Bloom: a stochastic growth-based fast method of community detection in networks

Phillip Schumm and Caterina Scoglio

Published Version Information

Citation: Schumm, P., & Scoglio, C. (2012). Bloom: A stochastic growth-based fast method of community detection in networks. Journal of Computational Science, 3(5), 356-366.

Digital Object Identifier (DOI): doi:10.1016/j.jocs.2012.03.006

Publisher's Link: http://www.sciencedirect.com/science/article/pii/S1877750312000269

Bloom: A Stochastic Growth-based Fast Method of Community Detection in Networks

Phillip Schumm and Caterina Scoglio
Sunflower Networking Group and K-State Epicenter
Electrical and Computer Engineering Department,
Kansas State University, Manhattan KS, USA
Email: {pbschumm, caterina}@ksu.edu

Abstract

Networks are characterized by a variety of topological features and dynamics. Classifying nodes into communities, community structure, is important when exploring networks. This paper explores the community detection metric called modularity. The theoretical definitions of modularity are connected with intuitive insights into the compositions of communities. Local modularity costs/benefits are explored and an efficient stochastic algorithm, Bloom, is introduced, based on growing communities using local improvement measures. Three extensions of Bloom are presented that build upon the basic version. A numerical analysis compares Bloom with the popular fast-greedy algorithm and demonstrates the successful performance of the three modifications of Bloom.

I. Introduction

Many systems can be understood as networks, as sets of actors and the interactions that occur among them. In understanding systems in this way, networks have been explored through a variety of metrics and methods [1][2][3]. The community structure of networks explores the organization of nodes into communities. Within simple networks, a community is usually understood to be a set of nodes that are well connected within themselves and less connected to the remainder of the network. Network communities can play critical roles in understanding individual function as well as system-wide dynamics such as epidemics on networks [4]. Community structure is often described through a topological metric known as modularity [5], which is a measure that attempts to capture the strength of a division of the topology into 'modules' or communities. Several methods have been developed to divide networks into communities based on maximizing this measure [6][7] and extensive work has been done to better understand and characterize the (maximal) modularity of a network with respect to the expected value of modularity for a random network, the upper bounds, and the partitioning resolution [7][8][9][10][11][12][13][14]. Many other metrics have been used to search out communities in networks and some of these can be seen in [15][16]. It is important to note that modularity is a topology-based partitioning or clustering method as opposed to a data-dimensional clustering method [17]. Past work has highlighted several interesting aspects of modularity including that the modularity of real world networks rarely approaches its maximum possible value of one [5][18] and how the partitioning problem is NP hard and sensitive to even the simplest network perturbation [14][19]. When considering the computation of modularity, the two objectives of most community detection methods are either efficient running times or performance in obtaining higher value of the objective function given by modularity. The optimal method is the ILP formulation in [19], yet with an exponential running time, while well-known, efficient heuristics include the fast-greedy method [20], label passing [21], and map equation methods [22][23].
The content and contributions of this paper are distributed as follows. In section II, we introduce modularity, present theoretical insights into what makes a good community according to modularity's composition, and explain one reason why real world networks rarely can be partitioned with modularity values near one. Section III introduces the efficient, stochastic algorithm, Bloom, which creates a diverse set of solutions to the hard or crisp clustering problem that helps to capture the degenerate characteristic of the modularity function for a network. In section IV, a numerical comparison between Bloom and the fast-greedy heuristic is presented and we demonstrate the potential for modified versions of the basic Bloom algorithm to outperform more complex ones. We conclude the work of this paper in section V.

II. Modularity

Let N be the number of nodes and L the number of edges within a graph G. A binary element of the adjacency matrix of G, a_{ij}, represents the existence of an edge between nodes i and j. The degree of node i, d_i, counts the number of nodes connected to node i. A partitioning P of a graph divides the nodes into c communities $X_1, X_2, ..., X_c$. The quality measure modularity is then expressed as a function of a partitioning P on a graph G,

$$Q\left(P,G\right)=\frac{1}{2L}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(a_{ij}-\frac{d_i d_j}{2L}\right)\delta_{c_i,c_j}, \tag{1}$$

where c_i represents the community of node i. An equivalent expression derived in [24] makes use of the volume or degree-sum of community,

$$Q\left(P,G\right)=\frac{1}{2L^2}\sum_{r=2}^{c}\sum_{s=1}^{r-1}D_{X_r}D_{X_s}-\frac{L_{\text{inter}}}{L}, \tag{2}$$

where D_{X_r} is the sum of the degrees of the nodes within community X_r. This expression sorts the edges of a graph into those which connect within each community X_r (L_r) and those which stretch between the communities (L_{inter}).

Equation 1 represents the contribution of each pair of nodes to the modularity value as given by the existence of an edge within a community minus the expectation (from a graph construction based on a given degree sequence) of an edge to lie between the pair. Alternatively, equation 2 composes the modularity value from a product over all pairs of the community volumes less a second term that counts the edges lying between the communities. In this community level description of modularity, the second term is readily understood. Communities should be clearly defined with sparse interconnections; thus reducing L_{inter} directly increases Q. In a dual manner, the contents of the sum in equation 1 point out that any pair of nodes that belong to the same community and do not have an edge connecting them will decrease Q. Shown in [24], modularity is bounded above as

$$Q\left(P,G\right)\leq 1-\frac{1}{c}-\frac{L_{\text{inter}}}{L} \tag{3}$$

when the community degree-sums are uniform. This implies that the modularity value can be increased by a partitioning that reduces the heterogeneity among the community volumes. Therefore the modularity function is maximized by a partitioning that assigns communities of similar volumes that have very few edges that fall across the community boundaries. Given a graph G with N nodes, it is possible to design a theoretical maximum-Q topology and a corresponding partitioning for the maximal Q. Given a simple graph with N nodes, let us first set L_{inter} to zero, creating disconnected components. Let the partitioning P identify each component as a community. The second desirable trait is balanced community volumes. Although it is significantly more than what is necessary, making the graph d-regular with identically sized communities will create symmetric and balanced communities. The bound simplifies to

$$Q\left(P,G\right)\leq 1-\frac{1}{c} \tag{4}$$

as noted in [19][24][25]. Maximizing Q is the same as maximizing c so let us consider this. As a side note, the work of [14] suggests that the more communities a graph has, the more degenerate will be the solution space of determining the maximal partitioning. If each community has n_c nodes, then the number of communities, assuming N is divisible by c, is given by $c=N/n_c$. Within the simple graph described, we note the constraint on d,

$$d\leq n_c-1.$$

Therefore $n_c\geq d+1$ and

$$c\leq\frac{N}{d+1}$$

For a d-regular graph with N nodes and equally sized communities, we find

$$Q\left(P,G\right)\leq 1-\frac{d+1}{N}. \tag{5}$$

This bound is tight when each community is a clique ($d=n_c-1$). Bound 5 shows Q increasing as the graph becomes sparser. This agrees with the analysis of [8] on Erdos Renyi graphs. For a complete graph with $d=N-1$, the well known result $Q=0$ appears. With no constraint on the number of edges, this bound would lead us to remove all the edges from our graph. Rather than an edgeless graph, we consider the case when $d=1$. Assuming N is even, the resulting maximally modular graph would be composed of $N/2$ cliques (also components), each having 2 nodes. This yields $Q=1-2/N$. Thus for a graph with edges to be partitioned such that $Q=1$, this graph would need to have N tending to infinity. If we also have a given number of edges L, then

$$Q\left(P,G\right)\leq 1-\frac{2L+N}{N^2}, \tag{6}$$

and the bound is tight with appropriate divisibility. Theoretically for a graph of N nodes and L edges, the uniform optimal degree d^* is given by $2L/N$ and the optimal number of communities c^* is given by $N/(d^*+1)$. In constructing a similar graph from a given node degree sequence, it is likely that the partitioning and number of communities will drift significantly from the above discussion due to challenges that arise in creating balanced, dense communities with heterogeneous degrees.

