A Separability Framework for Analyzing Community Structure

BRUNO ABRAHAO, SUCHETA SOUNDARAJAN, JOHN HOPCROFT,
and ROBERT KLEINBERG, Cornell University

Four major factors govern the intricacies of community extraction in networks: (1) the literature offers a multitude of disparate community detection algorithms whose output exhibits high structural variability across the collection, (2) communities identified by algorithms may differ structurally from real communities that arise in practice, (3) there is no consensus characterizing how to discriminate communities from noncommunities, and (4) the application domain includes a wide variety of networks of fundamentally different natures. In this article, we present a class separability framework to tackle these challenges through a comprehensive analysis of community properties. Our approach enables the assessment of the structural dissimilarity among the output of multiple community detection algorithms and between the output of algorithms and communities that arise in practice. In addition, our method provides us with a way to organize the vast collection of community detection algorithms by grouping those that behave similarly. Finally, we identify the most discriminative graph-theoretical properties of community signature and the small subset of properties that account for most of the biases of the different community detection algorithms. We illustrate our approach with an experimental analysis, which reveals nuances of the structure of real and extracted communities. In our experiments, we furnish our framework with the output of 10 different community detection procedures, representative of categories of popular algorithms available in the literature, applied to a diverse collection of large-scale real network datasets whose domains span biology, online shopping, and social systems. We also analyze communities identified by annotations that accompany the data, which reflect exemplar communities in various domain. We characterize these communities using a broad spectrum of community properties to produce the different structural classes. As our experiments show that community structure is not a universal concept, our framework enables an informed choice of the most suitable community detection method for identifying communities of a specific type in a given network and allows for a comparison of existing community detection algorithms while guiding the design of new ones.

Categories and Subject Descriptors: I.5.3 [**Computing Methodology**]: Pattern Recognition—*Clustering*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Class separability, community structure, detection algorithms, networks

ACM Reference Format:
Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. 2014. A separability framework for analyzing community structure. ACM Trans. Knowl. Discov. Data 8, 1, Article 5 (February 2014), 29 pages.
DOI: http://dx.doi.org/10.1145/2527231

1. INTRODUCTION

Community structure captures the tendency of entities in a network to group together in meaningful subsets whose members have a distinctive relationship to one another. The identification of these subsets allows for the analysis of networks at different levels

Support for this research is provided by AFOSR grants FA9550-09-1-0100 and FA9550-09-1-0675, a Microsoft Research New Faculty Fellowship, and a Google Research Grant.
Authors' addresses: B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg, Computer Science Department, Cornell University, Ithaca, NY 14850 USA; email: {abrahao, sucheta, jeh, rdk}@cs.cornell.edu.

DOI: http://dx.doi.org/10.1145/2527231

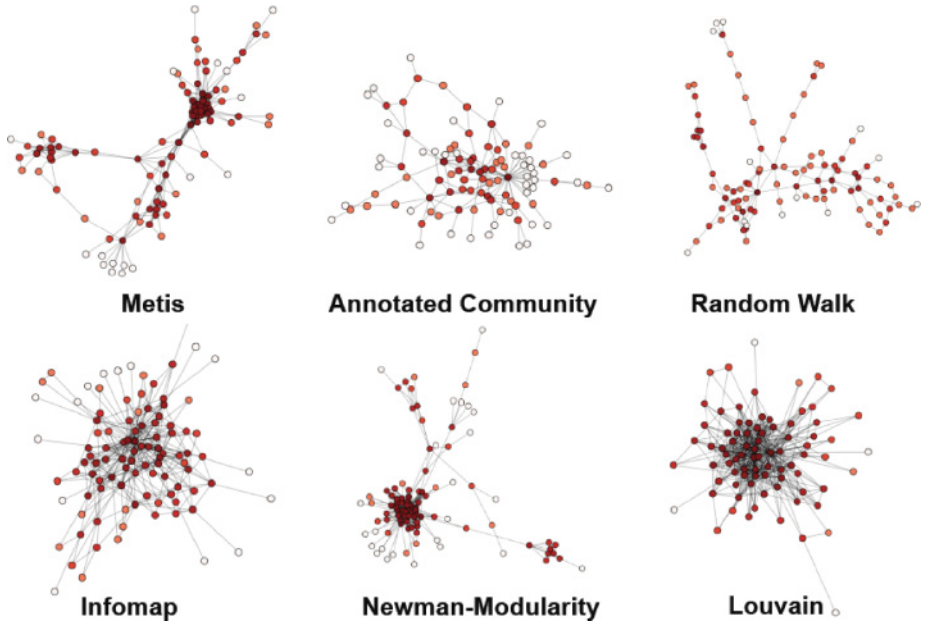

Fig. 1. Six different communities of 100 nodes each, identified on the LiveJournal network through different methods, namely Metis, Annotated Community, Random Walk, Infomap, Newman-Modularity, and Louvain. The communities comprise different node sets of the network, which were displayed by applying the same network layout algorithm. The visual diversity of the collection provides a rough and ready illustration of the structural variability that can be produced by the different methods. To aid the identification of structural nuances, the lightness of the red node colors reflect node degree, from fully illuminated (low degree) to dark (high degree).

of detail, which is instrumental in illuminating the structure underlying large-scale systems [Chung 1996; Fortunato 2010; Girvan and Newman 2002a; Newman 2004, 2006].

Despite playing a fundamental role in the structure and function of networks, community structure has proved to be frustratingly difficult to define, quantify, and extract. In addition to challenges related to computational tractability, four major factors account for the intricacies of community extraction.

First, the literature offers a multitude of disparate community detection algorithms. Due to differences in concept and design, the output of these procedures exhibits high structural variability across the collection. Given the diverse nature of networks, the notion of meaningful communities is necessarily context dependent, involving interpretations and expectations of domain experts. Therefore, many attempts to define communities are grounded on the notion of mathematical optimization. Starting with an a priori expectation about what a community should look like, researchers specify an objective function for a search problem whose solution provides the desired communities. This process has given rise to a large collection of community detection algorithms, each aiming at optimizing a particular objective function through a particular heuristic. As an illustration, Figure 1 shows six communities of size 100 extracted from the same network (the LiveJournal network, see Section 4.1), by different methods, namely Metis, Random Walk, Infomap, Newman-Modularity, and Louvain (described in Section 4.2.1) and one community of that network identified by data annotation. By visually inspecting these examples, we can see that even with this limited number of examples and despite the varying network layouts (which were a product of the

application of the same network layout algorithm to the subnetworks) the structural variability produced by the different methods is readily apparent. For example, some of the communities exhibit a dense, compact core, such as those extracted by the Infomap and the Louvain methods, whereas others spread along "tendrils" consisting of long paths of low-degree nodes that connect small clusters, such as the communities labeled as Metis, Random Walk, and Newman modularity. The annotated community, on the other hand, seems to posses a sparse core, including a larger number of low-degree nodes that spread on its periphery.

Second, communities in real networks often emerge as a result of multiple driving forces that make up the underlying complex system. Therefore, an attempt to capture community structure by maximizing a given objective function may represent an unrealistic expectation. As a consequence, communities identified by methods that reflect mathematical constructs may differ structurally from real communities that arise in practice.

Third, there is no established consensus on the question of what properties distinguish subgraphs that are communities from those that are not communities. Although we can examine examples of community structure (e.g., by asking experts to identify communities in a given domain), we can only characterize a population in the presence of negative examples. However, finding negative examples of community structure is a challenging task. Any other subset of nodes in the network that is not explicitly identified as a community is a potential negative example; however, in large networks, exhaustively enumerating all forms of negative examples is computationally intractable. Moreover, even if we could enumerate every other set in the network, we are still faced with the possibility that these seemingly negative examples could also be valid communities that were simply not identified by the expert.

Last, the application domain includes a wide variety of networks of fundamentally different natures. Each of these networks contains meaningful communities that may possess their own distinctive structural profiles.

In this article, we present a framework to tackle these challenges through a comprehensive characterization of community properties. By using different notions of communities as references, our methodology enables the analysis of community structure without requiring the identification of negative examples. Our method presents a scalable framework that enables researchers to assess the structural dissimilarity among the output of new and existing community detection algorithms, and between the output of algorithms and communities that arise in practice. The analysis may guide the design of novel community detection procedures. Given the significant structural variability among the output of different algorithms (which we established in our experimental analysis, introduced at the end of this section and discussed in Section 5), our framework serves as a tool for practitioners to decide on the most suitable algorithm for identifying communities of a specific structure in a given network. The intended structure can be specified by producing examples of communities that are either representative of real communities in the network or possess the particular structural features that we wish to find. Another dividend of our method is a way to organize the menagerie of community structures that the collection of algorithms in the literature exhibit. The framework exposes the tendencies produced by the different algorithms, which allows us to group those that behave similarly. By complementing the approach with a feature selection analysis, we are able to determine what graph-theoretical properties of a subgraph are the most discriminative of community signature and identify those properties that account for most of the biases of the different community detection algorithms. Finally, our approach can be used to study the consistency of the output of algorithms across different networks.

We frame our approach as a class separability problem, which is able to simultaneously handle a large number of classes of communities and a diverse set of structural properties. To this end, we specify a learning problem in which we map the distinct communities into a feature space, where the dimensions represent measures that characterize a community's link structure. The separability of classes provides information on the extent to which different communities come from the same (or fundamentally different) distributions of feature values.

The heart of our framework is the assessment of class separability. To this end, we use traditional methods in machine learning, such as Scatter Matrices [Theodoridis and Koutroumbas 2008]. In addition, to produce more fine-grained separability information, we propose quantifying class separability through the cross-validation performance of existing multiclass supervised classifiers, both parametric, namely Support Vector Machines (SVMs) [Vapnik 1998], and nonparametric, namely k-Nearest Neighbors (kNN) [Aha et al. 1991]. To study the most relevant properties to analyze communities, we employ a feature selection analysis using a correlation-based method [Hall 1999].

In this work, we consider communities defined in two different ways and extract different classes of communities that can be grouped into two main categories: intrinsically defined and extrinsically defined communities.

We define the first set of communities by properties intrinsic to their link structure. For our purposes, these are the sets that are output by community detection algorithms. Each class of intrinsically defined communities comprises a set of examples that a specific algorithm extracts. The separability of these classes demonstrates the extent to which different algorithms output structurally distinguishable subgraphs. A feature selection analysis can then be employed to highlight the properties that exhibit the highest degree of interclass variability, thereby making explicit the structural bias produced by different algorithms.

We also define communities by the context, the dynamics, or the function associated with the networks, but extrinsic to the link structure. We identify these communities through meaningful annotations provided with the datasets, such as explicit declaration of group membership, product categories, grouping by protein function, and so on. In this fashion, for each network, we form a class of extrinsically defined communities, henceforth called *annotated communities*. These communities enable a large-scale rigorous analysis of community detection methods. The separability of the class comprising annotated communities from the classes of intrinsically defined communities determines the extent to which community detection algorithms succeed in extracting subgraphs that are structurally comparable to the communities formed by nodes sharing extrinsic properties in common.

