Machine Learning with Graphs (MLG)

Network Community Detection (3)

Louvain and LPA algorithms

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

chengte@mail.ncku.edu.tw

Community Detection Approaches

- Propagation-based Method
  - Structural Clustering Algorithm for Networks (SCAN)
- Edge-Removal
  - Girvan-Newman Algorithm (GNA)
  - Fast Newman Algorithm
- Louvain Algorithm
- Label Propagation Algorithm

Louvain Algorithm

- V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. "**Fast unfolding of communities in large networks**" J. Stat. Mech. P10008, 2008.

11087 cites

- Implementations/Packages
  - https://github.com/taynaud/python-louvain
  - https://github.com/patapizza/pylouvain
  - https://github.com/vtraag/louvain-igraph
  - https://python-louvain.readthedocs.io/en/latest/

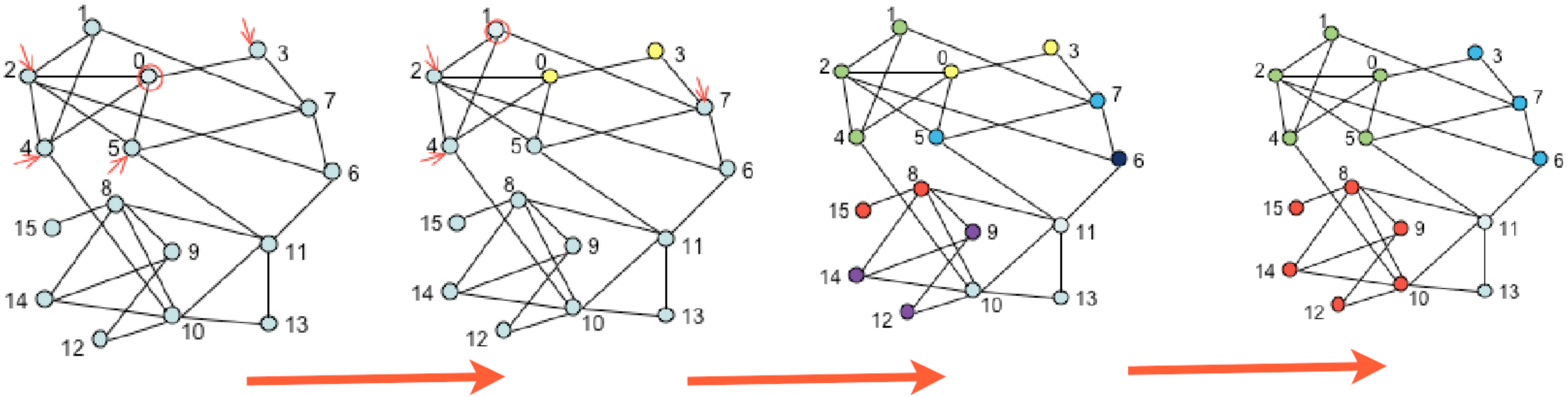
Louvain Algorithm

- One of the state-of-the-art methods
- A greedy optimization method that attempts to maximize the modularity of a partition of the network
- Two main steps repeated iteratively

```
while (improvement) {
        mod = detect-communities()        # phase 1
        if( (mod – prev_mod) > 0)
            improvement = true
        else
            improvement = false
        prev_mod = mod
        collapse-graph()                  # phase 2
}
```

Louvain Algorithm: Phase 1

- Initially, all the nodes belong to their own communities
- One pass: looks through all the nodes in an ordered way
- The nodes look among their neighbors and
  adopt the community of their neighbor
  if there is an increase of modularity ($\Delta Q > 0$)
- Perform passes iteratively until a local maximum of modularity is reached (each node could be considered several times)



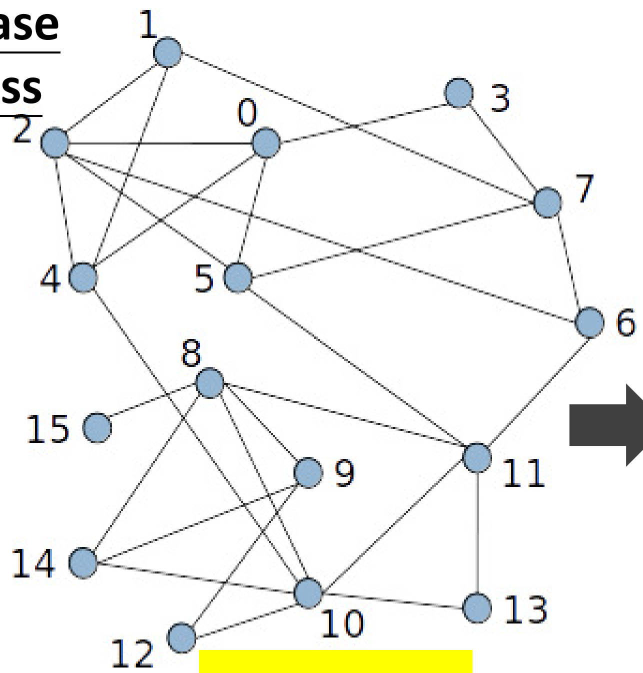
Node 0 moves to the community of Node 3

After N nodes have been considered

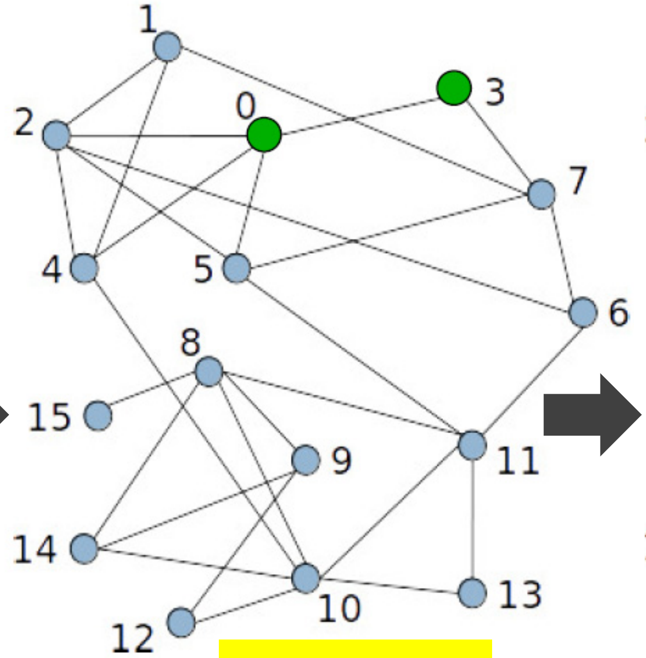
After each nodes has been considered 4 times

