# VLP: A Label Propagation Algorithm for Community Detection in Complex Networks

Sharon Boddu, Maleq Khan, and Mais Nijim

*Department of Electrical Engineering and Computer Science*
*Texas A&M University-Kingsville*
*sharonbdd25@gmail.com, {maleq.khan, mais.nijim}@tamuk.edu*

**Abstract.** Community detection is a commonly encountered problem in social network analysis and many other areas. A community in a graph or network is a subgraph containing vertices that are closely connected to other vertices within the same subgraph but have fewer connections to the other vertices. Community detection is useful in analyzing complex systems and recognizing underlying patterns and structures that govern them. There are several algorithms that currently exist for community detection, ranging from simple and fast approaches, such as the label propagation algorithm (LPA), to more complex and time-consuming methods, such as the state-of-the-art Louvain method. We propose a new method called vector label propagation (VLP), which is a generalization of the LPA approach. The VLP algorithm significantly enhances the quality of the detected communities compared to LPA while being much faster than the Louvain method. For example, on the Twitter network, VLP has a normalized mutual information (NMI) score of 0.82, while LPA has an NMI score of 0.47. With rigorous experimentations, we demonstrate that the VLP algorithm is significantly faster than state-of-the-art algorithms such as Louvain and Infomap. On the Twitter network, VLP is 2.8 times faster than Louvain.

**Keywords:** Community detection · graph algorithms · network analysis · graph mining

## 1 Introduction

Complex networks and systems can be analyzed by representing them as graphs in which entities are denoted by vertices and the relationships between entities by edges [4]. The study and analysis of complex networks has revealed various structural properties that provide insight into the organization and behavior of such systems [4, 14, 15, 17]. Examples of such structural properties include the small world effect [4], skewed power-law degree distributions [17], scale-free phenomenon [15], and community structure [8, 14]. Community structure refers to the existence of differentiated heterogeneous modules known as communities in the underlying structure of the system [8]. A community refers to a group of vertices within a graph or network that are densely connected to each other but

only sparsely connected with vertices outside of the community [8]. The terms networks and graphs are used interchangeably in this paper.

Community detection is a problem of interest in many disciplines, including computer science, physics, sociology, and biology [7]. In many real-world networks, such as social networks and citation networks, communities form spontaneously and reflect the underlying functional or organizational structure of the system being analyzed [14]. For example, in social networks, communities can represent real-world social circles, such as friends, family, or colleagues [17]. Identifying and analyzing communities in real-world graphs can provide valuable insights into the underlying phenomena and help to deliver better solutions to complex problems. For example, in VLSI circuits, community detection can be used to group the functional units of a circuit, optimize the layout, and minimize the required chip area [7].

In recent years, many community detection algorithms have been proposed [2–5, 9, 11, 12, 14, 15]. Raghavan et al. [19] proposed a novel algorithm called the Label Propagation Algorithm (LPA), which has since become a popular method for community detection in complex networks. LPA works by iteratively spreading labels among vertices until a consensus is reached [19].

We propose a generalization of the label propagation method called Vector Label Propagation (VLP). VLP identifies communities by maintaining a vector of candidate community labels of length $k$ for each vertex in the graph. Each vertex's label vector is updated based on the label vectors of its neighbors. To assess VLP's performance, we conducted extensive experiments on various real-world networks. To evaluate the quality of the detected communities, we use two goodness measures: modularity and normalized mutual information (NMI). VLP improves the quality of detected communities compared to LPA and addresses known issues with LPA, such as solution instability and label oscillations [19]. Furthermore, our experiments show that VLP is faster than state-of-the-art methods, such as Louvain [2] and Infomap [5], while detecting communities of comparable quality. For example, VLP is 3.2 times faster than Louvain on the Orkut graph. Furthermore, VLP shows a slightly improved NMI score of 0.60 compared to the Louvain method's score of 0.58.

The remainder of the paper is organized as follows. In Section 2, we discuss preliminary concepts and related work. We present our VLP algorithm in Section 3. The experimental results are given in Section 4, and we conclude in Section 5.