An important question that arises in our framework is how to select a collection of community classes that are independent enough to populate the feature space with enough diversity. A space with these properties provides a strong reference to analyze the structure of communities based on structural diversity that other notions of community exhibit. For this purpose, we propose the application of a pairwise Scatter Matrix measurement [Theodoridis and Koutroumbas 2008] for analyzing class similarity and determine which classes from a collection are redundant or independent. Depending on the application, redundant classes can be merged or be represented in a reduced collection by one representative.

We illustrate our approach with an experimental analysis, which reveals nuances of the structure of real and extracted communities. In our experiments, we furnish our framework with the output of 10 different community detection procedures, representative of categories of popular algorithms available in the literature, as well as with annotated data, which reflect exemplar communities in various domains. We comprehensively characterize these examples, which we extract from large-scale real network

data spanning diverse domains, such as biology, online shopping, and social systems, using a broad spectrum of community properties.

In our experimental analysis, we reach the following conclusions about the communities in question. First, for all networks, the strong cross-validation performance indicates that the different community detection algorithms produce fundamentally different structures that are separable on the feature space defined. Second, we observe that in nearly all cases, the annotated communities are structurally distinguishable from the output of all community detection algorithms. Nevertheless, a surprising outcome reveals that the structure of annotated communities bears closest structural resemblance to the output of simple procedures that encode little structure and were originally included in our framework as baseline procedures, such as random walk and breadth-first search. In addition, despite the diversity of the domains from which our networks are drawn, this observation applies to all of the networks, except two of them for which we have a small population size. Third, we show that the different community detection algorithms produce structures that are not only consistent within a network but are consistent even across networks. Finally, a small subset of the features is consistently observed as the most discriminative. This observation allows for a dimensionality reduction by a factor as large as 4, preserving an equivalent 10-fold cross-validation performance. The most discriminative features identify the graph-theoretical properties that account for most of the biases of the different algorithms. In addition to considering class structure and separability, we show that even though communities generated through the same method applied to different networks resemble one another in important ways, they also have significant differences. We demonstrate that when considering only one community detection method at a time, communities from different networks are highly separable. Even more interestingly, we show that when a classifier is trained on examples produced from only one specific community detection method, it can still identify which network other communities came from, even when those communities were produced through other methods.

This article is organized as follows: Section 2 discusses background information and related work. Section 3 presents an overview of our framework. Section 4 introduces the datasets we use, the algorithms we consider, and the measures we apply to construct the feature space. Section 5 describes the heart of our framework and presents an experimental analysis thereof. Next, Section 6 presents the structural tendencies of communities through a feature selection analysis. Section 7 contains results of our network separability experiments. Finally, Section 8 offers our concluding remarks.[1]

2. RELATED WORK

The work by Girvan and Newman [2002a] sparked a recent wave of interest in the notion of *community structure* as a decomposition of a network that reflects meaningful properties of the underlying system [Fortunato 2010]. Nevertheless, this area has its roots in the related problem of graph partitioning, whose initial contributions date back to the 1970s [Kernighan and Lin 1970].

As mentioned in the preceding section, the multitude of community structure definitions is a source of high variability between the output of different community detection algorithms. Among the objective functions introduced in previous work, the notion of *modularity* [Girvan and Newman 2002a] has become an influential one. Modularity assigns high scores to communities whose internal edges outnumber the ones established in expectation by a random-network model that preserves the degree distribution of

[1]Preliminary results appear in an earlier conference publication [Abrahao et al. 2012].

the original network. Another notion, inspired by electrical networks, is that of *conductance* [Chung 1996]. The conductance of a set S with complement S^C is the ratio of the number of edges connecting nodes in S to nodes in S^C by the total number of edges incident to S or to S^C (whichever number is smaller). The common theme underlying the preceding notions is the search for node sets that are internally cohesive and yet sparsely connected to the rest of the network. Therefore, these measures tend to penalize sets having a large number of edges crossing the set relative to the count of internal edges.

Communities in general, however, display features that modularity and conductance may not capture, such as a preponderance of links to the outside over internal links and an arbitrary degree of overlap. This fact is substantiated by an investigation of real networks revealing that they do not split well into low-conductance communities [Leskovec et al. 2008], as most networks are expander like [Hoory et al. 2006]. These considerations lead to the development of alternative definitions, such as (α, β)-*community* [Mishra et al. 2008], and algorithms, such as *Link Communities* [Ahn et al. 2010] and *Clique Percolation* [Palla et al. 2005].

Despite the vast literature on community detection, the works of Ahn et al. [2010] and Yang and Leskovec [2012], as well as ours, are among the few that attempt to analyze the structural resemblance between communities extracted by algorithms and annotated communities, which represent examples of meaningful communities in various domains.

Even though network analysts expect the output of the different algorithms to display dissimilar structural profiles due their conceptual diversity, the structural variability does not hinge simply on the choice of optimization problem. In most cases of interest, the search for a collection of node sets that maximize a given objective function is computationally intractable [Fortunato 2010]. Therefore, in an attempt to handle the massive scale of today's networks, popular methods of community detection rely on efficient heuristics. As a consequence, previous works have quantified a significant output variability among different approximation algorithms that aim at maximizing the exact same function [Lancichinetti and Fortunato 2009; Leskovec et al. 2010].

Coscia et al. [2011] provide an excellent survey of modern community detection algorithms, which proposes a useful categorization of the different methods based on their definition. However, the authors do not include an experimental analysis to estimate the output variability that algorithms in the same category exhibit, or an assessment of whether the structures produced by these algorithms are faithful to the communities that they aim to extract. Our work provides a framework that allows for a comparison between algorithms empirically, based on their behavior when applied to networks, without the need to understand their definition.

In the spirit of studying the structural variability exhibited by different algorithms, closest to ours is the work of Leskovec et al. [2008], which discusses properties of communities produced by multiple algorithms that aim at maximizing conductance. They consider the values of a handful of features—for example, set compactness and internal conductance—produced by different algorithms. In contrast, here we present the first study that is simultaneously comprehensive with respect to the diversity of structural properties, of domains, of algorithms, and of scale. To illustrate this point, we demonstrate the applicability of our approach through the analysis of a collection of different communities. We take account of a set of 36 features, measured from the output produced by 10 different community detection processes. We derive our results from a diverse collection of datasets from small- and large-scale networks arising from multiple domains.

3. FRAMEWORK OVERVIEW

The purpose of our framework is to assist researchers and practitioners in understanding the behavior of different community detection algorithms. Given the large collection of community detection algorithms in the literature, we expect that different methods might produce different outputs when given the same input. However, little is understood about the dissimilarities among the output of different algorithms, and as these methods optimize for different criteria and use different heuristics, they may indeed search for fundamentally different types of communities. How can we understand existing algorithms, and which methods should we use as comparisons against new algorithms?

To answer this question, we need to understand what communities are and what properties they possess. However, it is not clear that the community detection problem is well defined, and different people may have very different notions of communities. Moreover, the concept of community may not only vary from individual to individual but also may be context and domain specific. For example, there is no reason to expect that the communities in a social network would resemble the structure of those in biological networks.

To address these issues, we can analyze real communities from different domains as identified by domain experts, and by looking at these examples, we may attempt to determine what properties they have. To identify these properties, one might exhaustively enumerate all possible forms of noncommunities and compare these sets against known communities. However, finding negative examples of community structure is a challenging task. Any other subset of nodes in the network that is not explicitly identified as a community is a potential negative example; therefore, in large networks, exhaustively enumerating all forms of negative examples is computationally intractable. Moreover, even if we could enumerate every other set in the network, we are still faced with the possibility that these seemingly negative examples could also be valid communities that were simply not identified by the expert.

The traditional statistical characterization task demands a training set where we distinguish communities from noncommunities. Given such data, we could then train a binary classifier to distinguish between the two structures and use the output of community detection algorithms as the test set. The classifier would then tell us the extent to which the output of some community detection algorithm resembles communities or noncommunities. From this model, we would be able to extract the exclusive features that communities possess and would better understand how well the algorithms capture these properties, which can be used as a performance measure. However, as discussed in the preceding paragraph, setting up this experiment is nontrivial due to the absence of negative examples.

Our contribution in this work is to provide a way to overcome these challenging obstacles in characterizing community structure by applying machine learning techniques in unorthodox ways. The key idea in our approach is to eliminate the requirement of presenting the classifier with negative examples by learning the structural distinction among known community notions. We build a feature space that allows us to model the problem as a separability framework [Fatemi-Ghomi et al. 1999; Theodoridis and Koutroumbas 2008] by extracting a comprehensive set of features—that is, community properties, using graph-theoretical concepts, from each example of community that we have extracted. These examples of communities, which are labeled with the name of the method used to identify them in networks, are then projected onto a feature space. This experiment allows for any outcome between two extreme scenarios. The different notions of communities that we want to study may encode and capture similar enough structures so that the different classes would be hard to separate in a feature space,

or at the other extreme, we might observe a clear separation of the classes in the feature space. The extent to which the classes are separable can be used as a measure of dissimilarity among the different community notions considered. Given this model, we can additionally assess the structural similarities of real communities with the communities produced by some algorithm. Here we use the diversity of community structures exhibited by different processes as references to understand other community structures. This powerful framework allows us to assess the structural differences among different notions of communities as well as to structurally assess the quality of the output of algorithms with respect to real communities. Furthermore, we are able to concurrently consider a broad spectrum of structural features in a scalable way.

In Section 4, we discuss how we build structural classes to use in the training and test sets of the separability framework. In Section 5, we discuss how to set up the separability experiments to answer the questions in which we are interested, such as measuring the extent to which different structural classes, each containing labeled examples of communities extracted by different algorithms (one structural class per algorithm), are separable. Furthermore, the distance, in terms of interclass separability and intraclass dispersion, between the classes corresponding to the output of community detection algorithms and the examples of real communities tells us which algorithms produce communities that most closely resemble real communities. We also discuss existing measures of class separability and propose the cross-validation performance of classifiers as a measure. Finally, Section 6 concludes the presentation of our framework with a method for reducing the dimensionality of the data to reveal the most discriminative features on which the different algorithms load their biases. From this analysis, we can compare the behavior of different algorithms in terms of a few structural features.

4. BUILDING STRUCTURAL CLASSES

Our main goal is to capture the structural signature exhibited by different communities. We do not know a priori the extent and the source of structural variability among seemingly different communities. In addition, to answer the research questions posed in the preceding section, we want to relate specific structural properties with the methods that tend to produce them. To address this question, we divide the space of known community examples into classes, each corresponding to the method that generated or identified the instances therein contained. As we expect that the structural tendencies of a given community identification method are going to be reflected by the examples in its corresponding class, we refer to these classes as *structural classes*. This denomination does not imply that the classes have distinctive structures. In fact, some classes may be compact, exhibiting a clear signature, whereas others may exhibit high variance that could be difficult to characterize. Moreover, multiple classes may overlap, indicating that they contain somewhat similar structures. Accordingly, the purpose of the class separability measures proposed here is to assess the structural dissimilarity among the structural classes that we consider.