First Phase

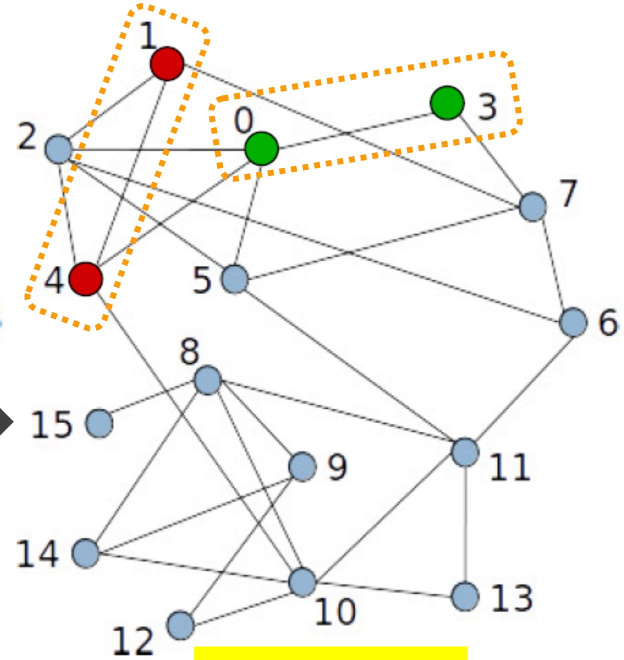
First Pass



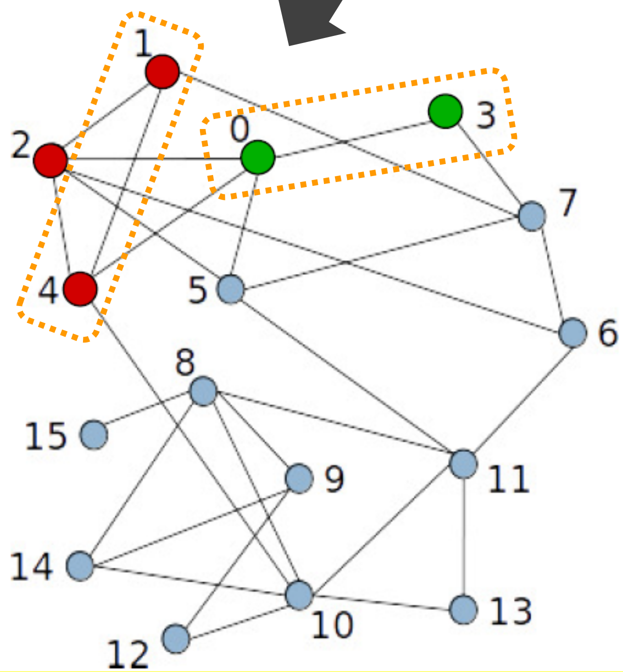
Consider 0



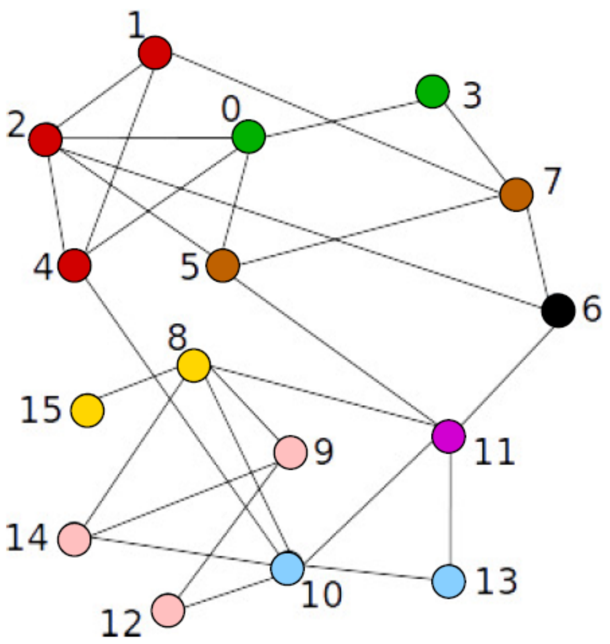
Consider 1



Consider 2

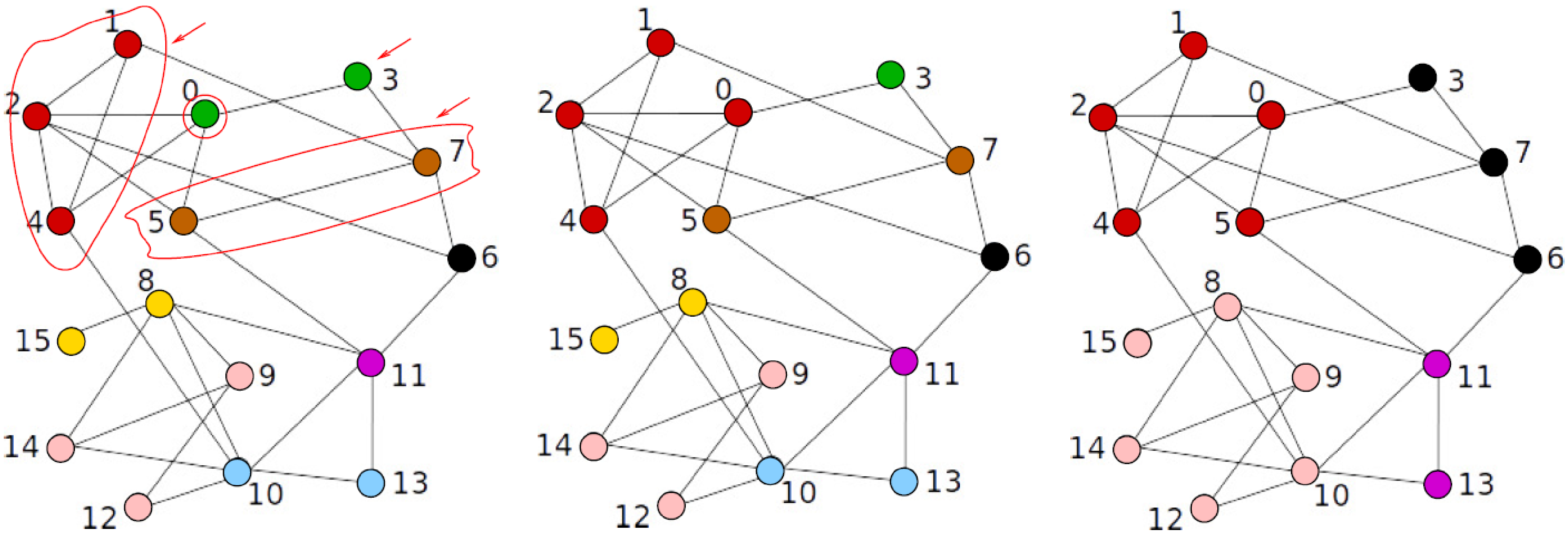


Consider 3→Stay with 0 (otherwise Q decreases)