## 2 Preliminaries and Related Work

In this section, we describe some of the notation and preliminary concepts used in this paper. A graph $G = (V, E)$ consists of a set of vertices or nodes $V$ and a set of edges $E \subseteq V \times V$. The number of vertices and edges is denoted by $n = |V|$ and $m = |E|$, respectively. We assume that all edges are undirected. If $(u, v) \in E$, $u$ and $v$ are called neighbors. $N(v)$ denotes the set of neighbors of vertex $v$; that is, $N(v) = \{u : (u, v) \in E\}$. The degree of $v$, denoted by $d(v)$, is

given by $d(v) = |N(v)|$. A community $C \subseteq V$ refers to a group of vertices in a network that has a relatively high number of internal edges within the vertices in $C$ and relatively fewer external edges to the vertices in the rest of the graph. Community detection involves a process of identifying a community partition or set of communities $P = \{C_1, C_2, \ldots, C_k\}$ in a graph $G(V, E)$, where each $C_i \subseteq V$. If the communities $C_1, C_2, \ldots, C_k$ are disjoint, they are called non-overlapping communities. Otherwise, they are called overlapping communities.

When the true community structure (also called ground truth) is known in advance, solutions of community detection algorithms can be evaluated by comparing the detected communities against the ground truth communities. Mutual information (MI) is an information-theoretic measure that quantifies the correlation between two sets of communities [7]. However, MI has limitations when differentiating communities derived from hierarchical splitting. Normalized mutual information (NMI) overcomes this issue by normalizing the MI score using the partitions' entropy [7]. NMI is widely used to evaluate the performance of community detection algorithms when the ground truth is known [7].

In the absence of known ground truth, detected communities can be evaluated using quality scores, such as modularity [14], which measure the strength of the community structure [7]. Modularity is defined as the deviation of the internal edges from the expected number of edges in a random graph [7, 14]. Higher modularity scores indicate a better community structure, as vertices are more densely connected within the community and sparsely connected to the rest of the network [14].

Community detection algorithms can be broadly categorized into partitioning-based, clustering-based, optimization-based, and label-propagation-based algorithms. Partitioning-based community detection algorithms group the vertices of a graph into some $k$ groups of nearly equal size and a small number of edges between the groups. Popular approaches include bisection-based algorithms [15] and edge removal-based algorithms [16]. Clustering-based community detection algorithms are often stochastic in nature [11], and examples include Infomap, WalkTrap, and spectral clustering [5,11,15]. Optimization-based community detection algorithms seek to identify communities by maximizing or minimizing some global quality score, such as modularity. Modularity maximization produces communities of high quality by repeatedly maximizing the gain in modularity [2,3,14,15]. Louvain [2] is a state-of-the-art modularity-based community detection algorithm [11] and is widely adopted due to its speed and good community quality. However, modularity-based algorithms are not effective in detecting smaller communities [6].

**Label Propagation Algorithm**: Raghavan et al. [19] proposed a community detection algorithm called the label propagation algorithm (LPA) (Algorithm 1) as described below. Each vertex $v$ is assigned a label $L(v)$, and these labels are updated iteratively so that, at the end, all vertices in the same community have the same label. We have the initial labels $L(v) = v$ for all $v \in V$. In each iteration, the label $L(v)$ of each vertex $v$ is updated to a new label that is the most frequent label among neighbors $N(v)$. Let $\mathrm{NL}(v)$ be a multiset contain-

**Algorithm 1** Label Propagation Algorithm

```
1: for all v ∈ V do
2:     L(v) ← v
3: Converged ← false
4: while not Converged do
5:     Converged ← true
6:     for all v ∈ V do
7:         NL(v) ← {L(v)}
8:         for all u ∈ N(v) do
9:             NL(v) ← NL(v) ∪ {L(u)}
10:        μ(v) ← mode(NL(v))
11:    for all u ∈ N(v) do
12:        if μ(v) ≠ L(v) then
13:            Converged ← false
14:            L(v) ← μ(v)
15: end while
```

ing the labels of $v$ and its neighbors; that is $\mathrm{NL}(v) = \{L(v)\} \cup \{L(u) | u \in N(v)\}$. Then the new label of $v$ is simply the most frequent element in $\mathrm{NL}(v)$, that is, $mode(\mathrm{NL}(v))$. When there is a tie for the most frequent label, one of them is chosen arbitrarily. After some iterations, the algorithm converges, and no vertices change their labels from the previous iteration. At this point, the algorithm is terminated, and we say that the community labels have stabilized or that the consensus has been reached.