Before describing our framework and delving into our analysis, in this section we present the datasets that we use, as well as our methodology for building structural classes of communities from the network data. We also describe the process of projecting the communities that we extract onto a feature space, which allows us to treat the question of class dissimilarity as a learning problem.

4.1. Datasets

We analyze eight large-scale datasets, namely LiveJournal; DBLP; two portions of the Facebook network (denoted by Facebook–Rice University Undergraduate [Ugrad] and Graduate [Grad]); Amazon; and three biological networks denoted by HS, SC, and Fly.

The collection encompasses different forms of entities and relationships originating from diverse domains.

The LiveJournal dataset consists of a snapshot of a large network of bloggers, previously explored by Backstrom et al. [2006]. The snapshot includes 4,847,571 bloggers who explicitly declare their friendship links. Due to the massive size of this dataset, we consider two portions of it, which we obtain by starting at a random node and performing a breadth-first search from that node. The datasets, henceforth named LJ1 and LJ2, contain 500,000 nodes each. LJ1 and LJ2 contain 10,736,588 and 10,640,429 edges, respectively.

DBLP data is publicly collectible, and our dataset consists of a snapshot taken in May 2009 of the online publication's database site DBLP. The data include a collection of editions of publication venues (i.e., conferences and journals) in computer science. A pair of the 744,386 authors present in the dataset are linked if they have coauthored at least one paper in any of the venues.

Facebook–Rice University Ugrad and Grad are an anonymized portion of the Facebook network that includes Rice University students, collected by crawling public friends lists on Facebook on May 17, 2008. They consist of two disjoint sets of 1,220 undergraduate students and 503 graduate students, respectively. Mislove et al. [2010] present a detailed description of these datasets.

The Amazon dataset [Leskovec et al. 2006] is a product copurchasing network from the online retailer Amazon.com. Each node represents a book, and an edge exists between two nodes if one was frequently purchased with the other. The network contains 270,347 nodes and 741,142 edges. For each book, Amazon.com reports up to five other items that were frequently purchased with the book.

Biological networks HS, SC, and Fly describe protein-protein interactions for *H. sapiens* (human), *S. cerevisiae* (a type of yeast), and *Drosophila* (a fruit fly species) [Park et al. 2011], respectively. In these networks, a node represents a protein, and two nodes are connected if scientific evidence of their interaction exists. HS contains 10,298 nodes and 54,655 edges; SC contains 5,523 nodes and 82,656 edges; and Fly contains 15,326 nodes and 486,970 edges.

4.1.1. Annotated Communities. The networks that we analyze contain annotations reflecting examples of communities that arise in these domains.[2] Some of these sets are user defined (i.e., users explicitly declare their participation in the community) whereas others reflect contextual information of the underlying process or organization (e.g., university department, protein function, product category). Next we describe how we identify and clean the annotated communities for each dataset.

For the social networks, LiveJournal users explicitly declare their membership in zero or more communities created and administered by users. In DBLP, conferences where authors publish their research work reflect the community memberships. Finally, for Facebook–Rice University Ugrad and Grad, users who possess common academic attributes, such as department, major, or dormitory, form the communities. These attributes were obtained by matching Facebook names with student records from the university's directory [Mislove et al. 2010].

For each item in Amazon.com, the online store provides several product categories, such as Photo Essays or Landscape Architecture Textbooks. We identify a set of nodes possessing a common categorical label as a community.

For HS, SC, and Fly, a number of proteins (although not all) have annotations regarding one or more gene ontology IDs describing the known functions that the protein

[2]These communities, however, may not represent an unbiased sample of communities in these networks, as other communities that are not annotated might also exist.

serves (e.g., metabolic regulation). We use these gene ontology values to identify the communities.

Because we define annotated communities by identifying sets of nodes with common labels, we sometimes encounter annotated communities containing multiple components. Considering disconnected communities would produce a less informative feature space, because some metrics achieve extreme values, such as infinity, for disconnected subgraphs (shortest paths, diameter, and so on). In these cases, rather than discarding the communities, we simply consider each component to be a separate community. Therefore, to capture the community information implicit in the annotations, we consider each connected component of the graph induced by a node set possessing a common label as an annotated community by itself.

Earlier experiments suggest that these structural features are not well represented in small communities, as the extracted features of these communities exhibit high correlation—that is, the intrinsic dimensionality of their feature space is too low to represent the structural variability that we want to observe. Therefore, it was our early decision to establish a cutoff point for the community size, below which the objects are too small to be representative. In the other extreme are the communities that are too large and sparse that make the features distorted in meaningless ways. Therefore, we filtered out small communities with fewer than 10 members and large communities with more than 1,000 members.

Overall, we identified 29,955 annotated communities for LJ1; 39,598 for LJ2; 10,595 for DBLP; 24 for Rice Grad; 41 for Rice Ugrad; 9,439 for Amazon; 64 for HS; 76 for SC; and 54 for Fly.

4.2. Structural Classes and Feature Space

In this section, we describe how to produce examples that constitute the structural classes and how to build the feature space for our learning framework. The process consists of two steps. First, we produce the examples by applying community detection algorithms, one for each class, to the network data. Second, we extract features by measuring a broad spectrum of properties of the subgraphs induced by communities. This latter step uses a set of examples consisting of the output produced in the previous step along with the set of annotated communities.

4.2.1. Producing the Examples. To illustrate the study of classes representing intrinsically defined communities, we selected a collection of 10 community detection procedures. We applied these procedures to each of the nine network datasets to extract examples of subgraphs produced by these methods. We labeled examples with the identity of the community detection procedure that produced them. In total, for each network, we created 11 structural classes of communities: one class of extrinsically defined communities, which comprises examples of annotated communities, and each of the other 10 classes corresponding to intrinsically defined communities, which comprise examples extracted by each of the 10 community detection algorithms, respectively. Figure 2 presents a graphical illustration of this process.

Without any assumptions on the structures that the algorithms produce, we chose our collection with the goal of including algorithms that are representative of strategies employed by a broad range of algorithms in the literature, purely based on their description. This description-based diversity is reflected in the categorization of Coscia et al. [2011]. Next, we briefly describe the community detection procedures that we consider, together with some examples of algorithms in the same description-based category according to Coscia et al. [2011].

(1) **Breadth-First Search (BFS):** To establish a baseline, we use breadth-first search to extract sets that serve as examples of random connected communities.

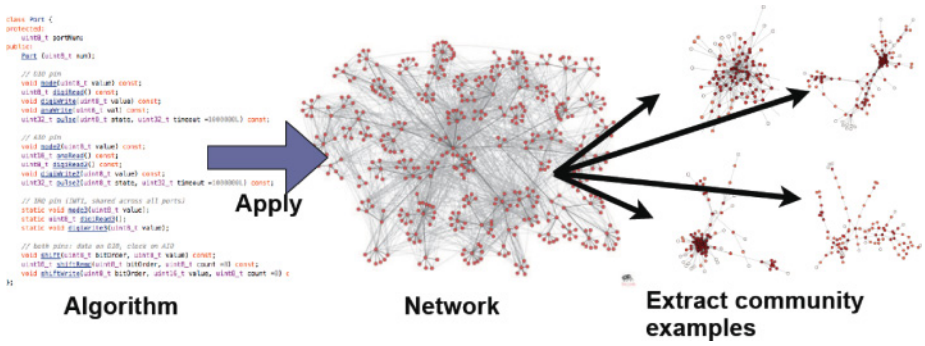

Fig. 2. The process of extracting examples of communities. We apply a given algorithm to a network to extract examples of typical structures that it produces (i.e., the communities extracted as its output). The set of examples extracted is further annotated with the name of the algorithm that produced them.

To create one BFS community of size k, we begin with a randomly selected node and perform a breadth-first search from that node until we visit k elements.

(2) **Random Walk 0 (RW0):** The central idea in many community detection algorithms is that random walks tend to concentrate within a community [Pons and Latapy 2006; Weinan et al. 2008]. To create communities of size k, we begin with a random node and perform a uniformly random walk from that node until k different nodes are visited. This method represents a way to extract a connected community that encodes little structure and serves as another baseline procedure.

(3) **Random Walk 0.15 (RW15):** This is similar to the preceding method with the twist that at each step we restart the walk from the starting node with 0.15 probability. RW15 concentrates the random walk distribution around a center, thereby forming more compact sets, whereas RW0 communities tend to spread out.

(4) **(α, β) (AB):** An (α, β)-community, for $\alpha < \beta$, requires every member of the community to be connected to at least β other members, whereas nonmembers have at most α links to the community [Mishra et al. 2008]. This definition allows for overlapping communities whose out-links may outnumber the in-links. To produce an (α, β)-community of size k, we produce a BFS community of size k and then apply a limited number of sequential node swaps that aim at making the set an approximate or exact (α, β)-community. In each step, we remove the community node with the fewest member neighbors and add the fringe node with the most member neighbors. The $\alpha - \beta$ algorithm can be considered an example of the class of algorithms that search for sets of nodes that meet some specific structural criteria. Other algorithms that find communities meeting specific structural criteria are Clique Percolation [Palla et al. 2005], S-Plexes Enumeration [Komusiewicz et al. 2009], Bi-Clique [Lehmann et al. 2008], and EAGLE [Shen et al. 2009].

(5) **Link Communities (LC):** In contrast to the majority of the available algorithms, Link Communities [Ahn et al. 2010] aims at addressing the overlapping and hierarchical nature of community structure by treating communities as groups of links rather than nodes, defining a similarity function on edges based on their shared node neighborhoods, and then using hierarchical single-linkage clustering to identify communities of edges. We extract examples of this structure by applying a standard implementation of this algorithm to our networks. We include Link Communities as a representative of algorithms that cluster links rather than nodes. Other algorithms in this category include the Link Modularity [Evans and Lambiotte 2009] and Link Maximum Likelihood [Ball et al. 2011] algorithms.

(6) **Infomap (IM):** The Infomap algorithm [Rosvall and Bergstrom 2011] views the problem of finding communities as akin to the problem of a mapmaker deciding on a level of granularity. The communities and the nodes therein have names. A random walk in the network is described by appending the community name followed by the name of nodes visited while in the community to a transcript. The goal is to find the community structure that minimizes the expected length of the description. Intuitively, such a structure would cause random walks to rarely escape communities. Infomap is a representative of algorithms that define communities as a group of nodes that are closer to each other than to nodes outside the community with respect to the number of hops between two nodes. Other examples of these approaches are Walktrap [Pons and Latapy 2006] and DOCS [Wei et al. 2009].

(7) **Louvain:** The Louvain method [Blondel et al. 2008] is a popular method for greedy modularity optimization. The algorithm consists of iteratively aggregating nodes into communities whenever this move locally improves modularity. The process outputs communities when no further merge produces a significant gain in modularity. The Louvain method is a classic example of a community detection method that optimizes for internal community density. Other such algorithms include the MetaFac [Lin et al. 2009], Variational Bayes [Hofman and Wiggins 2008], $LA \rightarrow IS^2$ [Baumes et al. 2005], and Local Density [Schaeffer 2005] algorithms.