and so on..., end of first pass

First Phase
Second Pass



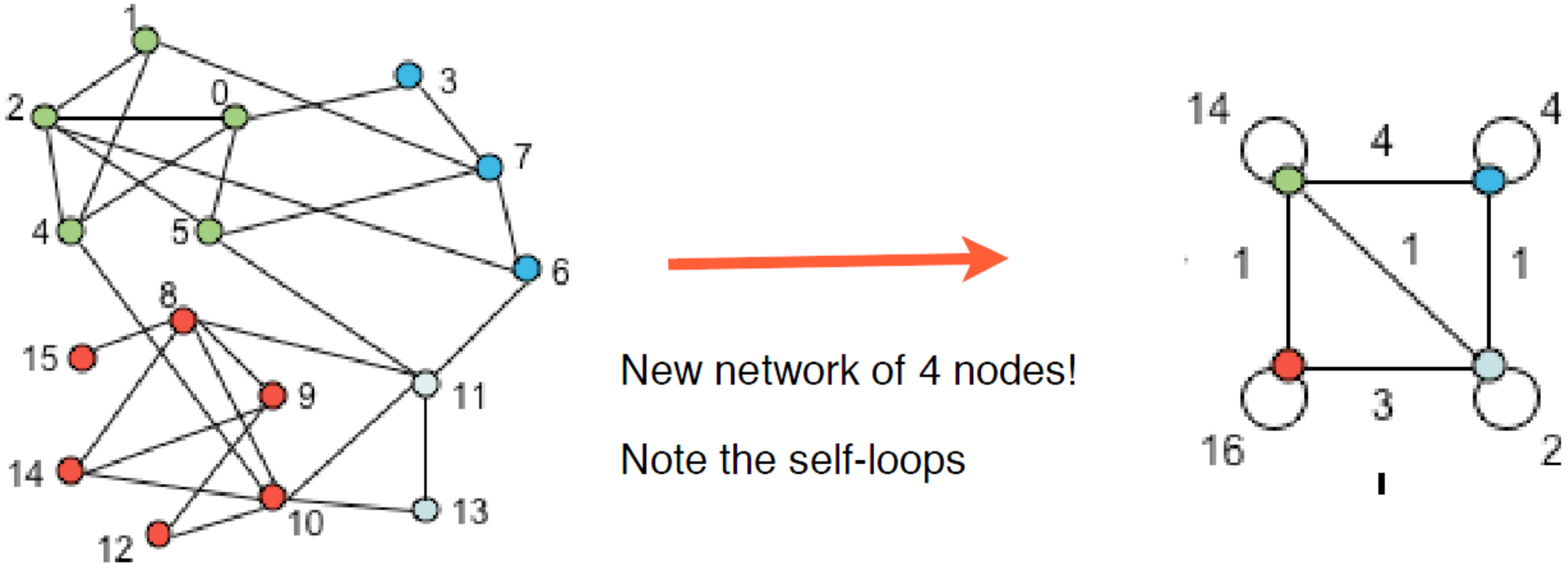
Consider 0

0 is put in C(1,2,4), best Q increase

after 4 iterations, no change anymore

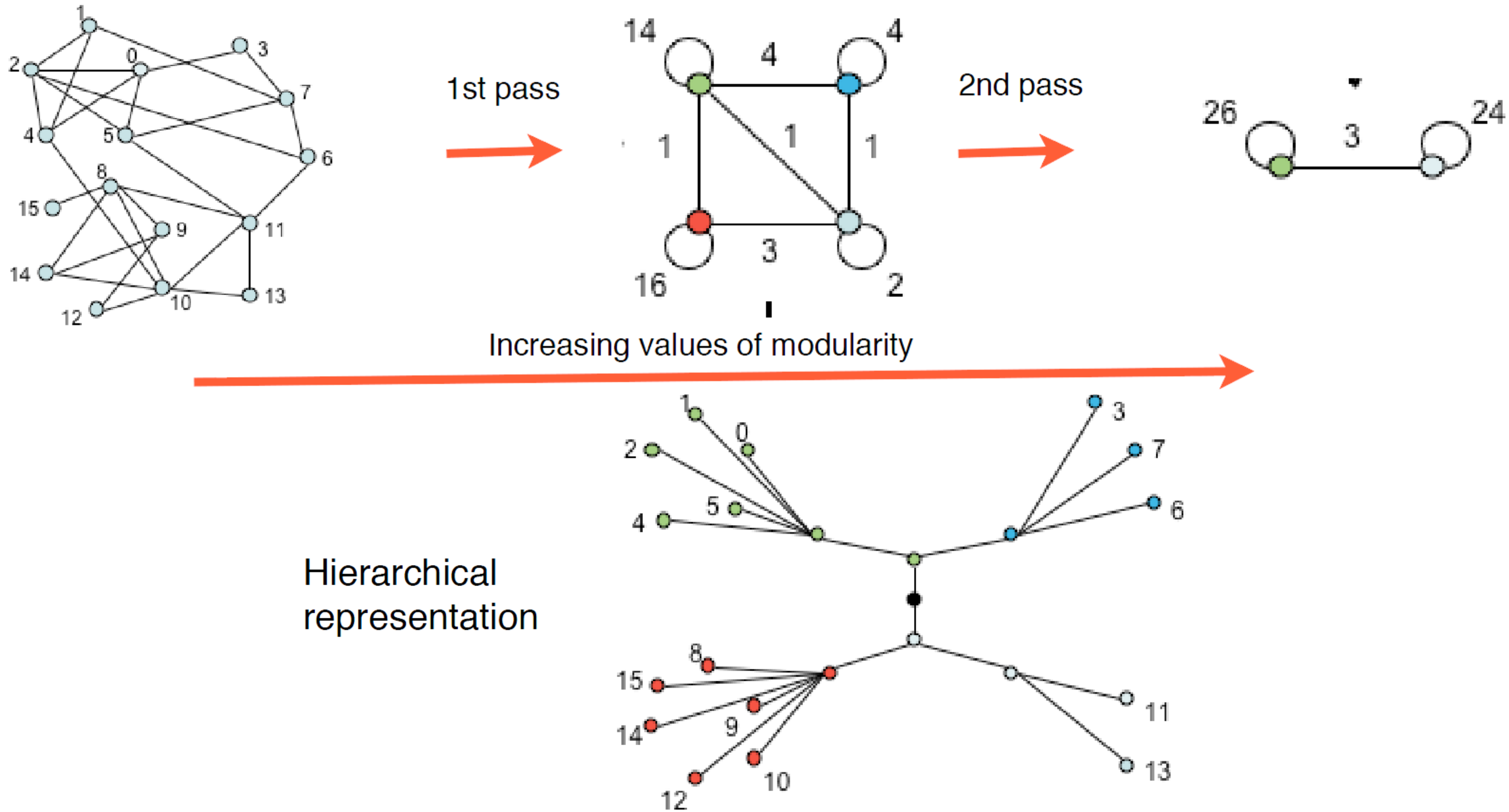
Louvain Algorithm: Phase 2

- Once a local maximum has been attained
  → build a new network whose nodes are communities