Locally, as seen in equation 1, any pair of nodes that belong to the same community and do not have an edge connecting them will decrease Q. However, it is interesting to note that a pair of nodes belonging to the same community and connected by an edge may still decrease Q. This happens when the expected number of edges $\left(d_i d_j\right)/\left(2L\right)$ is greater than one[1]. Although

[1]A simple example of this: Take two stars, each with 5 leaf nodes and connect them with another edge. The expected number of edges between the two hubs is $36/22$ and the corresponding entries in the modularity matrix are both $-7/11$.

local to an edge that connects two nodes of sufficiently high node degrees, this is a deterrent to these two high degree nodes belonging to the same community. Thus it could be said that nodes of relatively high degree don't fit together in communities or at least they are discouraged locally by the contribution of their shared edge to Q. If they do not share an edge, this deterrent will be significantly stronger.

At the community level, the decrease in the number of edges that do not fall within a community, L_{inter}, and the balance of the degree-sum (volumes) of the communities, D_{X_r}, are the driving forces that best maximize modularity. Due to topological properties, these objectives may drive the optimal partitioning away from intuitively dense communities. Consider a sparse graph that is characterized by a scale free degree distribution or that, at least, contains a few nodes with notably high degrees. Social networks typically contain several such hubs [26]. Each node of relatively high degree (hub) will either be partitioned into a large community with a size approaching its degree or several of its adjacent edges will fall outside of its community. This forcing of a larger community leads to a lower density of this community, while the larger size (by its degree-sum) of the community encourages all other communities to similar (larger) degree-sums, which will encourage larger and less-dense communities. Interestingly, the authors of [24] show that the maximum difference between any pair of community volumes can be roughly estimated by a factor of $\sqrt{2}$ times the average D_X. They also derive a lower bound on Q that is a quadratic function of this maximum difference. With this estimate, the heterogeneity of the community volumes cannot be significantly increased without a corresponding increase in the average community volume. Therefore, when a community has a relatively large community volume, perhaps from containing one or two relatively high degree nodes, the optimization of Q will tend to increase the other community volumes, reducing the total number of communities. In brief, nodes of relatively high degree do not "fit" into the ideal communities previously described, but rather they will force larger and sparser communities. Similarly, nodes with relatively low degrees do not exactly "fit" into ideal communities; however, they would not be as influential in altering the average community size. For the above reasons, real networks often cannot be partitioned such that the partitionings yield modularity values near 1.

Modularity defines "good" community structure as a partitioning of communities such that few or no edges connect the communities, the communities have very similar degree-sums, and several communities exist. If such an ideal community structure could be seen, the modularity value could still increase if the graph density were to be decreased. However, the heterogeneous degree distributions of real-world networks create significant challenges in finding any such maximal community structure as defined by modularity.

III. Bloom

A. Agglomerative algorithms

Network partitioning has classically focused on defining communities through a variety of divisive methods such as minimum cuts or though aggregation methods that collect individual nodes into partitions [6][7]. One well known agglomerative method is based on modularity and is known as the fast-greedy algorithm [18][20]. This algorithm has been found to serve as a useful approach for computationally difficult community analysis. Although it admittedly does not usually return very near-optimal partitionings, it efficiently provides good approximations. The fast-greedy algorithm begins with a simple graph of N nodes divided into $c = N$ communities of 1 node each. A community-pairwise benefit matrix (it defines the change in the modularity value caused by merging a pair of communities) is then computed. The pair of communities that offers the maximum increase in modularity is merged and the benefit matrix is updated. The merging process repeats until no community pair offers an increase to the modularity value. The resulting partitioning is the solution of the fast-greedy method.
There is also another type of agglomerative method that *grows* communities [27][28][29]. Instead of choosing the best merge among all communities, these heuristics work on a expanding each local community up to a certain limit. Each has a different benefit metric and grows communities until the additional benefit disappears. In this paper, we focus on using the modularity function as the benefit objective. In the work of [30], they use a modularity based growth method and then add to it an agglomeration of the resulting communities and then further refinement. Their method can be as fast as $O\left(N^2\right)$ for sparse networks, but can have a slower worst case running time [30].

An ideal community should have boundaries that are reasonably identifiable. Therefore in growing a partition to cover a community, there should exist a natural local maximum (in the community quality metric) that is reached as the partition comes to cover all the nodes belonging to the community. Similarly, an ideal community should not be easily divisible and thus should not be significantly cut by a local maximum. Therefore in ideal communities, the local maximum is both an approximate upper bound and an approximate lower bound on the correct partition, and as such it would easily identify these communities. It is known, however, that the real world networks of interest to scientists are rarely composed of many poorly connected cliques. Still, this local maximum growth idea can be used to help capture even realistic community structures.

B. Local modularity benefits

From the community quality metric under consideration, modularity, the benefit or cost of selecting a node to add to your community can be quantified. In considering the growth of a first community, we examine the modularity of two communities.

We derive expressions to quantify the change to the modularity function as a node switches from one community to the other. An expression of modularity for a partitioning of two communities is

$$Q\left(P_2, G\right) = \frac{1}{4L}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(a_{ij} - \frac{d_i d_j}{2L}\right)s_i s_j, \tag{7}$$

where the community assignments are given by a vector s. Each of the N elements of s takes on a $+1$ or -1 to describe which community the node belongs to. Recalling the modularity matrix of [31], we can rewrite equation 7.

$$B_{ij} = a_{ij} - \frac{d_i d_j}{2L} \tag{8}$$

$$Q\left(P_2, G\right) = \frac{1}{4L}s^T B s = \frac{1}{4L}\sum_{i=1}^{N}\left[s_i\left(s^T b_i\right)\right], \tag{9}$$

where b_i is the i^{th} column (or equivalently, i^{th} transposed row as B is symmetric) of B. Similar to a Laplacian matrix, each row and column of B sums to 0. When a node i switches from one community to the second, the i^{th} element of s will be multiplied by -1. By expanding the expression, we derive the change in the modularity value that would come from node i switching from his community to the other as a function of the states s before the switch occurs.

$$\Delta Q_i\left(P_2, G\right) = \frac{1}{4L}\left[-2s_i\left(\sum_{j=1,\ j\neq i}^{N} s_j\left(B_{ij} + B_{ji}\right)\right)\right] = \frac{1}{4L}\left[-4s_i\left(\sum_{j=1,\ j\neq i}^{N} s_j B_{ij}\right)\right] \tag{10}$$

Rather than recomputing the $\Delta Q\left(P_2, G\right)$ vector each time a node i switches, $\Delta Q_i\left(P_2, G\right)$ can be negated and the remaining entries (each $\Delta Q_j\left(P_2, G\right)$) can be updated by adding $2s_i s_j B_{ij}/L$. In the case of the partitioning such that there exists only a single community and all nodes belong to it, all s_i would be of the same sign. In this case, we simplify equation 10 to

$$\Delta Q_i\left(P_2, G\right) = -d_i^2/\left(2L\right). \tag{11}$$

The initial cost of leaving the "great" community is function of node degree and does not depend on the network structure. Thus for modularity, the ease of initial defection is determined solely by the node's connections or degree. Another equivalent expression,

$$Q\left(P, G\right) = \sum_{t=1}^{c}\left(\frac{L_t}{L} - \left(\frac{D_{X_t}}{2L}\right)^2\right) = \sum_{t=1}^{c} Q\left(X_t, G\right), \tag{12}$$

describes modularity as a function of community variables, where L_t is the sum of edges that connect pairs of nodes within community X_t and D_{X_t} is the degree-sum of community X_t. Due to the contributions to the modularity value being separable (as in equation 12) and the negative contribution of a single node community shown in equation 11, a node will not form a community by himself, thus a node with a degree equal to one will only follow his neighbor to whichever community the neighbor chooses.