## 3 Vector Label Propagation

In this section, we present our community detection algorithm called vector label propagation (VLP), which is a generalization of the label propagation algorithm (LPA) described in Section 2. Our experiments show that VLP significantly improves the quality of communities compared to LPA. VLP is also much faster than state-of-the-art algorithms such as Louvain [2] and Infomap [5], and at the same time, it detects communities with quality comparable to that of the Louvain and Infomap algorithms.

### 3.1 Overview of VLP

Vector label propagation (VLP) uses the label propagation technique to propagate the most dominant labels selected from the neighboring vertices. To enhance the community detection ability of the algorithm, we generalize this technique by maintaining $k$ candidate community labels for each vertex, where $k$ is a parameter of the algorithm. The candidate labels are the $k$ most dominant or frequent labels in the local neighborhood. On the other hand, LPA tracks only the most frequent label for each vertex. Our experimental results (given in Section 4) show that it is enough to use a small value of $k$, such as $k = 3$, to significantly improve the quality of the detected communities compared to LPA. VLP stores the candidate labels in label vectors, which are vectors consisting of pairs as follows: $\mathbb{L} = [(\ell_0, c_0), (\ell_1, c_1), \ldots, (\ell_s, c_s)]$, where $s$ is the size or length of the vector. The $i$-th pair $(\ell_i, c_i)$ consists of a candidate label $\ell_i = \mathbb{L}.\ell_i$ and a label confidence $c_i = \mathbb{L}.c_i$. The confidence $c_i$ represents the count or frequency of $\ell_i$. VLP maps each vertex $v$ to a label vector $\mathbb{L}(v)$ of length $k$ with each of the $k$

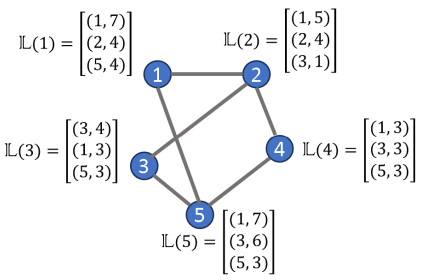


**Fig. 1.** A graph with five vertices showing label vectors of length $k = 3$

$$\begin{bmatrix}(1,7)\\(2,4)\\(5,4)\end{bmatrix} \oplus \begin{bmatrix}(3,4)\\(1,3)\\(5,3)\end{bmatrix} = \begin{bmatrix}(1,10)\\(5,8)\\(2,4)\\(3,4)\end{bmatrix}$$

**Fig. 2.** Label-wise sum

$$\mathbb{Z}(5) = \begin{bmatrix}(1,7)\\(3,6)\\(5,3)\end{bmatrix} \oplus \begin{bmatrix}(1,7)\\(2,4)\\(5,4)\end{bmatrix} \oplus \begin{bmatrix}(3,4)\\(1,3)\\(5,3)\end{bmatrix} \oplus \begin{bmatrix}(1,3)\\(3,3)\\(5,3)\end{bmatrix}$$

$$\mathbb{Z}(5) = \begin{bmatrix}(1,20)\\(2,4)\\(3,13)\\(4,0)\\(5,13)\end{bmatrix} \quad \begin{matrix}sorted\\ \mathbb{Z}(5) =\end{matrix} \quad \begin{bmatrix}(1,20)\\(3,13)\\(5,13)\\(2,4)\\(4,0)\end{bmatrix} \quad \mathbb{L}(5) = \begin{bmatrix}(1,20)\\(3,13)\\(5,13)\end{bmatrix}$$

**Fig. 3.** Updating label vector of vertex 5

elements in the vector corresponding to one candidate label. The $k$ pairs in $\mathbb{L}(v)$ are sorted by confidence in descending order. At the end of the algorithm, a vertex $v$ is assigned to the label $L(v) = \mathbb{L}(v).\ell_0$. The label vectors of the vertices in an example graph are shown in Fig. 1. The VLP algorithm initializes the label vectors $\mathbb{L}(v)$ for each vertex $v$ as follows: $\ell_0 \leftarrow v$, $c_0 \leftarrow 1$, and for $1 \leq i < k$, $\ell_i \leftarrow$ NULL, and $c_i \leftarrow 0$. Similarly to LPA, the algorithm proceeds in multiple iterations. In each iteration, the label vector $\mathbb{L}(v)$ of each vertex $v$ is updated using the following three steps.

