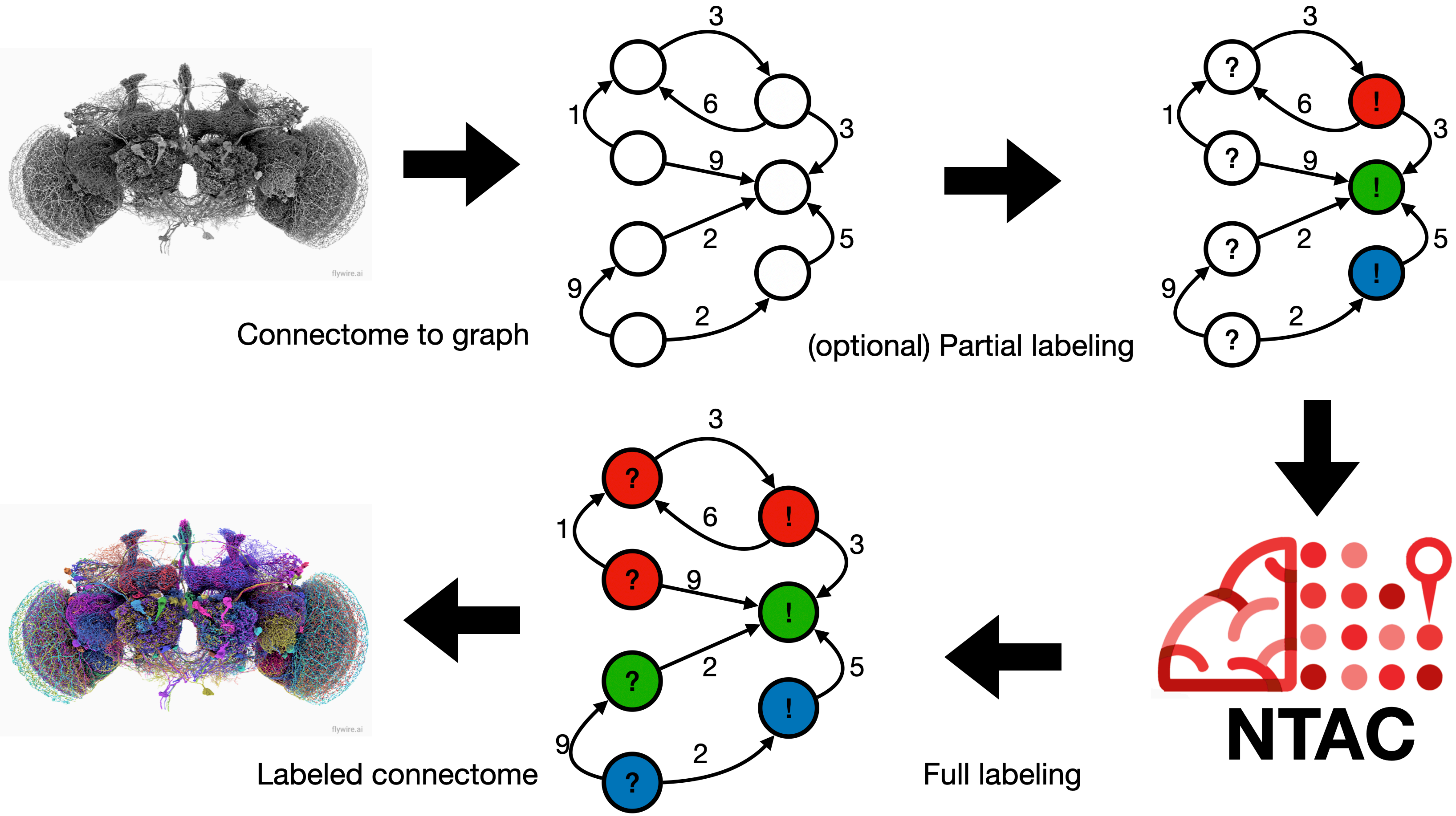
Connectivity Is All You Need: Inferring Neuronal Types with NTAC

Gregory Schwartzman, Ben Jourdan, David García-Soriano and Arie Matsliah

Speaker order:
Gregory Schwartzman
Ben Jourdan
David García-Soriano

bioRχiv
THE PREPRINT SERVER FOR BIOLOGY

NTAC - Neuronal Type Assignment from Connectivity



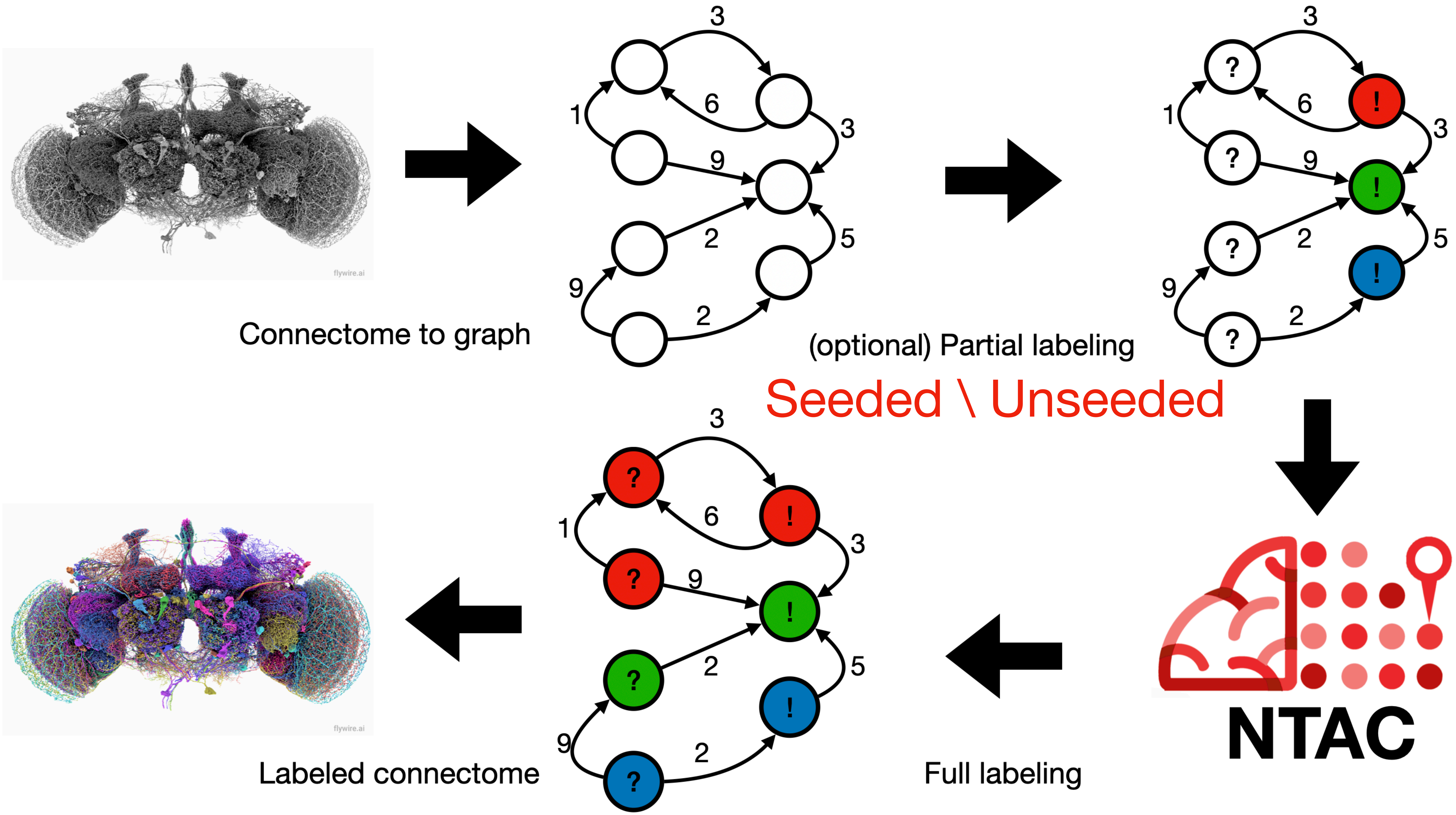
Connectome to graph

(optional) Partial labeling

Full labeling

Labeled connectome

NTAC - Neuronal Type Assignment from Connectivity



Connectome to graph

(optional) Partial labeling

Seeded \ Unseeded

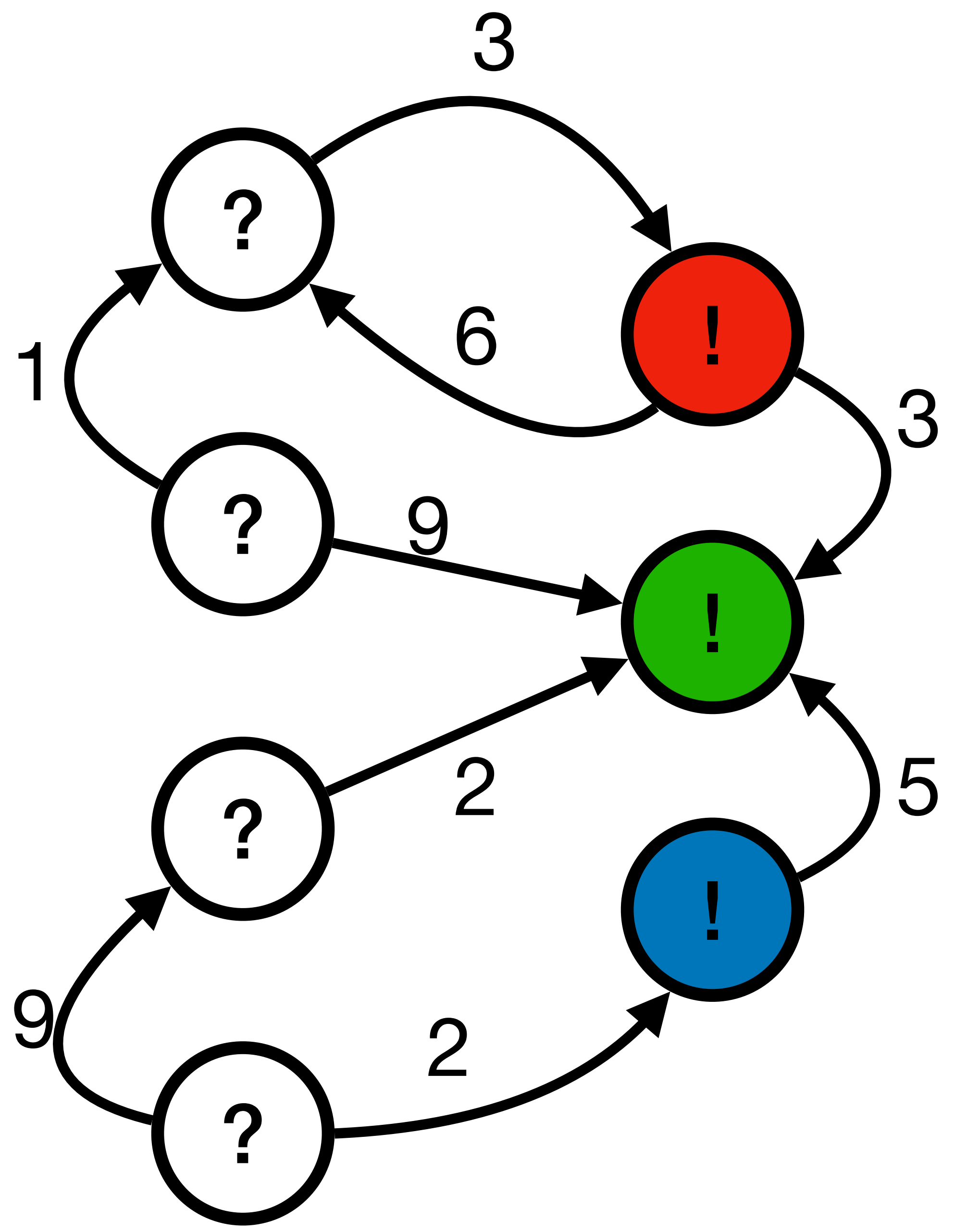
NTAC

Full labeling

Labeled connectome

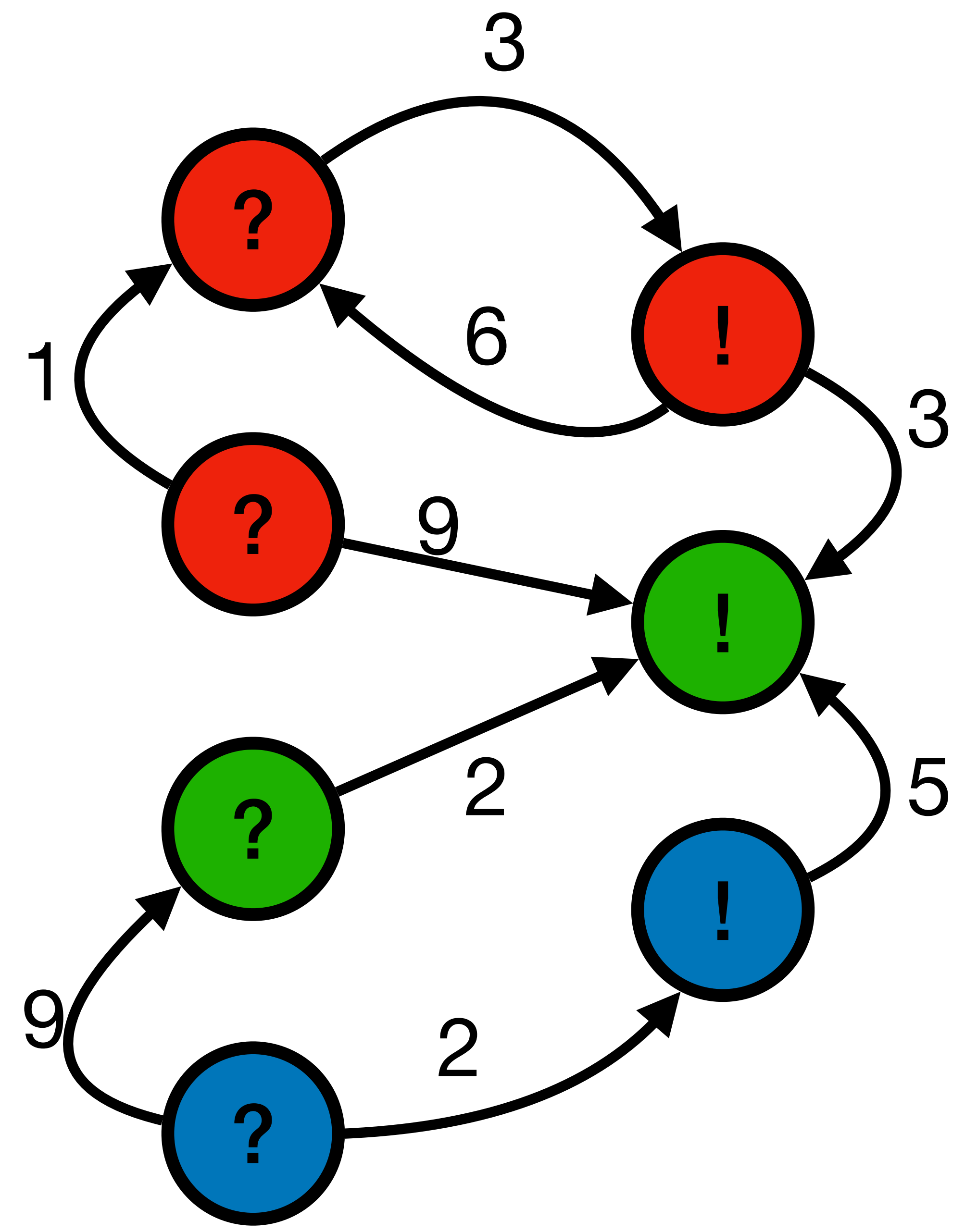
Seeded NTAC - Problem statement

- **Input:** edge weighted directed graph with partial labels
- **Output:** labels for all unlabeled nodes
- **Goal:** maximize accuracy



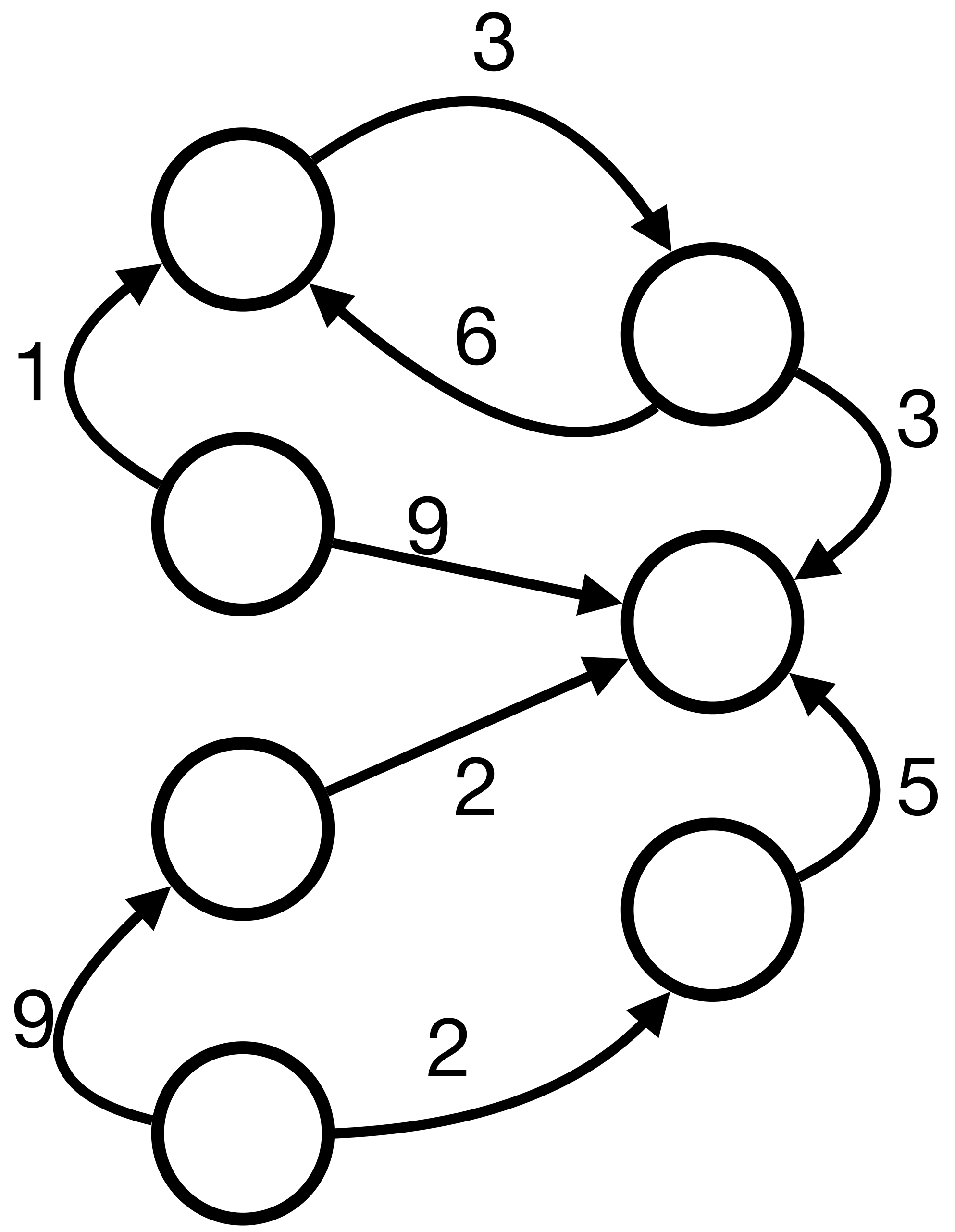
Problem statement

- **Input:** edge weighted directed graph with partial labels
- **Output:** labels for all unlabeled nodes
- **Goal:** maximize accuracy



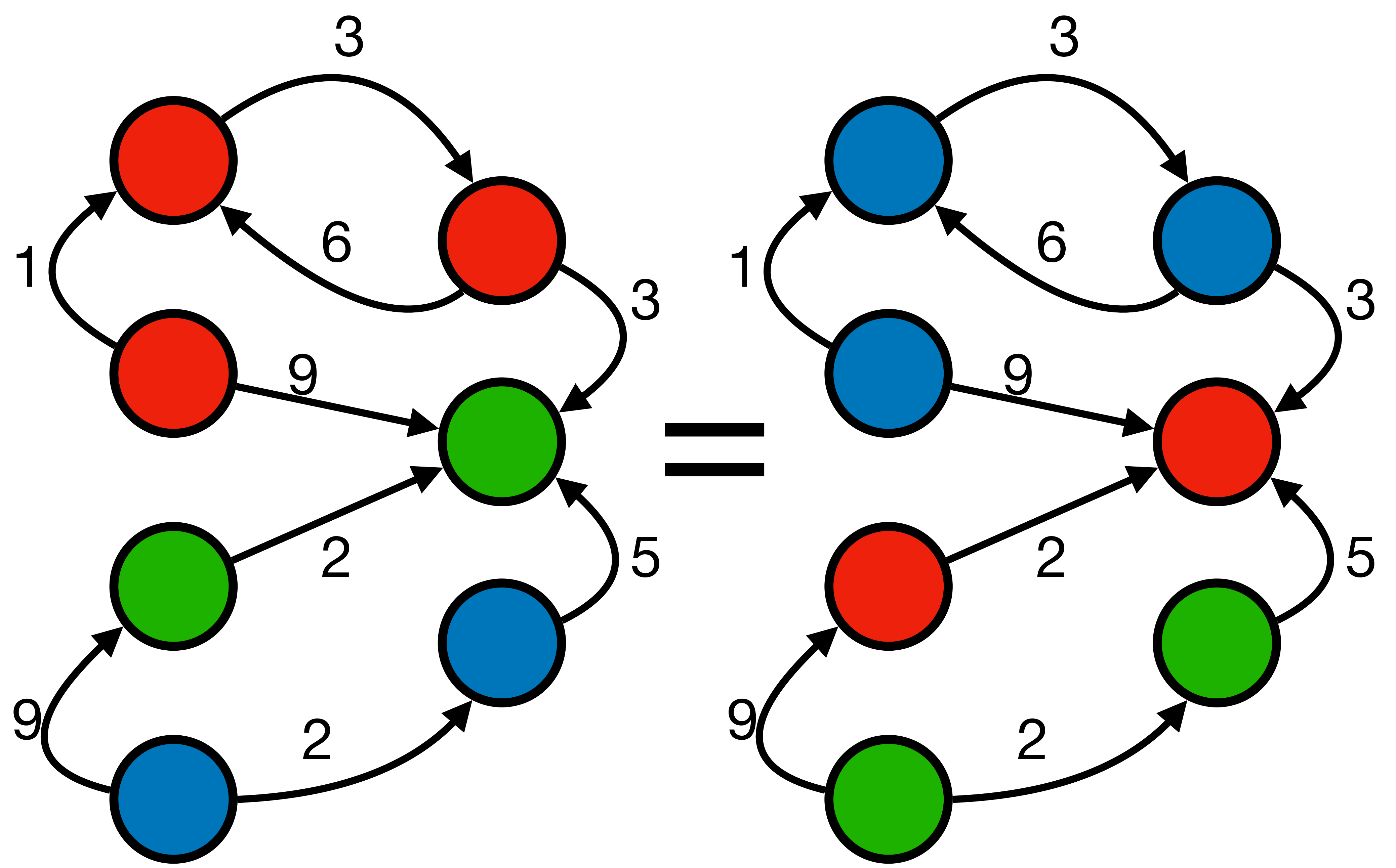
Unseeded NTAC - Problem statement

- **Input**: edge weighted directed graph
- **Output**: labels for all nodes
- **Goal**: maximize accuracy (via Hungarian alg)



Unseeded NTAC - Problem statement

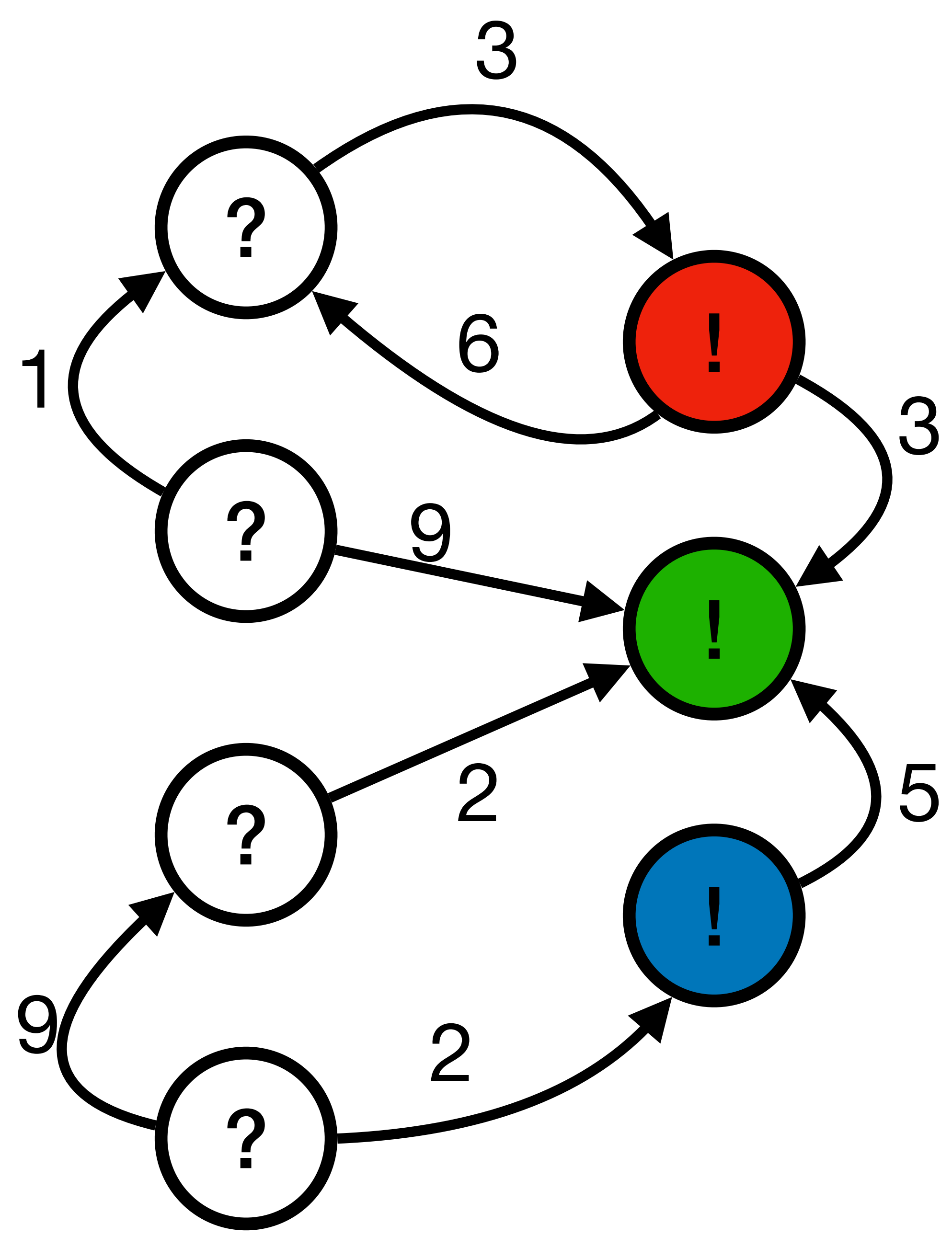
- **Input:** edge weighted directed graph
- **Output:** labels for all nodes
- **Goal:** maximize accuracy (via Hungarian alg)



