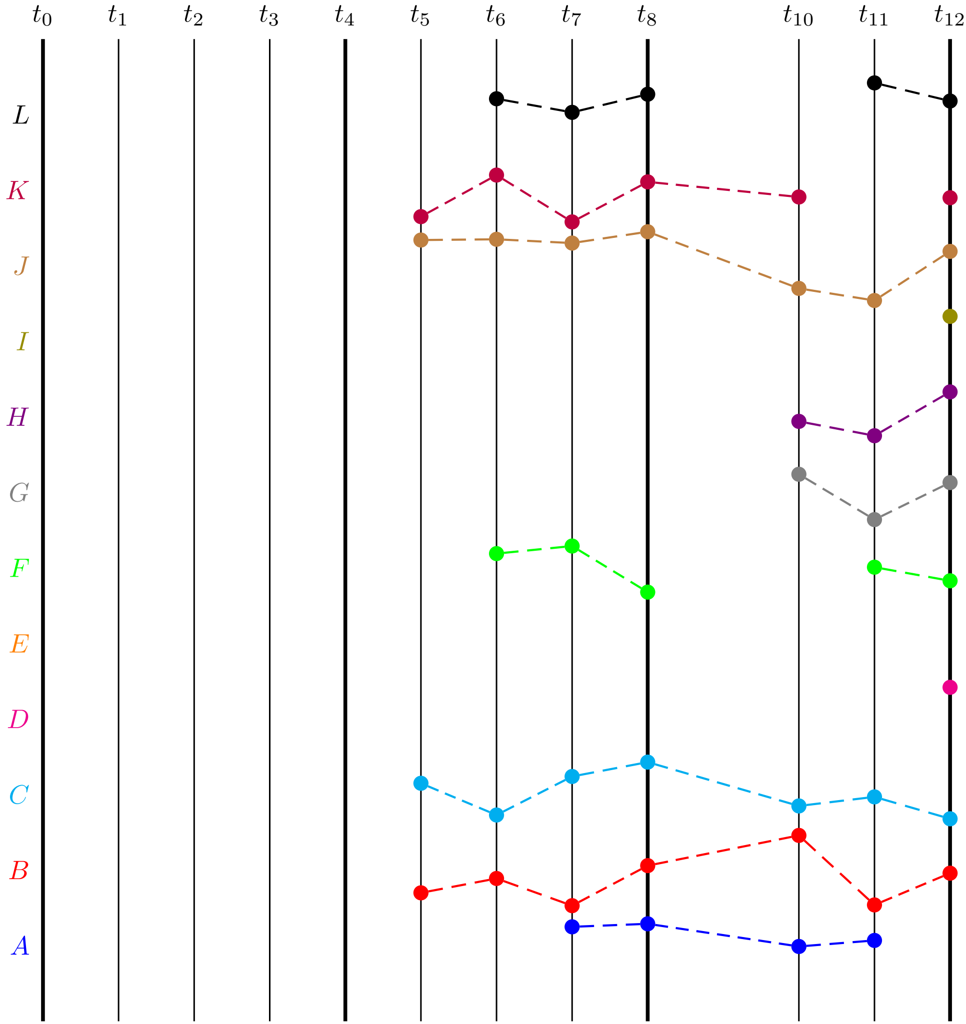
Maximal disks t_{11}



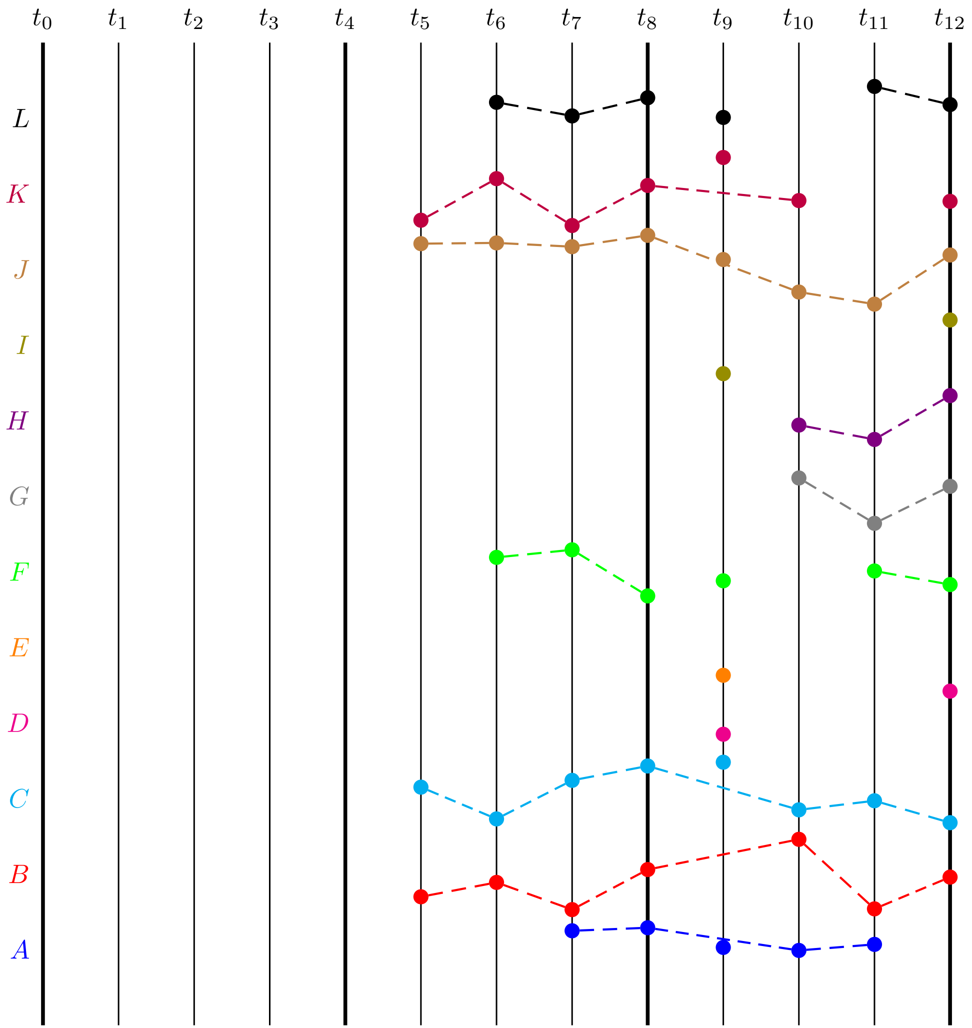
Join t_{11}



