Fast Hierarchical Graph Clustering in Linear-Time

Ryan A. Rossi
Adobe Research

Nesreen K. Ahmed
Intel Labs

Eunyee Koh
Adobe Research

Sungchul Kim
Adobe Research

ABSTRACT

While there has been a lot of research on graph clustering (community detection), most work (i) does not address the hierarchical community detection problem *or* are (ii) inefficient for large networks. In this work, we describe an approach called HLP that addresses both these limitations. Notably, HLP is fast and efficient for discovering a hierarchy of communities in large networks with a worst-case time and space complexity that is linear in the number of edges and nodes, respectively. The experiments demonstrate the effectiveness of HLP. Finally, we show an application for visualizing large networks with hundreds of thousands of nodes and edges.

1 INTRODUCTION

Communities in a graph are sets of vertices $C_1, \ldots, C_k$ such that each set C_k has more connections inside the set than outside [3]. While there are many different methods for finding communities [3, 8], it is generally agreed that a community $C_k \subseteq V$ is "good" if the induced subgraph is dense (*e.g.*, many edges between vertices in C_k) and there are relatively few edges from C_k to other vertices $\bar{C}_k = V \setminus C_k$ [8]. Semantically, communities may represent a tightly-knit group of friends, a household or organization, web pages of the same topic, or researchers that frequently publish together. In this work, we address the following problem:

Definition 1 (Hierarchical Community Detection). *Given an (un)directed graph $G = (V, E)$, the problem of hierarchical community detection is to find*

(i) a hierarchy of communities $\mathbb{H} = \{\mathcal{C}^1, \ldots, \mathcal{C}^L\}$ where $\mathcal{C}^t = \{C_1^t, \ldots, C_k^t\}$ are the communities at level t s.t. $|\mathcal{C}^{t-1}| > |\mathcal{C}^t|, \forall t$.

(ii) a hierarchy of community (super) graphs $G_1, \ldots, G_t, \ldots, G_L$ such that $G_t = (V_t, E_t)$ succinctly captures the relationships between the communities (nodes in G_t) at a lower $t-1$ level in the hierarchy where $E_t = \{(i, j) : r \in C_i^t, s \in C_j^t \wedge (r, s) \in E_{t-1} \wedge i \neq j\}$

Unlike previous work [3, 8], we focus on *hierarchical community detection* and propose an approach called HLP that is fast *and* efficient for large networks with a worst-case time and space complexity that is linear in the number of edges and nodes, respectively.

2 APPROACH

We now present our fast linear-time approach called HLP for revealing hierarchical communities in large graphs. A summary of HLP is shown in Alg. 1. The approach begins with each node belonging to its own community. Let $\Gamma_i = \{j \in V \,|\, (i, j) \in E\}$ denote the neighbors of i. For each node i, we assign it to the community $C_k \in \mathcal{C}$ with the max number of neighbors in it. More formally,

$$\underset{C_k \in \mathcal{C}}{\operatorname{argmax}} \sum_{j \in \Gamma_i} \mathbb{I}\big[\, j \in C_k \,\big] \tag{1}$$

where $\mathbb{I}\big[\, j \in C_k \,\big] = 1$ iff $j \in C_k$, and 0 otherwise. Eq. 1 can be easily modified to take into account other important aspects. This step converges when an iteration results in no further change to the community assignments. To further speedup this step, if a node has been assigned to the same community for δ consecutive rounds, we make this assignment final.

Algorithm 1 HLP

Input: a graph $G = (V, E)$
Output: hierarchical communities $\mathbb{H} = \{\mathcal{C}^1, \ldots, \mathcal{C}^L\}$

```
Set G_0 ← G to be the initial graph and t ← 0
repeat
    t ← t + 1
    C^t ← LabelProp(G_{t−1})
    G_t = (V_t, E_t) ← CreateSuperGraph(G_{t−1}, C^t)
until |V_t| < 2                          ▷ Stop when no nodes to combine
```

Algorithm 2 Create Super Graph

Input: a graph $G_{t-1} = (V_{t-1}, E_{t-1})$, communities $\mathcal{C}^t$ from G_{t-1}
Output: community (super) graph $G_t = (V_t, E_t)$ for layer t

```
V_t ← C^t = {C_1, ..., C_k} and E_t ← ∅
Let c be the community assignment vector where c_i = k if i ∈ C_k
parallel for i ∈ V_{t−1} do
    for j ∈ Γ_i do                       ▷ Neighbor of vertex i
        if c_i ≠ c_j and (c_i, c_j) ∉ E_t then
            E_t ← E_t ∪ (c_i, c_j)
```