(8) **Newman-Clauset-Moore (Newman):** This method is another example of greedy modularity maximization [Clauset et al. 2004]. Unlike the Louvain method, which considers merges that locally improve modularity, Newman-Clauset-Moore identifies a hierarchical community structure from which communities are extracted by cutting the dendrogram that reflects the hierarchy at the level that maximizes a global value of modularity. As with the Louvain method described earlier, the Newman-Clauset-Moore method for modularity optimization is an example of an algorithm that attempts to find communities with high internal density.

(9) **Markov Clustering Algorithm (MCL):** MCL [Dongen 2008] is a random walk–based method. It consists of two alternating steps. It begins with the random walk matrix of a graph (the normalized adjacency matrix). The first step, namely *expansion*, squares this matrix; this corresponds to computing the flow between clusters. The second step, called *inflation*, squares each element of the matrix individually, and then renormalizes; this step corresponds to increasing the strength of intracommunity ties. This process converges to a stationary matrix with several connected components, which the algorithm outputs as the communities. MCL is a member of the class of community detection algorithms that also includes Infomap.

(10) **Metis:** Metis [Karypis and Kumar 1998] is a graph partitioning method that is a variation of the Kernighan-Lin algorithm [Kernighan and Lin 1970]. Metis partitions a node-weighted network into a specified number of equal weight sets while minimizing the number of edges between the sets. Here we used a version of Metis that we adapted for finding high-conductance sets. The original Metis algorithm can be considered as representative of the class of 'bridge detection' community detection algorithms, which identify communities by removing bridges between dense sections of the network. Other algorithms in this class include Edge Betweenness [Girvan and Newman 2002b], CONGA [Gregory 2008], L-Shell [Bagrow and Bollt 2005], and Internal-External Degree [Lancichinetti et al. 2009]. Additionally, because we modify the algorithm to identify high-conductance sets, it might also be considered as part of the class of algorithms that optimize for internal community density.

As we are interested in the structural signature produced by the different methods, we run the parametrized algorithms multiple times for each network, randomly varying the parameter settings. Some of the procedures are nondeterministic and generate different communities at each run, even if the same parameters are used. To preserve uniformity with the set of annotated communities, in the process of generating examples, we discard communities of size less than 10 or greater than 1,000, as we are interested in the structure of reasonably sized communities. (See Subsection 4.1.1 for the details.) We also filtered out communities that contain multiple components, which are rarely extracted by the methods we used. The number of examples extracted varies among the procedures.

However, we inherit sensitivity to class imbalance from the methods we use for class separability. The effects of class imbalance on the performance of machine learning methods is the subject of an extensive literature in machine learning. For our experiments, we have applied standard methods in the literature for minimizing the problem. For an overview, we refer the reader to the work of Chawla [2005]. More specifically, we undersample the large classes and to a lesser extent oversample small classes to reduce this source of bias. Naturally, undersampling can result in some portions of a network (particularly a large network) not being represented in our final set of communities. Table II contains the average number of communities from each class to which each node belongs after under- or oversampling. For the small networks (Grad, Ugrad, HS, SC, and Fly), we sample 100 communities from each class, and for the larger networks, we sample 1,000 communities from each class. If a community detection method tends to produce a large number of small communities, and we then undersample this set, we will see small average values, whereas if a method produces a small number of large communities that we oversample, we will see higher average values.

Our algorithm selection has the purpose of illustrating the applicability of our framework. The approach, however, is not limited to the list that we consider. Our method scales to a large number of classes, and a collection of classes should include enough information to reflect the analysis intended. As discussed in the next section, a pair of classes may be highly correlated to each other (e.g., RW0 and RW15). As a result, they may split the predictions in such a way as to obfuscate the interpretation of the outcome. To avoid this pitfall, in Section 5.2 we consider an interclass correlation analysis to assess the independence of an algorithm collection.

4.2.2. Feature Extraction. In the feature extraction phase, we measure the subgraphs induced by the communities produced in the previous step and those induced by annotated communities. We use a large spectrum of measurements that cover many properties of both the internal link structure and the external interaction of the community with the rest of the network. Each measurement corresponds to a dimension of our feature space. Table I lists the features and describes their corresponding measures.

Most of the features can be understood from their table description. The feature Information Centrality, however, deserves further explanation. This measure captures a node's degree of centrality as a function of how fast its information can sequentially reach every other node in the network. For a given node, the information centrality computes a harmonic mean of the amount of "signal" that a node receives from other nodes. A signal between two nodes corresponds to a path between them, which varies according to the "noise," instantiated here as the path length [Stephenson and Zelen 1989].

By measuring the structural properties described in Table I for each example of a community derived in the previous phase, we obtain 11 classes of labeled examples in feature space, which constitute the input in our framework.

Table I. List of Features Corresponding to Measures of the Subgraphs That Communities Induce

#	Feature	Description
1	n	Number of nodes
2	m	Number of edges
3	Diameter	Greatest distance between two nodes by traversing shortest paths
4	Edge Density	Ratio of m to the maximum possible number of edges
5	Conductance	Ratio of m to the sum of the total degrees of the n nodes, including edges to rest of the network
6	Transitivity	Ratio of the number of three-node cycles (triangles) to the number of two-hop paths (open triangles)
7	Triangle Density	Ratio of the number of three-node cycles (triangles) to the number of possible node triples
8–11	Shortest Path	All-pairs shortest paths. The features are the three quartiles and the maximum.
12–15	Edge Betweenness	For each edge, fraction of all-pairs shortest paths that include that edge. The features are the three quartiles and the maximum.
16–20	Node Betweenness	For each node, fraction of all-pairs shortest paths that include that node. The features are the minimum, the three quartiles, and the maximum.
21–25	α	For each nonmember on the fringe of the community, number of members to which this node is connected. The features are the minimum, the three quartiles, and the maximum.
26–30	β	For each member, number of other members to which this node is connected. The features are the minimum, the three quartiles, and the maximum.
31	Treesum	Total number of spanning trees of the community graph, divided by the total number of spanning trees of a K_n-clique (computed using Kirchoff's matrix tree theorem [Lyons and Peres 2012])
32–36	Information Centrality	For each node, its Stephenson and Zelen's information centrality index [Stephenson and Zelen 1989]. The features are the minimum, the three quartiles, and the maximum.

Table II. Average Number of Communities from Each Class to Which Each Class Belongs after Sampling

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
BFS	9.9	5.3	0.9	1.0	0.2	0.2	0.2	0.2	0.1
RW0	9.9	5.3	0.9	1.0	0.2	0.2	0.1	0.1	0.1
RW15	9.9	5.3	0.9	1.0	0.2	0.2	0.2	0.1	0.1
AB	9.9	5.3	0.9	1.0	0.2	0.03	0.1	0.1	0.1
IM	2.7	7.3	0.1	0.4	1.1	0.04	0.06	0.3	0.1
LC	4.1	0.5	0.2	0.4	0.2	0.004	0.01	0.3	0.3
Louv.	9.1	12.5	4.8	10.0	9.0	2.1	1.2	20.4	16.4
Newm.	16.8	24.8	5.3	19.8	9.0	0.8	0.9	1.0	1.0
MCL	1.7	4.1	0.03	0.06	0.9	0.01	0.02	0.01	0.01
Metis	4.3	10.4	0.5	1.7	0.7	0.1	0.2	0.5	0.3
Annot.	10.6	6.2	1.4	1.3	3.2	0.1	0.2	0.2	0.1

5. FRAMEWORK AND APPLICATION

In this section, we describe our class separability framework and illustrate its applicability with an experimental application using the data that we processed through the steps described in the previous section.

Table III. Percentage of the Probability Mass of Classification of Elements in the Test Set into the Correct Class, Using SVM, for All Networks

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
BFS	60%	88%	73%	70%	(40%)	63%	55%	86%	81%
RW0	44%	55%	43%	(39%)	(27%)	52%	43%	61%	63%
RW15	40%	(29%)	44%	42%	34%	46%	39%	57%	57%
AB	83%	91%	90%	71%	60%	70%	74%	90%	89%
IM	27%	(23%)	72%	73%	(2%)	62%	51%	82%	66%
LC	68%	96%	83%	85%	83%	67%	56%	90%	89%
Louv.	24%	(3%)	49%	(1%)	(0%)	45%	58%	38%	49%
Newm.	(14%)	(25%)	(15%)	(0%)	90%	26%	39%	45%	56%
MCL	19%	(22%)	57%	28%	(34%)	59%	46%	80%	74%
Metis	61%	73%	81%	90%	(42%)	88%	66%	92%	86%
Annot.	37%	33%	50%	46%	(8%)	47%	40%	72%	71%
Global	1.44	2.11	1.7	1.81	1.35	1.58	1.38	1.68	1.76

The last row presents global separability ratio between the scatter matrices J3 scores measuring the separability of classes to a baseline consisting of the J3 scores of the same data with shuffled class labels. If a value is in parentheses, this indicates that a plurality of the probability mass from that class was assigned to some other class.

5.1. Class Separability Measures

Methods for measuring class separability are popular in machine learning for guiding feature selection analysis. Accordingly, effective feature sets for classification tasks are the ones that simultaneously lead to high interclass and low intraclass variability [Theodoridis and Koutroumbas 2008]. Methods of class separability allow for a rigorous analysis of independence among classes. Unfortunately, many of these methods are computationally demanding or depend on assumptions that are often mismatched with applications [Fatemi-Ghomi et al. 1999].

In this work, we frame the research question of discriminating the structure of different communities as a class separability problem. The separability of structural classes of communities provides information on whether different communities come from the same (or fundamentally different) distributions of feature values. This analysis is informative of the extent to which different algorithms produce structural differences and the extent to which community detection algorithms succeed in producing sets that resemble annotated communities.

To measure class separability, we first use the J3 metric [Theodoridis and Koutroumbas 2008], which is based on within-class and between-class scatter matrices. The J3 criterion calculates two scatter matrices. The between-class scatter matrix S_m is simply the global covariance matrix, and the within-class scatter matrix S_w is the average of the covariance matrices for each class, weighted by the size of that class. The J3 score is then the trace of $|S_w^{-1}S_m|$. A high J3 score indicates that within-class covariance is low and global covariance is high. The last row of Table III contains the ratio of the global J3 value for each network to a baseline J3 value that is obtained by measuring the separability of the same data, but with the class labels shuffled. A value of 1 indicates separability close to a random relabeling, and higher values indicate greater separability. These values demonstrate that the classes in each network are separable to some extent, even when evaluated through a fairly simple measure.