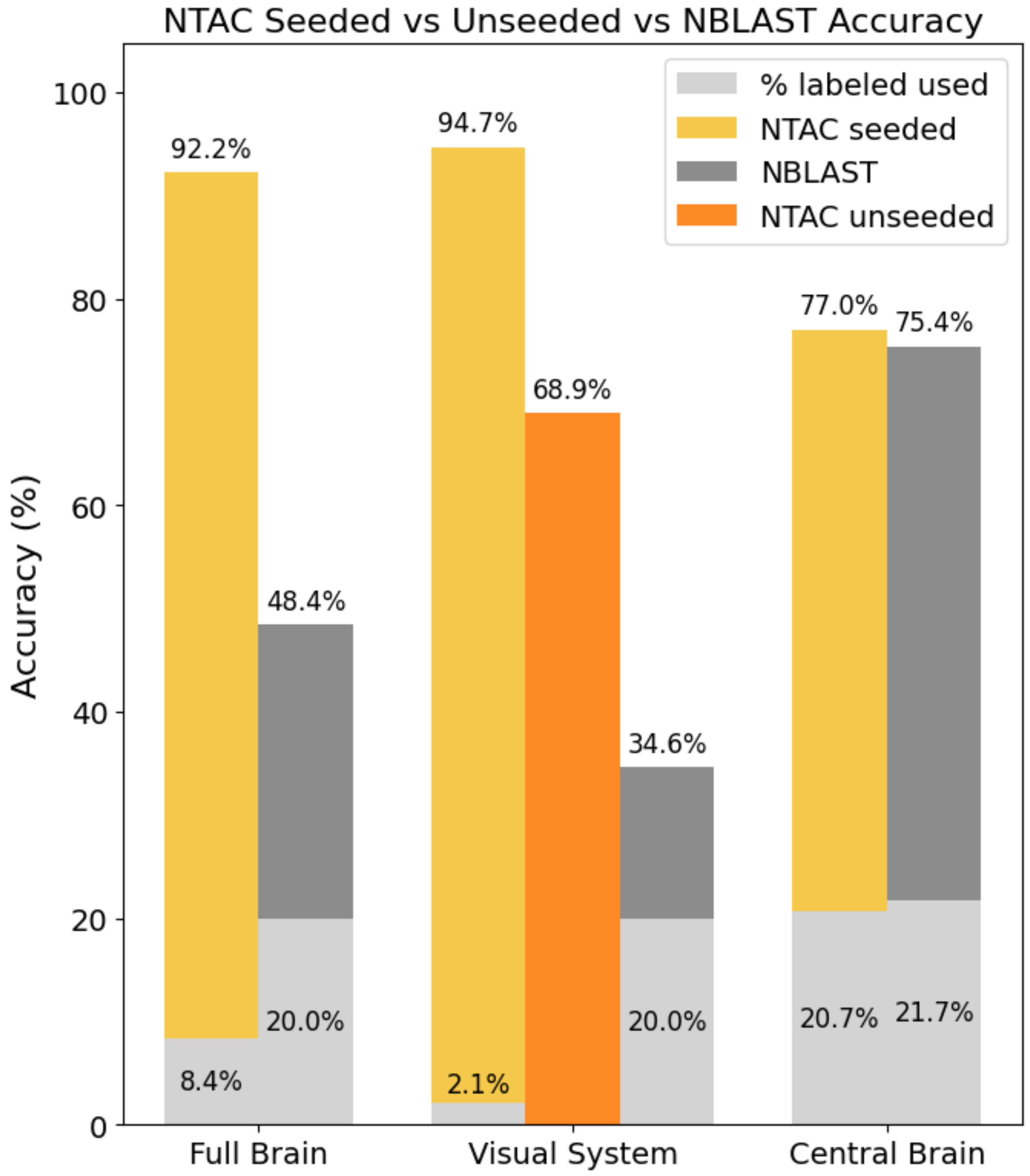
*Hungarian algorithm: computes clustering accuracy by optimally matching predicted cluster labels to true labels to maximize correct assignments

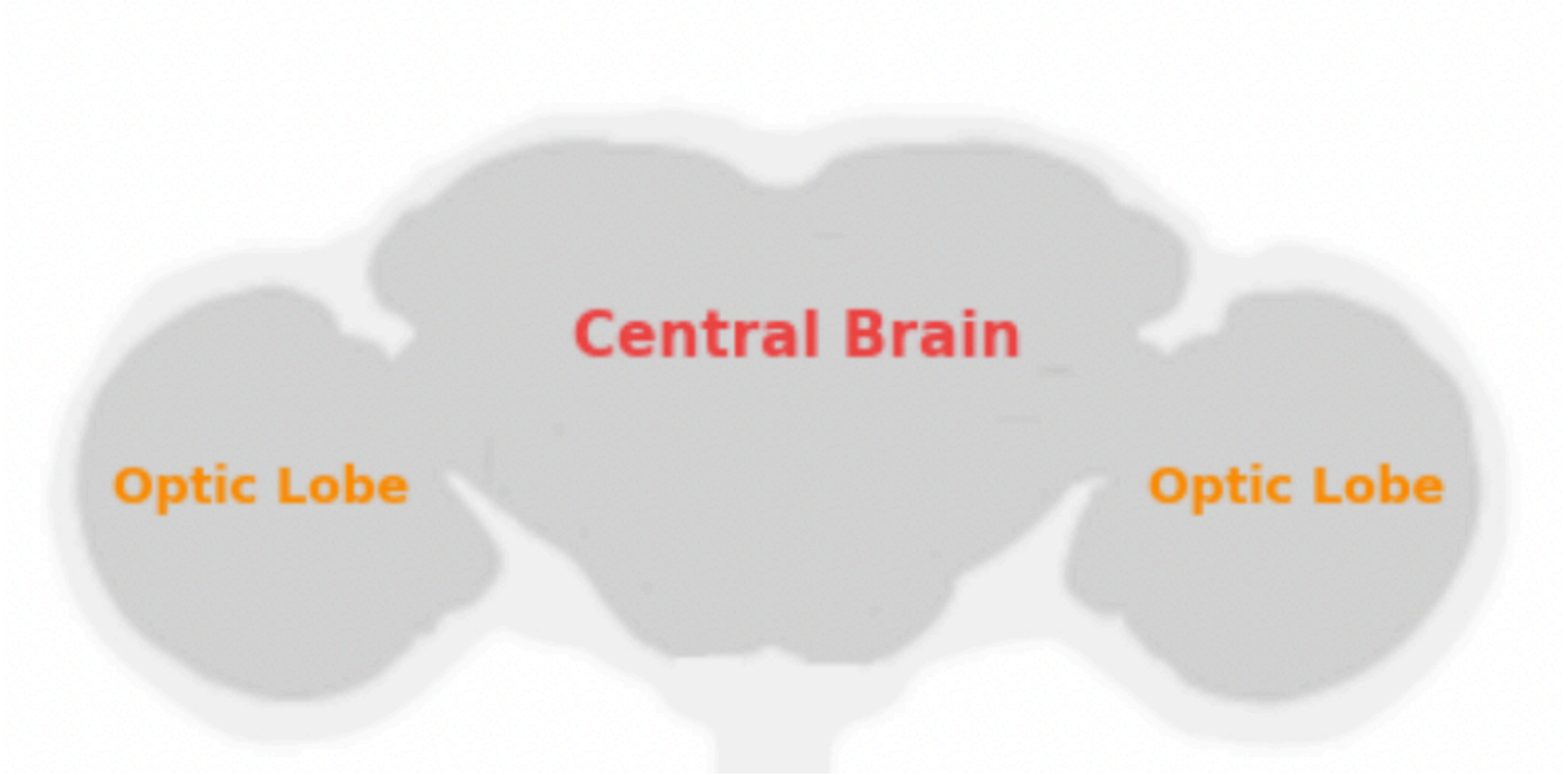
Prior research

- NBLAST (Costa et al. 2016)
  - Use morphological data to calculate neuron similarity
- Deep learning approaches (Troidel et al. 2025, Liao et al. 2024, Chen et al. 2022, Jiang et al. 2023)
  - Use morphological data
  - Small datasets \ exclude rare types \ consider meta-types
- **NTAC**: iterative classification without node features



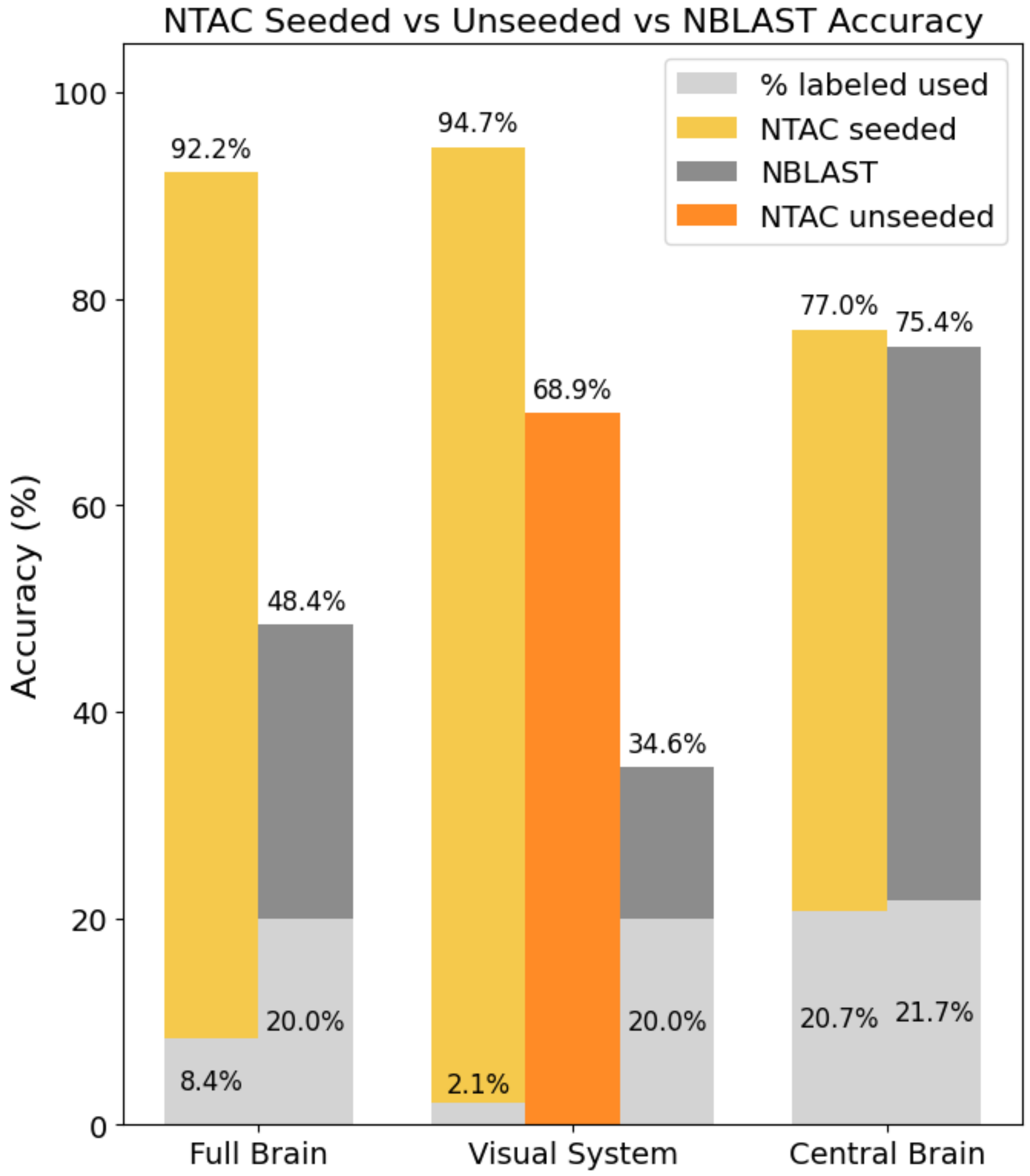
Accuracy comparison (Flywire)