Let us consider the general case when there are more than two communities. According to the original expression of modularity in equation 12, the contributions of each community can be separated [5]. Considering a node p that is a member of community X_r, if node p were to change his allegiance from X_r to X_s, then $\Delta Q\left(P, G\right) = \Delta Q\left(X_r, G\right) + \Delta Q\left(X_s, G\right) + \sum_{t=1, t\neq r,s}^{c}\Delta Q\left(X_t, G\right)$. The only changes in the terms of the sum in equation 12 are in the terms of communities X_r and X_s, thus $\Delta Q\left(P, G\right) = \Delta Q\left(X_r, G\right) + \Delta Q\left(X_s, G\right)$. We derive the changes to the community left (X_r) and the community arrived at (X_s).

$$\Delta Q\left(X_r, G\right) = -\frac{1}{L}\left[d_p^{(r)} - \frac{d_p\left(D_{X_r} - d_p/2\right)}{2L}\right],$$

$$\Delta Q\left(X_s, G\right) = \frac{1}{L}\left[d_p^{(s)} - \frac{d_p\left(D_{X_s} + d_p/2\right)}{2L}\right],$$

where the degree of node p that is within community X_r (X_s) is given by $d_p^{(r)}$ ($d_p^{(s)}$). For an increase in modularity, $\Delta Q\left(P, G\right) > 0$, we find the inequality

$$d_p^{(s)} - \frac{d_p\left(D_{X_s} + d_p/2\right)}{2L} > d_p^{(r)} - \frac{d_p\left(D_{X_r} - d_p/2\right)}{2L}. \tag{13}$$

Previous work has expressed these changes in a similar manner by separating the terms by their dependance on the characteristics of the other community X_s [30]. These are then referred to as forces that either hold node p in his community (X_r) or pull him out towards other communities.

$$F_{\text{in}}^{(r)} = d_p^{(r)} - \frac{d_p\left(D_{X_r} - d_p\right)}{2L} \tag{14}$$

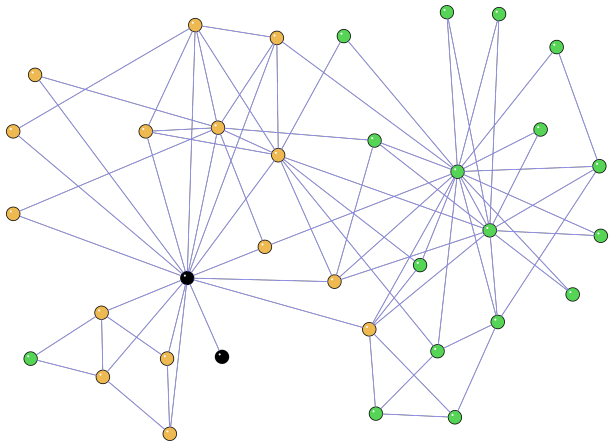

Fig. 1. A growth process of a first community in the Bloom algorithm on the Zachary Karate network [32]. The black nodes are currently in the community and considered covered. The golden nodes compose the current boundary set. The green nodes represent the remainder of the network. Two nodes have left the rest of the network and started the growth of the first community.

$$F_{\text{out}}^{(s)} = d_p^{(s)} - \frac{d_p D_{X_s}}{2L} \tag{15}$$

Intuitively, when for some community X_s, one finds that $F_{\text{out}}^{(s)} > F_{\text{in}}^{(r)}$, the move of node p from X_r to X_s modifies the partitioning P in a manner that increases $Q(P, G)$. Our fast method, described in the following subsection, uses the results of equations 10 and 11 for its basic improvement function and equation 15 to assign communities to nodes who are uncertain of where they belong. Equations 14 and 15 are also used for a refinement procedure in subsection IV-C.

C. A *fast growth algorithm: Bloom*

The algorithm Bloom is based on growing communities within a network using these local improvement measures. Bloom starts by considering the entire network to be one large community. Let us call it community X_0. In this situation, equation 11 describes the cost to the modularity value of a node i choosing to leave X_0. The first step is then to stochastically select one node as the seed node for the first community X_1. Once the seed node is chosen, we update the δQ vector as described in the discussion after equation 10. If, at this point, no other nodes wish to join the seed node in his new community, we return the seed node and pick another seed. After a node has been a seed node, it will be considered "tried." New seeds are only drawn from the "untried" nodes. If other nodes see a benefit to joining the seed node ($\Delta Q_i(P_2, G) > 0$), the node with the largest benefit is added to X_1. The δQ vector is updated and the next node is added repeatedly until the growing community reaches a local maximum, as given by $\Delta Q_i(P_2, G) \leq 0$ for all i. The addition of a node (with a non-zero node degree) to a community with which it has no connecting edge will likely decrease the modularity value. So rather than considering all nodes as potential additions to the growing community X_1, we maintain a set of "boundary" nodes, those connected to the community by at least one edge. Once X_1 has grown to its maximum, we plant a new seed stochastically in the remaining nodes of X_0. This seed is allowed to grow as if from the entire network, again guided by equation 11 and the updates after equation 10, but it is not allowed to add any nodes to its community that have already been covered by the previous community. This restriction on the candidate nodes reduces the current boundary to include only the nodes which are adjacent to the growing community and also belong to X_0. Bloom continues to plant new seeds and grow communities until either the entire network is covered or every node has been "tried" as a seed node. In the second situation, the remaining nodes in X_0 are iteratively added to their surrounding communities guided by equation 15. The final result is that community X_0 is empty. The final partitioning P is then taken as the resulting grown communities, $X_1, \ldots, X_c$. This process is outlined in algorithm 1.

An interesting characterisitic of Bloom is that it grows one community at a time. Figures 1 and 2 depict the growth process of a first community in the Zachary karate network [32]. The process of selecting the next node to add from the boundary is a search over the nodes in the boundary to see which offers the most benefit to the modularity value. For each node added to a community, his neighbors (that are not yet in the boundary) are added to the boundary. Thus each growth step of adding node i to a community takes at most $O(d_i + |boundary|)$, where d_i is the node degree of node i and $|boundary|$ is the current size of the community boundary set of nodes. Here, boundary is defined as in algorithm 1 and does not refer to the full set of nodes adjacent to the currently growing community, but rather the boundary is a subset of the adjacent nodes, being those which are currently uncovered. Therefore, an average step of the algorithm takes $O(\langle d \rangle + \langle |boundary| \rangle)$, and as there are c communities with an average of n_c nodes in each, the average running time scales as $O(cn_c(\langle d \rangle + \langle |boundary| \rangle))$. Since $cn_c = N$, this can be written as $O(N(\langle d \rangle + \langle |boundary| \rangle))$. To simplify this for comparison to other methods, we note that at any time the boundary of a community should intuitively be smaller than the fully grown community as each node in the

Algorithm 1 The Basic Bloom Algorithm for Maximizing Modularity

Given: A simple graph G with N nodes, L edges
Initialize: $c \leftarrow$ The number of communities set to 0
$P_{\text{init}} \leftarrow$ A partitioning of 1 community with N nodes
$\delta Q_{\text{init}} \leftarrow$ Benefit vector of equation 11
$tried \leftarrow \emptyset$, $untried \leftarrow$ nodes of G
$covered \leftarrow \emptyset$, $uncovered \leftarrow$ nodes of G
$community \leftarrow N \times 1$ integer community labels set to 0
while $|untried| > 0$ **do**
  $P \leftarrow P_{\text{init}}$
  $\delta Q \leftarrow \delta Q_{\text{init}}$
  $boundary \leftarrow \emptyset$
  Pick a $seed$ node pseudo-randomly such that $seed \in untried$ AND $seed \in uncovered$
  $community\,[seed] \leftarrow c + 1$
  Copy any $uncovered$ neighbors of $seed$ to $boundary$
  Update P, δQ
  if there are any nodes within $boundary$ that have positive δQ_i **then**
    $c \leftarrow c + 1$
    $covered \leftarrow covered \cup \{seed\}$, $uncovered \leftarrow uncovered \setminus \{seed\}$
    while δQ contains positive entries for nodes within $boundary$ **do**
      //Grow community until it stops
      Node $p \leftarrow$ the node $p \in boundary$ such that δQ_p is maximal
      Copy any $uncovered$ neighbors of p to $boundary$
      $boundary \leftarrow boundary \setminus \{p\}$
      $covered \leftarrow covered \cup \{p\}$, $uncovered \leftarrow uncovered \setminus \{p\}$
      $tried \leftarrow tried \cup \{p\}$, $untried \leftarrow untried \setminus \{p\}$
      $community\,[p] \leftarrow c$
      Update P, δQ
    end while
  else
    $boundary \leftarrow \emptyset$
    $community\,[seed] \leftarrow 0$
  end if
  $tried \leftarrow tried \cup \{seed\}$, $untried \leftarrow untried \setminus \{seed\}$
end while
while $|uncovered| > 0$ **do**
  for each node $p \in uncovered$ **do**
    if p has any neighbors within $covered$ **then**
      Calculate $F_{out}^{(s)}$ for p from each $covered$ neighbor's community X_s
      $F_{out} \leftarrow$ maximal $F_{out}^{(s)}$ for node p with community $c_{F_{out}}$
      $covered \leftarrow covered \cup \{p\}$, $uncovered \leftarrow uncovered \setminus \{p\}$
      $community\,[p] \leftarrow c_{F_{out}}$
    end if
  end for
end while
Update P according to $community$
return $P,\ Q\,(P, G)$

