Local Partition in Rich Graphs

Scott Freitas*, Nan Cao†, Yinglong Xia‡, Duen Horng (Polo) Chau* and Hanghang Tong§
Georgia Institute of Technology, Atlanta, GA*
Tongji University†, Shanghai, China
Huawei‡, Santa Clara, CA
Arizona State University§, Tempe, AZ
Email: safreita@gatech.edu, nan.cao@gmail.com, yinglong.xia@huawei.com, polo@gatech.edu, hanghang.tong@asu.edu

***Abstract*—Local graph partitioning is a key graph mining tool that allows researchers to identify small groups of inter-related nodes (e.g., people) and their connective edges (e.g., interactions). As local graph partitioning focuses primarily on the graph structure (vertices and edges), it often fails to consider the additional information contained in the attributes. We propose a scalable algorithm to improve local graph partitioning by taking into account *both* the graph structure and attributes. Experimental results show that our proposed ATTRIPART algorithm finds up to 1.6× denser local partitions, while running approximately 43× faster than traditional local partitioning techniques (PageRank-Nibble).**

***Keywords*-local partition, rich graph, attributes, conductance, subgraph, pagerank**

I. INTRODUCTION

Motivation. As network data is being generated at an unprecedented rate across multiple disciplines, a critical challenge before us is the translation of this large-scale network data into meaningful information. A key task in this translation is the identification of *local communities* with respect to a given seed node (we interchangeably refer to local community as a local partition). This community identification has applications to many important areas, including: recommender systems and ego-centric network identification. In practical terms, the information discovered in these local communities can be utilized in a wide range of high-impact areas, from protein interaction networks [1] [2] to social [3] [4] and transportation networks.

Problem Overview. How can we quickly determine the local graph partition around a given seed node? This problem is traditionally solved using an algorithm like Nibble [5], which identifies a small cluster in time proportional to the size of the cluster, or PageRank-Nibble, [6] which improves the running time and approximation ratio of Nibble with a smaller polylog time complexity. While both of these methods provide powerful techniques in the analysis of network structure, they fail to take into account the attribute information contained in many real-world graphs. Other techniques to find improved rank vectors, such as attributed PageRank [7], lack a generalized conductance metric for measuring cluster "goodness" containing attribute information. In this paper, we propose a novel method that combines the network structure and attribute information contained in graphs—to better identify local partitions using a generalized conductance metric.

Applications. Local graph partition plays a central role in many application scenarios. For example, a common problem in *recommender systems* is that of social media networks and determining how a local community will evolve over time. The proposed ATTRIPART algorithm can be extended (utilizing link prediction) to determine the evolution of local communities, which can then assist in user recommendations. Another example, utilizing social media networks, is *ego-centric network identification*—where the goal is to identify the locally important neighbors relative to a given person. To this end, we can use our ATTRIPART algorithm to identify better ego-centric networks using the graph's network structure and attribute information.

Contributions. Our primary contributions are three-fold:

1) The formulation of a graph model and generalized conductance metric that incorporates both attribute and network structure edges.
2) The design and analysis of local clustering algorithm ATTRIPART utilizing the proposed graph model, modified conductance metric and subgraph identification technique.
3) The evaluation of the proposed algorithms on three real-world datasets—demonstrating the ability to identify **1.6**× denser local partitions, while running approximately **43**× faster than traditional techniques.

Deployment. The local partitioning algorithm ATTRIPART is currently **deployed** to the PathFinder [8] web platform (www.path-finder.io), with the goal of assisting users in mining local network connectivity from large networks.

This paper is organized as follows—Section II defines the problem of local partitioning in rich graphs; Section III introduces our proposed model and algorithms; Section IV presents our experimental results on multiple real-world datasets; Section V reviews the related literature; and Section VI concludes the paper.

II. Problem Definition

In this paper we consider three graphs—(1) an undirected, unweighted structure graph $\boldsymbol{G} = (V, E)$, (2) an undirected, weighted attribute graph $\boldsymbol{A} = (V, E)$ and (3) a combined graph consisting of both $\boldsymbol{G}$ and $\boldsymbol{A}$ that is undirected and weighted $\boldsymbol{B} = (V, E)$. In each graph, V is the set of vertices, E is the set of edges, n is the number of vertices and m is the number of edges (i.e. $\boldsymbol{G}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ contain the same number of vertices and edges by default). In order to denote the degree centrality we say $\delta(v)$ is the degree of vertex v. We use bold uppercase letters to denote matrices (e.g. $\boldsymbol{G}$) and bold lowercase letters to denote vectors (e.g. $\mathbf{v}$).

For the ease of description, we define terms that are interchangeably used throughout the literature and this paper—(a) we refer to network as a graph, (b) node is synonymous with vertex, (c) local partition is referred to as a local cluster, (d) seed node is equivalent to query and start vertex, (e) topological edges of the graph refers to the network structure of the graph, (f) a rich graph is a graph with attributes on the nodes and or edges.

