Stad: Stateful Diffusion for Linear Time Community Detection

Amira Soliman[1*], Fatemeh Rahimian[*] and Sarunas Girdzijauskas[*,†]

[*]*Swedish Institute of Computer Science, RISE SICS, Stockholm, Sweden*
[†]*School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden*
Email: [*]*{amira.soliman.el.hosary, fatemeh.rahimian}@ri.se,* [†]*sarunasg@kth.se*

***Abstract*—Community detection is one of the preeminent topics in network analysis. Communities in real-world networks vary in their characteristics, such as their internal cohesion and size. Despite a large variety of methods proposed to detect communities so far, most of existing approaches fall into the category of global approaches. Specifically, these global approaches adapt their detection model focusing on approximating the global structure of the whole network, instead of performing approximation at the communities level. Global techniques tune their parameters to "one size fits all" model, so they are quite successful with extracting communities in homogeneous cases but suffer in heterogeneous community size distributions.**

In this paper, we present a stateful diffusion approach (Stad) for community detection that employs diffusion. Stad boosts diffusion with a conductance-based function that acts like a tuning parameter to control the diffusion speed. In contrast to existing diffusion mechanisms which operate with global and fixed speed, Stad introduces stateful diffusion to treat every community individually. Particularly, Stad controls the diffusion speed at node level, such that each node determines the diffusion speed associated with every possible community membership independently. Thus, Stad is able to extract communities more accurately in heterogeneous cases by dropping "one size fits all" model. Furthermore, Stad employs a vertex-centric approach which is fully decentralized and highly scalable, and requires no global knowledge. So as, Stad can be successfully applied in distributed environments, such as large-scale graph processing or decentralized machine learning. The results with both real-world and synthetic datasets show that Stad outperforms the state-of-the-art techniques, not only in the community size scale issue but also by achieving higher accuracy that is twice the accuracy achieved by the state-of-the-art techniques.

I. Introduction

Community detection has been an important problem for network analysis in a number of fields including physics, computer science, social science, and biology [1], [2], [3]. In network analysis tasks, graph structure has been utilized as a fundamental tool to represent real-world networks, such that systems can be represented by a set of nodes (vertices) and a set of edges that are connections between pair of nodes (neighbors). Community detection is the task of grouping nodes into clusters with better internal connectivity than external connectivity [1], [3], [4]. However, like any clustering task that is known to be NP-hard, it is prohibitively expensive to search

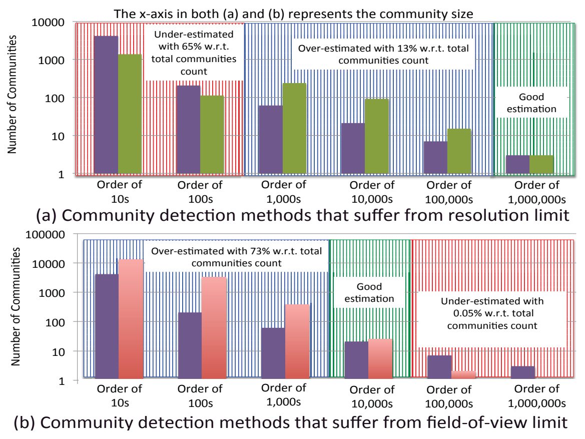


Fig. 1. Community size distribution of Friendster dataset. The red and blue shaded regions highlight the accuracy limitation of existing community detection methods with heterogeneous (skewed) community size distribution. (a) The results obtained using Louvain method show the limitation of methods that suffer from resolution limit by favoring large communities over small ones. (b) The results obtained using Infomap mehthod show the limitations of methods that suffer from field-of-view limit by favoring small communities over large ones.

all such groups for the optimal value of internal and external connectivity [1], [3], [4]. Accordingly, different heuristics [5], [6] or approximation algorithms [7], [8] have been introduced to achieve good performance in a reasonable time [4], [1], [2], [3].

Hierarchical organization is one of the important aspects that is related to community structure exhibited by most networked systems in the real-world [9]. Real-world networks are usually composed by communities which in turn have smaller sub-communities. In social networks, for instance, Facebook users can be grouped into hierarchical communities represented with different granularity levels of cities, countries, or continents. Another important aspect related to community structure is that communities may overlap by sharing some of the vertices. For example, a person usually has connections to multiple groups or communities, such as family members, colleagues, friends, and co-workers. Detecting overlapping membership mixed with the hierarchical structure makes community detection a particularly hard task. Specifically, bigger overlap results in more densely connected communities. Thus, it becomes harder to correctly identify the

[1]This work has been accomplished during the PhD studies of the first author at School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden.

2575-8411/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDCS.2018.00107

overlapping nodes as members of multiple communities and not to identify them as a separate community nor merge the overlapping communities into a single community.

Most of the existing community detection algorithms fall into the category of global approaches [10]. These global approaches adapt their detection model focusing on the global structure of the whole network, instead of addressing the approximation at the communities level. Particularity, existing approaches are designed to tune their model parameters globally, thus, they either fail to detect small communities if the parameters are tuned with respect to large communities, or the other way around, large communities might be over-partitioned and detected as multiple communities. Therefore, global techniques tune their parameters to "one size fits all" model, so they are quite successful with extracting communities in homogeneous cases but suffer in heterogeneous communities. However, existing studies show that community size distributions in real-world networks follow power-law distributions [1], [2]. Accordingly, applying global approaches on such skewed cases results in low accuracy in terms of the extracted communities.

Figure 1 shows an example of community size distribution using a real-world dataset. As shown, community size distribution is skewed and some approaches have been shown to have a resolution limit [11], [4], such that they fail to identify communities that are smaller than specific scale. For example, on Friendstr dataset up to half of small communites are not correctly extracted, and got merged with bigger communities. On the other hand, other approaches suffer from field-of-view limit [12], such that they fail to detect large communities, while overestimating the number of small communities. Figure (1) (b) shows that up to two thirds of small communities are over-estimated among extracted communities. Therefore, there is a need to develop community detection approaches that target optimization at community level to adapt to heterogeneous community size distributions in order to avoid discrepancies of existing approaches.

Moreover, the globally based detection algorithms usually run with high computational cost. Therefore, the random walks technique has been extensively adopted to extract disjoint communities as one of the lowest computational overhead approaches. In particular, random walks are used to explore local structure around very few subset of nodes chosen as seed set, such that those nodes are labeled using their ground-truth community memberships [13], [14], [10]. Accordingly, this method requires some known members as the prior for the semi-supervised clustering to perform the probability diffusion from them. Furthermore, diffusion is a random walk-based mechanism that adopts the label propagation approach, which defines rules that simulate the spread of labels of vertices in the network. Specifically, diffusion captures how a flow starting from a specific node spreads on a graph, such that spread of that flow mixes fast within well connected regions, and slowly in less connected ones. The intuition behind random walks and diffusion is that once a random walker (or the diffusion process) enters a region, it tends to stay there for a long time,

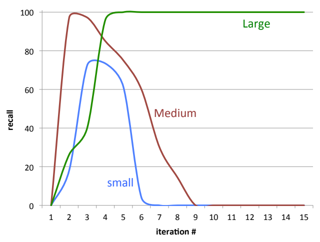


Fig. 2. Random Walker mixing time w.r.t. community size. As shown small community can be covered within short time, however, longer mixing time results in merging it with bigger communities.

and movements between regions are relatively rare via one of the few outgoing edges. Thus, this phenomenon can be used for community detection by capturing the boundaries of well connected regions. Yet, mixing time of random walker needs be controlled, as very long random walk reaches a stationary distribution which is not expected to indicate the extraction of well connected clusters.