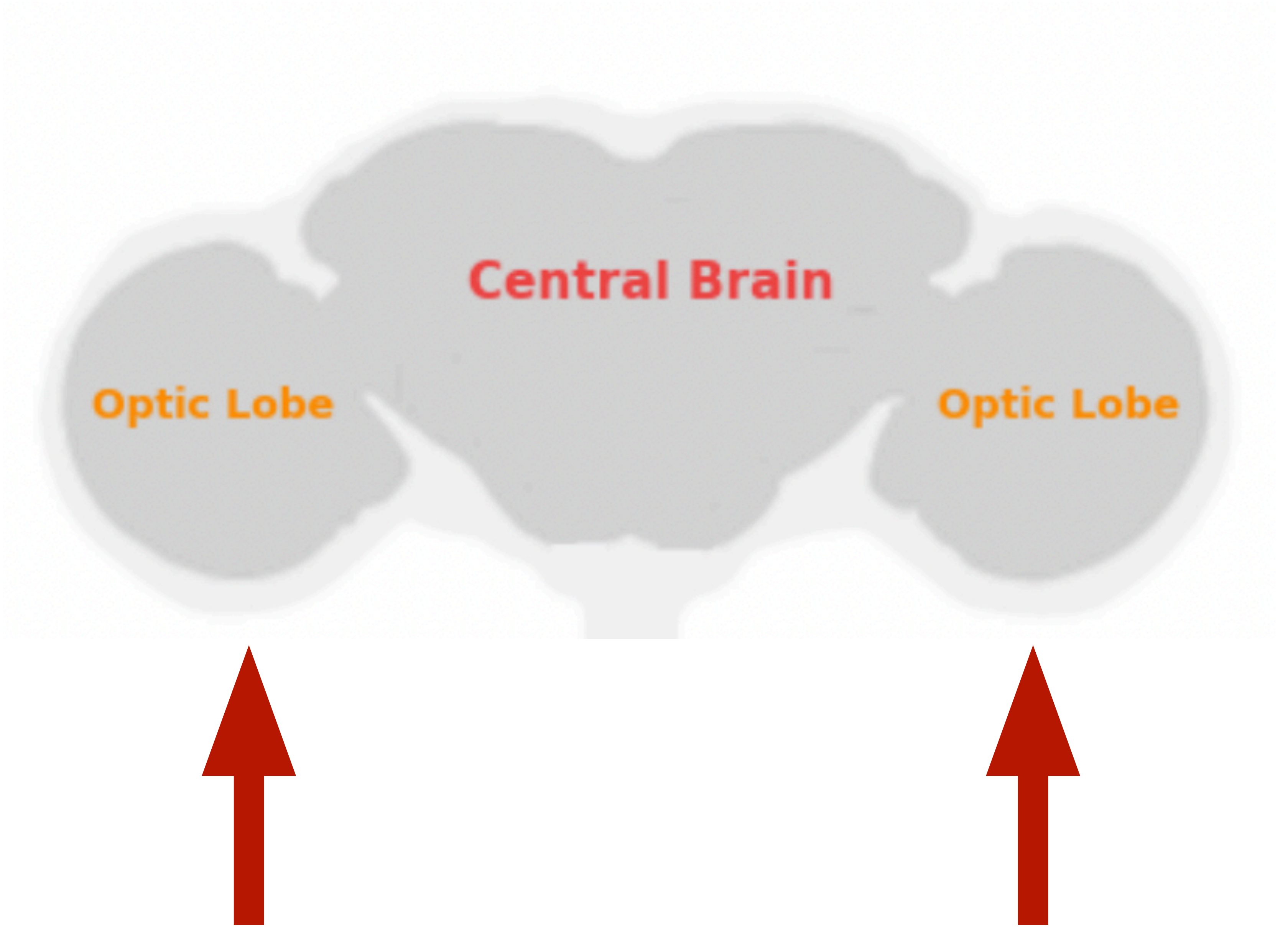




Many types, few neurons per type

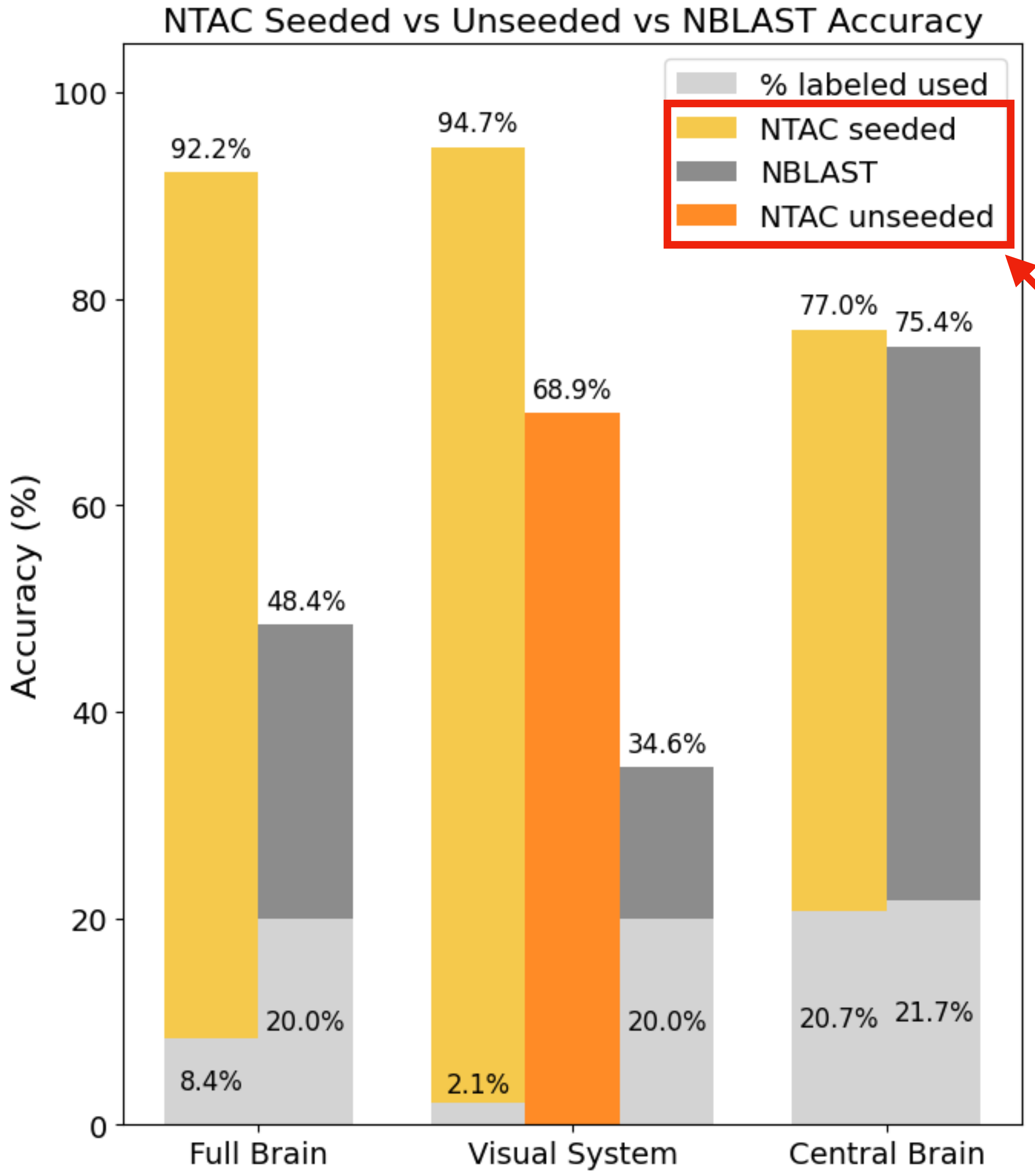
Accuracy comparison



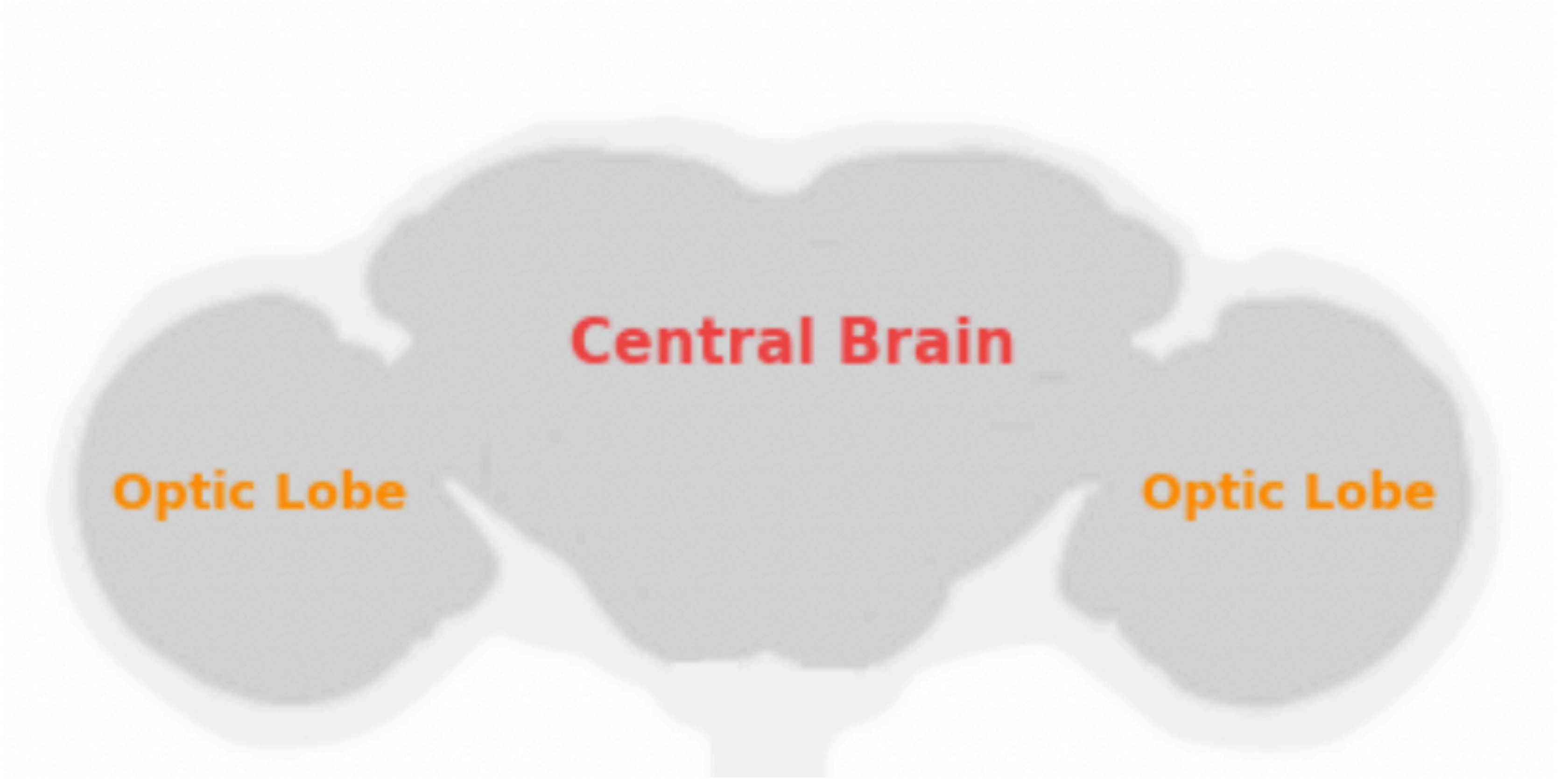


Fewer types, many neurons per type

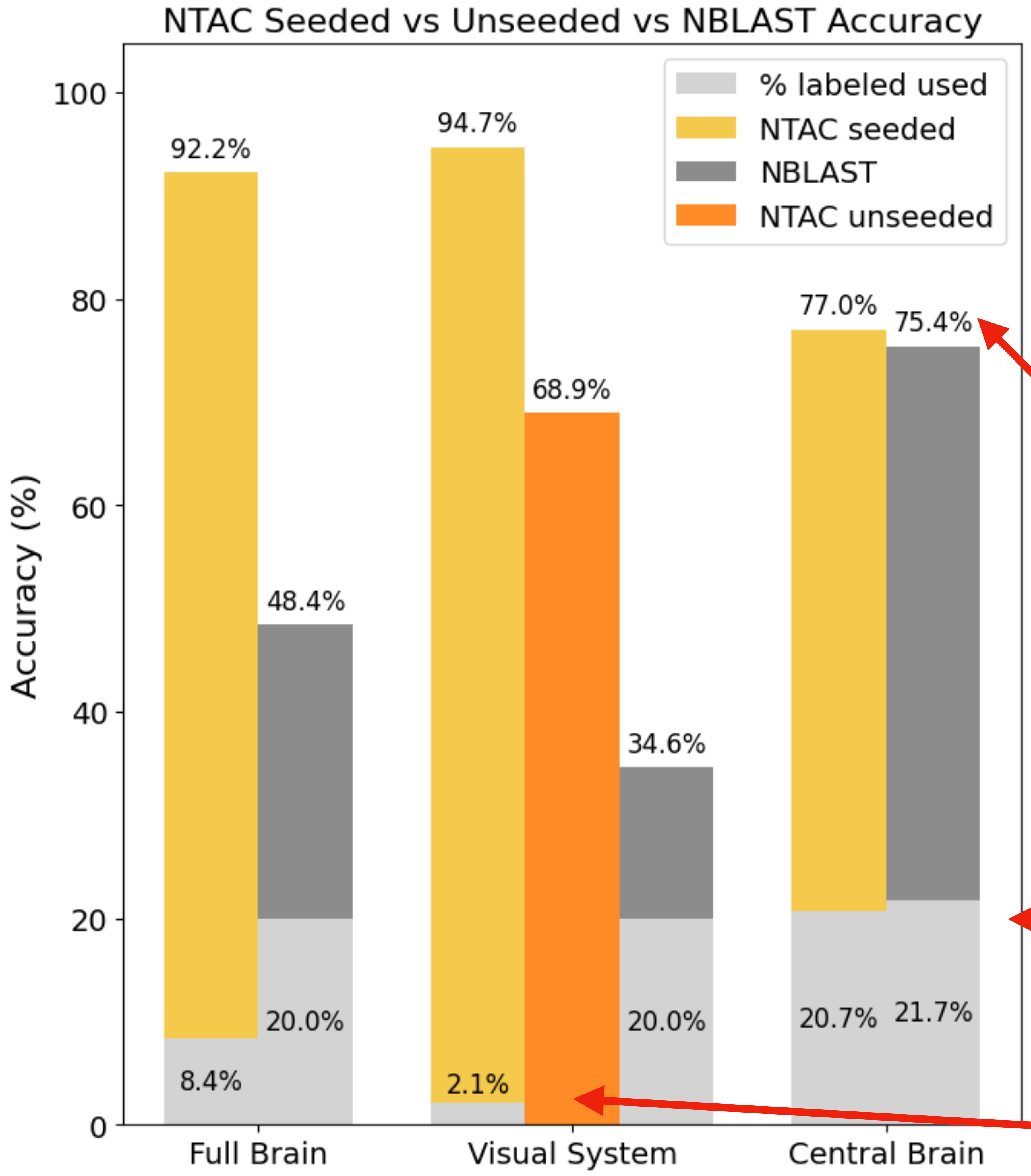
Accuracy comparison

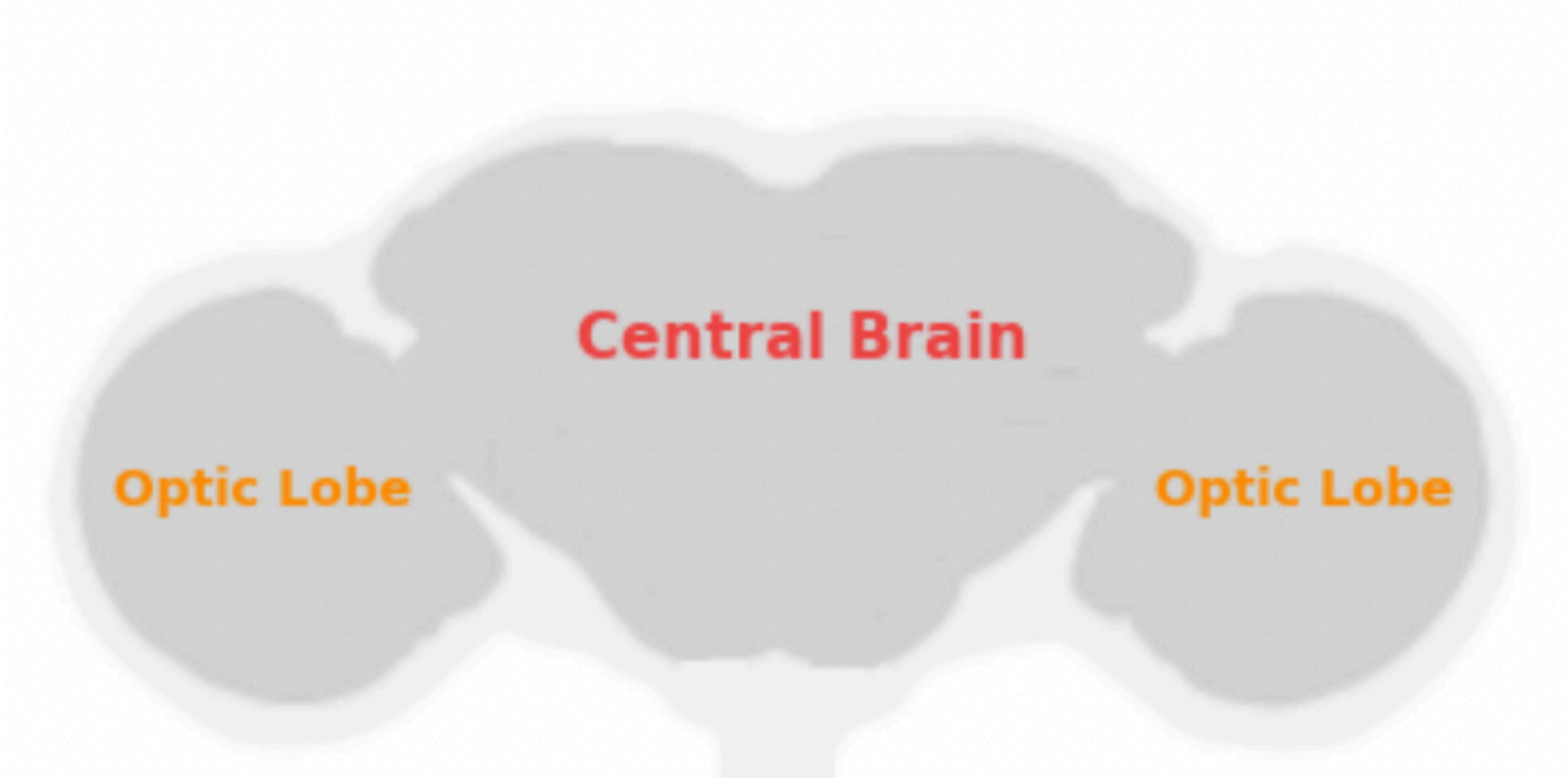


NBLAST + knn classifier



Accuracy comparison



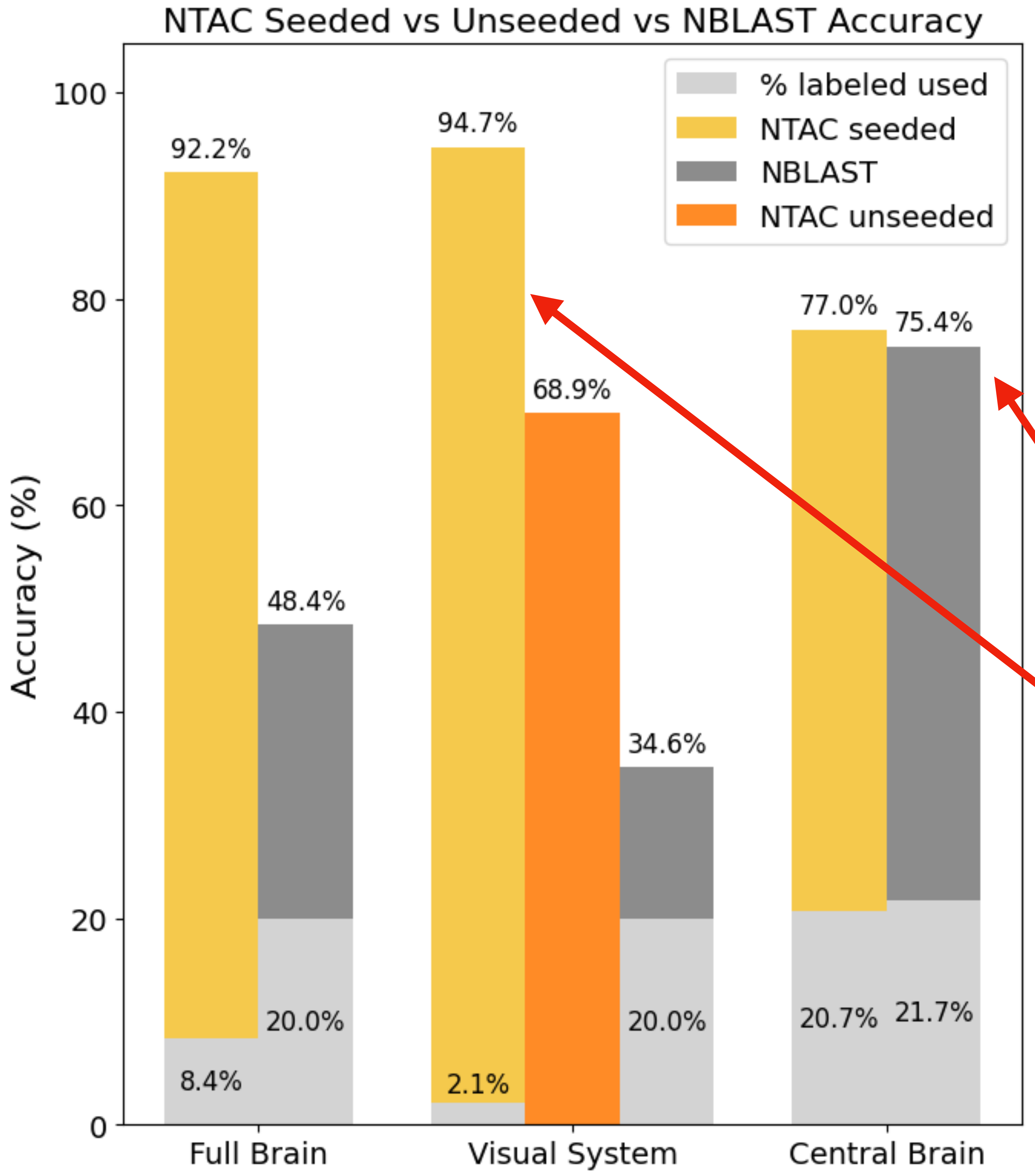


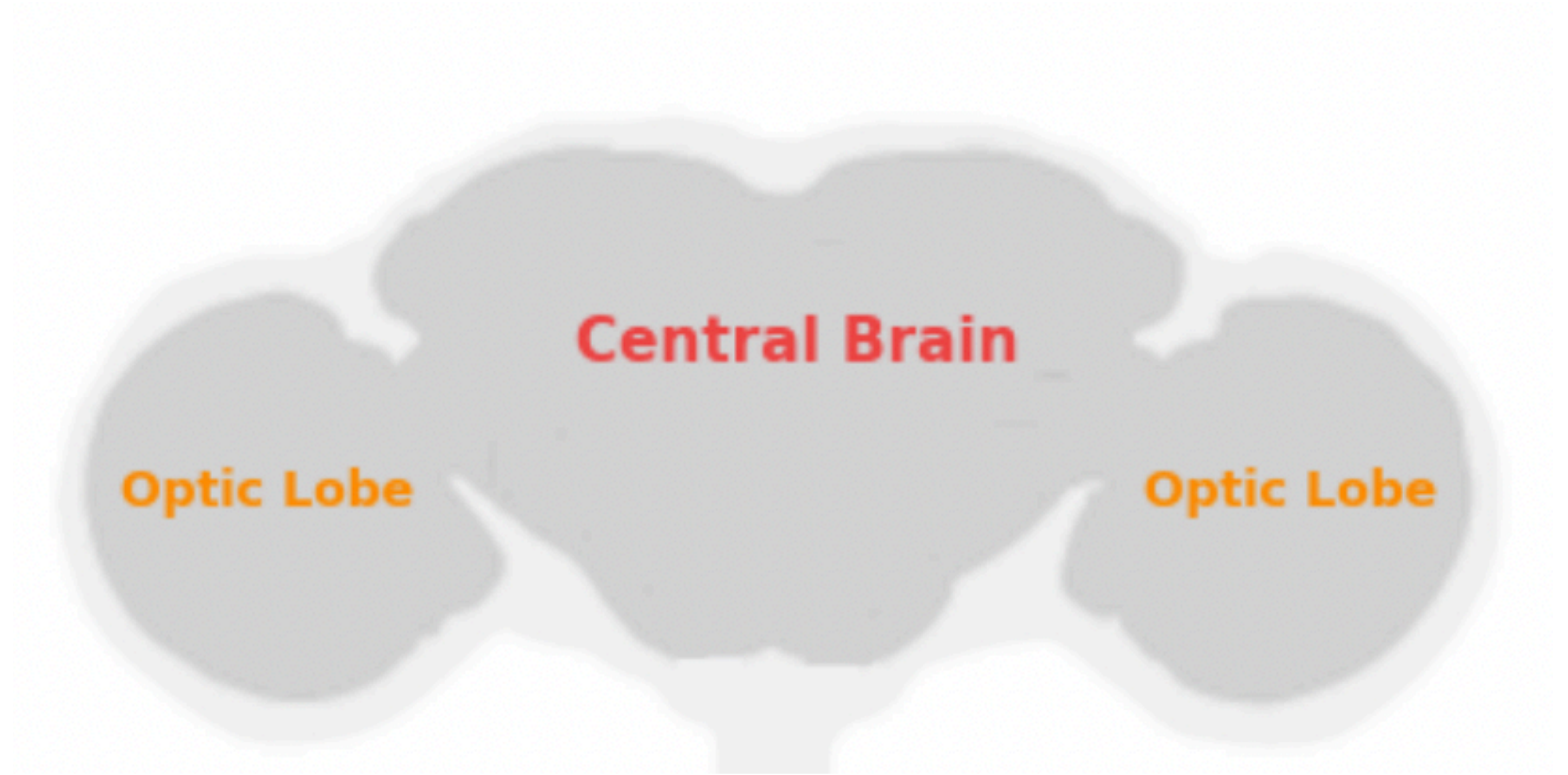
Accuracy

% labeled used
We assume at least one labelled node from each type is given

No labels required

Accuracy comparison

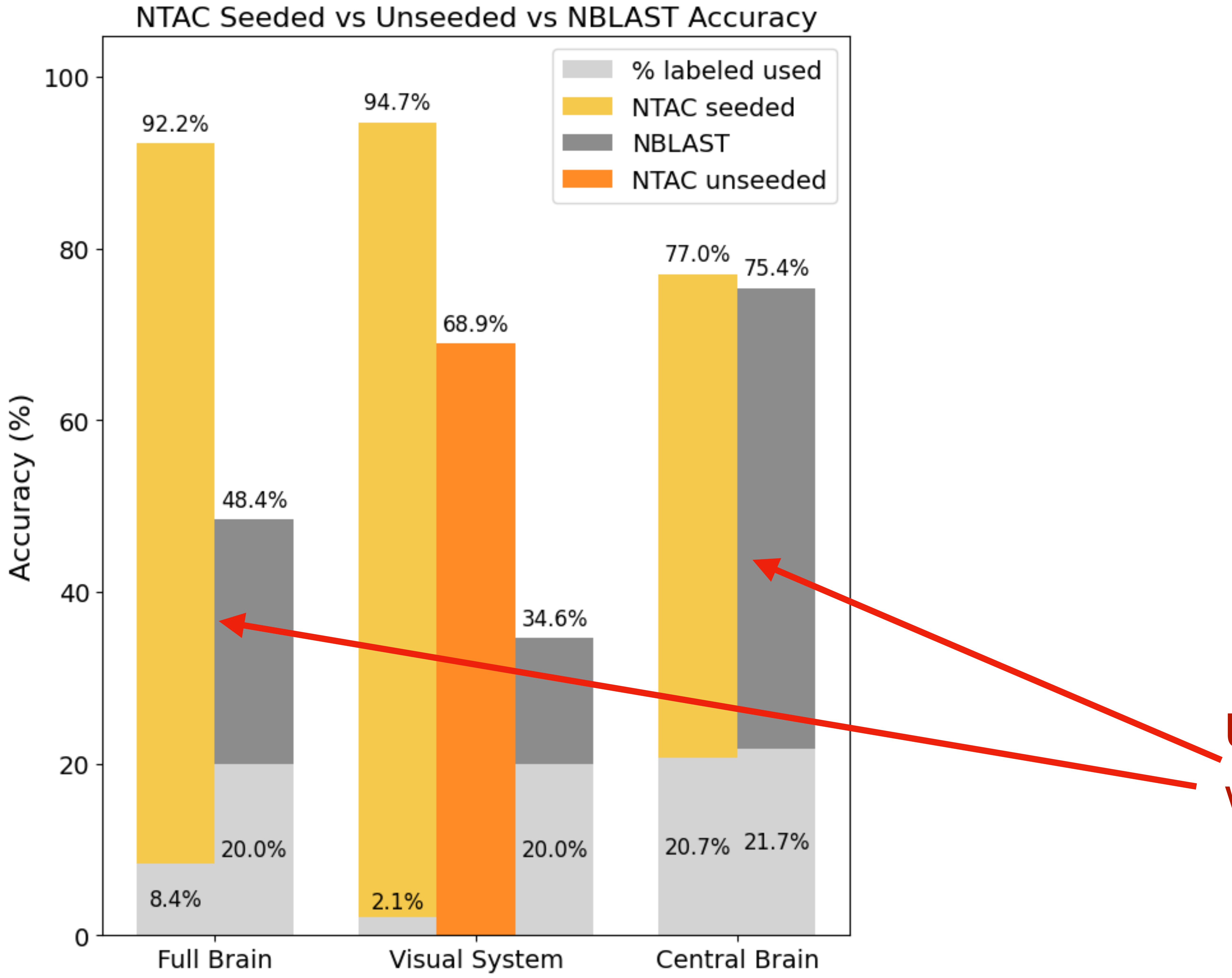


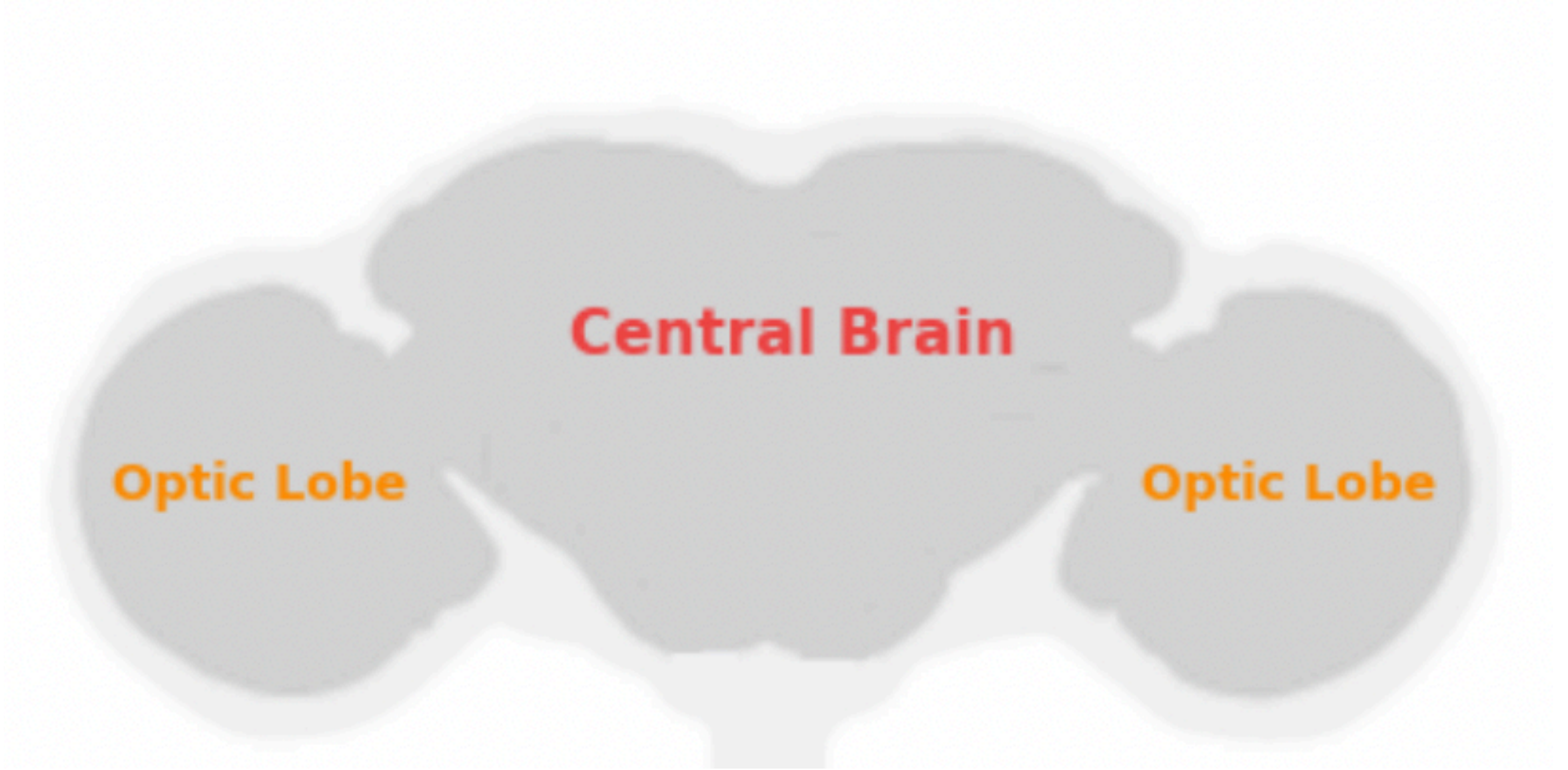


About the same on the central brain
NTAC is much faster (minutes vs hours)

