Pattern Recognition Letters 31 (2010) 413–421

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Identifying and evaluating community structure in complex networks

Karsten Steinhaeuser, Nitesh V. Chawla *

University of Notre Dame, Department of Computer Science and Engineering, Interdisciplinary Center for Network Science and Applications (iCeNSA), Notre Dame, IN 46556, USA

ARTICLE INFO

Article history:
Received 3 March 2009
Received in revised form 4 September 2009
Available online 6 November 2009

Communicated by F. Roli

Keywords:
Complex networks
Community detection
Social networks
Random walks
Scalability
Evaluation metrics

ABSTRACT

We compare and evaluate different metrics for community structure in networks. In this context we also discuss a simple approach to community detection, and show that it performs as well as other methods, but at lower computational complexity.

1. Introduction

Modern data mining is often confronted with problems arising from complex relationships in data. Recently, datasets that can be represented as graphs, or *interaction networks*, have received considerable attention in various domains. Application areas include the analysis of social networks (Girvan and Newman, 2002; Wasserman and Faust, 1994), chemical interactions between proteins (Asur et al., 2007; Enright et al., 2002), transactions of goods and services (Clauset et al., 2004), and many others. With the increasing availability of rich network data, there is also a need for effective and efficient analysis methods.

One problem of great interest for pattern recognition in complex networks is *community detection*, or the unsupervised discovery of densely connected subgroups which are known to exist in many real-world networks. On the surface the concept of communities appears intuitive, and if properly arranged their structure can be identified by visual inspection as illustrated in Fig. 1.

Of course this method is infeasible for networks larger than a handful of nodes, prompting the development of automated detection techniques. The formulation of an algorithm and, more importantly, the validation of its output requires a more concise definition of a community. Newman and Girvan (2004) were among the first to address this issue and proposed *modularity* to quantify the strength of community structure. This metric, based on the intuition that nodes within the same community should be more tightly connected than they would be by chance, has been adopted for a variety of uses including the validation and comparison of community structures (Newman and Girvan, 2004; Pons and Latapy, 2006), but also as an objective function for optimization algorithms to identify communities (Clauset et al., 2004; Donetti and Muñoz, 2004; Duch and Arenas, 2005; Newman and Girvan, 2004; Reichardt and Bornholdt, 2006; Ruan and Zhang, 2007). While modularity quickly became a de facto standard, we posit that it is important to carefully evaluate its utility in discovering community structure. *We show here that the maximum modularity does not necessarily coincide with the correct division of the network*; in this case algorithms that maximize modularity converge on a suboptimal solution, that is, miss the discovery of the actual and meaningful communities. *We demonstrate this using a variety of metrics on diverse datasets* for which the actual communities are known as ground truth.

Another issue with community detection algorithms is their computational complexity. These methods, rooted in graph theory, are often confounded by large networks and become fragile as datasets approach 10^5–10^6 (or more) nodes. We believe that in order to achieve this level of scalability, a method much simpler than an optimization algorithm must be employed. To this end, *we outline an intuitive approach to community detection based on random walks and compare it to several published algorithms using a variety of metrics*. Our experimental results show that this simple method is as good or better at discovering the true communities than other more complex algorithms. Finally, *we discuss several possible extensions* to the approach, and *demonstrate its scalability on a network of over 1 million nodes*, where other methods falter.

* Corresponding author. Tel.: +1 574 631 8716; fax: +1 574 631 9260.
E-mail address: nchawla@nd.edu (N.V. Chawla).

0167-8655/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2009.11.001

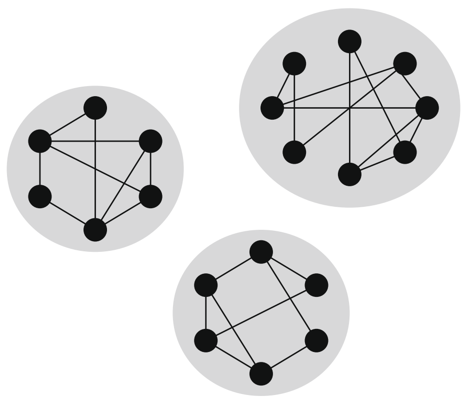

Fig. 1. In many networks the nodes form communities (shaded) within which the edges are more dense (solid lines) than between communities (dashed lines).

1.1. Organization

The remainder of this paper is organized as follows. Sections 2 and 3 discuss validation metrics and community detection algorithms, respectively. In Section 4 we present an experimental evaluation of the metrics and algorithms. Section 5 elaborates on community detection using random walks. We conclude with a discussion of our most notable observations and findings in Section 6.

2. Validation metrics

In this section, we examine modularity as well as several other quantitative measures for validating the output of community detection algorithms. While modularity relies strictly on network structure, the other three metrics use node labels when available and are true estimates of performance. Each method is presented and advantages and drawbacks are discussed; the metrics are applied in the evaluation and comparison of algorithms in Section 4.

2.1. Modularity

The most widely used and accepted metric designed specifically for the purpose of measuring quality of a network division into communities is *modularity* (Q) (Newman and Girvan, 2004), calculated as follows. Assume the network has been partitioned into k communities. Define a $k \times k$ symmetric matrix $\boldsymbol{e}$ whose element e_{ij} is the fraction of all edges in the network that link nodes in community i to nodes in community j. Computing the trace of this matrix $tr(\boldsymbol{e}) = \sum_i e_{ii}$ gives the fraction of all edges in the network that connect nodes in the same community. Obviously a high value of $tr(\boldsymbol{e})$ indicates strong community structure, but the trace alone is insufficient because placing all nodes in a single community would always result in the maximum value of $tr(\boldsymbol{e}) = 1$. To refine the metric let the row (column) sums $a_i = \sum_j e_{ij}$ denote the fraction of edges that connect to nodes in community i. Modularity is then defined as

$$Q = \sum_i \left(e_{ii} - a_i^2\right) = tr(\boldsymbol{e}) - \|\boldsymbol{e^2}\|, \tag{1}$$

where $\|\boldsymbol{x}\|$ represents the sum of the elements of matrix $\boldsymbol{x}$. Thus Q effectively measures the fraction of edges in the network that connect nodes in the same community minus the expected value of this quantity if the edges were placed at random. The value ranges from $Q = 0$, when the within-community edges are no better than random, to $Q = 1$, although Newman and Girvan (2004) found that real-world networks typically range from about 0.3 to 0.7. Note that modularity is more of a descriptive measure of data than a true performance metric, because it does not strictly quantify a good or a bad partitioning.