1. Aggregation: The label vectors $\mathbb{L}(u)$ of neighbors $u \in N(v)$, along with $\mathbb{L}(v)$, are aggregated into a single label vector $\mathbb{Z}$ using a label-wise sum, denoted by $\oplus$, in which the confidences corresponding to the same labels are added together. The details of the aggregation step are discussed in Section 3.2.
2. Sorting: The aggregated label vector $\mathbb{Z}$ is sorted by confidence values $c_i$ in descending order.
3. Top $k$ selection: The label vector $\mathbb{L}(v)$ is updated to the first $k$ pairs in $\mathbb{Z}$, which are the $k$ most frequent candidate labels.

After some iterations of the VLP algorithm, no vertices change their most frequent candidate label $\mathbb{L}(v).\ell_0$ from the previous iteration. At this point, the most frequent label $\mathbb{L}(v).\ell_0$ is chosen to be the final community label $L(v)$, and the algorithm is terminated.

### 3.2 Aggregating Label Vectors

The aggregation of the label vectors is performed using a *label-wise sum*. A *label-wise sum*, denoted by $\oplus$, of two label vectors $\mathbb{X}$ and $\mathbb{Y}$ is another label vector $\mathbb{Z} = \mathbb{X} \oplus \mathbb{Y}$. The label vector $\mathbb{Z}$ accumulates the total confidences of all distinct labels that appear in $\mathbb{X}$ and/or $\mathbb{Y}$ as follows:

- If $\ell$ appears as a pair $(\ell, c')$ in only one of $\mathbb{X}$ and $\mathbb{Y}$, then we have the pair $(\ell, c) = (\ell, c')$ in $\mathbb{Z}$.
- If $\ell$ appears as pairs $(\ell, c')$ in $\mathbb{X}$ and $(\ell, c'')$ in $\mathbb{Y}$, then we have the pair $(\ell, c) = (\ell, c' + c'')$ in $\mathbb{Z}$.

**Algorithm 2** Vector Label Propagation VLP($k$, $G$)

```
1: for all v ∈ V do
2:     L(v).ℓ_0 ← v
3:     L(v).c_0 ← 1
4:     for all i = 1, 2, ...k − 1 do
5:         L(v).ℓ_i ← NULL
6:         L(v).c_i ← 0
7: Converged ← false
8: while not Converged do
9:     Converged ← true
10:    for all v ∈ V do
11:        Z(v) ← L(v)
12:        for all u ∈ N(v) do
13:            Z(v) ← Z(v) ⊕ L(u)
14:        sort(Z(v))
15:        if L(v).ℓ_0 ≠ Z(v).ℓ_0 then
16:            Converged ← false
17:    for all v ∈ V do
18:        for all i = 1, 2, . . . , k − 1 do
19:            L(v).ℓ_i ← Z(v).ℓ_i
20:            L(v).c_i ← Z(v).c_i
21:    for all v ∈ V do
22:        L(v) = L(v).ℓ_0
23: end while
```

An example of label-wise sum is shown in Fig. 2. For each vertex $v$, label vectors of its neighbors are aggregated using the label-wise sum as follows: $\mathbb{Z} = \mathbb{L}(v) \oplus \mathbb{L}(u_1) \oplus \mathbb{L}(u_2) \oplus \ldots \oplus \mathbb{L}(u_{d(v)})$, where $u_1, u_2, \ldots u_{d(v)}$ are neighbors of $v$. For example, consider the vertex 5 in Fig. 1, which has neighbors 1, 3, and 4. Fig. 3 illustrates how vertex 5 updates its label vector. The label vectors $\mathbb{L}(1)$, $\mathbb{L}(3)$, and $\mathbb{L}(4)$ of the neighbors of the vertex 5 and its own label vector $\mathbb{L}(5)$ are aggregated into $\mathbb{Z}(5) = \mathbb{L}(5) \oplus \mathbb{L}(1) \oplus \mathbb{L}(3) \oplus \mathbb{L}(4)$. $\mathbb{Z}(5)$ has 5 pairs, which are sorted by confidence values. The updated label vector $\mathbb{L}(5)$ is set to the most frequent $k = 3$ labels in $\mathbb{Z}(5)$.