Although the J3 criterion is useful for gaining a coarse overview of class separability, it is a global measure, which tells us little about the separability behavior of each class. Aiming at achieving more fine-grained separability information while including all of the classes simultaneously and preserving computational scalability, we propose the use of the cross-validation performance of existing probabilistic multiclass supervised

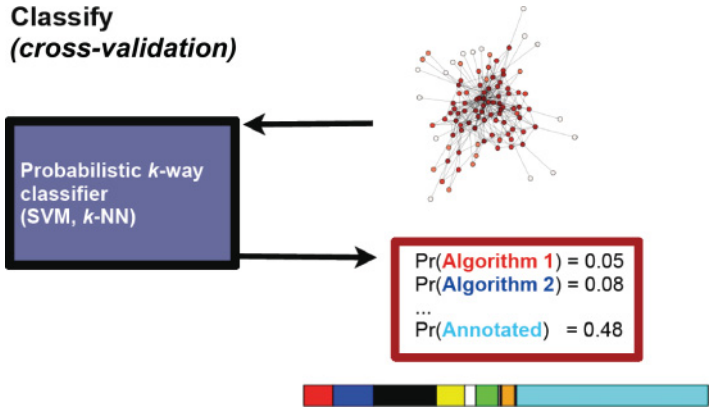

Fig. 3. Probabilistic multiclass supervised classification of an element in the test set in the cross-validation phase of the method to assess the separability of structural classes of communities.

classifiers as a measure of class separability. To our definition, classes are separable to the extent that a classifier can correctly distinguish their structure by exhibiting an accurate classification. More specifically, we employ two techniques, one parametric, namely SVMs [Vapnik 1998], and one nonparametric, namely kNN [Aha et al. 1991], to confirm each other's outcomes while ruling out variability due to the specifics of each algorithm. We select hyperparameters in both cases via grid search using the performance of 10-fold cross validation as the objective function.

The primary goal is to gain insight into whether the classes are separable in the feature space defined. This will tell us the extent to which the community detection algorithms produce structural profiles that are specific to each algorithm. Using a slight variation of the same approach, we can determine which algorithms produce outputs that most closely resemble the annotated communities.

Our approach to measure the separability of the structural classes of algorithms using probabilistic multiclass supervised classifiers is as follows. We measure class separability using the performance of a threefold cross validation. For each network, we train a multiclass classifier on a set containing two-thirds of the examples, which are selected at random, and then evaluate the performance of the model on the remaining third, which constitute the test set. We perform three rounds of this process and average the outcomes. For each element in a test set, the probabilistic SVM or kNN model outputs a probability mass vector indicating the probability that each data point belongs to each class. Figure 3 illustrates the cross-validation phase as applied to one element in the test set.

Using the structural classes computed via the steps described in the preceding section, we perform this cross validation on a dataset containing all 11 classes: 10 classes corresponding to the structure that the algorithms produce, and one class corresponding to the structure of annotated communities. We first observe that the experiments suffer little variability between the two classifiers. Figure 4 presents the analysis of the outcome produced by the SVM-based method applied to the DBLP network. In the picture, we show a bar graph of the distribution of probability mass for each class derived from the network DBLP. This graph visually demonstrates that the bulk of the probability mass from each class was correctly classified.

Table III contains a summary of results for all networks. Each entry in the table represents the fraction of probability mass from that class that was correctly assigned. When a value appears in parentheses, this indicates that more of the probability mass was assigned to some other class. As this table shows, only 17 out of 99 network-class

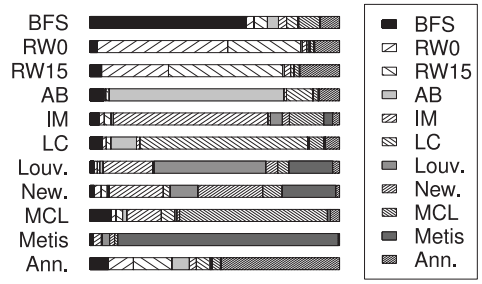

Fig. 4. Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the 11 classes.

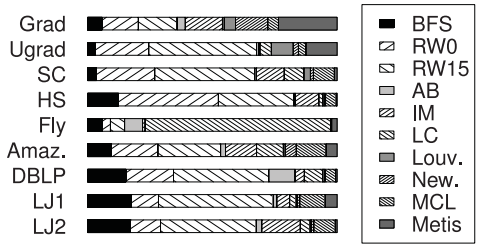

Fig. 5. Distribution of probability mass resulting from the SVM, classification of annotated communities.

pairs failed to have a plurality of the probability mass correctly classified. Newman Modularity is frequently misclassified; however, it is a small class in all networks, especially in the smaller ones (e.g., on network SC, Newman Modularity found only three communities of size between 10 and 1,000). In the case of annotated communities, a plurality of their corresponding classes tend to be correctly classified, with the exception of network Fly. Figure 4 serves as a visual reference of network DBLP, whose classes have a global separability score of 1.58. The other networks exhibit similar distributions, with the exception of network Fly, whose classes are less well separated.

The previous experiment shows that annotated communities tend to form their own, separable class that is significantly distinct from all other classes. However, a question of interest to the design and application of community detection procedures is which algorithms output communities bearing the closest structural resemblance to the annotated communities. To answer this question, we perform a slight variation of the classification task described previously. We train a classifier on the 10 classes corresponding to the community detection algorithms and leave the class of annotated communities out of the training set. The goal of this experiment is to evaluate to which class of intrinsically defined communities the annotated examples of the test set are classified.

Figure 5 shows the distribution of probability mass of the annotated communities classified into the different classes corresponding to community detection algorithms. The structure that the random walk and BFS procedures produce is clearly the most similar to that of the annotated communities. For seven of the nine networks, a plurality of the probability mass from the annotated communities was assigned to either RW15 or RW0, followed by BFS. Due to the high similarity between the two random walk classes, the classifier confuses these two as shown in Figure 4. The exceptions to this trend are networks Grad and Fly. For Grad, the annotated communities' probability is spread across many classes, with Metis receiving the plurality of the mass. In the Fly network, the greatest share of the mass of annotated communities is assigned to LC. These exceptions are associated with small network datasets; therefore, the variability could be due to small population sample size.

Given the diverse nature of these networks, it is perhaps surprising that in virtually all domains, the random walk and BFS communities bear the closest structural resemblance to the annotated communities. Even more astonishing is the fact that the structure of annotated communities is closer to that of the baseline procedures than to that of more structured approaches.

Table IV. Pairwise Separability for Classes in Network Amazon, Calculated Using Scatter Matrices

	BFS	RW0	RW15	AB	IM	LC	Louv.	Newm.	MCL	Metis	Annot.
BFS	1.00	1.14	1.14	1.16	1.22	1.19	1.40	1.22	1.15	1.57	1.13
RW0	1.14	1.00	1.04	1.33	1.40	1.33	1.52	1.31	1.28	2.01	1.14
RW15	1.14	1.04	1.00	1.27	1.33	1.36	1.54	1.35	1.28	1.94	1.12
AB	1.16	1.33	1.27	1.00	1.15	1.08	1.22	1.18	1.15	1.25	1.16
IM	1.22	1.40	1.33	1.15	1.00	1.36	1.39	1.15	1.13	1.14	1.12
LC	1.19	1.33	1.36	1.08	1.36	1.00	1.65	1.26	1.09	1.27	1.09
Louv.	1.40	1.52	1.54	1.22	1.39	1.65	1.00	1.06	1.63	1.19	1.26
Newm.	1.22	1.31	1.35	1.18	1.15	1.26	1.06	1.00	1.24	1.13	1.12
MCL	1.15	1.28	1.28	1.15	1.13	1.09	1.63	1.24	1.00	1.22	1.06
Metis	1.57	2.01	1.94	1.25	1.14	1.27	1.19	1.13	1.22	1.00	1.27
Annot.	1.13	1.14	1.12	1.16	1.12	1.09	1.26	1.12	1.06	1.27	1.00

Ratios are measured relative to the baseline value obtained by shuffling class labels.

5.2. Class Selection Method

The feature space generated by the data forms a reference system for analyzing community structure from the viewpoint of the diversity of structures that come from different notions of communities. When we employ class separability measures, it is particularly important to ensure that the classes considered are independent and not redundant. For example, when identifying which type of community detection method produces structure that most resembles that of the annotated communities, we saw that one of the two random walk classes tended to be assigned the bulk of the probability mass of the annotated communities on most networks. However, these two classes are remarkably similar. When we present a classifier with two almost indistinguishable classes, the classifier splits the examples' probability masses equally between the classes.

To provide a rigorous method of class selection, we propose the application of a pairwise Scatter Matrix measurement [Theodoridis and Koutroumbas 2008] for analyzing class similarity and determine which classes from a collection are redundant or independent. We again use the J3 criterion that we originally used to measure overall class separability within a network, but this time we measure pairwise separability by considering each pair of classes separately. These values give us insight into whether certain pairs of classes are redundant, are correlated, or overlap significantly (e.g., the two methods for maximizing modularity). Values for network Amazon are shown in Table IV. In this table, values are presented as a ratio of the J3 score of a pair of classes to the J3 score of a baseline separability value, which is obtained by shuffling the examples' class labels. As we see in Table IV, no two classes are completely identical, although the two random walk classes are quite similar, as are the two modularity-based classes. We see this pattern repeated consistently in other networks, although occasionally other pairs of classes are similar (e.g., in network Amazon, the MCL and Annotated classes have a fairly low separability score, but this pattern does not occur in other networks).

We now repeat our earlier experiments after applying the results obtained by our class selection method. We learned from Table IV that some pairs of classes, namely the two random walk and the two modularity classes, are somewhat redundant. We thus merge each of these pairs of classes into one single class, and again perform the earlier experiments.[3]

[3]Here we could have preserved one representative of each group of similar structural classes instead of merging the similar classes. Whether to use one approach or the other depends on the intended application.

Table V. Percentage of the Probability Mass of Classification of Elements in the Test Set into the Correct Class, Using SVM, for All Networks, After Merging Classes

	Grad	Ugrad	HS	SC	Fly	DBLP	Amazon	LJ1	LJ2
BFS	70%	90%	74%	69%	(39%)	65%	60%	85%	81%
RW	74%	91%	83%	83%	61%	68%	65%	91%	88%
AB	79%	93%	92%	77%	54%	70%	76%	91%	89%
IM	4%	(15%)	76%	72%	(2%)	63%	53%	83%	67%
LC	62%	95%	87%	81%	84%	68%	58%	91%	88%
Modul.	35%	(25%)	39%	33%	(34%)	44%	52%	53%	62%
MCL	25%	(21%)	59%	33%	(25%)	61%	47%	81%	74%
Metis	67%	74%	73%	83%	(41%)	89%	70%	93%	85%
Annot.	27%	40%	52%	47%	(7%)	53%	45%	74%	74%

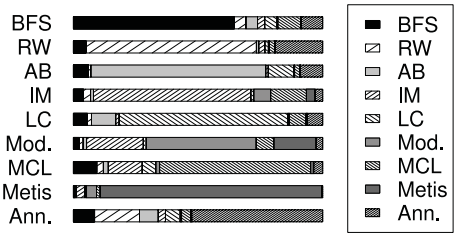

Fig. 6. Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the nine classes after merging redundant classes.

