Fast Hierarchical Graph Clustering in Linear-Time
Ryan A. Rossi¹ | Nesreen K. Ahmed², Eunyee Koh¹, and Sungchul Kim¹

¹Adobe Research
²Intel Labs
Motivation

Communities are sets of densely connected nodes
- Important for many applications

Most previous work has two main limitations:
1. Most previous work does not address the hierarchical community detection problem
2. Inefficient for large graphs with a runtime that is not linear in the number of edges

This work proposes an approach called hLP that addresses both these limitations.
- Fast linear-time approach for revealing hierarchical communities in large graphs
Problem

Given $G$, hLP computes
(i) a hierarchy of communities $\mathbb{H} = \{C^1, \ldots, C^L\}$ s.t. $|C^{t-1}| > |C^t|$, $\forall t$
(ii) a hierarchy of super graphs $G_1, \ldots, G_t, \ldots, G_L$

$$V_t \leftarrow C^t$$
$$E_t = \{(i, j) : r \in C^t_i, s \in C^t_j \land (r, s) \in E_{t-1} \land i \neq j\}$$

Desired properties:
- Finds "good" high quality communities
- Fast algorithm for large graphs – linear time and space complexity
- Summarizes structure at various levels of granularity
Overview

Two main steps:
1. Label propagation
2. Super graph construction

Repeat 1-2 until convergence

---

**Algorithm 1** HLP

**Input:** a graph \( G = (V, E) \)

**Output:** hierarchical communities \( \mathbb{H} = \{\mathcal{C}^1, \ldots, \mathcal{C}^L\} \)

1. Set \( G_0 \leftarrow G \) to be the initial graph and \( t \leftarrow 0 \)
2. repeat
3. \( t \leftarrow t + 1 \)
4. \( \mathcal{C}^t \leftarrow \text{LABELPROP}(G_{t-1}) \)
5. \( G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, \mathcal{C}^t) \)
6. until \( |V_t| < 2 \) \hspace{1em} \( \triangleright \) Stop when no nodes to combine
Overview

Two main steps:
1. Label propagation
2. Super graph construction

Repeat 1-2 until convergence

Algorithm 1  HLP

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

1. Set $G_0 \leftarrow G$ to be the initial graph and $t \leftarrow 0$
2. repeat
3. \hspace{1em} $t \leftarrow t + 1$
4. \hspace{1em} $C^t \leftarrow \text{LABELPROP}(G_{t-1})$
5. \hspace{1em} $G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, C^t)$
6. until $|V_t| < 2$ \hspace{1em} » Stop when no nodes to combine
Overview

Two main steps:
1. Label propagation
2. Super graph construction

Repeat 1-2 until convergence

Algorithm 1  HLP

Input: a graph $G = (V, E)$
Output: hierarchical communities $\mathbb{H} = \{\mathcal{C}_1, \ldots, \mathcal{C}_L\}$

1. Set $G_0 \leftarrow G$ to be the initial graph and $t \leftarrow 0$
2. repeat
3. $t \leftarrow t + 1$
4. $\mathcal{C}^t \leftarrow \text{LABELPROP}(G_{t-1})$
5. $G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, \mathcal{C}^t)$
6. until $|V_t| < 2$  ▶ Stop when no nodes to combine
Overview

Two main steps:
1. Label propagation
2. Super graph construction

Repeat 1-2 until convergence

Algorithm 1  HLP

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

1. Set $G_0 \leftarrow G$ to be the initial graph and $t \leftarrow 0$
2. repeat
3. $t \leftarrow t + 1$
4. $\mathcal{C}^t \leftarrow \text{LABELPROP}(G_{t-1})$
5. $G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, \mathcal{C}^t)$
6. until $|V_t| < 2$  // Stop when no nodes to combine
Overview

Two main steps:
1. Label propagation
2. Super graph construction
Repeat 1-2 until convergence

Algorithm 1 HLP

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

1. Set $G_0 \leftarrow G$ to be the initial graph and $t \leftarrow 0$
2. repeat
   3. $t \leftarrow t + 1$
   4. $C^t \leftarrow \text{LABELPROP}(G_{t-1})$
   5. $G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, C^t)$
6. until $|V_t| < 2$ ▶ Stop when no nodes to combine
Overview

Two main steps:
1. Label propagation
2. Super graph construction

Repeat 1-2 until convergence

Time Complexity:

\[ O(LTM) \]

L = number of layers
T = number of iterations
M = number of edges

---

**Algorithm 1  HLP**

<table>
<thead>
<tr>
<th>Input:</th>
<th>a graph ( G = (V, E) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>hierarchical communities ( \mathbb{H} = {C^1, \ldots, C^L} )</td>
</tr>
</tbody>
</table>

1. Set \( G_0 \leftarrow G \) to be the initial graph and \( t \leftarrow 0 \)
2. repeat
3. \( t \leftarrow t + 1 \)
4. \( C^t \leftarrow \text{LABELPROP}(G_{t-1}) \)
5. \( G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, C^t) \)
6. until \( |V_t| < 2 \)  \[ \triangleright \text{Stop when no nodes to combine} \]
Overview

Two main steps:
1. Label propagation
2. Construct super graph

Repeat 1-2 until convergence

Space Complexity:
\[ O(L(M + N)) \]

L = number of layers
M = number of edges
N = number of nodes

---

**Algorithm 1** HLP

**Input:** a graph \( G = (V, E) \)

**Output:** hierarchical communities \( \mathbb{H} = \{ C^1, \ldots, C^L \} \)

1. Set \( G_0 \leftarrow G \) to be the initial graph and \( t \leftarrow 0 \)
2. repeat
3. \( t \leftarrow t + 1 \)
4. \( C^t \leftarrow \text{LABELPROP}(G_{t-1}) \)
5. \( G_t = (V_t, E_t) \leftarrow \text{CREATESUPERGRAPH}(G_{t-1}, C^t) \)
6. until \( |V_t| < 2 \) \( \quad \triangleright \) Stop when no nodes to combine

---

**Algorithm 2** Create Super Graph

**Input:** a graph \( G_{t-1} = (V_{t-1}, E_{t-1}) \), communities \( C^t \) from \( G_{t-1} \)

**Output:** community (super) graph \( G_t = (V_t, E_t) \) for layer \( t \)

1. \( V_t \leftarrow C^t = \{ C_1, \ldots, C_k \} \) and \( E_t \leftarrow \emptyset \)
2. Let \( \mathbf{c} \) be the community assignment vector where \( c_i = k \) if \( i \in C_k \)
3. parallel for \( i \in V_{t-1} \) do
4. \( \quad \text{for } j \in \Gamma_i \text{ do } \quad \triangleright \) Neighbor of vertex \( i \)
5. \( \quad \quad \text{if } c_i \neq c_j \text{ and } (c_i, c_j) \notin E_t \text{ then } \)
6. \( \quad \quad \quad E_t \leftarrow E_t \cup (c_i, c_j) \)
Results

- Use modularity for evaluation

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

Table 1: Quantitative evaluation using modularity.
Runtime comparison
Summary of Contributions

- Proposed a new hierarchical graph clustering algorithm
- Fast with linear time and space complexity
- Outperforms other algorithms in terms of cluster quality
- Useful for visualization and interactive exploration of large networks
Thanks for listening!