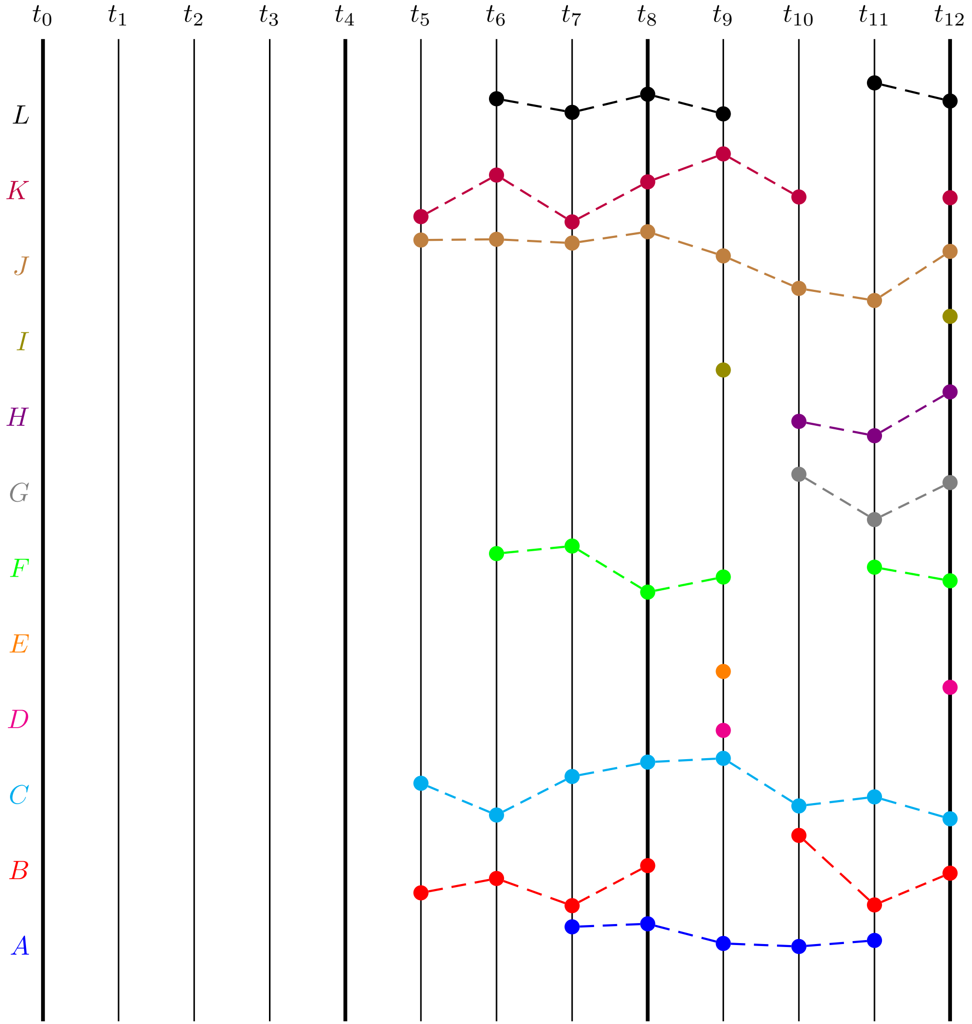
Filter t_{11}



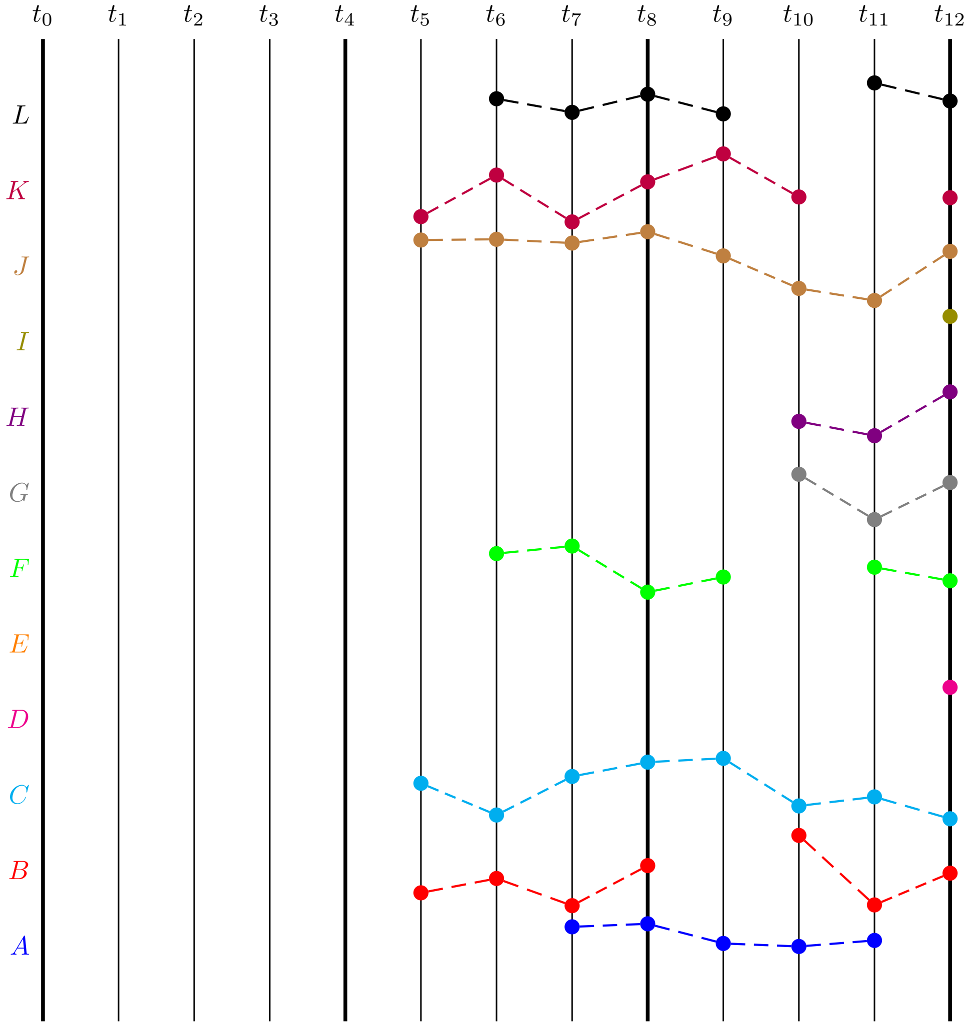
Maximal disks t_9



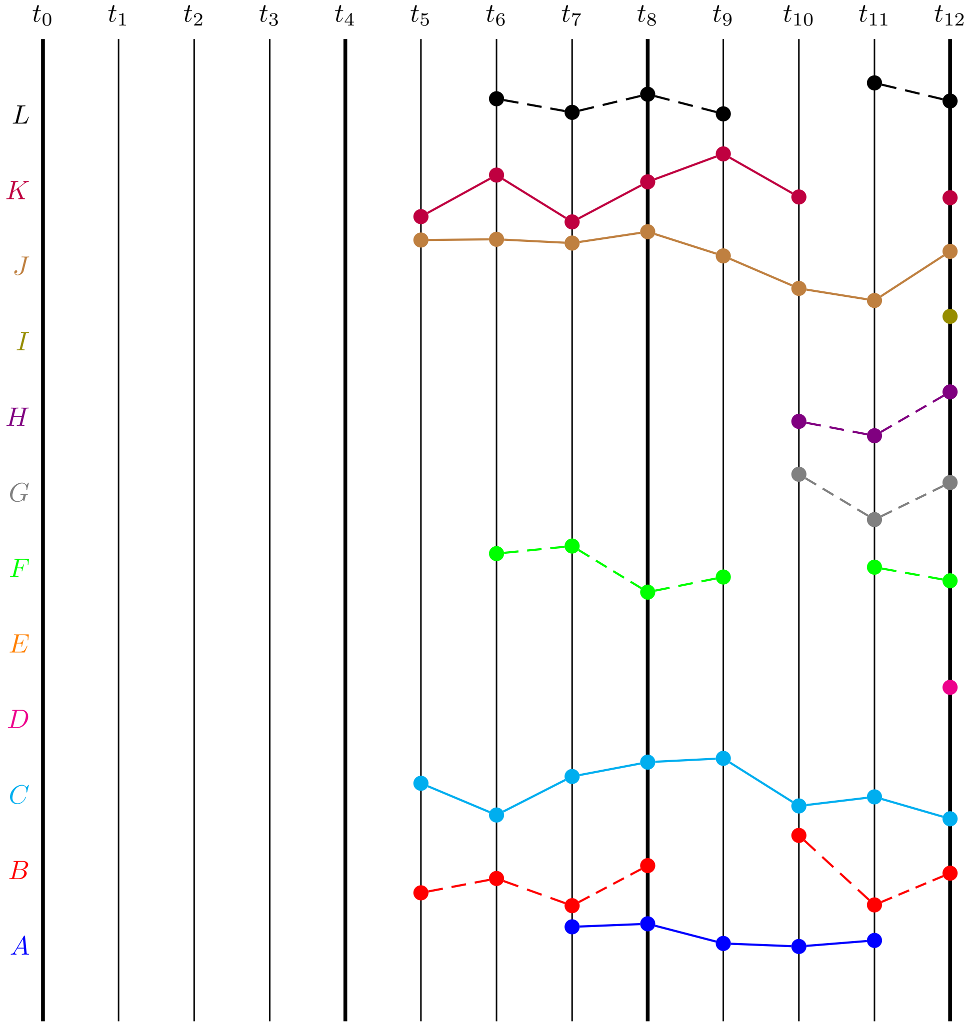