Community detection can be performed using diffusion by representing communities with multiple flows in the network and assigning each flow a different color representing it [15], [16]. However, the mixing time of diffusion process depends on the size of the community, such that larger communities require longer mixing time. Figure 2 illustrates how the coverage percentage (i.e., recall) of the random walks differs with respect to the size of communities. As shown, the random walker for large communities requires longer time in order to achieve full coverage of community members. During this time, random walkers of smaller communities get trapped and lost within bigger communities, resulting in merging smaller communities with larger ones. For example, Figure 3(b) demonstrates three successive iterations of random walk mechanism, as shown when the walker get trapped in a region, the set of nodes belonging to that region are grouped as a community. However, the random walker might cause the merging of different communities when the mixing time of the walker increases. Figure 3(c) shows three successive iterations with diffusion method. The histogram colors show the average volume of colors per each community. As shown, diffusion speed is the same for any walker and does not depend on the specific cluster where it happens. As a result, diffusion process treats all colors equally and mixes all the colors everywhere in the graph, thus it decreases the chances of identifying small communities as separate modules.

To address these limitations, in this paper we present Stad that introduces the concept of sateful diffusion and boosts diffusion with adaptive speed function, hence detecting heterogeneous size communities. Stad adaptively scales down the speed of the diffusion process in the regions where topological communities start to emerge. Moreover, Stad treats the random walkers independently and controls the diffusion speed at each node in the graph. Specifically, Stad allows each node to create a dynamic resistance to slow down the diffusion. Thus, Stad allows nodes to assign different levels of importance to the flows passing them. Importance of a flow reflects how

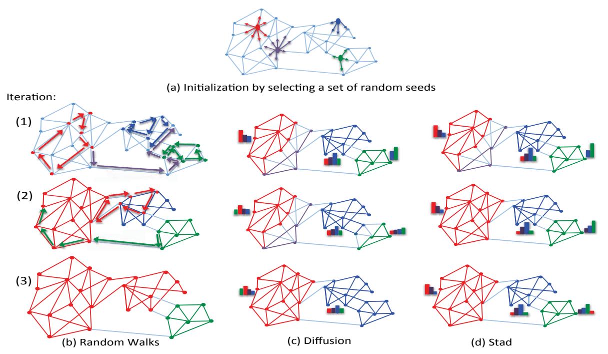


Fig. 3. Visulaization of different iterations in flow models. (a) The initial graph, while some random nodes are selected as seeds and indicated using different colors. Afterwards, illustrating three successive iterations of each community detection method: (b) Random Walks, (c) Diffusion, and (d) Stad. The histograms in (c) and (d) show the average volume of colors per each community.

strongly a node is connected to the community formed by this flow, such that, the more important the flow is, the slower the mixing time. As shown in Figure 3(d), Stad succeeds in maintaining the smallest community as a separate module. Additionally, Stad follows a decentralized, iterative and node-centric approach, such that each node executes the algorithm in parallel. Therefore, Stad can be successfully applied in distributed environments, such as large-scale graph processing or decentralized machine learning. Furthermore, nodes need only to communicate their updated memberships at the end of each iteration with their direct neighbors. Experiments with real-world datasets show that Stad outperforms the state-of-the-art techniques by achieving higher accuracy that is twice the accuracy achieved by the state-of-the-art techniques.

Summary of contributions: Stad provides a novel flow-based community detection algorithm that is based on adaptive optimization function that allows it to avoid community size limitations. Furthermore, Stad employs a dynamic diffusion model that treats different flows independently and controls the diffusion speed of each flow separately. Stad is decentralized, scalable and highly parallel community detection algorithm with no global knowledge required. Results confirm Stad's ability to extract disjoint as well as multiple community membership(s) with double of the accuracy achieved by the current state-of-the-art techniques using real-world datasets.

The remaining of the paper is organized as the following. Section II briefly describes the state-of-the-art methods for community detection. Section III illustrates the core concepts and definitions introduced with stateful diffusion, as well as the implementation of Stad. Furthermore, in Section IV we discuss the initialization phase implemented in Stad and the different ways of selecting seeds. Then, in Section V, we discuss the details related Stad's performance in terms of convergence and complexity analysis. In Section VI, we present evaluation of Stad compared to the state-of-the-art methods. Finally, Section VII concludes the paper.

TABLE I
COMPLEXITY AND CATEGORY OF EXISTING APPROACHES.

Algorithm	Complexity	Category
Louvain	$O(N \log N)$	Modularity-based optimization
Infomap	$O(\|E\|)$	Random walk heuristics
Oslom	$O(N^2)$	Statistical inference
Bigclam	$O(\|E\|)$	Statistical inference
Stad	$O(\|E\|)$	Diffusion with conductance-based optimization

II. Related Work

In this section, we summarize the state-of-the-art approaches used to identify communities in undirected graph $G = (V, E)$, which is a tuple containing a set of vertices V with size N and a set of edges E. Table I lists the optimization method as well as the computational complexity of each of the discussed approaches.

Louvain [17] is widely known as a very fast approach to maximize modularity. The modularity of a graph compares the presence of each intra-cluster edge of the graph with the probability of that edge would exist in a random graph. Louvain first assigns a different community to each node of the network, then a node is moved to the community of one of its neighbours with which it achieves the highest positive contribution to modularity. The above step is repeated for all nodes until no further improvement can be achieved. Then each community is considered as a single node on its own and the second step is repeated until there is only a single node left or when the modularity can not be increased in a single step. Yet, Louvain follows a centralized approach and extracts only disjoint community membership.

Infomap [18] figures out communities by employing random walks to analyze the information flow through a network. The general idea behind the algorithm is to find a binary code with unique short names (codewords) for each node within a community that can be used to describe the position of a random walker in the network. In a network with a marked community structure, the probability of flipping membership is small. Therefore, Infomap compresses the description by reusing codewords within individual communities. However, Infomap suffers from field-of-view limit and fails to detect large communities [19].

Oslom [20] follows statistical inference approach, such that it evaluates the statistical significance of a cluster with respect to a random graph during community expansion. Oslom begins with assigning nodes to random clusters, then the second phase evaluates growing the current community structure by merging separate modules. To grow a specific community, a cumulative probability is calculated for each outsider node to measure the gain of adding it to the community. If the cumulative probability is more significant than the value computed for this community in the random model, the node is added to the community.

Bigclam is another approach that follows statistical inference [21]. Bigclam formulates community detection as a variant of nonnegative matrix factorization. Specifically, Bigclam tries to optimize the likelihood of explaining the links of the observed network. However, Bigclam as well as Oslom usually result in a significant number of singleton communities as discussed in Section VI.