Having outlined the notation, we define the problem of local partitioning in rich graphs as follows:

Problem 1. Local Partitioning in Rich Graphs
Given: *(1) an undirected, unweighted graph $\boldsymbol{G} = (V, E)$, (2) a seed node $q \in V$ and (3) attribute information for each node $v \in V$ containing a k-dimensional attribute vector $\boldsymbol{x_i}$—with an attribute matrix $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}] \in R^{k \times n}$ representing the attribute vector for each node v.*
Output: *subset of vertices $S \subset V$ such that S best represents the local partition around seed node q in graph $\boldsymbol{B}$.*

Symbol	Definition
$\boldsymbol{G}$, $\boldsymbol{A}$, $\boldsymbol{B}$	network, attribute & combined graphs
n, m	number of nodes & edges in graphs $\boldsymbol{G}$, $\boldsymbol{A}$, $\boldsymbol{B}$
p, m_p	number of nodes & edges in $\boldsymbol{T}$
$\boldsymbol{s}$, q, ϕ_o	preference vector, seed node & target conductance
$\boldsymbol{W}$	lazy random walk transition matrix
S	set of vertices representing local partition
ϵ, ϵ_t	rank truncation and iteration thresholds
t_m, n_s	rank vector iterations; number of vertices to sweep
α_n, α_r	AttriPart & LocalProximity teleport values
t_e	edge addition threshold
t_s, n_w	subgraph relevance threshold & number of walks
$\boldsymbol{T}$; D, L	subgraph of $\boldsymbol{B}$; walk count dictionary & list
$\mu(L)$, $\sigma(L)$	mean and standard deviation of L

Table I: Symbols and Definition

III. Methodology

This section first describes the preliminaries for our proposed algorithms, including the graph model and modified conductance metric. Next, we introduce each proposed algorithm—(1) LocalProximity and (2) AttriPart. Finally, we provide an analysis of the proposed algorithms in terms of effectiveness and efficiency.

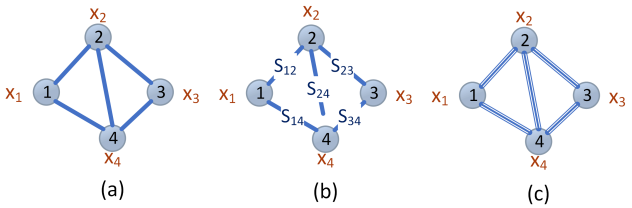


Figure 1: Example of the three graph models: (a) graph $\boldsymbol{G}$ is the network structure with nodes $\{1, 2, 3, 4\}$ and corresponding attribute set $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}\}$ given as input. (b) Graph $\boldsymbol{A}$ is the attribute network with the same set of edges as $\boldsymbol{G}$ with each edge (u, v) assigned a positive similarity weight s_{uv}. (c) Graph $\boldsymbol{B}$ is a linear combination of the each respective edge (u, v) from $\boldsymbol{G}$ and $\boldsymbol{A}$.

A. Preliminaries

Graph Model: Topological network $\boldsymbol{G}$ represents the network structure of the graph and is formally defined in Eq. (1). Attribute network $\boldsymbol{A}$ represents the attribute structure of the graph and is computed based on the similarity for every edge $(u, v) \in E$ in $\boldsymbol{G}$. In order to determine the similarity between the two nodes, we use Jaccard Similarity $J(u, v)$. $\boldsymbol{A}$ is formally defined in Eq. (2) where 0.05 is the default attribute similarity between an edge $(u, v) \in E$ in $\boldsymbol{G}$ if $J(x_u, x_v) = 0$. In addition, t_e is the similarity threshold for the addition of edges not in $\boldsymbol{G}$ where $0 < t_e \leq 1$. Combined Network $\boldsymbol{B}$ represents the combined graph of $\boldsymbol{G}$ and $\boldsymbol{A}$ and is formally defined in Eq. (3).

We define each of the three graph models $\boldsymbol{G}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ in Eq. (1), Eq. (2) and Eq. (3). Figure 1 presents an illustrative example.

$$\boldsymbol{G}(u, v) = \begin{cases} 1, & \text{if } (u, v) \in E \text{ and u}\neq\text{v} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\boldsymbol{A}(u, v) = \begin{cases} J(u, v), & \text{if } (u, v) \in E, \ u \neq v \text{ and } J(u, v) > 0 \\ 0.05, & \text{if } (u, v) \in E, \ u \neq v \text{ and } J(u, v) = 0 \\ J(u, v), & \text{if } (u, v) \notin E, \ u \neq v \text{ and } J(u, v) > t_e \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\boldsymbol{B}(u, v) = \begin{cases} 1 + \boldsymbol{A}(u, v), & \text{if } (u, v) \in E \text{ and } (u, v) \in \boldsymbol{A} \\ \boldsymbol{A}(u, v), & \text{if } (u, v) \notin E \text{ and } (u, v) \in \boldsymbol{A} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

***Conductance**:* Conductance is a standard metric for determining how tight knit a set of vertices are in a graph [9]. The traditional conductance metric is defined in Eq. (4), where S is the set of vertices representing the local partition. The lower the conductance value $\phi(S)$, where $0 \leq \phi(S) \leq 1$, the more likely S represents a good partition of the graph.

$$\phi(S) = \frac{cut(S)}{min(vol(S), vol(\bar{S}))} \tag{4}$$

Where the cut is $Cut(S) = \{(u, v) \in E | u \in S, v \notin S\}$, and the volume is $vol(S) = \sum_{v \in S} \delta(v)$.