Given a graph G_{t-1} and $\mathcal{C}^t = \{C_1^t, \ldots, C_k^t\}$, Algorithm 2 computes the community (super) graph $G_t = (V_t, E_t)$ for layer t in the community hierarchy where $V_t \leftarrow \mathcal{C}^t$ and $E_t = \{(i, j) : r \in C_i^t, s \in C_j^t \wedge (r, s) \in E_{t-1} \wedge i \neq j\}$. HLP terminates when $|V_t| < 2$.

Property 1. *Let $|E(G_t)|$ and $|V(G_t)|$ be the number of edges and nodes in G_t and $G_0 \leftarrow G$, then $|E(G_0)| > \cdots > |E(G_L)|$ and $|V(G_0)| > \cdots > |V(G_L)|$.*

Let L be the number of layers and T be the max number of iterations at any given layer. Further, let $N = |V|$ and $M = |E|$.

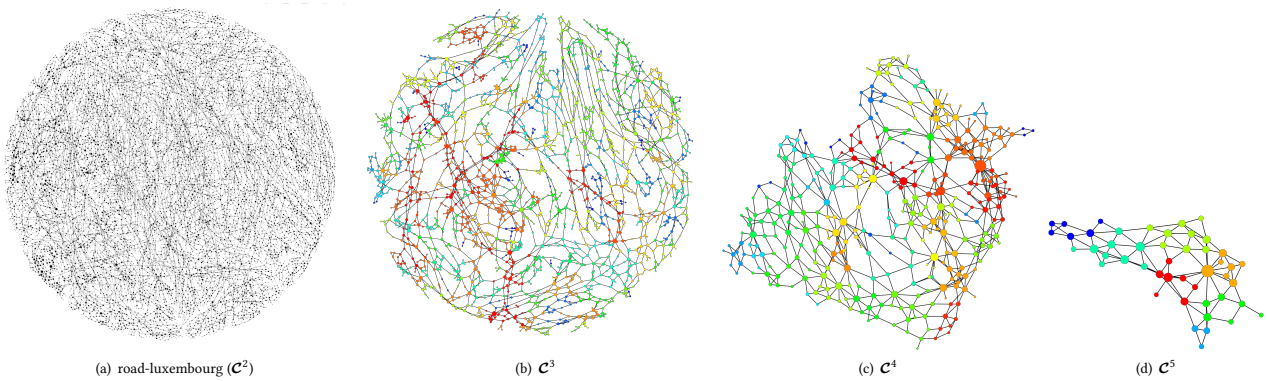
Lemma 1. *The worst-case time complexity of HLP is $O(LTM) = O(M)$ since L and T are small constants.*

A single iteration of label propagation takes $O(M)$ time to update all N nodes. Since $M \gg |E_t|$ for any $t > 0$, then the subsequent layers are much faster. The worst-case time complexity of Algorithm 2 is $O(|E_{t-1}|)$, which is bounded above by $M = |E|$.

We now provide the runtime and output *space complexity*.

WWW '20 Companion, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7024-0/20/04.
https://doi.org/10.1145/3366424.3382673



(a) road-luxembourg ($\mathcal{C}^2$) (b) $\mathcal{C}^3$ (c) $\mathcal{C}^4$ (d) $\mathcal{C}^5$

Figure 1: This case study uses road-luxembourg consisting of 114.6k nodes and 239k edges making it impossible to visualize the entire network. (a) Super graph $\mathcal{C}^2$ consisting of 9.4k supernodes (communities) with 25.3k superedges (between-community edges). (b) consists of 2,023 communities with only 6,588 between community edges whereas (c)-(d) consists of 372 and 48 communities with 1,580 and 214 between community edges, respectively. Nodes are weighted by degree. See text for discussion.

Lemma 2. *The runtime space complexity of* hLP *is* $O(L(M+N))$ *in the worst-case where* L *is a small constant, and therefore linear in the number of edges and nodes in* G.