All methods described above fall into the category of global approaches. Specifically, these global approaches adapt their detection model focusing on approximating the global structure of the whole network, instead of performing approximation at the communities level. In addition, several of these methods usually run with high computational cost. Differently from existing work, the primary goal of our work is to develop a decentralized community detection approach that is capable of tuning the community detection parameters for each community separately. Therefore, Stad employs diffusion as a vertex-centric approach and boosts it with a conductance-based function. Specifically, conductance is used to evaluate the quality of communities by measuring the fraction of inter-cluster edges per each cluster to the intra-cluster edges [3], [22]. Most importantly, Stad controls the diffusion speed for each possible community membership at each node as illustrated in the next section.

III. STAD: STATEFUL DIFFUSION

In this section we discuss Stad which extracts disjoint as well as overlapping communities in undirected and weighted graphs. Stad presents a self-adaptive approach that tunes the detection parameters on community level. Stad is an iterative algorithm that starts by diffusing different colors from some nodes identified as seed set, such that each color represents a separate community. Afterwards, nodes propagate these colors following the underlying notwork for some iterations. Stad allows nodes to decide their community memberships based on the amount of colors they receive at each iteration. Specifically, each node joins the community having the maximum amount of color received by the node. Differently from existing diffusion methods that operate with fixed diffusion speed, Stad boosts diffusion with conductance-based function that acts like a tuning parameter to control the diffusion speed at community level. To illustrate this further, we start by introducing the notations and definitions used, then we describe the proposed algorithm.

A. Notations

We denote by V the set of nodes in a graph G. Let W be the adjacency matrix of G, where w_{ij} denotes the weight of an edge between nodes i and j. In unweighted graphs w_{ij} is either 0 or 1, and in undirected graphs $w_{ij} = w_{ji}$. In this paper we consider weighted and undirected graphs. A community, also called a cluster, $C \subseteq V$, is a subset of nodes, and its complement $\overline{C} = V \setminus C$, which contains all the nodes that are not in C. Accordingly, we can define the strength of connection between two communities x and y by the weight of edges connecting them, and write $w_{xy} = \sum_{i \in x} \sum_{j \in y} w_{ij}$. Note that we consider any subset of nodes to be a community, and the goal of community detection is to find *good* communities.

Conductance is one of the most widely used scoring functions to evaluate the goodness of extracted communities, such that it measures how strongly a set of nodes is connected to the rest of the graph. Conductance is a metric that takes into consideration both external and internal connections of a community. It is defined as the ratio of the weight of edges on the boundary of the community over the sum of weight of edges connecting nodes inside the community [3]. If this value is small, then it indicates a good clustering as the extracted communities have few edges leaving them. However, minimizing conductance might result in enlarging C with irrelevant members just to minimize the volume of external edges. Therefore, conductance of community C is defined using the following formula:

$$\phi(C) = \frac{w_{C\overline{C}}}{min(w_{CC}, w_{\overline{C}V})}. \quad (1)$$

As shown in the above formula, the denominator is set to the minimum value between volume of internal edges (i.e., w_{CC}) and volume of edges connecting nodes that are not members of C (i.e., $w_{\overline{C}V}$) [3], [22]. Thus, the conductance of G is then given by:

$$\phi(G) = \Sigma_{C \subset V}\ \phi(C). \quad (2)$$

Finding the minimum conductance is known to be NP-Complete [22]. Therefore, Stad employs a local heuristic for minimizing conductance, such that each node in the graph decides its community membership(s) by choosing community (or multiple memberships in case of overlapping communities) from the communities adopted by the node's neighbors. More formally, each node minimizes $\phi(C)$ with respect to its direct neighbors.

B. Stateful and Dynamic Diffusion

At first, the relationship of community detection to diffusion and random walks might seem unclear, but they are neatly related. We desire to extract communities that are well connected internally, but less connected to the rest of the graph. Intuitively, if a diffusion process is started in one of those communities, it is unlikely to leave that community since it is poorly connected to the rest of the graph. Accordingly, community detection can be performed using diffusion by representing communities with multiple flows in the network and identifying each flow with unique color assigned to it. Regular diffusion spreads the flows in the graph while considering that the amount of flow entering a node equals the amount of flow leaving it. Specifically, each node passes forward the received flow to its neighbors, such that each one of them receives partial volume that is proportional to the strength of the edge touching it. A general diffusion process that affects node v_i having set of friends $v_i.\mathcal{F}$ can be defined as:

$$\Omega_{v_i}(t+1) = \Sigma_{v_j}(w_{ij} \times \Omega_{v_j}(t)), \forall v_j \in v_i.\mathcal{F}. \quad (3)$$

As shown in Equation 3, $\Omega_{v_i}(t+1)$ represents the amount of flow that node v_i is going to receive from its neighbors at iteration $t+1$. Diffusion is an iterative process, so $\Omega_i(t+1)$ is updated according to the magnitude of the flow

TABLE II
BASIC NODE VARIABLES AND THEIR DEFINITIONS FOR NODE v_i.

Symbol	Name	Definition
$\mathcal{F}$	Neighbors	$\mathcal{F} = \{v_j \in V \mid w_{ij} > 0\}$
d	Degree	$d = \sum_{v_j} w_{ij}, \forall v_j \in \mathcal{F}$
$\mathcal{C}$	Neighbor colors	$\{c_1, ..., c_m\}$ collected from $\mathcal{F}$
$\mathcal{I}$	Color intensities	$\{i_1, ..., i_m\}$ collected amount of colors
dc	Dominant color	$dc \in \mathcal{C}$ s.t. $\mathcal{I}[dc]$ is maximum
$\mathcal{W}^{int}$	Internal strength	$\mathcal{W}^{int}[c] = \frac{\sum_{v_j \in \mathcal{F}} w_{ij}}{d}$ s.t. $v_j.dc = c$, $\forall c \in \mathcal{C}$
$\mathcal{W}^{ext}$	External strength	$\mathcal{W}^{ext}[c] = 1 - \mathcal{W}^{int}[c]$

received by neighbors of v_i during the previous iteration (i.e., $(\Omega_{v_j}(t), \forall v_j \in v_i.\mathcal{F})$, as well as the weight of edges connecting them (i.e., $w_{ij}, \forall v_j \in v_i.\mathcal{F}$).

As aforementioned, Stad treats the color flows independently and controls the diffusion speed at each node in the graph. Accordingly, we introduce the notion of node state, such that, each node keeps portions of the flows passing it for itself and stores these portions in its state. Our stateful diffusion process for node v_i can be defined as:

$$\Omega_{v_i}(t+1) = \Sigma_{v_j}(w_{ij} \times S \times (\Omega_{v_j}(t))), \forall v_j \in v_i.\mathcal{F}, \quad (4)$$

where we introduce S as the function that controls the diffusion speed and allows nodes to keep portions of the flows passing them into their states. The more important the flow is, the higher the amount of color is kept inside the node state. The intuitive idea of employing the speed function S is to have a clear distinguish between internal and boundary nodes of each community. Specifically, internal nodes represent the core of the community, therefore, they need to maximize the concentration of community color around them. On the other hand, boundary nodes may belong to multiple communities and their task is to allow colors to travel further and expand. Therefore, S is computed in adaptive and flexible way by each node according to the importance of the flows passing it. The importance of a particular flow to specific node is decided according to the community membership of that node and diffusion assessment phases as described in the next subsections. We follow that with describing the algorithmic steps for color diffusion in Stad, as well as different stopping criteria.