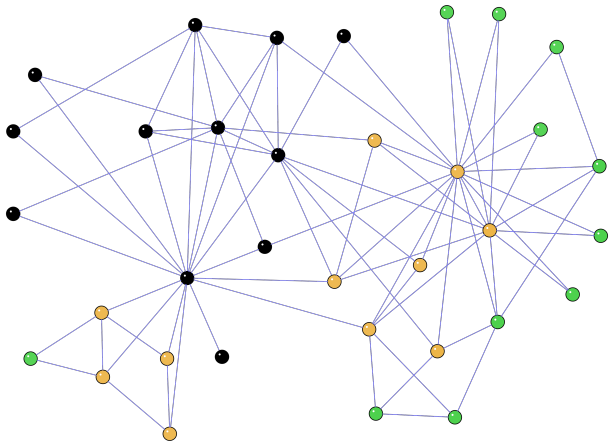

Fig. 2. A growth process of a first community in the Bloom algorithm on the Zachary Karate network [32]. The first community has reached a local maximum and no longer sees any beneficial node to add from the boundary set. Thus the black nodes will be considered the first community and next a seed will be planted (within the the uncovered nodes, golden and green) to start the second growth process.

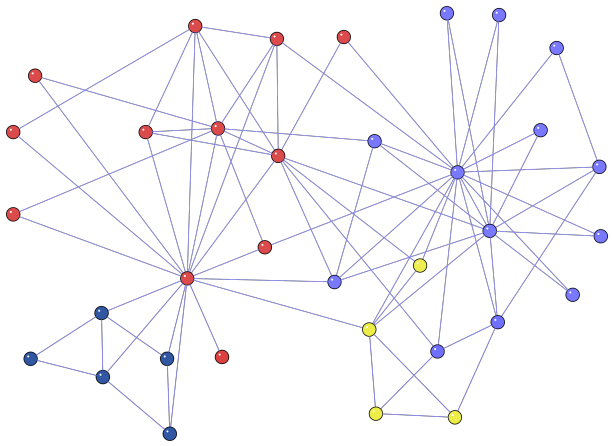

Fig. 3. A final state of the basic Bloom algorithm on the Zachary Karate network [32]. The first community is shown in red and the others in light blue, yellow, and dark blue.

boundary requires at least one edge exiting the community. To have a community with a boundary set of nodes that is as large as the community itself would suggest that the community should be further expanded (or grown). This partially-expanded set of nodes would represent a poor choice of a community [33]. In either case, the size of the community and the size of the boundary are both bounded very loosely above by N. Thus we can describe the worst case running time as $O\left(N^2\right)$. Figure 3 shows the final result of the Bloom process shown in figures 1 and 2. We found the optimal partitioning of the Karate network with $Q = 0.4198$ using the formulation given in [19]. This solution from Bloom is not the optimal partitioning, but it comes comparably close with $Q = 0.4156$, even with 3 nodes misclassified.

IV. Analysis

A. Networks

Before we decribe the analysis, we introduce the networks studied. We collected the following networks and are using them as simple graphs, without weights on the edges, directed edges, self-loops, nor multiple edges between pairs of nodes. The network of the Zachary Karate club [32] needs little introduction to the modularity community and had been depicted in the previous section. A list of 105 books on US politics sold on Amazon.com during the 2004 US presidential elections was collected and a network was built from them using copurchasing information offered on the website. We refer to this network by the name of the author, Krebs [34]. The network we call Jazz, is a collection of Jazz musicians and the interactions between them [35]. We also include the topology of the US powergrid as built in [36], a sample Autonomous System map of the Internet from CAIDA from March 2007 [37], and the largest component of the network of coauthorships among scientists who posted papers on the Condensed Matter E-Print Archive between Jan 1, 1995 and March 31, 2005 [38]. Figure 4 shows the topology of the US powergrid in a suspended visualization that does not depict the actual spatial locations of nodes. A summary of the network names and basic properties is shown in table I.

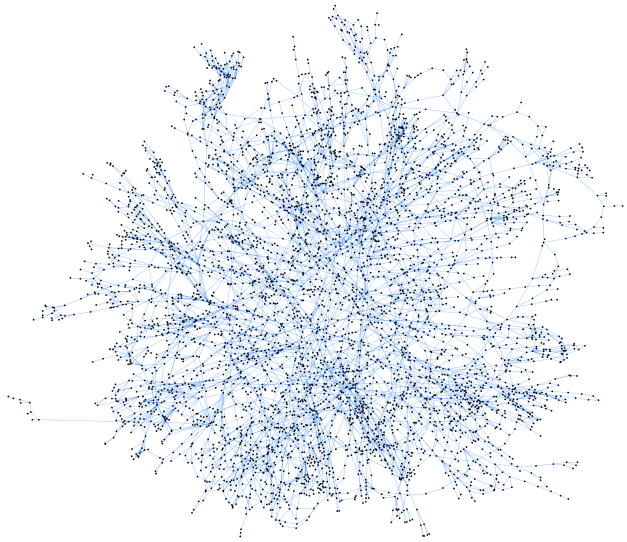

Fig. 4. The topology of the Powergrid network from [36]. Node positions do not represent physical locations.

TABLE I
SIX REAL-WORLD NETWORKS WITH THEIR RESPECTIVE NUMBERS OF NODES AND EDGES.

Network	Nodes	Edges
Karate	34	78
Krebs	105	441
Jazz	198	2742
Powergrid	4941	6594
CAIDA	24491	49826
CM05	36458	171736

In addition to these networks, we generated benchmark networks with designed community structure from the work of [39]. The networks are characterized by a set of design parameters, including, p, the mixing parameter, the size and density of the network, and the exponents of the degree distribution and the community size distribution. The distributions are motivated by measurements on real world networks. In the context of section II, it is interesting to note that the heterogeneity of the community sizes, although size is not directly related to the degree-sum, seems to conflict with the desire of modularity to capture balanced (by community degree-sum) communities. For the networks we generated, we varied the mixing parameter from 0.01 to 1.00 and set the distribution exponents at their suggested defaults of 1 and 2 for the communities and degrees, repsectively. The number of nodes we set to 1000 and the average degrees to 25. The average degree is not strictly constrained in the generation process. The generator also outputs the designed community structure.