This definition of conductance will serve as the benchmark to compare the results of our parallel conductance metric.

Parallel Conductance. We propose a parallel conductance metric which takes into account both the attribute and topological edges in the graph. Instead of simply adding the cut of each vertex $v \in S$, we want to determine whether v is more similar to the vertices in S or $\bar{S}$. The new cut and conductance metric is formally defined in Eq. (5) and Eq. (6), respectively. The *key* idea behind the parallel conductance metric is to determine whether each vertex in S is more similar to S or $\bar{S}$ using the additional information provided by the attribute links.

$$parallel\_cut(S) = \sum_{i \epsilon S} \frac{\sum_{j \notin S} \left[\boldsymbol{A}(i,j) + \boldsymbol{G}(i,j)\right]}{\sum_{j \epsilon S} \left[\boldsymbol{A}(i,j) + \boldsymbol{G}(i,j)\right]} \tag{5}$$

By definition, $\boldsymbol{B}$ can be split into its representative components, $\boldsymbol{G}$ and $\boldsymbol{A}$. We also note a few *key* properties of the parallel cut metric below:

1) $Parallel\_cut = 1$ means that the vertices in S have connections of equal weighting between S and $\bar{S}$.
2) $Parallel\_cut < 1$ means that the vertices in S have only a few strong connections to $\bar{S}$.
3) $Parallel\_cut > 1$ means that the vertices in S are more strongly connected to $\bar{S}$ than S.

Eq. (6) uses the cut as defined in Eq. (5) and the volume as defined above with the modification that $\delta(v)$ is a sum of it's components in $\boldsymbol{G}$ and $\boldsymbol{A}$.

$$\phi(S) = \frac{parallel\_cut(S)}{vol(S)} \tag{6}$$

We note that the parallel conductance metric has a different scale compared to the traditional conductance metric. For example, a conductance of 0.3 in the traditional conductance doesn't have the same meaning as a conductance of 0.3 in the parallel definition. We also bound the volume of S to $vol(S) < 1/2vol(B)$. This allows us to reduce the $min(vol(S), vol(\bar{S}))$ computation to $vol(S)$. A toy example of the parallel conductance can be seen in Figure 2.

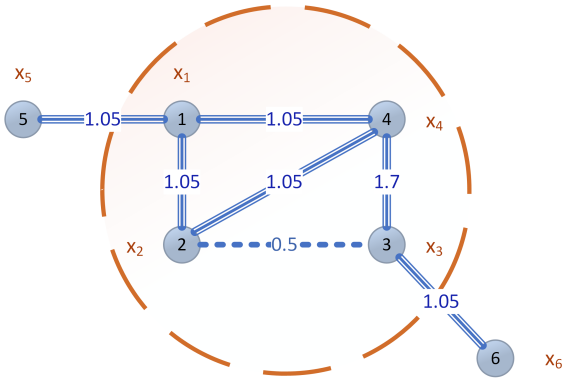


Figure 2: A toy example calculating the parallel cut and conductance with local partition S containing vertices $\{1, 2, 3, 4\}$. Parallel cut(V_1) = 1.05/2.1 = 0.5, parallel cut(V_2) = 0, parallel cut(V_3) = 1.05/2.2 = 0.477, parallel cut(V_4) = 0, parallel cut($Total$) = 0.5 + 0.477 = 0.977. Volume(S) = 12. Parallel conductance(S) = 0.977/12 = 0.0814.

B. Algorithms

We propose two algorithms in this subsection—(1) LocalProximity and (2) AttriPart. First, we introduce the LocalProximity algorithm as a key building block for speeding-up the AttriPart algorithm by finding a subgraph containing only the nodes and edges relevant to the given seed node. We then introduce the AttriPart algorithm to find a local partition around a seed node by minimizing the parallel conductance metric. In addition, we note that the algorithms are presented in matrix notation but implemented using NetworkX.

LocalProximity. The primary purpose of the LocalProximity algorithm is to reduce the computations required by AttriPart. We experimentally found that the PageRank vector utilized in the AttriPart algorithm is significantly faster to compute after running the proposed LocalProximity algorithm.

Algorithm Details. The goal is to find a subgraph $\boldsymbol{T}$ around seed node q, such that $\boldsymbol{T}$ contains only nodes and edges likely to be reached in n_w trials of random walk with restart. We base the importance of a vertex $v \in V$ on the theory that random walks can measure the importance of nodes and edges in a graph [10][11]. This is done by defining node relevance proportional to the frequency of times a random walk with restart walks on a vertex in n_w trials (nodes walked on more than once in a walk will still count as one). Instead of using a simple threshold parameter to determine node/edge relevance as in [10], we utilize the mean and standard deviation of the walk distribution in order for the results to remain insensitive of n_w given that n_w is sufficiently large. In conjunction with the mean and standard deviation, we introduce t_s as a relevance threshold parameter to determine the size of the resulting subgraph $\boldsymbol{T}$.

See section III-C for more details.