NTAC is much better on visual system

Accuracy comparison

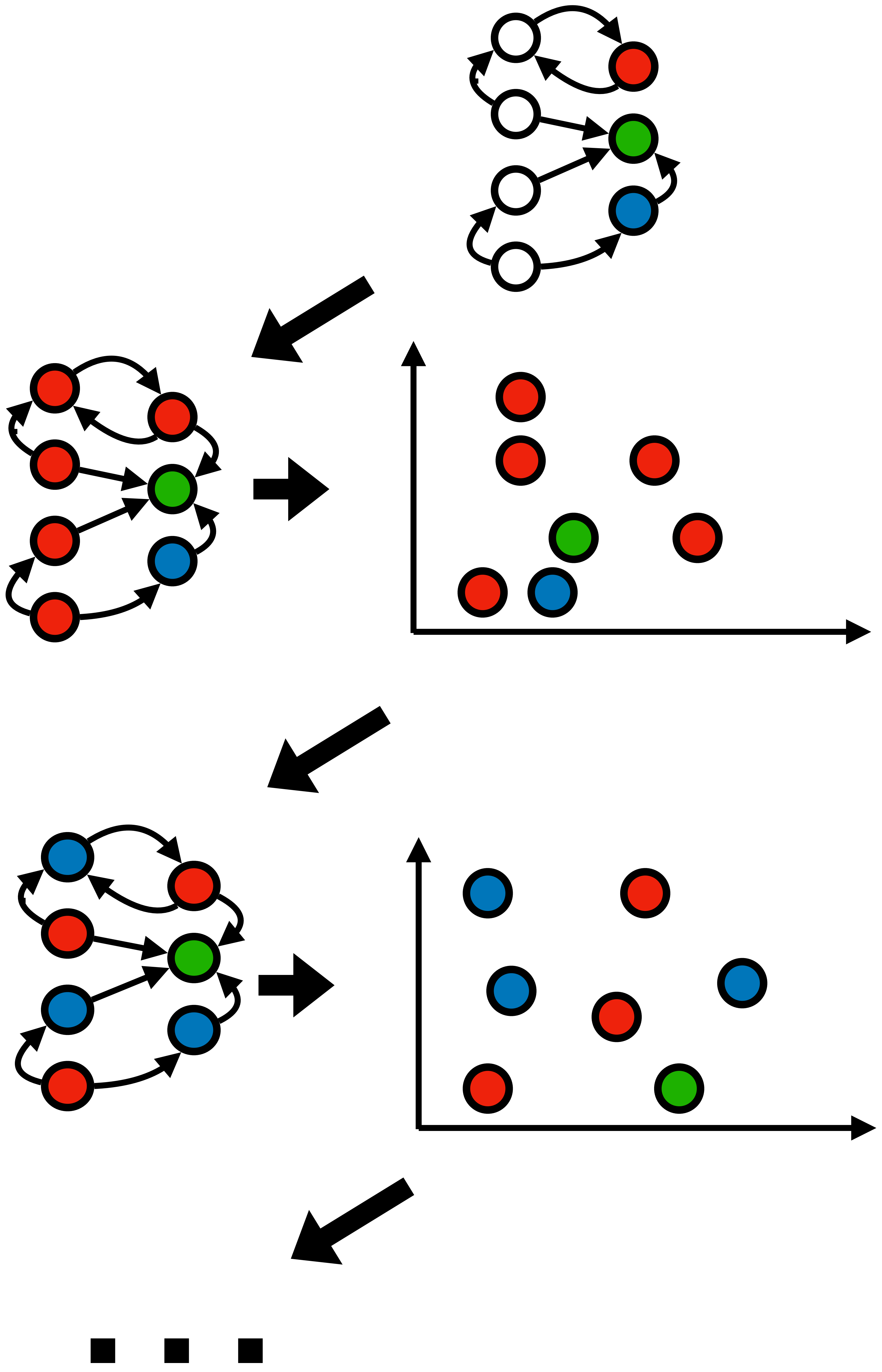




Unseeded NTAC is very slow when there are many types

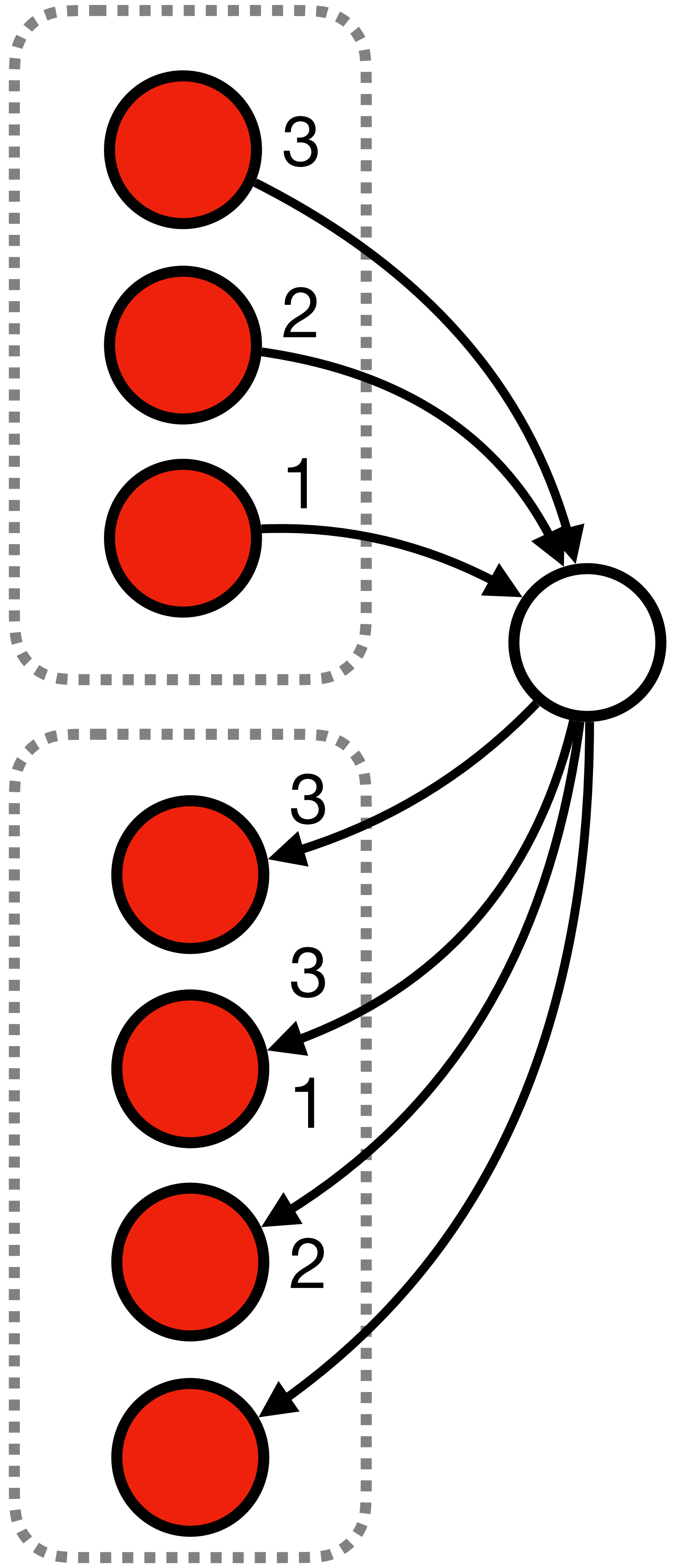
Seeded algorithm outline

- Compute initial node partition
- Repeat t times
  - Compute node **embedding** using node **partition**
  - Compute node **partition** using node **embedding**
- Output **partition**

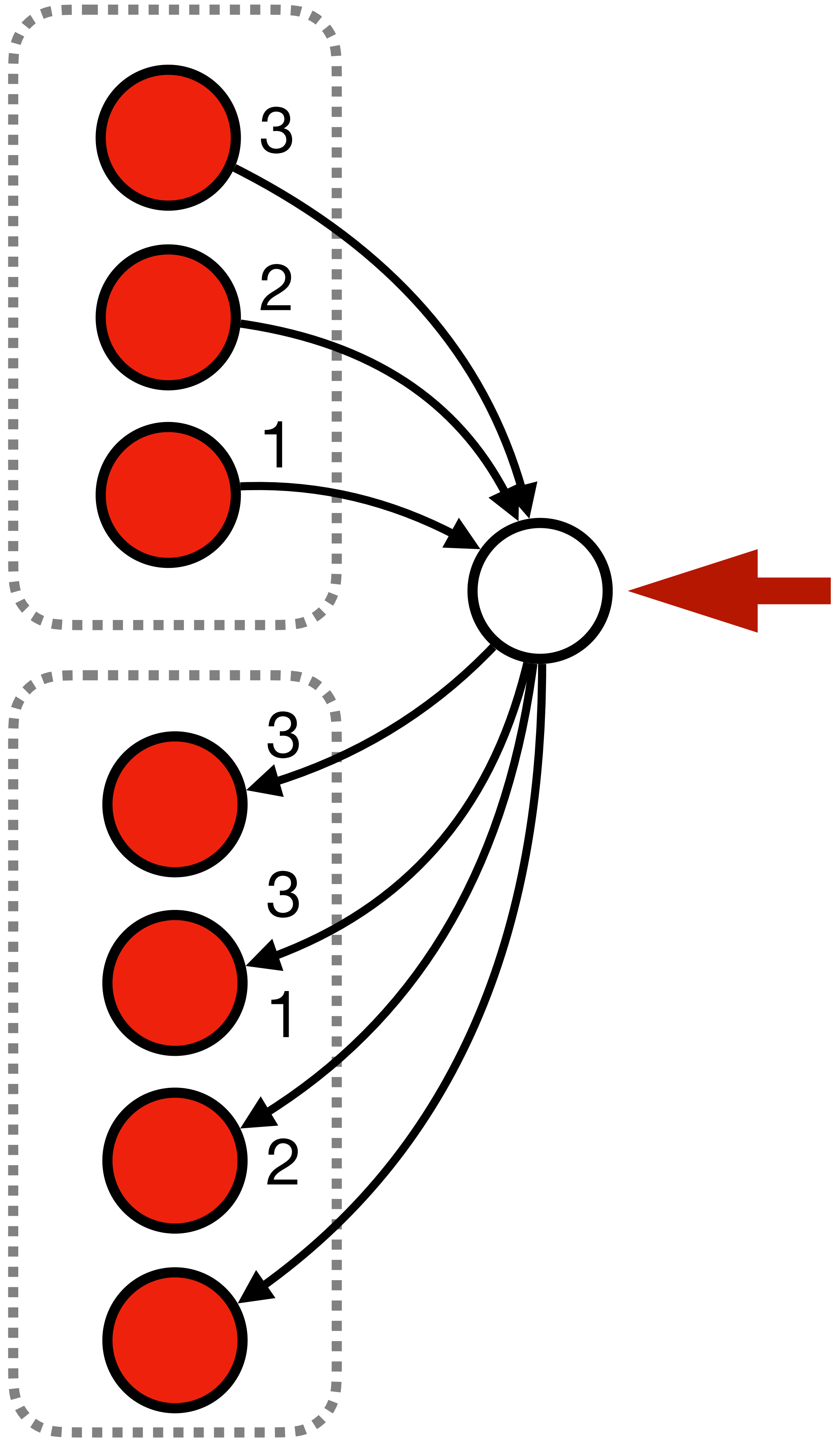


Partition → embedding

Focus on a single color for now



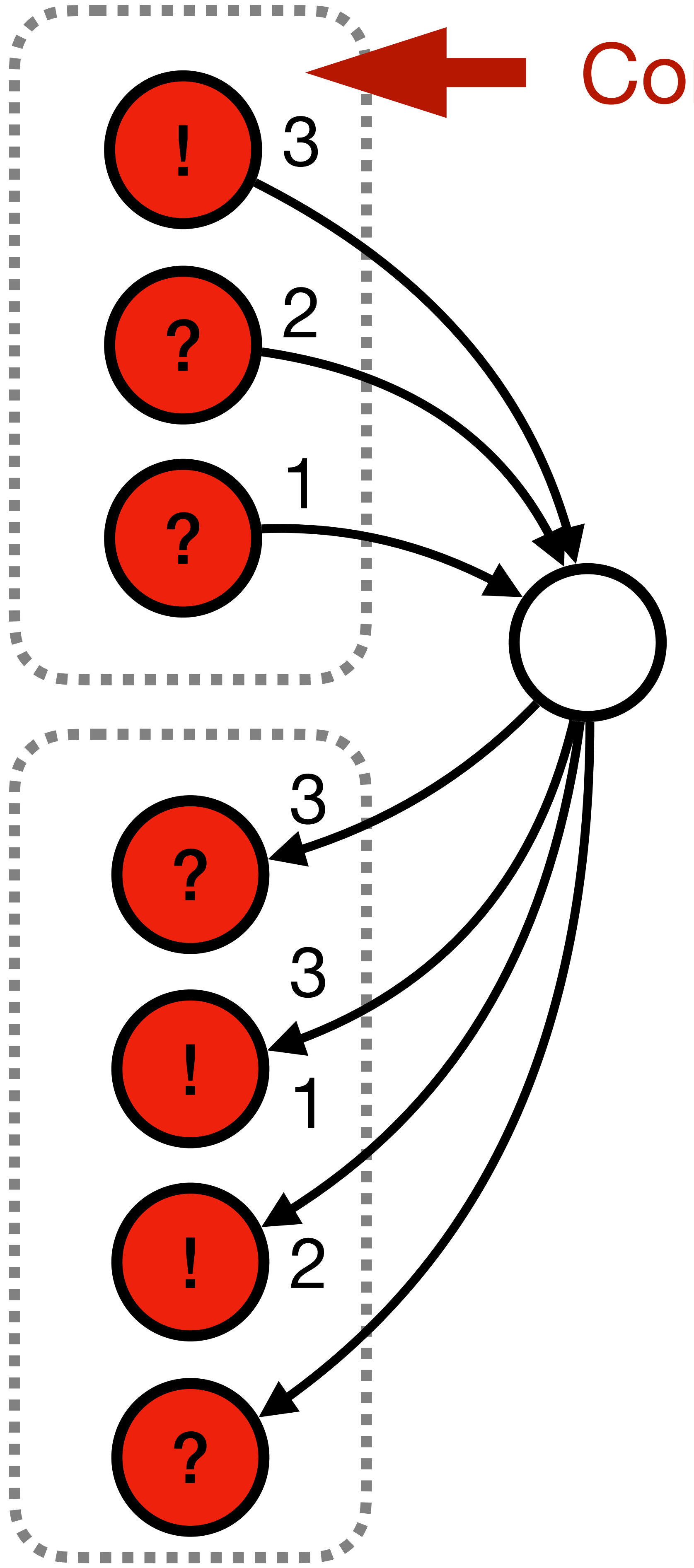
Partition → embedding



Node's color doesn't matter

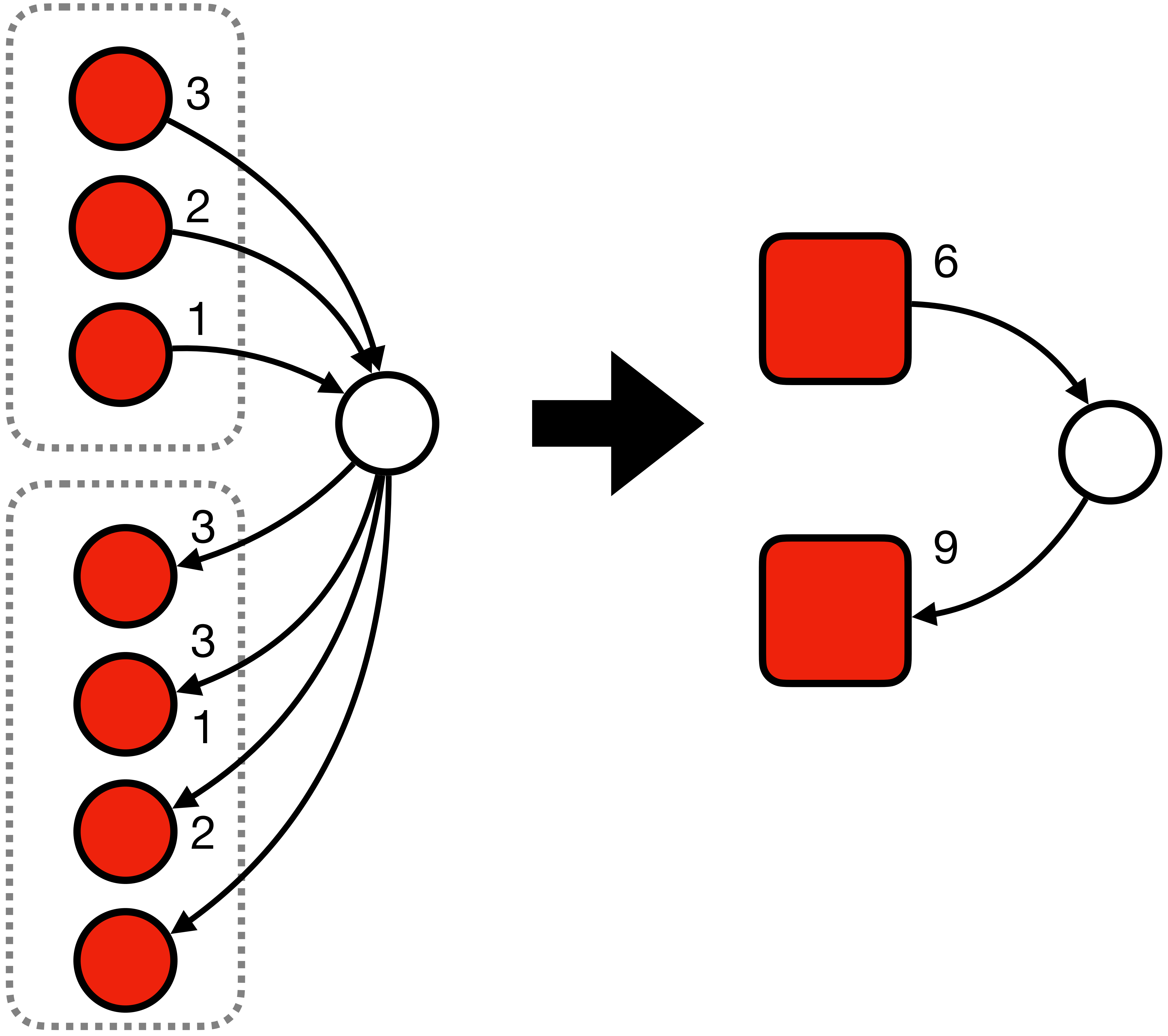
(Ignoring self loops for simplicity)

Partition → embedding

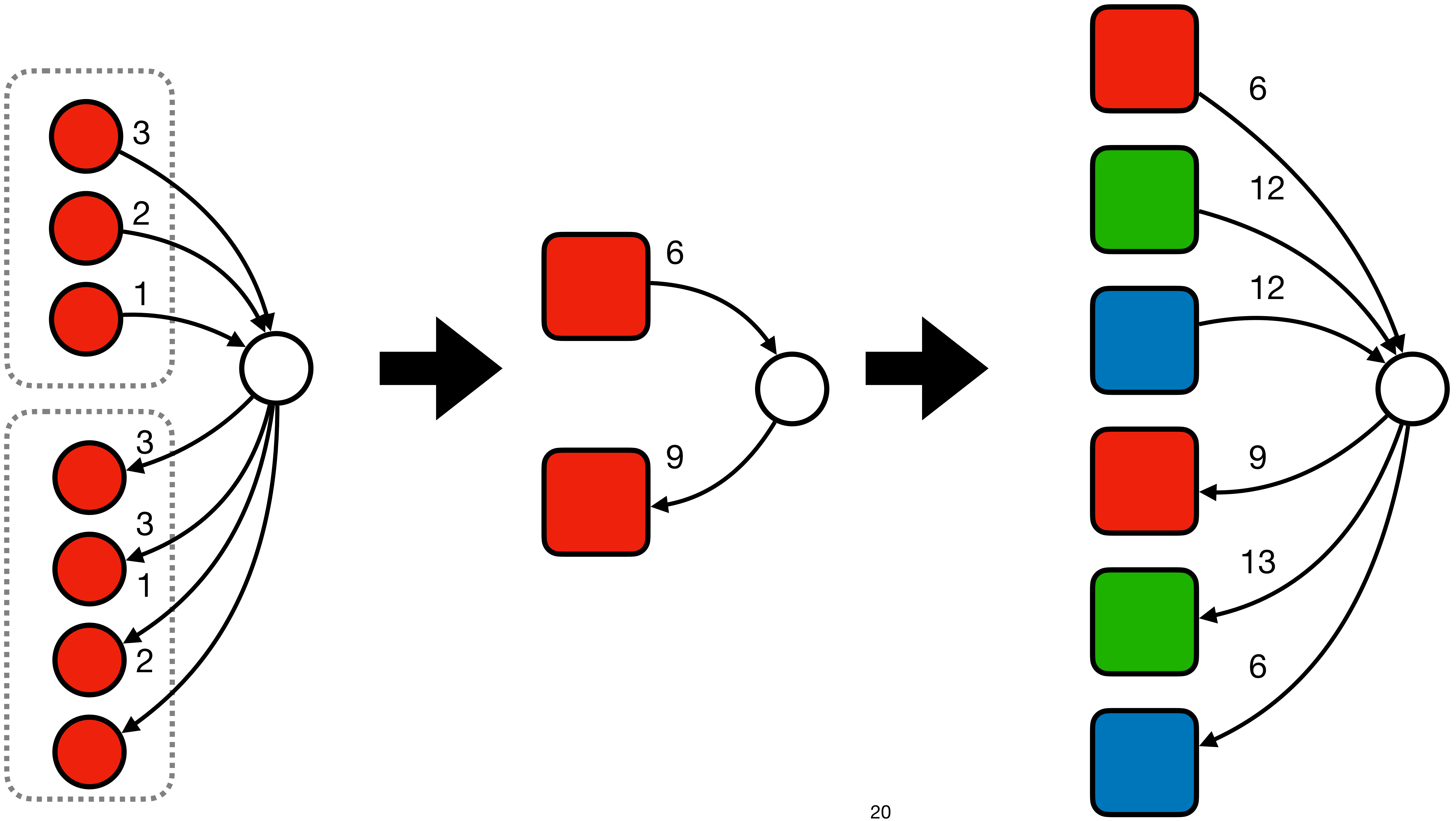


Consider both labelled and unlabeled neighbors

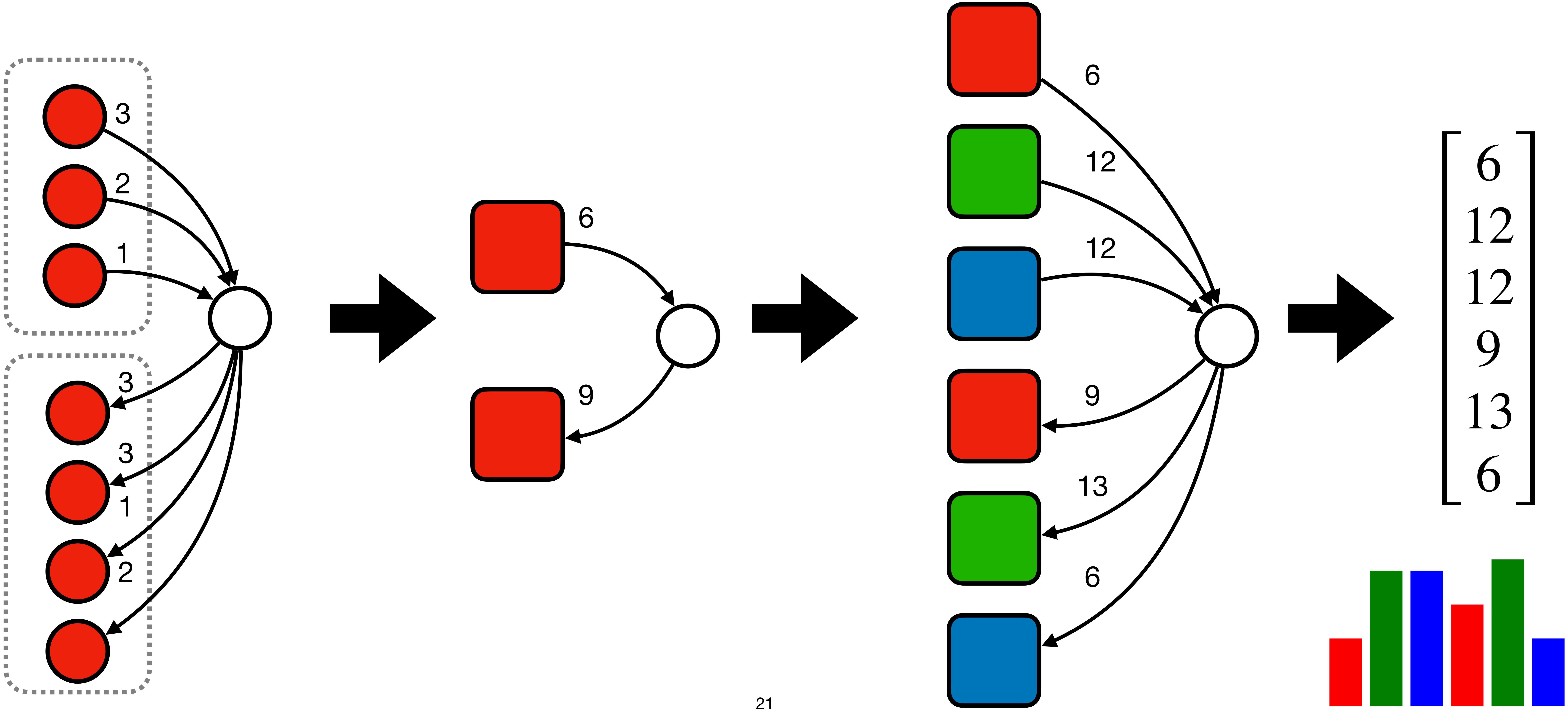
Partition → embedding



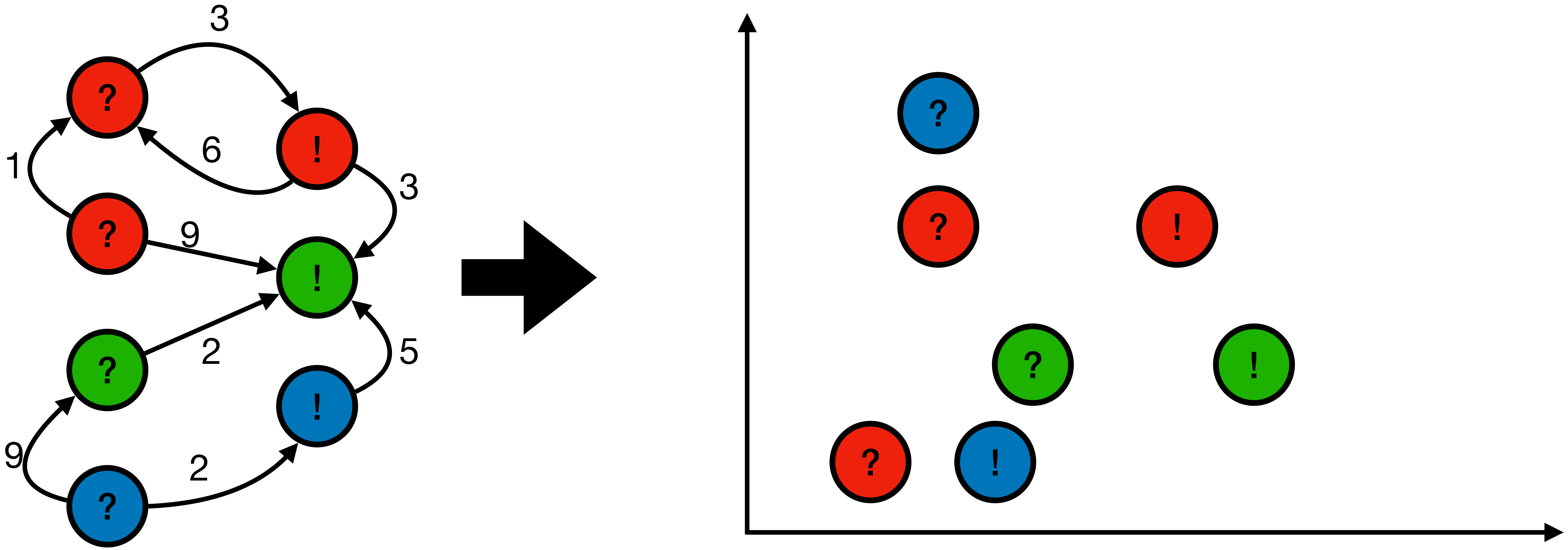
Partition → embedding



Partition → embedding


3
2
1
3
3
1
2
6
9
6
12
12
9
13
6
$\begin{bmatrix} 6 \\ 12 \\ 12 \\ 9 \\ 13 \\ 6 \end{bmatrix}$