B. Bloom and fast-greedy

The fast-greedy and Bloom algorithms are compared across the six real-world networks listed in table I. As Bloom is a stochastic method, we conduct 1000 partitionings on each network and report the sample average number of communities found ($\langle c_{\mathrm{b}} \rangle$) and the average ($\langle Q_{\mathrm{b}} \rangle$), minimum (min Q_{b}), maximum (max Q_{b}), and standard deviation ($\sigma\left(Q_{\mathrm{b}}\right)$) of the modularity values found. For the fast-greedy algorithm, we list the number of communities found (c_{fg}) and the respective modularity values (Q_{fg}) in table II. For the real world networks, it can be seen that the basic Bloom method typically performs worse than the deterministic fast-greedy algorithm. Bloom finds especially poor modularity values for the powergrid and CAIDA networks. For some networks, the simple Bloom method brings reasonably close values to the modularity values of the fast-greedy partitionings in table II. We notice that the discrepancies in the modularity values seem to be related to the differences in the number of communities found by each method. However, this observation does not hold for the generated networks.

While Bloom does not typically provide the best performance in modularity, Bloom can be used efficiently for another purpose. Bloom doesn't deterministically output only one partitioning, but stochastically offers a set of solutions with repeated applications to a network. Although this diversity may not be desired by all network partitioners, it allows a user to explore a set of approximations of the optimal partitioning. The degeneracy of the modularity function has been described in [14], where the authors demonstrate that many near optimal solutions may exist for a network. The diversity of the modularity values found by Bloom can be partially captured through the $\sigma\left(Q_{\mathrm{b}}\right)$ values for each network as seen in table II. The distributions

TABLE II
A COMPARISON OF THE PARTITIONING RESULTS OF THE FAST-GREEDY (fg) ALGORITHM OF [20] AND THE BASIC VERSION OF BLOOM (b) ON SIX REAL WORLD NETWORKS.

Network	Q_{fg}	c_{fg}	max Q_{b}	$\langle Q_{\text{b}} \rangle$	min Q_{b}	$\sigma\,(Q_{\text{b}})$	$\langle c_{\text{b}} \rangle$
Karate	0.3807	2	0.4156	0.3989	0.3715	0.0153	3.262
Krebs	0.5020	3	0.5017	0.4860	0.4180	0.0210	2.867
Jazz	0.4389	3	0.4357	0.4235	0.3832	0.0116	3.000
Powergrid	0.9350	39	0.7987	0.7653	0.7267	0.0138	22.388
CAIDA	0.6516	70	0.5984	0.5772	0.5244	0.0111	47.241
CM05	0.6128	532	0.5874	0.5651	0.5305	0.0118	550.654

of Q_b can help characterize the partitioning results. Figures 5, 6, and 7 display the sample distributions of Q_b for the Karate, Krebs, and Jazz networks, respectively. Note that, even though the range of Q_b is different in each of these three figures, the plotted width of the range of Q_b is identical in each figure for comparison. When a network demonstrates a lot of diversity in partitioning solutions, this implies that the communities are not well defined, and Bloom allows a user to see this. It is important to remember that the set of possible partitioning solutions explodes as the number of nodes increases. So the diversity of a set of solutions from a stochastic partitioning method will increase with both the partitioning difficulty and the network size. The influence of this second element would be reduced as more solutions are sampled and the distribution concentrates on a set of preferred local maximums of the modularity value.

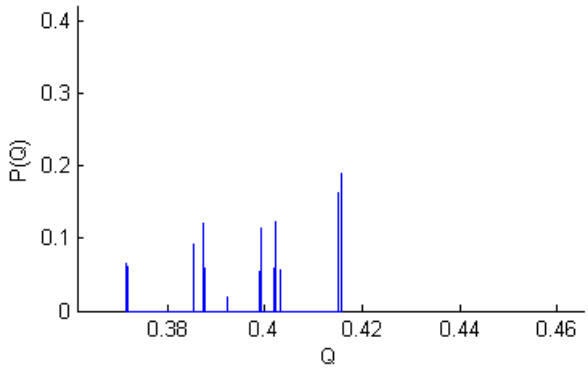

Fig. 5. The distribution of the modularity values Q_b found by the basic Bloom method for 1000 runs on the Zachary Karate network [32].

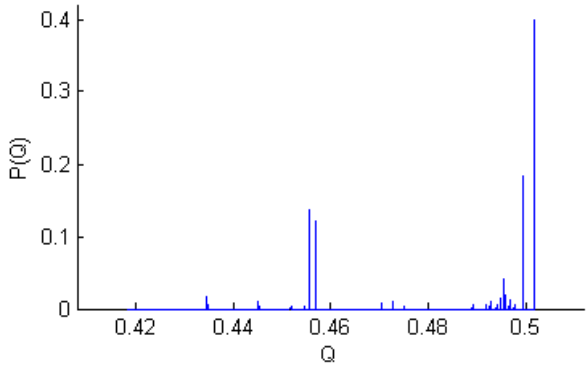

Fig. 6. The distribution of the modularity values Q_b found by the basic Bloom method for 1000 runs on the Krebs network [34].

We repeated the comparison of Bloom and the fast-greedy on the set of generated networks. Figure 8 plots the resulting modularity values. With the designed community partitions of these networks, we are able to compare Bloom and the fast-greedy method with an additional measure known as the normalized mutual information (NMI) [9][39]. Based on information theory, this metric captures the quality of a match between two partitioning solutions. A perfect matching of node community assignments corresponds to an NMI value of 1. Poor matchings correspond to positive values closer to 0. With the designed communities as the reference, figure 9 plots the resulting NMI values versus the mixing parameter p. Across the set of networks, the fast-greedy algorithm produces higher modularity values than Bloom and mostly higher NMI values. That the fast-greedy method usually will slightly outperform Bloom is consistent with our observations on the real-world networks. The numbers of

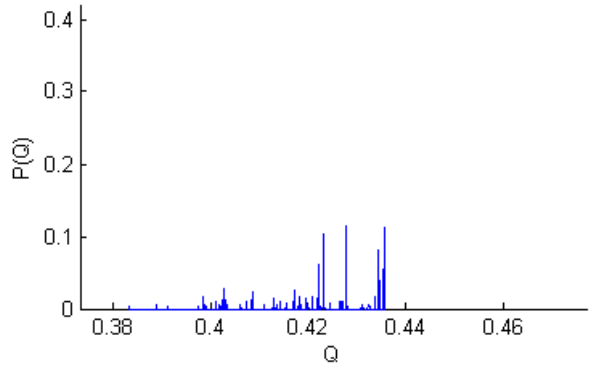

Fig. 7. The distribution of the modularity values Q_b found by the basic Bloom method for 1000 runs on the Jazz network [35].

communities found by each method are plotted along with the designed numbers for each network in figure 10. It is interesting to observe that, for the small increment in modularity that the fast-greedy provides over Bloom, the fast-greedy has often twice the number of communities. This would imply that the communities found by Bloom have a higher average modularity value and significance as compared to the fast-greedy method. From a user's point of view, this means that Bloom's poorer "performance" may come with more significant or robust communities.

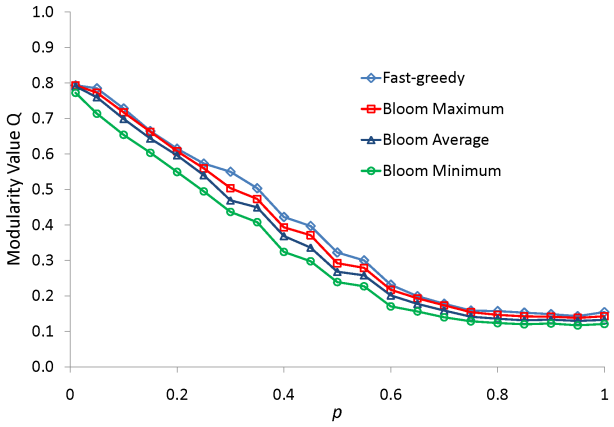

Fig. 8. A comparison of modularity values between the fast-greedy method and the basic Bloom method across 21 generated networks with synthetic community structure. The mixing parameter p controls the probabilities of edges being generated within communities versus among different communities. The 21 benchmark networks are generated with inputs of the number of nodes (1000), the average degree (25), the node degree distribution exponent (2), the community size distribution exponent (1), and the mixing parameter (p) [39].