One advantage of modularity is that it can be computed using only connectivity of the network, in the absence of any node labels or other information. However, this property can also be considered a weakness because modularity is unable to incorporate metadata (e.g. node labels) even if it is available. The empirical comparison will help illustrate the practical consequences of this limitation, but first we present several alternate metrics for validating community structure from statistics and clustering literature, which operate under the assumption that true node labels are known a priori.

2.2. Accuracy

One very crude method of quantifying the ability of a community detection algorithm to identify the true network structure is *Accuracy*. Given a network consisting of n nodes wherein each node v is assigned a true label l_{tv}, the accuracy of a particular division of a network is calculated as follows. Assume the network has been partitioned into communities. For each community i, scan all nodes in i and identify the true label that occurs most frequently; this label is assigned as predicted label l_{pv} to each node v in the community. The accuracy is then defined as the fraction of all nodes whose predicted label l_{pv} equals the true label l_{tv}:

$$\text{Accuracy} = \frac{\sum_{v=1}^{n} \text{equal}(l_{tv}, l_{pv})}{n}, \tag{2}$$

where

$$\text{equal}(x,y) = \begin{cases} 1 & \text{if } x \text{ is identical to } y \\ 0 & \text{otherwise} \end{cases}.$$

This metric is easily computed from the data and takes into account (only) the node labels for evaluating the division of the network. Much like a classification task, it views each node individually and does not consider the communities as entities or relationships between them.

2.3. Rand Index

The Rand Index is a statistical tool used in clustering literature to measure the degree of overlap between two partitionings (Rand, 1971). It also requires two labels for each node – one corresponding to its true community and one to the predicted community – but the assignment is more straightforward. Given a network wherein each node v is assigned a true label l_{tv}, the Rand Index is computed as follows. Assume the network has been partitioned into communities. For each community i, simply assign every node the predicted label $l_{pv} = i$ and define the following quantities:

a = pairs of nodes i,j s.t. $l_{ti} = l_{tj}$, $l_{pi} = l_{pj}$
b = pairs of nodes i,j s.t. $l_{ti} = l_{tj}$, $l_{pi} \neq l_{pj}$
c = pairs of nodes i,j s.t. $l_{ti} \neq l_{tj}$, $l_{pi} = l_{pj}$
d = pairs of nodes i,j s.t. $l_{ti} \neq l_{tj}$, $l_{pi} \neq l_{pj}$

The Rand Index is then given by the ratio:

$$\text{Rand Index} = \frac{a+d}{a+b+c+d} = \frac{a+d}{\binom{n}{2}}, \tag{3}$$

and captures the extent to which the two partitionings agree with one another. The possible values range from 0 when there is no overlap to 1, indicating complete agreement. The Rand Index is also easily computed but, unlike accuracy, it incorporates a notion of communities by evaluating the relationship between *pairs of nodes*. One criticism of the Rand Index is that the expected value of two random partitions does not take a constant value (e.g. zero).

2.4. Adjusted Rand Index

To correct for the scaling problem Hubert and Arabie (1985) proposed the *Adjusted Rand Index (ARI)*, which varies between 0 and 1 according to expectation of random partitions. The expression for ARI takes the general form (index − expected index)/(maximum index − expected index), which can be computed as follows. Let n_{ij} be the number of nodes that have true label l_{ti} and predicted label l_{pj}, and let $n_{i\cdot}$ and $n_{\cdot j}$ be the number of nodes labeled l_{ti} and l_{pj}, respectively. These values can be summarized in a confusion matrix as shown in Table 1.

The Adjusted Rand Index can then be computed as

$$\mathrm{ARI} = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i\binom{n_{i\cdot}}{2}\sum_j\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}\right] - \left[\sum_i\binom{n_{i\cdot}}{2}\sum_j\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}}. \tag{4}$$

This formulation ensures that the expected value for a random partitioning is ARI = 0, while the value for a perfect agreement remains ARI = 1. ARI has all the same advantages of the Rand Index, but it is a more robust measure. An independent study of different indices for measuring agreement between partitions recommended that the ARI be used to validate clustering results Milligan and Cooper, 1986.

2.5. Normalized mutual information

The final metric we consider is an information-theoretic measure of the agreement between two partitions called *normalized mutual information (NMI)* (Fred and Jain, 2003). Similar to the Rand Indices, *NMI* assumes that the network was partitioned into communities and in each community i, every node v has been assigned the label $l_{v,p} = i$. NMI is computed as follows. If k_t denotes the number of true communities, then treating frequency counts like probabilities gives the entropy of true partition T as $H(T) = -\sum_{i=1}^{k_t}\frac{n_i^t}{n}log\left(\frac{n_i^t}{n}\right)$, where n_i^t represents the number of nodes in community i. The mutual information between true partition T and predicted community structure P can thus be computed as $I(T,P) = \sum_{i=1}^{k_t}\sum_{j=1}^{k_p}\frac{n_{ij}^{tp}}{n}log\left(\frac{n_{ij}^{tp}}{n}\Big/\left(\frac{n_i^t}{n}\cdot\frac{n_j^p}{n}\right)\right)$, and normalizing by the maximum value $\frac{H(T)+H(P)}{2}$ leads to the alternate definition:

$$\mathrm{NMI} = \frac{-2\sum_{i=1}^{k_t}\sum_{j=1}^{k_p} n_{ij}^{tp}\log\left(\frac{n_{ij}^{tp}\cdot n}{n_i^t\cdot n_j^p}\right)}{\sum_{i=1}^{k_t} n_i^t\log\left(\frac{n_i^t}{n}\right) + \sum_{j=1}^{k_p} n_j^p\log\left(\frac{n_j^p}{n}\right)}. \tag{5}$$

This formulation reveals one drawback of NMI, namely that this quantity is more complicated to compute from the output of a community detection algorithm than those previously discussed. However, it provides a statistically sound method for comparing different partitionings and has been shown to work well in practice (Fred and Jain, 2003).