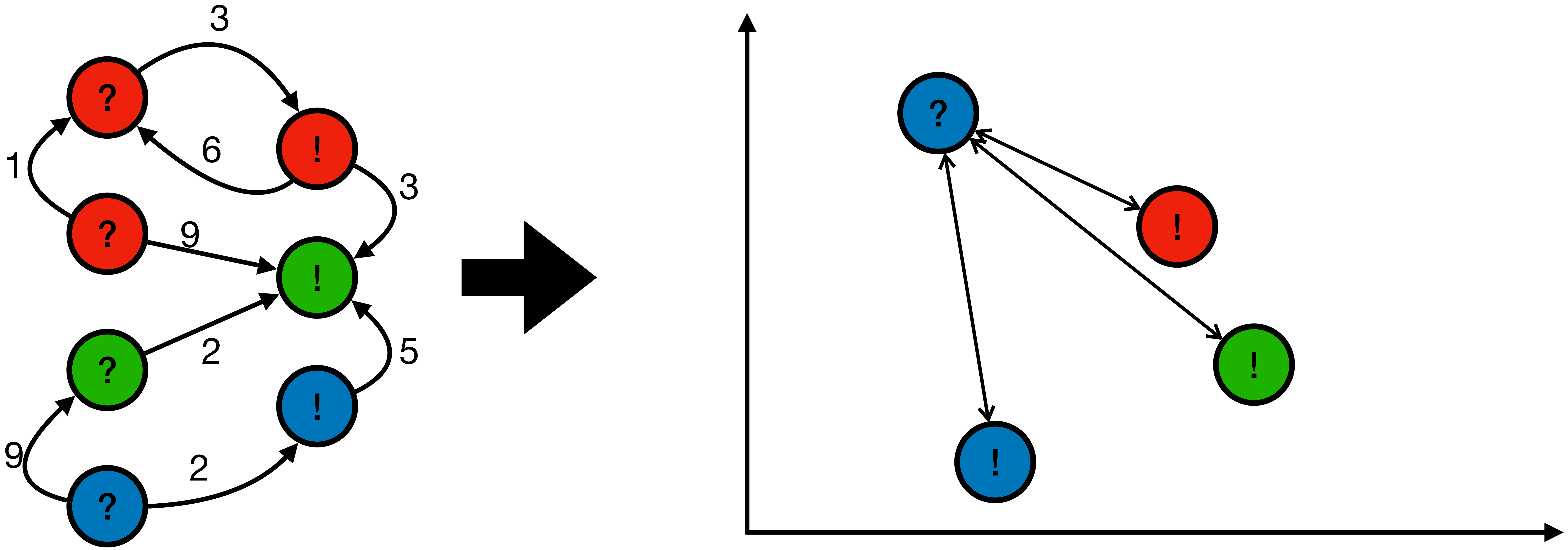
Embedding → partition



Embedding → partition

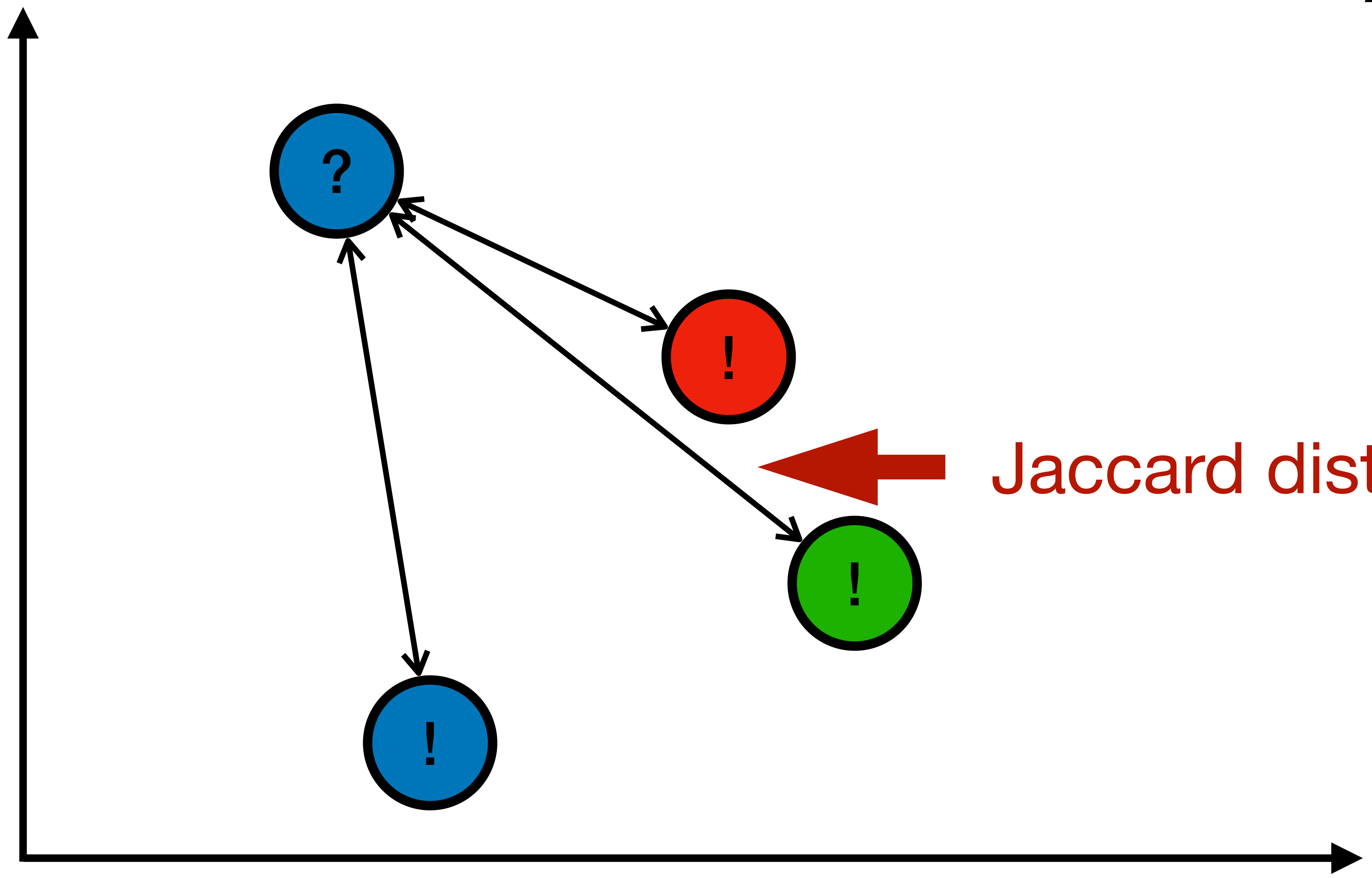
Assign color according to closest **labeled** node

(or weighted k-neighbor majority)



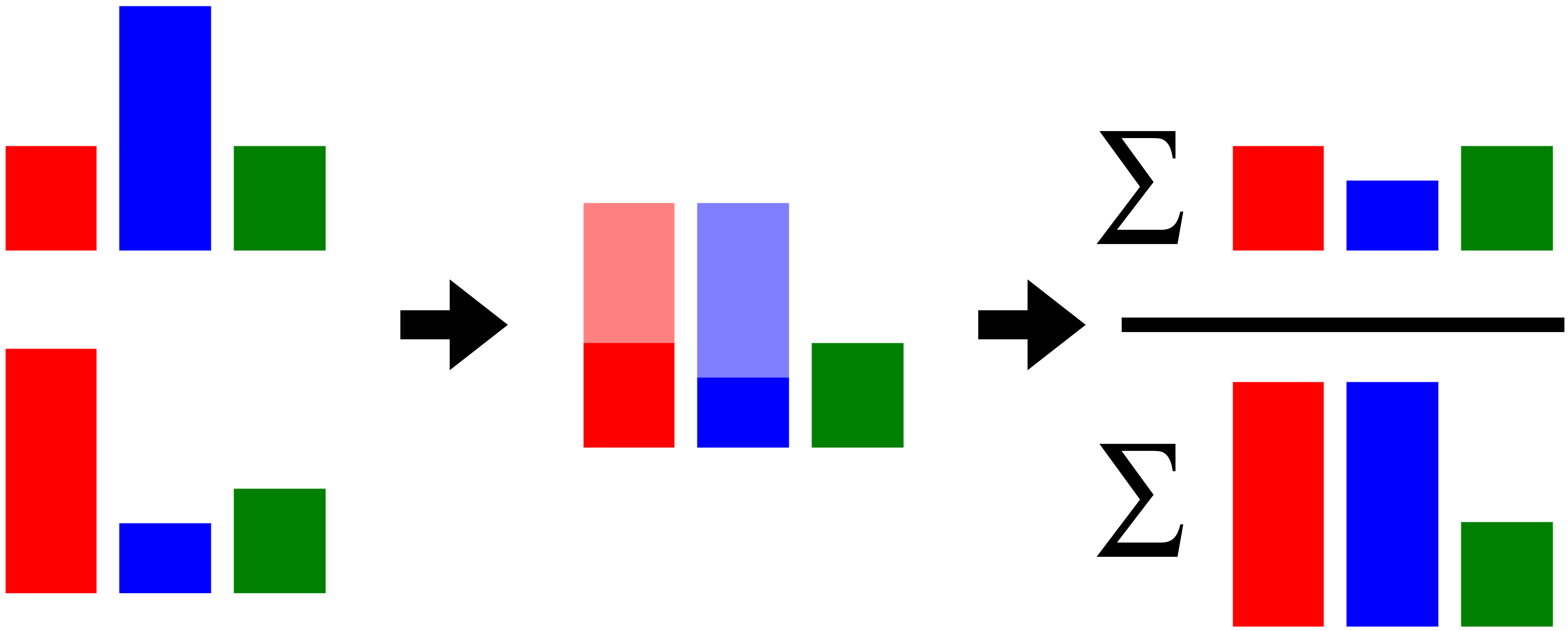
Embedding → partition

$$D_J(\vec{v}, \vec{u}) = 1 - \underbrace{\frac{\sum_i \min(v_i, u_i)}{\sum_i \max(v_i, u_i)}}_{\text{Jaccard similarity}}$$