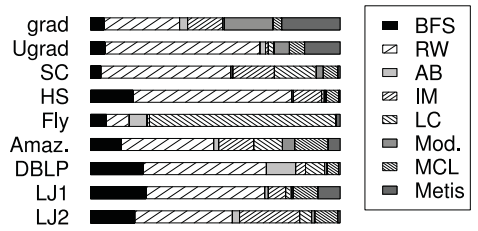

Fig. 7. Distribution of probability mass resulting from the SVM, classification of annotated communities on the nine classes after merging redundant classes.

Figure 6 contains the distribution of probability mass obtained by applying the SVM to all nine classes on network DBLP, and Table V contains the percentage of probability mass from each class in each network that was correctly classified. We saw in our first experiment that the classifiers tended to confuse the two random walk classes with one another; for example, elements from RW15 were frequently misclassified as belonging to class RW0. As expected, we typically see that the single random walk class and single modularity class are substantially more consistent than either of their subclasses alone. This effect is particularly pronounced on certain networks, such as Ugrad, where more than 90% of the probability mass from the large random walk class is correctly classified (as opposed to values of 55.1% and 61.4% for RW0 and RW15 separately).

Figure 7 contains the distribution of probability mass obtained when classifying the annotated communities into one of the eight algorithm classes. As predicted by our class selection method, the merged classes receive all of the mass assigned to the corresponding separate classes. Moreover, previously, for network Grad, we saw that a plurality of the mass of the annotated communities was previously assigned to the Metis class; however, now that we combine the two random walk classes, we see that the single random walk class receives more probability mass than either of the two random walk classes alone, and thus more than the Metis class.

5.3. Class Consistency Across Networks

We now turn our attention to the problem of analyzing class consistency across networks. In our previous experiments, we analyzed each network separately and saw that the 11 classes of communities all tended to be fairly consistent (e.g., elements from class BFS in network Grad shared important structural features with one another). In the

Table VI. The Percentage of Probability Mass from Each Class That Was Correctly Classified

Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	60.3%	34.7%	42.6%	36.4%	25.2%	46.2%	63.4%	37.1%	48.5%
RW0	44.5%	26.2%	30.6%	21.0%	20.8%	38.8%	30.8%	22.8%	20.9%
RW0.15	41.0%	23.6%	24.8%	16.0%	20.1%	48.4%	26.4%	19.8%	24.9%
AB	58.4%	41.7%	63.6%	44.5%	9.9%	20.0%	55.1%	64.3%	62.8%
InfoMap	4.7%	5.5%	22.0%	42.5%	23.8%	6.0%	25.1%	34.0%	33.0%
LinkCom	6.6%	14.6%	29.4%	33.8%	15.6%	14.8%	27.7%	23.2%	28.4%
Louvain	2.1%	43.5%	8.0%	25.0%	41.0%	1.1%	7.8%	20.9%	9.1%
Newman	6.3%	4.9%	8.4%	23.2%	3.6%	9.9%	5.4%	34.2%	53.6%
MCL	7.3%	13.9%	23.7%	20.3%	27.1%	13.6%	18.9%	40.4%	7.3%
Metis	20.2%	11.9%	31.0%	12.1%	34.8%	3.5%	5.9%	14.8%	14.0%
Ann.	5.8%	19.7%	23.8%	10.8%	20.6%	31.0%	17.3%	36.1%	34.2%

The column titles indicate the networks on which the classifiers were trained. Row labels indicate the different algorithm classes. The value in Row C, Column N, indicates the average percentage of probability mass from class C that was correctly classified when the classifier was trained on network N and evaluated on all networks except for N.

next experiment, rather than studying each network in isolation, we simultaneously examine classes across all networks.

We perform nine experiments, each corresponding to a network N. In each experiment, we create a training set containing elements from all 11 community classes in network N, labeled with their community type (e.g., BFS, Louvain, etc.). We create a test set containing elements from the 11 community classes of the other eight networks. In this test set, each element is labeled with its community type, but we do not identify the network from which it came.

The purpose of this experiment is to determine whether communities generated by a specific method tend to share structural features, even across networks. For example, we wish to learn whether an SVM trained only on representatives from network Grad can correctly classify communities from other networks: does a BFS community from network DBLP resemble a BFS community from network Grad?

Table VI contains the results of this experiment. The element in row C, column N contains the average percentage of probability mass from class C that was correctly classified when the classifier was trained on elements from network N. For example, when the SVM was trained on elements from network Grad, an average of 60.3% of the probability mass from class BFS in the other networks was correctly classified as BFS.

We see that some classes, particularly BFS, RW0, RW0.15, and AB, tend to have a great deal of consistency across networks. The other classes tend to have a less distinct signature, often performing worse than random.

6. STRUCTURAL TENDENCIES OF COMMUNITIES

As we have seen in the preceding experiment, each community detection algorithm extracts a distinct structure, which our method is able to separate when projected onto the feature space that we define. In this section, we are concerned with identifying the ways in which the algorithms produce these distinguishable biases by finding which properties exhibit the highest degree of between-class variability. We are also concerned with identifying the properties that are the most discriminative to distinguish between annotated communities and the synthetic data produced by the algorithms. Finally, a dividend of our approach is the ability to organize a collection of algorithms by grouping those that exhibit similar behaviors.

To address this question, we use the Correlation-based Feature Selection (CFS) algorithm [Hall 1999] to identify subsets of the most discriminative features for each

Table VII. Summary of the Feature Selection Results
Features are ranked in order of their frequency in the selection list over the networks.
(*Reporting how many quartiles of the property were selected. **Feature number according to Table I.)

Network		Grad	Ugrad	HS	SC	Fly	DBLP	Amaz.	LJ1	LJ2
Number of Features Selected		6	7	10	5	6	10	8	12	11
Rank	Feature									
1	Conductance	1	1	1	1		1	1	1	1
1	Diameter		1	1	1	1	1	1	1	1
3	Info Centrality*	2	2	3	1	1	2	1	2	2
4	Node Betweenness*			2	2		2	1	5	5
5	Shortest Path*			1		3	2	1	1	1
6	β^*	1	1	1			2	1	1	1
7	α^*	1		1		1		2	1	
Other Features**		#6	#4, #7							

network. CFS is intended to identify a set of features that are well correlated with the class label and poorly correlated with each other. For a given subset S containing k features, CFS defines a merit function M_S:

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}. \quad (1)$$

In this equation, $\overline{r_{cf}}$ represents the mean correlation between each individual feature f in S with the class label c, and $\overline{r_{ff}}$ represents the mean correlation between pairs of features in K. Intuitively, this function gives a high score to sets of features that are highly predictive of the class and are not redundant with one another. Using this function, one can rank all subsets of features, but this would be inefficient in most cases. CFS begins with no elements in the feature set, and it then employs a hill-climbing algorithm to search the space of feature subsets. The algorithm includes the ability to backtrack up to five times per iteration to search for a subset S with a greater value of M_S.

Table VII lists the features selected by CFS for each network ranked in order of the frequency with which they appear in the selection over the networks. The table lists the most frequent features or, for those properties calculated with quartiles, sets of features. The entries for row "Features" and column "Network" that contain the value 1 indicate the presence of that feature in the feature selection process applied to the data from that particular network, whereas empty cells indicate the absence thereof. Integers larger than 1 can be found in some of the entries and indicate the number of quartiles from that feature that were selected by CFS. In nearly every network, conductance, diameter, information centrality, and node betweenness were the most discriminative features.

Surprisingly, in several cases, multiple quartiles of a feature appear: for example, Fly has three path length quartiles, and LJ1 and LJ2 each contain all five node betweenness features. We had expected that different quartiles of the same feature would be highly correlated to each other, and therefore they would be unlikely to co-occur among the features selected by CFS. Instead, these results suggest that varying the choice of community detection algorithm results in fine-grained variation in the distribution of such features.

To assess the effectiveness of the features that CFS found, Table VIII presents for all networks the classification performance of the kNN cross validation using both the full set of features and the subset of features found by CFS. We see that in most cases, there is very little loss in accuracy. We observe a similar qualitative outcome for the SVM cross validation. In the table, the largest drops happened for LJ1 and LJ2 and

Table VIII. *k*-NN Classification Performance Using Both the Full Set of Features and the Subset of the Most Discriminative Features Selected by CFS

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz.	LJ1	LJ2
All Features	62.9%	86%	82.2%	80.9%	93.6%	81.3%	65.3%	89.1%	88.5%
With selection	61.5%	84.7%	85.1%	81%	90.6%	79.4%	63.0%	78.8%	76%

Table IX. Summary of the Feature Selection Results When Classifying Implicitly and Explicitly Defined Communities
Features are ranked in order of their frequency in the selection list over the networks.(*Reporting how many quartiles of the property were selected. **Feature number according to Table I.)

Network		Grad	Ugrad	HS	SC	Fly	DBLP	Amaz.	LJ1	LJ2
Number of Features Selected		6	5	3	6	5	7	6	4	5
Rank	Feature									
1	Node Betweenness*	2	3	1	2	2		2	2	3
2	Conductance			1	1	1	1	1	1	1
3	Info Centrality*	1	2	1	3	2				
4	α*						5	1	1	
5	Edge Betweenness*	2						1		
Other features**		#7					#9	#4		#3

reduced the accuracy by less than 15%. Being nearly as discriminative as the full set, a reduced set containing a handful of features retains the relevant information needed to analyze the bias produced by different algorithms.

To identify the properties that are the most discriminative to distinguish between annotated communities and the synthetic data produced by the algorithms, we next perform an experiment similar to the preceding one. However, instead of considering separate classes of communities that the algorithms produce, we merge all implicitly defined classes into a single class. We consider the set of annotated communities to be one class and the set of communities extracted by all algorithms to be another single class. In this experiment, we wish to identify those features that are most useful for distinguishing between explicitly and implicitly defined communities. As before, we use the CFS method to identify useful features. Table IX contains the results of this experiment. Again, we see that conductance, node betweenness, and information centrality are particularly important.

Finally, to organize a collection of algorithms by grouping those that exhibit similar behaviors, we use the sets of the most discriminative features found in Table VII to study which tendencies in feature values are associated with which algorithms. To this end, we conducted a range analysis that distinguishes the different algorithms according to the value of their features. In the interest of space, we summarize the qualitative outcome of this experiment in Figure 8. The entries correspond to the bias produced by each of the algorithms, considering all networks. Features take on a varying range of values across different networks. Thus, to label the magnitude of features, we compute the mean value of each class and compute a global median of these averages over all classes. The averages occurring between the 33rd and 67th percentile constitute the *medium* denomination, whereas those below the 33rd and above the 67th constitute *low* and *high*, respectively. Finally, we count how many times each feature produced each of the denominations across all of the networks. From this count, we calculate three times the number of networks on which the feature had a high score on that class plus two times the number of networks on which the feature had a medium score on that class plus the number of networks on which the feature had a low score on that class, and we present these values in Figure 8.