### 3.3 Pseudocode of the VLP Algorithm

The pseudocode of the VLP algorithm is given in Algorithm 2. The label vectors are initialized in Lines 1-6 and iteratively updated in the while loop in Lines 8-23. The label vectors of the neighbors are aggregated in Lines 11-13. The aggregated label vector $\mathbb{Z}(v)$ is sorted in Line 14. The first $k$ pairs, corresponding to the most frequent $k$ labels, in $\mathbb{Z}(v)$ are selected as the new label vector $\mathbb{L}(v)$ in Lines 18-20. In the end (Lines 21-22), the most frequent label $\mathbb{L}(v).\ell_0$ is chosen as the final label $L(v)$. We implemented our algorithm in C++ and used *std::unordered_map* of the Standard Template Library (STL) to implement the label vectors. This allows us to perform label-wise sum efficiently using *std::accumulate*.

### 3.4 Weighted Aggregation of Label Vectors

We enhance label vector aggregation by giving more importance to label vectors from vertices with more common neighbors, in a mechanism called *weighted aggregation*. When a vertex $v$ aggregates the label vectors of neighbors, we assign a weight $\lambda = J(u, v)$ to neighbors $u \in N(v)$ corresponding to the Jaccard similarity index given by $J(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$. In weighted label vector aggregation, the label confidences in vectors $\mathbb{L}(u)$ of the neighbors $u \in N(v)$ are scaled up or down by a factor of $\lambda$ as follows. If $\mathbb{L}(u) = [(\ell_0, c_0), (\ell_1, c_1), \ldots, (\ell_{k-1}, c_{k-1})]$,

then $\lambda\mathbb{L}(u) = [(\ell_0, \lambda c_0), (\ell_1, \lambda c_1), \ldots, (\ell_{k-1}, \lambda c_{k-1})]$. Now, $\mathbb{Z} \oplus \lambda\mathbb{L}(u)$ is computed by performing a label-wise sum on two label vectors $\mathbb{Z}$ and $\lambda\mathbb{L}(u)$. To use the weighted aggregation mechanism, we replace Line 13 in Algorithm 2 with $\mathbb{Z} \leftarrow \mathbb{Z} \oplus \lambda\mathbb{L}(u)$, where $\lambda = J(u, v)$ is the Jaccard index of the edge. Note that the Jaccard index weight of the edges needs to be calculated only once at the beginning of the algorithm, as they depend only on the graph's topological structure. The weight also serves as a damping factor for confidence values, which can be useful when the algorithm goes through too many iterations or the confidence values become too high. The experimental results show that the weighted label vector aggregation detects better quality communities than the unweighted aggregation.

## 4 Experimental Results

In this section, we rigorously evaluate the performance of our VLP algorithm experimentally on a wide variety of real-world graphs. Our experiments test the runtime and the quality (measured in NMI) of the detected communities. The performance of our algorithm is compared to recent state-of-the-art algorithms [2, 5, 13, 15, 19]. The experimental setup and the nature of the graph datasets used to evaluate the VLP algorithm are detailed below, along with the choice of parameters used in the algorithm.

### 4.1 Experimental Setup

We conducted our experiments on the Bridges-2 supercomputer at the Pittsburg Supercomputing Center (PSC), which was available to us via the Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) [18]. On the Bridges-2 system, we used the Regular Memory (RM) nodes having 2xAMD EPYC 7742 CPUs (2.25-3.40GHz and 256MB L3 cache) with 64 cores each and 256GB RAM on each node. Our implementations were made in the C++ language and compiled using gcc version 8.2.0.

### 4.2 Dataset

Our graph data set consists of undirected, unweighted graphs with known ground truth communities. The data set is chosen to represent a wide variety of graph types and graph sizes. These networks are taken from the Stanford Large Network Dataset Collection [10] and the GAP Benchmark Suite [1]. Table 1 lists the graphs used in our experiments and the number of vertices, the number of edges, and the average degree of the graphs.