  - Edge weights between communities are the total numbers of edges between nodes of two communities
  - In typical realizations, the number of nodes diminishes drastically at this step
  - This ensures the rapidity of the algorithm for large networks



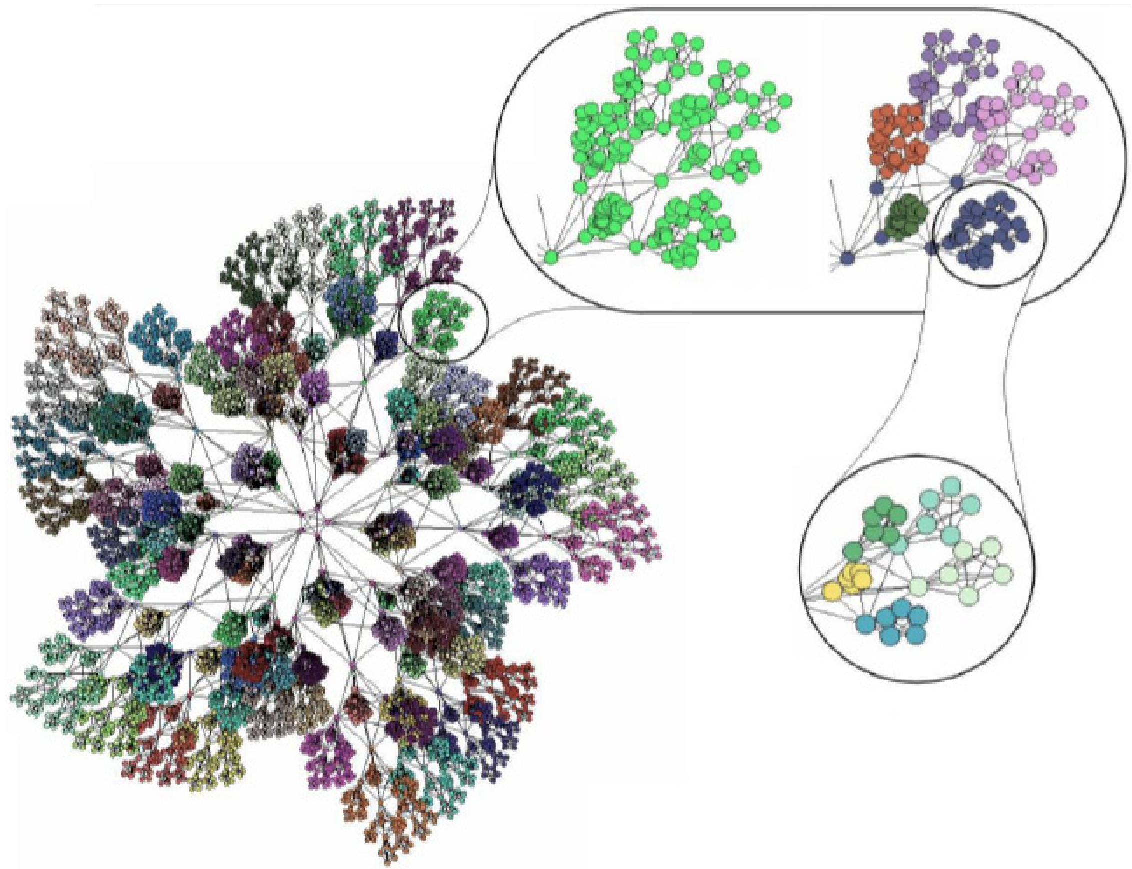
Louvain Algorithm

- The two steps are repeated iteratively, thereby leading to a hierarchical decomposition of the network