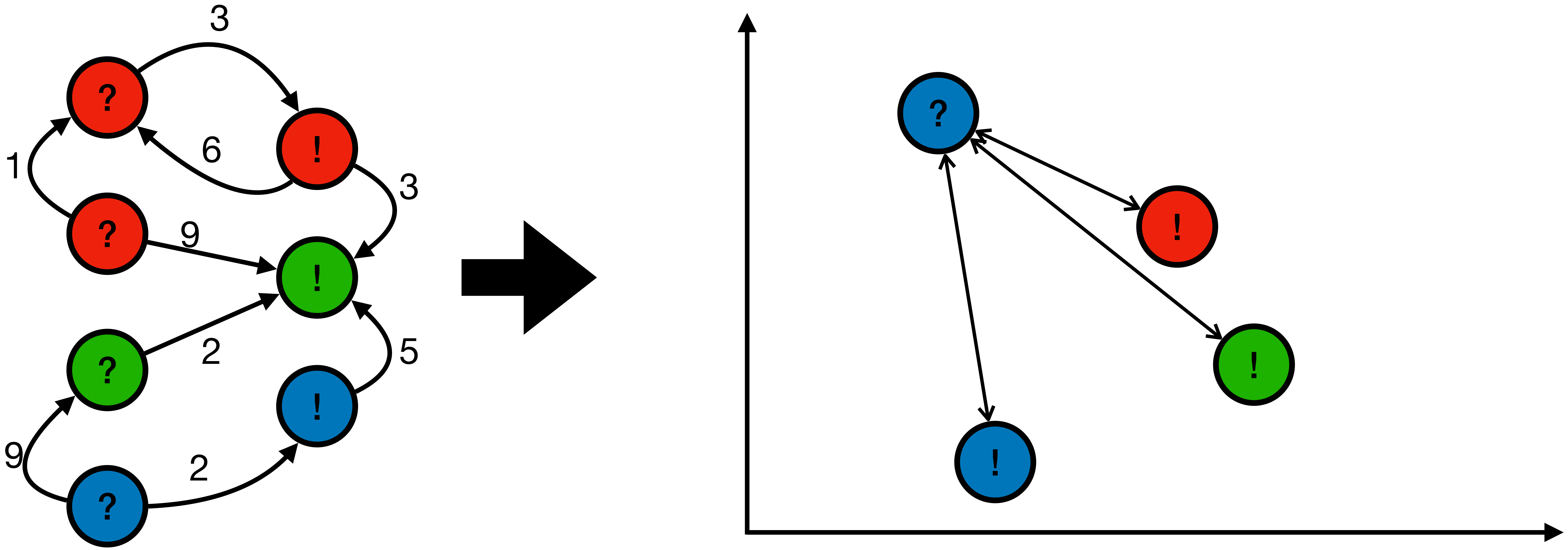


Jaccard distance

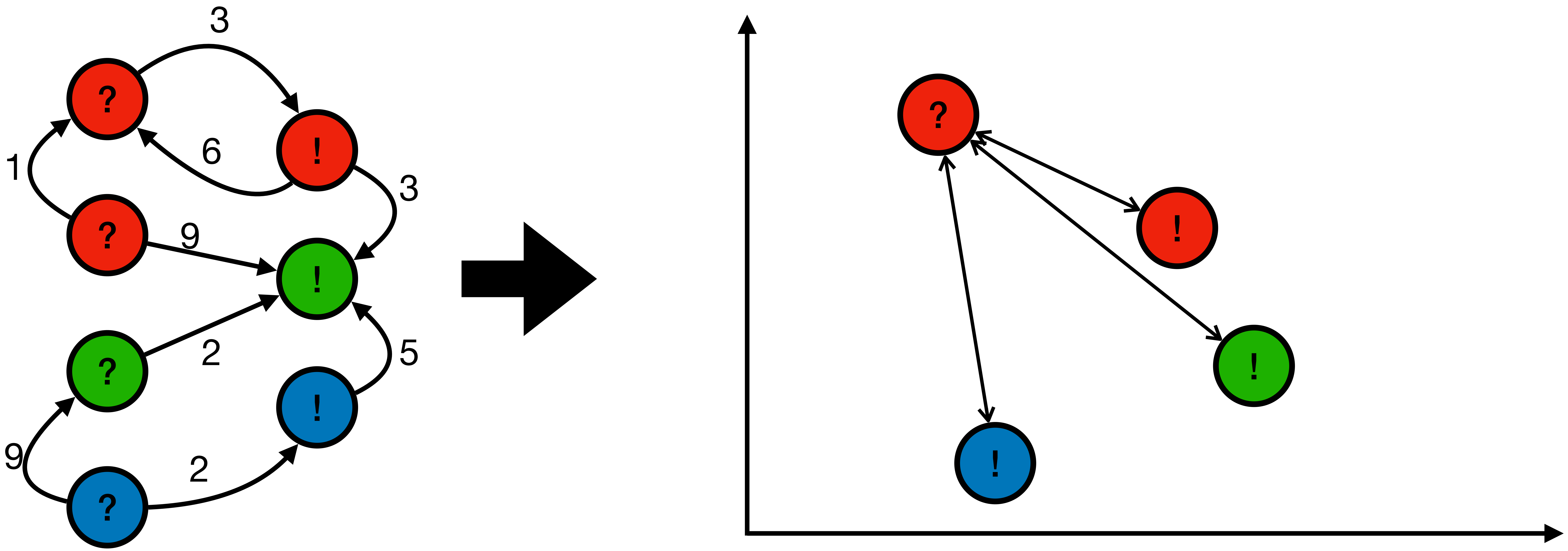
Jaccard similarity



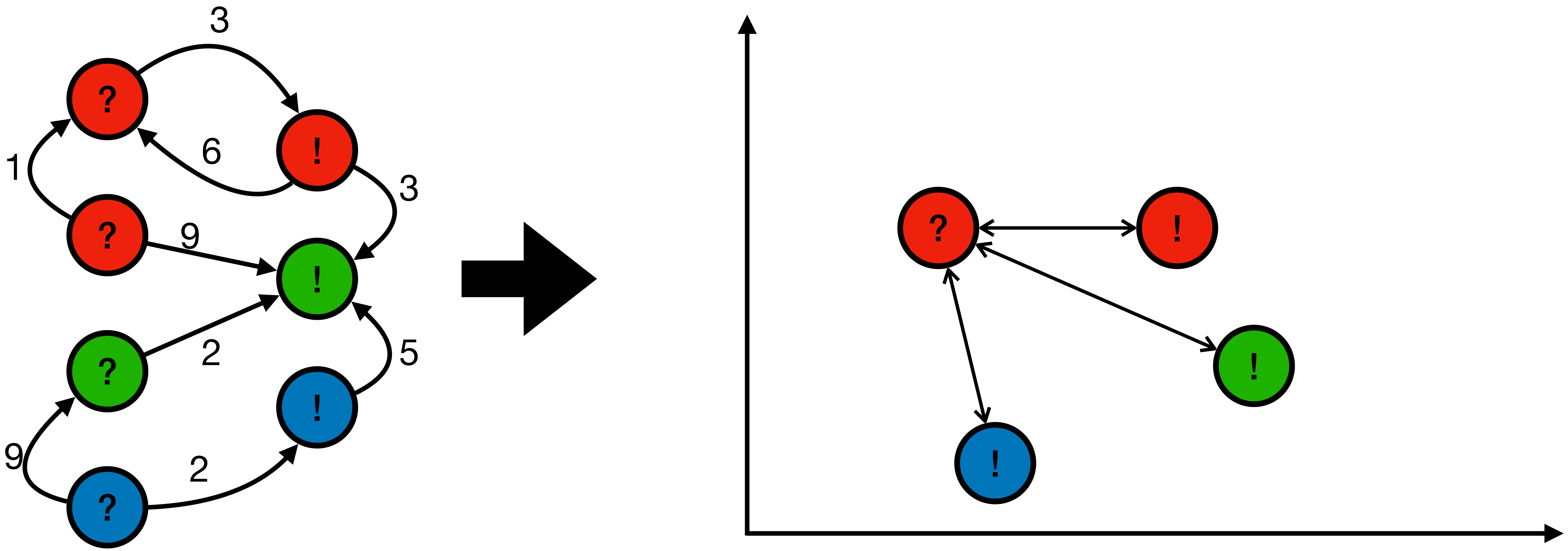
Embedding → partition



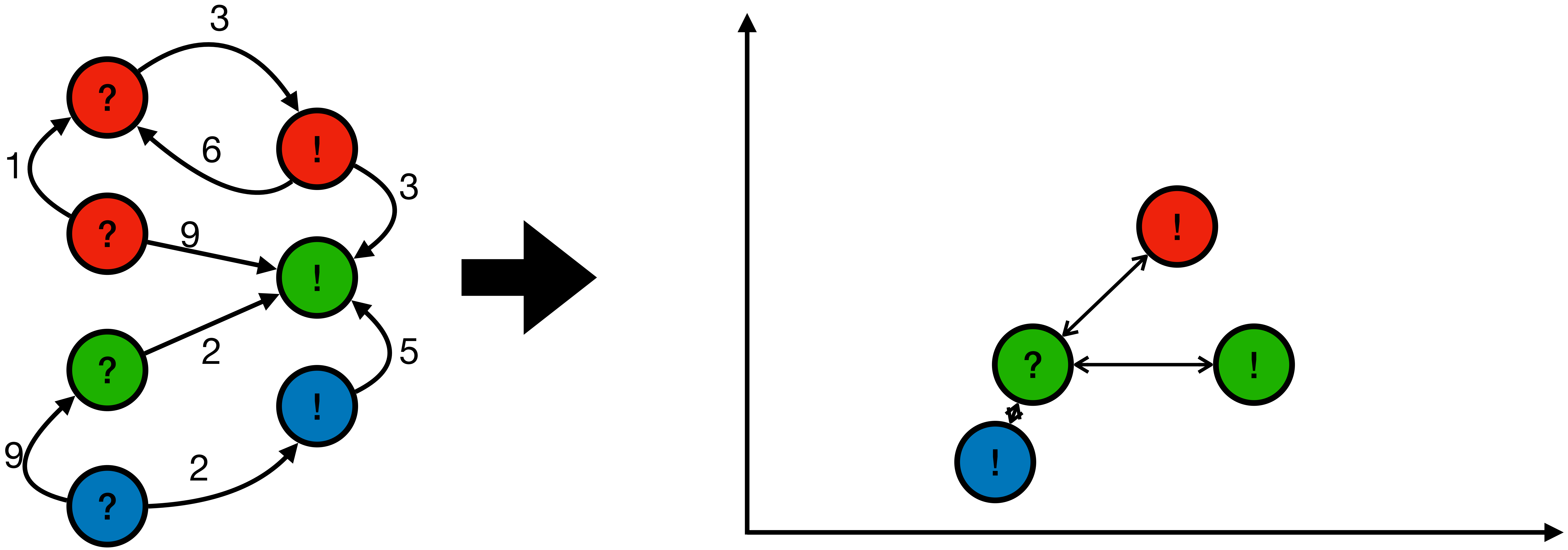
Embedding → partition



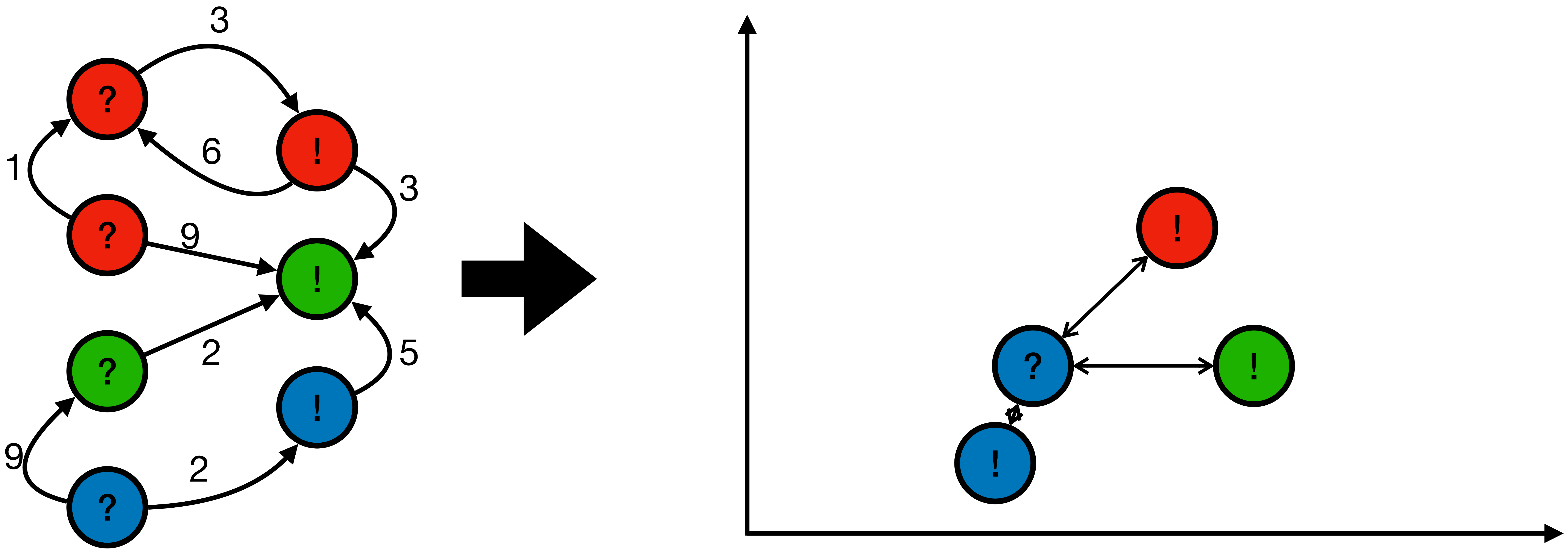
Embedding → partition



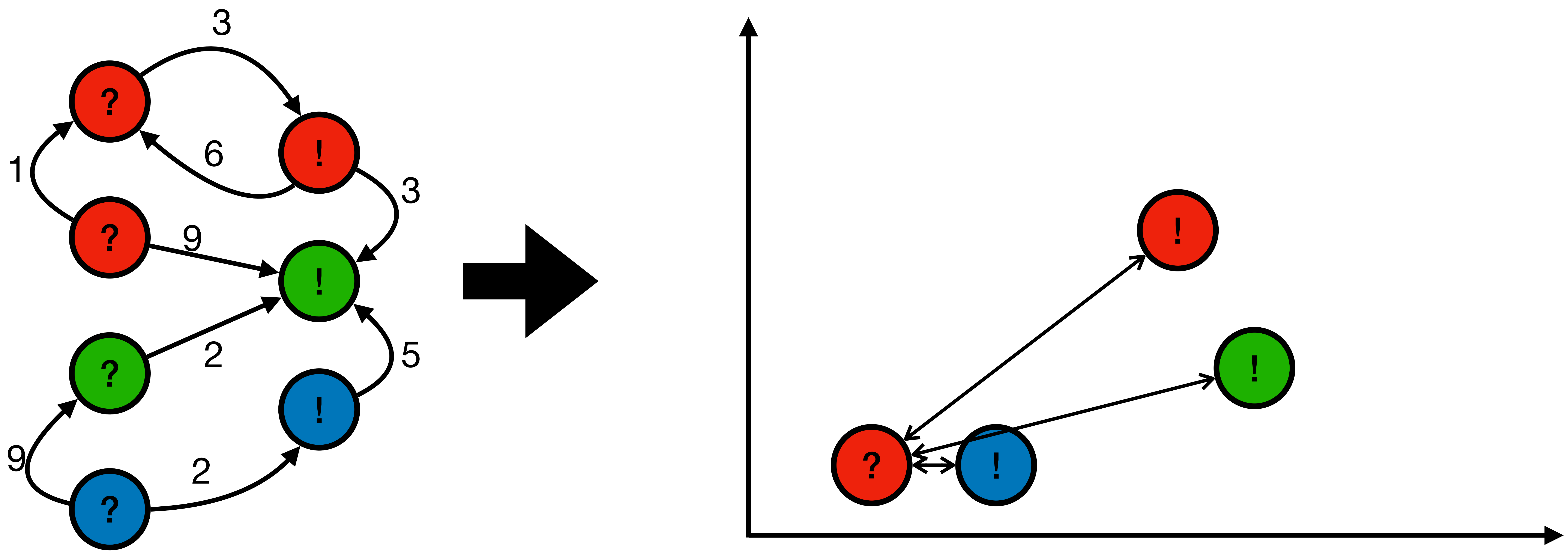
Embedding → partition



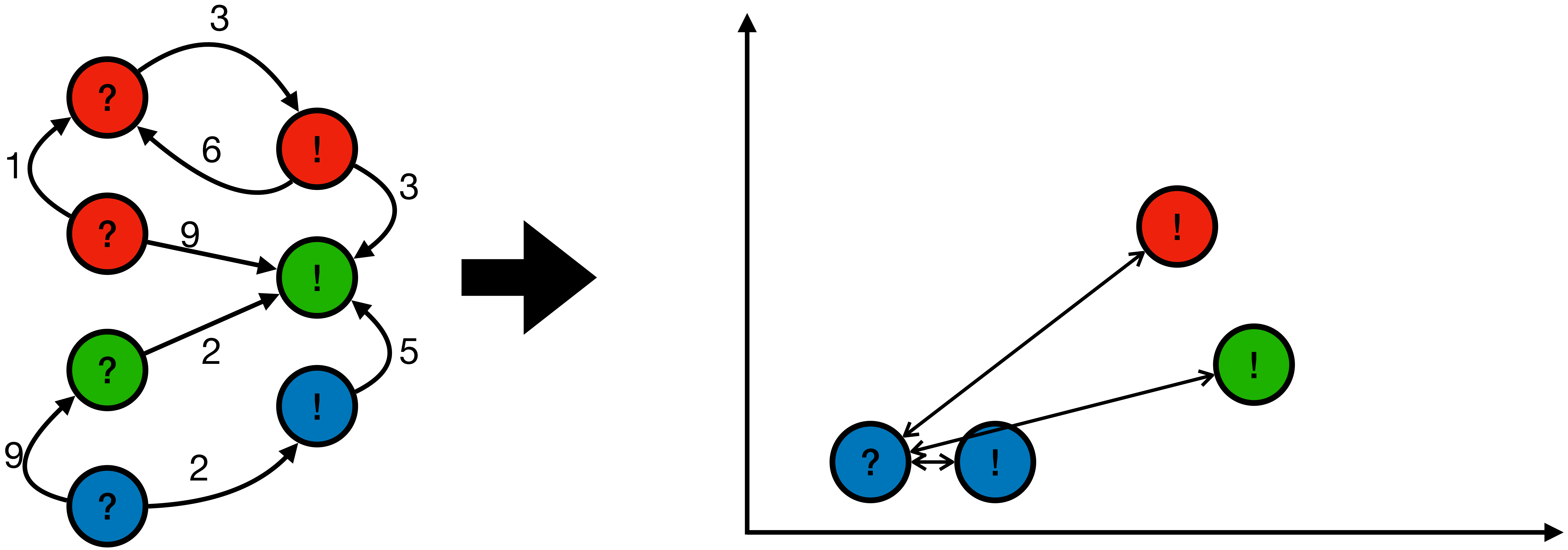
Embedding → partition



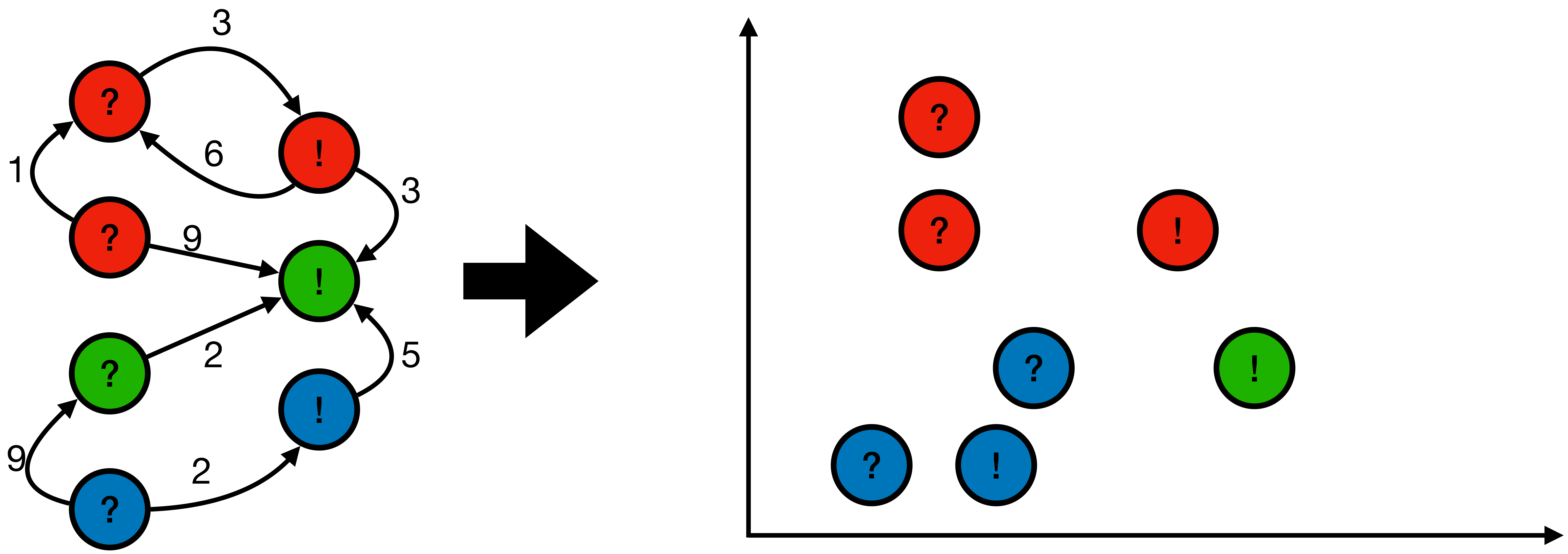
Embedding → partition



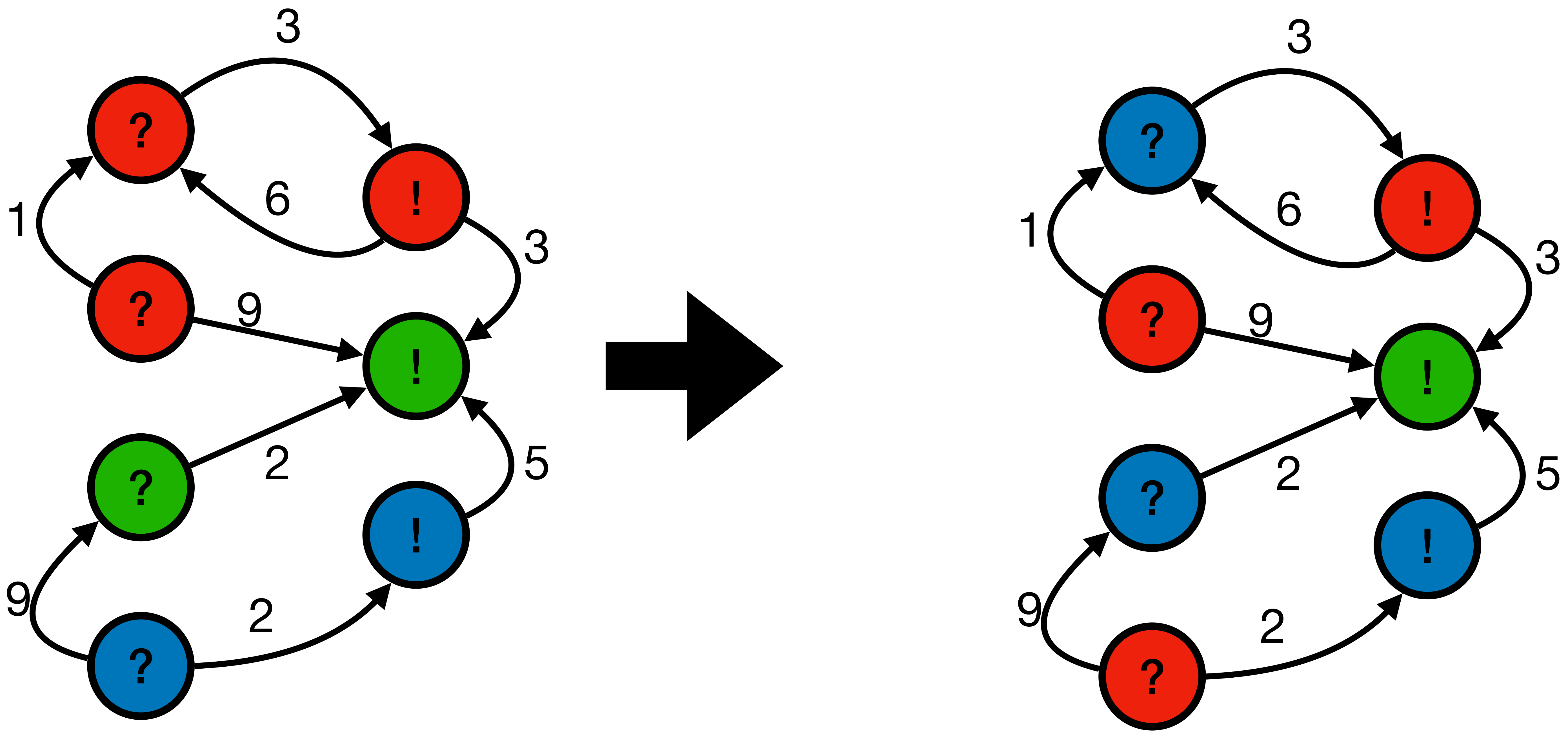
Embedding → partition



Embedding → partition



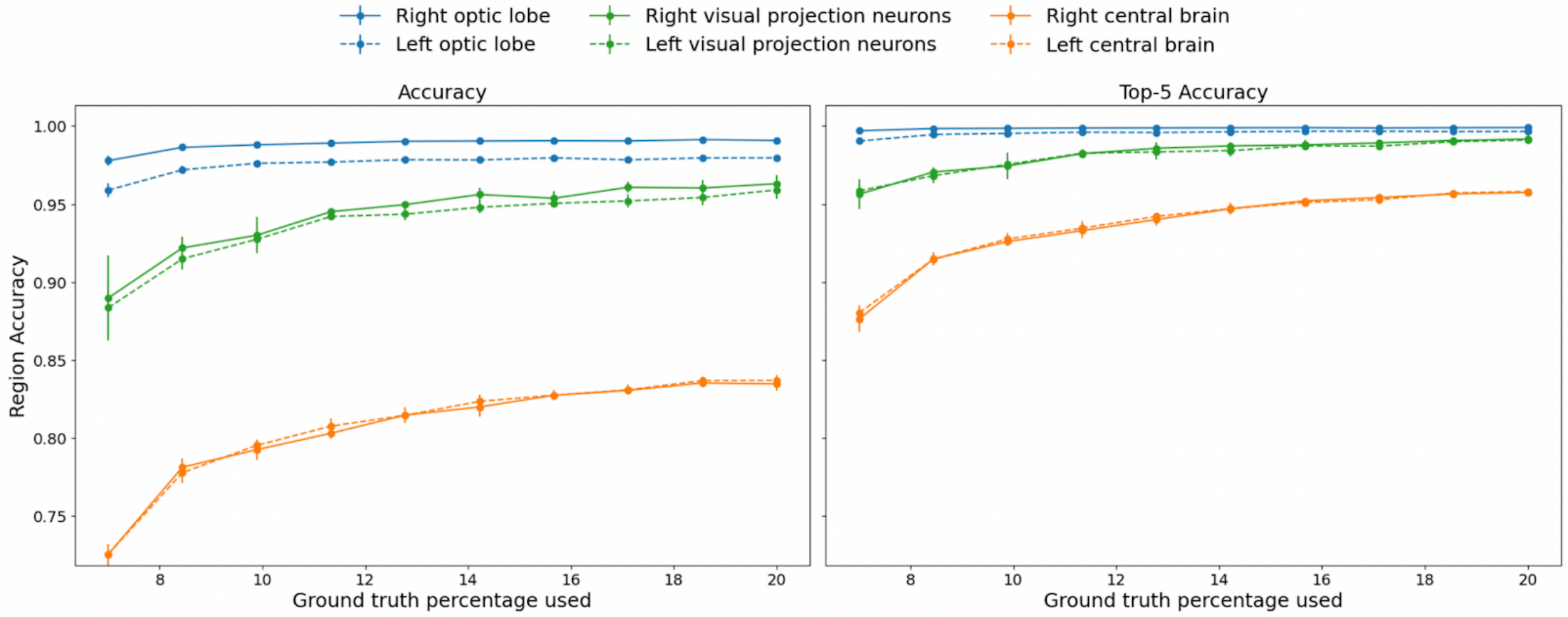
Embedding → partition



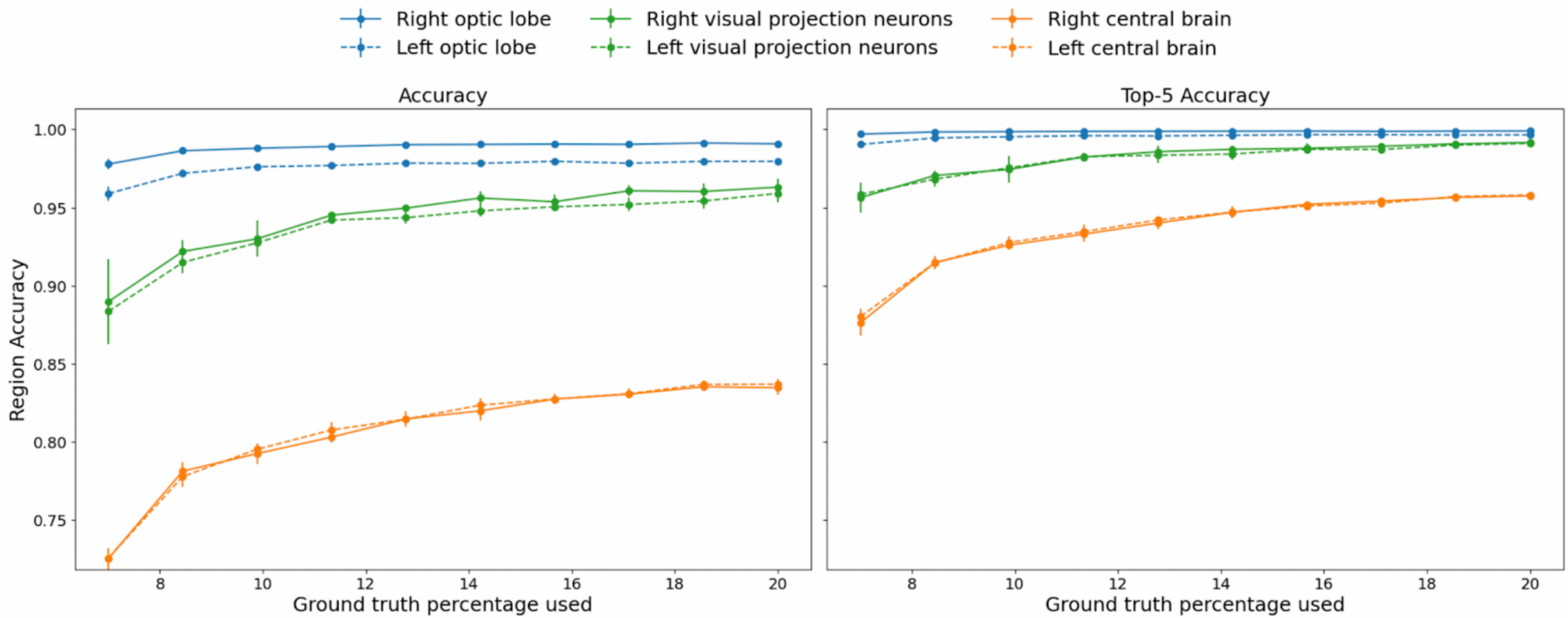
Runtime

- Algorithm can be parallelized
- Only a few (~10) iterations are sufficient
- Entire brain can be classified in ~10 minutes on a mac