Table 1
Overlap of partitions – or in this case communities – used to compute ARI.

True\predicted	l_{p1}	l_{p2}	$\cdots$	l_{pk_p}	Sums
l_{t1}	n_{11}	n_{12}	$\cdots$	n_{1k_p}	$n_{1\cdot}$
l_{t2}	n_{21}	n_{22}	$\cdots$	n_{2k_p}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
l_{tk_t}	n_{k_t1}	n_{k_t2}	$\cdots$	$n_{k_tk_p}$	$n_{k_t\cdot}$
Sums	$n_{\cdot1}$	$n_{\cdot2}$	$\cdots$	$n_{\cdot k_p}$	$n_{\cdot\cdot} = n$

3. Community detection

We now turn our attention to methods for automatically identifying community structure in complex networks. In this section, we consider three algorithms from literature: FastQ, which explicitly employs modularity optimization to guide hierarchical agglomeration (Clauset et al., 2004); WalkTrap, which performs agglomeration with a different heuristic (Pons and Latapy, 2006); and MCL, which partitions the network by simulating stochastic flow (van Dongen, 2000). In addition, we propose a fourth, much simpler approach to community detection based on random walks. An empirical evaluation follows, which compares the different algorithms presented here using all of the metrics described in Section 2.

3.1. Fast modularity

The *fast modularity*[1] algorithm (*FastQ*) is one of the more recent among the modularity maximization approaches (Clauset et al., 2004). The basic premise of the algorithm is the following: cognizant that optimizing modularity directly by trying all possible partitions is computationally infeasible (Brandes et al., 2007), FastQ instead uses modularity to guide a greedy hierarchical agglomeration process. Initially, each node is assigned to its own cluster. The task is thereby reduced to (repeatedly) finding an appropriate pair of nodes or communities, connected by an edge, to merge into a single new community.

Two questions remain: Which is the best pair to merge? and when should communities stop merging? The answer to both is, of course, modularity. To select a pair to merge, the algorithm computes Q for the current structure. Each possible merge operation is then considered and the one with the highest ΔQ is selected. Modularity also determines the final partitioning. The algorithm is run to completion (until all nodes belong to one giant community) producing a dendrogram, and Q is recorded at each iteration. A plot of Q vs. the merge steps can then help determine the appropriate point to cut the dendrogram. The global maximum likely corresponds to the best community structure, or a domain expert can select among multiple local maxima.

The FastQ algorithm has a computational complexity of $O(md\log n)$, which could be $O(n^3\log n)$ in the worst case (if $m \sim n^2$ and $d \sim n$). But in most real-world networks $m \sim n$ and $d \sim log(n)$, resulting in a complexity of $O(nlog^2n)$ (Clauset et al., 2004).

[1] Source code for the fast modularity algorithm is available for download at http://cs.unm.edu/~aaron/research/fastmodularity.htm.

3.2. WalkTrap

As the name suggests, the *WalkTrap*[2] algorithm employs the idea of random walks through the network for community detection (Pons and Latapy, 2006). More specifically, the authors propose a node similarity measure based on short walks and show that it provides sufficient information to be used (instead of modularity) for community detection via hierachical agglomeration. However, modularity is still applied as stopping criterion and metric for comparing their results to other algorithms. WalkTrap has complexity $O(mn^2)$, which could be $O(n^4)$ in the worst case but according to Pons and Latapy (2006) behaves as $O(n^2 \log n)$ on real-world networks.

3.3. Markov clusters

The *Markov cluster algorithm*[3] (MCL) is implemented as a simulation of flow through the network (van Dongen, 2000), but the underlying intuition here also stems from a contemplation of random walks. The author stipulates that a random walker placed in a network would spend a disproportionately long time walking around the same community before crossing into a different one. Therefore, assuming the walk starts at some node *i*, if another node *j* has a high probability of being visited during the random walk, then the probability of nodes *i* and *j* belonging in the same community is also high.

Building on these principles, the MCL algorithm uses a series of alternating expansions and inflations to identify weakly connected components by simulating network flow until an equilibrium state is reached. The amount of flow along a given path is related to the likelihood of traversing that path, similar to the probability of starting at some node *i* and visiting node *j*. As the experimental results in Section 4 show, this method is quite effective at identifying community structure. However, performance comes at a premium: the algorithm uses a matrix-representation of the network and operations rely on manipulating this data, which can be costly both in space and time. Therefore, using the default settings only work for relatively small networks, and several parameters may require tweaking to process larger networks.

The MCL algorithm has complexity $O(n^3)$, but van Dongen (2000) describes an optimization which, under certain assumptions, can significantly reduce the effective execution time for sparse networks.

3.4. Proposed method: random walks

The authors of the two previous methods both note that the behavior of a random walker in a network can be related to the concept of a community, yet only use this notion implicitly in their algorithms. We implemented a method that applies this idea directly by *actually taking random walks*. It only seems reasonable that a random walker with a limited number of steps is more likely to remain within the community than crossing community boundaries; keeping track of nodes visited during the walk serves as evidence that they should belong to the same community. We exploited this property to devise a simple, intuitive method for community detection using random walks.

The basic idea is to perform many short random walks and interpret visited nodes during the same walk as an indication that they belong in the same community. This information is aggregated over all walks and used as the basis for creating a consensus clustering representing the community structure of the network. Pseudocode outlining this procedure is shown in Algorithm 1.