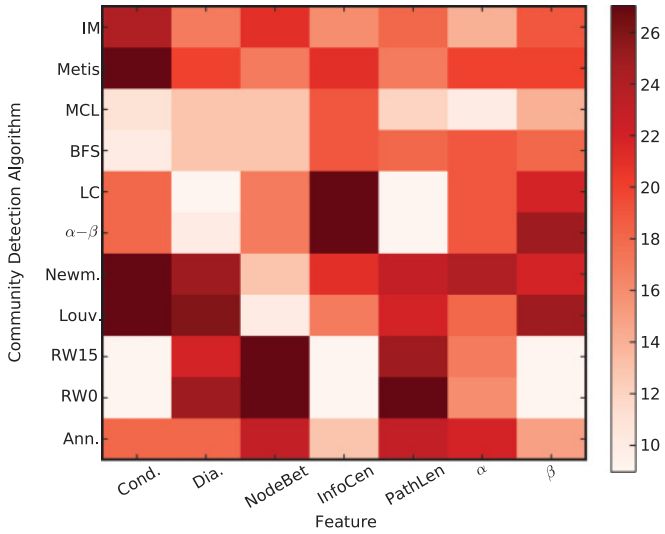

Fig. 8. Tendency of algorithms with respect to various features. Scores are calculated by three times the number of networks on which the feature had a high score on that class plus two times the number of networks on which the feature had a medium score on that class plus the number of networks on which the feature had a low score on that class.

Using this analysis, we are able to group algorithms with similar behavior. For example, the random walk procedures produce the same structural bias. The same holds for Louvain and Newman, and AB and LC. Our method identified these similarities without any prior knowledge about the similar goals shared by these algorithms. Metis and IM differ only in behavior of the node betweenness feature. The profile of annotated communities is close to that of random walk procedures, with a few nuances. Annotated communities exhibit medium conductance, whereas RW0 and RW15 extract low conductance sets. In addition, the diameter of annotated communities was measured as high for four of the networks, medium for one of them, and low for the remaining four. This contrasts with RW0 and RW15, which produce sets with high diameter. Nevertheless, the similarity due to other features explains the ways in which annotated communities resemble the output of random walk–based algorithms.

7. NETWORK CONSISTENCY

Our previous experiments have considered various problems of distinguishing between communities generated through different methods. We saw that many classes of communities tended to have a strong *signature*, both within and even across networks; for example, communities generated by a BFS algorithm were fundamentally different from communities generated by the Louvain method. In the next three experiments, instead of analyzing whether communities generated by different methods have different structures, we ask whether communities from different networks are distinguishable from one another.

In the first of these experiments, we consider each of the 11 methods of defining communities separately (we perform a different experiment for each of BFS, Louvain, etc.). In each experiment, we train an SVM classifier on a set containing communities from one specific class from all nine networks, where each community is labeled with the network from which it came. We then evaluate this classifier on other communities from the same class, again from all nine networks, also labeled with their networks of origin. For example, in our first experiment, both the training and test sets contain

Table X. The Percentage of Probability Mass from Each Network That Was Correctly Classified as Belonging to That Network
The row titles indicate the class on which the classifier was trained.

Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	68.9%	67.4%	36.6%	25.0%	61.3%	78.6%	71.0%	47.3%	43.3%
RW0	67.5%	74.2%	45.1%	49.5%	68.2%	82.8%	74.1%	55.4%	51.0%
RW0.15	69.3%	76.2%	47.6%	50.1%	77.2%	83.7%	76.0%	55.5%	49.2%
AB	69.8%	55.1%	30.5%	31.2%	72.0%	82.1%	70.9%	43.4%	40.4%
InfoMap	33.2%	87.8%	47.6%	51.9%	54.7%	82.1%	78.3%	62.3%	57.1%
LinkCom	38.8%	94.2%	72.9%	66.7%	74.6%	81.1%	78.1%	54.1%	53.5%
Louvain	50.3%	71.9%	72.4%	46.4%	2.6%	90.1%	88.3%	39.9%	48.5%
Newman	0.2%	0.1%	0.3%	0.1%	0.1%	94.8%	21.5%	15.2%	60.8%
MCL	20.0%	48.1%	49.9%	22.3%	22.4%	80.6%	72.5%	53.0%	55.8%
Metis	92.3%	98.1%	88.5%	83.9%	81.5%	98.2%	98.0%	92.0%	92.1%
Ann.	60.4%	52.8%	22.4%	27.3%	38.3%	91.8%	85.7%	44.5%	45.8%

elements from Grad BFS, Ugrad BFS, DBLP BFS, and so on, labeled as Grad, Ugrad, or DBLP, respectively.

Table X contains the results of this experiment. The element in row C, column N contains the percentage of the probability mass from network N that was correctly classified when the SVM was trained on representatives of class C from all networks. We see that all of the networks have a very distinctive signature. (As before, the Newman classes are often very small, so training on this class produces unreliable results.)

We saw earlier that elements from the same class and different networks had structural similarities (for certain classes). For example, a Grad BFS community had some resemblance to a DBLP BFS community. In this experiment, we see that elements from the same class and different networks generally also have important differences— that is, although a Grad BFS community resembles a DBLP BFS community in important ways, it is still possible to distinguish between the two. However, there are some notable exceptions to this statement. For most classes, the classifier does a very good job at distinguishing between networks; however, the performance of the classifier trained on elements from the Newman (and, for network Fly, Louvain) class is particularly weak. We see also that some networks tend to have a stronger signature than others. For instance, the SVMs consistently classify elements from Amazon and DBLP correctly; in contrast, they are less accurate for LJ1 and LJ2 (a closer examination of the data shows that, unsurprisingly, elements from LJ1 and LJ2 tend to be confused with one another).

Our next experiment is structured similarly, except instead of considering each method of community definition separately, we merge all 11 community detection methods together into a single experiment. In this experiment, the training and test sets again contains elements from all nine networks; however, each network is represented by elements from all 11 community identification methods. For example, the training and test sets contain elements from both Grad BFS and Grad Louvain, all of which are labeled simply as Grad, and elements from both Ugrad BFS and Ugrad Louvain, all of which are labeled as Ugrad.

Table XI contains the results of this experiment, with values along the diagonal representing the percentage of probability mass that was correctly classified. We see again from this experiment that all of these networks have a distinct signature, although some are stronger than others. As before, LJ1 and LJ2 are often confused with one another. Ugrad and Amazon, in particular, have communities that are highly distinguishable from those of other networks. Again, we see that most networks have

Table XI. The Percentage of Probability Mass from Elements in Each Network That Was Classified as Each Network
Row *N*1, column *N*2 contains the fraction of probability mass of elements from network *N*1 in the test set that was classified as network *N*2.

	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
Grad	62.5%	0.5%	6.0%	1.8%	0.6%	6.9%	8.3%	4.5%	8.8%
Ugrad	3.7%	74.0%	2.0%	9.0%	1.4%	0.4%	0.9%	5.0%	7.0%
HS	2.3%	1.6%	44.2%	7.1%	2.0%	13.0%	5.9%	11.4%	12.6%
SC	1.6%	4.9%	6.3%	42.8%	4.3%	2.1%	4.7%	17.7%	15.5%
Fly	0.5%	2.2%	4.1%	6.5%	60.7%	1.2%	3.1%	11.7%	9.9%
Amazon	1.8%	0.3%	4.6%	0.8%	0.4%	71.6%	11.3%	3.4%	5.3%
DBLP	2.4%	0.5%	3.0%	2.1%	0.9%	12.5%	64.1%	6.1%	48.5%
LJ1	2.0%	1.9%	6.4%	7.7%	4.3%	4.8%	6.8%	35.8%	30.3%
LJ2	2.8%	2.2%	6.1%	7.4%	3.9%	5.1%	8.1%	27.3%	37.3%

Table XII. The Percentage of Probability Mass from Each Network That Was Correctly Classified as Belonging to That Network, Where the Training Set Contained Elements from the Specified Class

Training Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	43.5%	11.5%	11.0%	9.7%	79.2%	73.9%	45.4%	31.2%	5.2%
RW0	40.0%	13.1%	11.0%	11.5%	68.5%	76.0%	44.1%	22.7%	7.3%
RW0.15	35.9%	14.7%	10.9%	15.2%	48.8%	76.6%	47.7%	19.5%	9.2%
AB	30.5%	12.0%	8.7%	12.6%	44.3%	73.4%	46.8%	18.2%	11.2%
InfoMap	25.4%	12.7%	9.0%	19.2%	49.6%	66.0%	44.8%	17.5%	14.2%
LinkCom	21.8%	11.7%	14.4%	21.8%	49.1%	60.4%	44.2%	15.1%	12.2%
Louvain	19.2%	10.6%	13.4%	19.8%	43.2%	52.8%	43.9%	15.8%	11.3%
Newman	17.3%	10.3%	12.0%	17.1%	37.1%	50.3%	40.3%	18.1%	16.8%
MCL	16.4%	11.1%	14.2%	18.2%	32.7%	49.0%	39.9%	19.8%	16.4%
Metis	16.8%	11.4%	14.6%	16.7%	39.2%	45.9%	37.0%	20.3%	16.8%
Ann.	28.6%	20.5%	10.9%	12.8%	60.4%	60.3%	41.7%	22.3%	16.8%

a strong signature (and even LJ1 and LJ2 can be somewhat distinguished from one another).

In our final experiment, we again perform a separate experiment for each of the 11 methods of defining communities. For each method M (e.g., BFS), we train the classifier on a set containing representatives of method M from all nine networks, where each element is labeled with its network of origin. The test set contains representatives of every method *except* M, again from all nine networks, also labeled by the network. In this experiment, we determine whether a classifier can identify which network a community came from, even when that community was defined through a different method than the examples on which the classifier was trained.

Table XII contains the results of this experiment. Naturally, the classifier's performance in this experiment is much worse than in the preceding two experiments, because the elements in the training set are fundamentally different from the elements in the test set. Even so, we see again that some networks have very strong signatures and are easily differentiated from the others, whereas for other networks, the accuracy is approximately what one would expect if the classifier made decisions at random. In particular, communities in Amazon tend to have structural similarities, regardless of how that community was defined; that is, a classifier trained to identify BFS sets in Amazon can also identify other types of communities in Amazon. We see similar behavior in networks Fly and DBLP. Interestingly, we also see that the algorithm class used in the training set is important as well. For example, SVMs trained on BFS communities tend to do much better at classifying other communities than do

Table XIII. Runtime Results for Networks DBLP, Amazon, LJ1, and LJ2
Each row represents a different class sample size. Cells contain average classification accuracy across all classes and running time (in seconds) for that sample size.