Experimental results

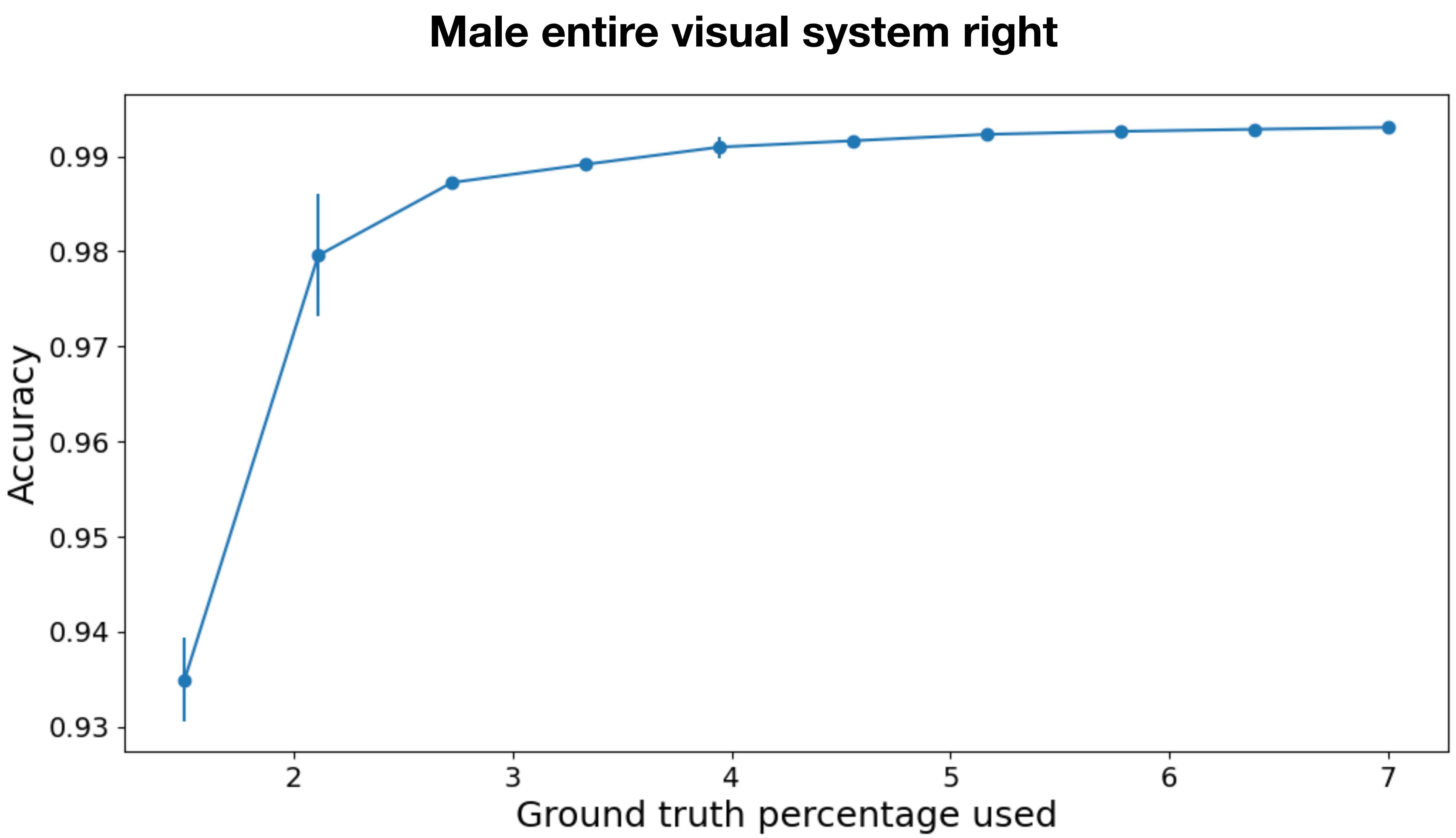


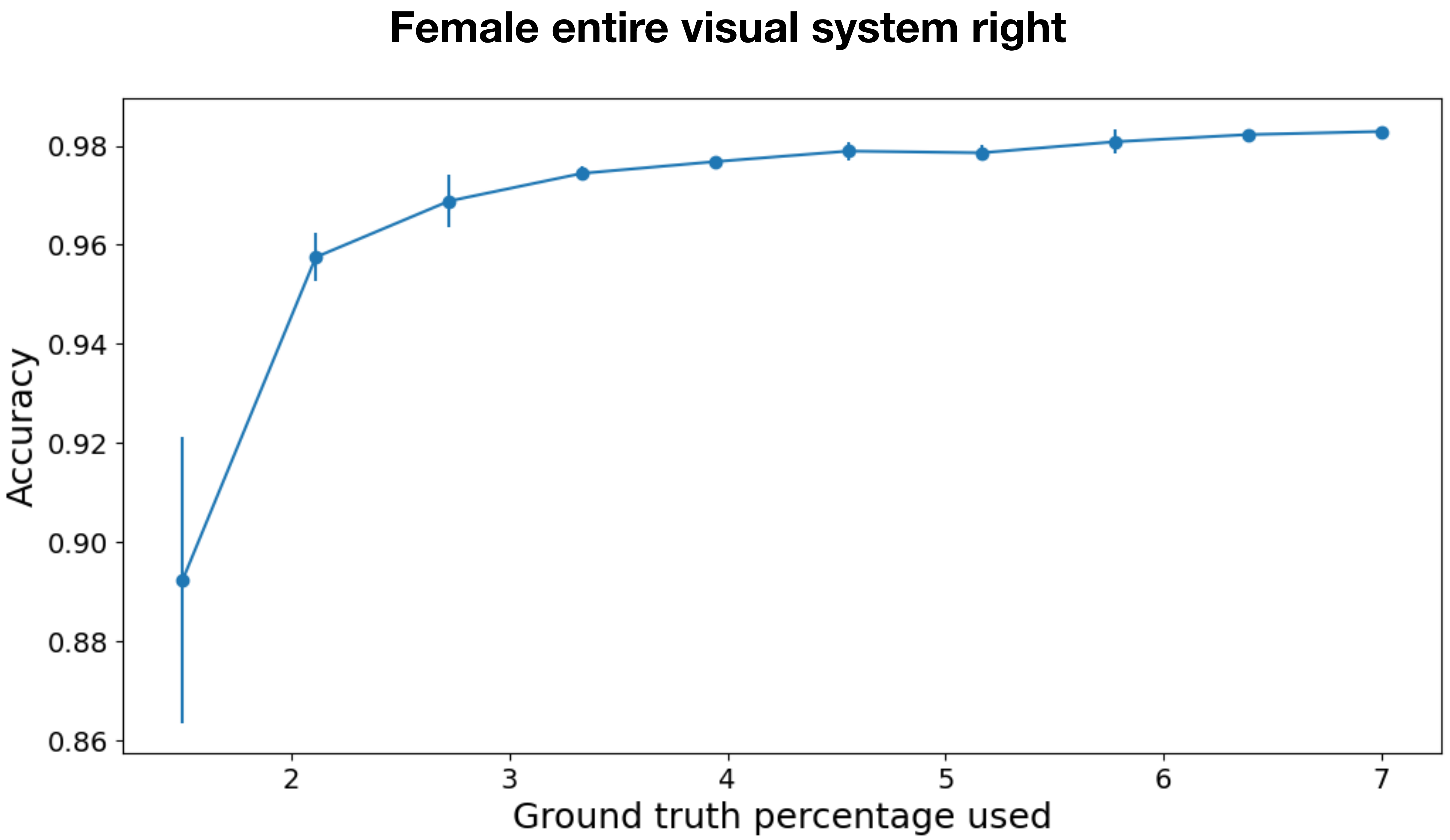
Experimental results



We assume at least one labelled node from each type is given

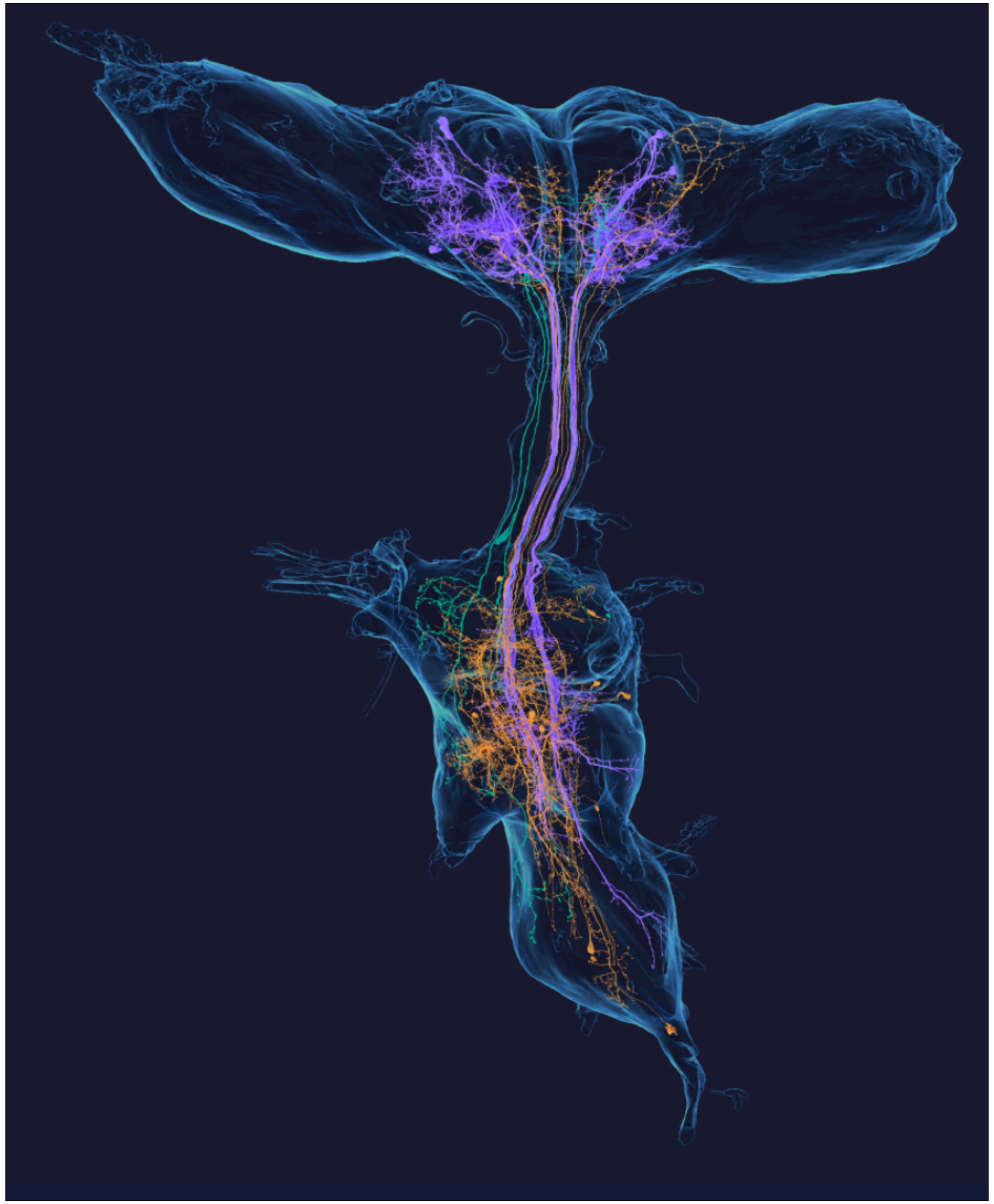
Experimental results



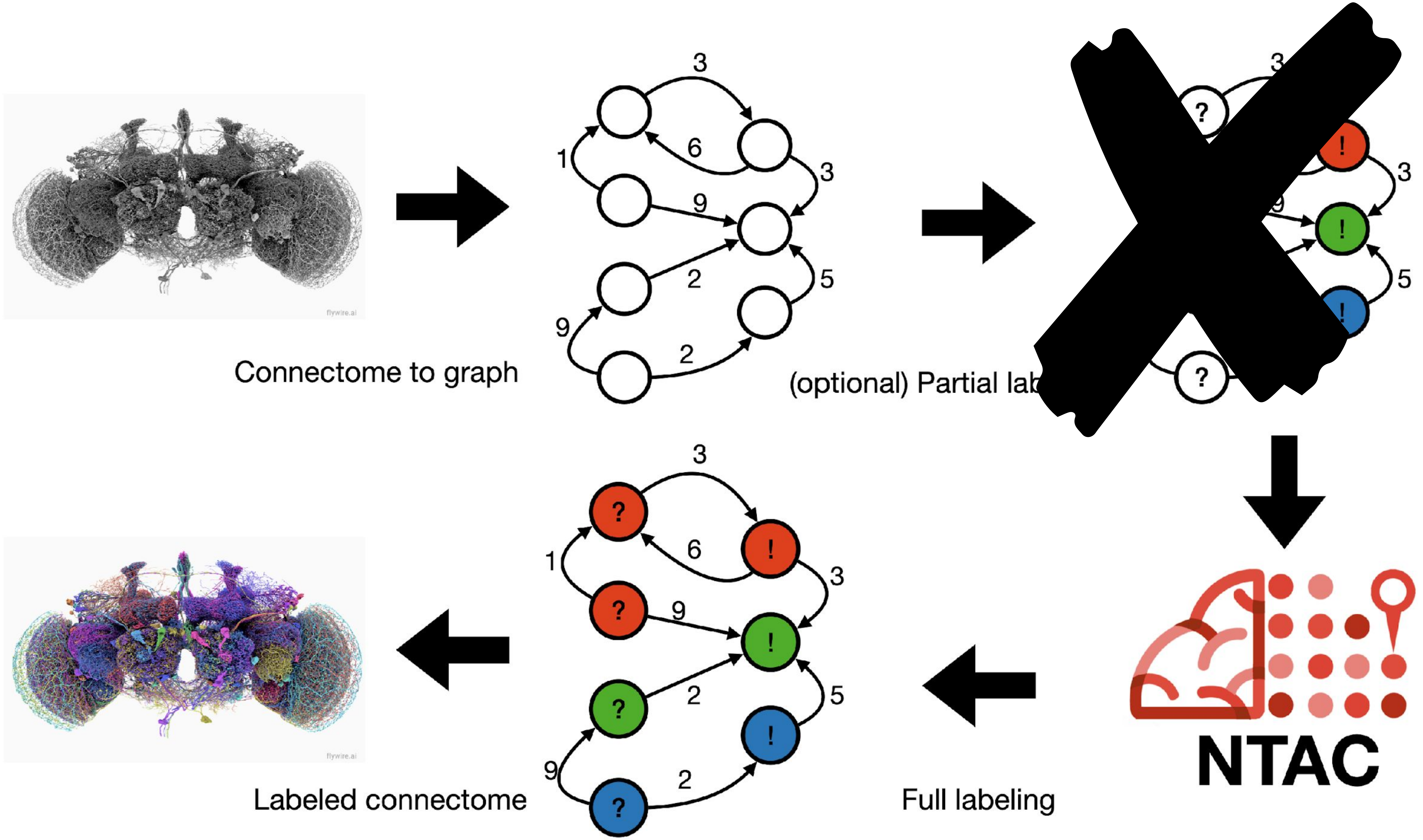


Incomplete data?

- **BANC** - brain and nerve cord
  - Adult female fruit fly
- ~100k neurons / ~2.5 million synaptic connections
  - Incomplete topology
- ~6k cell types
  - Missing types
- ~45% labeled cells
  - Labeling is not uniform



Unseeded version



Connectome to graph

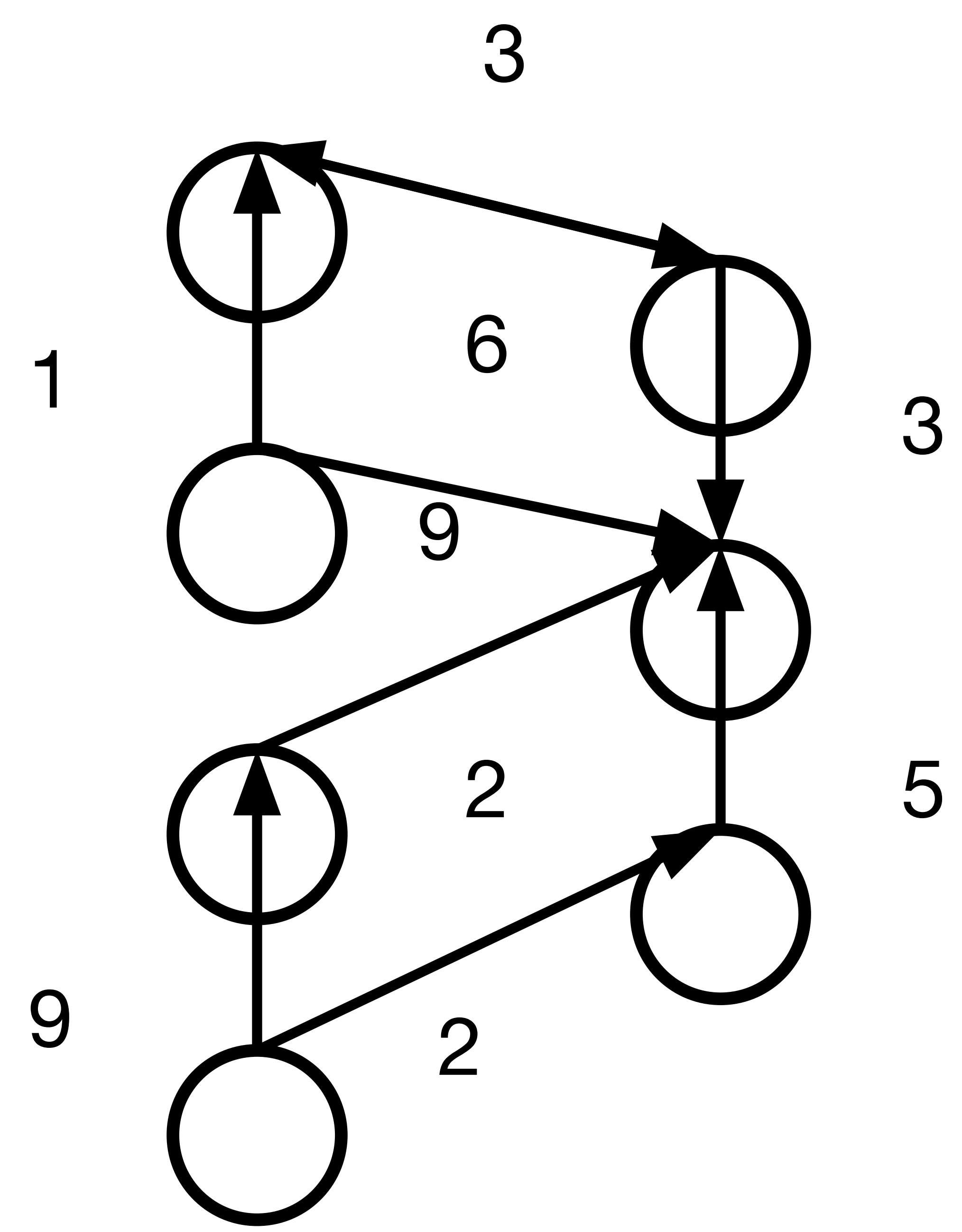
(optional) Partial lab

Full labeling

Labeled connectome

Unseeded version

- **Input**: weighted graph,
  desired # of clusters



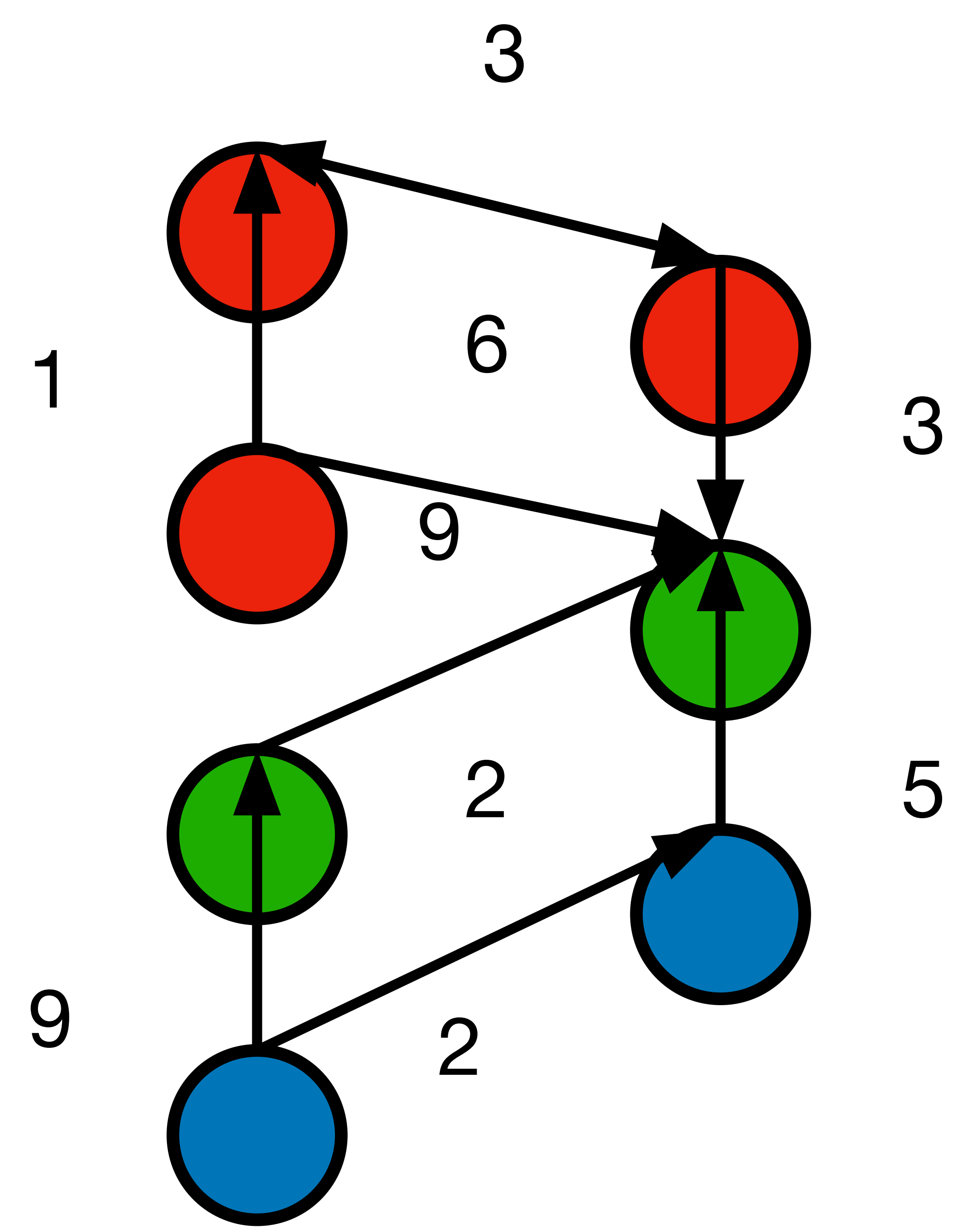
Unseeded version: setting

- **Input**: graph *G*,
  desired # of clusters *k*

- **Output**: a clustering (= partition = coloring) of *G*

- **Goal**: nodes of the same color should have

  the same embedding



Exact Equitable partitioning

- Recall: embedding of *v* = degree counts between *v* and each of the clusters in *C*

- If nodes of the same color have **exactly** the same embedding

  → partition is called *equitable*

- Classic problem studied since the 70s (Schenk 1974).

- An iterative algorithm finds the coarsest equitable partition

  Problem: exact equality requires too many clusters!

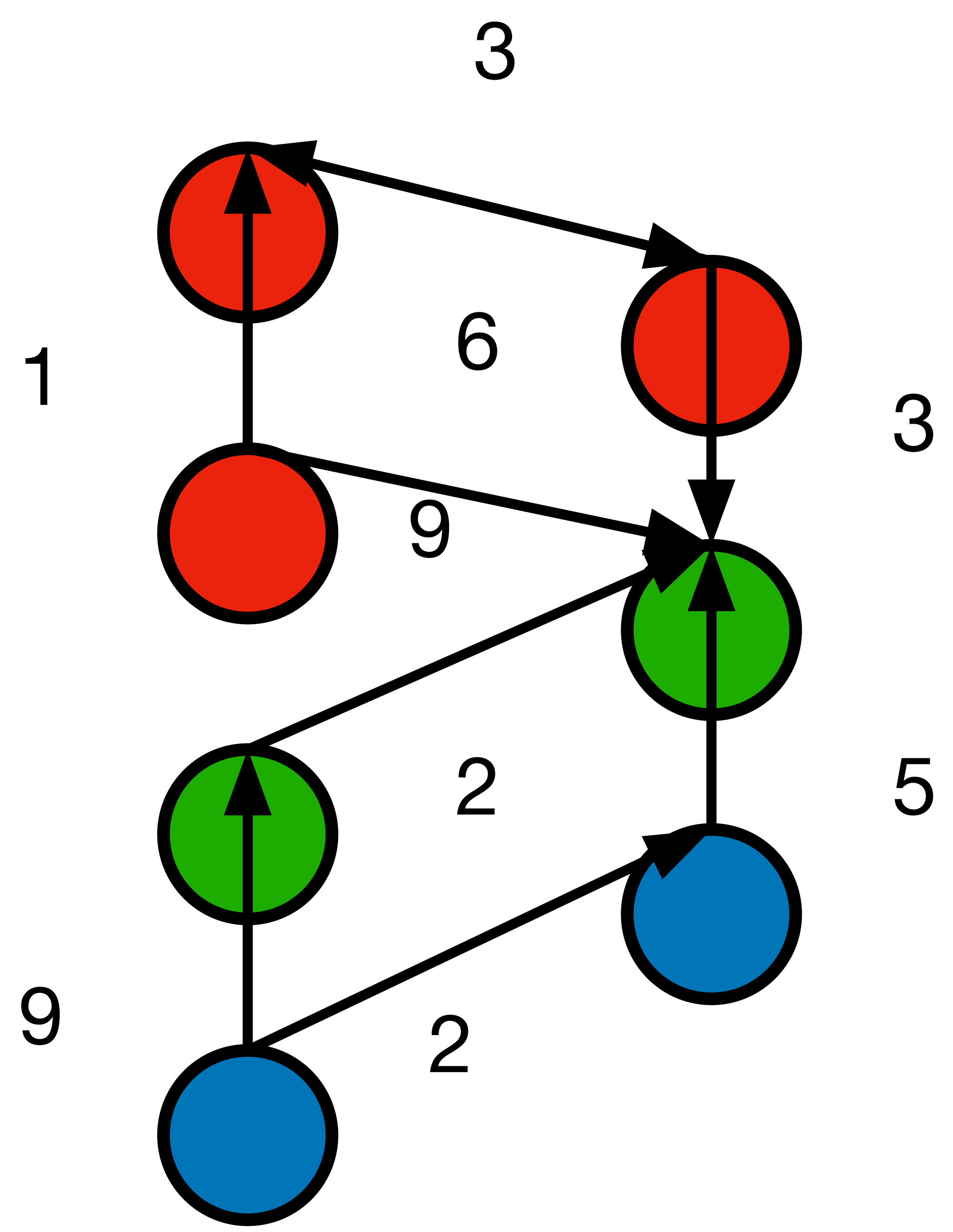
Approximate equitable partitioning

- **Input**: graph *G*,
  desired # of clusters *k*

- **Output**: a clustering (= partition = coloring) of *G*

- **Goal**: nodes of the same color should have

  almost the same embedding

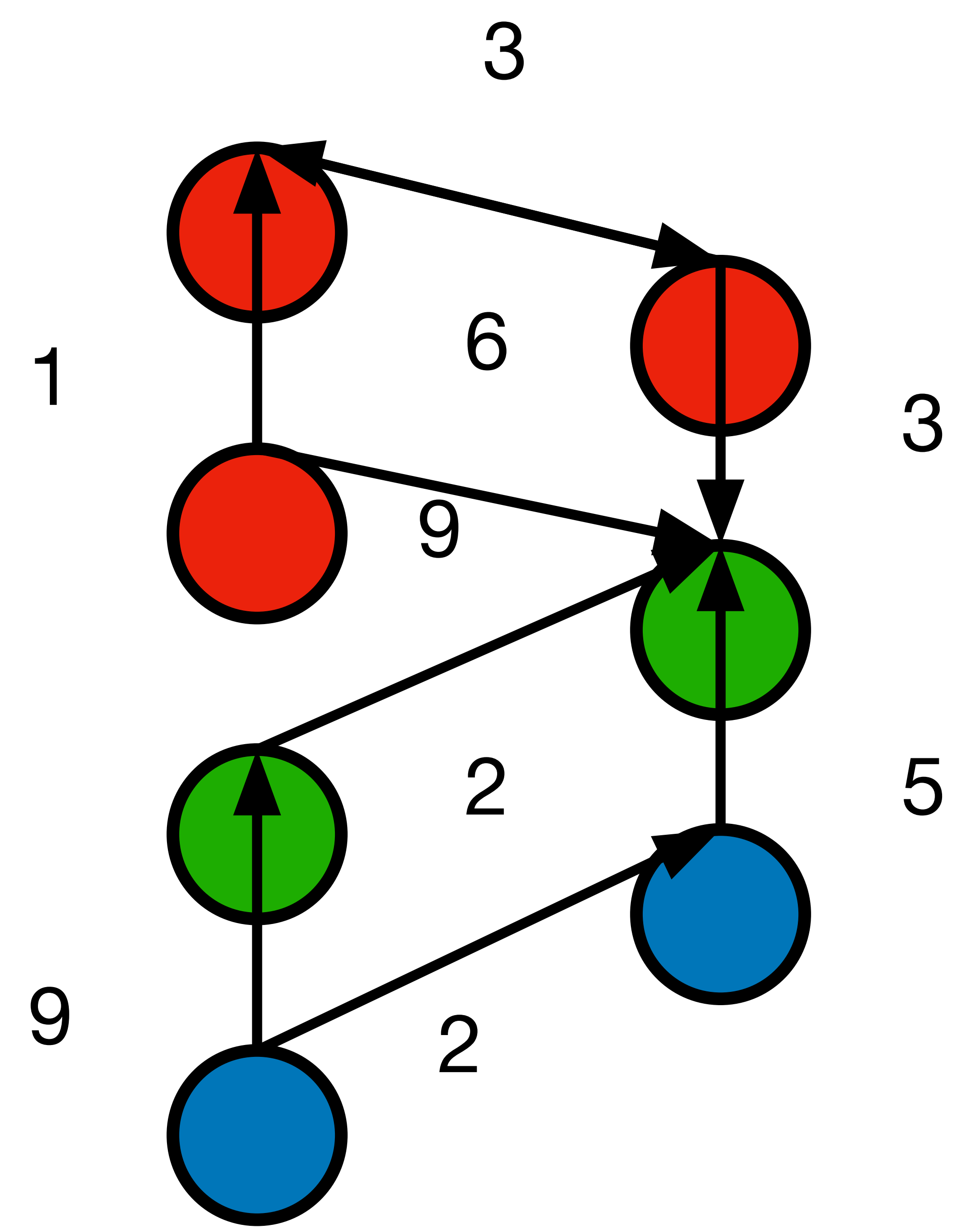


Approximate equitable partitioning

- **Input**: graph *G*,
  desired # of clusters *k*
- **Output**: a clustering (= partition = coloring) of *G*
- **Goal**: nodes of the same color should have

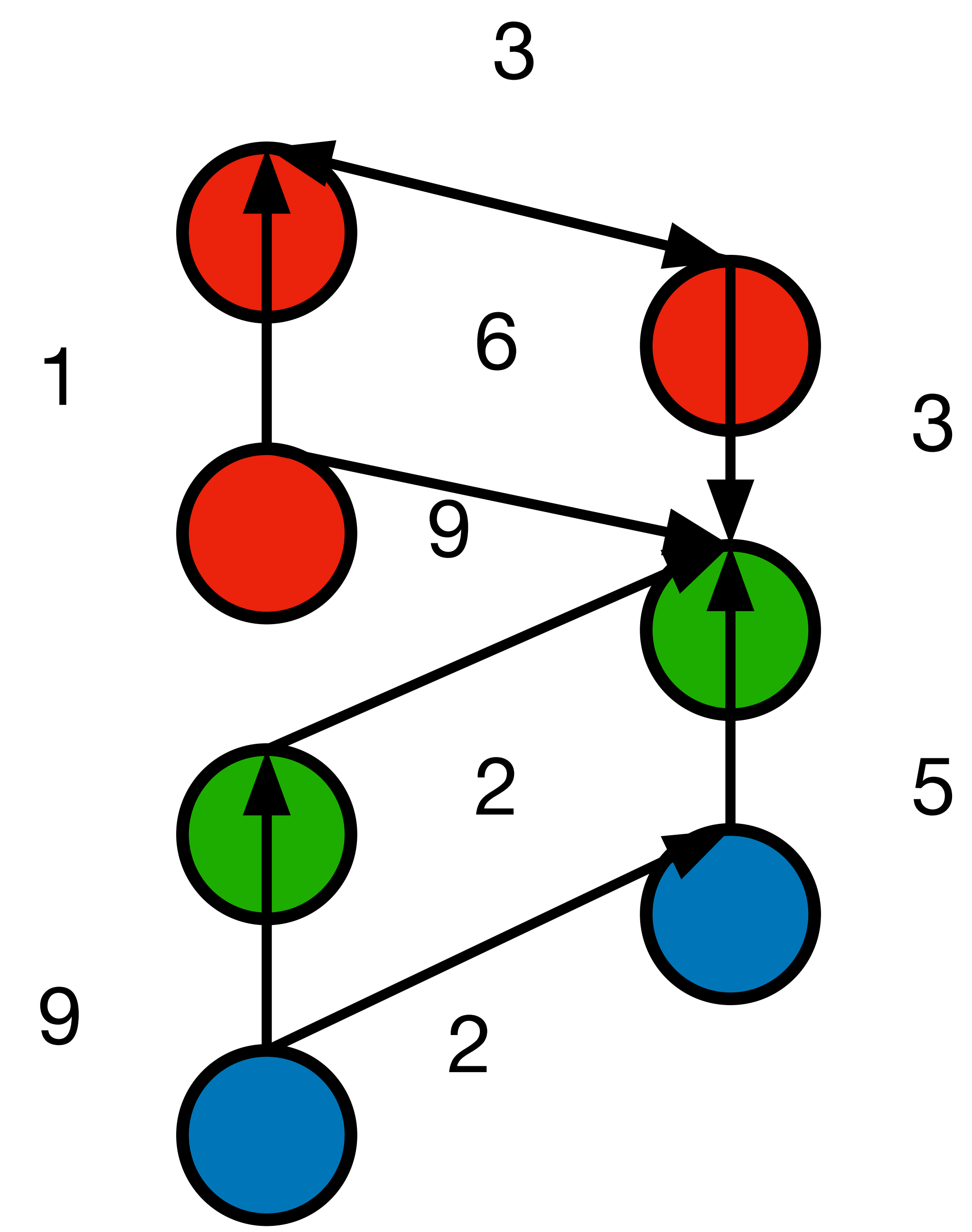
  almost the same embedding

Jaccard distance between the embedding of v and the median embedding of v's cluster