1) Membership Assessment: Stad allows each node to individually choose its community membership(s) from the different communities surrounding it. The membership calculations are performed using the set of variables maintained by each node as listed in Table II:

- **Neighbors** ($\mathcal{F}$): the list of node's direct neighbors.
- **Degree** (d): the weighted degree of a node.
- **Neighbor Colors** ($\mathcal{C}$): the list of colors received from node's neighbors so far.
- **Color Intensities** ($\mathcal{I}$): the corresponding amount of colors received from the neighbors.
- **Dominant Color** (dc): dc identifies the community membership of a node. Specifically, dc is the mostly common color among node's neighbors and it is the color with the highest intensity among the received ones. In case of overlapping community detection, dc is considered as the set of colors that have the highest color intensities ordered descendingly.
- **Internal Strength** ($\mathcal{W}^{int}$): $\mathcal{W}^{int}$ represents the probability of considering a node as a member of the communities that are adopted by the neighbors of that node. Thus, $\mathcal{W}^{int}[c]$ for color c is calculated by summing the volume of edges connecting a node to its neighbors adopting c as their dc divided by the node degree.
- **External Strength** ($\mathcal{W}^{ext}$): $\mathcal{W}^{ext}$ represents the volume of external edges if a node decides not to be a member of the communities adopted by its neighbors.

2) Diffusion Assessment: Our objective is to minimize the amount of flow/color leaving a community, hence avoid the situations where colors compete to impose authority to dominate other regions. Accordingly, each node individually controls the amount of flow leaving it by providing different speed functions that either slow down for important colors or speed up the diffusion for non-important ones at each iteration. Furthermore, colors are treated independently from each other, hence no congestion or conflicts can happen among colors passing a node. Stad provides two different speed functions, one for single community membership while the second for multiple memberships.

The first function is an entropy-based step function that is used for single community membership. Entropy is commonly used in thermodynamics and is associated with the amount of order in the system, such that the lower the entropy the more order exhibited in the system [23]. Accordingly, Stad calculates entropy-based resistance for each node in terms of $\mathcal{W}^{int}$ and $\mathcal{W}^{ext}$ as shown in Equation 7. The higher the probability of a node being a member of a specific community, the lower the entropy, hence, the lower the diffusion rate of that color associated to that community and the higher amount of that color the node stores in its state. In case of single community membership, the nodes are interested only to preserve more amount from their dc. Therefore, the whole amount of any color is diffused as long as $\mathcal{W}^{int}[c] < 0.5$, such that we have used 0.5 to represent the majority of node's neighbors. Thus, to slow down the speed of diffusion when the node is 50% (or higher) confident about its membership, the speed is calculated as using the speed function S_1.

$$e_{int}(c) = -\mathcal{W}^{int}[c] \times \log_2 \mathcal{W}^{int}[c]. \quad (5)$$

$$e_{ext}(c) = -\mathcal{W}^{ext}[c] \times \log_2 \mathcal{W}^{ext}[c]. \quad (6)$$

$$S_1(c) = \begin{cases} 1 & \text{if } \mathcal{W}^{int}[c] < 0.5 \\ e_{int}(c) + e_{ext}(c) & \text{otherwise.} \end{cases} \quad (7)$$

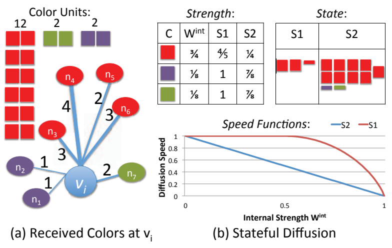
Furthermore, we have implemented a linear diffusion speed function for the case of overlapping community detection. Particularly, nodes maintain states for all colors (communities) surrounding them. Then, the diffusion speed is going to be continuously scaled down for each color according to the strength of nodes membership to that color. Hence, the linear-based function defined as the following:

$$S_2(c) = 1 - \mathcal{W}^{int}[c]. \tag{8}$$

Example: Figure 4(a) shows an example of node v_i having 7 edges, where the number in top of the edge represents its weight. As shown, $v_i.dc$ is red as it has the maximum amount of collected units. After each node decides its dominant color, it starts to evaluate the status of its membership by computing both internal and external strength. First table in Figure 4(b) lists the values of internal strength associated with each color received by v_i. The external strength is the complement of the computed value of the internal strength. Our objective is minimize the conductance between different communities. In our case, conductance can be computed as the fraction of node's neighbors that are different in terms of dc value. S_1 is the diffusion speed using the entropy-based function, whereas S_2 is the diffusion speed using linear function. As aforementioned, when the diffusion speed is decreased for some colors, the node keeps some amount of those colors in its state. Specifically, the amount kept in node's state is equivalent to the remaining amount of color after scaling down the diffusion speed. For example, as shown in second table in Figure 4, v_i diffuses only 3 units of red color (i.e., that equals to quarter of the amount it has) and keeps 9 units in its state when applying S_2. For simplicity, we assume that v_i's state was empty at the first iteration, yet the state is incrementally updated by adding more units at the successive iterations.

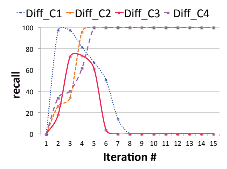
Furthermore, Figure 5 shows the difference between Stad and regular diffusion during the successive iterations of color spread. The plot is generated using a synthetic graph with four ground truth communities. The diffusion process is initialized with four seeds, each seed is selected to represent a community and is assigned a different color. As shown, in each iteration we report the percentage of nodes that correctly determine their community membership based on amount of colors they received. However, diffusion creates a competition among different colors throughout the first four iterations. For example, starting from the 3rd iteration, *Diff_C4* dominates and takes control of *Diff_C1* and *Diff_C3*. At the 6th iteration, the regular diffusion method stabilizes with two balanced partitions that are far from describing the ground-truth communities. On the contrary, Stad diffuses colors at different speed and succeeds in extracting communities that are similar to the ground-truth communities, as shown in Figure 5 (c) and (d).

3) Stad Algorithm: Stad follows a decentralized, iterative and node-centric approach, such that each node executes the algorithm in parallel. Nodes need only to communicate their updated memberships and the amount of colors they send, at the end of each iteration with their direct neighbors. Algorithm 1 shows the execution steps performed by each node. As described in procedure *diffuse*, each node calculates diffusion speed associated with each color received from its neighbors in the previous iteration. Then, each node computes $unit_c$ of color c by dividing the collected intensity associated with c by its degree. Consequently, the amount of color to be diffused to a particular neighbor, $amount_c$, equals to the multiplication of $unit_c$ with the S's value and the weight of the edge connecting that neighbor. After the colors are diffused to the neighbors, in the successive iterations nodes can calculate their community membership (i.e. dominant color: dc) by summing up received amounts per each color and selecting the one that has the maximum intensity. In case of overlapping community detection, each node maintains a list of surrounding colors ordered by the volume of colors from the highest to lowest.