	DBLP		Amazon		LJ1		LJ2	
	Accuracy	Runtime	Accuracy	Runtime	Accuracy	Runtime	Accuracy	Runtime
100	42%	174	38%	187	56%	146	53%	149
250	44%	912	40%	1,069	58%	751	56%	773
500	47%	3,368	42%	4,051	59%	2,724	58%	2,736
750	48%	7,725	43%	9,253	61%	5,841	59%	6,059
1,000	49%	13,718	44%	16,678	61%	10,219	60%	10,723

SVMs trained on Metis. This suggests that BFS communities are, in some sense, more representative of the entire set of communities from a particular network.

8. DISCUSSION

This article presents a methodology to address the complexity of analyzing community structure in light of the different notions of communities. This approach contrasts with traditional approaches where the characterization of objects requires the exhaustive enumeration of negative examples. Here we use the diversity of community structures exhibited by different processes as references to understand other community structures. Our approach simultaneously considers a large number of algorithms, multiple domains of application, and a broad spectrum of metrics to characterize community structure.

The machine learning methods that we used can easily handle large numbers of features and community detection methods, but the performance of the framework is limited by the performance of these methods.

Almost every step of our method is highly parallelizable and can take advantage of simple task distributions over large clusters and multicores. Once we obtain output from various community detection methods (where each algorithm can run on an independent processor), we can easily calculate the community properties by spreading the computation thereof among an arbitrary number of computers. Calculations of the features that we considered in this article were fairly fast. In addition, most of the communities that arise in practice are relatively small graphs. In the classification portion of our framework, classifiers may have difficulty with large datasets, but one can simply sample feature vectors from each class; indeed, this may even be necessary to ensure class balance.

Table XIII contains runtime and accuracy values for each of our four larger networks at different sample sizes. In this experiment, which is similar to the experiment described in Section 5 and presented in Table III, we trained an SVM on balanced classes of various different sizes and evaluated the accuracy of the resulting classifier on withheld elements on those classes. For each sample size, we present both the average accuracy of the classifier over all classes as well as the time required to create and evaluate the model.

We see that smaller sample sizes result in remarkably faster model creation and evaluation, with only a slight drop in accuracy. A practitioner highly concerned with efficiency can thus obtain a fairly accurate model very quickly. These results illustrate the scalability of our framework and so demonstrate its feasibility on large datasets.

It is important to emphasize that our work focuses on structural similarity, which is a weaker requirement than accuracy. In other words, communities with similar properties to real communities may not correspond exactly to the communities that we may expect to find. Nevertheless, we firmly believe that mastering structure is a

fundamental stepping stone in the development of algorithms to accurately find the communities of interest.

Our experimental analysis includes 10 community detection algorithms, representative of popular algorithms in the literature, a collection of nine different networks from diverse domains, and 36 structural properties. The results reveal, first, a high variability among the output of community detection methods, which is demonstrated by cross-validation performance of the classifiers, showing that each of these 10 classes of communities was remarkably structurally consistent. Second, annotated communities have a distinct structure from what we expect; a classifier can distinguish most of the mass of annotated communities from that of communities produced by algorithms. In fact, their structure is closer to the output of baseline procedures, such as random walks and breadth-first search, than to that of more structured popular algorithms. Third, a small set of features explain the biases produced by different algorithms and expose the structural signature of real communities, which were represented in this study via meaningful annotations present in our datasets. Fourth, we can organize the menagerie of available community detection algorithms by grouping them with respect to structural similarities in behavior. This can be done via two mechanisms: (1) by analyzing the behavior of algorithms with respect to the most relevant features found in the preceding step, and (2) by applying a class selection method we defined based on the scatter matrices to determine whether two classes are independent (i.e., each class bringing a new viewpoint to our analysis) or redundant (i.e., where multiple classes contribute the same information to the collection). The latter approach is also applied in our frameworks as a preprocessing step to improve the interpretability of the separability measures. We saw that the two random walk classes and the two modularity-based classes were quite similar and thus repeated our earlier experiments after merging these pairs of classes. This naturally resulted in greater class consistency and further cemented the dominance of the random walk methods when classifying the annotated communities. Last, in contrast to our earlier experiments, which analyzed the consistency of classes defined by community detection methods, we analyzed whether communities from the same network tended to resemble each other. We first restricted our analysis to one type of community detection method at a time and showed that communities produced by the same method but from different networks tended to be different in important ways. Interestingly, we showed that even when a classifier was trained to differentiate between networks by using communities produced by one particular community detection method, that same classifier was successful even when applied to a test set containing communities produced by the other community detection methods.

Our approach differs fundamentally from previous work in the area due to its supervised nature. The main message that resulted from our experimental analysis is that community structure is not clearly defined and, moreover, not universal. In fact, there is a broad range of structures that are generated by multiple definitions, heuristics, and expectations. Accordingly, our supervised approach may be used by a practitioner to make an informed decision about the most suitable algorithm for a given network in the following way. First, we produce a test set comprising examples of the communities that we are interested in finding, which could be either real or synthetic. Second, we choose a set of algorithms that we want to evaluate. Finally, we apply our approach using the target network and present the classifier with the test set. The classifier assigns the probability mass of the test set to the class of algorithm that bears closest resemblance to the examples. The algorithm that receives the bulk of the mass is the algorithm that is most likely to succeed in extracting communities that structurally resemble the ones in which we are interested. Researchers may also benefit from our methodology when designing new community detection algorithms as a way to

compare the behavior of new methods with existing ones. Finally, our framework suggests a change in the way that we approach the problem of community detection. Typical community detection methods treat the task of extracting communities as an unsupervised problem in which a particular structure is extracted. However, this approach presents little sensitivity to different purposes, different structures of interest, and different domains of application. In contrast, we could start thinking about a supervised approach that allows the user to specify the particular community structure that they intend to find through examples. Then, a hypothetical algorithm would learn structure from these examples and retrieve similar structures from the network.

REFERENCES

Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. 2012. On the separability of structural classes of communities. In *Proc. of the 18th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*.

David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6, 1, 37–66.

Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature* 466, 7307, 761–764.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: Membership, growth, and evolution. In *Proc. of the 12th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*.

J. P. Bagrow and E. M. Bollt. 2005. A local method for detecting communities. *Physical Review E* 72, 046108.

Brian Ball, Brian Karrer, and M. E. J. Newman. 2011. Efficient and principled method for detecting communities in networks. *Physical Review E* 84, 3, 036103.

Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismail. 2005. Efficient identification of overlapping communities. In *Proc. of the 2005 IEEE Intl. Conf. on Intelligence and Security Informatics*. 27–36.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 10, P10008+.

Nitesh V. Chawla. 2005. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*. Springer, 853–867.

Fan R. K. Chung. 1996. *Spectral Graph Theory*. American Mathematical Society.

Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6, 066111+.

Michele Coscia, Fosca Giannotti, and Dino Pedreschi. 2011. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining* 4, 5, 512–546.

Stijn Van Dongen. 2008. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications* 30, 1, 121–141.

T. S. Evans and R. Lambiotte. 2009. Line graphs, link partitions and overlapping communities. *Physical Review E* 80, 016105.

N. Fatemi-Ghomi, P. L. Palmer, and M. Petrou. 1999. The two-point correlation function: A measure of interclass separability. *Journal of Mathematical Imaging and Vision* 10, 1, 7–25.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 75–174.

M. Girvan and M. Newman. 2002a. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12, 7821–7826.

M. Girvan and M. E. J. Newman. 2002b. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12, 7821–7826.

Steve Gregory. 2008. A fast algorithm to find overlapping communities in networks. In *Proc. of the 2008 European Conf. on Machine Learning and Knowledge Discovery in Databases: Part I*. 408–423.

Mark A. Hall. 1999. *Correlation-Based Feature Subset Selection for Machine Learning*. Ph.D. Dissertation. Department of Computer Science, University of Waikato.

Jake M. Hofman and Chris H. Wiggins. 2008. Bayesian approach to network modularity. *Physical Review Letters* 100, 25, 258701+.

S. Hoory, N. Linial, and A. Wigderson. 2006. Expander graphs and their applications. *Bulletin of the American Mathematical Society* 43, 4, 439.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20, 1, 359–392.

B. W. Kernighan and S. Lin. 1970. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal* 49, 1, 291–307.

Christian Komusiewicz, Falk Huffner, Hannes Moser, and Rolf Niedermeier. 2009. Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science* 410, 38a-40, 3640–3654.

Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: A comparative analysis. *Physical Review E* 80, 056117.

Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3, 033015.

Sune Lehmann, Martin Schwartz, and Lars K. Hansen. 2008. Biclique communities. *Physical Review E* 78, 1, 016108+.

Jure Leskovec, Lada Adamic, and Bernardo Huberman. 2006. The dynamics of viral marketing. In *Proc. of the 7th ACM Conf. on Electronic Commerce*.

Jure Leskovec, Kevin Lang, Anirban Dasgupta, and Michael Mahoney. 2008. Statistical properties of community structure in large social and information networks. In *Proc. of the 17th Intl. Conf. on World Wide Web*.

Jure Leskovec, Kevin Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proc. of the 19th Intl. Conf. on World Wide Web*.

Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. 2009. MetaFac: Community discovery via relational hypergraph factorization. In *Proc. of the 15th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 527–536.

Russell Lyons and Yuval Peres. 2012. *Probability on Trees and Networks*. Cambridge University Press.

Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert Tarjan. 2008. Finding strongly knit clusters in social networks. *Internet Mathematics* 5, 1, 155–174.

Alan Mislove, Bimal Viswanath, Krishna Gummadi, and Peter Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *Proc. of the 3rd ACM Intl. Conf. on Web Search and Data Mining*.

M. E. J. Newman. 2004. Detecting community structure in networks. *European Physical Journal B* 38, 2, 321–330.

M. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103, 23, 8577–8582.

Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043, 814–818.

Daniel Park, Rohit Singh, Michael Baym, Chung-Shou Liao, and Bonnie Berger. 2011. IsoBase: A database of functionally related proteins across PPI networks. *Nucleic Acids Research* 39, suppl 1, D295–D300.

Pascal Pons and Matthieu Latapy. 2006. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications* 10, 2, 191–218.

Martin Rosvall and Carl Bergstrom. 2011. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE* 6, 4, e18209.

Satu Elisa Schaeffer. 2005. Stochastic local clustering for massive graphs. In *Proc. of the 9th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining*. 354–360.

Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. 2009. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and Its Applications* 388, 1706–1712.

Karen Stephenson and Marvin Zelen. 1989. Rethinking centrality: Methods and examples. *Social Networks* 11, 1, 1–37.

Sergios Theodoridis and Konstantinos Koutroumbas. 2008. *Pattern Recognition* (4th ed.). Academic Press.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Fang Wei, Weining Qian, Chen Wang, and Aoying Zhou. 2009. Detecting overlapping community structures in networks. *World Wide Web* 12, 2, 235–261.

E. Weinan, Tiejun Li, and Eric Vanden-Eijnden. 2008. Optimal partition and effective dynamics of complex networks. *Proceedings of the National Academy of Sciences* 105, 23, 7907–7912.

Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *12th IEEE Intl. Conf. on Data Mining*.

Received October 2012; revised May 2013; accepted August 2013