One aspect of modularity that has been discussed a lot recently is the resolution of the function. We compared the fast-greedy algorithm with Bloom on three of the resolution networks designed in [11], two of which are beyond the resolution threshold and one which is just below the threshold. The F3A network is a ring of 30 cliques, each of 5 nodes, with each pair connected by a single edge for a total of 330 edges. Being beyond the resolution threshold, this network has a natural solution of each clique being a community and an optimal modularity solution of each community being a pair of cliques. Also beyond the resolution threshold, the F3B network is composed of four cliques connected by single edges to create a triangle with a leaf. The leaf clique and associated clique on the triangle each have 20 nodes and the remaining two have 5 nodes each. The optimal modularity partitioning for this network will combine the two smaller cliques into one community. The third resolution network, F3C, has the same structure as F3B, but with 10 nodes in each of the smaller cliques. The optimal solution of this network is the natural solution of assigning each clique to a community. More details on these networks, as well as visualizations, can be found in [11]. The results of the resolution comparison are summarized in table III, and table IV lists the optimal and natural equivalents for the three networks.

Table III shows that the fast-greedy algorithm performs near optimally in finding 16 communities, close to the 15 of the optimal solution. Bloom finds an average of close to 6 communities, which shows a tendency to over-aggregate in this first resolution network. Although neither method succeeds in identifying the natural communities in the F3A network, we notice that Bloom finds an average of 6 communities with an average Q_b of 0.797 which means that the average modularity of the communities found is roughly 0.13, while for the fast-greedy it is 0.055. Recall that this is only the average modularity per

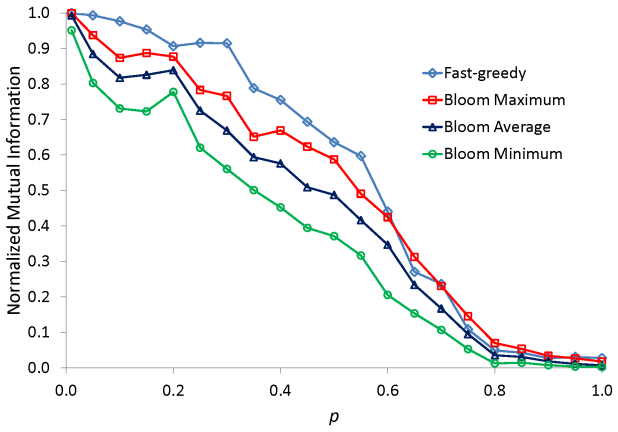

Fig. 9. A comparison of Normalized Mutual Information values between the fast-greedy method and the basic Bloom method across 21 generated networks with synthetic community structure [9]. The mixing parameter p controls the probabilities of edges being generated within communities versus among different communities. The 21 benchmark networks are generated with inputs of the number of nodes (1000), the average degree (25), the node degree distribution exponent (2), the community size distribution exponent (1), and the mixing parameter (p) [39].

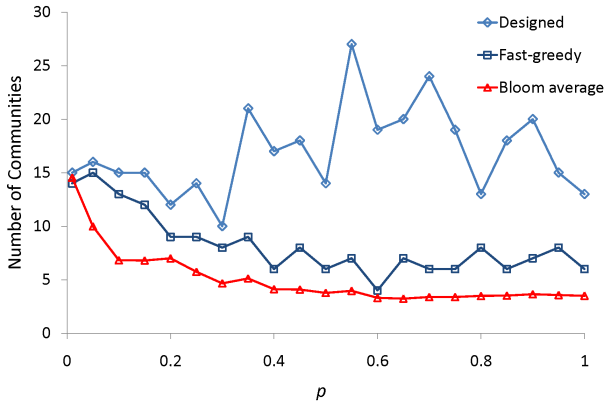

Fig. 10. A comparison of the number of communities found between the fast-greedy method and the basic Bloom method across 21 generated networks with synthetic community structure. The designed number of synthetic communities (chosen by the generation method) is also plotted for comparison. The mixing parameter p controls the probabilities of edges being generated within communities versus among different communities. The 21 benchmark networks are generated with inputs of the number of nodes (1000), the average degree (25), the node degree distribution exponent (2), the community size distribution exponent (1), and the mixing parameter (p) [39].

TABLE III
A COMPARISON OF THE PARTITIONING RESULTS OF THE FAST-GREEDY (fg) ALGORITHM OF [20] AND THE BASIC VERSION OF BLOOM (b) ON THREE RESOLUTION NETWORKS [11].

Network	Q_{fg}	c_{fg}	max Q_{b}	$\langle Q_{\text{b}} \rangle$	min Q_{b}	$\sigma\,(Q_{\text{b}})$	$\langle c_{\text{b}} \rangle$
F3A	0.8863	16	0.8085	0.7971	0.7848	0.0065	6.131
F3B	0.5426	2	0.5426	0.5393	0.4951	0.0120	3.000
F3C	0.6480	3	0.6480	0.6463	0.5974	0.0067	4.000

TABLE IV
THE OPTIMAL AND NATURAL PARTITIONING SOLUTIONS AS DESCRIBED BY THE MODULARITY VALUES AND NUMBERS OF COMMUNITIES [11].

Network	Q_{opt}	c_{optimal}	Q_{nat}	c_{natural}
F3A	0.888	15	0.876	30
F3B	0.5426	3	0.5416	4
F3C	0.6480	4	0.6480	4

community provided by Bloom, it still may provide solutions very similar to the fast-greedy method. The user then can explore

these solutions to see a range of possible partitionings with various levels of community significance. For the F3B and F3C networks, Bloom typically finds the optimal number of communities while the fast-greedy method finds one less. In general, these algorithms both focus on efficiency rather than precision, yet the question of resolution is an interesting point for further exploration.

C. Bloom Modifications

We considered how minor modifications to the basic Bloom algorithm impact its partitioning abilities in comparison to other well-known methods. We compared the basic version to the spectral bipartitioning of [31], a random walk method [40], the fast-greedy algorithm [18][20], and three modified versions of Bloom. The bipartitioning method uses the first eigenvector of the modularity matrix B to sort the nodes into two groups and then refines them with a method similar to the one found in [41]. The random walk method is called Walktrap and uses the idea that a typical random movement is likely to be contained within a community to estimate good communities. The Walktrap method was set to a four step approximation of the random walk. The first of the modified versions of Bloom, called Bloom1, is the basic version of Bloom plus a refinement of the partitions. The refinement allows single nodes to switch between communities guided by equations 14 and 15 in a manner similar to the method used in [30]. This extra refinement, which is appended to the end of the basic Bloom method to make Bloom1, takes time $O(N)$. Therefore the running time of Bloom1 is at worst $O\left(N^2+N\right)=O\left(N^2\right)$. The second modified version of Bloom, Bloom2, takes the partitions of the basic version of Bloom and sorts them into two groups to create an initial bipartition. Next it refines the bipartition using the same refinement as the spectral bipartitioning of [31]. The only difference between the spectral bipartitioning method and Bloom2 is the method of deterimining the initial bipartition, yet even this makes a difference. Since the refinement takes $O\left(N^2 log\left(N\right)\right)$, the running time of Bloom2 becomes $O\left(N^2 log\left(N\right)\right)$ [41]. The third extension of basic Bloom is called Bloom3 and uses a multipartitioning approach. It starts with running Bloom1 and then for each of the resulting communities it again applies Bloom1 on the community as a subgraph. This iterative multipartitioning continues until every resulting community is no longer partitionable. To explore the complexity of Bloom3, consider that the initial partition takes $O\left(N^2\right)$ and produces c_0 communities with an average of n_{c_0} nodes in each. Applying Bloom1 to each of these resulting communities takes an additional $O\left(c_0 n_{c_0}^2\right)$. From these $c_0=c_0^-+c_0^+$ partitionings, c_0^- communities with an average size of $n_{c_0^-}$ are not partitionable by Bloom1, and c_0^+ with average size $n_{c_0^+}$ are. Defining c_1 and n_{c_1} as the resulting number of communities and average size from the c_0^+ communities, we can interate to say that the running time of Bloom3 can be described as $O\left(N^2+c_0 n_{c_0}^2+c_1 n_{c_1}^2+c_2 n_{c_2}^2+\ldots\right)=O\left(N^2+N_0 n_{c_0}+N_1 n_{c_1}+N_2 n_{c_2}+\ldots\right)=O\left(N^2+N_0^2/c_0+N_1^2/c_1+N_2^2/c_2+\ldots\right)$, where N_i is the number of nodes that compose the c_i communities which remain to be partitioned again after the i^{th} partitioning. Note that $N_0=N$ and that $N_{i+1}\leq N_i$. It has been our observation that these terms converge quickly to zero, and therefore we would argue that Bloom3 also typically takes $O\left(N^2\right)$. A modification similar to Bloom3 was suggested for the fast-greedy algorithm in [18]. The spectral bipartitioning method takes $O\left(N^2 log\left(N\right)\right)$, and the Walktrap method takes $O\left(N^2 log\left(N\right)\right)$ for sparse networks [31][40]. It is important to consider that modern computational abilities allow for significant parallelization. For most of these methods, a factor of N can be removed through skillful computation, making them all linear or nearly linear in scaling [42][43][44]. These 7 algorithms are compared across the first 3 networks listed in table I and the results are summarized in table V.