Algorithm Description. The LOCALPROXIMITY algorithm takes a graph $\boldsymbol{B}$, a seed node $q \in \boldsymbol{B}$, a teleport value α_r, the number of walks to simulate n_w, a relevance threshold t_s—and returns a subgraph $\boldsymbol{T}$ containing the relevant vertices in relation to q. This algorithm can be viewed in three major steps:

1) Compute the walk distribution around seed node q in graph $\boldsymbol{B}$ using random walk with restart (line 2). We omit the Random Walk algorithm due to space constraints, however, the technique is described above.
2) Determine the number of vertices to include in the subgraph $\boldsymbol{T}$ based on the relevance threshold parameter t_s, mean of the walk distribution list $\mu(L)$ and the standard deviation of the walk distribution list $\sigma(L)$ (lines 4-6).
3) Create a subgraph based on the included vertices (line 8).

Algorithm 1: Local Proximity

Input: Graph $\boldsymbol{B}$, seed node q, teleport value α_r, number of walks to simulate n_w, relevance threshold t_s

Result: Subgraph $\boldsymbol{T}$

1 subgraph_nodes = [];
2 D = RandomWalk(q, α_r, n_w, $\boldsymbol{B}$);
3 L = D.values;
4 **for** *vertex* u *in* $\boldsymbol{B}$ **do**
5 **if** $D[u] > \mu(L)$ + $\sigma(L)$ / t_s **then**
6 subgraph_nodes.append(u);
7 **end**
8 $\boldsymbol{T}$ = $\boldsymbol{B}$.subgraph(subgraph_nodes);
9 return $\boldsymbol{T}$;

ATTRIPART. Armed with the LOCALPROXIMITY algorithm, we further propose an algorithm ATTRIPART, which takes into account the network structure and attribute information contained in the graph to find denser local partitions than can be found using the network structure alone. The foundation of this algorithm is based on [5][6][12] with subtle modifications on lines 1, 4 and 9. These modifications incorporate the addition of a combined graph model, approximate PageRank computation using the LOCALPROXIMITY algorithm, and the parallel cut and conductance metric. In addition, ATTRIPART doesn't depend on reaching a target conductance in order to return a local partition—instead it returns the best local partition found within sweeping n_s vertices of the sorted PageRank vector.

Algorithm Description. Given a graph $\boldsymbol{B}$, seed node $q \in V$, target conductance ϕ_o, rank truncation threshold ϵ, the number of iterations to compute the rank vector t_{last}, teleport value α_n, rank iteration threshold ϵ_t and number of nodes to sweep n_s—ATTRIPART will find a local partition S around q within n_s iterations of sweeping. This algorithm can be viewed in five steps:

1) Set values for ϵ and t_{last} as seen in Eq. (7) and Eq. (9) respectively. We experimentally set $b = \frac{1+log(m)}{2}$ and ϵ_t to 0.01. For additional detail on parameters ϵ, t_{last} and b see [5]. For all other parameter values see Section IV.
2) Run LOCALPROXIMITY around seed node q in order to reduce the run time of the PageRank computations (line 1).
3) Compute the PageRank vector using a lazy random transition with personalized restart—with preference vector s containing all the probability on seed node q. At each iteration truncate a vertex's rank if it's degree normalized PageRank score is less than ϵ (lines 2-7).
4) Divide each vertex in the PageRank vector by its corresponding degree centrality and order the rank vector in descending order (line 8).
5) Sweep over the PageRank vector for the first n_s vertices, returning the best local partition S found (lines 9-10). The *sweep* works by taking the re-organized rank vector and creating a set of vertices S by iterating through each vertex in the rank vector one at a time, each time adding the next vertex in the rank vector to S and computing $\phi(S)$.

$$\epsilon = 1/(1800(l+2)t_{last}2^b) \tag{7}$$

$$l = \lceil log_2(2m/2) \rceil \tag{8}$$

$$t_{last} = (l+1)\lceil \frac{2}{\phi^2} ln(c_1(l+2)\sqrt{2m/2}) \rceil \tag{9}$$

Algorithm 2: AttriPart

Input: Graph $\boldsymbol{B}$, seed node q, target conductance ϕ_o, truncation threshold ϵ, iterations t_{last}, teleport value α_n, iteration threshold ϵ_t, vertices to sweep n_s

Result: Local partition S

1 $\boldsymbol{T}$ = Local_Proximity($\boldsymbol{B}$, q, α_r, n_w, t_s);
2 $\boldsymbol{D}_{i,i} = \delta(v_i)$;
3 $\boldsymbol{W} = \frac{1}{2}(\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{T})$;
4 **for** $t = 1$ *to* t_{last} ***and*** *sum($\boldsymbol{q}_t$) - sum($\boldsymbol{q}_{t-1}$)* $< \epsilon_t$ **do**
5 $\boldsymbol{q}_t = (1-\alpha)\boldsymbol{q}_{t-1}\boldsymbol{W} + \alpha s$;
6 $\boldsymbol{r}_t(i) = \boldsymbol{q}_t(i)$ if $\boldsymbol{q}_t(i)/d(i) > \epsilon$, else 0;
7 **end**
8 Order i from large to small based on $\boldsymbol{r}_t(i)/d(i)$;
9 Sweep Parallel_Conductance $\phi(S\{i = 1..j\})$ while $i < n_s$;
10 Return S with min $\phi(S_j)$;

C. Analysis

Effectiveness: LocalProximity (Algorithm 1). The objective is to ensure that all relevant nodes in proximity to seed node q are included. We use the fact that many real-world graphs follow a scale-free distribution [13] [14], with many nodes containing only a few links while a handful encompasses the majority. In Figure 3, we found that after running n_w trials of random walk with restart, a scale-free like distribution formed—with a large majority of the nodes containing a small number of 'hits', while a few nodes constituted the bulk.