Join t_9



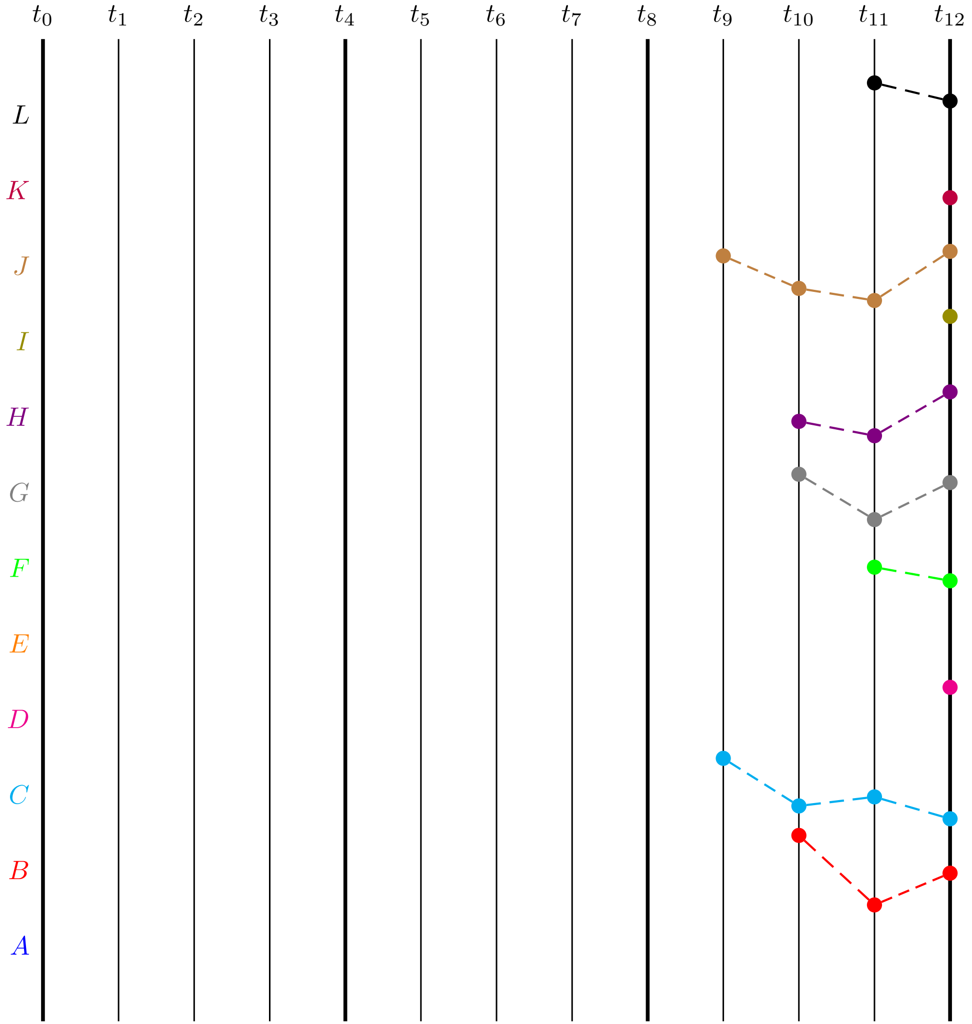
Filter t_9



Flocks t_8 - t_{12}



Prune t_8 - t_{12}



MergeLast issues