### 4.3 Performance

In order to determine the best value of parameter $k$ for Algorithm 2, a comparative analysis of the algorithm runtime and the quality score of the identified

**Table 1.** List of graphs used in experiments.

| Graph name | Nodes | Edges | Avg-Deg |
|---|---|---|---|
| Karate | 34 | 78 | 4.6 |
| Dolphins | 62 | 159 | 5.1 |
| Football | 115 | 613 | 10.7 |
| Polbooks | 105 | 441 | 8.4 |
| web-Google | 0.9M | 5.1M | 11.3 |
| web-BerkStan | 0.7M | 7.6M | 21.7 |

| Graph name | Nodes | Edges | Avg-Deg |
|---|---|---|---|
| soc-LiveJournal1 | 4.8M | 68.8M | 28.6 |
| com-Orkut | 3.1M | 117.2M | 75.6 |
| twitter | 61.6M | 1.4B | 61.7 |
| com-FriendSter | 65.6M | 1.8B | 55.3 |
| uk2002 | 39.5M | 1.9B | 45.5 |

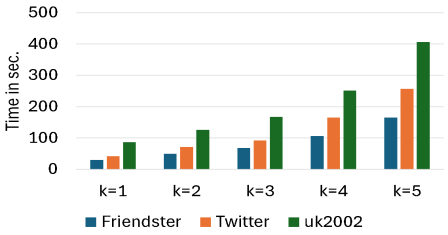


**Fig. 4.** Effect of parameter $k$ on the runtime of the algorithm in the Friendster, Twitter, and uk2002 networks.

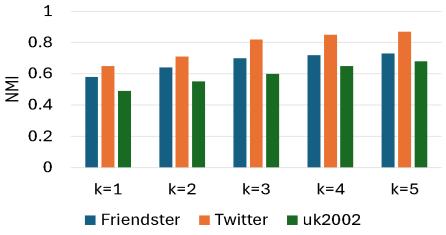


**Fig. 5.** Effect of parameter $k$ on NMI scores of detected communities in the Friendster, Twitter, and uk2002 networks.

communities is performed for various values of $k$. The runtimes (in seconds) for our VLP algorithm on three graphs - Friendster, Twitter, and uk2002 - are presented in Fig. 4. The normalized mutual information score (NMI) is used to compare the detected communities with the ground truth, and the results are presented in Fig. 5. The objective is to have an algorithm that has a fast runtime and a high NMI score. After careful consideration, $k = 3$ is chosen as it demonstrated a satisfactory quality score while still being reasonably fast. $k = 3$ is used for all subsequent experiments.

To find the effect of weighted aggregation, we compare VLP with and without weighted label vector aggregation. Fig. 6 shows the quality of the detected communities, measured using NMI, on various graphs. Using weighted label vector aggregation shows a consistent improvement in the quality of the detected communities. For example, on the Orkut graph, using weighted aggregation improves the NMI score from 0.456 to 0.603. In the rest of our experiments, we use VLP with weighted aggregation.

We compare the performance of VLP using weighted label vector aggregation with the following state-of-the-art algorithms: *i*) Label Propagation (LPA) [19], *ii*) LPAm+ [13], *iii*) Infomap [5], *iv*) ICSD+ [15], and *v*) Louvain [2]. In Fig. 7, we compare the NMI scores on four small networks - Karate, Dolphins, Football, and Polbooks - which have widely been referenced in the community detection literature. Fig. 8 compares the NMI scores for seven medium to large networks. VLP showed an average improvement of 1.52 times in NMI scores over LPA. Moreover, compared to the algorithm with the best NMI score for each network, VLP was only worse by 0.072 NMI points on average. Our VLP algorithm out-