As the number of random walks n_w is increased, the scale-free like distribution is maintained since each node is proportionally walked with the same distribution. We therefore need only some minimum value for n_w, which we set to 10,000. We use this skewed scale-free like distribution in combination with Eq. (10) below to ensure the extraction of relevant nodes in relation to a query vertex.

$$D(v) > \mu(L) + \sigma(L)/t_s \tag{10}$$

Mathematically we define node relevance based on Eq. (10), where D is a dictionary containing the walk count of each vertex and $D(v)$ represents the number of times vertex v is walked in n_w trials of the random walk with restart. L is a list of each node's walk count in the graph, $\mu(L)$ is the average number of times all of the nodes in the graph are walked and $\sigma(L)$ is the standard deviation of the number of times all of the nodes in the graph are walked. In section IV we discuss values of t_s that have been shown to be empirically effective.

After determining the relevant nodes we create a subgraph $\boldsymbol{T}$ from a portion of the long-tail curve as defined by threshold parameter t_s in conjunction with $\mu(L)$ and $\sigma(L)$. We say that subgraph $\boldsymbol{T}$ contains $p \ll n$ nodes—with p increasing nearly independently of the graph size (depending on threshold t_s). As seen in Figure 3, the number of nodes with r walks converges independent of graph size.

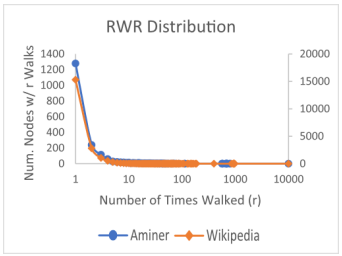


Figure 3: Random walk w/ restart—distribution of node walk counts. n_w = 10,000, α_r = 0.15; dataset: wikipedia, start vertex: 'ewok', y-axis: right; dataset: Aminer, start vertex: 364298, y-axis: left. We omit nodes walked zero times in the graph, however, they're used in calculating $\mu(L)$, $\sigma(L)$.

Efficiency: All algorithms use the same data structure for storing the graph information. If a compressed sparse row (CSR) format is used, the space complexity is $O(2m + n + 1)$. Alternatively, we note that with minor modification to the algorithms above we can use an adjacency list format with $O(n + m)$ space.

Overview. LocalProximity has a time complexity of $O(n + m_p + n_w)$ while AttriPart has a time complexity of $O(p^2 + pm_p + n + n_w)$.

Analysis. LocalProximity: There are three major components to this algorithm: (1) n_w random walks with walk length l for a time complexity of $O(n_w)$ (line 2). (2) Linear iteration through the number of nodes taking $O(n)$ (lines 4-7). (3) Subgraph $\boldsymbol{T}$ creation based on the number of included vertices p with node set V_t—requiring iteration through every edge of node $v \in V_t$ for m_p total edges. Iterating through every edge is linear in the number of edges for a time complexity of $O(m_p)$ (line 8). This leads to a total time complexity of $O(n + m_p + n_w)$

AttriPart: There are six major steps to this algorithm: (1) calling LocalProximity which returns a subgraph $\boldsymbol{T}$ containing p nodes and m_p edges for a time complexity of $O(n + m_p + n_w)$ (line 1). (2) Creating a diagonal degree matrix by iterating through each node in $\boldsymbol{T}$ with time complexity $O(p)$ (line 2). (3) Creating the lazy random walk transition matrix $\boldsymbol{W}$, which requires $O(m_p)$ from multiplying the corresponding matrix entries (line 3). (4) In lines 4-7 we iterate for t_{last} iterations, with each iteration (i) updating the rank vector by multiplying the corresponding edges in the transition matrix $\boldsymbol{W}$, with the rank vector $\boldsymbol{q}$ for a time complexity of $O(m_p)$ and (ii) truncating every vertex with rank $\boldsymbol{q}_t(i)/d(i) \leq \epsilon$ for a time complexity linear in the number of nodes in the rank vector $O(p)$. (5) Sort the rank vector which will be upper bounded by $O(plogp)$ (line 8). (6) Compute the parallel conductance, which takes $O(p^2 + pm_p)$ time (lines 9-10). Combining each step leads to a total time complexity of $O(p^2 + pm_p + n + n_w)$.

While AttriPart scales quadratically with respect to p, we note that in practice these algorithms are very fast since $p \ll n$ and p scales nearly independent of graph size as shown in section III-C.

IV. Experiments

In this section, we demonstrate the effectiveness and efficiency of the proposed algorithms on three real-world network datasets of varying scale.

