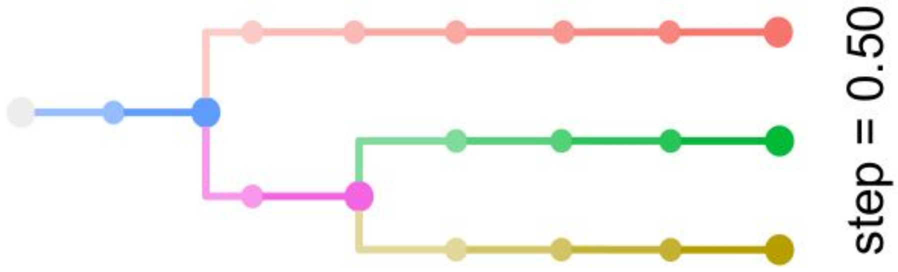
TedSim

TedSim uses 2 trees to calculate the expression profile of a cell:

Only adata.obs['depth'] captures this

State tree:

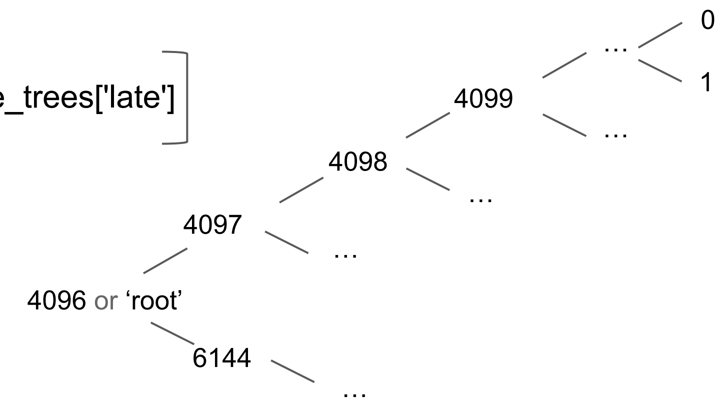
A cell progresses on the state tree only if it was selected for asymmetric division (pa), and in that case the will go up to max_walk states down the state tree.



Lineage tree:

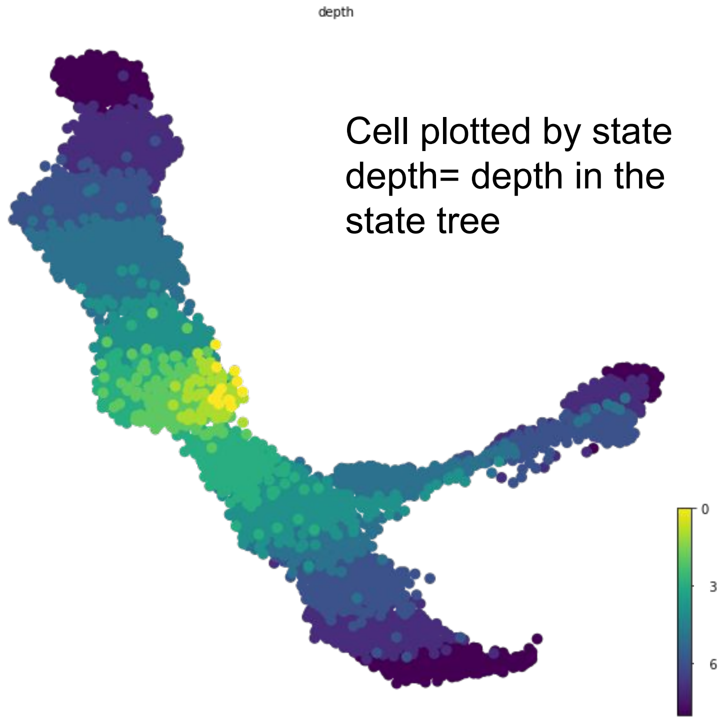
Cells always progresses one step on the lineage tree.

G or true_trees['late']



TedSim - realistic looking data

The expression profile of a cell is calculated using both the state and the lineage tree. We have to use the lineage tree to calculate the true coupling. But the clustering according to state depth looks much better.