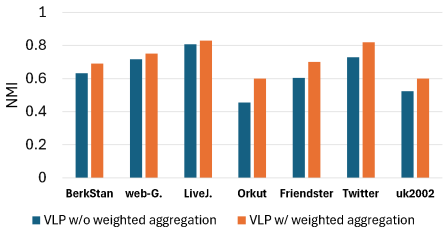


**Fig. 6.** Effect of using weighted aggregation on NMI scores of detected communities in various networks.

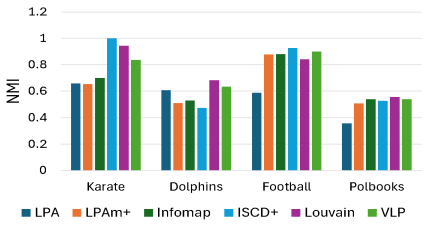


**Fig. 7.** NMI scores of detected communities in four small networks.

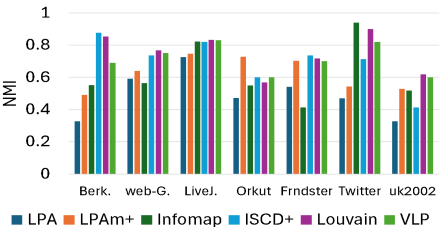


**Fig. 8.** NMI scores for communities detected in various medium and large networks by different algorithms.

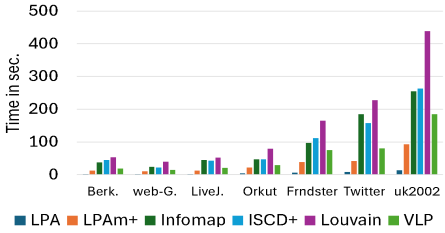


**Fig. 9.** The runtime of various algorithms, including our VLP algorithm, on various medium and large networks.

performs LPA by a wide margin and performs comparably to state-of-the-art algorithms.

Fig. 9 illustrates the runtime comparison of VLP with several state-of-the-art algorithms on seven medium to large networks. VLP outperformed Louvain and Infomap on all graphs and was considerably faster. For example, on large graphs such as Friendster, Twitter, and UK2002, VLP was 2.2 times, 2.8 times, and 2.4 times faster than Louvain and 1.3 times, 2.3 times, and 1.4 times faster than Infomap, respectively. In summary, the experimental results demonstrate that the proposed VLP algorithm is significantly faster than the state-of-the-art algorithms while maintaining comparable quality of the communities.

## 5 Conclusion

We have introduced a community detection algorithm called vector label propagation (VLP), which is a generalization of LPA. VLP detects communities with higher quality than LPA and is much faster than other state-of-the-art algorithms like Louvain. It offers efficient and high-quality community detection and is useful for analyzing complex networks in various fields. Our VLP algorithm will greatly benefit analysts and researchers and enable them to quickly find communities of good quality in real-world networks.

## Acknowledgements

The work was partially supported by NSF BIGDATA grant IIS-1633028 and used PSC Bridges-2 RM at Pittsburg Supercomuting Center (PSC) through allocation CIS230052 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program [18], which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296.

## References

1. Beamer, S., Asanović, K., Patterson, D.: The gap benchmark suite. arXiv preprint arXiv:1508.03619 (2015)
2. Blondel, V., Guillaume, J., Lambiotte, R.: Fast unfolding of communities in large networks. Jour. of stat. mech.: theory and experiment **2008**(10), P10008 (2008)
3. Clauset, A.: Finding local community structure in networks. Phy. review E **72**(2), 026132 (2005)
4. Deweese, K., Gilbert, J., Lugowski, A., Reinhardt, S., Kepner, J.: Graph clustering in sparql. In: SIAM Workshop on Network Sci. vol. 34, pp. 930–941. Citeseer (2013)
5. Edler, D., Guedes, T., Zizka, A., Rosvall, M., Antonelli, A.: Infomap bioregions: Interactive mapping of biogeographical regions from species distributions. Systematic biology **66**(2), 197–204 (2017)
6. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proc. of the national academy of sci. **104**(1), 36–41 (2007)
7. Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics reports **659**, 1–44 (2016)
8. Girvan, M., Newman, M.: Community structure in social and biological networks. Proc. of the national academy of sci. **99**(12), 7821–7826 (2002)
9. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physical review E **80**(5), 056117 (2009)
10. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
11. Leskovec, J., Lang, K.: Empirical comparison of algorithms for network community detection. In: Proc. of the 19th int. conf. on WWW. pp. 631–640 (2010)
12. Leung, I., Hui, P., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. Physical Review E **79**(6), 066107 (2009)
13. Liu, X., Murata, T.: Advanced modularity-specialized label propagation for detecting communities. Phys. A: Stat. Mech. and its Apps. **389**(7), 1493–1500 (2010)
14. Newman, M.: Modularity and community structure in networks. Proc. of the national academy of sci. **103**(23), 8577–8582 (2006)
15. Newman, M.: Spectral methods for community detection and graph partitioning. Phys. Review E **88**(4), 042822 (2013)
16. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Phys. review E **69**(2), 026113 (2004)
17. Papadopoulos, S., Kompatsiaris, Y., Vakali, A.: Community detection in social media: Performance and applications. Data min. and know. dis. **24**, 515–554 (2012)
18. Parashar, M., Friedlander, A., Gianchandani, E., Martonosi, M.: Transforming science through cyberinfra. Communications of the ACM **65**(8), 30–32 (2022)
19. Raghavan, U., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. review E **76**(3), 036106 (2007)