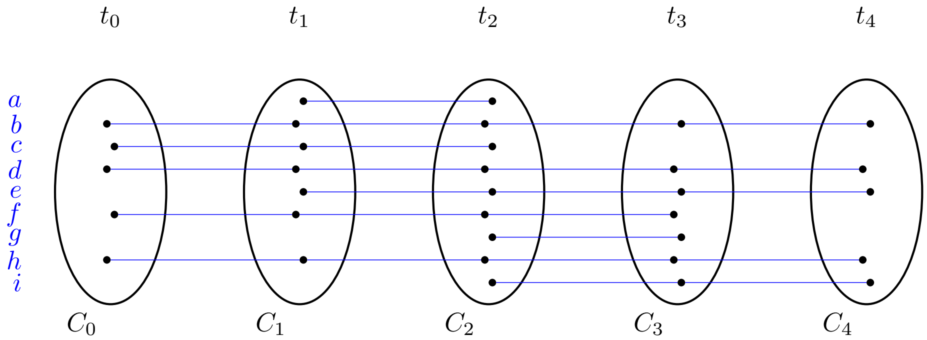
Drawbacks:

- Has to store intermediate points to deal appropriately with ‘holes’ in flocks.
- Join, subset elimination and consecutive checking at each timestamp.
- Expensive operations to maintain set of candidate flocks.