Advantages of Louvain Algorithm

- Incredibly simple (local greedy approach)
  → Easy to implement and to improve
- Very fast! Even in large-scale networks (5.5M)
- Good accuracy
- Multi-resolution



Community Detection Approaches

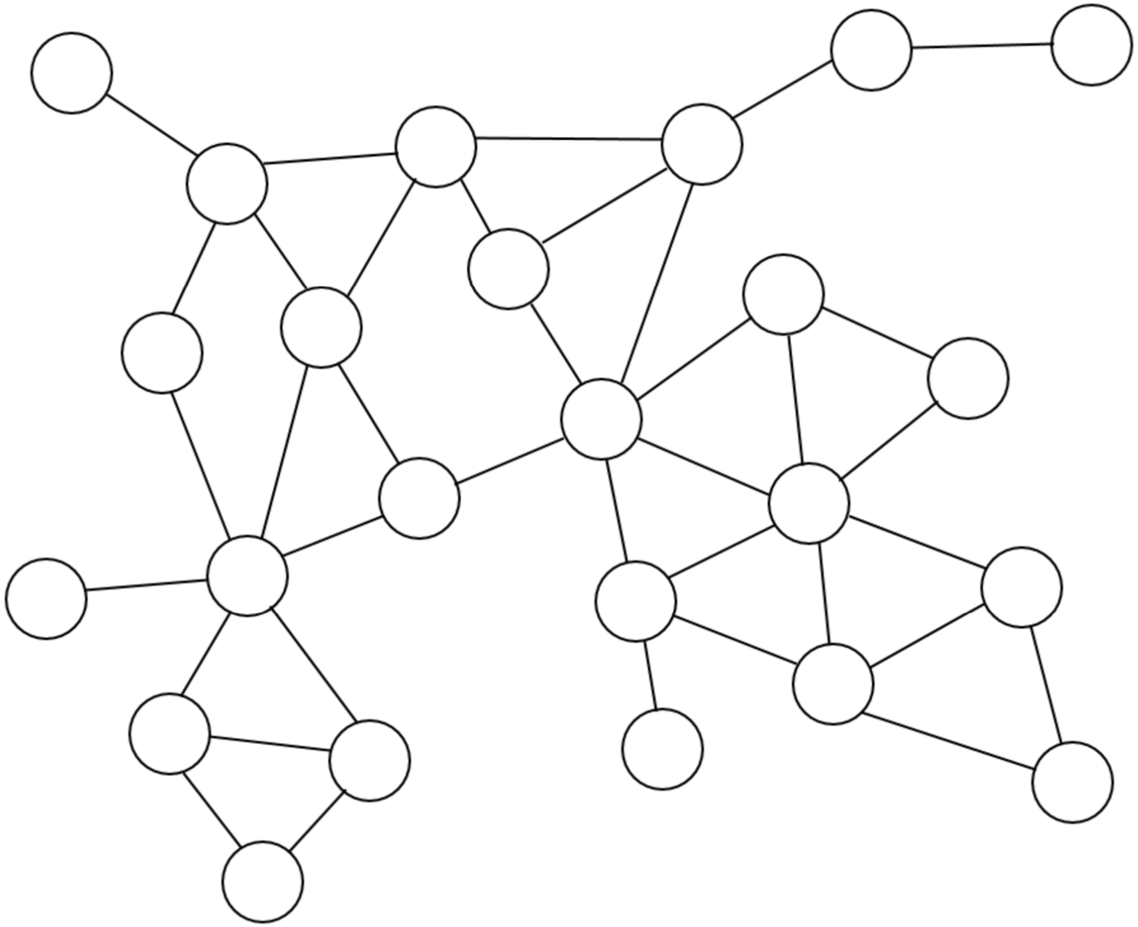
- Propagation-based Method
  - Structural Clustering Algorithm for Networks (SCAN)
- Edge-Removal
  - Girvan-Newman Algorithm (GNA)
  - Fast Newman Algorithm
- Louvain Algorithm
- Label Propagation Algorithm

Label Propagation Algorithm

- U. Nandini Raghavan, R. Albert, and S. Kumara. “**Near linear time algorithm to detect community structures in large-scale networks**” PHYSICAL REVIEW E 76, 036106, 2007. 2582 cites
- Implementations/Packages
  - https://github.com/benedekrozemberczki/LabelPropagation
  - https://networkx.github.io/documentation/stable/_modules/networkx/algorithms/community/label_propagation.html
  - https://igraph.org/r/doc/cluster_label_prop.html