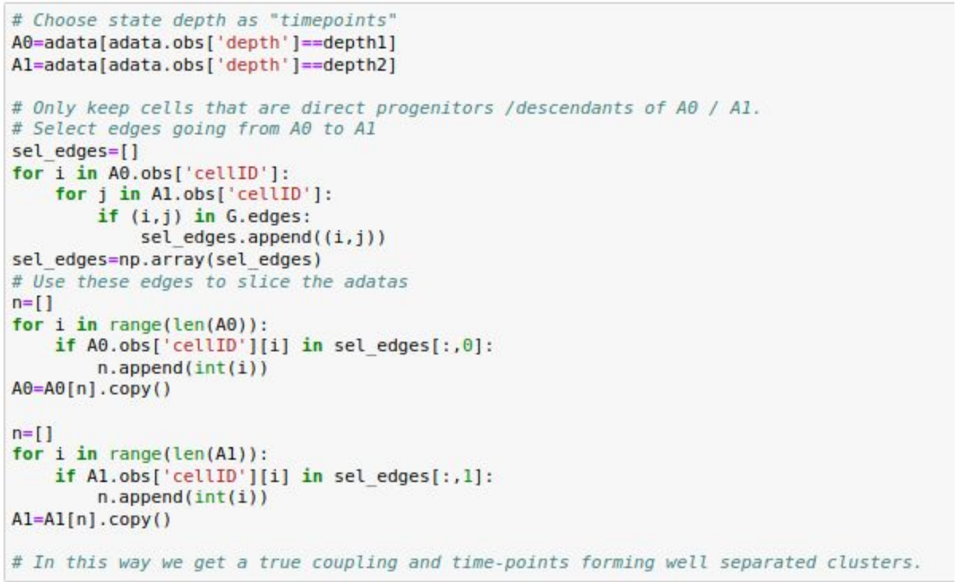


Cell plotted by state depth= depth in the state tree

TedSim - realistic looking data

My approach:

- Use state depth as “time points” (denoted as depth1 and depth2).
- Only keep cells that form an edge in the lineage tree, i.e. remove all the cells that are not direct descendants/progenitors of cells from depth1/depth2.
- In this way the core computation is like in moscot-Lineage, just the clusters selected as start /end points are different.

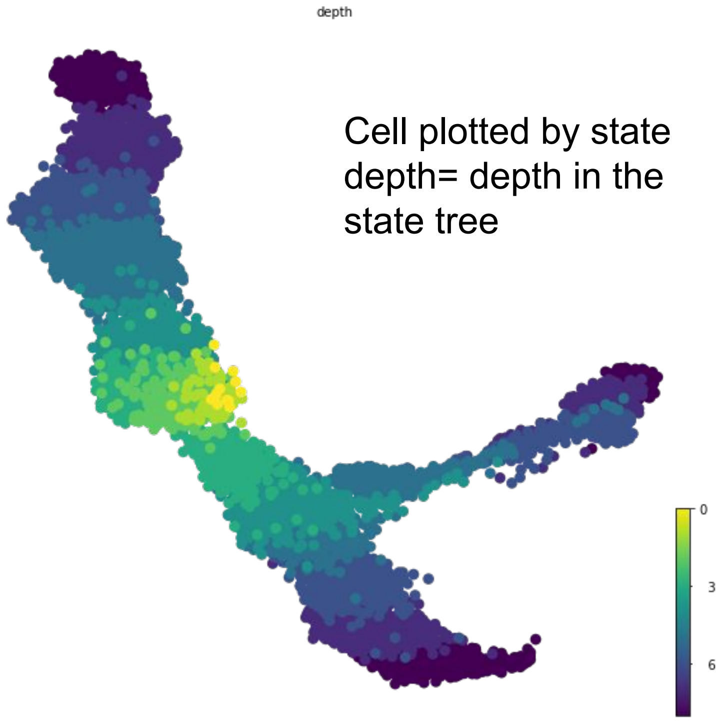


```python
# Choose state depth as "timepoints"
A0=adata[adata.obs['depth']==depth1]
A1=adata[adata.obs['depth']==depth2]

# Only keep cells that are direct progenitors /descendants of A0 / A1.
# Select edges going from A0 to A1
sel_edges=[]
for i in A0.obs['cellID']:
    for j in A1.obs['cellID']:
        if (i,j) in G.edges:
            sel_edges.append((i,j))
sel_edges=np.array(sel_edges)
# Use these edges to slice the adatas
n=[]
for i in range(len(A0)):
    if A0.obs['cellID'][i] in sel_edges[:,0]:
        n.append(int(i))
A0=A0[n].copy()

n=[]
for i in range(len(A1)):
    if A1.obs['cellID'][i] in sel_edges[:,1]:
        n.append(int(i))
A1=A1[n].copy()

# In this way we get a true coupling and time-points forming well separated clusters.
```



Cell plotted by state depth= depth in the state tree