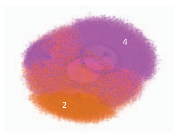


C	W^{int}	S1	S2
🟥	¾	⅘	¼
🟪	⅛	1	⅞
🟩	⅛	1	⅞

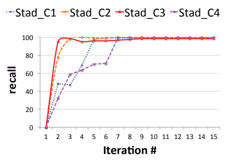
Fig. 4. Example of the sateful diffusion at node v_i. (a) The received colors at node v_i at time $t = 2$. (b) First table lists the internal strength calculated for each color, whereas second table shows amount of colors kept in v_i's state using different speed functions.



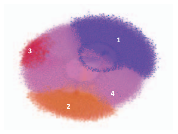
(a) Diffusion



(b) Found communities



(c) Stad



(d) Found communities

Fig. 5. Example of the extracted communities by Stad and Diffusion mechanisms using a synthetic dataset generated with 4 communities. (a) The percentage of ground truth nodes discovered inside each community during the execution of Diffusion. (c) The ground truth percentage achieved by Stad. Visualization of extracted communities by Diffusion and Stad is shown in (b) and (d), respectively.

Algorithm 1: Stad Diffusion at node v_i

```
Procedure diffuse()
    forall the $c \in v_i.\mathcal{C}$ do
        $S[c] \leftarrow calcSpeed(c)$
        $unit_c \leftarrow v_i.\mathcal{I}[c]/v_i.d$
        forall the $v_j \in v_i.\mathcal{F}$ do
            $amount_c \leftarrow unit_c \times w_{ij} \times S[c]$
            $send(v_j, amount_c)$
        end
    end
```

Then, node decides its membership to a color c, if $\mathcal{W}^{int}[c]$ is greater than some threshold. In Section VI-B2, we show the performance evaluation with different threshold values.

4) Stopping Criteria: The more natural and simple way to check for convergence is to deactivate the nodes that are not interested anymore in changing their community membership(s). Thus, if the number of those nodes, compared to the total number of nodes in the graph, is higher than specific threshold, then the algorithm stops. Furthermore, we can focus on the community level rather than the node level and keep track of the communities detected in each iteration. If the number of nodes per detected communities remains the same for particular number of iterations, then the speed function succeeds in preventing dominant colors to travel further-away from their original communities, hence random walks reach the stable state and stay inside their communities. Lastly, we can specify the number of iteration needed as a parameter to the algorithm, and relate it to the properties of the underlying graph, i.e., how good/bad expander the graph is. As aforementioned, Stad follows a decentralized approach, however, checking for convergence requires having orchestrators among participating nodes. Thus, some nodes are chosen to audit changes occur in communities memberships and notify remaining nodes when the convergence criterion is achieved.

IV. Choosing Initialization Seeds

Flow based models are very sensitive to the initial seeds of diffusion from which the process starts (e.g., set of nodes from which random walks start). In such models, a seed set is selecting by choosing some nodes and assign each node a different color. Thus, identifying the set of the most central nodes is important to help disseminating information in the network faster and more efficient. There are various centrality measure, such as degree, closeness, and eigenvector centrality that can be used to choose those central nodes . In the following subsections we discuss the different initialization methods used in Stad and their relations to different node centrality measures. First, we use degree centrality as a global centrality measure. Then, we introduce two different local approximations for closeness centrality.

A. Degree Centrality

Degree is the simplest node centrality measure, such that it is calculated using only the local structure around nodes. The more edges nodes have, the faster the dissemination of colors is, if those nodes are selected as seeds. Accordingly, we randomly choose seeds from the set of high degree nodes. We define the high degree nodes as the nodes ranked in the top one third among the degree of all nodes in the network.

B. Closeness Centrality

This calculates scores for each node based on its closeness to all other nodes within the network. Closeness is defined as the sum of distances to all other nodes. The intent behind using this measure is to identify the nodes which could reach others quickly. Specifically, this centrality measure calculates the paths between all nodes, then assigns each node a score based on its sum of paths crossing it. However, calculating all the paths between all nodes is very costly especially with the large networks we have nowadays. Therefore, we used clustering coefficient (CC) as a local approximation to closeness. CC of a node is the ratio between the number of existing links among its direct neighbors and the number of links that could possibly exist among them. The more connected neighborhood surrounding a node, the higher the probability to have more paths crossing that node. Stad operates as a node-centric approach, so we assume that every node calculates its CC locally by keeping track of two-hop neighbors (i.e., neighbors of the neighbors). We calculate CC using the following formula:

$$CC_{v_i} = \frac{|e_{jk} : j, k \in v_i.\mathcal{F}, e_{jk} \in E|}{2 \times |v_i.\mathcal{F}| \times (|v_i.\mathcal{F}| - 1)} \tag{9}$$

However, iterating over 2-hop neighbors to calculate nodes CC still can be costly in large networks. Thus, we use random walk (RW) as another approximation for closeness centrality. The random walk starts at some vertex $u \in V$. Then, the process moves forward from u to one of its neighbors chosen uniformly at random. We add node u to the set of initialization seeds, if RW visits it again after specific number of intermediate steps. Specifically, when a node releases a short random walker (i.e. with a walk of length 3 to 6 steps away) and the random walk visits it back, this indicates that this node has well connected neighborhood.

V. Convergence and Complexity Analysis

In the following subsections we discuss the details related Stad's performance in terms of convergence and complexity analysis.

A. Convergence Analysis

Convergence analysis of Stad is twofold: reaching convergence and extracting communities while minimizing the conductance. The starting point for analyzing diffusion (i.e., a random walk) on a graph is a simple Markov process on that graph. Specifically, at each time step t, the random walker has a state $x(t) \in V$, indicating which node the random walker is on at time t. If $x(t) = i$, at the next time step $t+1$, the random walker moves to one of the i's neighbors chosen at random. It has been shown that as long as the graph is undirected, connected, and not bipartite, the random walker from any starting point will converge to the stationary distribution [24]. Furthermore, properties of the convergence of a random walk (i.e., bounds on mixing rate and convergence) are determined by the *spectrum* of the graph [25], [26].

However, reaching stationary distribution is not expected to indicate the extraction of well connected clusters. Hence, the mixing time of random walker needs to be controlled in a way not to produce communities that are too small or too big. Thus, we have used conductance to take the size of resulting communities into consideration. There are many theoretical studies that use *Cheeger ratio* [24] of a graph, that is computed the same way as local conductance of extracted communities.

Lovász and Simonovits [27] show a theoretical proof that finding a cluster containing a vertex can be accomplished by simulating a random walk starting from that vertex. Authors [28], [29] extend that to prove the following two properties: **(i)** large random walk can find a cluster C with conductance, defined in terms of Cheeger ratio, that is lower than a given target Cheeger ratio. Particularly, this given target Cheeger ratio can be initialized with the number of edges found in the graph divided by 2 (i.e. $|E|/2$). Furthermore, **(ii)** having C with lower Cheeger ratio minimizes the probability that any random walk that started at node $v \in C$ leaves C.