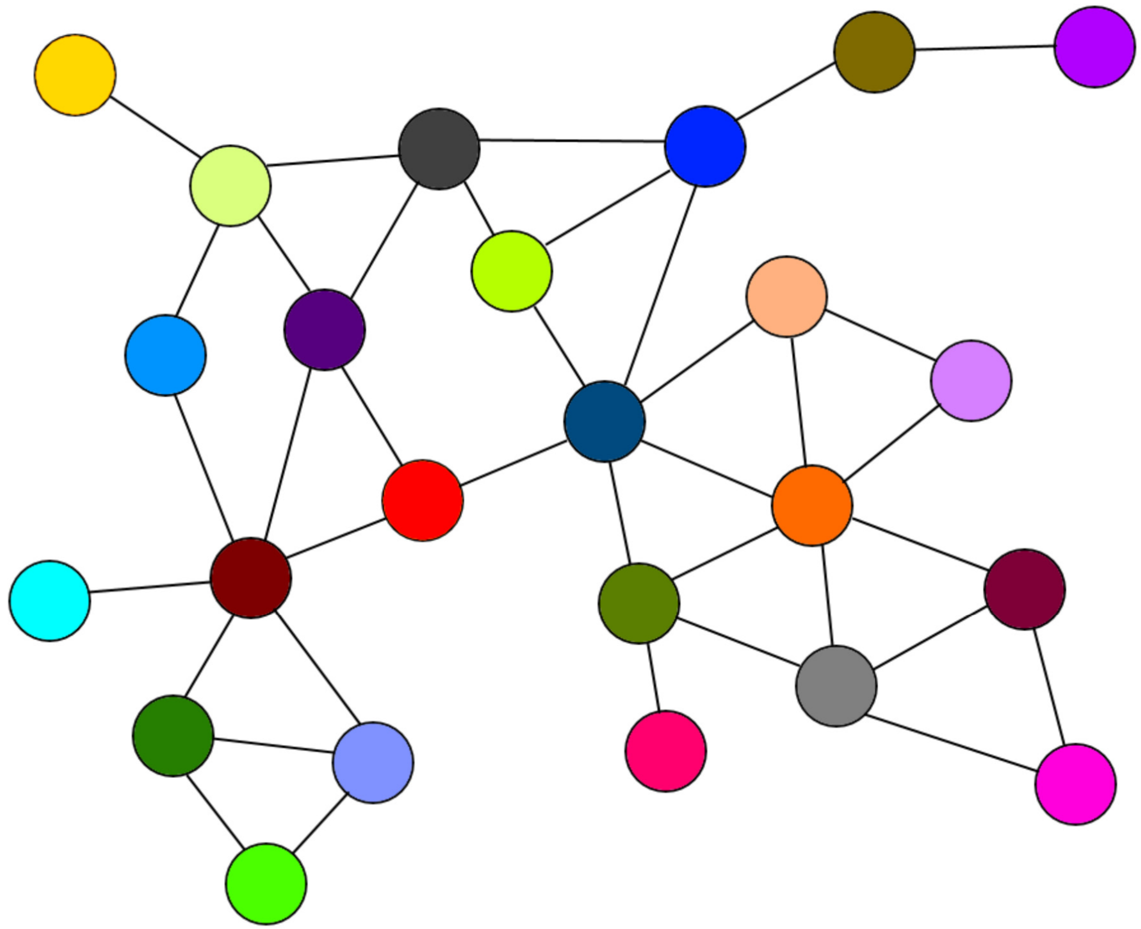
Label Propagation Algorithm (LPA)

- Step 1: Randomly label with n labels



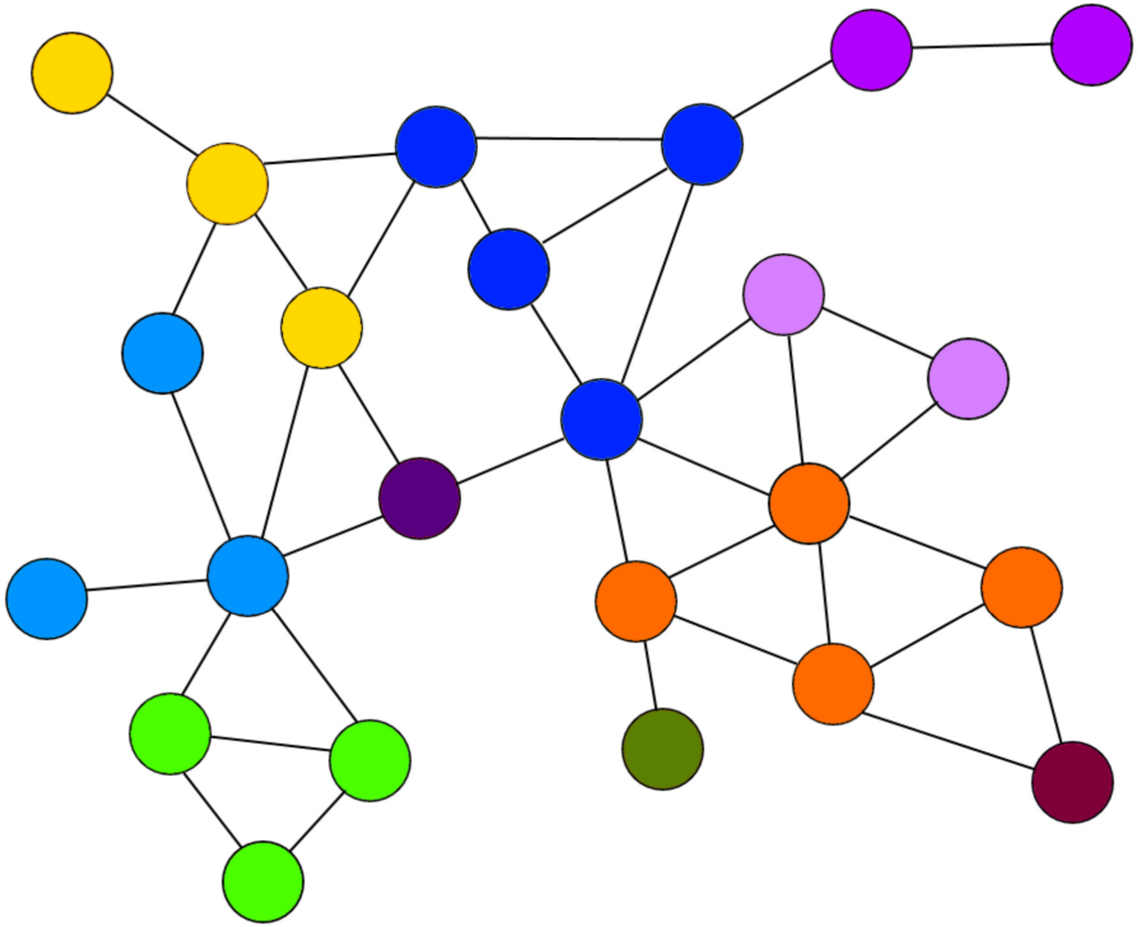
Label Propagation Algorithm (LPA)