A. Experiment setup

Datasets. We evaluate the performance of the proposed algorithms on three datasets—(1) the Aminer co-authorship network [15], (2) a Musician network mined from DBpedia and (3) a subset of Wikipedia entries in DBpedia containing both abstracts and links. All three networks are undirected with detailed information on each below:

- **Aminer.** Nodes represents an author, with each author containing a set of topic keywords, and an edge representing a co-authorship. To form the attribute network,

we compute attribute edges based on the similarity between two authors for every network edge, using Jaccard Similarity on the corresponding authors's topic set.

- **Musician.** Nodes represent a Musician, with each Musician containing a set of music genres, and an edge representing two Musicians who have played in the same band. To form the attribute network, we compute attribute edges based on the similarity between two Musicians for every network edge, using Jaccard Similarity on the corresponding artist's music genre set.
- **Wikipedia.** Nodes represent an entity, place or concept from Wikipedia which we will jointly refer to as an item. Each item contains a set of defining key words; with edges representing a link between the two items. The dataset originates from DBpedia as a directed graph with links between Wikipedia entries. We modify the graph to be undirected for use with our algorithms—which we believe to be a reasonable as each edge denotes a relationship between two items. In addition, this dataset uses only a portion of the Wikipedia entries containing both abstracts and links to other Wikipedia pages found in DBpedia. To form the attribute network, we compute attribute edges based on the similarity between two items for every network edge using Jaccard Similarity on the corresponding item's key word set.

Category	Network	Nodes	Edges
Aminer	Co-Author	1,560,640	4,258,946
Musician	Co-Musician	6,006	8,690
Wikipedia	Link	237,588	1,130,846

Table II: Network Statistics

Metrics. (1) To benchmark the LOCALPROXIMITY algorithm's effectiveness and efficiency, we compare (i) the difference between local partition created with and without the LOCALPROXIMITY algorithm on ATTRIPART and (ii) the run time and difference between the top 20 PageRank vector entries with and without the LOCALPROXIMITY algorithm. (2) To benchmark the ATTRIPART algorithm's effectiveness and efficiency we compare the triangle count, node count, local partition density and run time to a relaxed variation of PageRank-Nibble. Normally, PageRank-Nibble does not return a local partition if the target conductance is not met, however, we modify it to return the best local partition found—even if the target conductance is not met. This modification allows for more comparable results to ATTRIPART. In addition, we utilize a variation of PageRank-Nibble with loosened volume constraints on the returned partition.

Repeatability. All data and source code used in this research will be made publicly available. The Aminer co-authorship network can be found on the Aminer website [1]; the Musician and Wikipedia datasets used in the experiments will be released on the author's website. All algorithms and experiments were implemented using Python and NetworkX. In addition, we recommend the following parameter values for most applications: α_n = 0.2, α_r = 0.15, ϕ_o = 0.05, t_s = 2, n_w = 10,000, n_s = 200.

B. Effectiveness

LOCALPROXIMITY. In Figure 4 parts (a)-(c), we can see that the proposed LOCALPROXIMITY algorithm significantly reduces the computational run time, while maintaining high levels of accuracy across both metrics. Parts (a)-(b) demonstrate to what extent the accuracy of the results are dependent upon the parameter values. In particular, a low value of α_r (random walk alpha) and a high value of t_s (relevance threshold) are critical to providing high accuracy results.

In Figure 4 part (a), we measure accuracy as the number of vertices that differ between the local partitions w/ and w/o the LOCALPROXIMITY algorithm on ATTRIPART. A small partition difference indicates that the LOCALPROXIMITY algorithm finds a relevant subgraph around the given seed node and that the full graph is unnecessary for accurate results. In part (b), we define the accuracy of the results to be the difference between the set of top *20* entries in the PageRank vectors for the full graph and subgraph using the LOCALPROXIMITY algorithm. Overall, the results from part (b) correlate well to (a)—showing that for low values of α_r (random walk alpha) and high values of t_s (relevance threshold), their is negligible difference between the results computed on the full graph and the subgraph found using the LOCALPROXIMITY algorithm.

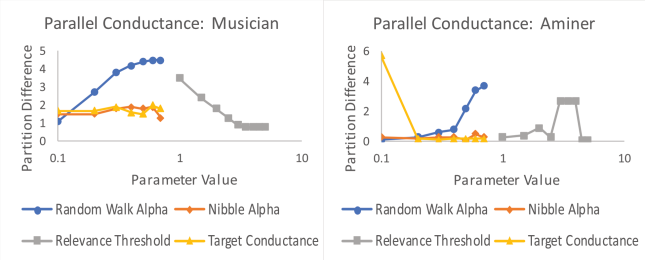
ATTRIPART. In Figure 5, we see that ATTRIPART finds significantly denser local partitions than PageRank-Nibble—with local partition densities approximately **1.6**×, **1.3**× and **1.1**× higher in ATTRIPART than PageRank-Nibble in the Aminer, Wikipedia and Musician datasets respectively. Density is measured as $\frac{2m}{n(n-1)}$ where m is the number of edges and n is the number of nodes.