Having these two properties proved, it is easy to show that Stad satisfies both of them. First of all, Stad decides community memberships of all nodes in the graph based on the diffusion processes passing them. Instead of having only a single starting vertex as in previous work, Stad processes all nodes in the graph and finds clusters for those nodes. Secondly, Stad follows iterative approach, such that in each iteration nodes try to minimize conductance. Thus, Stad extracts communities having lower Cheeger ratio. More interestingly, the speed functions introduced in Stad that control the diffusion speed at each node allow Stad to stabilize faster, while satisfying the second property.

B. Complexity Analysis

Stad's complexity cost is expected to be low given that nodes perform their local computations independently and in parallel. We discuss the complexity of Stad in terms of communication traffic among all of the nodes in the graph. Particularly, every node exchanges set of colors with its direct neighbors for specific number of iterations till reaching convergence. Accordingly, in each iteration all of the edges of the graph will be touched, such that source and destination nodes exchange some information with each other. So as, the algorithm complexity is $\mathcal{O}(|E| * R)$, where E is the edge set in the network, and R is the total number of iterations needed for the algorithm to converge. The number of iterations required for convergence is proportional to the time required for the existing colors in the graph to spread out to all the reachable regions. However, for all practical reasons $R << |E|$, thus, Stad's complexity is linear in terms of the number of edges.

VI. Evaluation

We proceed by evaluating the performance of Stad and comparing it with the state-of-the-art community detection methods described in Section II, we have used the implementation available by the original authors for each method and we used their default settings for the parameters introduced by each algorithm. Furthermore, we have performed the comparison on a range of networks, some of them are real-world networks with their ground-truth communities available from Snap[1]. Additionally, we have used some synthetic weighted and undirected networks generated using LFR benchmark [30]. LFR benchmark generates graphs with a built-in community structure, and simulates properties of real networks accounting for heterogeneity of node degree and community size distributions. For each of the following experiments, we have executed Stad for three times and reported the average performance of the three runs.

[1]https://snap.stanford.edu/data/

A. Evaluation Metrics

Given a network $G(V, E)$, we consider a set of ground-truth communities C^{gt} and a set of extracted communities C^{ex} where each ground-truth community $C_i \in C^{gt}$ and each extracted community $C_j \in C^{ex}$ is defined by the set of nodes belonging to it. In order to quantify the level of correspondence of C^{ex} to C^{gt} we consider:

1) **Average F1-Scores**. F-Score measures the clustering accuracy in terms of precision and recall. Precision reflects mixing of different ground-truth communities into the extracted ones. Thus, the higher the precision, the less mixing of ground-truth communities. Furthermore, recall reflects the goodness of grouping nodes that belong to the same ground-truth communities. There are different versions of F-Score according to the weighted average of the precision and recall. We consider F_1 that is computed using the following formula:

$$F_1 = 2.\frac{precision.recall}{precision + recall} \tag{10}$$

2) **Average B^3 Scores**. B^3 is commonly used in information theory to evaluate the coreference resolution for entity disambiguation problems [16]. B^3 evaluates a gold-standard clustering of entities (which are nodes in our case) against a system-produced clustering of entities at entity level, such that, B^3 quantifies for each entity how well it is clustered with similar ones. The B^3 precision and recall are usually defined as:

$$B^3 Precision = \frac{1}{|V|} \sum_{i \in V} \frac{|C_i^{gt} \cap C_i^{ex}|}{|C_i^{ex}|} \tag{11}$$

$$B^3 Recall = \frac{1}{|V|} \sum_{i \in V} \frac{|C_i^{gt} \cap C_i^{ex}|}{|C_i^{gt}|} \tag{12}$$

Having the values of B^3-based precision and recall computed, we calculate B^3-based F_1 measure as described in the previous evaluation metric.

3) **Normalized Mutual Information (NMI)**. NMI is adopted as well from the information theory domain and it is used to measure the similarity between two clusters (i.e., the ground-truth and system generated one). Refer to [31] for more details.

4) **Average Internal Density**. Internal density of a cluster computes the fraction of intra-cluster edges to the possible number of edges that can be created among the nodes in that cluster. We compute the average internal density

TABLE III

CHARACTERISTICS OF THE REAL-WORLD DATASETS. AVG-CC REPRESENTS THE AVERAGE CLUSTERING COEFFICIENT, AVG-CONDUCTANCE LISTS THE AVERAGE CONDUCTANCE FOR THE GROUND-TRUTH COMMUNITIES, OVERLAP LISTS THE TOTAL PERCENTAGE OF OVERLAP ACROSS ALL COMMUNITIES, (2)-OVERLAP SHOWS THE PERCENTAGE OF NODES THAT HAVE TWO COMMUNITY MEMBERSHIPS, FINALLY, (3+)-OVERLAP LISTS THE PERCENTAGE OF NODES THAT ARE MEMBERS IN THREE OR MORE COMMUNITIES.

	Vertices	Edges	Avg-CC	Communities	Avg-Conductance	Overlap %	2 Overlap %	3+ Overlap %
Amazon	334,863	925,872	0.09	75,149	0.55	96	3.6	92.4
Dblp	317,080	1,049,866	0.03	13,477	0.68	42.5	16.6l	25.9
Youtube	1,134,890	2,987,624	0.67	8,385	0.9	37.6	16.1	21.5
Livejournal	3,997,962	34,681,189	0.15	287,512	0.94	64.2	17.2	47
Friendster	65,608,366	1,806,067,135	0.06	957,154	0.4	54.5	20.8	33.7

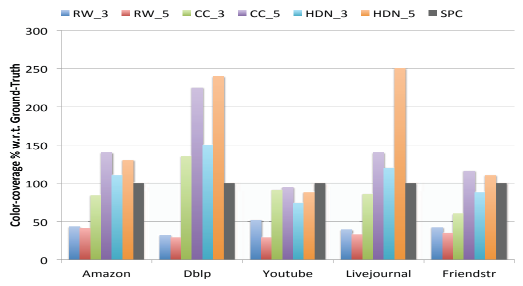


Fig. 6. Evaluating the performance of the different initialization methods w.r.t. percentage of covered ground-truth communities.

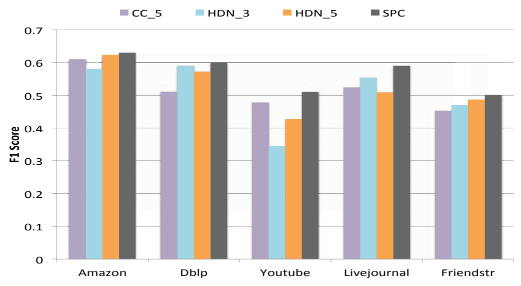


Fig. 7. Evaluating the performance of the best coverage initialization methods w.r.t. F1 score.

for all extracted clusters as the following formula:

$$AvgDensity = \frac{1}{|C^{ex}|} \sum_{i \in C^{ex}} 2.\frac{|E_{xy} : x, y \in C_i^{ex}|}{|C_i^{ex}|.(|C_i^{ex}| - 1)} \tag{13}$$

5) **Conductance**. We follow the same definition introduced in Equation 1.

B. *Real-world Graphs*

Table III summarized the characteristics of the used real-world networks. As shown, there is high overlapping percentage across all the datasets.