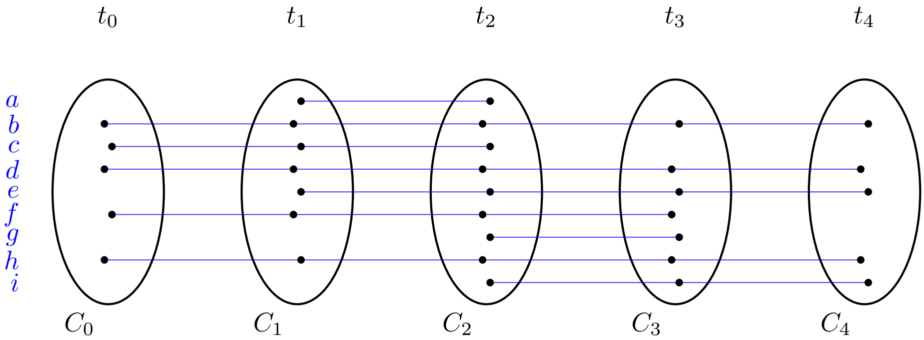
However, it has some interesting features to keep in mind:

- Performs an early pruning of candidate flocks which do not touch the borders.
- Keeping tracking of candidates which touch the borders ease a parallel implementation.

From trajectories to transactions

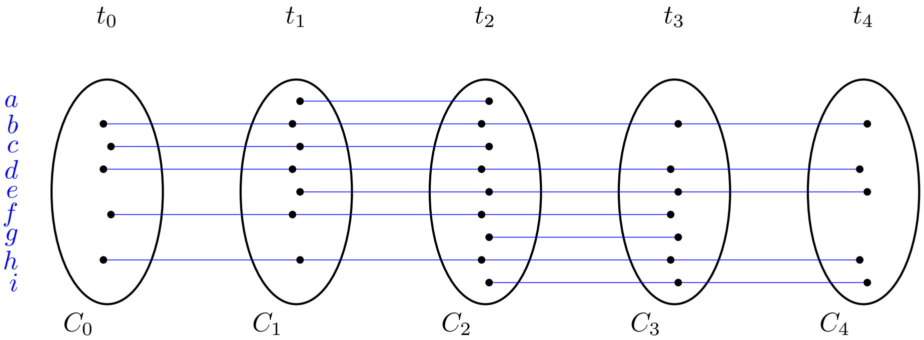


From trajectories to transactions



a		$\langle C_1,$	$C_2\rangle$		
b	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
c	$\langle C_0,$	$C_1,$	$C_2\rangle$		
d	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
e		$\langle C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
f	$\langle C_0,$	$C_1,$	$C_2,$	$C_3\rangle$	
g			$\langle C_2,$	$C_3\rangle$	
h	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
i			$\langle C_2,$	$C_3,$	$C_4\rangle$