- Step 1: Randomly label with n labels



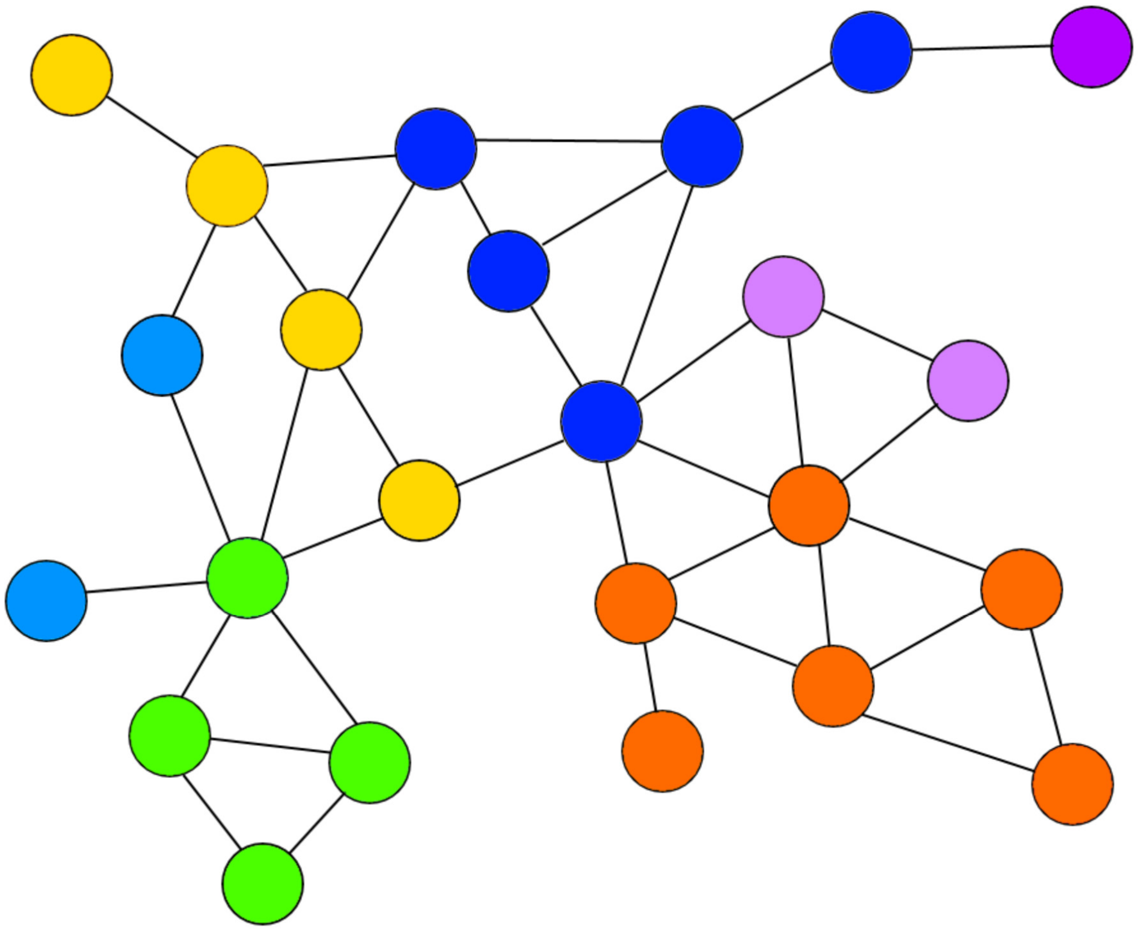
Label Propagation Algorithm (LPA)

- Step 2: Iteratively update each v with max per-label count over neighbors, ties broken randomly