Lemma 3. *The output space complexity of* hLP *is* $O(NL)$ *in the worst-case. Since* $L \ll N$, *it is linear in the number of nodes* N *in* G.

Since we avoid storing the node community assignments at each layer and store only how these communities are merged at each subsequent layer, all that we need are the sets $\mathcal{C}^1, \ldots, \mathcal{C}^L$ that take significantly less space than NL. Hence, instead of using $O(NL)$ space, hLP uses only $O\big(\sum_k |C_k^1| + \cdots + \sum_k |C_k^L|\big) \ll O(NL)$ space.

3 EXPERIMENTS

We investigate the quality, runtime performance, and utility of hLP for a visualization application. For data and statistics, see [7].

Table 1: Quantitative evaluation using modularity.

	DS	KCore	LP	Louv	Spec	hLP
soc-yahoo-msg	0.0003	0.0004	0.0479	0.0394	0.0005	**0.0569**
bio-gene	0.0195	0.0217	0.0315	0.0408	-0.0208	**0.0846**
ca-cora	0.0089	0.0304	0.0444	0.0608	0.0164	**0.1026**
soc-terror	0.0888	0.0892	0.0967	0.0967	0.0999	**0.1243**
inf-US-powerGrid	0.0027	0.0027	0.0061	0.0212	0.1127	**0.1242**
web-google	0.0272	0.0275	0.0429	0.0471	0.1010	**0.1122**
ca-CSphd	0.0224	0.0224	0.0234	0.0198	0.0131	**0.1201**
ca-netscience	0.0164	0.0168	0.1063	0.0561	0.1229	**0.1233**
road-luxem.	0.0629	0.0629	0.0077	0.0046	-0.1170	**0.1141**
bio-DD21	0.0865	0.0866	0.0106	0.0202	0.1241	**0.1247**

Quantitative evaluation: We compare hLP to baselines that are fast with *linear-time* complexity (with the exception of Louvain). This includes Densest Subgraph (DS) [4], KCore Communities (KCore) [9], Label Propagation (LP) [6], Louvain (Louv) [1], and Spectral Clustering (Spec) [2]. We evaluate the communities using modularity [5]. We report the best result from any method. Results using modularity are provided in Table 1. Notably, hLP outperforms all baselines across all graphs (Table 1). In particular, hLP typically achieves at least an order of magnitude improvement over the other methods. These results demonstrate the effectiveness of hLP across a wide variety of graphs.

Runtime Performance: In Figure 2, we compare the runtime of hLP for two large networks including a social network (soc-yahoo-msg) and a road network dataset (road-luxembourg). In both cases, hLP and LP perform the best followed by KCore, DS, Spectral clustering, and Louvain.

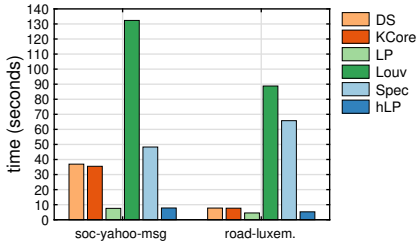


Figure 2: Runtime comparison.

Visualizing Large Networks: One important application of hLP is for visualization and exploration of large networks in an interactive fashion. In Figure 1, visualizing the entire graph would cause information overload and be extremely slow to render and explore in real-time. However, we can simply apply hLP to summarize the graph structure at multiple levels as shown in Figure 1. Thus, hLP provides a more manageable view of the key network structures from which the user can explore further by clicking on each super node that represents a lower-level community.

REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *JSTAT* 10 (2008).
[2] Fan RK Chung. 1997. *Spectral graph theory*. AMS.
[3] Santo Fortunato. 2010. Community detection in graphs. *Phy. Rep.* 3 (2010).
[4] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *ICALP*.
[5] M.E.J. Newman. 2001. The structure of scientific collaboration networks. *PNAS* 98, 2 (2001), 404.
[6] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (2007), 036106.
[7] Ryan A. Rossi and Nesreen K. Ahmed. 2016. An Interactive Data Repository with Visual Analytics. *SIGKDD Exp.* (2016). http://networkrepository.com
[8] Satu Elisa Schaeffer. 2007. Graph clustering. *Comp. sci. rev.* 1, 1 (2007), 27–64.
[9] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. CoreScope: Graph Mining Using k-Core Analysis–Patterns, Anomalies and Algorithms. In *ICDM*.