[Pol highlight 01]

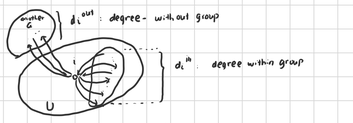
- Target problem:

  Community detection and analysis for a large-scale network
  (Finding Community structures)

- Features

  1. Previous other algorithms: (community size / number) a prior information / + network structure ⟶ community information
     optimization of a predefined objective function (prior labels)
     LPA: Only network structure ⟶ community information
  2. Almost linear time
  3. synchronous or asynchronous updating

- Conceptions: community, network partitioning problem (known community number ⟶ min split),
  community detection (unknown community number ⟶ split)




if $\forall i \in U$, $d_i^{in} > d_i^{out}$, U is a community in a weak sense

a measure that can quality ⟶ find the best
the strength of a community obtained community structures

- Procession:

  ① Initialization: every node with unique labels,
  ② Iteration: for each node, update its community based on the labels of its neighbors. choose to join the community to which the maximum number of its neighbors belong with.

[Pos]

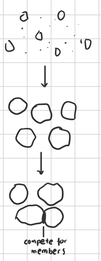
(i) Initialize the labels at all nodes in the network. For a given node $x$, $C_x(0)=x$.
↓
(ii) Set $t=1$.

} Initialization

↓

(iii) Arrange the nodes in the network in a random order and set it to $X$.
↓
(iv) For each $x \in X$ chosen in that specific order, let $C_x(t)=f(C_{x_{i1}}(t),\ldots,C_{x_{im}}(t),C_{x_{i(m+1)}}(t-1),\ldots,C_{x_{ik}}(t-1))$. $f$ here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.
↓
(v) If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t=t+1$ and go to (iii).

} pretreatment ⟵ Need more completed implement & provement □
} update (asynchronous)
} end check
} Iterations



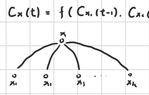

First few iteration, because initialization of unique labels for each node, result in various small pockets.

Then, these consensus groups try to acquire more nodes.

Further, when a consensus group reaches the border of another consensus group, they start to compete for members.

Finally, when there is no label update, we get the structure of communities.

$C_x(t) = f(C_{x_1}(t-1), C_{x_2}(t-1), \cdots, C_{x_k}(t-1))$



$C_x(t)$: node x's lable in t th iteration time
$f(\cdot)$: choose the lable to which maximum number of parameters belong to. ↑ which ensures for each node, its neighbors / degrees within community is not less than its degrees without.

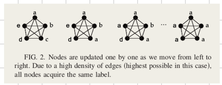


FIG. 2. Nodes are updated one by one as we move from left to right. Due to a high density of edges (highest possible in this case), all nodes acquire the same label.

Problems: (like "rank sink" and "rank leak" in PageRank...)

1. bipartite or nearly bipartite subgraphs ⟶ oscillations of labels (especially for a star graph)

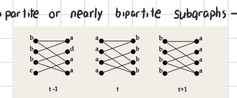
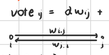



FIG. 3. An example of a bi-partite network in which the label sets of the two parts are disjoint. In this case, due to the choices made by the nodes at step $t$, the labels on the nodes oscillate between $a$ and $b$.

Solution:

Fixed Iteration Frequency? ⊗
Asynchronous updating + Random update order ✓
key!!!

Q3: In the "Label Propagation Algorithm", have a step "Arrange the nodes in the network in a random order and set it to $X$" if we do not use "random order", what will happen? ○
1. inital order may relates to the community structure, which will have effect to the algorithm's time & result efficiency and performance (How to understand?)

Q1: for undirected graph, neighbors are easy to determine, but for directed graph, neighbors are determined by in-edge / out-edge? □
(Temporay, using $vote_y = \alpha w_{ij} + \beta w_{ji}$, i's neighbor j's contribution.  , use the parameter $\alpha \cdot \beta$ for easy adjustment)

Q2: If we have two or more equal neighbors set, we should choose which lable for updating? □
[see Pol4] Randomly select? ○
(If is, How

2. Community Overlap Issues:
   a node probably belongs to two and more communities.

   Solution:
   Use "single label" strategy, which means one node only belongs to one community, which can avoid communities overlap to a certain extent.