Algorithm 1. Community detection with random walks

```
Input: num_steps, the length of the random walks
1: for all nodes $i = 1,\ldots,n$, $j = 1,\ldots,n$ do
2:   $S[i][j] = 0$
3: end for
4: for each node $start\_node = 1, \ldots, n$ do
5:   $i = start\_node$
6:   $C = \{start\_node\}$
7:   for number of steps $h = 1,\ldots,num\_steps$ do
8:     randomly select $next\_node$ from $neighbors(i)$
9:     $C = C \cup \{next\_node\}$
10:      $i = next\_node$
11:   end for
12:   for each node $i \in C$ do
13:     for each node $j \in C$, $i \neq j$ do
14:       $S[i][j] += 1$
15:     end for
16:   end for
17: end for
```

We first define an $n \times n$ similarity matrix S to aggregate the walks, where each entry $S[i][j]$ denotes the similarity of nodes i and j; all entries are initialized to zero. Every node in the network is then used as starting point for a random walk once. From that node some user-specified number of steps *num_steps* is taken through the network, selecting the next node probabilistically from all neighbors (a node may be visited any number of times during this walk).

Nodes reached during one such walk are recorded in set C as evidence of belonging to the same community. After each walk, entries in S corresponding to the nodes in C are incremented. The number of steps can either be determined based on some graph-theoretic measure (e.g. diameter, number of nodes) or provided as input by the user. Once all walks are complete, each entry in the matrix denotes how often two nodes appeared along the same walk. A higher value indicates an increased likelihood of belonging to the same community.

Now we are left with a similarity matrix and the task of extracting community structure in an efficient manner, which is analogous to the problem of clustering spatial data. We use an agglomerative technique similar to the hierarchical clustering schemes described by Johnson (1967). The general idea is to iteratively merge nodes into communities according to their similarities, starting with the highest. Algorithm 2 outlines this procedure, where $dim(S)$ indicates the dimension of the similarity matrix.

Algorithm 2. Merging clusters into a consensus clustering representing the community structure of the network

```
1: Let $C = \{1, 2, \ldots, n\}$ be the set of communities
2: while $\underset{(i,j)}{argmax}\, S[i][j] > 0$ do
3:   $(max\_i, max\_j) = \underset{(i,j)}{argmax}\, S[i][j]$
4:   for each cluster $m = 1 \ldots dim(S)$ do
5:     $S[max\_i][m] = \text{merge}(S[max\_i][m], S[max\_i][max\_j])$
6:   end for
7:   Remove column $max\_j$ from $S$
8:   $C \cdot max\_i = C \cdot max\_i \cup C \cdot max\_j$
9:   Remove cluster $C \cdot max\_j$ from $C$
10: end while
11: return set of communities $C$
```

With this procedure, the dimension of the similarity matrix is reduced by one at each step. Different variations of the algorithm

[2] Source code of the WalkTrap algorithm is available for download at http://www.liafa.jussieu.fr/~pons/index.php?item=prog.

[3] Source code of the MCL algorithm is available at http://micans.org/mcl/#source.

arise from the way the new similarities are computed when the rows (columns) corresponding to two nodes or communities are merged (line 5). Several options including the minimum, maximum, and mean methods are described in literature Johnson (1967), Sokal and Sneath (1963). We empirically determined that the mean method works best for merging networks; results are shown in the next section. Depending on the agglomeration method, the complexity can be $O(n^2 \log n)$ or $O(n)$. We detail the complexity analysis in Section 5.2.

4. Experimental results

In this section we present our experiments evaluating the different community detection algorithms using various metrics; algorithm scalability is discussed in Section 5. We begin by describing the datasets used, followed by individual subsections explaining the setup for each experiment along with the corresponding results and analyses.

4.1. Datasets

Because this work demands thorough qualitative and quantitative evaluation of the experimental results, datasets used for evaluation satisfy the following criteria:

- True community structure known *a priori (node labels)*
- *Size of dataset sufficiently small to visualize* & *interpret results by inspection*
- *Suitable for all algorithms (for example, some require full connectivity)*

In addition, we limited ourselves to publicly available or easily reproducible datasets, resulting in the selection of the following three real-world networks: Zachary (1977) karate club, a map of the popular board game Risk[4], and a network of NCAA Division-I football programs (Girvan and Newman, 2002). Properties of these datasets are summarized in Table 2 and visualizations of the networks, including their true community structure as indicated by the node shading, are shown in Figs. 2–4.

4.2. Merging the similarity matrix

We remind the reader of our discussion about computing new distances for the similarity matrix when merging clusters in Section 3.4. Before evaluating the algorithms and validation metrics, we first want to determine which agglomerative method is the most effective. If we assume clusters i and j are getting merged and let $d(i \cup j, k)$ be the distance of the new cluster $\{i \cup j\}$ to some arbitrary cluster k, then we consider the following options:

(1) Minimum distance: $d(i \cup j, k) = min(d(i,k), d(j,k))$
(2) Maximum distance: $d(i \cup j, k) = max(d(i,k), d(j,k))$
(3) Mean distance: $d(i \cup j, k) = \frac{d(i,k)+d(j,k)}{2}$

The algorithm was implemented using a walk length equal to the number of nodes in the network. We performed experiments on all three datasets and cut the dendrogram at the (known) number of true communities. To account for the element of randomness in the algorithm, all values reported here are an average of ten runs using different seeds. The results are shown in Table 3.

For space reasons we report only modularity as this is a tangential result, but we performed the same experiments with other metrics as well and found trends consistent with those shown here. The most important observation is that rank order of the three distance metrics is identical across all datasets, namely mean performs best, followed closely by max, with min in distant third place. We only use mean distance for the remaining experiments.

[4] http://en.wikipedia.org/wiki/Risk_(game).

Table 2
Summary of network datsets.