In Figure 5, we observe that the triangle count of the ATTRIPART algorithm is lower than PageRank-Nibble in the Musician and Aminer datasets. We attribute this to the fact that ATTRIPART is finding smaller partitions (as measured by node count) and, therefore, there are less possible triangles. We also note that each triangle is counted three times, once for each node in the triangle. While no sweeps across algorithm parameters were performed, we believe that the gathered results provide an effective baseline for parameter selection.

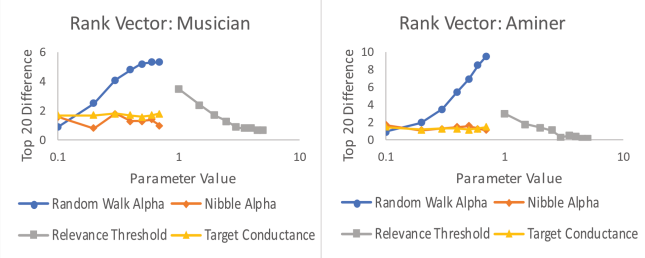
C. Efficiency

For both the proposed and baseline algorithms, the efficiency results represent only the time taken to run the

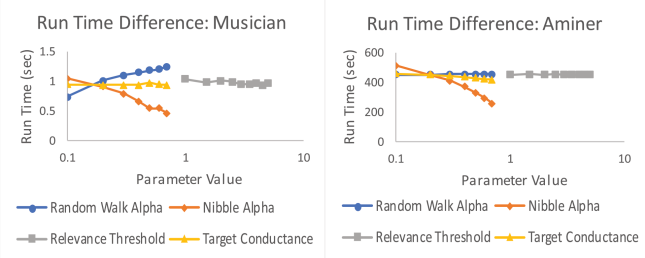
[1] https://Aminer.org/data



(a) Y-axis represents the difference in vertices between the local partition calculated w/ and w/o the LOCALPROXIMITY algorithm.



(b) Y-axis represents the # of vertices differing between the top 20 rank vector entries w/ and w/o the LOCALPROXIMITY algorithm.



(c) Y-axis represents the difference in run time between the PageRank calculation w/ and w/o the LOCALPROXIMITY algorithm.

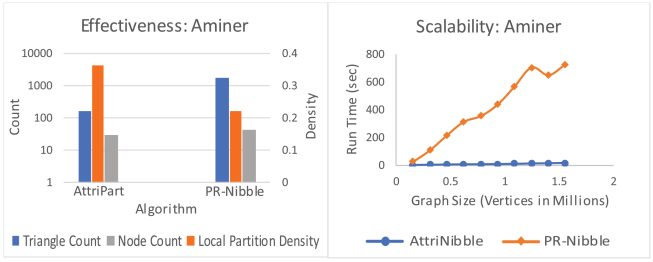
Figure 4: X-axis represents multiple parameter values. Each data point averages 10 randomly sampled vertices in both the Aminer and Musician datasets. Default parameters (unless sweeped across): α_n = 0.2, α_r = 0.15, ϕ_o = 0.2, t_s = 2, n_w = 10,000, n_s = 200. Parameter ranges: α_r, α_n and ϕ_o [0.1-0.7] in 0.1 intervals; t_s [1-5] in 0.5 intervals.

algorithm (e.g. not including loading data). **LOCALPROXIMITY.** Across a majority of the parameters the run time for the full graph PageRank computation is approximately 450 seconds longer compared to computing the PageRank vector based on the LOCALPROXIMITY sugraph. **ATTRIPART.** In Figure 5, we see that the ATTRIPART algorithm finds local partitions **43**× faster than PageRank-Nibble.

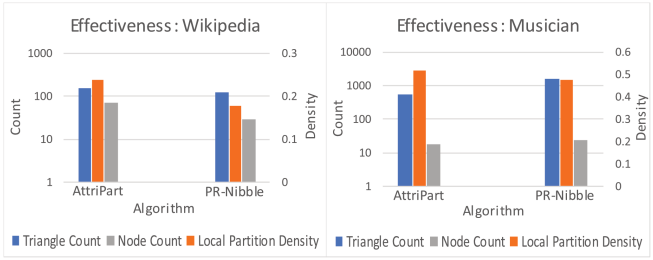
V. RELATED WORK

We provide a high level review of local community detection methods, with a focus on the research that pertains to the algorithms we propose in this paper.

Local Community Detection. Given an undirected graph, start vertex and a target conductance—the goal of Nibble is



(a) Scalability: Each data point represents the Aminer dataset in 1/10th intervals, with each point averaged over 3 randomly sampled vertices. Parameters: α_n = 0.2, α_r = 0.15, ϕ_o = 0.2, t_s = 2, n_w = 10,000, n_s = 200.