Also link to

Q4: Label Propagation Algorithm will produce not unique solutions, so how to evaluate each solution and choose the best solution? □

[Pos highlight 02]

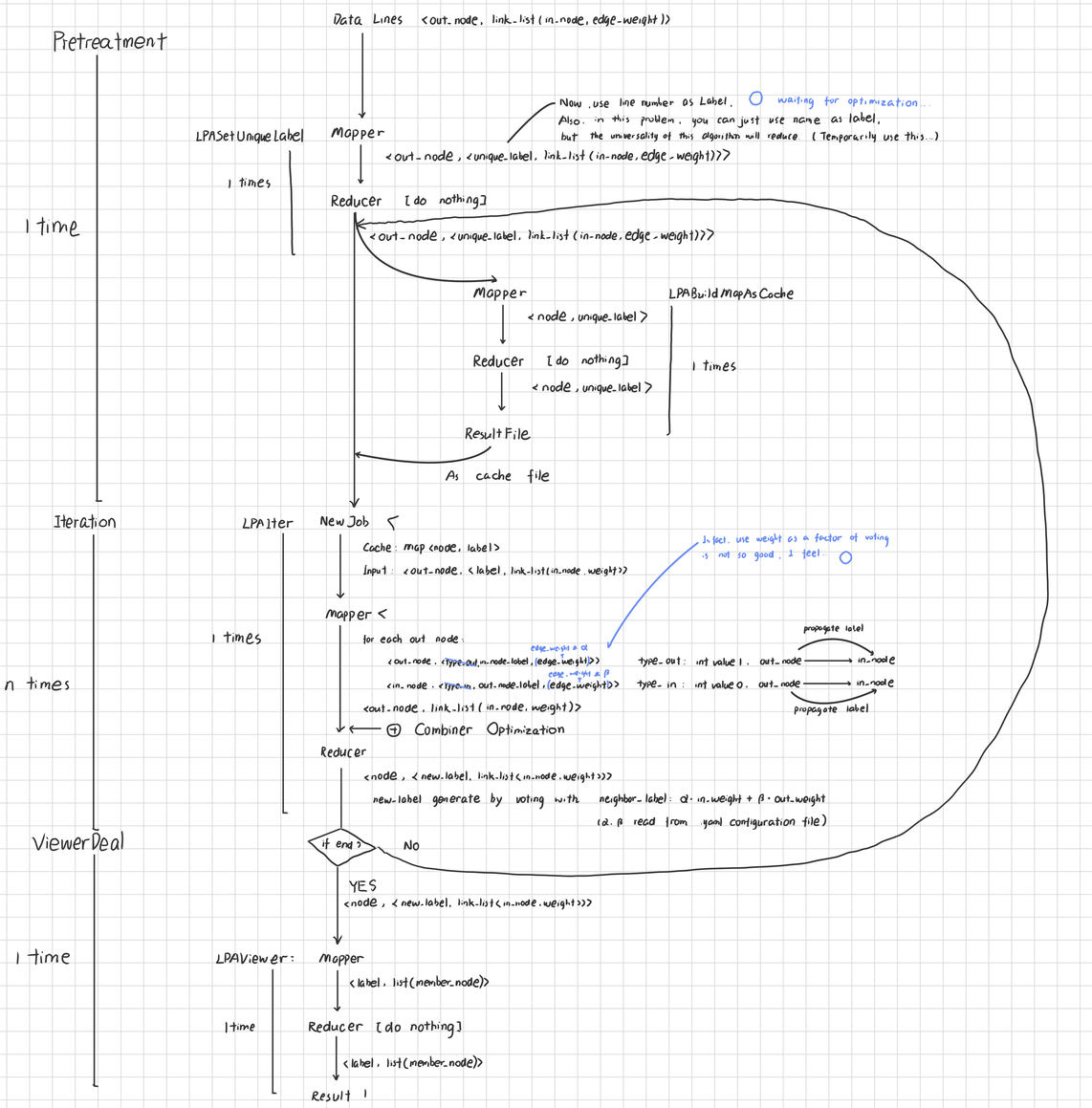
- Implement Label Propagation Algorithm by Hadoop MapReduce

  Implement Idea:

  4 Parts:
  1. LPASetUniqueLabel (Pretreatment1)
  2. LPABuildMapAsCache (Pretreatment2)
  3. LPAIter
  4. LPAViewer

  Used:
  Mapper, Reducer, Cache, Combiner

Data Lines <out_node, link_list(in_node, edge_weight))




Now, use line number as Label. ○ waiting for optimization...
Also, in this problem, you can just use name as label, but the universality of this algorithm will reduce. (Temporarily use this...)

Factually: "asynchronous updating + randomly update" is the best
Temporay NEw: Synchronous updating (update order is nonsense)

Hard Points:
(kernel)
1. Implementation of asynchronous updating is nearly impossible by Hadoop MapReduce, so only use synchronous updating □ (or maybe think about using Spark to implement synchronous updating)
2. How to implement random sorting method which is used to build a random order of data items before each iteration □ (but if not use asynchronous updating, the update order is nonsense!)
3. How to guarantee that the initialization use unique labels? Because the parallel running may lead to repeated some label. □ (maybe can solve this problem by using data item's line number in text file as the unique number, which ensures non-repeated and unique)

Q: Asynchronous Updating can run parallelly? ○
Hard, I think there is a parent relying chain, which can run one by one only.