Dataset	Nodes	Edges	Communities
Karate	34	78	2
Risk	41	83	6
Football	115	613	12

4.3. Evaluating validation metrics and community detection algorithms

This section contains our most significant empirical results. We ran each algorithm discussed in Section 3 on all three datasets and, for every output, computed all metrics presented in Section 2. We also provide the modularity for the true communities as reference. The results are summarized in Table 4.

We used default parameters except in the following case. All algorithms from literature found the correct number of communities for the Karate and Risk networks. However, for the Football dataset two of the algorithms produced the incorrect number of communities – six for FastQ and ten for WalkTrap. To ensure a fair comparison we subsequently provided the algorithms with the correct number (twelve) and repeated the experiments. Results from these runs are shown under the heading *Football-12*. We rank the algorithms according to their performance as measured by the different metrics. For the Football network, we always use the better of the two values (Football or Football-12). In case of a tie among algorithms, the average rank is assigned to each. In the rightmost column we also report the average ranks for each metric where the algorithm with the highest rank is shown in **bold**. In the following subsections we provide a discussion and analysis of the results.

4.3.1. Validation metrics

We first assess our concern that the maximum modularity does not necessarily correspond to the true communities in the network. As shown in Table 4, for each network at least one algorithm finds a division with modularity greater than the true communities. We also observe that even modularity close to that of the true communities does *not* mean that the correct structure was found. For example, the division of the Football network produced by FastQ has a modularity of 0.577, the same value as the true communities. However, the other metrics suggest that this is deceiving as the partitioning obtained with FastQ is actually quite dissimilar from the true communities. Further, it is worth noting that higher modularity does not imply better structure, as the correct division for the Karate network has a modularity of only 0.371 but even poor divisions of the Football network exceed 0.5. Therefore we caution against the use of modularity as the only criterion for the evaluation and comparison of community detection algorithms and instead advocate a combination of metrics evaluating both the structure and the agreement with true communities to provide a more robust validation.

4.3.2. Algorithm comparison

Here we compare algorithm performance by examining average ranking over all datasets for each of the validation metrics. First, we find that FastQ produces the highest modularity, which is not surprising since the algorithm greedily optimizes on modularity. WalkTrap and random walks rank slightly lower with similar

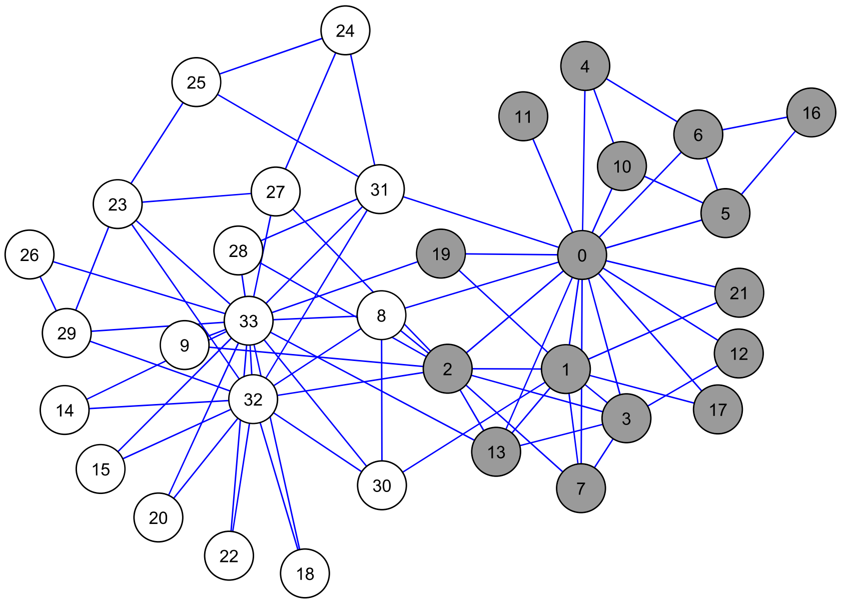

Fig. 2. Zachary's karate club network; node shading indicates group membership.