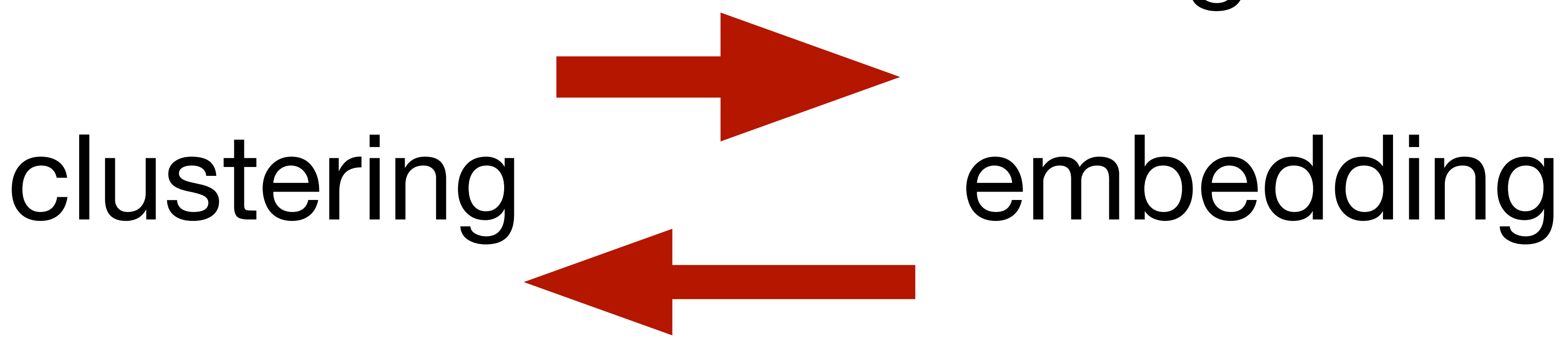
Approximate equitable partitioning

- **Input**: graph *G*,
  desired # of clusters *k*
- **Output**: a clustering (= partition = coloring) of *G*
- **Goal**: minimize the sum of Jaccard distances between the embedding of each vertex and the median embedding of its cluster



Algorithm: main idea

- Recall that the embedding of a vertex depends on the clustering C



clustering

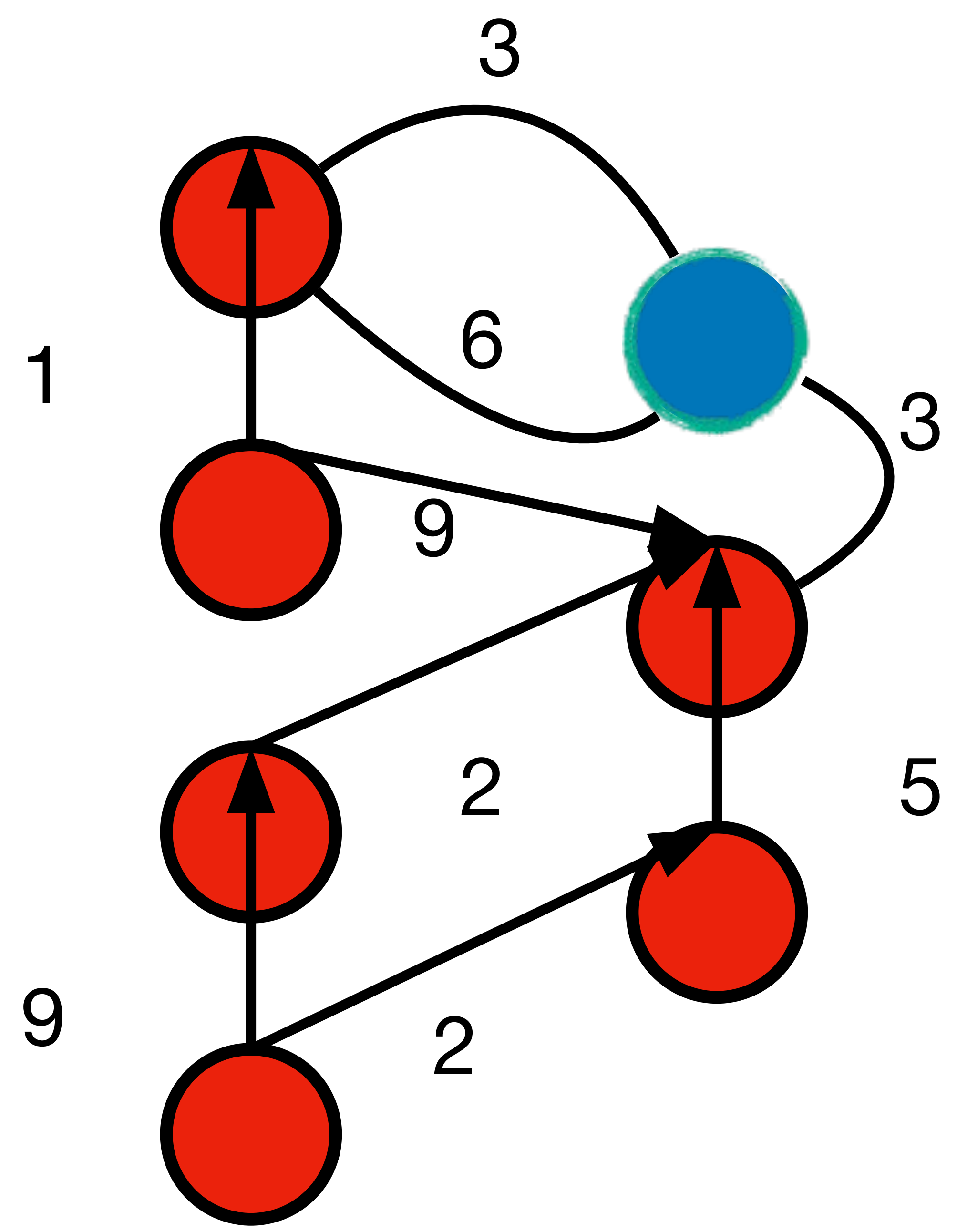
embedding

- Circular dependency: classical clustering algorithms do not work.
- But if we could identify a good set of seeds, we could use the **seeded** algo!
- **Idea**: grow a set of seeds iteratively

Algorithm outline

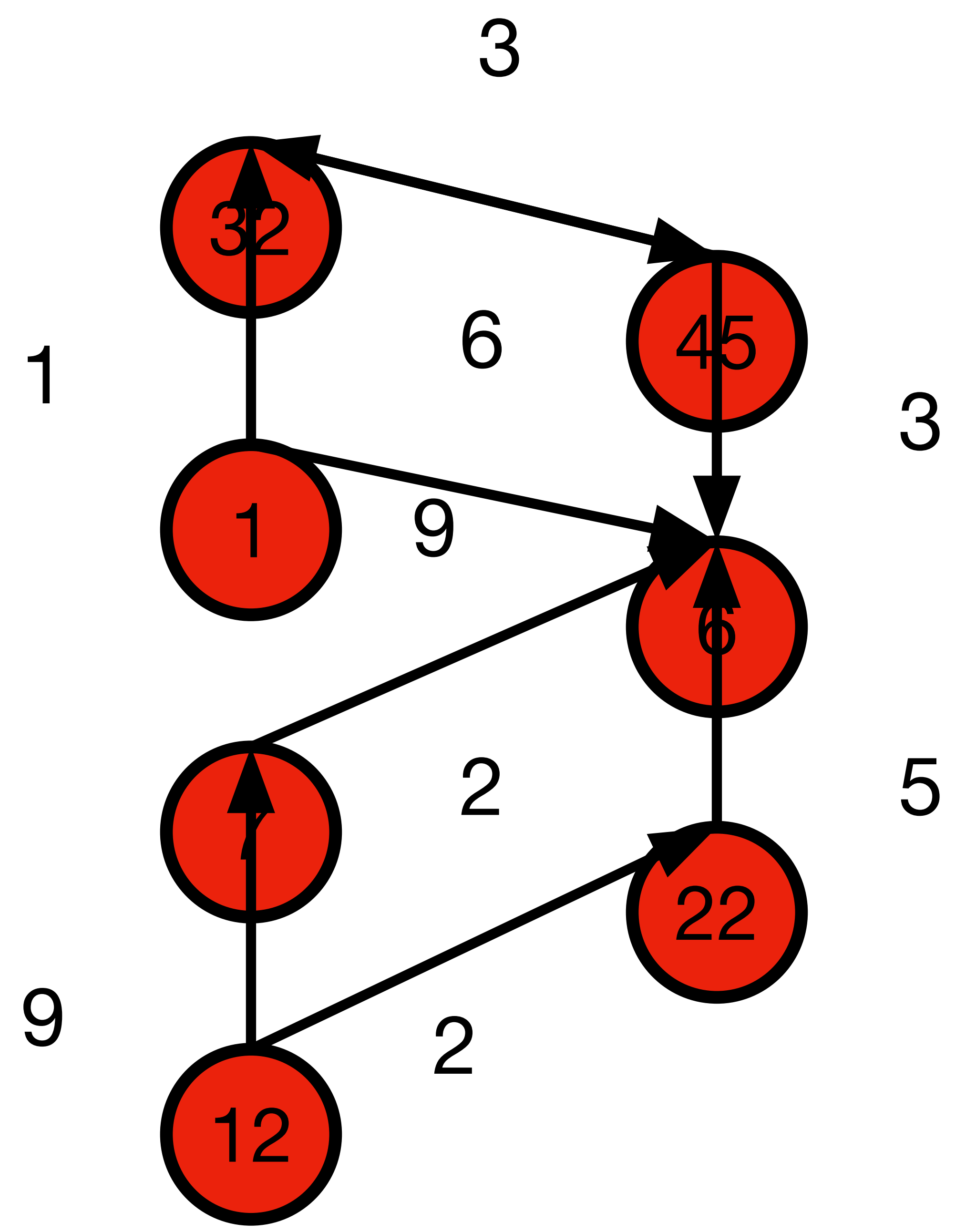
- Initially all nodes are in the same cluster
- For each number of clusters from 1 to *k*
  - Assign a score to every node

Score: How much the goal improves by making a new cluster with v and moving nodes that “prefer” to go with v, using current embedding



Algorithm outline

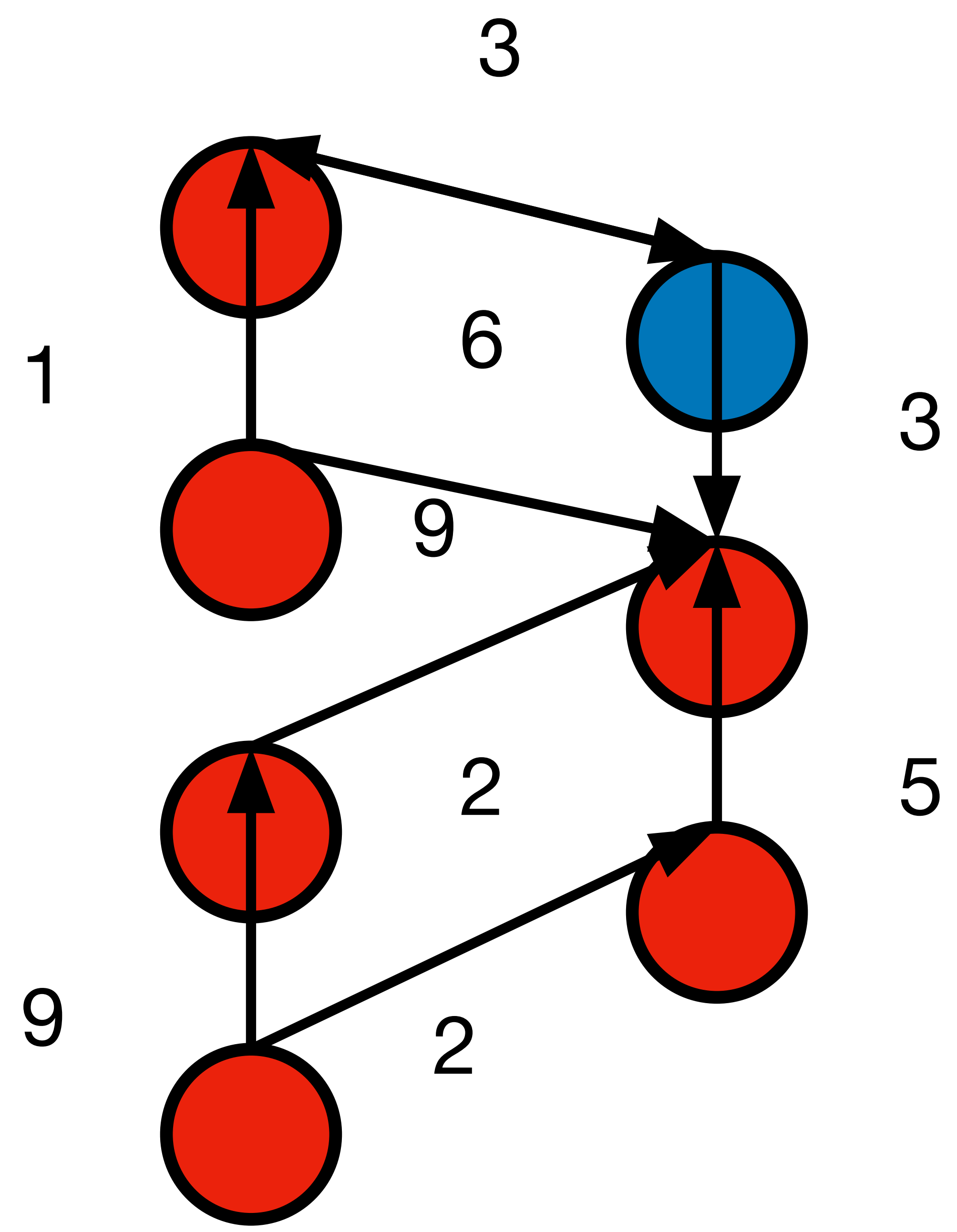
- Initially all nodes are in the same cluster
- For each number of clusters from 1 to *k*
  - Assign a score to every node



Scores may be negative

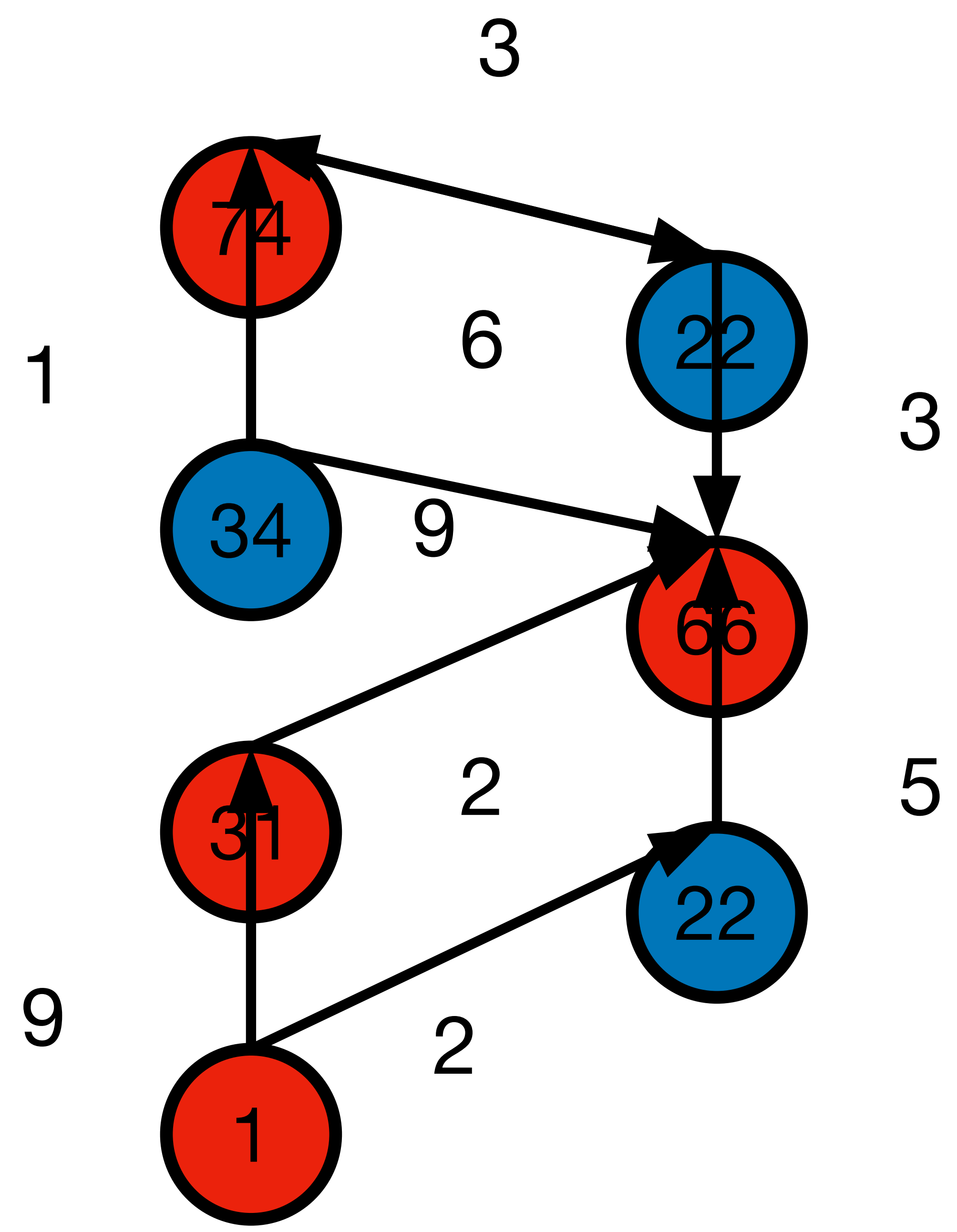
Algorithm outline

- Initially all nodes are in the same cluster
- For each number of clusters from 1 to *k*
  - Assign a score to every node
  - Pick the highest-score node as a new seed

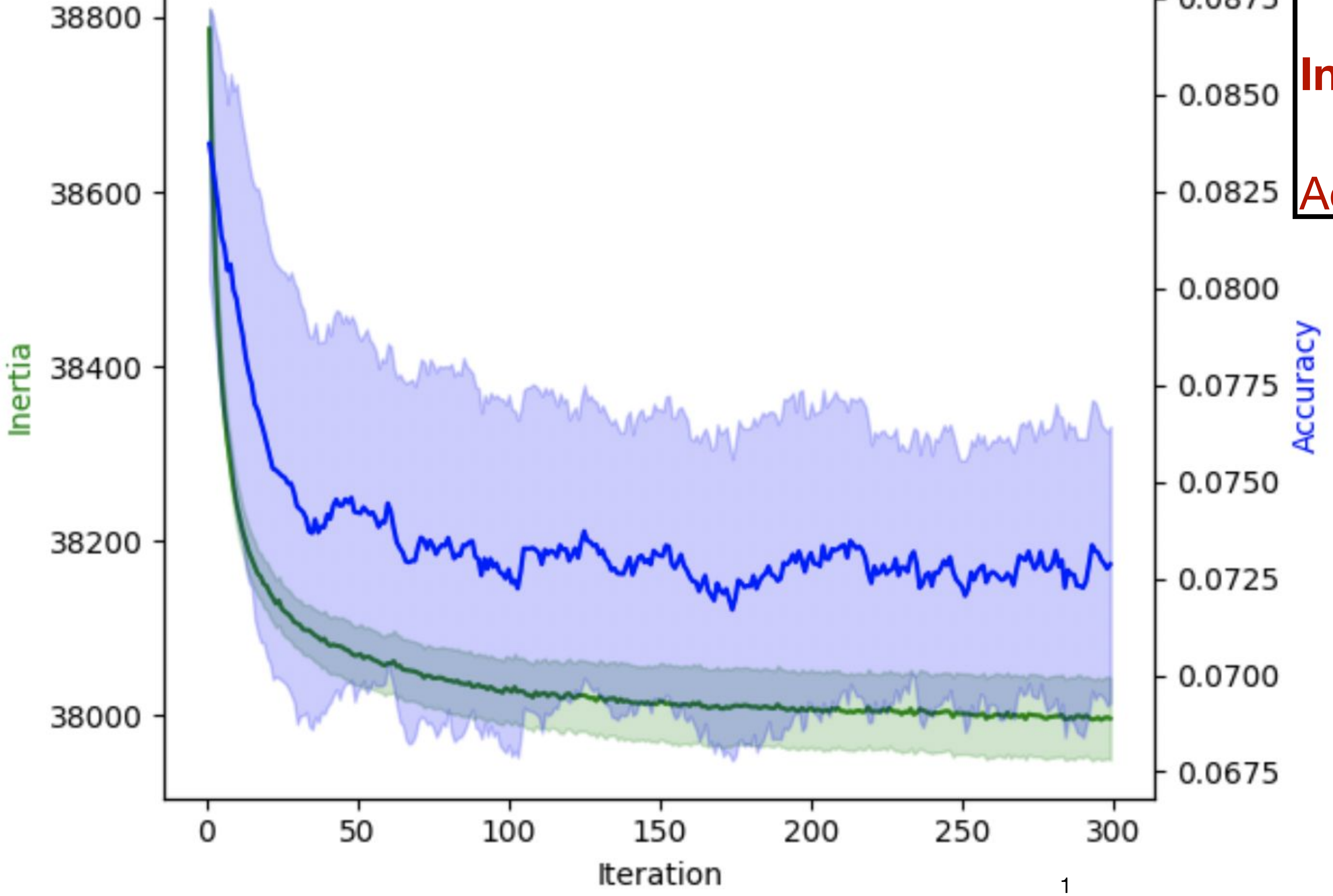


Algorithm outline (final)

- Initially all nodes are in the same cluster
- For each number of clusters from 1 to *k*
  - Assign a score to every node
  - Pick the highest-score node as a new seed
  - Run seeded algorithm to find clustering C_k
- Update seeds:
  let them be the closest nodes to each cluster median



Unseeded baseline - experimental results

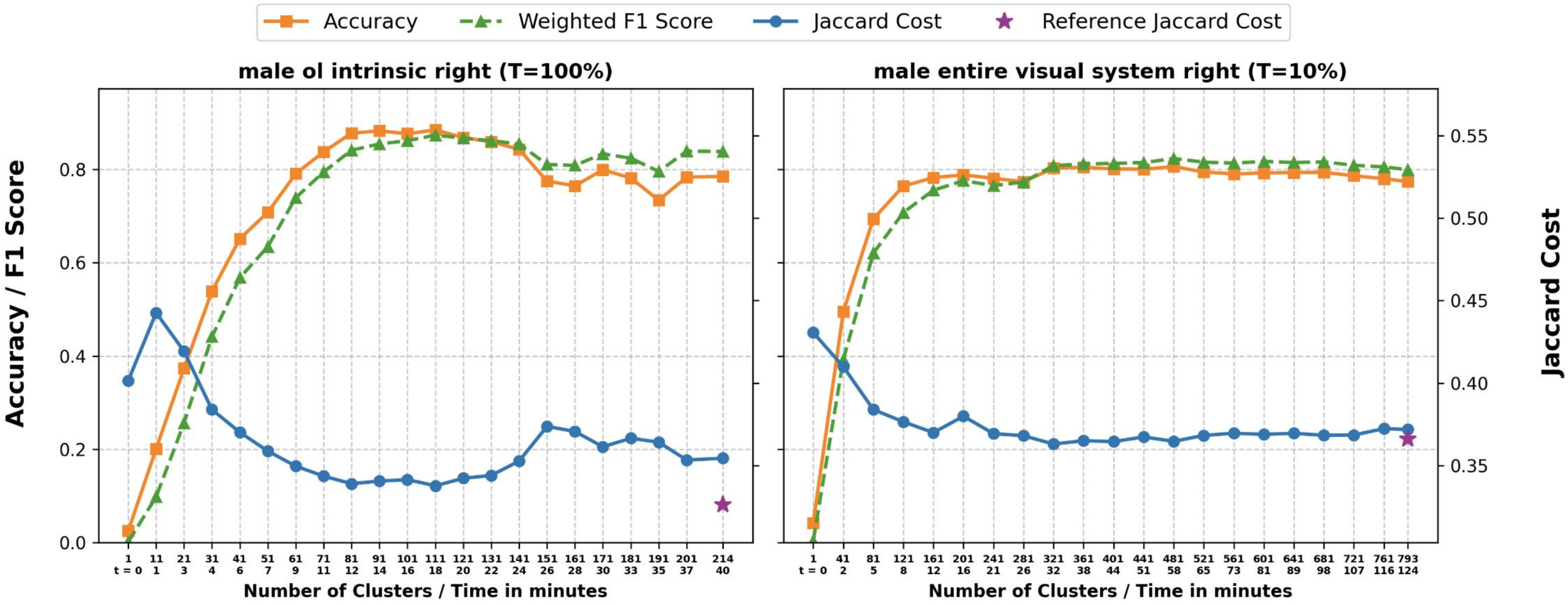


Kernel k-means + NBLAST similarity matrix

Inertia - kernel k-means goal function

Accuracy **goes down** the more we optimize

Unseeded NTAC - experimental results



T=10%: Only consider 10% of candidates per cluster (speed optimization)

Future work

- Unseeded algo:
    - Improve accuracy and/or speed
    - Is it possible to detect the best *k* automatically?
    - Study approximation guarantees / hardness results for approximate equitable partitioning
- Seeded algo: improve accuracy using ML
- Other applications for NTAC

https://github.com/BenJourdan/ntac