TABLE V
A COMPARISON OF THE PARTITIONING RESULTS OF THE BASIC VERSION OF BLOOM (b), FAST-GREEDY (fg) [20], WALKTRAP (rw) [40], SPECTRAL BIPARTITIONING (sb) [31], AND THREE MODIFIED VERSIONS OF BLOOM [WITH REFINEMENT ($b1$), BIPARTITIONING ($b2$), AND MULTIPARTITIONING ($b3$)] ON THREE REAL WORLD NETWORKS.

Method		Karate	Krebs	Jazz
Bloom	$\langle Q_{\mathrm{b}}\rangle$	0.3989	0.4860	0.4235
	max Q_{b}	0.4156	0.5017	0.4357
Fast-greedy	Q_{fg}	0.3807	0.5020	0.4389
Walktrap	Q_{rw}	0.3532	0.5070	0.4384
Spectral	Q_{sb}	0.4188	0.5266	0.4422
Bloom1	$\langle Q_{\mathrm{b1}}\rangle$	0.4026	0.4909	0.4443
	max Q_{b1}	0.4198	0.5123	0.4445
Bloom2	$\langle Q_{\mathrm{b2}}\rangle$	0.4188	0.5258	0.4342
	max Q_{b2}	0.4188	0.5270	0.4445
Bloom3	$\langle Q_{\mathrm{b3}}\rangle$	0.4056	0.5043	0.4443
	max Q_{b3}	0.4198	0.5266	0.4445

While exploring simple modifications to the basic Bloom algorithm, we see encouraging results. Bloom1 and Bloom3 find the optimal value for the Karate network, and Bloom2 finds the highest value for the Krebs network [19][2]. Although not shown

[2]This reference contains an error in the optimal value of Q for the Karate network as the network used in the paper is missing an edge that falls between two communities. See figure 5 of the referenced paper.

here, we noticed that the first stage of Bloom2, the sorting of the basic and unrefined Bloom communities into two groups, very often brought a better intitial bipartition than the equivalent stage in the spectral bipartitioning algorithm. After they both use the same refinement method, Bloom2 is still often ahead in the maximization of modularity. Therefore, the modification of fast algorithms can build more efficient and successful methods than the newer sophisticated methods. An example of this is the extended version of the fast-greedy algorithm in [45], where the benefits of this method over other algorithms are shown in [46]. Other clustering methods that are not modularity-based have also gone in the direction of modified basic ideas [47]. We demonstrate how the modified versions of Bloom shift the distributions of the modularity values for the Krebs network and Jazz network in figures 11 and 12, respectively.

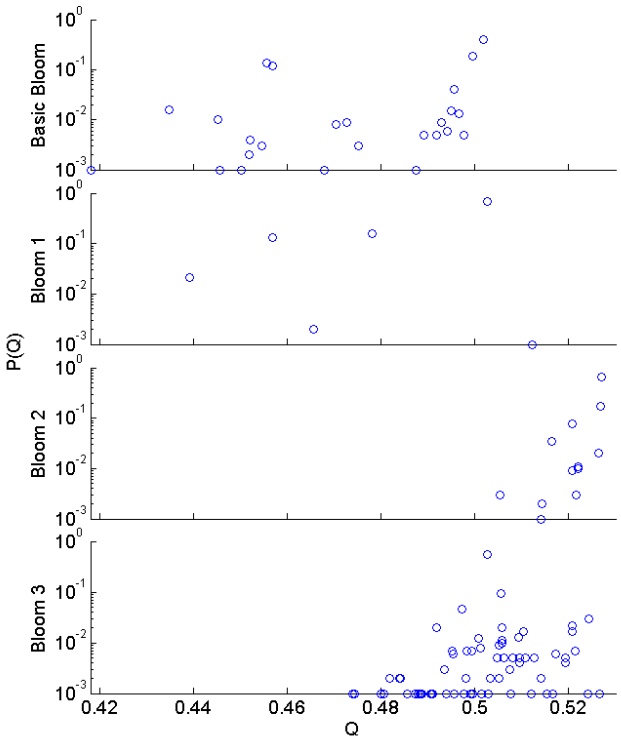

Fig. 11. The distribution of the modularity values Q_b found by the basic Bloom, Bloom1, Bloom2, and Bloom3 methods for 1000 runs each on the Krebs network [34].

It can be seen that Bloom2 is more successful in the Krebs network, but less in the Jazz network when compared to Bloom1 and Bloom3. As Bloom3 is an improvement on Bloom1, it outperforms Bloom1 in both cases. Observing how small the variance is for Bloom1 and Bloom3 on the Jazz network, we consider this to be an indicator of a decently well defined community structure among the higher modularity values. However, Bloom2 faces difficulty in finding this structure. This is likely due to the concept that forcing communities into bipartitions can sometimes significantly restrict the solution. It remains important to note that one algorithm will not be suitable for all types of networks. Due to differing network structures, different approaches can be more successful than others. We note a possible demonstration of this in Bloom2 finding the highest value for the Krebs network, while being outperformed on the Karate network. This also might be due to the more sophisticated refinement of Bloom2 in comparison to the simple refinement used in Bloom1 and Bloom3.

V. Conclusions

In this paper, first we have explored insights into the definition, as proposed by modularity, of a good community and proposed an efficient stochastic modularity algorithm, Bloom. In section II, we explain how the modularity of a network is constrained by both the density of the network and the degree distribution. Although we don't present a solution for optimally constructing a maximally modular graph from a degree sequence, we have demonstrated the difficulties that arise with heterogeneity. Hub nodes force the network's communities to be larger and sparser. In the context of human friendship networks, these difficulties do not imply that you cannot have a close-knit group of friends where one or two of the friends are notably popular. Rather it implies that such popularity will spill into other communities beyond your own. We also showed that two (relatively) very popular people will have a local discouragement from sharing a community, even if they are friends (connected), yet this can be overcome if they have enough common friends falling into the same community.