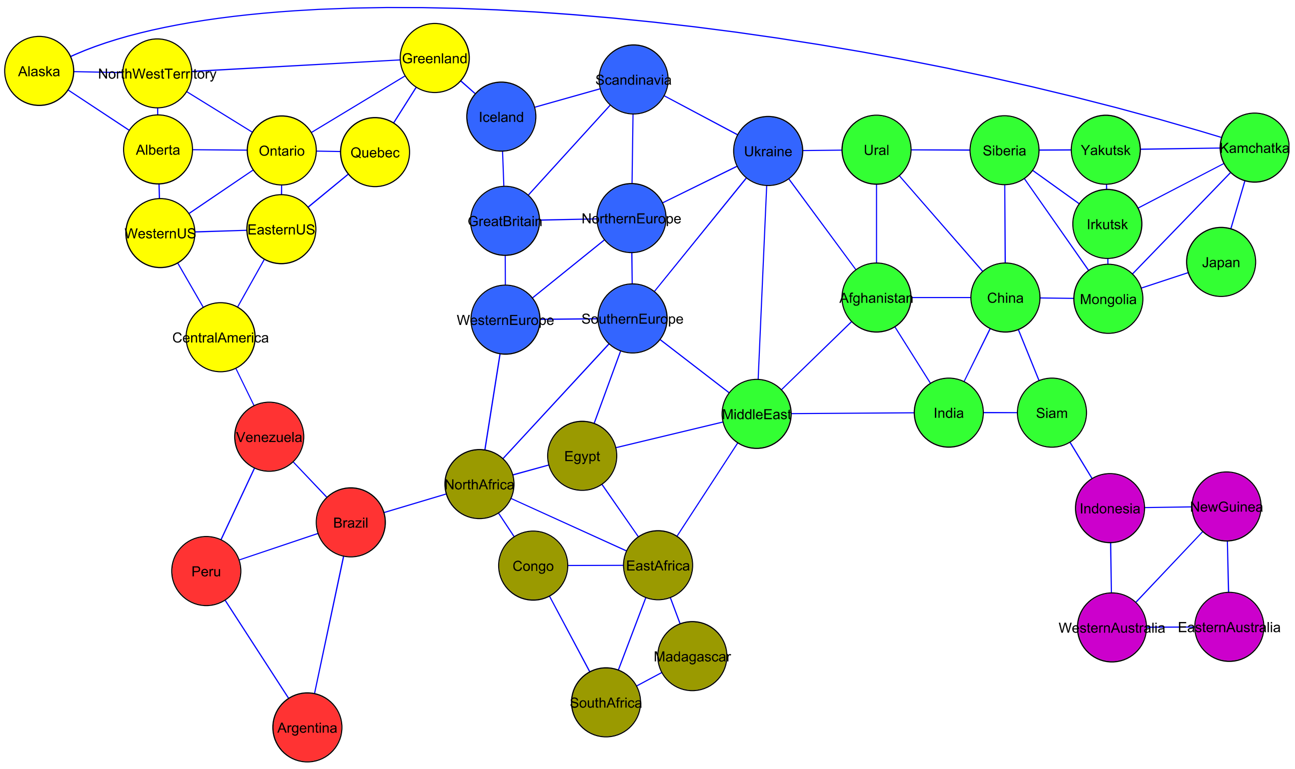

Fig. 3. Risk map; node shading indicates the six distinct continents in the game.

values, while MCL comes in last place for all datasets. However, when we move beyond modularity and determine how successful each algorithm is at discovering true network structure, the results look drastically different. We find that FastQ actually ranks last for accuracy, primarily due to difficulties with the football network. Random walks achieve the highest accuracy on average, with the other two algorithms falling inbetween. These trends largely hold for the other metrics as well. Rand Index and its variant ARI produce very similar rankings. Most notably, random walks again achieve the highest rank, followed by MCL, FastQ, and WalkTrap. NMI also echoes these findings, although it is somewhat more favorable for FastQ than the others. The remainder of this paper further explores extensibility and scalability of the random walks approach.

5. Extending random walks

In this section, we elaborate on the flexibility of the random walks approach by proposing extensions that enable it to take into

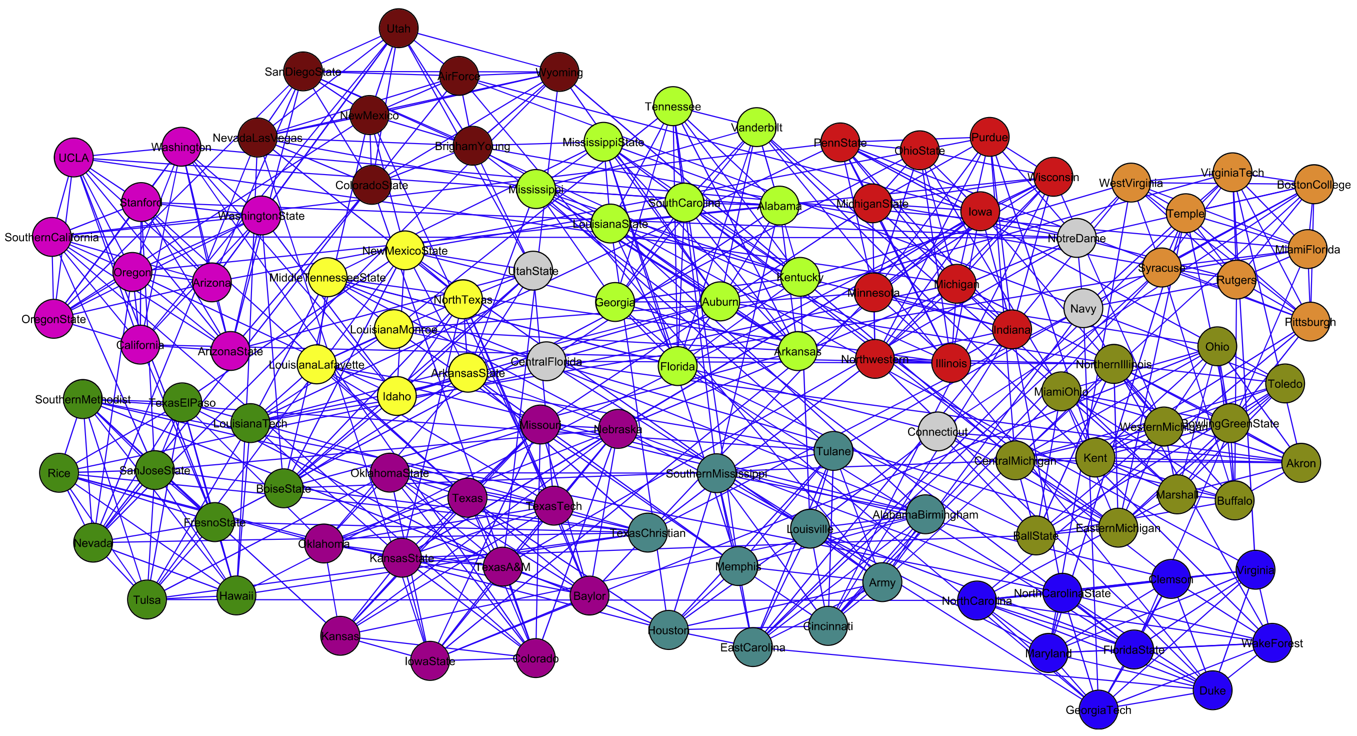

Fig. 4. Network of football schedules; node shading indicates the NCAA conferences.

account additional data, such as edge weights or node attributes, and explain how this can lead to more meaningful results. We also compare its computational complexity to the other algorithms, identify the performance bottleneck, describe a modification that significantly reduces complexity, and demonstrate scalability of the new version by applying it to a real-world network of over 1 million nodes.

5.1. Metadata and weighted networks

Real-world networks are often constructed from datasets that contain more information than just connectivity of entities. For instance, a social network dataset might have demographic data or a list of interests available for each person (Steinhaeuser and Chawla, 2008). These are examples of *node attributes*, because they characterize nodes in the network. Other information, such as the frequency or mode of communication between individuals, can be represented as *edge weights* and indicate the strength of relationships between individuals in the network.