1) Selecting Seeds:

As aforementioned, the number of communities is not known beforehand, so the diffusion process needs to be initialized with number of seeds higher than the number of expected communities. In Section IV, we briefly discussed different approaches to select the seed set for initialization. In the following experiments we evaluate the different seeding methods and investigate how the size of the seed set affects the performance of our algorithm. Our objective is to maximize the coverage of the selected seed set, such that the best case scenario is to select at least one node from each ground-truth as a seed.

Figure 6 depicts the performance of each initialization method in terms of the percentage of covered ground-truth communities, such that at least one of their members has been chosen as a seed. We compare random walks (RW), clustering coefficient (CC) and high degree nodes (HDN) seeding methods. For RW, we used two different random walk lengths, namely RW_3 and RW_5 with 3 and 5 steps, respectively. For CC, we tried two different settings, CC_3 and CC_5. CC_3 represents selecting at random 30% of nodes having $CC > 0.5$ (i.e., majority of node's neighbors are mutually connected). CC_5 is the same as CC_3, except for we select 50% of the nodes. For HDN_3 and HDN_5, we randomly select 30% and 50% of the high degree nodes, respectively. Lastly, SPC represents selecting one seed per community.

If the coverage percentage of a seeding methods is less than 100%, then the number of selected seeds is less than the number of ground truth communities, which leads to extracting less communities than the ground-truth. On the other hand, when the coverage is 250%, then the number of colors is up to 2.5x the number of ground-truth communities. Yet, initialization with too many colors is considered as overhead during the community detection process, as nodes have to keep track of many colors, while some of those colors are going to be redundant. As shown, CC_5 and HDN_5 achieve the highest coverage performance across all the datasets. It is interesting to notice that CC_5 achieves the best coverage for Youtube dataset, as Youtube contains a high percentage of isolated and small communities. Thus, members of those communities have low degree compared to other nodes in the network, so none of them were selected as seeds using HDN_3 and HDN_5 methods. However, HDN is preferable than CC as a seeding method, as it is computationally less expensive to compute, though we need to compute CC for each node only once.

Figure 7 shows the performance comparison of the seeding methods with best coverage in terms of detection accuracy. It is interesting to note that the HDN seeding methods consistently achieve the highest F1 score in all of datasets except for Youtube. Particularity, HDN_3 represents the best coverage initialization method with the lowest overhead.

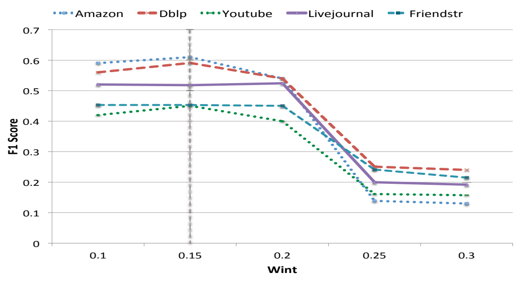


Fig. 8. Selecting threshold value for overlapping membership assessment.

Therefore, in the following experiments, we have used HDN_3 as our initialization method. Yet, for Youtube dataset, we have used CC_5.

2) Membership Assessment:

As aforementioned, in case of overlapping community detection, each node maintains a list of surrounding communities ordered by the volume of colors from the highest to the lowest. Then, nodes assess their membership according to the strength of $\mathcal{W}^{int}$ associated with each color. In particular, a node is a member to the community of color c if $\mathcal{W}^{int} > \alpha$. The higher the value of α is going to be used, the less the number of communities that are chosen by nodes during the membership assessment. Figure 8 shows the detection accuracy with different values of α. As shown, accuracy decreases when α is greater than 0.2, and this is due to the high overlapping percentage exist in the datasets as shown in Table III. The highest F1 score is achieved when $\alpha = 0.15$. Thus, in the following experiments, we set α's value to 0.15.

3) Stad vs. State-of-the-art approaches:

Figure 9 displays the composite performance of the methods. We use only the first letter of each method in the x-axis (i.e., S: Stad, I: Infomap, O:Oslom, L: Louvain, B: Bigclam, and D as regular Diffusion). We implement regular diffusion following the same decentralized vertex-centric approach used in Stad, yet with a fixed diffusion speed that equals to 1. Accordingly, nodes diffuse all the amount of colors they receive. Additionally, for regular diffusion, we use the same seeding and membership assessment methods implemented in Stad.

As shown, for each community detection method and each dataset we measure the average value of $F1$, B^3, NMI, and $Conducance$ evaluation metrics. Each evaluation metric has a score of 1, thus, the composite performance scale is up to 4 and the higher the better. The column of Infomap is missing for Friendstr dataset, as the algorithm did not converge for three weeks. Also, we find that Oslom and Bigclam result in high percentage of singleton communities. After convergence, around 25% and 40% of nodes in each dataset were left without decided memberships in Oslom and

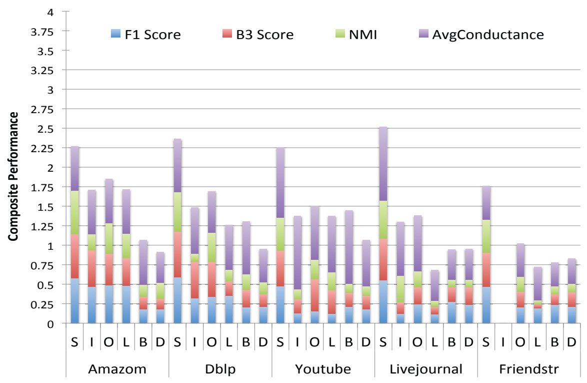


Fig. 9. Composite performance metric for different community detection algorithms.

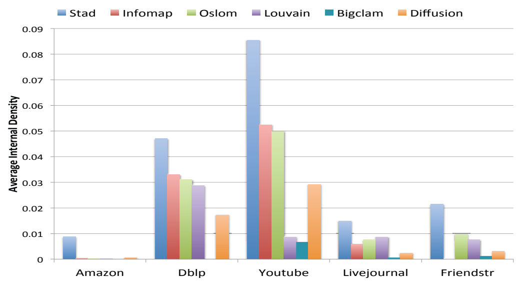


Fig. 10. Performance of different algorithms in terms of average internal density of extracted communities.

Bigclam, respectively.

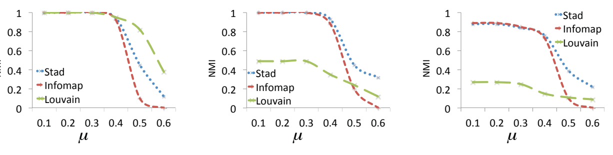
From the results, we observe that $F1$, B^3, and NMI metrics are correlated, though some small deviations exist among them. For AvgConductance, Stad achieves the value that is closest to the average conductance of the ground-truth communities. Other approaches have lower conductance values, which means that some of the ground-truth communities are merged into one single extracted community. Furthermore, Figure 9 depicts the average internal density of all methods. As shown, Stad achieves the highest values across all the datasets. In conclusion, Stad obtains the best detection accuracy, followed by Oslom and Bigclam. Broadly speaking, Stad achieves double the detection accuracy in terms of average F1 score.