In the next part of the paper, we introduced Bloom, which provides stochastic approximations of the optimal partitioning of a network. The complexity of partitioning a network can be related to the diversity of a set of outputs from Bloom to describe

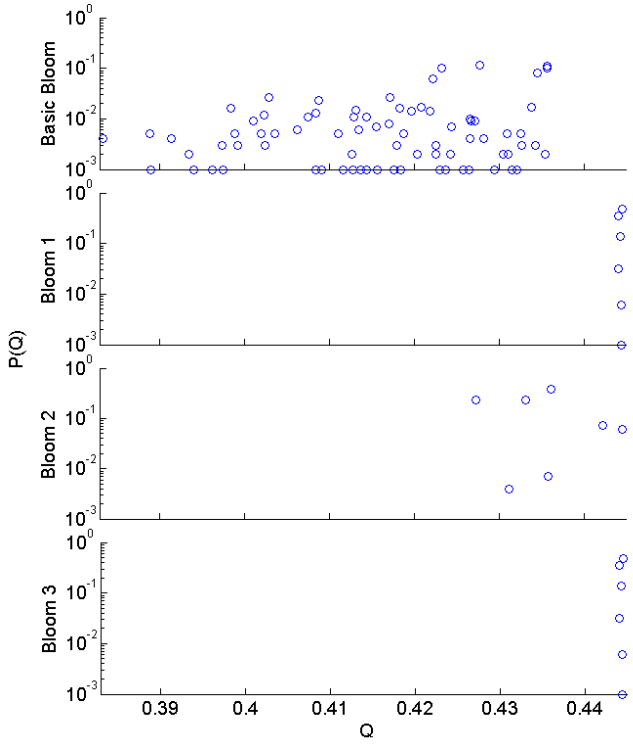

Fig. 12. The distribution of the modularity values Q_b found by the basic Bloom, Bloom1, Bloom2, and Bloom3 methods for 1000 runs each on the Jazz network [35].

how well or poorly the communities are defined. Bloom allows for the analysis of multiple near-optimal solutions, given its stochastic nature. In the presented experiments, Bloom is almost always outperformed by the fast-greedy algorithm of [20] with respect to the maximization of modularity. However, we show that simple modifications of Bloom can produce higher results than even the sophisticated spectral bi-partitioning method of [31]. From our analysis, we saw that different networks may be better matched with different partitioning methods, even if the basic idea behind the methods is the same. In our work, this basic idea was growing communities. This analysis would suggest further explorations into modifications of other efficient methods.

VI. Acknowledgements

This work was supported by the National Agricultural Biosecurity Center (NABC) at Kansas State University. We would like to thank S. Pahwa for providing figure 4.

References

[1] M. E. J. Newman, *Networks, An Introduction*, Oxford University Press, 2010.
[2] A. Barrat, M. Barthelemy, and A. Vespignani, *Dynamical Processes on Complex Networks*, Cambridge University Press, Cambridge, U.K., 2008.
[3] P. Van Mieghem, *Graph Spectra for Complex Networks*, Cambridge University Press, Cambridge, U.K., 2010.
[4] M. Youssef, R. Kooij, and C. Scoglio, "Viral Conductance: Quantifying the robustness of networks with respect to spread of epidemics," *Journal of Computational Science*, vol. 2, no. 3, pp. 286–298, Aug. 2011.
[5] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, Feb. 2004.
[6] I. A. Kovacs, R. Palotai, M. S. Szalay, and P. Csermely, "Community landscapes: an integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics," *Public Library of Science ONE*, vol. 5, no. 9, p. e12528, 2010.
[7] S. Fortunato, "Community detection in graphs," *Physical Reports*, vol. 486, pp. 75–174, Feb. 2010.
[8] R. Guimer, M. Sales-Pardo, and L. A. Nunes Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical Review E*, vol. 70, no. 2, pp. 025101, Aug. 2004.
[9] L. Danon, A. Daz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 9, p. P09008, Sep. 2005.
[10] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, p. 016110, Jul. 2006.
[11] S. Fortunato and M. Barthlemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences USA*, vol. 104, no. 1, pp. 36-41, Jan. 2007.
[12] J. M. Kumpula, J. Saramki, K. Kaski, and J. Kertsz, "Limited resolution in complex network community detection with Potts model approach," *The European Physical Journal B*, vol. 56, no. 1, pp. 41-45, Mar. 2007.
[13] A. Arenas, A. Fernandez, and S. Gomez, "Analysis of the structure of complex networks at different resolution levels," *New Journal of Physics*, vol. 10, no. 5, p. 053039, May 2008.
[14] B. H. Good, Y. A. de Montjoye, and A. Clauset, "The performance of modularity maximization in practical contexts," *Physical Review E*, vol. 81, no. 4, p. 046106, Apr. 2010.

[15] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29-123, 2009.

[16] A. S. Brahim, B. Le Grand, L. Tabourier, and M. Latapy, "Citations among blogs in a hierarchy of communities: Method and case study," *Journal of Computational Science*, vol. 2, no. 3, pp. 247–252, Aug. 2011.

[17] S. Fouchal, M. Ahat, S. Ben Amor, I. Lavallee, and M. Bui, "Competitive clustering algorithms based on ultrametric properties," *Journal of Computational Science*, In Press, Jan. 2012.

[18] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, Jun. 2004.

[19] U. Brandes et al., "On Modularity Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 172-188, Feb. 2008.

[20] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, Dec. 2004.

[21] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, Sep. 2007.

[22] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences USA*, vol. 105, no. 4, pp. 1118-1123, Jan. 2008.

[23] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13-23, Apr. 2010.

[24] P. Van Mieghem, X. Ge, P. Schumm, S. Trajanovski, and H. Wang, "Spectral graph analysis of modularity and assortativity," *Physical Review E*, vol. 82, no. 5, p. 056113, Nov. 2010.

[25] S. Lehmann and L. K. Hansen, "Deterministic modularity optimization," *The European Physical Journal B*, vol. 60, no. 1, pp. 83-88, Nov. 2007.

[26] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi, "The pursuit of hubbiness: Analysis of hubs in large multidimensional networks," *Journal of Computational Science*, vol. 2, no. 3, pp. 223–237, Aug. 2011.

[27] D. Chen, Y. Fu, and M. Shang, "A fast and efficient heuristic algorithm for detecting community structures in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 13, pp. 2741-2749, Jul. 2009.

[28] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, no. 2, p. 026132, Aug. 2005.

[29] A. Lancichinetti, S. Fortunato, and J. Kertsz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, p. 033015, Mar. 2009.

[30] Z. Ye, S. Hu, and J. Yu, "Adaptive clustering algorithm for community detection in complex networks," *Physical Review E*, vol. 78, no. 4, p. 046115, Oct. 2008.

[31] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences USA*, vol. 103, no. 23, pp. 8577-8582, May 2006.

[32] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, no. 33, pp. 452-473, 1977.

[33] M. W. Mahoney, "Algorithmic and statistical perspectives on large-scale data analysis," *To Appear in* Uwe Naumann and Olaf Schenk, editors, *Combinatorial Scientific Computing*, Chapman and Hall/CRC Press, 2012.

[34] V. Krebs, Thanks to Vladis Krebs for creating and sharing this network data. unpublished.

[35] P. Gleiser and L. Danon, "Community Structure in Jazz," *Advances in Complex Systems*, vol. 6, no. 4, pp. 565-573, 2003.

[36] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440-442, Jun. 1998.

[37] The Cooperative Association for Internet Data Analysis (CAIDA) AS Relationships Dataset <Mar. 12, 2007>, Available at http://www.caida.org/data/active/as-relationships/

[38] M. E. J. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences USA*, vol. 98, no. 2, pp. 404-409, Jan. 2001.

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

[40] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *in Computer and Information Sciences - ISCIS 2005*, vol. 3733, pInar Yolum, T. Gngr, F. Grgen, and C. zturan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 284-293.

[41] B. M. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, 49:291, 1970.

[42] T. Glavelis, N. Ploskas, and N. Samaras, "A computational evaluation of some free mathematical software for scientific computing," *Journal of Computational Science*, vol. 1, no. 3, pp. 150–158, Aug. 2010.

[43] M. Chorley and D. Walker, "Performance analysis od a hybrid MPI/OpenMP application on multi-core clusters," *Journal of Computational Science*, vol. 1, no. 3, pp. 168–174, Aug. 2010.

[44] C. Feichtinger, S. Donath, H. Kostler, J. Gotz, and U. Rude, "WaLBerla: HPC software design for computational engineering simulations," *Journal of Computational Science*, vol. 2, no. 2, pp. 105–112, May 2011.

[45] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, p. P10008, Oct. 2008.

[46] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, no. 5, p. 056117, Nov. 2009.

[47] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *Public Library of Science ONE*, vol. 6, no. 4, p. e18961, Apr. 2011.