Although it seems reasonable that the additional data is valuable for community detection, most algorithms cannot incorporate this information to aid the discovery process. Yet simple extensions to the random walks method enable it to take into account both types of data. Edge weights can be used in a straightforward manner: instead of selecting the next step uniformly at random among all neighbors, simply base the probability of selection on edge weights connecting the neighbors. This makes visiting a strongly connected neighbor more likely, which is desirable in this context. Node attributes are not considered directly here, but they can be used to compute meaningful edge weights. For this purpose, we conceived a simple heterogeneous weighting function called

Table 3
Comparison of methods for hierarchical merging of clusters (modularity Q).

Dataset	Min	Max	Mean
Karate	0.085	0.132	0.371
Risk	0.222	0.502	0.604
Football	0.095	0.582	0.591

Table 4
Comparison of community detection algorithms on three datasets using different validation metrics. For modularity, we also report the value for the known community structure as a reference for each dataset. In cases where the default parameter produces an incorrect number of communities, the correct number was specified (indicated by Football-12) and the better of the two values used for the computation of ranks. We report the rank of each algorithm per dataset (in parentheses) and the algorithm with highest average rank for each metric is shown in bold.

Algorithm\Dataset	Karate	Risk	Football	Football-12	Average rank
Modularity (Q)					
True Communities	*0.371*	*0.621*	*0.577*		
FastQ	0.381 (1)	0.625 (1)	0.577 (3)	0.535	**1.67**
WalkTrap	0.360 (3)	0.624 (2)	0.604 (1)	0.602	2
MCL	0.359 (4)	0.617 (4)	0.596 (4)	–	4
Random walks	0.371 (2)	0.623 (3)	0.598 (2)	–	2.33
Accuracy					
FastQ	0.971 (2.5)	0.929 (3)	0.548	0.591 (4)	3.17
WalkTrap	0.941 (4)	0.929 (3)	0.878	0.939 (2)	3
MCL	0.971 (2.5)	0.929 (3)	0.939 (2)	–	2.5
Random walks	1.000 (1)	0.976 (1)	0.939 (2)	–	**1.33**
Rand Index					
FastQ	0.941 (2.5)	0.952 (2)	0.882	0.887 (4)	2.83
WalkTrap	0.886 (4)	0.951 (3)	0.973	0.987 (2)	3
MCL	0.941 (2.5)	0.947 (4)	0.987 (2)	–	2.83
Random walks	1.000 (1)	0.979 (1)	0.987 (2)	–	**1.33**
[Hubert and Arabie] Adjusted Rand Index (ARI)					
FastQ	0.882 (3)	0.834 (2)	0.492 (4)	0.486	3
WalkTrap	0.772 (4)	0.832 (3)	0.833	0.915 (2)	3
MCL	0.883 (2)	0.815 (4)	0.915 (2)	–	2.67
Random walks	1.000 (1)	0.927 (1)	0.915 (2)	–	**1.33**
Normalized mutual information (NMI)					
FastQ	0.837 (2)	0.894 (2)	0.732 (4)	0.727	2.67
WalkTrap	0.498 (4)	0.848 (3)	0.898	0.935 (2)	3
MCL	0.836 (3)	0.834 (4)	0.935 (2)	–	3
Random walks	1.000 (1)	0.955 (1)	0.935 (2)	–	**1.33**

Table 5
Summary of community detection algorithms (complexity for real-world networks).

Algorithm	Complexity	References	Comments/assessment
FastQ	$O(nlog^2n)$	Clauset et al. (2004)	Computationally efficient, limited by the drawbacks of modularity
WalkTrap	$O(n^2 \log n)$	Pons and Latapy (2006)	Finds divisions similar to FastQ, but at a higher complexity
MCL	$O(n^3)$	van Dongen (2000)	Better divisions, but matrix manipulations limit scalability
Random walks	$O(n^2 \log n)$	This work	Divisions best matching true structure, limited by merge step
Scalable random walks	$O(n)$	This work	Finds good divisions with high efficiency, but still parameterized

node attribute similarity (NAS). For an edge between a pair of nodes i and j, NAS is computed as follows: for each nominal attribute a_n, if i and j have the same value then increment the edge weight by one:

$$\text{if equal } (i \cdot a_n, j \cdot a_n), \text{ then } w_a(i,j) += 1.$$

For continuous attributes, we first normalize each attribute to (0,1) over all nodes and then take the arithmetic difference between the pairs of attribute values to obtain a similarity. More formally, for each continuous attribute a_c:

$$w_a(i,j) += (1 - |i \cdot a_c - j \cdot a_c|).$$

The computed weights can then be used like weights provided explicitly as described above.

5.2. Complexity and scalability

To describe the complexity of the random walks method, let the length of a walk be denoted by l. If we assume an adjacency-list representation of the network (so that neighbors of a node are accessible in constant time), then the walk phase has a complexity of $O(ln)$. Although we suggested a walk length of $l = n$ earlier, we empirically determined that walks of only a few steps suffice, effectively reducing the complexity to $O(n)$. But this quantity is dominated by the agglomeration phase. The mean method we have described has a complexity of $O(n^2 \log n)$ when implemented with a heap to store the similarities. This complexity makes the application to very large networks computationally prohibitive. Therefore, we must devise a more efficient agglomeration technique.

Fred and Jain (2002) discuss a concept they call evidence accumulation (EA) for clustering spatial data. This method combines similarity-based clusters more efficiently. First, entries in the similarity matrix are normalized to (0,1). Instead of finding the maximum element at each iteration, EA then takes a threshold $0 < t < 1$ and assigns all elements with a similarity greater than t to the same community. In the worst case this requires $n - 1$ operations, reducing the overall complexity from $O(n^2 \log n)$ to $O(n)$. We implemented the method with random walks and determined that, for all datasets used here, it achieves performance comparable to agglomeration by merging.