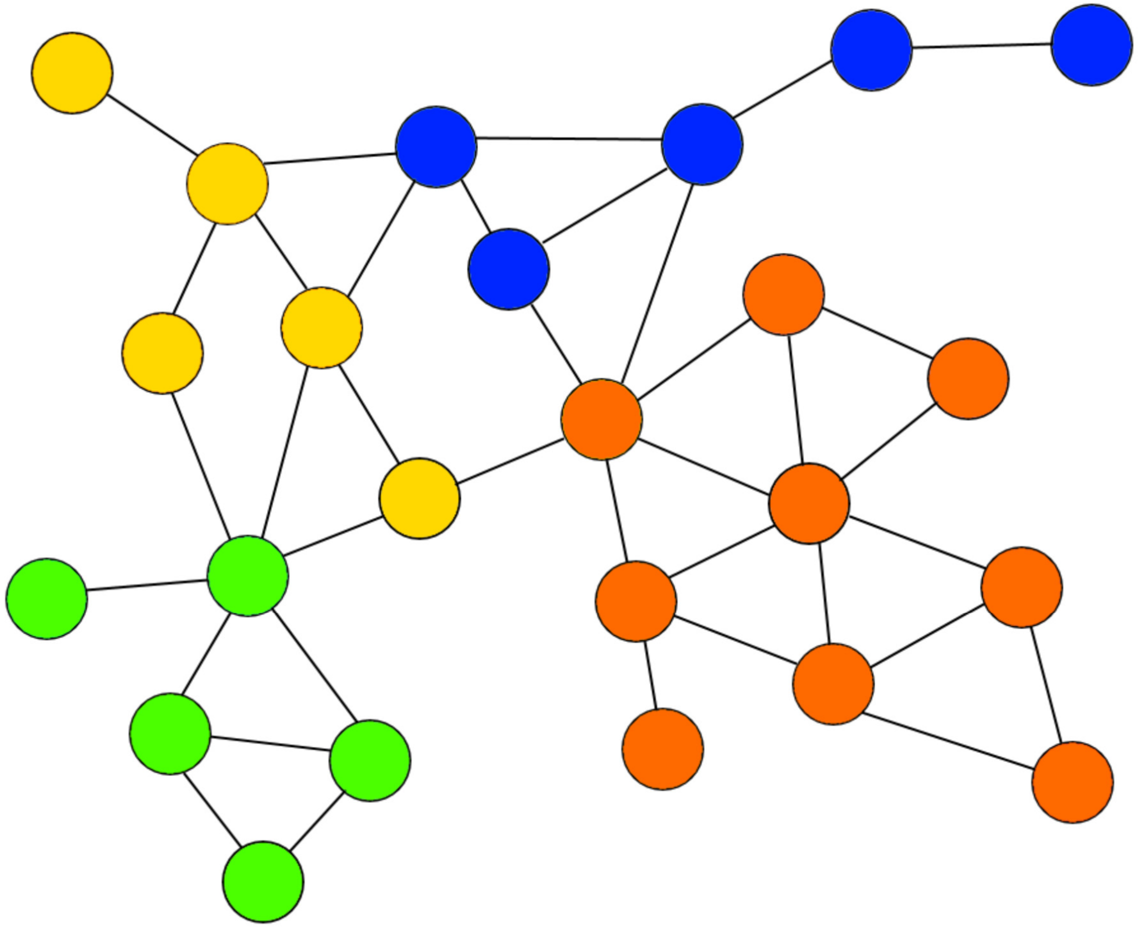
Label Propagation Algorithm (LPA)

- Step 2: Iteratively update each v with max per-label count over neighbors, ties broken randomly



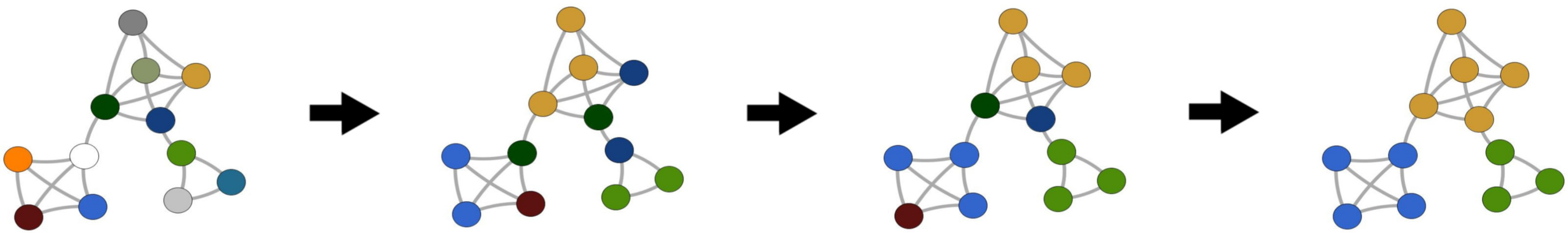
Label Propagation Algorithm (LPA)

- Step 3: Algorithm completes when no new updates possible. Otherwise, continue back to Step 2



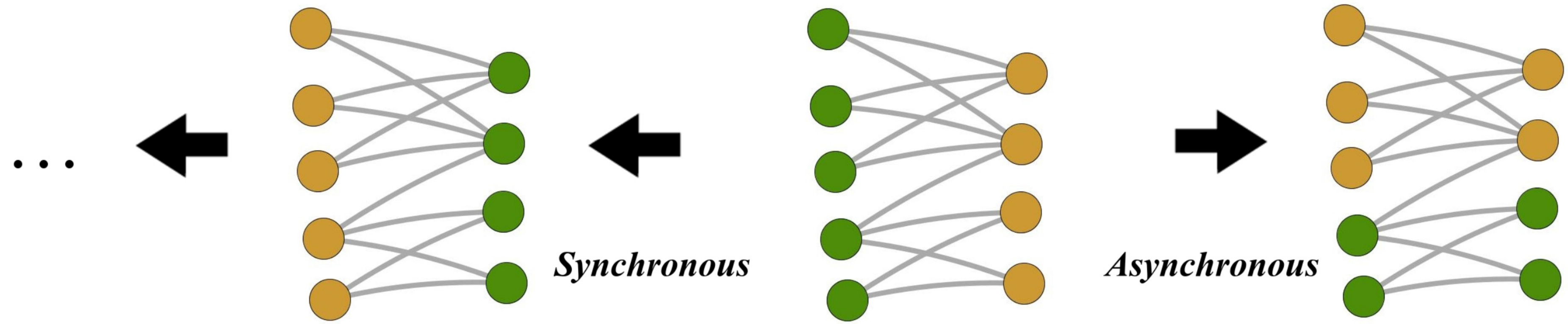
Label Propagation Algorithm (LPA)

- Given undirected graph $G = (V, E)$ with weights W (and communities $C = \emptyset$)
- Step 1: initialize nodes with unique labels $\forall v \in V: c_v = l_v$
- Step 2: set each node's label to the label shared by most of its neighbors, i.e., $\forall v \in V: c_v = \text{argmax}_l \sum_{u \in \mathcal{N}_v^l} w_{vu}$
  - Nodes are updated sequentially
  - Ties are broken uniformly at random
- Step 3: if not converged, continue to Step 2



Issues with LPA

- Oscillation of labels in, e.g., bipartite graphs
  - Nodes are updated sequentially (asynchronous) in random order



- Convergence issues for, e.g., overlapping communities
  - Node’s label is retained, when among most frequent