C. Synthetic Datasets

Table IV summarizes the parameter settings we used for generating the synthetic datasets using LFR benchmark [30]. LFR benchmark is designed to reproduce certain topological properties observed in real-world networks: the size of the communities is power-law distributed, and so is the node degree. The mixing coefficient parameter μ allows controlling the average proportion of neighbors a node has in other communities. Therefore, μ has a big impact on the network topology, such that it controls percentage of inter-community edges. μ is usually set to be 0.1 or 0.3 and the detection accuracy usually decays for a larger μ. s is the range for community sizes, small size community =1,500 nodes and

TABLE IV
PARAMETERS FOR THE LFR BENCHMARK.

Parameter	Description
$n = 50k$	number of nodes in the graph
$\mu = [0.1, 0.6]$	mixing parameter
$d = 500$	the average degree of the nodes
$d_{max} = 1500$	the maximum degree of the nodes
$s = [1500, 10000]$	range of the community size for graphs
$\tau_1 = 2$	node degree distribution exponent
$\tau_2 = 1$	community size distribution exponent
$om = \{5, 10\}$	overlapping membership
$on = [10, 50]$	percentage of overlapping nodes



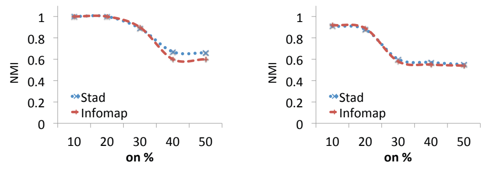
(a) Syn_sm (b) Syn_om_5 (c) Syn_om_10

Fig. 11. Performance of different algorithms using synthetic graphs in terms of NMI with different μ values.

large size community=10,000 nodes. Each node belongs to either one community or om overlapping communities, and the percentage of nodes in overlapping communities is specified by on. A larger om or on indicates more overlaps that are harder for the community detection tasks. Applying the described parameters, we get the following three network settings: **1) Syn_sm**: for networks with disjoint community membership, **2) Syn_om_5**: for overlapping membership networks with $om = 5$, and **3) Syn_om_10**: the same as previous setting except for $om = 10$.

As number of overlapping membership is known, we order communities according to their $\mathcal{W}^{int}$ and choose the top ones for each graph with a number equals to om. For each of the above network settings, we generate different graphs with the different μ values that range between 0.1 and 0.6. Furthermore, for each μ value we generate three different graphs and we report the average performance as detailed in the following experiments.

Figure 11 and Figure 12 illustrate the results of the analysis performed with synthetic graphs. As aforementioned, $F1$, B^3, and NMI metrics are correlated, as well as due space limitation and visualization simplicity we report the results of NMI. In the first experiment, we fix on percentage to 20% and vary the value of μ. In the second experiment, μ is fixed to 0.3 and we vary the percentage of overlapping nodes in communities. As shown in Figure 11, the trend is generally the same: for small values of μ and on, such that communities are well separated and most algorithms do a good job, so NMI is 1 or close to 1. When μ increases, as well as on, communities are more mixed and harder to detect, so NMI is quite different from 1, indicating that the communities extracted by the algorithms are sensibly different from the ones identified by the benchmark. From the results, we conclude that Stad achieves better performance followed by Infomap and Louvain.



(a) Syn_om_5 (b) Syn_om_10

Fig. 12. Performance of different algorithms using synthetic graphs in terms of NMI with different on percentages.

VII. CONCLUSION

In this paper we presented Stad that provides a novel flow-based community detection algorithm. Stad employs adaptive optimization function that allows it to detect communities more accurately in heterogeneous community size distributions. Specifically, Stad proposes a dynamic diffusion model that treats different flows independently and controls the diffusion speed of each flow separately. Furthermore, Stad is decentralized, scalable and highly parallel community detection algorithm with no global knowledge required. So as, Stad can be successfully applied in distributed environments, such as large-scale graph processing or decentralized machine learning. Results confirm Stad's ability to extract disjoint as well as multiple community membership(s) with double of the accuracy achieved by current state-of-the-art techniques using real-world datasets. As a natural extension of this work, we want to investigate applying Stad in weighted and directed networks, which in turn requires solving the challenges of the associated effect of edge directionality and dead-ends on the diffusion mechanism.

REFERENCES

[1] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.
[2] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.
[3] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 631–640.
[4] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 46–65, 2014.
[5] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.
[6] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, 2007.
[7] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 475–486.
[8] K. Lang and S. Rao, "A flow-based method for improving the expansion or conductance of graph cuts," in *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 2004, pp. 325–337.
[9] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.
[10] Y. Li, K. He, D. Bindel, and J. E. Hopcroft, "Uncovering the small community structure in large networks: A local spectral approach," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 658–668.

[11] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[12] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit," *PloS one*, vol. 7, no. 2, p. e32210, 2012.

[13] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi, "A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally," *Journal of Machine Learning Research*, vol. 13, no. Aug, pp. 2339–2365, 2012.

[14] K. He, Y. Sun, D. Bindel, J. Hopcroft, and Y. Li, "Detecting overlapping communities from local spectral subspaces," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 769–774.

[15] A. Guerrieri, F. Rahimian, S. Girdzijauskas, and A. Montresor, "Tovel: Distributed graph clustering for word sense disambiguation," in *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 623–630.

[16] F. Rahimian, S. Girdzijauskas, and S. Haridi, "Parallel community detection for cross-document coreference," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*. IEEE Computer Society, 2014, pp. 46–53.

[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[18] L. Bohlin, D. Edler, A. Lancichinetti, and M. Rosvall, "Community detection and visualization of networks with the map equation framework," in *Measuring Scholarly Impact*. Springer, 2014, pp. 3–34.

[19] S. Emmons, S. Kobourov, M. Gallant, and K. Börner, "Analysis of network clustering algorithms and cluster quality metrics at scale," *PloS one*, vol. 11, no. 7, p. e0159161, 2016.

[20] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PloS one*, vol. 6, no. 4, p. e18961, 2011.

[21] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 587–596.

[22] J. Šıma and S. E. Schaeffer, "On the NP-completeness of some graph cluster measures," in *Proceedings of the Thirty-second International Conference on Current Trends in Theory and Practice of Computer Science (Sofsem 06)*, vol. 3831. Springer, 2006, pp. 530–537.

[23] R. Clausius, *The mechanical theory of heat: with its applications to the steam-engine and to the physical properties of bodies*. J. van Voorst, 1867.

[24] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[25] L. Lovász, "Random walks on graphs," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1-46, p. 4, 1993.

[26] M. Zhong, K. Shen, and J. Seiferas, "The convergence-guaranteed random walk and its applications in peer-to-peer networks," *IEEE Transactions on Computers*, vol. 57, no. 5, pp. 619–633, 2008.

[27] L. Lovász and M. Simonovits, "The mixing rate of markov chains, an isoperimetric inequality, and computing the volume," in *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*. IEEE, 1990, pp. 346–354.

[28] F. Chung, "Random walks and local cuts in graphs," *Linear Algebra and its applications*, vol. 423, no. 1, pp. 22–32, 2007.

[29] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning," *arXiv preprint arXiv:0809.3232*, 2008.

[30] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[31] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.