To summarize, we provide an overview of the algorithms in Table 5. There are advantages and disadvantages of each, and the user must carefully weigh them when choosing the best algorithm for a given application.

5.3. Experiments on cell phone communications network

To demonstrate scalability of the modified random walks algorithm, we apply it to a large real-world social network constructed from cellular phone records (Madey et al., 2007). The data was collected by a major non-American service provider during the period from March 16 to April 15, 2007. Representing each customer as a node and placing an edge between pairs of users who interacted via a call or text message, we obtain a network of 1,341,960 nodes. In addition to improved scalability, we also take advantage of node attribute similarity by weighting the network based on demographic information (age and gender) of the customers. We ran the modified randoms walk method on this network using a threshold of $t = 0.1$, although we observed that the efficiency and resulting structure were not sensitive to variations in threshold.

The most significant result of this experiment is the execution time as it took just under 40 s to complete. Prior work by Pons and Latapy (2006) showed that many other algorithms can become intractable for datasets one and even two orders of magnitude smaller (10^4–10^5 nodes). The partitioning consisted of thousands of communities with a modularity of 0.911, but as pointed out by Fortunato and Barthélemy (2007) this value should be taken with some caution. Unfortunately, we do not (yet) have labels indicating true communities in the network and are therefore unable to evaluate the practical significance of the structure, but direct marketing strategies based on these communities are under consideration.

6. Conclusion

Here we have addressed two closely related problems: community detection in complex networks and validation of community structure. First, we showed that divisions with maximum modularity do not necessarily correspond to true community structure. We then identified alternate metrics from clustering literature and evaluated their utility in this context. Based on our experimental results, we conclude that a combination of metrics should be used to provide more robust validation of a network partition.

Concurrently we performed an empirical comparison between three different community detection algorithms from literature and included a fourth, simple method based on random walks. We found that modularity maximization only performed well when measured by modularity itself, but rather poorly in terms of the other metrics measuring agreement with the true structure. The random walks approach, however, performed very well in discovering the community structure, prompting us to further explore its capabilities.

Finally, we highlight the flexibility afforded by the general framework of the random walks approach and explore variations and extensions. We analyze the algorithm and suggest a simplification that leads to a significant reduction in computational complexity; we also show how to incorporate additional data and improve the community detection process. We demonstrate scalability of this method by applying it to a large real-world social network using node attribute similarity to identify community structure.

Acknowledgements

The authors thank Aaron Clauset, Pascal Pons, Stijn van Dongen and Mark Newman for making software and datasets available. The authors are also thankful to László Barabási for providing the cell phone data. This work was supported in part by the National Science Foundation under Grant No. 0826958.

References

Asur, S., Ucar, D., Parthasarathy, S., 2007. An ensemble framework for clustering protein–protein interaction. In: Proc. ISMB.

Brandes, U., Delling, D., Gaertler, M., et al., 2007. On finding graph clusterings with maximum modularity. LNCS, vol. 4769. Springer, Berlin, Heidelberg. pp. 121–132.

Clauset, A., Newman, M., Moore, C., 2004. Finding community structure in very large networks. Phys. Rev. E 70, 066111.

Donetti, L., Muñoz, M., 2004. Detecting network communities: A new systematic and efficient algorithm. J. Stat. Mech. Theor. Exp., P10012.

Duch, J., Arenas, A., 2005. Community detection in complex networks using extremal optimization. Phys. Rev. E 72, 027104.

Enright, A.J., Van Dongen, S., Ouzounis, C.A., 2002. An efficient algorithm for large-scale detection of protein families. Nucl. Acids Res. 30 (7), 1575–1584.

Fred, A., Jain, A., 2002. Data clustering using evidence accumulation. In: Proc. ICPR.

Fred, A., Jain, A., 2003. Robust data clustering. In: Proc. CVPR.

Fortunato, S., Barthélemy, M., 2007. Resolution limit in community detection. Proc. Natl. Acad. Sci. USA 104 (1), 36–41.

Girvan, M., Newman, M., 2002. Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99, 7821–7826.

Hubert, L., Arabie, P., 1985. Comparing partitions. J. Classif. 2 (1), 193–218.

Johnson, S., 1967. Hierarchical clustering schemes. Psychometrika 32 (3), 241–254.

Madey, G., Barabási, A.-L., Chawla, N.V., et al., 2007. Enhanced situational awareness: Application of DDDAS concepts to emergency and disaster management. LNCS, vol. 4487. Springer, Berlin, Heidelberg. pp. 1090–1097.

Milligan, G., Cooper, M., 1986. A study of the comparability of external criteria for hierarchical cluster analysis. Multivar. Behav. Res. 21 (4), 441–458.

Newman, M., Girvan, M., 2004. Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113.

Pons, P., Latapy, M., 2006. Computing communities in large networks using random walks. J. Graph Algorithms Appl. 10 (2), 191–218.

Rand, W., 1971. Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. 66 (336), 846–850.

Reichardt, J., Bornholdt, S., 2006. Statistical mechanics of community detection. Phys. Rev. E 74, 016110.

Ruan, J., Zhang, W., 2007. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In: Proc. ICDM.

Sokal, R., Sneath, P., 1963. Principles of Numerical Taxonomy. W.H. Freeman and Co., San Francisco.

Steinhaeuser, K., Chawla, N.V., 2008. Community detection in a large real-world social network. In: Liu, H. et al. (Eds.), Social Computing, Behavioral Modeling, and Prediction. Springer, US, pp. 168–175.

van Dongen, S., 2000. Graph Clustering by Flow Simulation, Ph.D. Thesis, University of Utrecht, Netherlands.

Wasserman, S., Faust, K., 1994. Social Network Analysis: Methods and Applications. Cambridge University Press.

Zachary, W.W., 1977. An information flow model for conflict and fission in small groups. J. Anthropol. Res. 33 (4), 452–473.