(b)

Figure 5: Effectiveness: results are averaged over 20 and 100 randomly sampled vertices in the Aminer/Wikipedia and Musician datasets, respectively. Parameters: α_n = 0.2, α_r = 0.15, ϕ_o = 0.05, t_s = 2, n_w = 10,000, n_s = 200.

to find a subset of vertices that has conductance less than the target conductance [5]. This algorithm has strong theoretical properties with a run time of $O(2^b(log^6 m)/\phi^4)$, where b is a user defined constant, ϕ is the target conductance and m is the number of edges. PageRank-Nibble builds on the work of Nibble by introducing the use of personalized PageRank [16], [17], in addition to a method for computing approximate PageRank vectors [6]. Since PageRank-Nibble and Nibble run on undirected graphs, they use truncated random walks in order to prevent the stationary distribution from becoming proportional to the degree centrality of each node [18]. There are also many alternative techniques for local community detection. To name a few, the paper by Bagrow and Bollt [19] introduces a method of local community identification that utilizes an l-shell spreading outward from a start vertex. However, their algorithm requires knowledge of the entire graph and is therefore not truly local. The research by J. Chen et. al. [4] proposes a method for local community identification in social networks that avoids the use of hard to obtain parameters and improves the accuracy of identified communities by introducing a new metric. In addition, the work by [20] and [21] introduces two methods of local community identification that take into account high-order network structure information. In

[20], the authors provide mathematical guarantees of the optimality and scalability of their algorithms, in addition to the generalization of it to various network types.

VI. Conclusion

This paper proposes a new algorithm for attributed graphs, with the goal of discovering denser local graph partitions. We believe that the proposed algorithm will be of particular interest to data mining researchers given the computational speed-up and enhanced dense local partition identification. The proposed local partitioning algorithm AttriPart is deployed to the web platform PathFinder (www.path-finder.io) and allows users to interactively explore the data presented in the paper.

VII. Acknowledgements

This work is supported by NSF (IIS-1651203, IIS-1715385, IIS-1743040, IIS-1563816 and DGE-1650044), DTRA (HDTRA1-16-0017), ARO (W911NF-16-1-0168), DHS (2017-ST-061-QA0001), NSFC Grants (61602306), Fundamental Research Funds for the Central Universities, and gifts from Huawei and Baidu.

References

[1] L. Bennett, A. Kittas, S. Liu, L. G. Papageorgiou, and S. Tsoka, "Community structure detection for overlapping modules through mathematical programming in protein interaction networks," *PLOS ONE*, 2014.

[2] S. L. Yong-Yeol Ahn, James P. Bagrow, "Link communities reveal multiscale complexity in networks," *Nature*, pp. 761–764, August 2010.

[3] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '07, 2007.

[4] J. Chen, O. Zaane, and R. Goebel, "Local community identification in social networks," in *2009 International Conference on Advances in Social Network Analysis and Mining*, July 2009, pp. 237–242.

[5] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 1–26, 2013. [Online]. Available: https://doi.org/10.1137/080744888

[6] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, Oct 2006, pp. 475–486.

[7] C.-C. Hsu, Y.-A. Lai, W.-H. Chen, M.-H. Feng, and S.-D. Lin, "Unsupervised ranking using graph structures and node attributes," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17. New York, NY, USA: ACM, 2017, pp. 771–779. [Online]. Available: http://doi.acm.org/10.1145/3018661.3018668

[8] S. Freitas, H. Tong, N. Cao, and Y. Xia, "Rapid analysis of network connectivity," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. New York, NY, USA: ACM, 2017, pp. 2463–2466. [Online]. Available: http://doi.acm.org/10.1145/3132847.3133170

[9] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, no. 3, pp. 497–515, May 2004. [Online]. Available: http://doi.acm.org/10.1145/990308.990313

[10] P. Dupont, J. Callut, G. Dooms, J. N. Monette, and Y. Deville, "Relevant subgraph extraction from random walks in a graph," 12 2017.

[11] M. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39 – 54, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378873304000681

[12] L. Zhukov. Structural analysis and visualization of networks. Youtube. [Online]. Available: https://www.youtube.com/watch?v=jIS5pZ8doH8&list=PLriUvS7IljvkBLqU4nPOZtAkp7rgpxjg1&index=11

[13] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. [Online]. Available: http://science.sciencemag.org/content/286/5439/509

[14] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '99. New York, NY, USA: ACM, 1999, pp. 251–262. [Online]. Available: http://doi.acm.org/10.1145/316188.316229

[15] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, J. Li, W. Luyten, and M.-F. Moens, "Fast and flexible top-k similarity search on large networks," *ACM Trans. Inf. Syst.*, vol. 36, no. 2, pp. 13:1–13:30, Aug. 2017. [Online]. Available: http://doi.acm.org/10.1145/3086695

[16] T. H. Haveliwala, "Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 784–796, July 2003.

[17] H. Tong, J. He, M. Li, W.-Y. Ma, H.-J. Zhang, and C. Zhang, "Manifold-ranking-based keyword propagation for image retrieval," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. Article ID 79 412, 10 pages, 2006, doi:10.1155/ASP/2006/79412.

[18] V. Grolmusz, "A note on the pagerank of undirected graphs," *Inf. Process. Lett.*, vol. 115, no. 6, pp. 633–634, Jun. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.ipl.2015.02.015

[19] J. P. Bagrow and E. M. Bollt, "Local method for detecting communities," *Phys. Rev. E*, vol. 72, p. 046108, Oct 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.72.046108

[20] D. Zhou, S. Zhang, M. Y. Yildirim, S. Alcorn, H. Tong, H. Davulcu, and J. He, "A local algorithm for structure-preserving graph cut," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 655–664. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098015

[21] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 555–564. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098069