From trajectories to transactions

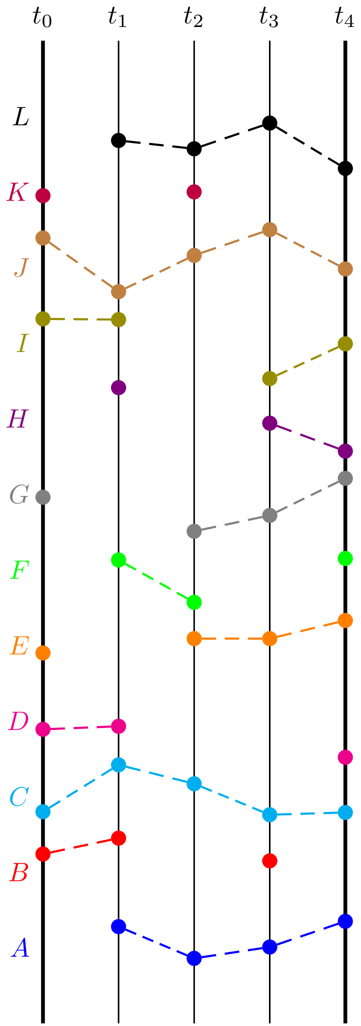


a		$\langle C_1,$	$C_2\rangle$		
b	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
c	$\langle C_0,$	$C_1,$	$C_2\rangle$		
d	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
e		$\langle C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
f	$\langle C_0,$	$C_1,$	$C_2,$	$C_3\rangle$	
g			$\langle C_2,$	$C_3\rangle$	
h	$\langle C_0,$	$C_1,$	$C_2,$	$C_3,$	$C_4\rangle$
i			$\langle C_2,$	$C_3,$	$C_4\rangle$

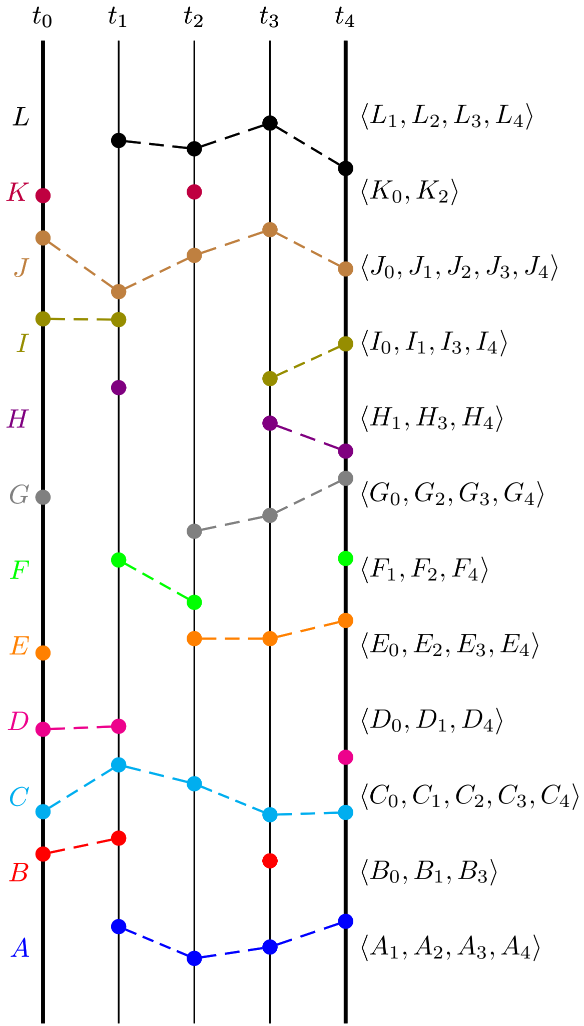
If we apply a Maximal Pattern (MP) algorithm over the new transactions...

$$MP = \langle C_0, \quad C_1, \quad C_2, \quad C_3, \quad C_4\rangle \quad : 3(min\_sup \geq \mu)$$

MP finding per window...



MP finding per window...



MP finding per window...

Each trajectory is associated with just the maximal disks it touches. MP algorithms returns sets of disks which are visited by the same trajectories. If they happen in consecutive order, it is a flock.

Pros:

- Do not perform distance join at each timestamp.
- Although still have to deal with consecutive checking, it is done just at the end of the window.
- It deals with subset elimination.

Cons:

- Overlapping disks could introduce false flocks. It will require an additional filter at the end of the window.

Some reading...

- B. Negrevergne, A. Termier, J.-F. Mhaut, and T. Uno, Discovering closed frequent itemsets on multicore: Parallelizing computations and optimizing memory accesses, in High Performance Computing and Simulation (HPCS), 2010 International Conference on, 2010, pp. 521528.
- M. Kirchgessner, Mining and ranking closed itemsets from large-scale transactional datasets, Universit Grenoble Alpes, 2016.
- S. Cong, J. Han, and D. Padua, Parallel mining of closed sequential patterns, in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, pp. 562567.